Access 2007

Недостающее руководство

(конспект)

Об авторе

Мэтью Мак-Дональд (Matthew MacDonald) - автор и выдающийся программист.

Он автор "Excel 2007: The Missing Manual", "Creating Web Sites: The Missing Manual" и десятков книг о программировании с помощью Microsoft .NET Framework.

В почти забытой, прошлой жизни он изучал английскую литературу и теоретическую физику.

Благодарности

Написать книгу о такой объемной и сложной программе, как Access, можно только из любви (т.е. любви к страданию и боли). Я глубоко признателен множеству людей, включая тех, кто помог мне исследовать все искусно разработанные и изобретательные средства, вошедшие в последнюю версию пакета Office (включая выдающихся блоггеров Дженсена Харриса (Jensen Harris) и Эрика Ракера (Erik Rucker), тех, кто сделал эту книгу ясной, лаконичной и технически достоверной (Питер Мейерс (Peter Meyers), Сара Милштейн (Sarah Milstein), Брайан Сойер (Brian Sawyer), Джуел Бортолусси (Juel Bortolussi) и Майкл Шмалц (Michael Schmalz)), и тех, кто терпел, пока я писал ее (больше об этом - чуть позже). Я также благодарен всем, кто трудился над форматированием книги, созданием предметного указателя и ее изданием.

Написание книги потребовало нескольких бессонных ночей (и множества дней с недосыпом). Я выражаю любовь и признательность моей дочери Майе, которая выдержала большую часть времени без слез; моей дорогой жене Фарии (Faria), поступавшей в основном так же; и нашим мамам и папам (Hope (Nora), Разин (Razia), Полу (Paul) и Хамиду (Hamid)), нянчившим внучку, готовившим вкусную еду и помогавшим по дому, что позволяло не останавливать работу над книгой.

Большое спасибо вам всем - без вас половина книги все еще оставалась бы только у меня в голове!

Оглавление

Введение

В прошлом люди использовали различные технологии для организации информации. Они применяли вращающиеся картотеки барабанного типа компании Rolodex, перфокарты, картонные коробки, картотечные устройства вертикального хранения, каталоги с десятками тысяч страниц и (когда все оказалось тщетным) огромные стопки на плоских поверхностях. Но после многолетних страданий люди обнаружили, что для обработки информации гораздо удобнее использовать компьютеры, особенно если ее объем велик, структура сложна, а корректировки очень часты.

Именно тогда и пригодилась программа Access корпорации Microsoft. Access - это система управления базами данных - тщательно структурированные каталоги информации (или данных). Базы данных могут хранить почти любой тип информации, включая числа, страницы текста и изображения. У баз данных очень разные размеры - они могут обрабатывать все, начиная со списка семейных номеров телефонов и заканчивая огромным каталогом изделий для магазинчика тетушки Этель, торгующего пуговицами и кнопками со скидкой (Aunt Ethel's Discount Button Boutique).

В этой книге вы узнаете, как проектировать законченные базы данных, поддерживать их, находить значимую информацию и создавать привлекательные формы для быстрого и легкого ввода данных. Вы даже проникнете в черную магию программирования в Access и освоите важные приемы и методы, применяемые для автоматизации широко распространенных задач, даже если до этого вы не написали ни одной строки программного кода.

Главное достоинство книги состоит в том, что она с самого начала писалась для программы Access 2007, последней и величайшей реинкарнации самого популярного программного обеспечения для управления базами данных, созданного корпорацией Microsoft. Программа Access 2007 слегка отличается от предыдущих версий, благодаря яркому новому интерфейсу, вызвавшему оживленные толки в среде ярых компьютерных фанатов. Но в данном случае он не просто диковинка. Как вы убедитесь, освоив новый стиль Access, вы сможете создавать большие базы данных в рекордное время.

Какие задачи можно решать в программе Access

Современный мир наполнен информацией. Поиск в Web-пространстве скучного словосочетания "консервированная морковь" вылавливает более миллиона Web-страниц. В результате неудивительно, что людям разных профессий нужны мощные средства для хранения информации и управления ею.

Невозможно описать даже часть различных баз данных, создаваемых приверженцами Access каждый день.

Для того чтобы дать вам профессиональное представление о базах данных, далее перечислены распространенные типы информации, которые можно легко хранить в базе данных Access:

- каталоги книг, CD-дисков, редких марочных вин, рискованных фильмов или еще чего-то, что вам хотелось бы коллекционировать и за чем вы хотели бы следить;
- списки почтовой рассылки, позволяющие поддерживать связь с друзьями, семьей и коллегами;
- деловая информация, например, списки клиентов, каталоги изделий, записи заказов и счета;
- списки гостей и подарков для свадеб и других торжеств;
- перечни расходов, вкладов и других подробностей финансового планирования.

Воспринимайте программу Access как личного ассистента, который может помочь организовать, обновить и найти любой тип информации.

Эта помощь - не только удобство: она позволяет делать то, что вы никогда бы не сделали самостоятельно.

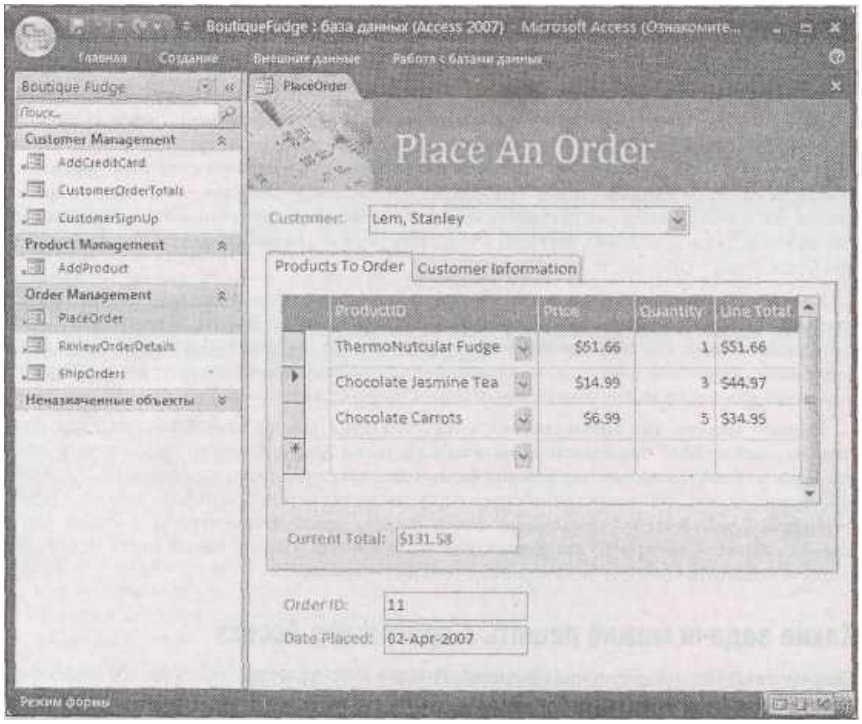


Рис. B1. Эта база данных о продажах содержит удобные формы, которые продавцы могут использовать для размещения новых заказов (показанных здесь), агенты по обслуживанию клиентов - для регистрации новых клиентов, а работники склада - для просмотра отправляемых заказов.

Самое главное в том, что люди, использующие формы в базе данных, могут ничего не знать об Access. После того как специалисты по базам данных (это и ваше будущее после прочтения книги) разработали данные формы, любой может пользоваться ими для ввода, редактирования и просмотра данных

Представьте, что вы только что закончили составление базы данных для вашей коллекции 10000 редких комиксов. По внезапному порыву вы решили взглянуть на все книги, написанные в 1987 г. или те, в которых изображен человек-амфибия (Aquaman), или же те, в заголовках которых есть слова "специальное издание". Поиск такого рода в бумажном каталоге занял бы несколько дней. На среднем компьютере Access может выполнить все три поиска меньше чем за секунду.

Программа Access также - король мелких предприятий из-за ее легендарной способности к самонастройке. В конце концов, на деле вы можете использовать любую программу управления базами данных для создания списка заказов клиентов. Но только Access облегчает создание полного пользовательского интерфейса для такой базы данных (как показано на рис. B1).

На профессиональном уровне.

Преимущества хорошо спроектированной базы данных

Множество людей применяет адресную книгу для связи с близкими друзьями, дальними родственниками и надоедливыми коллегами. По большей части простая адресная книга отлично справляется с задачей. Но подумайте, что произойдет, если вы решите хранить эту информацию в базе данных Access. Несмотря на то, что ваш список контактов не сравним с объемами информации, хранимыми компанией Google, он все равно приобретет новые функциональные возможности, не доступные без программы Access.

- ***Резервная копия.*** Если вы когда-нибудь пытались рассмотреть номер телефона сквозь кофейное пятно, то знаете, что порой выгодно хранить данные в электронной форме. После того как вы поместите контактную информацию в базу данных, вы сможете уберечь ее от стихийного бедствия и напечатать столько копий, сколько нужно (каждую с отображением частичной или полной информации). Вы даже можете поделиться этим списком с друзьями, которым нужны те же номера телефонов.
- ***Место.*** Несмотря на то, что большинству людей для хранения всех контактов достаточно маленькой адресной книжки, база данных гарантирует, что вы никогда не займете полностью раздел "М". Не говоря уж о том, что вы можете вычеркивать и записывать заново адрес

вашего странствующего дядюшки, пока не израсходуете всесвободное пространство в книжке.

- ***Поиск.*** В адресной книге контакты упорядочены одним способом - по имени и фамилии. Но что произойдет, если вы вносили всех, соблюдая алфавитный порядок для фамилий, а вам нужно найти контактную информацию человека, которого, кажется, зовут Джоу? Программа Access может без усилий выполнить такой поиск. Она также может найти записи по номеру телефона, что очень удобно, если ваш телефон сообщает перечень звонков и вы хотите выяснить, кто же это надоедает вам.
- ***Совместное использование.*** Только один человек в конкретный момент времени может редактировать самые обычные файлы, такие как документы программы Microsoft Word и электронные таблицы. Это ограничение создает проблемы, если всем сотрудникам офиса необходимо поработать над составлением обеденного меню.
  Программа Access позволяет одновременно многим людям просматривать и изменять данные на разных компьютерах. В *главе 18* приведена более подробная информация.
- ***Интеграция с другими приложениями.*** Access вводит вас в царство времясберегающих возможностей, таких как автоматическое составление стандартных писем (mail merge). Вы можете передать список контактов в форме письма, которое создаете в программе Word, и автоматически сгенерировать десятки писем с разными адресами. Вы увидите, как это делается, в *главе 19.*

Все эти примеры демонстрируют серьезные основания для перехода к электронному способу хранения информации любого типа.

Две стороны программы Access

Как вы увидите, с помощью программы Access решаются две разные задачи.

■ ***Проектирование вашей базы данных.*** В эту задачу входит создание таблиц для хранения данных, запросов, способных выискивать важные порции информации, форм, облегчающих ввод данных, и отчетов, формирующих привлекательные распечатки.

■ ***Обработка данных.*** Эта задача включает добавление новой информации в базу данных, обновление имеющейся или поиск необходимых подробностей. Для выполнения этой работы используются таблицы, запросы, формы и отчеты, которые вы уже сформировали.

Большая часть книги посвящена первой задаче - созданию и усовершенствованию вашей базы данных. Эта работа - суть программы Access и та часть, которая поначалу кажется самой страшной. Это именно то, что отличает знатоков Access от новообращенных.

После того как первая задача выполнена, вы можете переходить ко второй задаче - к активному использованию базы данных в повседневной жизни. Несмотря на то, что первая задача труднее, вы затратите больше времени (в конечном счете) на вторую задачу. Например, вы можете провести пару часов над созданием базы данных ваших любимых кулинарных рецептов, вводить же новую информацию и искать рецепты вы будете годами (скажем, каждый раз, когда вам нужно приготовить обед).

Access или Excel?

Программа Access - это не единственная составляющая пакета Office, способная обрабатывать списки и таблицы данных. В программу Microsoft Excel также включены функции для создания списков и управления ими.

Так в чем же разница?

Несмотря на то, что программа Excel отлично работает с малыми порциями простых данных, она не в состоянии обрабатывать информацию того объема и сложности, которая подвластна Access.

Excel плохо справляется с обработкой множественных списков со связанными данными (например, если вам нужно отслеживать список ваших деловых клиентов и список сделанных ими заказов). Программа Excel вынуждает полностью разделить эти списки, что затрудняет анализ ваших данных и создает возможность появления противоречивых результатов.

Access позволяет установить четкие связи между таблицами и, тем самым, устранить подобные проблемы.

Программа **Access также предоставляет полный набор функций**, не имеющих аналогов в мире электронных таблиц, таких как возможность создания настраиваемых процедур поиска, проектирование форм для ввода данных с тонкой настройкой и вывод на печатающее устройство ярко оформленных отчетов.

Конечно, все это не говорит о том, что Access лучше Excel.

В действительности вам захочется использовать обе эти программы.

Excel придает блеск бесконечным рядам чисел, создавая диаграммы, генерируя статистические характеристики или прогнозируемые тренды.

Во многих организациях программу Access применяют для хранения информации и управления ею, а затем экспортируют часть этой информации в электронную таблицу Excel для того, чтобы проанализировать ее. Вы узнаете, как это делается, в *главе 19.*

Access или SQL Server?

Корпорация Microsoft предлагает еще одну программу управления базами данных промышленного уровня - SQL Server, поддерживающую все, начиная от собственной поисковой машины корпорации Microsoft и кончая фондовой биржей NASDAQ. Ясно, что SQL Server - это промышленный магнат, и поклонники Access удивятся, как можно сравнивать с ним их любимую программу управления базой данных.

Одно из главных различий между Access и системами управления базами данных, подобными SQL Server, заключается в том, что Access - это программа, управляющая базой данных *на стороне клиента.* Если без технических терминов, это означает, что Access выполняется непосредственно на вашем персональном компьютере. Процессоры баз данных, такие как SQL Server, - это системы, использующие *сервер.* Они хранят данные на высокопроизводительном компьютере-сервере, к которому вы обращаетесь с обычного ПК. (Это взаимодействие происходит с помощью локальной сети.)

Базы данных на основе сервера (server-based databases) гораздо труднее создавать и сопровождать, но они обеспечивают более высокую производительность и нерушимы как скала, даже если одновременно их используют тысячи людей. Но высококлассные базы данных SQL Server нужны только крупным организациям. Amazon.com не продержался бы и пяти минут, если бы положился на базу данных Access. Тем не менее Access вполне подходит для большинства предприятий малого и среднего бизнеса. Программа идеальна для личного применения. (Если вы все еще сомневаетесь, подойдет ли вам Access, посмотрите далее *примечание "Для тех, кто понимает. Когда программы Access недостаточно* ".)

Другое важное различие между программой Access и серверными системами управления базами данных состоит в том, что Access предлагает единое решение для хранения и обработки данных. Серверные процессоры баз данных, такие как SQL Server, нацелены исключительно на хранение данных (и пересылку этих данных на другие компьютеры, когда они их запрашивают). За столь узконаправлеиный подход приходится платить. Обычный пользователь не может напрямую редактировать базу данных, сохраняемую SQL Server. Вместо этого вы вынуждены использовать еще одну программу, способную общаться с SQL Server и запрашивать нужную информацию. В большинстве случаев такая программа должна быть написана профессиональным программистом. Другими словами, если вы применяете SQL Server, вы должны написать целое приложение, прежде чем сможете эффективно использовать вашу базу данных.

Для тех, кто понимает.
Когда программы Access недостаточно

Если вы выбрали эту книгу, то, вероятно, считаете, что программа Access соответствует вашим потребностям. Если же есть какие-то сомнения, быстрая проверка в режиме реального времени покажет, на верном ли вы пути.

В следующем перечне описывается несколько предупреждающих признаков того, что программа Access не очень вам подходит. Если вы не попадаете в данные категории, поздравляю - вы готовы к использованию в любой области наиболее последовательного и эффективного программного обеспечения для управления базами данных!

• *Вы должны хранить огромные объемы информации (более 2 Гбайт данных).* Вы врядли превысите этот предел, если не будете хранить в базе данных больших изображений или цифрового контента других типов. У большинства баз данных Access размер - несколько мегабайтов (в 1000

раз меньше 2 Гбайт).

• *Вы собираетесь совместно использовать базу данных в сети, и десяткам людей придется обращаться к ней одновременно.* Это ограничение трудно корректно интерпретировать. Базу данных могут спокойно использовать время от времени сотни людей, но проблемы возникнут, когда группа людей настроена вносить изменения в один и тот же файл базы данных точно в один и тот же момент времени. Следует протестировать вашу базу данных, чтобы понять, сможете ли вы преодолеть это ограничение, не создавая проблем, В *главе 18* приведена дополнительная информация о совместном использовании программы Access группами пользователей.

• *Вы должны использовать вашу базу данных для запуска Web-приложения.* Web-приложение предоставляет большому числу людей возможность обращаться к базе данных одновременно. Программа Access может не справиться с нагрузкой. В этой ситуации вы выиграете от применения серверной базы данных, такой как SQL Server (и сильной команды программистов, которая выручит вас).

Иногда фанаты программы Access обращаются в гуру программы SQL Server. Можно начать с современной базы данных Access и затем перейти к SQL Server, когда ваши потребности превысят то, что может предложить программа Access. Этот процесс не всегда бывает гладким, но он возможен. Вы даже можете продолжать использовать Access как клиента для управления вашей базой данных, созданной SQL Server. Этот метод описан в *главе 20.*

Новый облик программы Access 2007

С тех самых нор, когда пакет Office корпорации Microsoft завоевал мир (в давних 1990 гг.), программы Word, Excel и Access не слишком изменились. Несмотря на то, что время от времени появляется действительно полезная новая функциональная возможность, Microsoft тратит больше времени на втискивание причудливых трюков вроде говорящей канцелярской скрепки.

Программа Access 2007 нарушает эту традицию и вводит ряд изменений, наиболее значительных из известных поклонникам программы со времен Office 95. Наиболее очевидное изменение - полностью исправленный пользовательский интерфейс (окна, панели инструментов, меню и ускоряющие сочетания клавиш, которые применяются для взаимодействия с программой Access). Потратив слишком много времени на упрощение бессистемного, засоренного панелями инструментов интерфейса, в конце концов, корпорация Microsoft набралась смелости и переработала его весь с нуля.

Лента

Лента программы Access 2007 - это инструментальная суперпанель, заменившая разнообразные панели инструментов и меню предыдущих версии.

Примечание

Программа не выводит на экран ленту до тех пор, пока вы не создадите базу данных. Если больше нет сил оставаться в неведенье и вы очень хотите увидеть ленту на вашем мониторе, выполните действия, описанные в *разд. "Создание новой базы данных" главы 1,* для создания пустой базы данных.

Лента разделена на зависящие от выполняемых задач вкладки - **Главная** (Home), Создание (Create), **Внешние данные** (External Data) и т. д. Запускается программа Access с выводом на экран четырех вкладок (другие вкладки появятся, когда вы начнете решать конкретные задачи). Программа начинает работу на вкладке **Главная** (Home). Щелкните кнопкой мыши вкладку **Создание** (Create) (как показано на рис. В2) и вы получите доступ к множеству мощных команд, позволяющих вставлять новые компоненты базы данных.

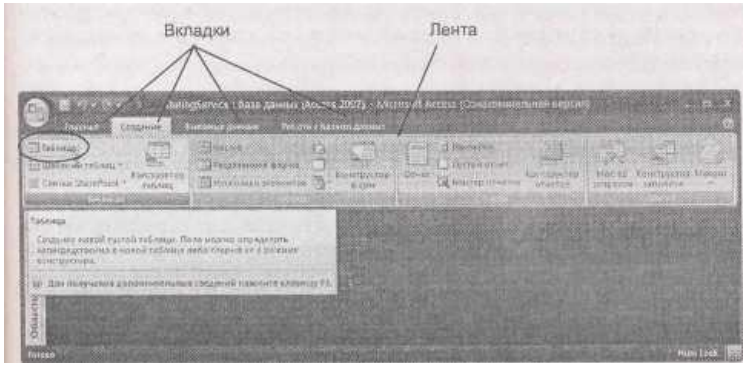


Рис. В2. Лента наполнена высокопрофессиональными средствами. Когда вы перемещаете указатель мыши над кнопкой ленты, вместо привычного скудного двух-трехсловного описания в желтом поле вы увидите привлекательное раскрывающееся поле с осмысленным кратким описанием. На рисунке мышь перемещается над командой **Таблица**

Подсказка

Хотите использовать пространство экрана, занятое лентой? Просто щелкните дважды кнопкой мыши текущую вкладку, и лента свернется, оставив видимой только строку с заголовками вкладок. Снова дважды щелкните мышью вкладку для того, чтобы вернуть кнопки на всеобщее обозрение. Это свойство более подробно описано в *примечании "Малоизвестная или недооцененная возможность. Сворачивание ленты" (см. разд. "Просмотр таблиц с помощью области переходов" в главе 1).*

Далее приведен краткий обзор четырех основных вкладок ленты.

- На вкладке **Главная** (Home) собраны вместе разнообразные общие команды, включая хорошо известные инструменты копирования и вставки и команды форматирования для выбора шрифтов и цветов оформления. Здесь же можно найти удобные средства, такие как сортировка, поиск и фильтрация, с которыми вы познакомитесь в *главе 3.*
- На вкладке Создание (Create) находятся команды вставки всевозможных объектов баз данных, о которых вы узнаете в этой книге (более подробную информацию *см. в разд. "Что такое базы данных Access" главы 1).* К ним относятся таблицы, хранящие данные, запросы, исследующие данные, формы, облегчающие редактирование данных, и отчеты, выводящие данные на печатное устройство.
- Вкладка **Внешние данные** (External Data) включает команды импорта данных в программу Access и экспорта их в другие программы. Вы также найдете здесь инструменты для интеграции с Microsoft SharePoint Server. Все эти команды будут применяться в *части VI.*
- На вкладке **Работа с базами данных** (Database Tools) собраны профессиональные средства, которые применяются для анализа базы данных, связывания таблиц, расширения базы данных до возможностей SQL Server. Здесь также можно найти команды для вставки кода на языке Visual Basic, подробному изучению которого посвящена *часть V.*

Имеет смысл потратить немного времени на освоение ленты и ее вкладок. Попытайтесь поочередно щелкнуть кнопкой мыши каждую из них, переходя вперед и назад по четырем разделам, чтобы увидеть их содержимое. В процессе чтения книги вы лучше познакомитесь со всеми этими командами.

Подсказка

Если у вас мышь со скроллером, вы сможете еще легче переходить от вкладки к вкладке, перемещая указатель мыши вдоль ленты и двигая колесико скроллера вперед и назад.

Одно из замечательных свойств ленты заключается в ее неизменности - другими словами, вы не увидите команд, таинственно перемещающихся или исчезающих с мгновение ока. Корпорация Microsoft решила разработать предсказуемую ленту, поэтому ее команды всегда остаются на одном и том же месте. Они лишь слегка изменяют компоновку при изменении окна программы Access, так чтобы оптимально использовать доступное пространство (рис. В3).

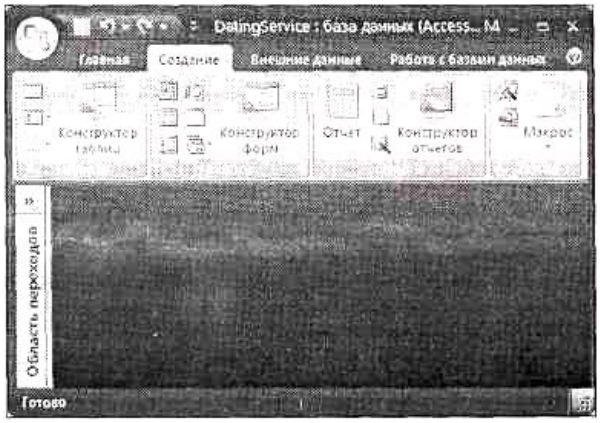


Рис. В3. В этом очень узком окне программы Access не слишком много места для кнопок ленты. Все команды, отображенные на рис. В2, по-прежнему остаются на ленте, но вы видите теперь только маленькие пиктограммы без какого-либо текста. Если вы сомневаетесь в назначении кнопки, проведите указателем мыши поверх нее, чтобы увидеть название

Уголок ностальгии.

Зачем опять изобретать колесо?

Некоторые ветераны программы Access по понятным причинам скептически относятся к ее новому интерфейсу. В конце концов, мы достаточно настрадались от некоторых болезненных экспериментов. В предыдущие версии программы вводились безумные идеи, например, индивидуализированные меню, которые всегда скрывались, как только вам понадобилась команда, раскрывающиеся боковые панели, которые появлялись тогда, когда вы меньше всего этого ожидали, и плавающие панели инструментов, усеивающие весь экран.

В действительности все приложения пакета Office, боровшиеся за сохранение лица более десяти лет, заслуживают нового интерфейса. Меню программ Office не менялись с тех пор, как Word 2.0 завоевал популярность в начале 1990 гг. В те времена базовое меню и единственная панель инструментов были визитной карточкой, поскольку количество команд было достаточно мало. Сегодня программы пакета Office тонут в обилии функциональных возможностей - и они запрятаны так глубоко в разные укромные уголки и щели, что даже профессионалы не знают, где их искать.

Для устранения этих недостатков и предназначена новая лента. Вы не только легко можете понять ее организацию и перемещаться по ней, но и найти в одном месте все необходимое для работы. У разработчиков пользовательского интерфейса корпорации Microsoft теперь новое заклинание: все есть на ленте. Другими словами, если вы хотите найти ту или иную функцию, ищите ее на одной из вкладок в верхней части окна программы Access. Как только вы освоитесь в новой системе, то поймете, что она не только помогает применять ваши любимые инструменты, но и облегчает обнаружение новых функциональных возможностей благодаря[1] просмотру вкладок.

Использование ленты с помощью клавиатуры

Если вы неисправимый любитель клавиатурных команд, то будете рады услышать, что инициировать команды ленты можно с помощью клавиатуры. Секрет в применении ускоряющих клавиш или сочетаний клавиш, начинающихся с клавиши <Alt> (этими сочетаниями вы пользовались для запуска команд меню). Применяя ускоряющие клавиши, не нужно нажимать одновременно все клавиши (как вы скоро увидите, в некоторых из них достаточно букв для того, чтобы скрутить ваши пальцы похлеще, чем в самой буйной игре Twister). Вместо этого вы можете нажимать клавиши одну за другой.

Важно усвоить, что после того как вы нажали клавишу <Alt>, вы делаете две вещи в следующем порядке:

1.Выбираете нужную вкладку.

2.На этой вкладке выбираете команду.

Прежде чем запустить конкретную команду, вы должны выбрать правильную вкладку (даже если она уже есть на экране). Каждое сочетание клавиш требует нажатия, как минимум, двух клавиш, после того как нажата клавиша <Alt>. Вам потребуется и больше нажатий, если нужно пробираться в подменю.

Уже сейчас весь процесс кажется практически бесполезным. Вы действительно готовы запоминать десятки различных сочетаний ускоряющих клавиш?

К счастью, программа Access готова помочь, благодаря новому инструменту, названному "Клавиатурные подсказки" (KeyTips). Вот как он работает. Как только нажата клавиша <Alt>, над каждой вкладкой по мановению волшебной палочки появляются буквы. После того как нажата клавиша выбора вкладки, над каждой кнопкой этой вкладки снова появляются буквы. Теперь можно нажать соответствующую клавишу для запуска команды. На рис. В4 показан этот инструмент о действии.

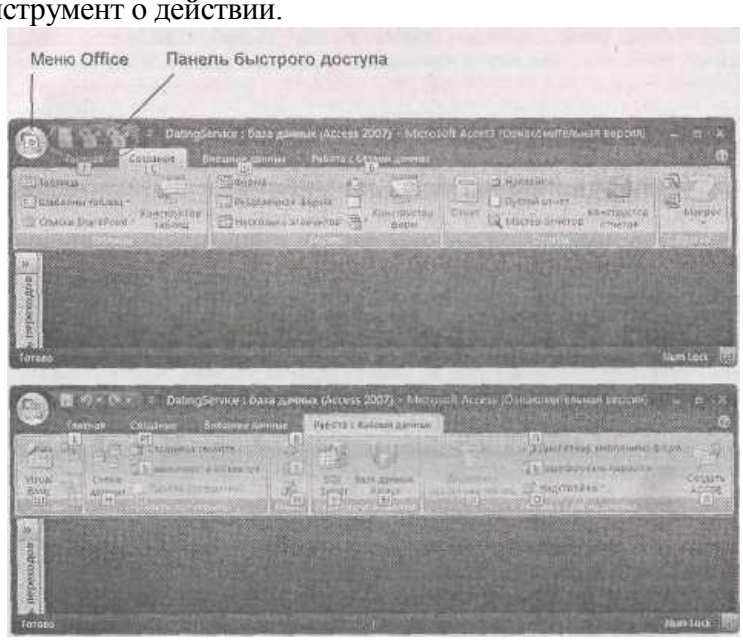


Рис. В4. *Вверху:* когда нажимается клавиша <Alt>, программа Access накалывает буквенные подсказки рядом с каждой вкладкой, над меню Office и над кнопками **Панели быстрого доступа.** *Внизу:* если вы нажмете клавишу <Б> (для выбора вкладки **Работа с базами данных),** то увидите буквы рядом с каждой командой этой вкладки. Теперь можно нажать одну из них для запуска команды (например, <У> перемещает данные на SQL Server). Не пытайтесь установить соответствие между буквами и именами вкладки или кнопки - в ленте спрятано так много функциональных возможностей, что в большинстве случаев буквы ничего не значат

Примечание

В некоторых случаях у команды могут быть две буквы-клавиши, и вам придется нажать поочередно обе клавиши. Выйти из режима Клавиатурных подсказок без запуска можно в любое время, повторно нажав клавишу <Alt>.

Есть и другие сочетания клавиш, не использующие ленту. Эти клавиатурные комбинации начинаются с клавиши <Ctrl>. Например, с помощью сочетания <Ctrl>+<C> копируется выделенный текст, а сочетание <Ctrl>+<S> сохраняет текущую работу. Как правило, узнать ускоряющие сочетания клавиш можно, проведя указателем мыши над командой.

Проведите указателем над кнопкой **Вставить** (Paste) на вкладке ленты **Главная** (Home), и вы увидите подсказку, сообщающую о том, что ускоряющее сочетание клавиш - <Ctrl>+<V>. Если вы работали в предыдущей версии программы Access, то обнаружите, что в Access 2007 сохранилась большая часть ускоряющих клавиатурных сочетаний.

Уголок ностальгии.

Сочетания клавиш в Access 2003

Если у вас есть опыт работы в предыдущий версии Access, вы, возможно, приучили себя использовать сочетания клавиш для вызова команд меню - клавиатурных комбинаций, открывающих меню и выбирающих нужную команду. Если нажать комбинацию клавиш <Alt>+<E> в Access 2003, раскрывается меню **Редактирование** (Edit) (в главном меню). Затем можно нажать клавишу <S> для выбора команды **Специальная** вставка (Paste Special).

На первый взгляд не кажется, что эти ускоряющие клавиши много значат в программе Access 2007. В конце концов, в Access 2007 даже нет главного меню! К счастью, корпорация Microsoft

согласилась на дополнительные трудности, чтобы облегчить жизнь многолетним приверженцам Access. Вы можете по-прежнему пользоваться вашими ускоряющими сочетаниями клавиш для команд меню, но они действуют несколько иначе. Если нажать комбинацию клавиш <Alt>+<E> в Access 2007, над лептой появляется всплывающая подсказка (рис. B5), сообщающая о том, что вы начали вводить ускоряющее клавиатурное сочетание для меню Access 2003. Если вы продолжите и нажмете клавишу <S>, то попадете в знакомое диалоговое окно команды **Специальная вставка** (Paste Special), поскольку программа знает, что вы собираетесь делать. Это поведение похоже на действие за кадром невидимого меню.

Конечно, это средство не всегда вас выручит. Оно не сработает, если вы попытаетесь применить одну из уже несуществующих команд. Если вы захотите увидеть меню для того, чтобы вспомнить, какую клавишу нажать далее, попытка будет безуспешной. Access просто выведет на экран всплывающую подсказку.

Рис. B5. Нажав сочетание клавиш <Alt>+<E>, вы запускаете "воображаемое" меню **Редактирование.**

Вы не можете увидеть его, поскольку в Access 2007 оно не существует. Но всплывающая подсказка дает знать о том, что программа обратила внимание на вашу команду. Теперь можно завершить действие, нажав следующую клавишу для запуска команды меню

Меню Office

В программе Access 2007 вроде бы сохранилась небольшая часть традиционного меню Access. Стандартное меню **Файл** (File), позволяющее открывать, сохранять и печатать файлы, преобразовано в меню **Office.** Вы попадаете в него с помощью кнопки **Office -** большого круглого логотипа в верхнем левом углу окна программы (рис. B6).

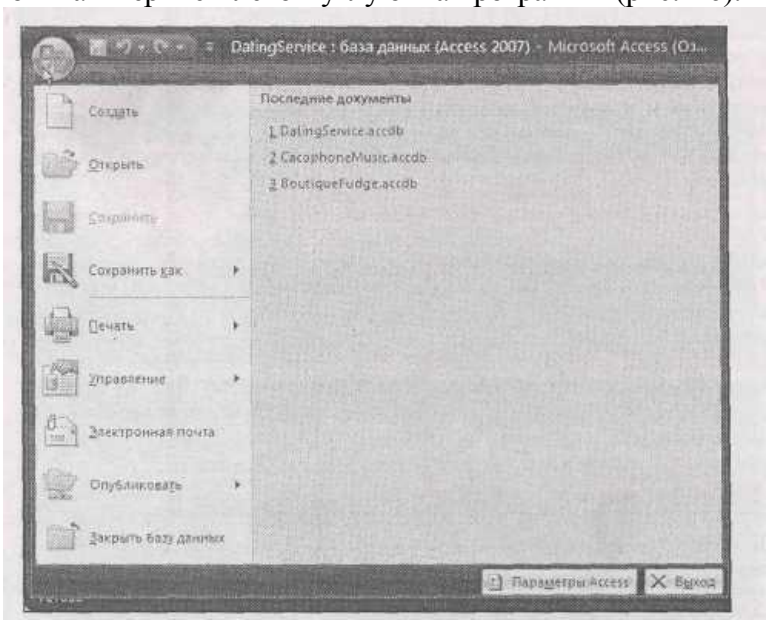


Рис. B6. Меню Office крупнее и легче читается, чем традиционное меню.

Если щелкнуть его кнопкой мыши, на экран выводится список команд меню *(слева)* и перечень недавно использовавшихся баз данных *(справа)*

Как правило, меню **Office** применяется для решения трех задач:

■ открытие, создание и сохранение вашей базы данных. Вы много раз будете делать это в *главе 1;*

■ печать вашей работы *(см. главу 3)* и рассылка ее другим людям по электронной почте *(см.*

главу 19);

■ настройка поведения программы Access. Выберите кнопку **Параметры Access** (Access Options) в нижней части меню для вывода диалогового окна **Параметры Access** - полнофункционального центра для настройки установочных параметров программы Access.

У меню есть одна особенность, к которой надо привыкнуть. Некоторые команды меню **Office** скрывают в подменю с дополнительными командами. Возьмем команду **Печать** (Print). Вы можете выбрать **Печать** в меню **Office** для быстрого вывода на принтер вашей работы. Но если щелкнуть направленную вправо стрелку у края строки с командой **Печать**

(или несколько секунд подвигать над ней указателем мыши), то вы увидите подменю с дополнительными командами, показанными на рис. В7.

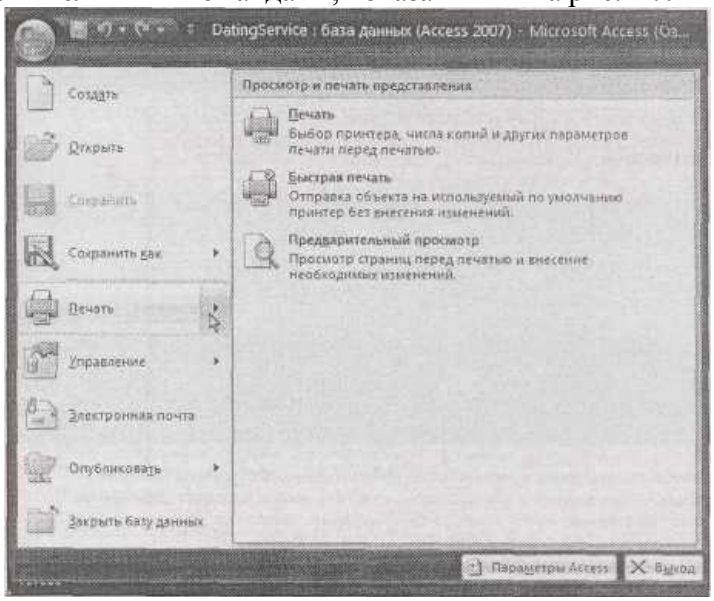


Рис. В7. **Печать -** одновременно и команда меню, запускаемая щелчком кнопки мыши, и подменю. Для вывода на экран подменю нужно подвигать указателем мыши над командой **Печать** (без щелчка кнопкой мыши) или щелкнуть мышью стрелку у правого края (показанную здесь). На ленте есть еще несколько кнопок, действующих аналогично

Инструментальная Панель быстрого доступа

Зоркий взгляд заметит крошечный кусочек экрана, участок, расположенный справа от кнопки **Office** сразу над лентой (рис. В8). На нем находится ряд маленьких пиктограмм, похожих на инструментальные кнопки в более ранних версиях программы Access. Эта область экрана называется **Панелью быстрого доступа** (Quick Access toolbar или QAT для фанатов Access).

Если бы **Панель быстрого доступа** была только инструментальной панелью с тремя командами, о ней не стоило бы и говорить. Но основная изюминка ее в том, что вы можете настраивать эту панель. Другими словами, можно удалить с панели неиспользуемые команды-кнопки и добавить свои любимые. В *приложении* рассказано, как это делается.

Корпорация Microsoft сознательно сохранила **Панель быстрого доступа** очень маленькой. Она спроектирована для того, чтобы дать выход желанию создать интерфейс, соответствующий вашим предпочтениям. Если вы, пользуясь отсутствием каких-либо ограничений, полностью замените команды **Панели быстрого доступа** своими собственными, остальная лента останется неизменной. (А это значит, что коллега или ваша супруга смогут пользоваться вашим компьютером, не страдая от головной боли.)

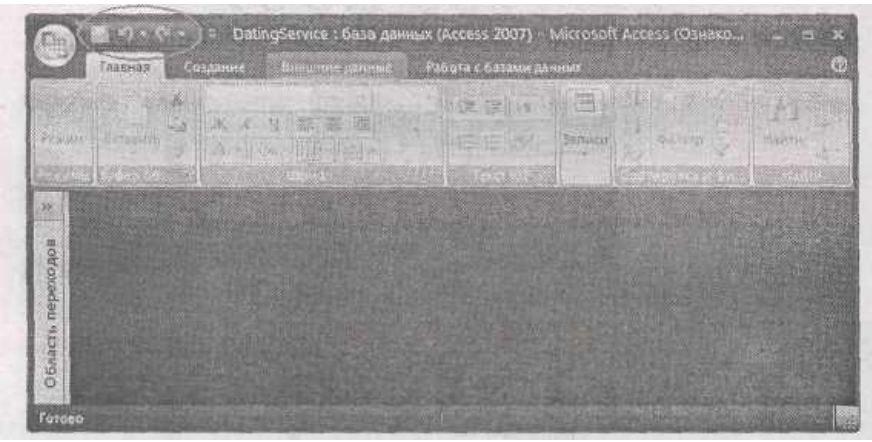


Рис. В8. Панель быстрого доступа держит у вас под рукой команды **Сохранить, Отменить** и **Вернуть.** Программа Access выделяет эти команды, поскольку ими пользуются чаще других. Но, как вы узнаете из *приложения,* на эту панель можно добавить любую нужную вам команду

Новые возможности в программе Access 2007

Самое впечатляющее нововведение Access 2007 - новый пользовательский интерфейс программы. Но новый внешний вид - не единственное усовершенствование. Когда корпорация Microsoft решила переработать программу Access 2007. она сформировала команду разработчиков в семь раз больше той, которая разрабатывала Access 2003. Этот дополнительный человеческий ресурс помог внести ряд долгожданных изменений. Далее перечислены основные из них.

■ *Новый усовершенствованный процессор баз данных (database engine).* Многолетние профессиональные пользователи программы Access знают, что программа использует за кадром процессор Jet для управления операциями (такими как добавление и обновление данных). У Access 2007 теперь есть собственная специализированная версия Jet, позволившая создателям Access добавить новые функциональные возможности и применить тонкую настройку рабочих характеристик программы.

Примечание

Не беспокойтесь, программа Access 2007 на 100% обратно совместима с предыдущими версиями программы. Это означает, что вы можете продолжать использовать базы данных старого стиля, созданные в Access 2003, но при этом не сможете применять новые возможности (например, вложения или сложные данные, описанные далее) до тех пор, пока не создадите файл базы данных нового формата.

■ *Вложения (Attachments).* Одна из самых замечательных функциональных возможностей Access - тип данных Вложение (Attachment), позволяющий хранить целые файлы в вашей базе данных, включая изображения, документы и электронные таблицы. (Не стоит использовать видео, аудио и другие объемистые медиафайлы, т. к. размер любой базы данных Access ограничен 2 Гбайт.) Как применять вложения, вы узнаете в *разд. "Вложение" главы 2.*

■*Легко конструируемые формы и отчеты.* Создание привлекательной формы для ввода данных или отчета для вывода на принтер всегда требовало множества рутинных операций. В Access 2007 сформировать их гораздо легче благодаря автоматической верстке - функциональной возможности, группирующей связанные порции данных в единое целое в виде подходящих столбцов или таблиц. В программе есть и новый Режим макета (Layout), позволяющий вставить форматирование и тут же увидеть результат.

■*Сложные данные (Complex data).* Это необязательное новое свойство, позволяющее хранить в одном поле несколько значений (или "сегмент" данных). Сложные данные облегчают процесс связывания таблиц. Например, с помощью сложных данных вы можете связать несколько авторов с одной книгой. Главная причина введения сложных данных - поддержка функции объединения сервисов SharePoint.

Примечание

Сложные данные не для всех. Некоторые специалисты по базам данных считают их

излишествами, способными создать дополнительную путаницу. В *главе 5 в*ы узнаете, как связывать таблицы, и решите, удобно ли для этого специальное, ускоряющее средство, и хотите ли вы ими пользоваться.

Интеграция с помощью SharePoint. Сервисы SharePoint - это популярный набор функций, разработанных для того, чтобы помочь группам людей совместно использовать данные и сотрудничать в пределах организации. Эти функции встроены в операционную систему Windows Server 2003. Применяя программу Access 2007, вы сможете обрабатывать и модифицировать информацию, хранящуюся в списке SharePoint.

Примечание

Вы вряд ли захотите использовать SharePoint, если удовлетворены средствами, предоставляемыми программой Access. Но вам понравятся возможности интеграции SharePoint, если вы уже применяете сервисы SharePoint для хранения данных или вам нужно предоставлять информацию для совместного использования множеству людей, а обычная база данных Access просто не поддерживает эти функции. В *главе 21* вы узнаете больше о возможности совместной работы сервисов SharePoint и программы Access.

■ *Улучшенная защита при работе с кодом.* Как вы узнаете в *части V,* специалисты по базам данных применяют макросы и написанные вручную программные процедуры для решения трудных задач. Но программа Access печально известна своей недоверчивостью к любому программному коду, поскольку у нее нет способов выявления кода, способного совершить что-либо опасное (например, удалить ваши файлы). В результате у программы Access появилась досадная привычка отключать на всякий случай весь ваш код. В Access 2007 вы можете использовать новый центр управления безопасностью (trust center), позволяющий задать базы данных, которым вы готовы доверять, полагаясь на их создателя или на место их хранения. Access 2007 лучше распознает безопасный код (код, не способный привести к серьезным повреждениям), который программа не отключает даже в базах данных, не заслуживших доверия.

Об этой книге

Несмотря на многолетние и многочисленные усовершенствования программного обеспечения, одна составляющая не претерпела никаких изменений: документация корпорации Microsoft. В поставку пакета Office 2007 вообще не входит печатное руководство пользователя. Microsoft по-видимому рассчитывает, что вы будете читать интерактивную help-информацию для того, чтобы познакомиться с тысячами функциональных возможностей, включенных в это программное обеспечение. Порой эти экраны помощи действительно очень полезны, например, если вы ищите краткое описание, объясняющее новую загадочную функцию, С другой стороны, если вы хотите узнать, скажем, о том, как создать привлекательную диаграмму, то не найдете ничего, кроме сжатых и иногда непонятных инструкций.

Эта книга представляет собой подробное руководство к программе Access 2007. На ее страницах вы найдете подробные инструкции и советы по использованию почти всех функциональных возможностей Access, включая те, о которых вы (пока) не слышали.

Краткое содержание

Книга разделена на семь частей, каждая из которых состоит из нескольких глав.

■ *Часть I. Хранение данных в таблицах.* В этой части вы создадите свою первую базу данных и узнаете, как добавлять и редактировать таблицы, хранящие данные. Затем вы приобретете практические навыки, необходимые для выявления ошибок до их возникновения, просмотра вашей базы данных и связывания таблиц.

■ *Часть II. Обработка данных с помощью запросов.* В этой части вы будете создавать запросы - специализированные команды, способные извлекать интересующие вас данные, вносить изменения в них и суммировать большие объемы данных.

■ *Часть III. Отчеты.* В этой части показано, как с помощью отчетов можно превратить "голые" данные в ваших таблицах в хорошо сделанные распечатки с помощью форматирования, порой изобретательного, и промежуточных итогов.

■ *Часть IV. Разработка пользовательского интерфейса с помощью форм.* В этой части вы создадите формы - настраиваемые окна, которые делают ввод данных легкой задачей даже для

новичков программы Access.

■ *Часть V. Программирование в Access.* Теперь, когда вы овладели основными сведениями о проектировании баз данных, то готовы погрузиться в черную магию программирования в Access. В этой части вы будете использовать макросы и программирование на языке Visual Basic для автоматизации сложных задач и решения распространенных проблем.

■ *Часть VI. Совместное использование Access.* В этой части вы научитесь извлекать паши данные из файлов других типов (или помещать в них данные), таких как текстовые документы и электронные таблицы программы Excel. Вы также увидите, как применять Access для взаимодействия с некоторыми наиболее мощными программами корпорации

Microsoft: процессором баз данных SQL Server и программным обеспечением для совместной работы SharePoint Server.

■ *Приложение.* Книга заканчивается приложением, показывающим, как настроить Панель быстрого доступа для облегчения использования ваших любимых команд.

Об → этих → стрелках

На всем протяжении книги вы будете встречать предложения, подобные следующему:

"Выберите последовательно **Создание** -> **Таблицы -> Таблица** (Create -> Tables -» Table)".

Это способ сокращенной записи последовательности действий, помогающий вам найти нужную функциональную возможность или команду на ленте программы Access. Его можно расшифровать следующим образом: "На ленте щелкните кнопкой мыши вкладку **Создание.** На вкладке найдите группу **Таблицы.** В группе **Таблицы** щелкните мышью кнопку Таблица". (Вернитесь к рис. В.2 и найдите нужную кнопку.)

Как было показано на рис. В3, вид ленты меняется в зависимости от размера окна программы. Если размер окна Access мал, вы можете не увидеть на экране текста, поясняющего назначение кнопки, которую вы должны щелкнуть мышью. Вместо этого отображается только маленькая пиктограмма. В подобной ситуации можно поводить указателем мыши над загадочной кнопкой для того, чтобы увидеть ее имя и решить, щелкать ее мышью или нет.

Если вы сделали окно программы совсем маленьким, то могли не оставить места для отображения раскрытой группы. В этом случае, группа отображается как одна кнопка с названием группы. Щелкните мышью эту кнопку, и пропавшие команды появятся на раскрывающейся панели (рис. В9).

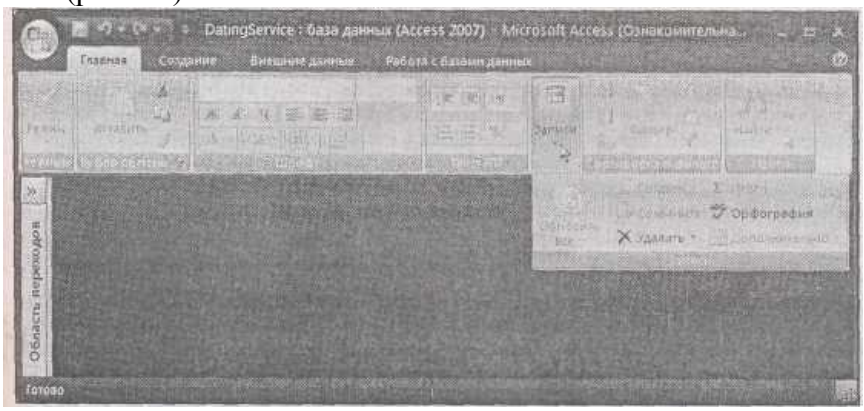


Рис. В9. В данном примере у программы Access недостаточно места для вывода на экран в раскрытом виде групп **Режимы, Записи** или **Найти,** поэтому они заменены кнопками.

Если щелкнуть мышью одну из них, на экране появится панель с искомым содержимым.

Контекстные вкладки

Несмотря на то, что хорошо сделанные, предсказуемые вкладки - замечательная идея, некоторые функции нужны только в определенных обстоятельствах. Скажем, вы начинаете проектирование таблицы. У вас должно быть немного больше функций, чем при вводе данных. Программа Access учитывает эту ситуацию, вставляя на ленту одну или несколько контекстных вкладок, зависящих от решаемой в данный момент задачи. На этих вкладках есть дополнительные команды, применение которых ограничено конкретным сценарием (рис. В10).

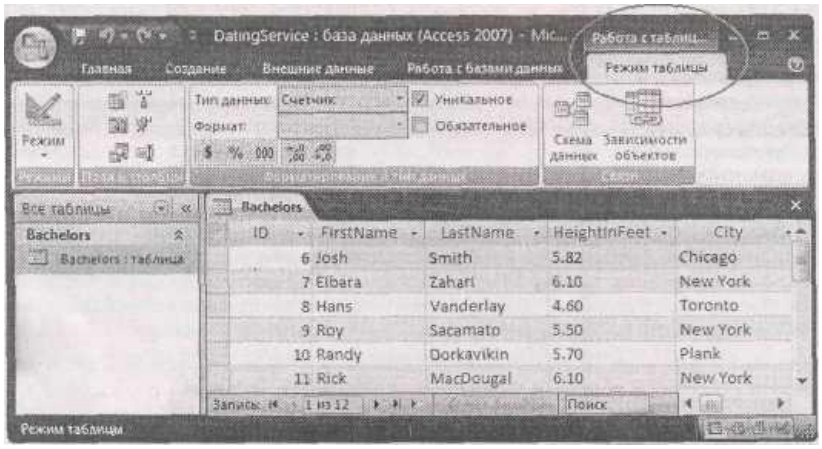


Рис. В10. Когда вы разрабатываете форму, под заголовком вкладки **Работа с таблицами** появляется новая контекстная вкладка, названная **Режим таблицы.**

Контекстные вкладки всегда появляются в правой части ленты

Если речь идет о контекстных вкладках, инструкции в этой книге всегда включают название основной вкладки **(Работа с таблицами** на рис. В10). Приведем пример: "Выберите **Работа с таблицами | Режим таблицы → Поля и столбцы → Новые поля** (Table Tools | Datasheet → Fields & Columns → New Fields)". Обратите внимание на первую часть этой инструкции - она включает название обычной вкладки **Работа с таблицами** (Table Tools) и отделенное символом | имя контекстной вкладки **Режим таблицы** (Datasheet).

Раскрывающиеся кнопки

Время от времени вы будете обнаруживать на ленте кнопки с вложенными в нее краткими: меню. У некоторых кнопок меню появляются, как только вы щелкните кнопку мышью, а у других - только если щелкнуть мышью стрелку кнопки, направленную вниз, как показано на рис. В11.

При работе с такими кнопками последнее действие, приведенное в книге, сообщает о варианте, который следует выбрать из раскрывающегося меню. Например, вам рекомендуется: **"Главная → Режимы → Режим → Конструктор** (Home → Views → View → Design View)". Это значит, что вы должны выбрать вкладку **Главная** (Home), найти группу Режимы (Views), щелкнуть мышью нижнюю часть кнопки **Режим** (View) (для вывода меню с дополнительными вариантами) и затем выбрать из меню **Конструктор** (Design View).

Примечание

Присмотритесь к направленным вниз стрелкам, расположенным на ленте, они поначалу кажутся коварными. Для того чтобы увидеть полный перечень возможных вариантов, нужно щелкнуть мышью часть кнопки с направленной вниз стрелкой. Если щелкнуть мышью другую часть кнопки, вы не увидите списка. Вместо этого программа Access запустит стандартную команду (ту, которую программа считает наиболее распространенным вариантом) или команду, которую вы использовали последней.

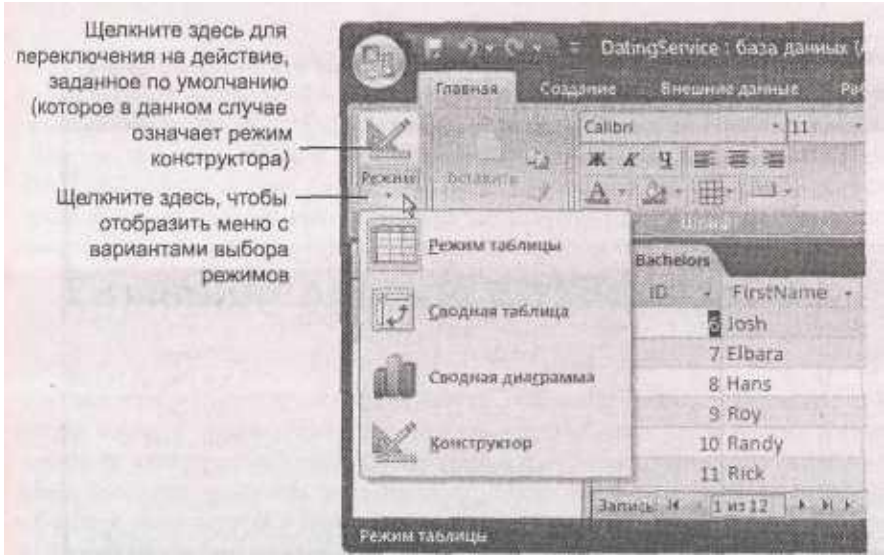


Рис. В11. Программа Access предоставляет возможность переключаться в разные режимы отображения базы данных. Щелкните мышью нижнюю часть кнопки **Режим,** чтобы увидеть показанное меню, или щелкните верхнюю часть кнопки, чтобы перейти в следующий режим в перечне без каких-либо уточняющих вопросов

Традиционные меню

Как вы уже видели, лента заменила традиционные панели инструментов и меню. Но в паре случаев вы все еще будете применять знакомое меню Windows, например, при использовании редактора Visual Basic *(см. главу 16).* В такой ситуации стрелки указывают на уровни меню. Инструкция "Выберите **Файл → Открыть** (File → Open)" означает, что нужно щелкнуть кнопкой мыши заголовок меню Файл (File). Затем в меню Файл (File) следует щелкнуть мышью команду **Открыть** (Open).

Аналогичные инструкции встретятся вам при использовании меню **Office.** Они выглядят примерно так: "Выберите кнопку **Office → Открыть** (Office → Open)". Эта инструкция переводится следующим образом: "Щелкните мышью кнопку **Office** в левом верхнем углу окна для вывода меню **Office.**

Далее выберите в меню команду **Открыть** (Open)".

О сочетаниях клавиш

Каждый раз, отрывая пальцы от клавиатуры для того, чтобы взять в руки мышь, вы теряете несколько микросекунд. Вот почему многие опытные компьютерные фанаты везде, где это возможно, пользуются сочетаниями клавиш вместо панелей инструментов и меню. Например, сочетание клавиш <Ctrl>+<S> сохраняет вашу текущую работу в программе Access (как и в большинстве других программ).

Если вы видите в книге сочетание <Ctrl>+<S>, это говорит о том, что вы должны нажать и держать нажатой клавишу <Ctrl> и, пока она нажата, нажать клавишу <S>, а затем отпустить обе клавиши. Аналогично заплетающее пальцы сочетание клавиш <Ctrl>+<Alt>+<S> означает, что нужно удерживать нажатой клавишу <Ctrl>, затем нажать и держать клавишу <Alt> и, наконец, нажать клавишу <S> (так, чтобы все три клавиши оказались нажатыми одновременно).

О щелчках кнопкой мыши

В этой книги приводится три вида инструкций, требующих применения компьютерной мыши или встроенной сенсорной панели (track pad). Фраза "щелкнуть кнопкой" означает навести указатель в виде стрелки на какой-то объект на экране и затем - совсем не двигая указатель - нажать и отпустить левую кнопку мыши (сенсорной панели ноутбука). Двойной щелчок, конечно же, означает два быстрых последовательных щелчка кнопкой мыши также без какого-либо перемещения указателя. И, наконец, "перетащить мышью" означает перемещать указатель мыши с нажатой левой кнопкой.

Примеры

Читая книгу, вы столкнетесь с рядом примеров, демонстрирующих функциональные возможности программы Access и средства построения хороших баз данных. Многие из них доступны в виде файлов баз данных Access, каждый из которых можно загрузить из Интернета. Просто перейдите на Web-сайт **www.missingmanuals.com,** щелкните кнопкой мыши ссылку на эту книгу, а затем ссылку **Missing** CD (отсутствующий CD) для того, чтобы перейти на страницу, с которой можно загрузить zip-файл с примерами, упорядоченными по главам.

О Web-сайте MissingManuals.com

На указанном Web-сайте можно найти новости, статьи и обновления книг серий "Missing Manual" (недостающее руководство) "For Starters" (для начинающих).

Кроме того, Web-сайт предлагает исправления и обновления для данной книги (чтобы увидеть их, щелкните кнопкой мыши заголовок книги и затем ссылку **Errata** (список опечаток)). Мы будем рады, если вы отправите собственные исправления и обновления. Для того чтобы книга оставалась современной и точной, насколько это возможно, при каждой допечатке тиража книги мы вносим предложенные вами и подтвердившиеся корректировки[1]. Мы также упоминаем о таких изменениях и на Web-сайте, поэтому вы сможете, если захотите, внести важнейшие

корректировки в ваши экземпляры книги.

Нам также очень хотелось бы услышать ваши собственные предложения, касающиеся будущих книг в сериях "Missing Manual" и "For Starters". На Web-сайте для них отведено специальное место, как место для подписки на бесплатную рассылку издательских планов, касающихся книг названных серий.

Safari Enabled

Когда вы видите пиктограмму Safari® Enabled на обложке книги о вашей любимой технологии, это означает, что книга доступна в Интернете благодаря сервису издательства O'Reilly Network Safari Bookshelf.

Safari предлагает лучшее решение, чем просто электронные книги.

Это виртуальная библиотека, позволяющая легко находить тысячи книг о передовых технологиях, вырезать и вставлять код примеров, загружать главы и находить быстрые ответы, если вам необходима точная и свежая информация.

Испытайте этот сервис бесплатно на Web-сайте **http://safari.oreilly.com.**

[1] *Это утверждение касается оригинального издания. - Ред.*

Часть I

Хранение данных в таблицах

1. Глава 1. Создание вашей первой базы данных

Несмотря на то, что корпорация Microsoft не признает этого, программа Access может произвести устрашающее впечатление, вызывая появление холодной испарины у самых самоуверенных сотрудников. Хотя Microsoft потратила миллионы на попытки облегчить работу с Access, большинство пользователей все еще считают ее самой сложной программой в пакете Office.

И, вероятно, они правы.

Access отпугивает больше, чем другие программы пакета Office из-за особенностей работы с базами данных (БД).

Просто БД *требуют строгого соблюдения правил.*

Другие программы не столь требовательны. Например, вы можете активизировать Word и сразу же начать набирать письмо. Или можно запустить Excel и энергично заняться финансовым отчетом.

Программа Access не так легко управляема. Прежде чем ввести малейшую порцию информации в базу данных, вам придется сформировать *схему* БД. И даже после этого шага, возможно, вам придется потратить время на создание других полезных инструментов, таких как легко управляемые процедуры поиска и удобные формы, которые можно использовать для упрощения поиска и ввода данных. Подобная настройка требует усилий и хорошего понимания принципов работы БД.

В этой главе вы преодолеете имеющееся у вас предубеждение против программы Access и научитесь создавать простую, но действующую БД.

Одновременно вы познакомитесь с новым удобным пользовательским интерфейсом программы Access и узнаете, что именно можно хранить в БД.

После этого вы будете готовы приняться за постижение высокого искусства проектирования баз данных, подробно описанного в данной книге.

1.1. *Что такое базы данных Access*

Как вы уже знаете, БД - это коллекция информации.

В программе Access каждая база хранится в одном файле.

Такой файл содержит объекты или компоненты БД.

Объекты БД играют главную роль в БД Access.

Всего у вас есть шесть объектов БД разных типов:

1) ***Таблицы*** хранят данные. Таблицы - главный компонент любой БД, и их можно создать столько, сколько нужно для хранения данных разных типов. В БД о вашей физической форме, состоящей из трех таблиц, можно было бы хранить регистрационный журнал ваших ежедневных пробежек, перечень тренажеров и количество богатых белком коктейлей из молочной сыворотки, которые вы проглатываете за день.
2) ***Запросы*** позволяют быстро обработать таблицу. Обычно такая обработка включает извлечение выбранной порции информации (например, 10 самых популярных продуктов ресторанов-закусочных Ed's Roadside Dinner или все ваши покупки за день). Запросы также можно использовать для внесения изменений.
3) ***Формы*** - это чудесные окна, которые вы создаете, структурируете и раскрашиваете. Формы облегчают просмотр или изменение данных таблицы.
4) ***Отчеты*** помогают напечатать часть данных или всю информацию из таблицы. Вы можете выбрать размещение информации на печатной странице, варианты ее группировки и сортировки, а также способ форматирования.
5) ***Макросы*** - это мини-программы, автоматизирующие выполнение пользовательских задач. Макросы помогают получить желаемые результаты без знания программирования.
6) ***Модули*** - это файлы, содержащие код на языке Visual Basic. Вы можете использовать этот код для выполнения задач, подобных обновлению 10 000 записей или отправки электронной почты. (*Глава 16* целиком посвящена языку Visual Basic.)

Знатоки программы Access называют все эти составляющие объектами БД, поскольку всеми

ими вы по существу управляете одинаково. Если нужно использовать конкретный объект, вы включаете его в вашу БД, присваиваете ему имя и затем выполняете его тонкую настройку. Впоследствии вы сможете просматривать свои объекты, переименовывать их или удалять тс из них, которые вам больше не нужны.

Примечание

Проектирование БД - это процесс вставки и настройки объектов БД. Для хранения этого множества БД Access может включать 32 768 разных объектов.

В данной главе вы познакомитесь с важнейшим типом объектов БД - таблицами. Но сначала вам следует узнать больше о БД и рабочей среде программы Access.

1.2. *Приступая к работе*

Пора начать ваше путешествие и запустить программу Access.

Вы начнете с яркой страницы **Приступая к работе с Microsoft Office Access** (Getting Started with Microsoft Office Access) (рис. 1.1).

Часто задаваемый вопрос.

Использование чужой БД

Могу ли я использовать БД Access, *которую проектировал кто-то другой?*

Несмотря на то, что все БД - это всегда результат двухэтапного процесса (сначала кто-то создаст их, а потом люди заполняют их данными), необязательно, чтобы один и тот же человек выполнял обе задачи. В реальном деловом мире часто разные люди работают над их выполнением.

Например, смышленый студент, проводящий летние каникулы на пивном складе, может создать БД для отслеживания заказов (задача № 1). Затем отдел продаж может использовать БД для ввода новых заказов (задача № 2), в то время как другие служащие будут искать заказы и заполнять их (тоже задача № 2). Сотрудники склада могут проверять достаточность уровня запасов (снова задача № 2), а штатный бухгалтер (resident accountant) может следить за общим объемом продаж (задача № 2).

Если задача № 1 (создание БД) выполнена хорошо, выполнение задачи № 2 (использование БД) может быть очень легким. На самом деле, если БД хорошо спроектирована, люди, немного знающие программу Access, могут спокойно использовать ее для ввода, обновления и поиска информации. Поразительно, но им даже совсем не надо знать, что они работают в программе Access.

О групповом использовании Access вы узнаете больше и *главе 18.*

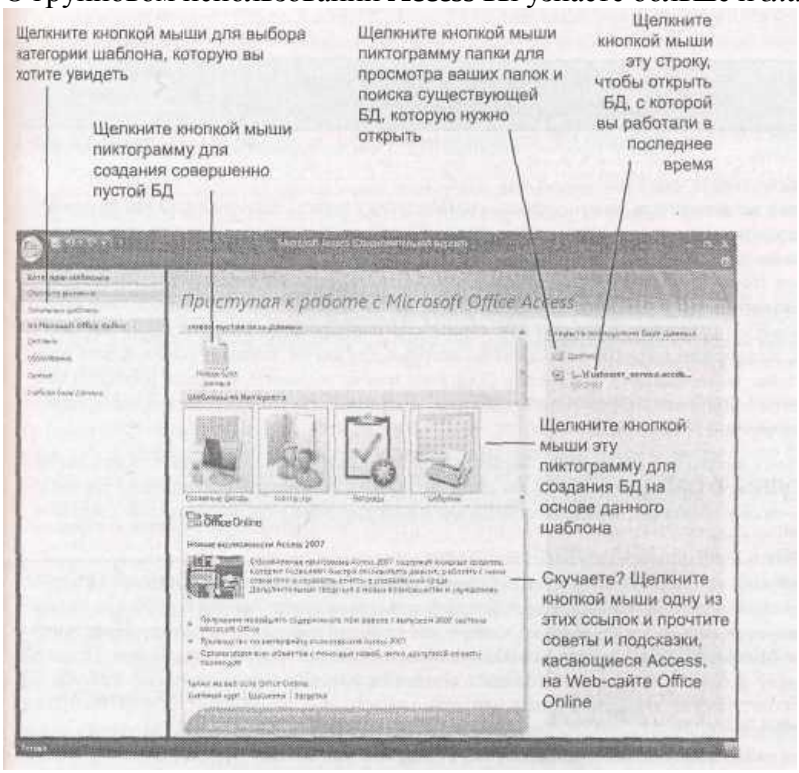


Рис. 1.1. Страница **Приступая к работе с Microsoft Office Access -** это гибрид окна Windows-программы и Web-страницы.

Используйте ссылки слева для просмотра разных категорий шаблонов (готовые к применению БД, которые вы можете загрузить и заполнить своими данными). Или проверьте ссылки внизу, которые указывают на последние новости и советы, касающиеся программы Access

На первый взгляд страница **Приступая к работе с Microsoft Office Access** немного сбивает с толку, но на самом деле она служит следующим трем целям.

■ Знакомит вас с новейшей информацией Web-сайта Microsoft's Office Online. Например, вы можете прочесть полезные статьи о программе Access, найти экономящие время подсказки или загрузить обновления. Все ссылки открываются в отдельном окне Web-обозревателя.

- Она позволяет вам открыть БД, которую вы использовали недавно. Найдите справа раздел **Открыть последнюю базу данных** (Open Recent Database), в котором отображается список БД.
- На этой странице можно создать новую БД. Вы можете начать с пустой БД (используя пиктограмму **Новая база данных** (Blank Database)) или попытаться найти готовый к использованию шаблон, отвечающий вашим требованиям.

На профессиональном уровне.

Шаблоны, подходящие для разных целей

Шаблоны - это заранее сформированные БД. Они избавляют вас от необходимости создавать свою БД и позволяют перейти сразу к ее тонкой настройке и вводу данных,

Как вы уже догадались, за это удобство приходится платить. Даже если вы найдете шаблон, хранящий данные, которые вы хотите отслеживать, может оказаться, что заранее определенная схема не вполне правильная. Например, если вы решили использовать шаблон Home Inventory (домашний склад) для учета всего хлама **с** вашем подвале, может оказаться что в шаблоне пропущены кое-какие данные, которые вы бы хотели использовать (например, планируемая стоимость перепродажи вашего хлама на сайте eBay), и содержатся другие подробности, которые для вас не важны (такие как дата приобретения вами каждого предмета). Для того чтобы заставить шаблон работать, вам придется изменить проект вашей таблицы и применить те же средства программы Access, что при ее создании.

Эта книга научит вас создавать собственные БД с нуля и подгонять в них каждый дюйм. Став знатоком программы Access, вы сможете проводить много доставляющих удовольствие часов в попытках переделать шаблоны и адаптировать их в соответствии с вашими потребностями.

Возможно, вам захочется настроить страницу **Приступая** к **работе с Microsoft Office Access.**

Программа Access предоставляет вам и такую возможность, но это не простая задача и рекомендуется только учреждениям, желающим стандартизировать страницу **Приступая** к **работе с Microsoft Office Access** для облегчения условий работы их сотрудников. Предприятие может добавить ссылки на Web-сайт компании или часто используемый шаблон БД. Если вас интересует эта функциональная возможность, вам необходимо другое программное средство: свободно распространяемый пакет Access Developer'sToolkit, который можно найти на сайте **http://msdn.microsoft.com.** (Во время написания этой книги он все еще не был выпущен.)

Страница **Приступая к работе с Microsoft Office Access -** это лишь входная дверь в программу Access, в этом хранилище можно найти гораздо больше, если начать прокручивать его содержимое. Но вы не можете опробовать другие возможности Access, пока не создадите новую БД, в следующем разделе показано, как это сделать.

1.2.1. *Создание новой базы данных*

В этой главе вы создадите на скорую руку довольно простую БД. Пример разработан для хранения списка кукол-болванчиков (bobblehead dolls). (Для тех, кто не знает,

кукла-болванчик - это игрушечная фигурка с необычно большой головой на пружине, создающей причудливые покачивания головы при легком касании. Куклы-болванчики обычно похожи на известных звезд, политиков, спортсменов или вымышленных персонажей.)

На профессиональном уровне.

Работа Access в интерактивном режиме

Одна из лежащих на поверхности функциональных возможностей страницы **Приступая к работе с Microsoft Office Access** - способ получения свежей информации из Web-пространства. Этот процесс настолько незаметен, что вы даже не подозреваете о его существовании. Когда вы запускаете программу Access, она за кадром связывается с высокопроизводительными Web-серверами корпорации Microsoft и запрашивает последнюю информацию для области ссылок в нижней части страницы **Приступая к работе с Microsoft Office Access.** Время от времени вы замечаете, что содержимое в этой области меняется. Один день вы можете видеть ссылку на статью о макровирусах, а на другой день появится статья с экономящими время подсказками или советами. (Если щелкнуть ссылку кнопкой мыши, статья загрузится в окно **Справка** программы Access, но не дайте себя одурачить: его содержимое - это по-прежнему просто Web-страница, взятая с сайта Office Online.)

Такое же таинство совершается, когда вы просматриваете шаблоны (щелкнув одну из категорий под заголовком **Из Microsoft Office Online).** Снова программа Access обращается к Web-пространству, на этот раз для того, чтобы получить список подходящих шаблонов.

Эта основанная на связи с Web-пространством система позволяет вам воспользоваться преимуществами, предоставляемыми новейшими разработками и самой свежей информацией, и освобождает от необходимости обновления программного обеспечения Access. Само собой разумеется, вы не увидите никаких обновлений, если ваш компьютер не сможет подключиться к Интернету. (Вместо этого вы будете бесконечно видеть одно и то же устаревшее содержимое.)

Если вы хотите просмотреть более полный каталог статей и источников, касающихся программы Access, то можете отправиться на Web-сайт Office Online самостоятельно (вне программы Access), указав адрес **http://office.microsoft.com** в вашем любимом Web-обозревателе.

Примечание

БД Bobblehead и все остальные БД, использованные в этой книге, можно найти в Интернете. Более подробную информацию *см. в разд. "Примеры" во введении.*

Далее перечислены действия, необходимые для создания новой пустой БД.

1. На странице **Приступая к работе с Microsoft Office Access** щелкните мышью пиктограмму **Новая база данных.**

Справа появится боковая панель (рис. 1.2).

2. Введите имя файла.

Программа Access хранит всю информацию о БД в одном файле с расширением accdb (что означает БД Access). Не соглашайтесь на имя, которое Access выбирает автоматически (например, Database1.accdb). Вместо него подберите что-нибудь более подходящее. В этом примере имя Bobblehead.accdb вполне подходит.

Названия файлов программы Access, как и имена любых других файлов, могут включать комбинацию букв, пробелов, цифр, скобок, дефисов (-) и знаков подчеркивания (_). Как правило, лучше избегать применения других специальных символов, некоторые из них недопустимы.

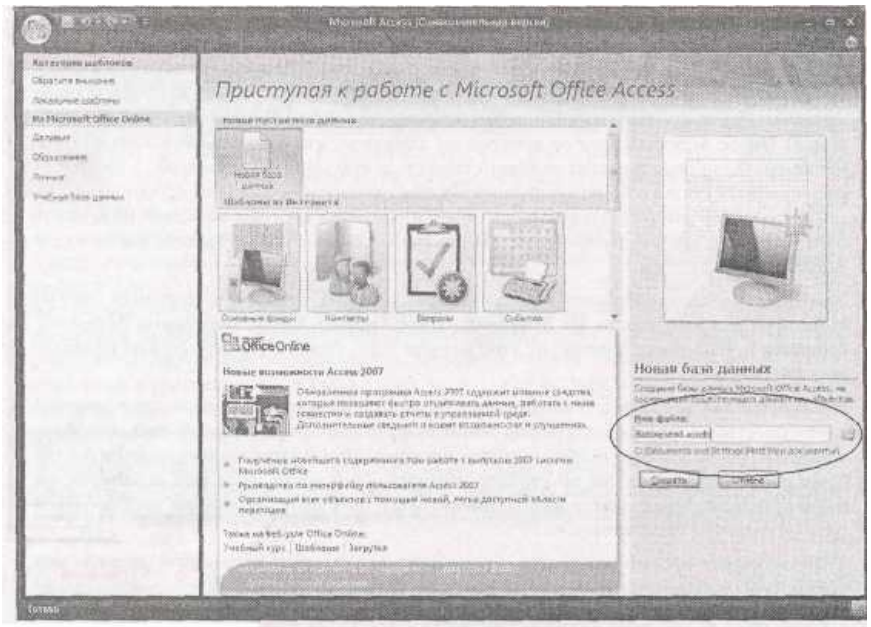


Рис. 1.2. БД Bobblehead.accdb будет размещена в папке C:\Documents and Settings\Matt\My Documents. Вы можете изменить имя файла, щелкнув кнопкой мыши в поле **Имя файла** (File Name), и выбрать другую папку для размещения, щелкнув кнопкой мыши пиктограмму папки

Замечание

ОС Windows может скрывать расширения файлов, это зависит от настроек вашего компьютера. Вместо имени файла БД Access MyScandalousWedding.accdb в программе просмотра, такой как Проводник (Windows Explorer), вы можете увидеть только имя MyScandaiousWedding (без завершающей части .accdb). В этом случае вы все же можете определить тип файла, посмотрев на его пиктограмму. Если рядом с именем файла видна маленькая пиктограмма Access (в виде ключа), это свидетельство того, что вы смотрите на БД программы Access. Если же вы видите что-то другое (например, крошечную палитру с красками), вам придется соображать, файл какого типа перед вами.

3. Выберите папку.

Как и в остальных программах пакета Office, в Access предполагается, что вы хотите хранить все созданные вами файлы в личной папке Мои документы (My Documents). Если это не так, щелкните кнопкой мыши пиктограмму папки для вывода на экран диалогового окна **Файл новой базы данных** (File New Database), перейдите в нужную папку (рис. 1.3) и щелкните мышью кнопку ОК.

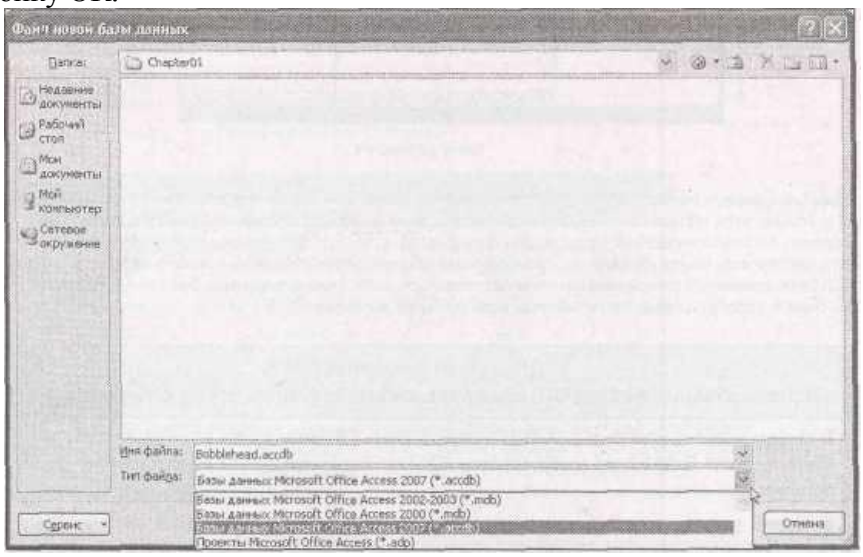


Рис. 1.3. Диалоговое окно **Файл новой базы данных** позволит вам выбрать место для хранения файла новой БД Access. Оно также дает возможность сохранить вашу БД в формате, применявшемся в предыдущих версиях программы Access (mdb). Для этого нужно выбрать в поле **Тип файла** (Save as type) вариант формата 2000 или 2002-2003- Если вы работаете под управлением ОС Windows Vista, то заметите, что у диалогового окна **Файл новой базы данных** совсем другой внешний вид,

но те же самые параметры

4. Щелкните мышью кнопку **Создать** (Create) (в правом нижнем углу окна Access).

Программа создаст файл вашей БД и затем выведет на экран лист данных, на котором вы сможете создать свою первую таблицу БД.

Как только вы создаете или открываете БД, окно Access немного меняется. В верхней части экрана появляется выразительная панель инструментов (лента), а слева - область переходов (navigation pane). В данный момент вы находитесь в центре управления, здесь вы будете выполнять все задачи, связанные с вашей БД (как показано на рис. 1.4).

Во *введении* даны базовые сведения **о** принципах работы ленты инструментов. (Для получения более подробной информации *см. разд. "Лента" во введении.)* Но сначала рассмотрим, как вы сможете воспользоваться вашей новейшей пустой БД, вставив в нее таблицу.

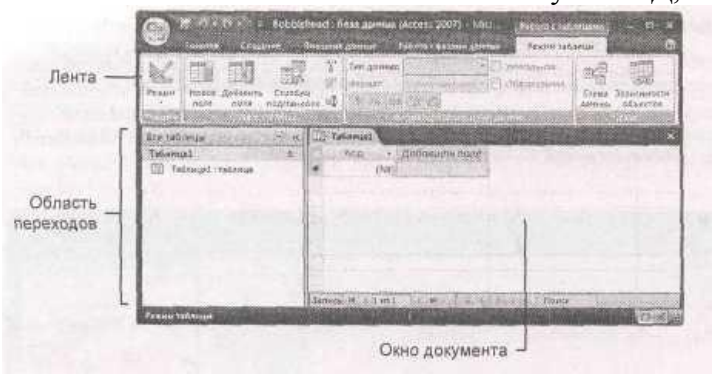


Рис. 1.4. Слева, в области переходов, отображаются различные элементы (или объекты) вашей БД. С помощью этой области вы сможете перейти от списка изделий к списку заказчиков и вернуться обратно. На ленте в верхней части экрана сгруппированы вместе все команды программы Access. Эта лента и есть центр управления, позволяющий вам выполнять различные задачи обработки вашей БД. Окно документа в середине занимает оставшуюся часть окна программы. Это окно, в котором вы будете работать, проектируя таблицы и заполняя их данными

Для тех, кто понимает.

Использование Access БД, созданных в более ранних версиях программы

В старых версиях программы Access не применяется формат accdb. Если бы попытаетесь открыть файл Bobblehead.accdb в Access 2003, то получите пустое окно и сообщение об ошибке.

В более ранних версиях программы Access используется файловый формат mdb (который обозначает БД Microsoft). Несмотря на то, что Access 2007 с успехом применяет файлы форматов accdb и mdb, предыдущие версии программы распознают только формат mdb. (И чтобы немного разнообразить жизнь, учтите, что у формата mdb на самом деле три версии: действующая (really) версия - действующий старый исходный формат, переоснащенная (retooled) версия, появившаяся в Access 2000, и еще раз улучшенная (improved-yet-again) версия, введенная корпорацией Microsoft в Access 2002 и повторно использованная в Access 2003.) Это вы должны знать, выбирая подходящий формат для ваших новых БД. Если вам не нужно беспокоиться о совместимости, лучше выбрать формат accdb, поскольку он обладает самой высокой производительностью и новыми дополнительными свойствами. Но если придется обрабатывать БД в других версиях программы Access, пропустите новое детище в списке типов и вместо этого положитесь на старый добрый формат mdb.

Для создания в Access 2007 файла БД со старым форматом mdb используйте поле со списком **Тип файла,** показанное на рис. 1.3. Вы можете выбрать формат файла версии Access 2002-2003 или более старый версии программы Access 2000. (Если вы твердо решили двигаться вспять в дальнейшем, скажем, к формату Access 95, то ваш лучший выбор - машина времени.)

1.2.2. *Что такое таблицы*

Таблицы - это информационные контейнеры. В любой БД должна быть хотя бы одна таблица, без нее вам негде хранить данные. В простой БД, такой как Bobblehead, достаточно одной таблицы (которую мы назовем Dolls). Но если окажется, что вы хотите сохранить несколько списков связанной информации, вам потребуется несколько таблиц. В БД BigBudgetWedding-accdb (Пышная свадьба) вы, возможно, захотите учесть гостей, приглашенных на вашу свадьбу, подарки, которые

вы попросили, и трофеи, которые вы на самом деле получили. В *главе 5* вы найдете достаточно примеров БД, использующих множественные таблицы.

На рис. 1.5 показан пример простой таблицы.

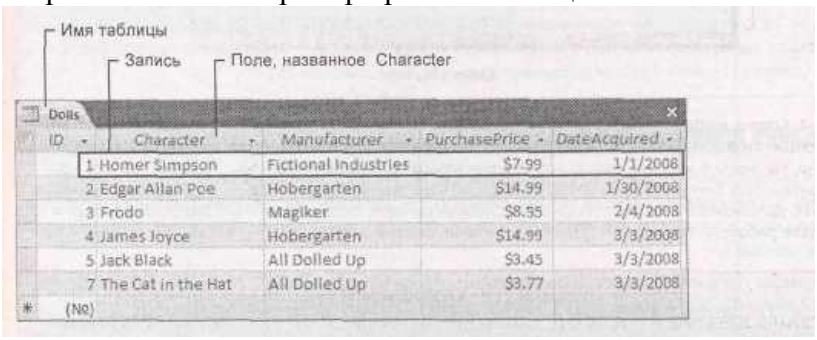


Рис. 1.5. В таблице каждая запись занимает отдельную строку. Каждое поле представлено в отдельном столбце. В этой таблице видно, что вставлены сведения о шести куклах-болванчиках.

Информация о каждой кукле хранится в пяти полях (ID (код), **Character** (персонаж), **Manufacturer** (изготовитель), **PurchasePrice** (покупная цена) и DateAcqutred (дата приобретения))

Прежде чем вы начнете конструировать таблицу, следует усвоить несколько очень важных правил,

■ Таблица - это всего лишь группа *записей,* Запись - это набор данных об одном предмете. В таблице **Dolls,** например, в каждой записи представлены данные об одной кукле-болванчике. В таблице **Family** (семья) каждая запись содержала бы сведения об отдельном родственнике. В таблице **Products** (товары) каждая запись представляла бы отдельный товар, предназначенный для продажи. Идея понятна.

■ Каждая запись состоит из *полей.* В каждом поле хранится конкретный фрагмент данных. Например, в таблице **Dolls** одно поле содержит имя прототипа куклы, другое поле - цену, а третье - дату покупки куклы и т. д.

■ У таблиц четкая структура. Другими словами, вы не можете неточно выполнять правила. Если вы создали четыре поля, у каждой записи должно быть четыре поля (хотя допускается отставлять некоторые поля пустыми, если они не используются).

На профессиональном уровне.

Проектирование БД для начинающих

Многие гуру БД полагают, что прежде чем запускать программу Access, вы должны точно определить с помощью мозгового штурма, какую информацию хотите хранить.

Вот как это делается. Сначала определите тип списка, который вы хотите поставить в конец следующего предложения "Мне нужен список..." (Пример: "Мне нужен список всех кукол-болванчиков, хранящихся в моем подвале.")

Затем бегло набросайте на листе бумаги все виды данных, которые у вас должны быть. Некоторые детали очевидны. Например, для коллекции кукол-болванчиков, возможно, вам потребуется хранить имя куклы, ее цену и дату ее покупки. Другие детали, такие как год ее производства, компания-изготовитель и краткое описание ее появления или состояния, могут потребовать некоторых раздумий.

После того как этот процесс завершен и определены все важные сведения, необходимые вам, можно приступать к созданию соответствующей таблицы в программе Access. Пример с куклами-болванчиками демонстрирует важный компонент проектирования БД: сначала вы планируете БД, а затем создаете ее с помощью программы Access. В *главе 5* вы узнаете гораздо больше о планировании более сложных БД.

1.2.3. *Создание простой таблицы*

Когда вы впервые создаете БД, она почти пуста. Но для того, чтобы вы могли начать, программа Access создает первый объект вашей БД - таблицу, названную **Таблица1,** Проблема заключается в том, что эта таблица появляется на свет пустой, без определенных полей (и без данных).

Если вы следовали шагам, описывающим процесс создания БД *(см. разд. "Создание новой базы данных" ранее в этой главе),* сейчас у вас на экране *лист данных* (см. рис. 1.5), в который вы будете вводить данные вашей таблицы. Вы должны настроить эту таблицу в соответствии с вашими потребностями.

Существуют два способа настройки таблицы.

- **Конструктор** (Design view) позволит вам точно определить все параметры таблицы до того, как вы начнете ею пользоваться. Почти все профессиональные разработчики БД предпочитают этот режим, и вы начнете его применять в *главе 2.*
- В **Режиме таблицы** (Datasheet view) вы вводите данные в таблицу. Этот режим также позволяет сконструировать таблицу на лету, когда вы вводите новую информацию.

В данной главе вы будете применять этот способ.

Следующие действия покажут вам, как в **Режиме таблицы** превратить новую пустую таблицу (такую как **Таблица!)** в таблицу **Dolls.**

1. Для определения таблицы вам нужно вставить первую запись.

В данном случае это значит мысленно выбрать куклу-болванчик для включения в список. В этом примере вы используете модель ловкача Гомера Симпсона (Homer Simpson).

Замечание

Не важно, сведения о какой кукле вы введете первыми. Таблицы Access *не отсортированы,* т. е. у них нет внутренней упорядоченности. Но в дальнейшем вы сможете отсортировать их любым способом, который потребуется вам для извлечения информации.

2. В столбце таблицы **Добавить поле** (Add New Field) введите первую порцию данных для формирования записи (рис. 1.6).

Благодаря простому анализу, проведенному вами ранее *(см. примечание "На профессиональном уровне. Проектирование БД для начинающих" в конце предыдущего раздела),* вы знаете, что вам нужно ввести четыре поля данных для каждой куклы. Для куклы Гомера Симпсона они следующие: "Homer Simpson" (имя), "Fictional Industries" (компания-изготовитель), $7.99 (цена) и текущая дата (дата покупки). Несмотря на то, что можно начать с любого поля, имеет смысл начать с имени, которое служит важной идентифицирующей характеристикой.

Рис. 1.6. Для заполнения первой записи начните с ввода данных в первое поле (например, имени куклы "Homer Simpson"). Затем нажмите клавишу <Tab> для перехода во второй столбец и введите вторую порцию данных. Пока не обращайте внимания на столбец Код - программа Access добавляет его к каждой таблице для идентификации ваших строк

3. Нажмите клавишу <Tab> для перехода в следующий столбец и повторите шаг 2.

Повторяйте шаги 2 и 3, пока не вставите все необходимые поля. Следите за тем, чтобы каждая отдельная порция данных помещалась в свой столбец.

На профессиональном уровне.

Вставка больших значений в узкие столбцы

Столбец может вмещать абзацы информации целиком, поэтому, начав набирать текст, вы можете обнаружить, что вышли за пределы видимой области столбца. Это не проблема (помимо всего прочего вы можете просто прокрутить ваше поле во время его редактирования), но

подобная ситуация может вызывать раздражение. Большинство пользователей предпочитает видеть всю информацию в столбце.

К счастью, нет нужды молча страдать из-за узких столбцов. Для того чтобы расширить столбец, поместите указатель мыши на правый край заголовка столбца. (Для того чтобы раздвинуть столбец с именем **Поле1** (Field1), поместите указатель мыши на правый край прямоугольника **Поле1.)** Затем перетащите столбец вправо, чтобы увеличить его до нужного вам размера.

Если вы немного нетерпеливы, есть сокращенная команда. Дважды щелкните кнопкой мыши правый край столбца для получения размера, вмещающего всю информацию в нем (при этом столбец не может выйти за пределы окна программы Access). Таким образом, вы автоматически получаете все необходимое вам пространство.

Если хотите быть профессиональнее, включите знак доллара ($) при вводе цены и убедитесь в том, что дата введена в распознаваемом формате для дат (например, January 1, 2008 (1 января 2008) или 01-01-2008). Эти особые метки сообщают программе Access о типе данных, помещаемых в столбец. (В *главе 2* вы узнаете, как в полной мере управлять типом данных в каждом столбце и избежать возможных недоразумений.) На рис. 1.7 показана полностью введенная запись.

Примечание

Если вы нажмете клавишу <Tab> без ввода какой-либо информации, то перейдете на следующую строку и начнете ввод новой записи. Если вы допустили ошибку, с помощью клавиш со стрелками можно вернуться назад.

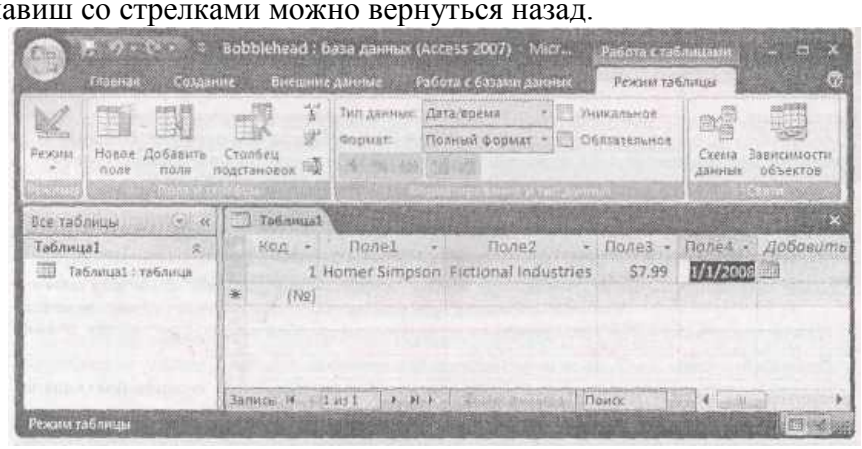


Рис. 1.7. В примере все еще остается единственная проблема: при вводе новой записи программа Access создает абсолютно неподходящие имена полей. Их варианты вы увидите в верхней части всех столбцов (это имена Поле1, Поле2, Поле3 и т. д.). Недостаток этих бессмысленных имен состоит в том, что они могут привести к вставке порции данных в неподходящее место. Вы можете очень легко вставить цену покупки в столбец дат. Для предотвращения таких оплошностей следует задать более информативные имена полей

4. И сейчас самое время их исправить. Щелкните дважды кнопкой мыши заголовок первого столбца (например, **Поле1).**

Имя поля переключится в *режим редактирования.*

5. Введите новое имя и нажмите клавишу <Enter>. Вернитесь к шагу 4.

Повторяйте этот процесс до тех пор, пока не подчистите все имена полей. Подходящие для данного примера имена - **Character** (персонаж), **Manufacturer** (изготовитель),

PurchasePrice (покупная цена) и **DateAcquired** (дата приобретения). Этот процесс изображен на рис. 1.8.

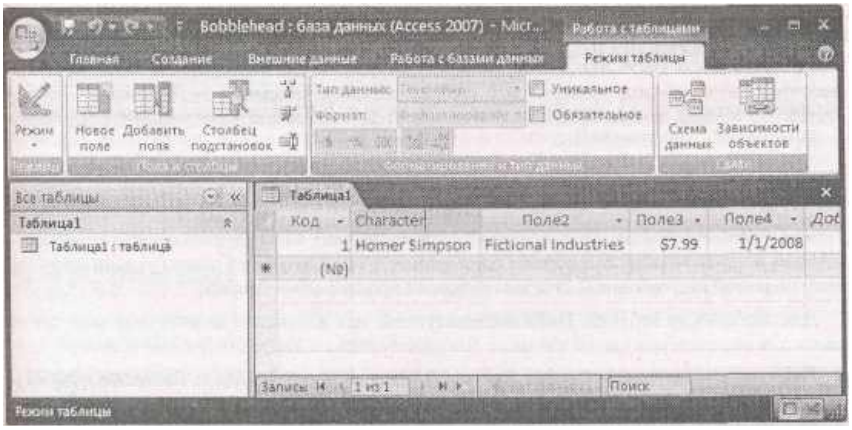


Рис. 1.8. Для выбора подходящих имен дважды щелкните кнопкой мыши заголовок столбца. Далее введите настоящее имя поля и нажмите клавишу <Enter>. В *разд. "Правило 1. Выбирайте подходящие имена полей" главы 2* приведена более подробная информация об именовании полей, но пока будем придерживаться коротких текстовых заголовков без пробелов

Подсказка

Не тратьте слишком много времени на тонкую настройку вашей таблицы. Вы всегда сможете переименовать поля впоследствии и даже вставить совершенно новые поля. (У вас также есть возможность удалить существующие поля, но при этом неизбежно будут удалены все данные, хранящиеся в этом поле.)

6. Выберите кнопку Office и команду Сохранить (Save) (или нажмите комбинацию клавиш <Ctrl>+<S>) для того, чтобы сохранить вашу таблицу.

Программа Access попросит вас ввести имя таблицы (рис. 1.9).

7. Введите подходящее имя и щелкните мышью кнопку ОК.

Поздравляю! Теперь таблица - часть вашей БД.

Рис. 1.9. Хорошее имя для таблицы - это короткий текстовой заголовок без пробелов (**Dolls** в этом примере) **Примечание**

Нет технической необходимости сохранять таблицу немедленно. Программа Access напомнит вам об этом, когда вы закроете лист данных (щелкнув мышью кнопку x в правом верхнем углу окна документа) или при выходе из программы.

Как видите, создание простой таблицы в Access почти так же просто, как подготовка данных в программах Excel или Word. Если вам не терпится попытаться еще раз, вы можете создать другую таблицу в вашей БД, выбрав на ленте **Создание → Таблица** (Create → Table). Но прежде чем вы приступите к этому, стоит повнимательнее рассмотреть способы редактирования вашей таблицы.

1.2.4. *Редактирование таблицы*

Теперь у вас есть вполне действующая (хотя и простая) БД, содержащая одну таблицу, в которой, в свою очередь, содержится одна запись. Ваша следующая задача - заполнить таблицу полезной информацией. Это часто нудный процесс *ввода данных.*

Для заполнения таблицы **Dolls** воспользуйтесь тем же листом данных, который применялся для определения вашей таблицы. Вы можете выполнять три основные задачи.

Редактирование записи. Перейдите в нужное место листа данных (с помощью клавиш со стрелками или мыши) и затем введите замещающую информацию. Вы можете использовать режим редактирования, который будет описан в следующем разделе.

■ *Вставку повой записи.* Переместитесь с нижнюю часть таблицы к строке с расположенной слева звездочкой (*). Эта строка на самом деле не существует до тех пор, пока вы не начнете

вводить в нее данные. В этот момент программа Access создает строку и перемещает звездочку в строку, расположенную непосредственно под данной. Этот процесс можно повторять бесконечно для того, чтобы ввести столько строк, сколько вам нужно (программа Access может обрабатывать миллионы строк).

■ *Удаление записи.* Существует несколько способов удаления записи, но легче всего щелкнуть правой кнопкой мыши в поле слева от записи и выбрать команду **Удалить запись** (Delete Record). Программа Access попросит вас подтвердить желание удалить выбранную запись, поскольку позже вы не сможете отменить это действие.

Для тех. Кто понимает.

Если сомневаетесь, не удаляйте

Наиболее опытные проектировщики редко удаляют записи из своих БД. Каждая капля информации важна.

Представьте себе, что у вас есть БД с перечислением товаров, которые продает по почте компания, изготовляющая оригами. Возможно, вы считаете, что имеет смысл удалить товары после того, как они сняты с производства и больше не могут быть заказаны. Но выясняется, что стоит хранить эти записи о старых товарах под рукой. Например, вам может понадобиться определить, какие категории товаров лучше всего продавались в прошлом году. Или производитель выпускает распоряжение об изъятии бумаги с добавлением асбеста и вам нужно найти всех, кто ее заказывал. Для выполнения любой из этих задач вам следует сохранять записи о ваших товарах. Это правило о сохранении всевозможной информации применимо ко всем видам БД. Допустим, что вы отслеживаете прием студентов в кулинарную академию высшей категории.

Если группа закончила курс обучения, вы не можете просто удалить запись о группе. Вам могут понадобиться сведения о том, есть ли у студента необходимая подготовка для другого курса, у каких преподавателей он учился и т. д.

Это же справедливо и для сотрудников, которые уволены или переведены на более высокую должность, проданных вещей, которые принадлежали вам ранее, и т. д. Все эти сведения вам нужны (и, возможно, их следует хранить вечно).

В большинстве случаев вы вставляете в свою таблицу дополнительные поля, помогающие отделить старые данные от новых. Например, в таблице Products (изделия) можно создать поле Discontinued (снятые с производства), обозначающее изделия, которых больше нет в наличии. Позже, создавая форму размещения заказов, вы сможете не учитывать эти изделия.

Режим редактирования

Возможно, вы проведете много времени, работая с листом данных. Итак, начнем. Знание некоторых деталей сможет облегчить вам жизнь.

Как вы уже знаете, перемещаться от поля к полю и от строки к строке можно с помощью клавиш со стрелками. Но при редактировании значения могут возникнуть некоторые трудности. Как только вы начнете набирать информацию, программа Access сотрет имеющиеся данные. Для изменения поведения программы следует перейти в *режим редактирования,* нажав клавишу <F2>. В режиме редактирования ваш набор не уничтожает уже имеющуюся в поле информацию. Вместо этого вы можете изменять ее или добавлять новые данные. Для выхода из режима редактирования снова нажмите клавишу <F2>. На рис. 1.10 крупным планом показана разница режимов.

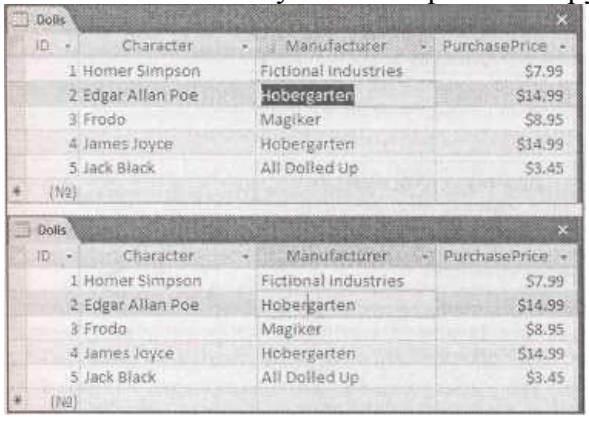


Рис. 1.10. *Вверху:* обычный режим. Если вы сейчас начнете ввод данных, вы тут же

сотрете существующий текст ("Hobergarten"). Суть в том, что текст в выбранном поле - это важная информация, которую вы намереваетесь удалить. *Внизу:* режим редактирования.

Курсор указывает текущую позицию в выбранном поле. Если сейчас начать набор, текст вставится между "Hober" и "garten".

Подсказка

Входить в режим редактирования и выходить из него можно также двойным щелчком кнопкой мыши в ячейке таблицы.

Режим редактирования также влияет на поведение клавиш со стрелками. В этом режиме они перемещают курсор в текущем поле. Например, для перехода в соседнюю ячейку вам придется в любом случае переместить курсор к концу текущего текста и затем снова нажать клавишу со стрелкой вправо (<→>). А в обычном режиме клавиши со стрелками всегда перемещают вас от ячейки к ячейке.

Комбинации клавиш

Опытные пользователи знают, что самый быстрый способ выполнения команды - использование замысловатых комбинаций клавиш, таких как <Ctrl>+<Alt>+<Shift>+<*>. Такие сочетания не всегда легко запомнить, но вам поможет пара приведенных далее таблиц. В табл. 1.1 перечислены некоторые клавиши, помогающие быстро перемещаться по листу данных.

Таблица 1.1. Клавиши для перемещения по листу данных

Клавиша	*Действие*
<Tab> (или <Enter>)	Перемещает курсор на одно поле вправо или вниз при достижении конца таблицы. Эта клавиша также отключает режим редактирования, если он был включен
<Shift>+<Tab>	Перемещает курсор на одно поле влево или вверх при достижении конца таблицы. Это сочетание клавиш также отключает режим редактирования
<→>	Перемещает курсор на одно поле вправо (в обычном режиме) или вниз при достижении конца таблицы, В режиме редактирования эта клавиша перемещает курсор вдоль текста в текущем поле
<←>	Перемещает курсор на одно поле влево (в обычном режиме) или вверх при достижении конца таблицы. В режиме редактирования эта клавиша перемещает курсор вдоль текста в текущем поле
<↑>	Перемещает курсор вверх на одну строку (пока вы не достигли начала таблицы). Эта клавиша также отключает режим редактирования
<↓>	Перемещает курсор вниз на одну строку (или в позицию "новой строки", если вы достигли конца таблицы). Эта клавиша также отключает режим редактирования (Edit)
<Home>	Перемещает курсор в первое поле текущей строки. Эта клавиша перемещает курсор к началу текущего поля, если вы находитесь в режиме редактирования
<End>	Перемещает курсор в последнее поле текущей строки. Эта клавиша перемещает курсор к концу текущего поля, если вы находитесь в режиме редактирования
<Page Down>	Перемещает курсор вниз на один полный экран (предполагается, что у вас большая таблица данных, которая не помещается целиком в окне программы Access). Эта клавиша также отключает режим редактирования
<Page Up>	Перемещает курсор вверх на один полный экран. Эта клавиша также отключает режим редактирования
<Ctrl>+<Home>	Перемещает курсор в первое поле первой строки. Это сочетание клавиш не делает ничего, если вы находитесь в режиме редактирования
<Ctrl>+<End>	Перемещает курсор в последнее поле последней строки. Это сочетание клавиш не делает ничего, если вы находитесь в режиме редактирования

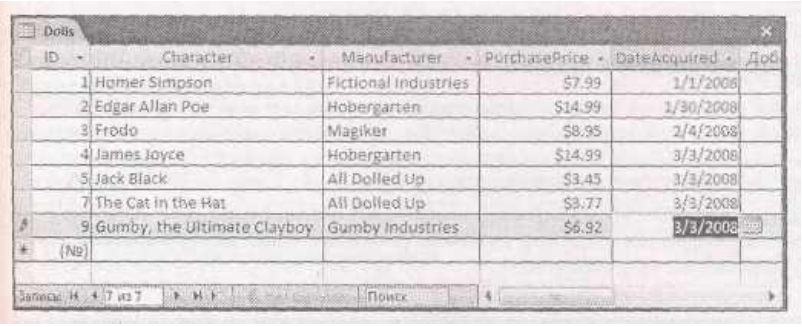


Рис. 1.11. Пользователь программы Access побывал на торгах Web-сайта eBay и ему нужно сразу добавить несколько кукол. С помощью одновременного нажатия сочетания клавиш <Ctrl>+<"> дату покупки из предыдущей записи можно вставить в текущее поле

В табл. 1.2 перечислены клавиши, облегчающие редактирование записей.

Таблица 1.2. Клавиши для редактирования записей

Клавиша	*Результат*
<Esc>	Отменяет любые изменения, произведенные вами в текущем поле. Эта клавиша действует только в режиме редактирования. После перехода в следующую ячейку изменения вносятся в запись. (Для дополнительного управления отменой изменений попробуйте команду Отменить (Undo), описанную далее.)
<Ctrl>+<Z>	Отменяет последнее изменение. К сожалению, команда Отменить в программе Access гораздо слабее одноименной команды в других программах пакета Office. Например, Access разрешает отменить только одно изменение, и если вы закроете лист данных, то даже этого не сможете сделать. Вы можете воспользоваться командой Отменить для уничтожения новой записи только сразу после ее вставки, но не сможете с помощью этой команды отменить операцию удаления
<Ctrl>+<">	Копирует значение из поля, находящегося непосредственно над текущим. Этот прием полезен, если нужно ввести группу записей с похожей информацией. На рис. 1.11 показан этот часто игнорируемый прием в действии
<Ctrl>+<;>	Вставляет текущую дату в текущее поле. Формат даты зависит от настроек компьютера, но, скорее всего, вы увидите нечто похожее на 24-12-2007. Вы узнаете больше о работе с датами в программе Access в разд. "Дата/время" главы 2
<Ctrl>+<Alt> +< Пробел>	Вставляет в поле значение по умолчанию. Вы узнаете, как задавать значение по умолчанию, в разд. "Задание значений по умолчанию" главы 4

Вырезать, Копировать и Вставить

Access, как практически любая Windows-программа, позволяет вырезать порции данных из одного места **и** вставлять их в другое место. Этот трюк легко применять с помощью только трех комбинаций клавиш: <Ctrl>+<C> - **Копировать,** <Ctrl>+<X> - **Вырезать** (аналог команды **Копировать,** но исходное содержимое удаляется) и <Ctrl>+<V> - **Вставить.** Находясь в режиме редактирования, можно использовать эти клавиши для копирования любого выделенного фрагмента. Если вы в обычном режиме, операция копирования или вырезания захватывает все содержимое поля.

Малоизвестная или недооцененная возможность.

Копирование записи целиком за один шаг

Обычно вы копируете и вставляете небольшие порции или фрагменты данных. Но у программы Access есть малоизвестная функциональная возможность, позволяющая копировать запись целиком. Для того чтобы воспользоваться ею, выполните следующие действия:

1. Щелкните кнопкой мыши поле слева от записи, которую хотите скопировать.
2. Это позволит выделить запись. (Если вы хотите скопировать группу смежных записей, нажмите клавишу <Shift> и затем смещайте указатель мыши вверх или вниз до тех пор, пока все

нужные записи не будут выделены.)

3. Щелкните правой кнопкой мыши и выберите команду **Копировать** (Copy).

4. Она скопирует содержимое в буфер.

5. Прокрутите таблицу от начала к концу пока не увидите маркер новой строки (звездочку).

6. Щелкните правой кнопкой мыши слева от маркера новой строки и выберите команду **Вставить** (Paste).

Точная копия появится немедленно. (Следует признаться, что один фрагмент копии не соответствует оригиналу. Программа Access обновляет столбец **Код** (ID) в вашей вставленной записи, присваивая ему новый номер. Делается это потому, что у каждой записи должен быть уникальный номер. В *разд. "Первичный ключ " главы 2* вы узнаете, почему.)

1.3. *Сохранение и открытие БД Access*

В отличие от других программ, Access не требует от вас сохранения результатов вашей работы. Она автоматически сохраняет все внесенные вами изменения.

Когда вы создаете новую БД *(см. разд. "Создание новой базы данных "ранее в этой главе),* программа Access сохраняет файл БД. Когда вы добавляете в БД таблицу или другой объект, Access снова сохраняет БД. И когда вы вводите новые данные или редактируете существующие, Access сохраняет БД почти мгновенно.

Этот процесс автосохранения протекает за кадром, и вы, возможно, ничего даже не заметите. Но не беспокойтесь о том, что при выходе программа Access никогда не напоминает вам о необходимости сохранения изменений, *все изменения были сохранены в тот момент, когда вы их вносили.*

1.3.1. *Создание резервных копий*

Автоматическое сохранение может стать проблемой, если внесены ошибочные изменения. Если вы быстро спохватились, то можете воспользоваться командой **Отменить** для отмены последнего изменения (рис. 1,12). К несчастью, эта команда отменяет только *самую последнюю корректировку,* поэтому она бесполезна, если вы отредактировали группу записей, а затем обнаружили проблему. Команда **Отменить** не поможет и в том случае, если вы закрыли вашу таблицу, а затем повторно ее открыли.

Рис. 1.12. Команда **Отменить** выводится на **Панели быстрого доступа** в верхней левой части окна программы Access (дугообразная стрелка), поэтому она всегда доступна.

По этой причине неплохо почаще делать резервные копии БД. Для создания резервной копии необходимо просто скопировать файл БД в другую папку или создать копию с другим именем (например, Bobblehead_Backupl.accdb). Эту задачу можно выполнить с помощью Проводника (Windows Explorer), но программа Access предлагает даже более легкий вариант. Выберите кнопку **Office → Управление → Резервная копия базы данных** (Office → Manage → Back Up Database), и Access создаст для вас копию вашей БД в том месте, которое вы укажете (рис. 1.13).

Примечание

Помнить о резервном копировании БД - *ваша* задача. В программе Access нет операции автоматического резервного копирования, но вы можете использовать другое средство для периодического копирования файла вашей БД. Примером может служить программа Диспетчер

задач Windows (Task Scheduler), входящая в состав большинства версий ОС Windows. (Вы можете прочесть краткое, но полезное руководство по применению Диспетчера задач на странице **www.pctechguide.com/tutoriais/SchedufeTasks.htm.)**

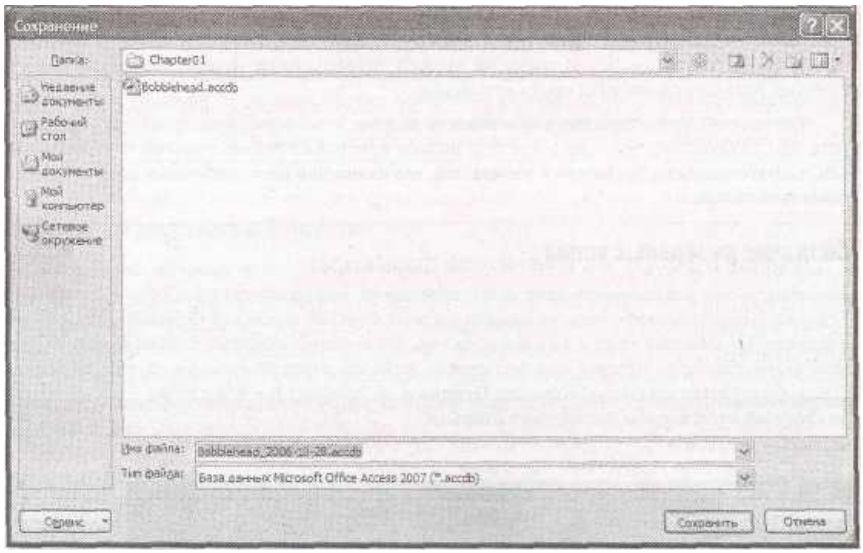


Рис. 1.13. После выбора последовательности кнопка **Office → Управление → Резервная копия базы данных** Access заполняет поле предлагаемым именем файла, в которое включена текущая дата. Таким образом, если у вас есть несколько резервных копий, вы можете выбрать ту, которая вам нужна

Малоизвестная или недооцененная возможность.

Сжатие БД

Когда вы добавляете данные в БД, программа Access не всегда упаковывает их с максимальной плотностью. Она больше озабочена максимально быстрым извлечением информации из БД и вставкой ее в БД.

После того как вы поработаете с БД некоторое время, может оказаться, что ее размер раздувается как рыба недельной давности, полежавшая на солнце. Если вы хотите сократить размер вашей БД, можно применить операцию, называемую *сжатием.* Для этого выберите последовательность: кнопка **Office → Управление → Сжать и восстановить базу данных** (Office → Manage → Compact and Repair Database). Объем освободившегося дискового пространства может быть разным, но нельзя считать чем-то необычным сжатие БД размером 10 Мбайт до четверти первоначального размера.

Единственная проблема, связанная с операцией сжатия, - своевременное ее выполнение. Если вы хотите всегда сохранять минимальный размер у ваших БД, следует включить режим, заставляющий программу Access сжимать текущую БД при каждом ее закрытии.

Далее перечислены необходимые действия.

1. Откройте БД, которую вы хотите автоматически сжимать.
2. Выберите последовательность: кнопка **Office → Параметры** Access (Office → Access Options). Программа Access откроет окно **Параметры** Access, в котором вы можете изменить ряд конфигурационных параметров.
3. В списке слева выберите **Текущая база данных** (Current Database).
4. В правой части страницы установите флажок **Сжимать при закрытии** (Compact on Close).
5. Щелкните мышью кнопку ОК для сохранения изменений.

Вы можете установить флажок **Сжимать при закрытии** для всех выбранных вами БД. Но помните, что он не устанавливается автоматически при первоначальном создании новой БД.

1.3.2. *Сохранение БД с другим именем или форматом*

Если вы решили сохранить вашу БД с другим именем в другом месте или в формате более ранней версии программы Access, вы можете воспользоваться заслужившей доверие командой **Сохранить как** (Save As).

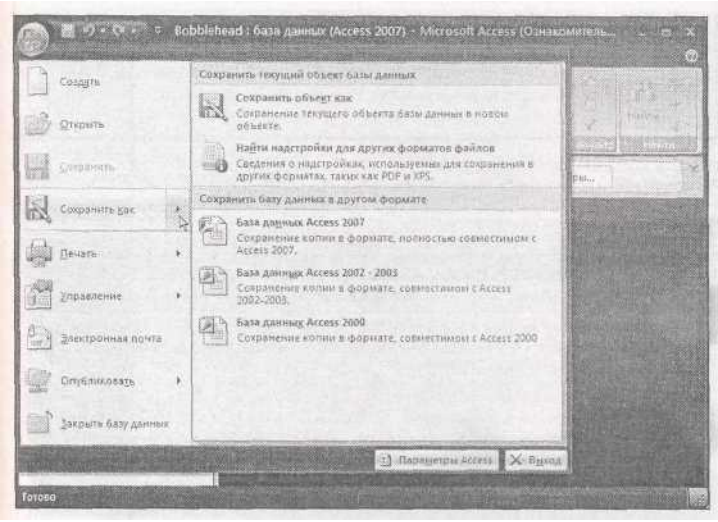


Рис. 1.14. Щелкните направленную вправо стрелку, расположенную рядом с командой меню **Сохранить как,** для того чтобы увидеть это подменю и возможные варианты выбора.

(Простой щелчок по команде **Сохранить как** приводит к выполнению стандартного варианта команды, который сохраняет копию выбранного в данный момент объекта БД, а не всей вашей БД.) Далее выберите один из вариантов, находящихся под заголовком **Сохранить базу данных в другом формате**

Начните с выбора: кнопка **Office → Сохранить как** (Office → Save As), а затем используйте один из вариантов, представленных на рис. 1.14. Помните о том, что после создания файла новой БД программа Access только его и продолжает использовать. Другими словами, когда вы создаете таблицу или редактируете какие-то данные, Access обновляет новый файл. (Если вы хотите вернуться к старому файлу, то должны открыть его в программе Access или повторно использовать команду **Сохранить как.**)

1.3.3. *Открытие БД*

После того как БД создана, ее легко открыть. Можно применить один из следующих способов.

■ Дважды щелкните кнопкой мыши файл БД. Добраться до файла можно с помощью окна **Мой компьютер** (My Computer), программы Проводник (Windows Explorer) или положить его на рабочий стол. Напоминаю, у файлов БД Access расширение accdb или mdb.

■ Запустите программу Access и найдите вашу БД в разделе **Открыть последнюю базу данных** справа на странице **Приступая к работе с Microsoft Office Access.** (Этот список можно увидеть с помощью меню **Office,** как показано на рис. 1.15.)

■ Запустите Access, выберите последовательность: кнопка **Office → Открыть** (Office → Open), а затем найдите файл вашей БД Access.

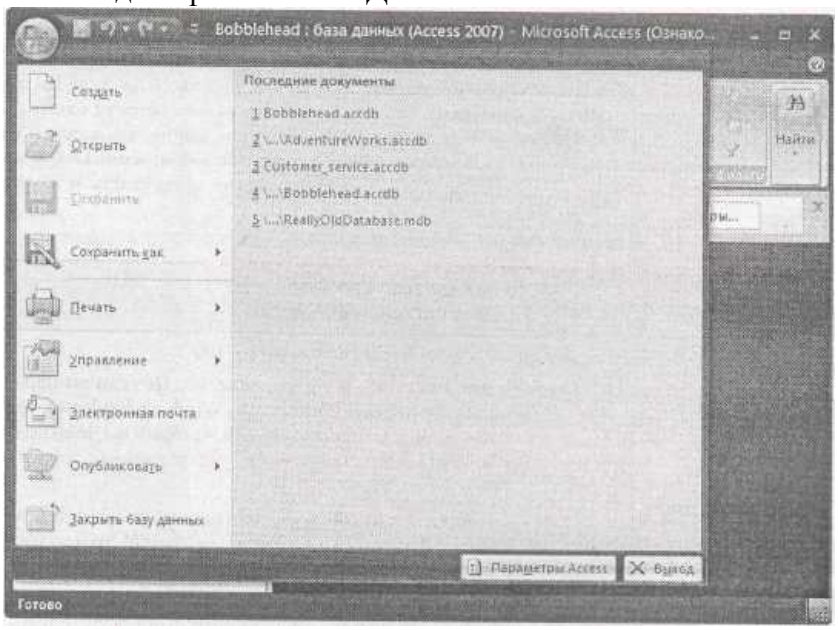


Рис. 1.15. В меню **Office** список **Последние документы** такой же, как список файлов в разделе **Открыть последнюю базу данных** на странице **Приступая к работе с Microsoft Office Access.** Но если у вас уже есть открытая БД, список **Последние документы** использовать удобнее, поскольку не надо возвращаться на страницу **Приступая к работе с Microsoft Office Access**

Когда вы откроете БД, то заметите нечто немного странное. Программа Access выведет панель сообщения с устрашающе звучащим предупреждением системы безопасности (рис. 1.16).

Рис. 1.16. Это предупреждение системы безопасности сообщает вам о том, что Access не доверяет вашей БД - другими словами, ваш файл открыт в специальном безопасном режиме, который препятствует выполнению вашей БД любых рискованных операций

Подобное предупреждение слегка сбивает с толку, потому что прямо сейчас ваша БД даже не пытается делать что-либо рискованное. Однако после того как вы начнете создавать БД с программируемыми процедурами (как описано в *части V)* или применять запросы на изменения (action queries) *(см. главу 8),* все будет совсем иначе. Тогда вам; возможно, захочется перенастроить программу Access так, чтобы она распознавала ваши файлы и научилась быть немного более доверчивой.

Часто задаваемый вопрос.
У какого файла расширение laccdb?
Я вижу дополнительный файл с расширением laccdb. Что происходит?

Пока вы познакомились с типом файла, имеющим расширение accdb. Но если вы привыкли просматривать папки с помощью программы Проводник (Windows Explorer), то могли заметить еще один файл с загадочным расширением laccdb, который вы не создавали. Вместе с файлом Bobblehead.accdb появляется таинственный файл Bobblehead.laccdb.

Программа Access создает файл с расширением laccdb, когда вы открываете файл БД, и закрывает его, когда вы закрываете вашу БД, поэтому вы можете увидеть его, только пока вы (или кто-то другой) просматриваете БД.

Access применяет файл с расширением laccdb для отслеживания пользователей, в данный момент работающих с БД. Символ "1" означает блокировку, этот файл гарантирует, что в случае одновременного использования БД несколькими людьми они не смогут изменять одну и ту же запись в одно и то же время (что может вызвать всевозможные проблемы).

Вы узнаете о том, как программа Access работает с многочисленными пользователями в *главе 18,* а сейчас можно без каких-либо опасений игнорировать файл с расширением laccdb. Вам не нужно включать его в свои резервные копии.

Сейчас вы, вероятно, раздумываете над тем, что вам следует делать с полосой сообщения. Есть два варианта:

■ щелкните мышью значок x справа от полосы сообщения для того, чтобы убрать его с экрана (оно появится опять, когда вы в следующий раз откроете БД);

■ сообщите программе Access о том, что она может доверять вашим БД, указав доверенное

место - папку на вашем жестком диске, в которой вы храните файлы ваших БД. Вы узнаете, как задавать доверенное место *в разд. "Задание надежного расположения "главы 15.*

1.3.4. *Одновременное открытие нескольких БД*

Каждый раз, когда вы применяете последовательность кнопка **Office → Открыть** (Office → Open), Access закрывает текущую БД и затем открывает выбранную вами. Если вы хотите видеть несколько БД одновременно, вам нужно запустить сразу несколько копий программы Access. (Компьютерные фанаты называют подобное действие запуском нескольких *экземпляров* программы.)

Это сделать почти до неприличия легко. Если вы уже находитесь в открытой программе Access, дважды щелкните кнопкой мыши файл другой БД, для нее на панели задач появится второе окно Access. Вы также можете запустить второй (или третий, или четвертый...) экземпляр программы Access из меню **Пуск** и затем использовать последовательность **Office → Открыть** (Office → Open) для загрузки разных БД в каждом из них.

Практические занятия для опытных пользователей.
Изменение папки, которую Access использует для хранения БД

Программа Access всегда полагает, что вы хотите хранить БД в папке Мои документы (My Documents). Несмотря на то, что вы можете выбрать другое местоположение при каждом сохранении или открытии БД, если есть другая папка, к которой вам придется часто обращаться, имеет смысл сделать ее стандартным местом хранения БД. Вы можете настроить программу Access на применение этой папки буквально за несколько шагов:

Выберите последовательность кнопок **Office → Параметры Access** (Office → Access Options). На экране появится окно **Параметры Access**.

1. В списке слева выберите **Основные** (Popular).
2. На странице справа найдите заголовок **Создание базы данных** (Creating databases). Ниже вы обнаружите текстовое поле **Рабочий каталог** (Default database folder). Введите в него имя папки, которую вы хотите использовать (например, C:\MyDatabases), или укажите ее с помощью кнопки **Обзор** (Browse).
3. Завершив настройку, щелкните мышью кнопку ОК для сохранения внесенных изменений.

1.3.5. *Открытие БД, созданной в более старой версии Access*

Можно воспользоваться последовательностью кнопка Office → Открыть (Office → Open) для того, чтобы открыть *БД* Access, созданную кем-то в предыдущей версии программы Access *(см, в примечании "Для тех, кто понимает. Использование Access БД, созданных в более ранних версиях программы" в разд. "Создание новой базы данных" ранее в этой главе о разных форматах файлов программы Access).*

Программа Access обрабатывает старые файлы БД по-разному в зависимости от того, насколько они устарели. Вот как это делается.

■ Если вы открываете файл версии Access 2002-2003, то не получите никакого уведомления или предупреждения, Access сохраняет текущий формат, и вы можете вносить любые изменения.

Если вы открываете файл версии Access 2000, ваше плаванье также остается спокойным. Однако, если вы измените структуру БД, Access 2000 может не принять добавленные вами новые компоненты.

Если вы откроете устаревший файл Access (например, созданный в версиях Access 97, 95 или 2.0), программа Access спросит, хотите ли вы преобразовать БД или просто открыть ее (рис. 1.17).

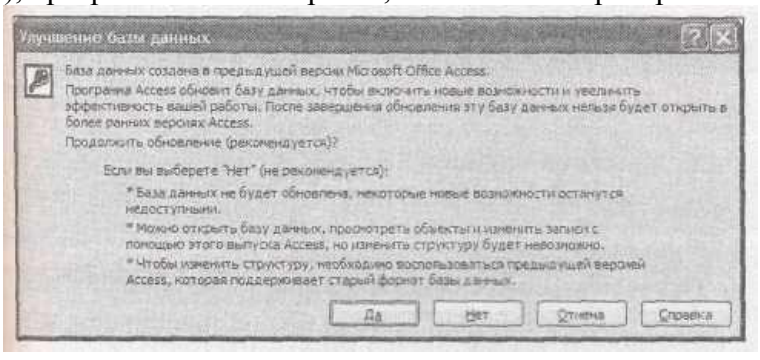


Рис. 1.17. Программа Access предлагает вам выбор, когда вы открываете файл БД, созданный в Access 97, 95 или 2.0. Если вы выбираете преобразование БД (щелкните мышью кнопку **Да**), Access копирует имеющийся файл БД в новый файл БД в формате Access 2002-2003. Затем вы можете редактировать эту копию обычным образом.

Если же выбираете открытие БД (щелкните мышью кнопку **Нет**), Access откроет исходный файл без создания копии. Вы все равно можете редактировать имеющиеся данные и вставлять новые, но будете лишены возможности изменять структуру БД

Совет

Вы всегда можете назвать текущий формат БД, посмотрев на текст в скобках, приведенный в заголовке окна Access. Если вы откроете файл в формате Access 2002-2003, в полосе заголовка можно прочесть: "Bobblehead: база данных (формат Access 2002-2003)" ("Bobblehead: Database (Access 2002-2003 file format)").

Если вы откроете БД Access "старой закваски", то заметите, что изменилось кое-что еще. Когда вы откроете таблицу, то не увидите окна с вкладками (подобными показанным на рис. 1.20). Вместо этого таблица откроется в обычном окне, которое может располагаться в любом месте в главном окне программы Access. Сначала этот вариант может показаться удачным, но лишь до тех пор, пока вы не откроете несколько таблиц одновременно. В этом случае вы столкнетесь с полной неразберихой, показанной на рис. 1.18.

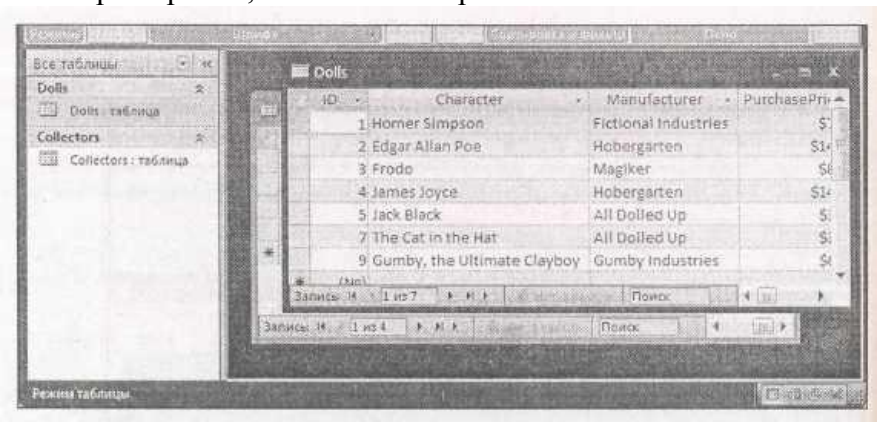


Рис. 1.18. В БД Access старого стиля разные окна могут перекрывать друг друга. Очень скоро таблица, которая вам нужна, будет погребена на дне кучи окон

Это недоброжелательное поведение задумано проектировщиками как уподобление более ранним версиям программы Access. Но не беспокойтесь - вы легко сможете вернуться к симпатичным вкладкам, даже если не преобразуете вашу БД в новый формат. Нужно задать всего лишь один конфигурационный параметр для вашей БД.

1. Выберите последовательность: кнопка **Office → Параметры Access** (Office → Access Options). На экране появится окно **Параметры Access.**
2. В списке слева выберите **Текущая база данных** (Current Database).
3. Под заголовком **Параметры приложений** (Application Options) найдите **Параметры окна документа** (Document Windows Options setting), где вы можете выбрать переключатель **Перекрывание окон** (Overlapping Windows) (стандарт Access 2003) или Вкладки (Tabbed Windows) (технологию будущего).
4. Щелкните мышью кнопку ОК.
5. Закройте и снова откройте вашу БД, чтобы новая установка подействовала.

Для того чтобы ощутить прикосновение прошлого, можно использовать эту же установку и заставить фирменную новую БД Access применять перекрывающиеся окна вместо вкладок.

1.3.6. *Создание еще одной БД*

Создание новой БД - теперь самая легкая задача. Нужно просто выбрать последовательность: кнопка **Office → Создать** (Office → New). Программа Access вернет вас на страницу

Приступая к работе с Microsoft Office Access (Getting Started with Microsoft Office Access), где вы сможете создать пустую БД, щелкнув мышью знакомую пиктограмму **Новая база данных** (Blank Database), как было описано ранее *(см. разд. "Создание новой базы данных" ранее в этой главе)*.

1.4. *Область переходов*

Сейчас самое время отступить на шаг и посмотреть на то, что вы сделали к настоящему моменту.

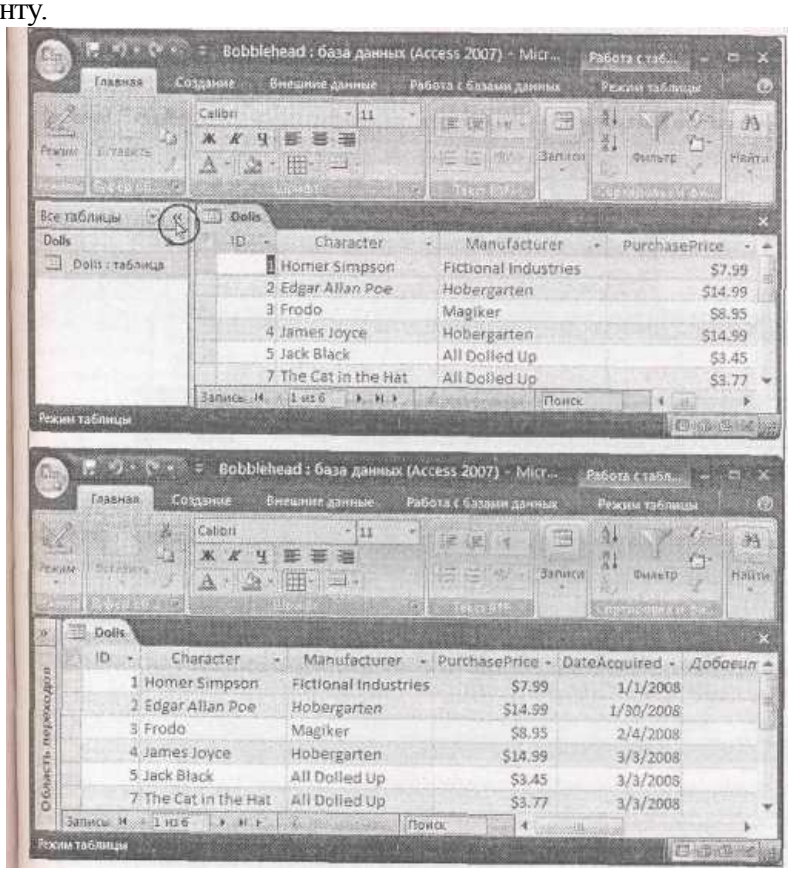


Рис. 1.19. Вам жаль пространства, занятого областью переходов? Щелкните мышью кнопку **Открыть/закрыть границу области переходов** в правом верхнем углу *(вверху), и* область переходов уйдет с дороги, освободив место для листа данных *(внизу)*. Для того чтобы вернуть отображение этой области, еще раз щелкните мышью ту же кнопку.

Вы создали БД Bobblehead и вставили один объект БД: таблицу, названную **Dolls.** Вы внесли в таблицу **Dolls** несколько записей. У вас нет причудливых окошек, отчетов и процедур поиска - всего того, что делает работу БД по-настоящему гладкой, но вы обладаете наиболее важной составляющей - организованными данными.

Одна задача все еще осталась нерешенной - способ управления объектами вашей БД. Например, если у вас несколько таблиц, вам нужно уметь переходить от одной к другой и обратно. Таким инструментом служит *область переходов,* показанная на рис. 1.19.

1.4.1. *Просмотр таблиц с помощью области переходов*

Область переходов отображает объекты *(см. разд. "Что такое базы данных Access" ранее в этой главе),* являющиеся частью вашей БД, и позволяет манипулировать ими. Но не всегда вы увидите все объекты вашей БД одновременно. У области переходов есть несколько разных режимов просмотра, поэтому вы можете выбрать именно тот, который вас интересует.

Когда вы впервые создаете БД, в области переходов отображаются только таблицы вашей БД. Пока этого достаточно - помимо всего прочего в вашей БД и нет ничего кроме созданных вами таблиц. (Вы научитесь настраивать область переходов в *главе 14.)*

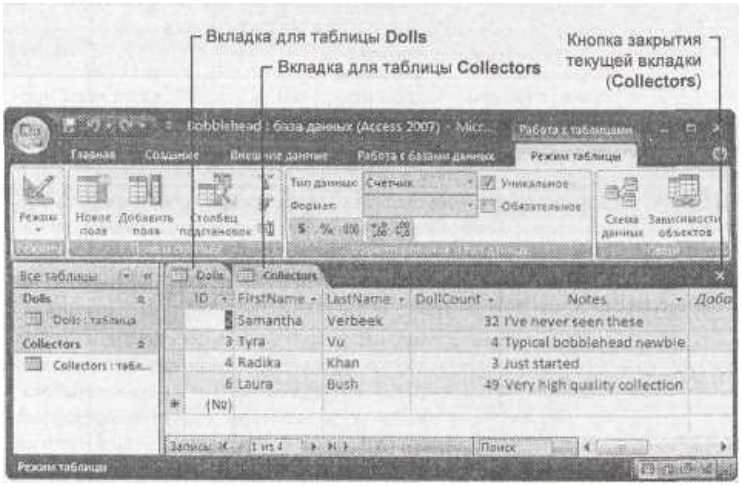


Рис. 1.20. С помощью области переходов вы можете открыть сразу столько таблиц, сколько захотите. Access отводит каждому листу данных отдельное окно-вкладку.

Для перехода из одного окна в другое вы просто щелкаете кнопкой мыши нужную вкладку. Если вам кажется, что открыто слишком много таблиц, щелкните мышью значок x, расположенный у правого края полосы вкладок, для того чтобы закрыть текущий лист данных

Для настоящего испытания области переходов нужна БД с несколькими таблицами. Выберите на ленте **Создание → Таблицы → Таблица** (Create → Tables → Table) для того, чтобы добавить новую пустую таблицу. Выполните действия, описанные в *разд. "Создание простой таблицы "ранее в этой главе* для определения стуктуры таблицы и вставки в нее одной-двух записей.

Совет

Не знаете, какую таблицу создать? Попробуйте создать таблицу Collectors (коллекционеры), которая отслеживает всех ваших друзей, разделяющих ваше увлечение куклами-болванчиками. Теперь попробуйте придумать несколько подходящих полей для этой таблицы (помня о том, что не стоит слишком зацикливаться на подробностях), а затем сравните вашу версию с примером на рис. 1.20.

После вставки новой таблицы вы увидите в области переходов и новую, и старую таблицы одновременно. Если вы хотите открыть таблицу, то в области переходов просто дважды щелкните ее кнопкой мыши. Если вы хотите открыть сразу несколько таблиц, программа Access отобразит их как вкладки (рис. 1.20).

Если вы открыли достаточно большое количество таблиц, в конце концов, осе нужные вкладки могут не поместиться на экране. В этой ситуации программа Access вставляет крошечные кнопки прокрутки слева и справа на полосе вкладок. Вы можете с помощью этих кнопок перемещаться по всем вкладкам, хотя это займет немного больше времени.

Малоизвестная или недооцененная возможность.

Сворачивание ленты

Большинство пользователей рады присутствию под рукой, в верхней части окна программы Access, ленты со всеми ее кнопками. Но серьезным обработчикам информации требуется максимум пространства для их данных. Они предпочтут смотреть еще на одну запись данных, а не на навороченную инструментальную панель. Если вам свойственен такой подход, то вас порадует возможность *свернуть ленту,* превратив ее в одну строку, состоящую из заголовков вкладок, как показано на рис. 1.21. Для этого дважды щелкните кнопкой мыши заголовок любой вкладки.

Но даже когда лента свернута, вы все равно можете использовать все ее функциональные возможности. Просто щелкните мышью вкладку. Если вы щелкнете мышью заголовок **Главная** (Home), вкладка **Главная** раскроется поверх вашего рабочего листа. Как только вы щелкнете мышью по нужной кнопке на этой вкладке (или щелкните мышью где-нибудь еще в окне программы Access), лента свернется снова. Тот же прием сработает, если вы запустите на выполнение с помощью клавиатуры, как описано в *разд. "Использование ленты с помощью клавиатуры" во введении,* команду, представленную на ленте.

Если вы пользуетесь лентой от случая к случаю или предпочитаете комбинации клавиш, имеет

смысл свернуть ленту. Даже при свернутой ленте ее команды доступны; потребуется всего лишь дополнительный щелчок мыши для открытия вкладки. С другой стороны, если вы часто обращаетесь к ленте или знакомитесь с программой Access и хотите просматривать ленту, чтобы узнать все ее доступные функциональные возможности, не тратьте время на ее сворачивание. Потеря дополнительного свободного пространства окупится с лихвой.

1.4.2. *Управление объектами БД*

Теперь вы знаете, как открывать таблицу с помощью области переходов. Но она предназначена не только для открытия таблиц. В действительности вы можете выполнять три более простые задачи, касающиеся любых объектов БД, отображенных в области переходов.

■ *Переименовать объект.* Щелкните его правой кнопкой мыши и выберите команду **Переименовать** (Rename). Затем введите новое имя и нажмите клавишу <Enter>. Действуйте предложенным способом, если решите, что вашу таблицу **Dolls** стоит переименовать в **DollsInMyWorldRenownedCoflection**.

■ *Создать копию.* Щелкните правой кнопкой мыши и выберите команду Копировать (Copy). Щелкните правой кнопкой мыши в любом месте области переходов и затем выберите команду Вставить (Paste). Программа Access попросит вас ввести новое имя для копии. Возможность копирования объекта очень полезна, если вы хотите попытаться реорганизовать имеющуюся таблицу, но пока не готовы удалить оригинал.

■ *Удалить объект.* Щелкните правой кнопкой мыши и выберите команду **Удалить** (Delete). Программа Access попросит вас подтвердить необходимость этой операции, потому что вы не сможете отменить ее.

Access предоставляет дополнительные возможности перемещения объектов БД и скрывания их. Вы познакомитесь с ними позже в этой книге.

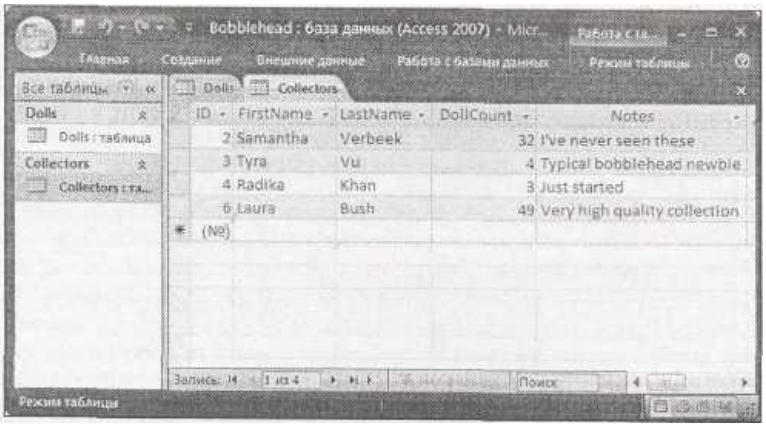


Рис. 1.21. Вы хотите использовать каждый квадратный дюйм пространства экрана для ваших данных? Можно свернуть ленту (как показано здесь), дважды щелкнув кнопкой мыши любую вкладку. Щелкните мышью вкладку для того, чтобы открыть ее на время, или дважды щелкните ее мышью, чтобы вернуть ленту на экран насовсем. Если же вы хотите выполнить этот трюк, не отрывая пальцев от клавиатуры, используйте комбинацию клавиш <Ctrl>+<F1>

Экономящая время подсказка.
Создание ярлыка для таблицы

Возможно, вы уже знаете, что можете поместить на ваш рабочий стол ярлык, указывающий на файл вашей БД.

Для этого просто щелкните правой кнопкой мыши на рабочем столе, выберите последовательность команд **Создать → Ярлык** (New → Shortcut), а затем в соответствии с инструкциями выберите файл вашей БД и задайте имя ярлыка. Теперь, дважды щелкнув кнопкой мыши ярлык, вы сможете в любой момент снова попасть в вашу БД.

Но, возможно, вы не знаете, что можно создать ярлык, который открывает БД и переходит непосредственно к конкретной таблице. На самом деле этот прием даже легче создания обычного стандартного ярлыка. Просто выполните следующие действия:

1.Измените размер окна программы Access так, чтобы оно занимало не весь экран, и затем сверните окна других программ. Благодаря этому вы сможете видеть рабочий стол за окном Access, что существенно для данного приема.

2.Найдите в области переходов таблицу, которую вы хотите использовать. Перетащите ее мышью из окна Access на рабочий стол.

3.Отпустите кнопку мыши. Программа Access создаст ярлык с именем похожим на следующее "Ярлык 'Dolls' (Bobblehead.accdb)" ("Shortcut to Dolls in Bobblehead.accdb"). Дважды щелкните кнопкой мыши ярлык для загрузки БД и немедленного открытия листа данных с таблицей **Dolls.**

2. Глава 2. Создание более сложных таблиц

В предыдущей главе вы научились создавать БД и без особых усилий вставлять в них таблицы. Но должен вас огорчить. Таблицы, которые вы создали к настоящему моменту, не отвечают предъявляемым требованиям.

Самое важное - вы не сообщили четко и ясно программе Access о том, какой тип данных вы намерены хранить в каждом поле вашей таблицы. БД обрабатывает текст, числа, даты и другие типы данных по-разному. Если вы поместите числовую информацию в поле, предназначенное для хранения текста, вы не сможете впоследствии выполнять вычисления (например, найти среднюю стоимость ваших кукол-болванчиков) и не сможете найти ошибки (такие как "кукла-болванчик с ценой "восемьдесят и двадцать"").

Для устранения подобных проблем необходимо определить тип каждого поля вашей таблицы. Это центральная задача, которую вам предстоит решить в данной главе. После того как вы освоите типы данных, можно перейти к рассмотрению более интересных аспектов проектирования БД.

2.1. *Типы данных*

Не все данные одинаковы. Рассмотрим таблицу **Dolls,** созданную вами в *главе 1 (см. разд, "Создание простой таблицы "главы 1).*

В ее полях содержится информация нескольких типов:

- *текстовая* - в полях **Character** (персонаж) и **Manufacturer** (изготовитель);
- *числовая* - в полях **ID** и **PurchasePrice** (покупная цена);
- *даты* - в поле **DateAcquired** (дата приобретения).

Для вас вполне естественно предполагать, что в поле **PurchasePrice** всегда содержатся числовые данные, а в поле **DateAcquired -** информация, которая может интерпретироваться как дата. Но если вы не зададите корректно типы данных, программа Access не будет разделять ваши предположения и следовать тем же правилам, что и вы.

Когда вы создаете новое поле в **Режиме таблицы,** Access делает обоснованное предположение о типе данных, анализируя введенную вами информацию. Если вы ввели 4 4, программа считает, что вы создаете числовое поле. Если вы вводите Янв 6, 2007, Access распознает дату. Однако Access легко запутать, что приводит к проблемам, показанным на рис. 2.1.

Для того чтобы устранить ошибки ввода, следует сообщить программе Access о том, какие сведения должно содержать каждое поле. После того как правила установлены, Access проводит их в жизнь неукоснительно. Вы задаете эти требования с помощью другого окна вашей таблицы - Конструктора.

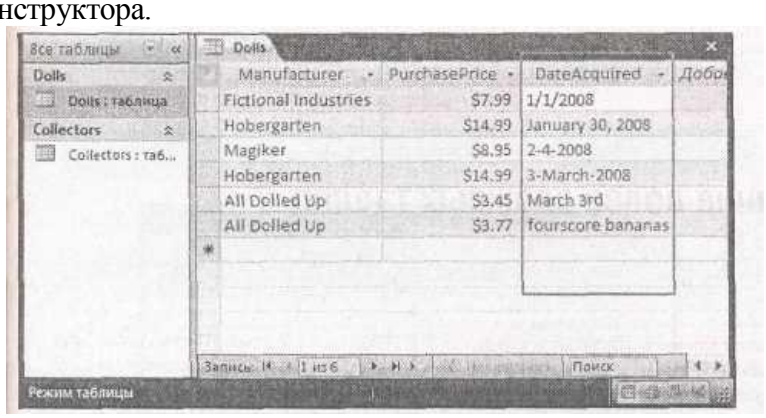


Рис. 2.1. Программа Access не распознает формат даты при создании поля **DateAcquired.** В результате это поле интерпретируется ею как обычный текст. Даты можно ввести в разных форматах (что затрудняет чтение данных в поле **DateAcquired** и делает невозможной их сортировку). Разрешен также ввод абсолютно бессмысленных данных, например "восемьдесят бананов"

2.2. *Конструктор*

Когда вы создаете новую БД, Access предлагает начать с единственной таблицы, отображаемой в **Режиме** таблицы. (В предыдущей главе вы узнали, что **Режим таблицы -** это разделенный на ячейки лист, на котором можно сформировать таблицу и ввести данные.) Для переключения в Конструктор щелкните правой кнопкой мыши имя вкладки (например, **Dolls)** и выберите **Конструктор.** (Вы также можете воспользоваться группой **Режим** (View) на вкладке **Главная** (Home), одноименной группой на вкладке **Работа с таблицам → Режим таблицы → Режим** (Table Tools → Datasheet → View) или кнопками режима в нижней части окна программы Access. Эти варианты показаны на рис. 2.2. Все перечисленные действия выполняют одно и то же, так что выбирайте наиболее удобный для вас способ.)

Примечание

Если вы открыли БД в формате Access 2003, то не увидите никаких вкладок. Вместо этого вы получите груду перекрывающихся окон. Эту проблему можно устранить и вернуть вкладки, выполнив инструкции, приведенные в *разд. "Открытие БД, созданной в более старой версии Access" главы 1.* Если же вы хотите оставить перекрывающиеся окна, пользуйтесь кнопками вида (view buttons) или лентой для смены видов (вместо щелчка правой кнопкой мыши по заголовку вкладки, описанного ранее).

Если вы переходите в Конструктор с таблицей нового формата, которую вы еще не сохраняли, программа Access спросит у вас имя таблицы. Таблица будет сохранена на диске, прежде чем программа переключит вас в **Конструктор.**

Совет

Для ускорения работы вы можете создавать новую таблицу, автоматически стартуя в **Конструкторе.** Для этого выберите на ленте **Создание → Таблицы → Конструктор таблиц** (Create → Tables → Table Design). Но если вы выберите этот путь, у вашей таблицы не будет очень важного столбца **Код** (ID), поэтому вам придется добавить его самостоятельно, как описано в *разд. "Создание поля для вашего собственного первичного ключа" далее в этой главе.*

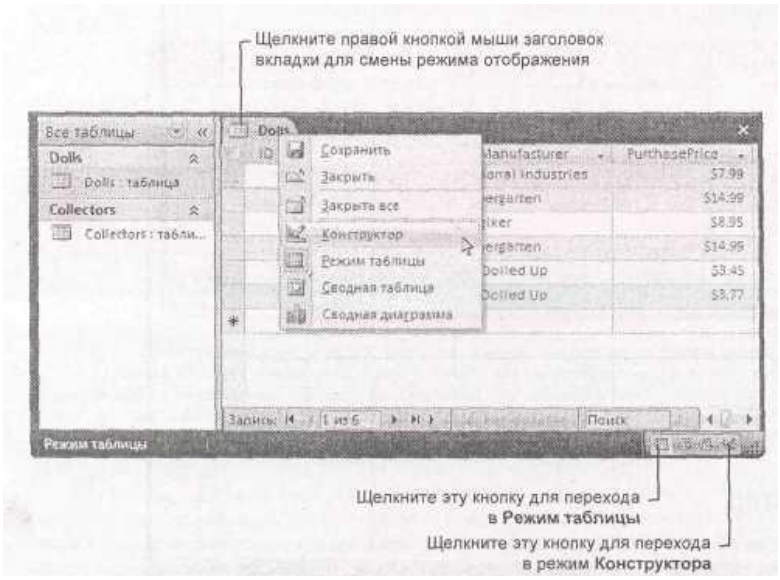


Рис. 2.2. Для отображения данного меню щелкните правой кнопкой мыши имя вкладки. Вы можете перейти в **Конструктор** (выбрать строку **Конструктор)** и вернуться обратно (выбрать **Режим таблицы).** В качестве альтернативы можно использовать маленькие кнопки вида в правом нижнем углу окна для переходов туда и обратно. (Пока не обращайте внимание на две другие кнопки вида. Вы будете использовать их в сводной таблице для анализа ваших данных, описанного в *главе 9.*)

В **Режиме таблицы** отображается содержимое вашей таблицы, а в **Конструкторе -** только ее структура (рис. 2.3).

Конструктор можно использовать для вставки, реорганизации и удаления полей, но не для

добавления новых записей. В таблицу **Dolls** в **Конструкторе** можно вставить поле **Quantity** (количество) для учета дубликатов кукол-болванчиков. Но без перехода обратно в **Режим таблицы** вы не сможете вставить вашу вновь купленную куклу Bono. Конструктор не предназначен для ввода данных.

На первый взгляд этот режим отображения кажется слишком сложным. Для того чтобы упростить его внешний вид, следует начать с закрытия **Окна свойств** (Property Sheet), расположенного в правой части окна программы. (В **Окне свойств** вы можете задать некоторые высокотехнологичные установочные параметры таблицы, принимать во внимание которые прямо сейчас нет никакой нужды.) Для того чтобы убрать это окно, выберите на ленте **Работа с таблицами | Конструктор → Страница свойств** (Table Tools | Design → Property Sheet). В дальнейшем для восстановления окна вам нужно просто повторить эту последовательность.

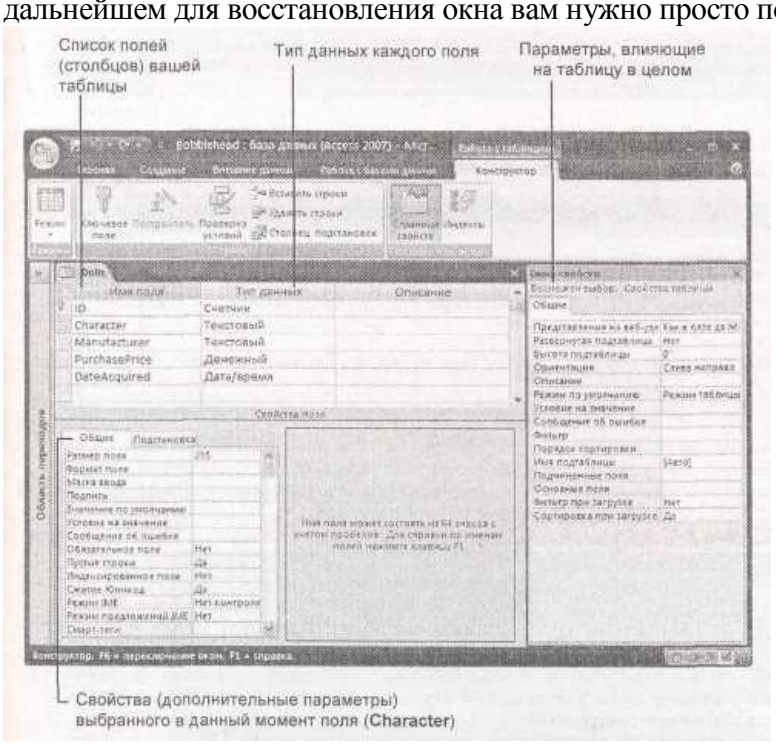


Рис. 2.3. В **Конструкторе** перечисляются поля вашей таблицы, каждое в отдельной строке. В этом режиме поля располагаются сверху вниз, а в **Режиме таблицы** они отображаются по порядку слева направо. Рядом с каждым полем приводится его тип данных и необязательное описание поля. Под списком полей располагается секция **Свойства поля** с дополнительной информацией о выбранном в данный момент поле. В этом режиме область переходов свернута для высвобождения дополнительного пространства

2.2.1. *Организация и описание ваших полей*

Конструктор позволяет изменить порядок следования полей, вставить новые, переименовать имеющиеся и т. д. Все это можно сделать и в **Режиме таблицы,** но знатоки программы Access считают, что легче работать в **Конструкторе,** поскольку вас не отвлекают данные в таблице.

Далее перечислено несколько простых способов изменения структуры вашей таблицы в **Конструкторе таблиц.**

■ *Вставка нового поля в конец таблицы.* Перейдите в последнюю строку списка полей и введите имя нового поля. Это действие эквивалентно вставке нового поля в Режиме **таблицы.**

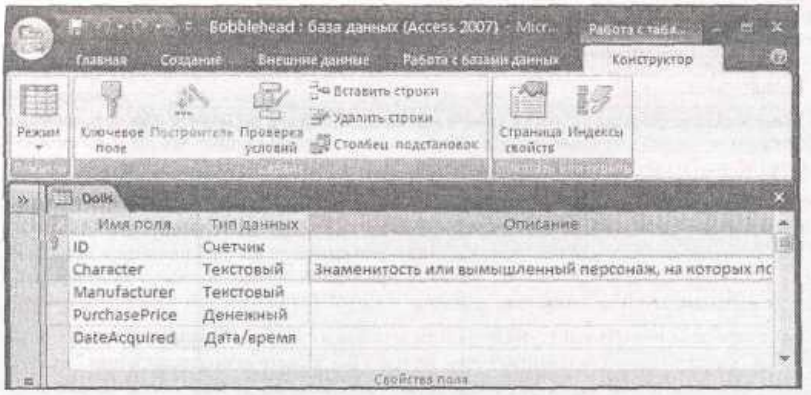


Рис. 2.4. Описания помогают напомнить о том, что есть что, если позже понадобится изменять таблицу. Описания - это чудесная идея в том случае, если несколько человек поддерживают одну и ту же БД и необходима уверенность в том, что поля максимально ясны и понятны. Описания также выводятся в строке состояния при вводе данных в таблицу (рис. 2.5)

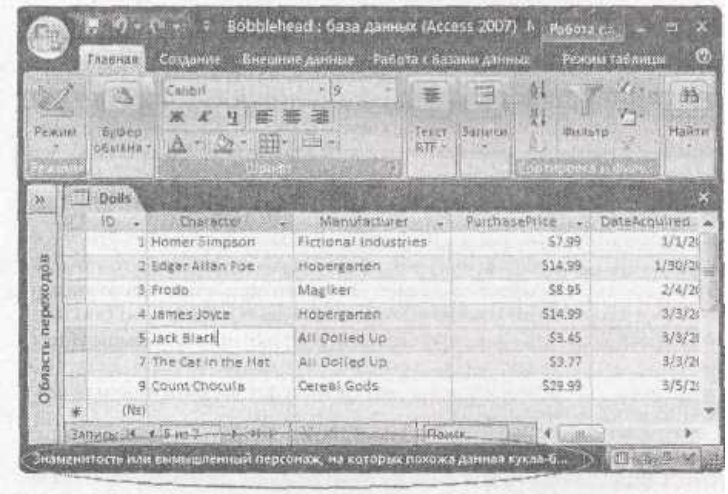


Рис. 2.5. Текст в строке состояния, основываясь на описании поля, сообщает о том, что хранится в данном столбце. К сожалению, это свойство не так полезно, как кажется, потому что большинство пользователей даже не обращают никакого внимания на строку состояния

■ *Добавление нового поля между имеющимися полями.* Перейдите к полю, находящемуся под тем местом, куда вы хотите вставить новое поле. Щелкните поле правой кнопкой мыши и выберите команду **Добавить строки** (Insert Rows). Затем введите имя нового поля в пустую строку.

■ *Перемещение поля.* Перетащите серый квадратик, расположенный у левого края поля, которое вы хотите переместить в новую позицию.

Примечание

Помните о том, что порядок полей совсем не важен, поскольку вы можете изменить порядок отображения полей в **Режиме таблицы.** Но большинство людей считают, что легче проектировать таблицу, упорядочив поля с самого начала.

■ *Удаление поля.* Щелкните правой кнопкой мыши серый квадратик, расположенный слева от поля, которое вы хотите удалить, и выберите команду **Удалить строки** (Delete Rows). Не забывайте о том, что, удаляя поле, вы также уничтожаете осе хранящиеся в нем данные. Это действие нельзя отменить, поэтому программа Access попросит подтвердить ваше желание выполнить именно это действие.

■ *Вставка описания поля.* Введите предложение или два в столбец **Описание** (Description), расположенный рядом с соответствующим полем. (Вы можете использовать имя "знаменитости или вымышленного персонажа, на которых похожа данная кукла-болванчик" как описание поля **Character** (персонаж) в таблице **Dolls,** как показано на рис. 2.4.)

2.2.2. *Как действуют обновления в Конструкторе*

Программа Access не вносит немедленно изменения, сделанные вами в **Конструкторе.** Она ждет, пока вы закроете таблицу или вернетесь в **Режим таблицы. В** этот момент Access спросит о том, хотите ли вы сохранить таблицу. (Обычный ответ, конечно, - да.)

Иногда вносимое вами изменение может создать некоторую проблему. Вы могли попытаться изменить тип данных поля, скажем, текстовый на числовой. (В *примечании "Для тех, кто понимает. Изменение типа данных может привести к потере информации" в разд. "Текстовый" далее в этой главе* данная проблема обсуждается более подробно.) В подобной ситуации вы не обнаружите проблему, пока не закроете таблицу или не вернетесь в **Режим таблицы**, что может произойти позже, чем вам хотелось бы.

Если вы внесли потенциально проблемное изменение и не можете отложить его, у вас есть возможность применить ваше обновление сразу, таким образом, вы сможете увидеть, возникает ли проблема, прежде чем двигаться дальше. Для этого щелкните мышью кнопку **Сохранить** (Save) на инструментальной **Панели быстрого доступа** (Quick Access) (это пиктограмма дискеты в левом верхнем углу окна программы Access) или просто воспользуйтесь сочетанием клавиш <CtrI>+<S>. Access внесет изменение и сохранит таблицу. Если возникнет проблема, программа Access сообщит вам о ней (и позволит выбрать способ ее устранения), прежде чем вы сделаете что-нибудь еще с вашей таблицей.

2.3. *Типы данных Access*

Для определения таблицы **Конструктор -** гораздо более мощное средство, чем **Режим** таблицы. Как вы увидите в этой главе, **Конструктор** позволяет откорректировать все мельчайшие подробности, недоступные (или трудно модифицируемые) в **Режиме таблицы.**

Одна из таких характеристик - *тип данных* вашего поля, параметр, сообщающий программе Access о типе информации, которую вы планируете хранить. Для изменения типа данных выделите столбец **Тип данных** (Data Type), расположенный рядом с соответствующим полем (рис. 2.6). Именно здесь вы отделите текст от чисел (и зададите другие типы данных). О принципах выбора наилучшего типа данных из длинного списка, предоставляемого программой Access, вы узнаете больше в следующем разделе.

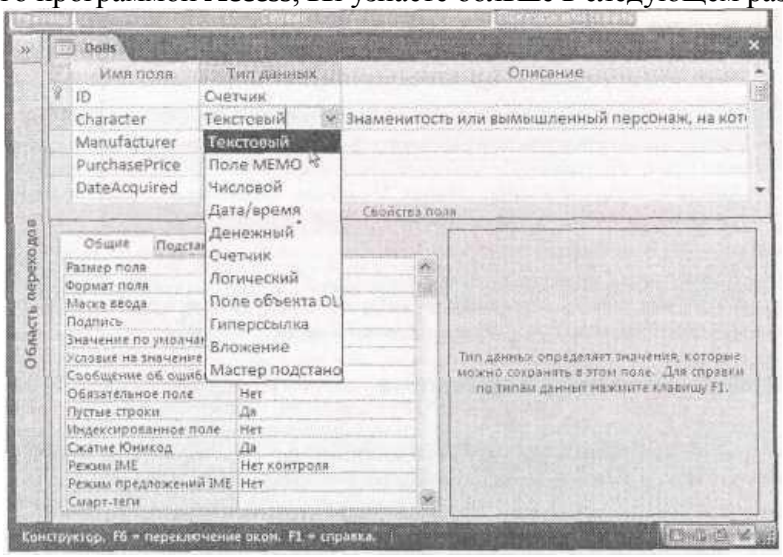


Рис. 2.6. Для выбора типа данных щелкните кнопкой мыши столбец **Тип данных,** расположенный рядом с соответствующим полем. На экране появится раскрывающийся список с 11 вариантами

Существуют и другие свойства поля, зависящие от выбранного типа данных, вы сможете откорректировать их для еще более точного определения типа. Если вы применяете текстовый тип данных, далее вы пользуетесь свойствами поля для указания его максимальной длины. Если выбирается десятичное значение (decimal value), то вы используете свойства поля для задания числа десятичных разрядов в дробной части. Задаются свойства поля в Конструкторе в окне **Свойства поля,** которое отображается под списком полей. В этой главе вы узнаете больше о свойствах поля (и рассмотрите их снова в *главе 4).*

Самое важное принимаемое вами решение, касающееся любого поля, - выбор для него типа

данных. Тип данных сообщает программе Access о том, какую информацию вы намерены хранить в данном поле. Access применяет эти сведения для отклонения лишенных смысла значений (рис. 2.7), выполнения надлежащей сортировки и других действий, таких как вычисления, подсчет итогов или фильтрация.

Примечание

У поля может быть только один тип данных. Вы не можете создать поле, способное хранить данные двух или трех разных типов, поскольку у программы Access не будет достаточной информации для корректной обработки поля. (В подобной ситуации вам, возможно, нужны два разных поля.)

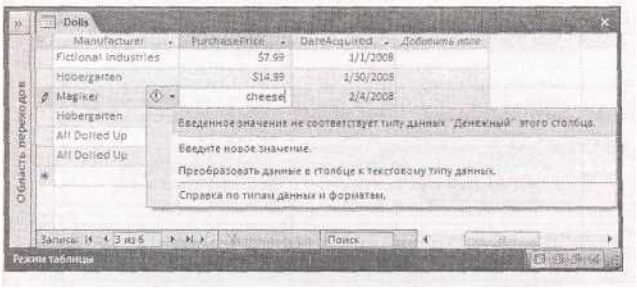


Рис. 2.7. Ввод текста в поле с денежным (currency) типом данных строго запрещен. Программа Access предоставляет возможность устранить проблему с помощью ввода нового значения (правильный подход) или изменения типа данных поля на текстовый, допускающий ввод любых значений (абсолютно неправильный подход)

Как вы уже знаете, существуют три основных типа данных; текст, числа и даты. Но программа Access в действительности предлагает 11 разнообразных типов данных, включающих более специализированные варианты. Прежде чем выбрать нужный тип данных, неплохо познакомиться со всеми возможными вариантами. В табл. 2.1 дан обзор первых 10 вариантов из списка **Тип данных.** (Вариант **Мастер подстановок** (Lookup wizard) не включен в нее, поскольку на самом деле это не тип данных. Этот элемент списка запускает **Мастер подстановок,** позволяющий задать список допустимых значений. Вы узнаете больше об этом варианте *в разд. "Создание простого списка подстановок, состоящего из констант" главы 4.)*

Таблица 2.1. Типы данных Access

Тип данных	*Описание*	*Примеры*
Текстовый (Text)	Числа, буквы, знаки пунктуации и символы, не более 255 (абзац среднего размера)	Имена, адреса, номера телефонов и описания товаров. Это наиболее распространенный тип данных
Поле MEMO (Memo)	Большие обьемы неформатированного текста до 65 536 символов (среднего размера глава в романе)	Статьи, заметки, письма, ордера на арест и другие короткие документы
Числовой (Number)	Все многообразие числовых данных, включая отрицательные и дробные числа	Любой тип чисел за исключением денежных значений. Хранит измерения, итоги и проценты
Денежный (Currency)	Аналогичен числовому типу, но оптимизирован для хранения сумм в денежном выражении	Цены, платежи и статьи расходов
Дата/время (Date/Time)	Календарная дата или время суток (или и то и другое). Не применяйте этот тип данных для задания временных интервалов (количество минут в песне или продолжительность вашей тренировки), для этого больше подойдет числовой тип данных	Дни рождений, даты заказов, даты доставки, свидания и время наблюдений НЛО
Логический	Содержит одно из двух	Строго двухвариантные поля, как

(Yes/No)	значений: Да или Нет. (Вы можете их считать значениями Истина (True) или Ложь (False))	мужской/женский или санкционирован-ный/несанкционированный
Гиперссылка (Hyperlink)	URL (uniform resource locator, унифицированный указатель информационного ресурса) Web-сайта, адрес электронной почты или полное имя файла	www.FantasyPets.com, nore-plies@antisocial.co.uk, f:\Documents\Report.doc
Вложение (Attachment)	Один или несколько отдельных файлов. Содержимое этих файлов колируется в БД	Изображения, документы Word, элек-тронные таблицы Excel, звуковые фай-лы и т. д.
Счетчик (AutoNumber)	Хранит число, генерируемое программой Access при вставке новой записи. Каждой записи автоматически присваивается уникальный номер, идентифицирующий ее	Применяется для уникальной иденти-фикации каждой записи, в особенности для первичного ключа (primary key) (см. разд. "Первичный ключ" далее в этой главе). Обычно столбец называется Код (ID)
Поле объекта OLE (OLE Object)	Хранит встроенные двоичные данные, соответствующие стандарту OLE (Object Linking and Embedding, применяется для обозначения технологий на основе COM, используемых для создания составных документов внедрением и связыванием) ОС Windows. Применяется редко, т. к. приводит к быстрому увеличению размера БД и другим проблемам. Почти всегда лучше выбирать тип данных Вложение (Attachment)	Некоторые типы изображений и доку-ментов, созданных в других програм-мах. Главным образом, применяется в БД Access старого стиля. В наши дни проектировщики БД используют тип данных Вложение (Attachment) вместо поля объекта OLE

В следующих разделах описаны все типы данных за исключением Поле объекта OLE, пришедшего из "средневековья" БД Access. В каждом разделе также описаны все важные свойства поля, характерные для определенного типа данных.

2.3.1. *Текстовый*

Текстовый (Text) - это универсальный тип данных. Он принимает любую комбинацию букв, цифр и других символов. Итак, вы можете применять текстовое поле для хранения двух слов (например, "Мэри Поппинс"), предложения ("Кандидатура - английская няня, склонная поэтическим взлетам.") или что-нибудь еще ("@#$d sf_&!").

Для тех, кто понимает.
Изменение типа данных может привести к потере информации

Лучше всего выбирать типы данных для ваших полей во время первоначального создания таблицы. В этот момент наша таблица практически пуста, и вы не столкнетесь ни с какими проблемами.

Если вы введете несколько записей, а затем решите изменить тип данных в одном из полей, жизнь станет не такой простой. Вы и теперь можете воспользоваться Конструктором для изменения типа данных, но программе Access придется выполнить дополнительное действие и преобразовать имеющиеся данные в новый тип.

В большинстве случаев процесс преобразования проходит гладко. Если в поле хранятся только числа, вы без труда измените текстовый тип данных на числовой. Но в некоторых случаях преобразование не столь безболезненно. Далее перечислены примеры проблем, с которыми вы можете столкнуться.

• Вы изменяете текстовый тип данных на **Дата/время**, но программа Access не может интерпретировать некоторые значения как даты.

• Вы изменяете тип данных **Текстовый** на **Числовой,** но у некоторых ваших записей есть текстовые значения в данном поле (даже если их не должно быть).

• Вы изменяете тип данных **Текстовый** на **Числовой.** Но в вашем поле содержатся дробные числа (например, 4,234), а вы забыли изменить свойство **Размер поля** (Field Size) (см. табл. 2.2). В результате программа Access полагает, что вы используете только целые числа, и обрезает все дробные знаки.

Лучший способ справиться с этими проблемами - создать резервную копию *(см. разд. "Создание резервных копий " в главе 1),* прежде чем вносить любые радикальные изменения, и следить за корректировками, которые сбиваются с пути истинного. В первых двух случаях из приведенного перечня программа Access предупредит вас о необходимости удаления некоторых значений, поскольку они не соответствуют правилам типа данных (рис. 2.8). Третья проблема более коварна - Access выдаст предупреждение, но не сообщит вам, возникнет на самом деле проблема или нет. Если вы предполагаете сбой, перейдите в **Режим таблицы** и проверьте ваши данные, прежде чем двигаться дальше.

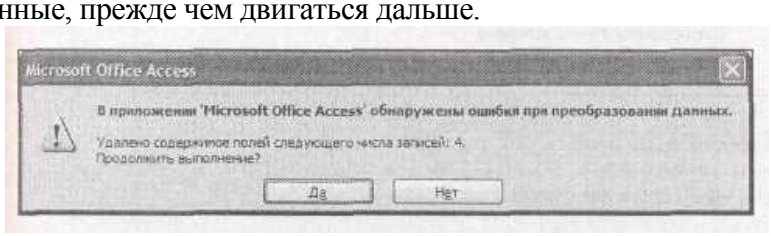


Рис. 2.8. Не говорите о том, что вас не предупреждали. Программа Access дает вам знать (на свой слегка заумный манер), что не может внести заданное вами изменение - преобразование типа данных поля из текстового в дату - без уничтожения значений в четырех записях. Самый разумный план действий - щелкнуть мышью кнопку **Нет** для отказа от изменения и повнимательнее посмотреть таблицу в **Режиме таблицы** для проверки проблемных значений

Примечание

Поскольку текстовые поля столь покладисты, вы, очевидно, можете вводить в них числа, даты и все что угодно. Но текст следует использовать, только если вы сохраняете информацию, которая не может быть обработана с применением другого типа данных, поскольку Access всегда интерпретирует содержимое текстового поля как обычный заурядный текст. Другими словами, если вы сохраняете число 43.99 а текстовом поле, Access не поймет, что вы имеете дело с числами, и не разрешит использовать его в вычислении.

Иногда кажется, что текстовый тип данных уж слишком всеяден. К счастью, вы можете применить некоторые более строгие правила, запрещающие использование определенных символов или вынуждающие текст следовать заранее заданному образцу. Например, программа Access обычно воспринимает номера телефонов как текст, поскольку они представлены последовательностью символов, такой как 123-4444 (а не одним числом 1 234 444). Но вы хотите помешать вставлять в телефонные номера буквы, которые к ним не относятся. Для реализации этого требования можно использовать *маски ввода* (input masks) *(см. разд. "Маски ввода" главы 4)* и проверку корректности (validation) - две функциональные возможности, обсуждаемые в *главе 4.*

Длина текста

У каждого текстового поля есть максимальная длина. Эта особенность вызывает изумление у людей, не привыкших работать с БД. Кроме того, с сегодняшними жесткими дисками гигантских размеров стоит ли беспокоиться об объеме? Не может ли ваша БД расширяться для того, чтобы вмещать любые данные, которые вы хотите в нее затолкать?

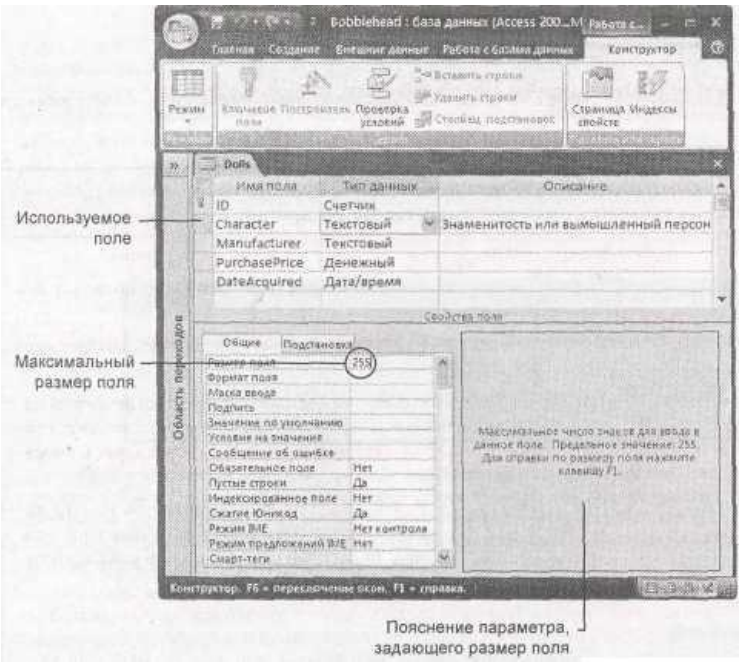


Рис. 2.9. Для задания максимальной длины выберите ваше поле и щелкните кнопкой мыши поле **Размер поля** в списке **Свойства поля** (показанном на рисунке). {Все свойства поля, которые понадобятся вам в этой главе, находятся на вкладке **Общие.)** Когда вы щелкнете кнопкой мыши область свойства поля, справа появится его описание

Максимальная длина имеет важное значение, потому что она определяет, насколько плотно Access может упаковать все ваши записи. Из соображений эффективности программа Access должна быть уверена, что запись целиком хранится в одном месте, поэтому она всегда отводит максимальный объем дискового пространства, который может потребоваться записи. Если в вашей таблице четыре поля, по 50 символов у каждого, Access может зарезервировать для каждой записи на вашем жестком диске объем, необходимый для хранения 200 символов. С другой стороны, если у каждого вашего поля максимальная длина 100 символов, Access хранит в два раза больший объем для каждой записи, даже если в действительности вы не используете его полностью. Дополнительное пространство - не главная проблема (возможно, у вас масса свободного пространства на компьютере), но чем больше места занимает БД, тем медленнее поиск в ней.

Стандартная максимальная длина - 50 - подходит в качестве отправной точки. В *примечании "На профессиональном уровне. Нормативы максимальной длины " далее в этом разделе* содержатся дополнительные рекомендации.

Для задания максимальной длины введите число в поле Размер поля (Field Size) в окне Свойства поля (рис. 2.9). Максимально допустимая величина равна 255 символам. Если нужно хранить большой абзац или целую статью, вам нужен тип данных **Поле MEMO** (*см. следующий раздел).*

Совет

Стоит быть достаточно щедрым, задавая максимальную длину, чтобы в дальнейшем избежать модификации БД.

На профессиональном уровне.

Нормативы максимальной длины

Далее перечислены некоторые рекомендуемые значения максимальной длинны.

• *Имена и фамилии.* 25 символов достаточно для имени, а 50 символов позволят без риска хранить длинную двойную фамилию.

• *Начальная буква отчества.* Один символ. (Иногда здравый смысл подсказывает верное решение.)

• *Адрес электронной почты.* Подойдут 50 символов. В дикой природе встречаются адреса электронной почты, приближающиеся к 100 символам (в качестве дополнительного примера поищите D Google самый длинный в мире почтовый адрес), но маловероятно, что они достигнут вашей БД.

• *Города, штаты, страны и другие географические названия.* Несмотря на то, что название

горы в Новой Зеландии на языке маори превышает 80 символов (см. **http://en.wikipedia.org/wiki/Longest_word__in_English),** для большинства практических целей достаточно 50.

• *Название улицы и номер дома (уличный адрес).* Адрес с указанием улицы состоит из числа, за которым следуют пробел, название улицы, еще один пробел и сокращение (такое как пр. или ул.). 50 символов хватит, если почтовые индексы, названия городов и другие подробности адреса вы поместите в другие поля.

• *Номера телефонов, почтовые индексы, номера кредитных карт и другой текст фиксированной длины*. Сосчитайте количество символов без учета заполнителей и задайте соответствующий максимум.

Если нужно хранить номер телефона (123) 456-7890, задайте длину поля, равной 10 символам. Вы сможете хранить номер в виде 1234567890, *но* при выводе на экран использовать маску ввода *(см. разд. "Маска ввода" главы 4),* для того чтобы добавить скобки и дефис. Этот подход хорош тем, что позволит избежать проблем из-за ввода однотипных номеров телефонов разными способами.

Описание или комментарии. 255 символов соответствуют трем или четырем предложениям среднего размера. Если вам нужно больше, рассмотрите возможность применения типа данных **Поле MEMO** *(см. следующий раздел).*

2.3.2. *Поле MEMO*

Корпорация Microsoft разработала тип данных **Поле MEMO** (Memo) для хранения больших объемов текстовой информации. Если вы хотите поместить в поле главу из книги, целую газетную статью или просто несколько абзацев текста, вам нужен тип данных **Поле MEMO.** Название немного странное - хотя поле Memo может хранить информацию из межофисного договора, оно также всегда полезно при наличии больших блоков текста.

Если вы создали поле Memo, вам не придется задавать его максимальную длину, программа Access хранит данные в поле этого типа не так, как данные других типов. По существу, она заталкивает данные типа Memo в отдельную секцию, поэтому может хранить оставшуюся часть записи настолько плотно и эффективно, насколько это возможно, вмещая при этом большой объем текста.

Длина поля Memo может достигать 65 536 символов. Учтите на будущее, что у этой главы примерно такой размер. Если вам нужен больший объем, добавьте несколько полей Memo.

Примечание

Технически ограничение в 65 536 символов - это ограничение пользовательского интерфейса в программе Access, а не БД. Если вы программируете приложение для обработки вашей БД, то она может хранить гораздо больше информации в поле Memo вплоть до гигабайта.

Если нужно отредактировать большой объем текста во время работы на листе данных, можно воспользоваться окном **Область ввода** (Zoom) (рис. 2.10). Просто перейдите в поле, которое вы хотите редактировать, и нажмите сочетание клавиш <Shift>+<F2>.

Форматированный текст

Текстовое поле и поле Memo хранят неформатированный текст. Но в поле Memo можно хранить и форматированный текст (rich text), содержащий разные шрифты, цвета, выравнивание и т. д. Для этого установите для параметра **Text Format** значение **Rich Text (в** отличие от стандартного значения **Plain Text** (обычный текст)).

Для форматирования фрагмента текста вам нужно просто выделить его и затем выбрать параметры форматирования на ленте в группах **Главная → Шрифт → Текст RTF** (Home → Font Home → Rich Text). Но в большинстве случаев вы не будете прибегать к этому способу, поскольку трудно редактировать большие фрагменты текста в узких столбцах листа данных. Вместо этого используйте сочетание клавиш <Shift>+<F2> для открытия окна **Область ввода** (рис. 2.11), а затем мини-панель инструментов (minibar).

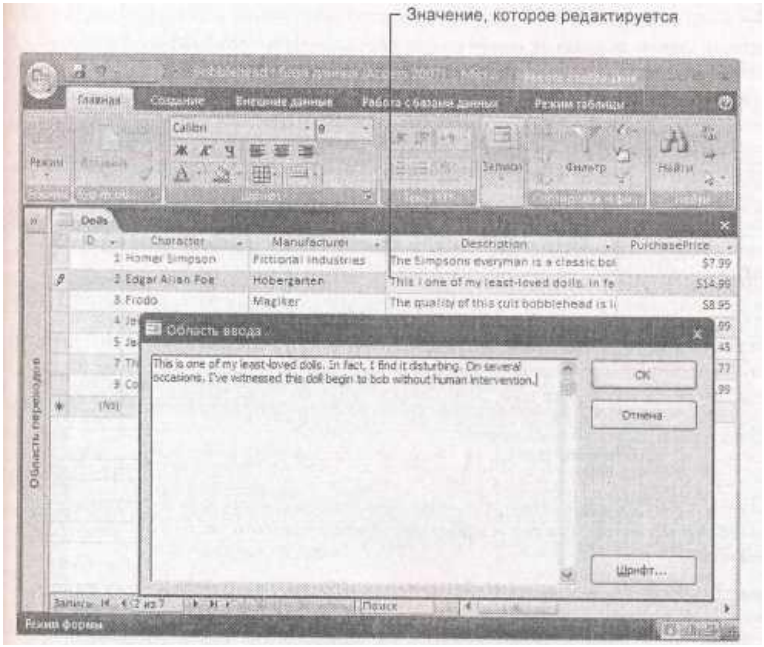


Рис. 2.10. Если у вас в поле длинный текст, его трудно увидеть целиком без долгой прокрутки. Открыв окно **Область ввода** (<Shift>+<F2>), вы увидите больше текста, и редактировать его будет гораздо легче. Вы должны будете щелкнуть мышью кнопку **ОК** (для принятия исправлений) или **Отмена** (для отказа от них) для того, чтобы снова вернуться на лист данных

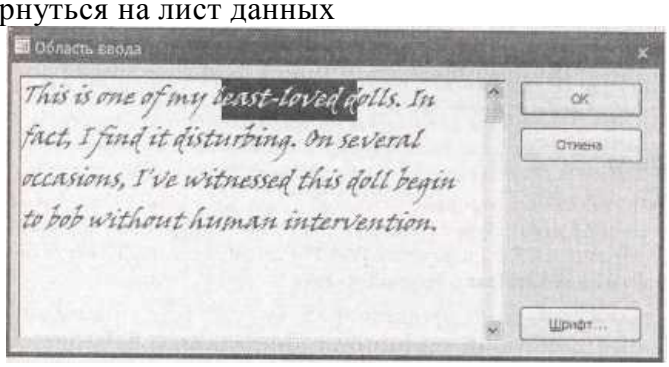


Рис. 2.11. Для отображения мини-панели инструментов выберите какой-нибудь текст и наведите на него указатель мыши. Мини-панель инструментов - компактная панель с параметрами форматирования - постепенно станет более отчетливой. Иногда она слегка капризна и возможно придется выделить текст заново, чтобы заставить панель появиться на экране

Совет

Существует другой, даже более легкий способ помещения форматированного текста в поле Memo. Создайте текст в программе, текстовом процессоре (таком как Word), отформатируйте его в этой программе и затем скопируйте и вставьте в поле. Все форматирование текста сохранится.

Аккуратно реализованное, как может показаться на первый взгляд, это средство редко заслуживает внимания. Пуристы БД уверены, что таблицы должны содержать чистые данные и что необходимо давать возможность другим программам (или разукрашенным формам) решать, как их форматировать. Проблема заключается в том, что после создания форматированного текста его поддержка может стать очень трудной задачей. Только представьте себе необходимость изменить шрифт в 30 000 разных записей.

Если вы действительно хотите хранить форматированное содержимое, рассмотрите возможность связывания вашей БД с отдельным документом, например файлом Word. В программе Access можно сделать это двумя способами:

■ *создать поле, указывающее на файл.* Например, c:\myfile\BonoBobbleheadDescription.docx. Для реализации этого способа используйте текстовый тип данных или гиперссылку *(см. разд. "Гиперссылка " далее в этой главе)',*

■ *встройте файл внутрь вашей БД.* Применение этого способа сделает невозможной потерю файла (или указание на неверное местоположение). Однако вам придется каждый раз

при необходимости редактирования удалять файл. Для этого используйте тип данных **Вложение** *(см. разд. "Вложение" далее в этой главе).*

2.3.3. *Числовой*

Числовой (Number) тип данных включает огромное разнообразие чисел разной величины. Можно выбрать вариант с дробной частью или использовать отрицательные числа (просто перед числовым значением поставить знак "минус"). Числовой тип данных следует применять для любой имеющейся числовой информации за исключением денежных сумм, которым больше подходит тип данных **Денежный** (Currency) *(см. разд. "Денежный" далее в этой главе).*

При использовании числовых полей вы не включаете информацию о применяемых единицах измерения. У вас могут быть поля, представляющие вес в фунтах, высоту в метрах и возраст в годах. По эти поля содержат только числа. Ваша задача,- знать, что обозначает каждое число. Если вы считаете, что другие могут запутаться, рассмотрите возможность включения единиц измерения в описание поля *(см. разд. "Организация и описание ваших полей" ранее в этой главе)* или включите эту информацию в имя поля (например, **HeightInMeters** (высота в метрах)).

Примечание

Ваше поле никогда не должно содержать такие данные, как "44 фунта". Программа Access интерпретирует это значение как текстовое, поэтому, допустив такую ошибку, вы не сможете применять все важные средства решения числовых задач большого объема (crunching) или проверки правильности (validation), о которых вы узнаете позже в этой книге.

Размер числа

Как и в случае текстового поля, создавая числовое ноле, вы должны задать свойство **Размер поля** (Field Size) для гарантии того, что программа Access зарезервирует для него нужный объем пространства на диске. Но в случае числового поля у вас более сложный выбор по сравнению с обычным текстом.

По существу, числа разделены на несколько подмножеств, в зависимости от того, поддерживают они или нет дробные значения (числовые разряды справа от десятичной точки или запятой) и сколько байтов программа Access использует для их хранения.

Примечание

Байт - это группа из 8 битов, мельчайшей единицы хранения в компьютерном мире. Например, мегабайт - это примерно миллион байтов.

В табл. 2.2 перечислены разные варианты значений поля **Размер поля** (Field Size), которые можно выбрать для данных числового типа, и объясняется, когда логичнее всего применять каждое из них. Первоначально Access выбирает для всех полей значение **Длинное целое** (Long Integer), предоставляющее достаточный объем, но запрещающее наличие дробных чисел.

Таблица 2.2. Варианты значений в поле Размер поля для числового типа данных

Свойство Размер поля	*Содержит*	*Когда применяется*
Байт (Byte)	Целые значения (целое число) а диапазоне от 0 до 255. Для хранения требуется 1 байт	Это рискованный размер, поскольку подходит только для маленьких чисел. Обычно безопаснее использовать для таких чисел значение Целое (Integer) и тем самым обеспечить немного больше места для их хранения
Целое (Integer)	Целые значения (целое число) в диапазоне от -32 768 до 32 767. Для хранения требуется 2 байта	Применяется для хранения чисел, не имеющих дробной части
Длинное целое (Long Integer)	Целые значения (целое число) в диапазоне от -2 147 483 648 до 2 147 483 647. Для хранения	Стандарт программы Access. Хороший выбор с достаточным объемом пространства для хранения. Используйте

	требуется 4 байта	этот вариант для хранения любых чисел, не превышающих максимум, если вам не нужна дробная часть
Одинарное с плавающей точкой (Single)	Положительные или отрицательные числа, содержащие до 38 нулей и 7 десятичных разрядов точности. Для хранения числа требуется 4 байта	Лучший выбор для хранения дробных чисел или чисел, которые слишком велики для размера Длинное целое (Long Integer)
Двойное с плавающей точкой (Double)	Положительные или отрицательные числа, содержащие до 308 нулей и 15 десятичных разрядов точности. Для хранения числа требуется 8 байтов	Полезен, если вам нужны необычно большие числа
Действительное (Decimal)	Положительные или отрицательные числа, содержащие до 28 нулей и 28 десятичных разрядов точности. Для хранения числа требуется 8 байтов	Подходит для хранения дробных чисел с большим количеством разрядов справа от десятичной точки

Примечание

В табл. 2.2 не включен вариант *Код репликации,* поскольку он применяется только с типом данных **Счетчик** *(см. разд. "Счетчик" далее в этой главе).*

Числовой формат

Свойство **Размер поля** (Field Size) определяет, как программа Access хранит ваше число в таблице. Но помимо этого вы можете выбрать способ его представления на листе данных. Например, 50, 50.00, 5E1, $50.00 и 5000% - все это одно и то же внутреннее число, но люди воспринимают эти варианты очень по-разному.

Для выбора формата задается свойство поля **Формат** (Format).

В основные встроенные варианты представления включены следующие.

- *Обычный.* Отображаются обычные числа, такие как 43.4534. Любые дополнительные нули справа от числа отбрасываются (поэтому 4.10 превращается в 4.1).
- *Денежный и Евро.* Оба варианта представления отображают числа с двумя дробными разрядами, разделителями тысяч (запятая в числе 1, 000 . 00) и знаком валюты[1]. Эти виды форматирования используются только с денежным типом данных *(см. разд. "Денежный" далее в этой главе).*
- *Фиксированный.* Числа отображаются с одинаковым числом десятичных разрядов в дробной части, при необходимости заполняемых нулями (например, 432.11 и 39.00). Длинный столбец, выровненный по позиции десятичной точки, облегчает чтение ваших таблиц.
- *С разделителями разрядов.* Похож на фиксированный формат, за исключением использования также разделителей для тысяч, чтобы помочь анализировать большие числа, например, 1,000,000.00.
- *Процентный.* Отображает дробные числа как проценты. Например, если вы введете число 0.5, оно преобразуется в 50 %.
- *Экспоненциальный.* Отображает числа в экспоненциальной форме, идеальной для обработки чисел с широким диапазоном изменения (например, 0, 0003 и 300). Экспоненциальное представление отображает первую ненулевую цифру числа с последующим фиксированным количеством цифр и затем указанием порядка, количества перемножений числа 10 для формирования задаваемого числа. Например, число 0.0003 преобразуется в 3.00×10^{-4}, отображаемое как 3.00E-4. С другой стороны, число 3 00 превращается в 3.00×10^{2} или 3E2.

Совет

Если вы используете фиксированный, процентный, экспоненциальный или с разделителями разрядов форматы, также следует задавать свойство поля **Число десятичных знаков** (Decimal Places) для указания количества выводимых на экран десятичных разрядов в дробной части. В противном случае вы всегда будете получать два.

■ *Строка пользовательского формата.* Это зашифрованный код, сообщающий программе Access точную форму представления числа. Вы должны ввести строку необходимого вам формата в поле Формат (Format).

[1]Разделитель тысяч и знак валюты настраиваются в Панели управления. - *Ред.*

Например, если ввести причудливо выглядящий код #, ##0, (включая запятую в конце), Access скроет три последние цифры каждого числа, поэтому 1 миллион будет выводиться как 1,000, а 15 000 как 15.

Примечание

Пользовательские числовые форматы не очень распространены в программе Access (гораздо чаще их используют в программе Excel). Позже вы узнаете о выражениях *(см. разд. "Определение вычисляемого поля" в главе 7),* которые позволят делать почти то же самое.

2.3.4. *Денежный*

Денежный (Currency) тип данных - это легкая вариация числового типа данных, предназначенная для финансовых расчетов. В отличие от числового типа данных, для денежного не ладо выбирать значение свойства **Размер поля** (Field Size), у Access есть универсальная стратегия, требующая восьми байтов для хранения каждого числа.

Примечание

Денежный тип данных лучше числового типа данных, поскольку он использует оптимизацию, препятствующую возникновению ошибок округления в очень маленьких дробях. У денежного типа данных точность до 15 цифр слева от десятичной точки и 4 цифр справа от нее.

Вы можете изменить количество разрядов дробной части, которые программа Access отображает для значений этого типа при выводе на лист данных, задав свойство **Число десятичных знаков** (Decimal Places). Обычно оно равно 2.

Форматирование, используемое Access для вывода денежных сумм, определяется установками в апплете вашего компьютера **Язык и региональные стандарты** *(см. примечание На профессиональном уровне. Представление даты на вашем компьютере" в следующем разделе).* Иногда эти установки могут создавать нежелательный эффект - например, у вас малое предприятие в Дании по производству хлопьев, продающее все свои изделия за границей в долларах США (не в кронах). Вы можете точно управлять форматированием денежных сумм, задав свойство **Формат** (Format), предоставляющее следующие варианты:

■*Денежный.* Это стандартный выбор. Он использует форматирование, базирующееся на региональных стандартах, заданных на вашем компьютере.

■*Евро.* Этот вариант всегда применяет символ евро (€).

■*Строка пользовательского формата.* Этот вариант позволяет задать символ любой нужной вам валюты (как описано далее). Вы должны ввести строку необходимого вам формата в свойство **Формат.**

Есть простой рецепт "приготовления" строк формата с пользовательским символом валюты. Начните со вставки символа, обозначающего валюту (введите то, что нужно) и затем добавьте #,###.## - код программы Access, означающий: "дай мне число с разделителями тысяч и двумя знаками в дробной части".

Например, датская компания по производству хлопьев может использовать следующую строку формата для отображения символа валюты Соединенных Штатов:

$#,###.##

Тогда как компания США, нуждающаяся в отображении поля с датской валютой (с форматом цены, таким как kr 342.99), воспользовалась бы следующей строкой формата:

kr #,###.##

Примечание

Инициативные пользователи могут поиграть с числовым форматом, изменяя число десятичных разрядов в дробной части (просто вставляя и убирая знаки цифр) и удаляя разделители разрядов (простым уничтожением запятой).

2.3.5. *Дата/время*

Программа Access использует тип данных **Дата/время** (Date/Time) для хранения определенного момента времени в сочетании с годом, месяцем, днем и временем суток, заданным с точностью до секунды. Внутри БД Access даты хранятся как числа, что позволяет использовать их в вычислениях.

Несмотря на то, что в Access всегда для хранения в поле даты используется одно и то же количество байтов, некоторую часть информации можно не отображать. Вы можете вывести на экран только дату (и игнорировать информацию о времени суток) или только время (и игнорировать дату). Для этого нужно просто задать свойство поля **Формат** (Format). В табл. 2.3 перечислены возможные варианты.

Таблица 2.3. Форматы типа данных Дата/время

Формат	*Пример*
Полный формат	2/23/2008 11:30:15 PM
Длинный формат	Февраль 23, 2008
Средний формат	23-Фев-08
Краткий формат	2/23/2008
Длинный формат	11:30:15 PM
Средний формат	11:30PM
Краткий формат	23:30

Примечание

В случае применения *Полного формата даты* и *Длинного формата даты* информация о времени выводится, только если она ненулевая.

Формат влияет только на способ отображения информации о дате - он не меняет способ ее ввода. Программа Access достаточно интеллектуально развита, чтобы правильно интерпретировать даты, введенные следующим образом:

- 2008-23-2 (всегда работает интернациональный стандарт "год-месяц-день");
- 2/23/2008 (наиболее распространенный вариант ввода, но, возможно, па компьютерах за пределами США вам придется поменять местами день и месяц);
- 23-Фев-08;
- Фев 23 (Access полагает, что имеется в виду текущий год);
- 23 Фев (аналогично).

Для вставки даты и времени просто следом за датой введите время, например, 23-Фев-08 5:06 PM. Не забудьте вставить в конце обозначение AM/PM или используйте 24-часовую шкалу.

Вместо набора даты можно использовать смарт-тег календаря (calendar smart tag). Смарт-тег - это пиктограмма, появляющаяся рядом с полем, как только вы переходите в него, как показано на рис, 2.12.

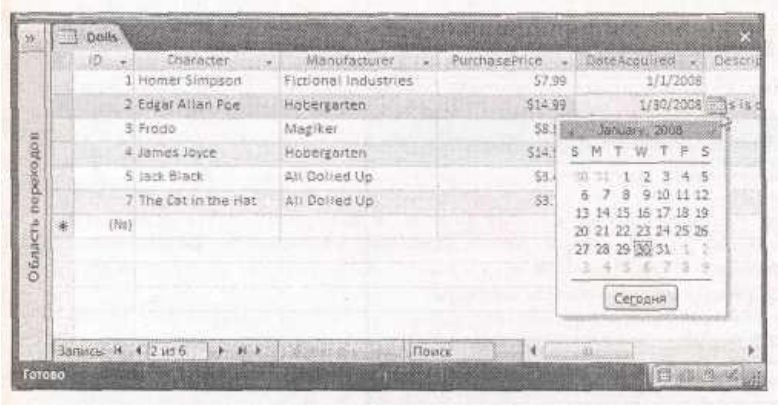


Рис. 2.12. Access автоматически высвечивает на экране этот смарт-тег для всех полей с датами. Щелкните кнопкой мыши пиктограмму для вывода на экран мини-календаря, в котором вы сможете выбрать нужную дату. Но календарь не поможет ввести сведения о времени

На профессиональном уровне.

Представление даты на вашем компьютере

На вашем компьютере и ОС Windows есть региональные установки, влияющие на способ отображения дат и валют. В Access региональные установки определяют способ отображения разных форматов для дат. Другими словами, в США на компьютере прямой поставки с завода *Краткий формат даты* отображается как 2/23/2008. А на британском компьютере он будет выводиться как 23/2/2008. В любом случае в БД хранится одна и та же информация. Но меняется способ ее вывода на лист данных.

Вы можете откорректировать региональные установки, и они вовсе необязательно должны соответствовать району вашего проживания, - например, вы можете задать их для центрального управления вашей компании, находящегося на другом континенте. Но помните, что это глобальные параметры, поэтому, изменяя их, вы оказываете влияние на все ваши программы.

Для внесения изменений перейдите на Панель управления (Control Panel). (В ОС Windows XP щелкните кнопкой мыши кнопку меню **Пуск** (Start) и выберите последовательность команд **Настройка | Панель управления** (Settings | Control Panel). В Windows Vista щелкните мышью **Пуск** и ищите **Панель управления** справа.) После того как вы открыли Панель управления, дважды щелкните кнопкой мыши пиктограмму **Язык и региональные стандарты,** которая выведет на экран диалоговое окно. Все нужные вам установочные параметры находятся на первой вкладке. Первое поле - самое важное, у него есть раскрывающийся список, из которого можно выбрать регион, предполагаемый для использования, например, Английский (США) или Шведский (Финляндия).

С помощью дополнительных параметров можно произвести более тонкую настройку. Делать это имеет смысл, только если у вас есть конкретные предпочтения, касающиеся форматирования дат и не совпадающие со стандартными установками. Щелкните мышью кнопку **Настройка,** расположенную рядом с полем выбора региона, для вывода на экран нового диалогового окна, затем щелкните кнопкой мыши вкладку Дата (показанную на рис. 2.13).

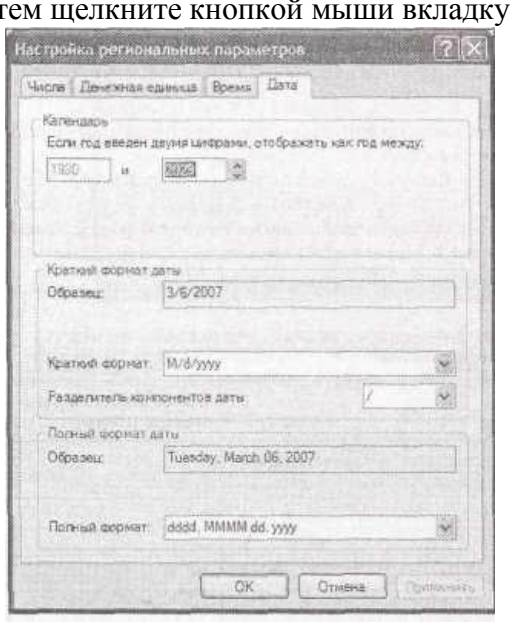


Рис. 2.13. Диалоговое окно **Настройка региональных параметров** позволяет настроить способ отображения дат на вашем компьютере. Воспользуйтесь раскрывающимися списками для выбора разделителя даты, порядка следования компонентов день, месяц, год в дате и способа интерпретации программой Access лет, задаваемых двумя последними цифрами. Можно формировать любые комбинации значений этих параметров, но в итоге вы можете так настроить компьютер, что его поведение покажется противоестественным всем остальным

Пользовательские форматы дат

Если вы не удовлетворены семью стандартными вариантами вывода дат, предлагаемыми

программой Access, можно сформировать собственную строку формата даты и ввести ее в свойство **Формат** (Format). Эта строка сообщает программе Access способ представления даты и времени.

Строка формата даты состоит из нескольких частей. Каждая часть представляет отдельный компонент даты, такой как день, месяц, год, минута, час и т. д.

Вы можете соединять эти части в любом порядке. Например, посмотрите на следующую строку формата: yyyy-mm-dd

Ее можно транслировать в следующие инструкции: выведи четырехзначный год с последующим дефисом, затем двузначный номер месяца с последующим дефисом и далее двузначный номер дня в месяце. Вы вольны располагать эти компоненты как вам захочется, но данный пример определяет их порядок в соответствии со стандартом ISO (International Organization for Standardization, Международная организация по стандартизации) для дат. Вы также можете управлять способом вывода года, дня и месяца в дате. Можно применять сокращенные или полные названия месяцев вместо номера месяца (просто замените код mm чем-то другим).

Если вы примените эту строку формата дат к полю, в котором содержится дата Январь 1, 2008, то увидите ее на листе данных в таком виде: 2008-01-01

Помните о том, что независимо от того, какую информацию вы решили отображать или скрывать при выводе, Access хранит в вашей БД одни и те же данные, касающиеся даты.

В табл. 2.4 приведены основные заполнители, используемые в строке формата для даты или времени.

Таблица 2.4. Код **для форматирования даты и времени**

Код	*Описание*	*Выводится (для даты 01.01,2008)*
d	Номер дня в месяце, 1-31 с номерами 1-9, выводимыми без ведущего нуля (0)	1
dd	Номер дня в месяце, в диапазоне 1-31, (для номеров 1-9 добавляется ведущий нуль (0))	01
ddd	Сокращенное название дня недели	Вт
dddd	Полное название дня недели	Вторник
m	Номер месяца в диапазоне 1-12 (ведущие нули не применяются)	1
mm	Номер месяца в диапазоне 1-12 (ведущие нули применяются для 01- 09)	01
mmm	Трехбуквенное сокращенное название месяца	Янв
mmmm	Полное название месяца	Январь
yy	Сокращенное двузначное обозначение года	08
yyyy	Год задается всеми четырьмя цифрами	2008
h	Час от 0 до 23 (ведущий нуль не применяется)	13
hh	Час от 0 до 23 (ведущий нуль применяется для значений 00-09)	13
:m	Минута в часе от 0 до 59 (ведущий нуль не применяется)	5
:mm	Минута в часе от 0 до 59 (ведущий нуль применяется для значений 00-09)	05
:s	Секунда в минуте от 0 до 59 (ведущий нуль не применяется)	5
: ss	Секунда в минуте от 0 до 59 (ведущий нуль применяется для значений 00-09)	05
AM/PM	Предписывает программе Access использовать 12-часовую шкалу с индикацией первой (AM) и второй половины (PM) суток	PM
am/pm	Обозначает 12-часовую шкалу с индикацией первой (am) и второй (pm) половины суток	pm
A/P	Предписывает программе Access использовать 12-часовую шкалу с индикацией первой (A) и второй половины (P) суток	P
a/p	Предписывает программе Access использовать 12-часовую шкалу с индикацией первой (a) и второй половины (p) суток	p

2.3.6. *Логический*

Поле с логическим типом данных (Да/Нет) - это чудо эффективности. Представляет собой простейший тип данных Access, поскольку допустимы только два возможных значения: *Да* или *Нет.*

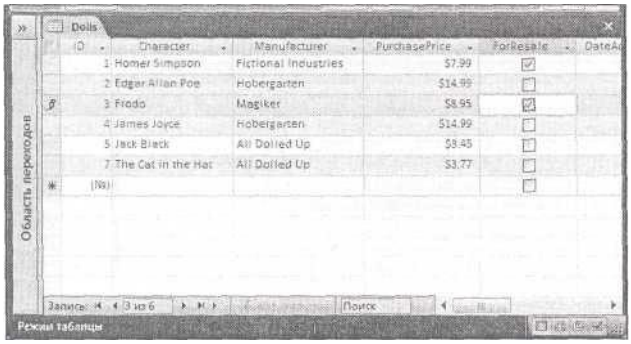


Рис. 2.14. В данном примере поле **ForResale** (для продажи) - поле с логическим типом данных. Установленный флажок отображает значение *Да* (или *Истина*, или *Вкл*). Сброшенный флажок означает *Нет* (или *Ложь*, или *Выкл*)

Применяя поле с логическим типом данных, представьте себе, что поле содержит ответ "да" или "нет" на вопрос, который получается, если добавить воображаемый вопросительный знак к названию поля. Вы можете применять поле с именем **InStock** для отслеживания наличия изделий на складе. В данном случае "да" или "нет" - ответ на вопрос "На складе?" Другими примерами могут служить поле **Shipped** (доставленные) (в списке заказов), **Male** (мужчина) (для разделения мальчиков и девочек) и **Republican** (республиканец) (при условии, что вы хотите различать только две политические ориентации).

Несмотря на то, что все поля логического типа одинаковы, для них можно выбрать слегка отличающиеся форматы, заменяя слова "Да" и "Нет" словами Вкл/Выкл или Истина/Ложь. Эти три варианта можно найти в списке свойства **Формат** (Format). Но у них мало различий, поскольку на листе данных поля этого типа отображаются с флажком, как показано на рис. 2.14.

2.3.7. *Гиперссылка*

Тип данных **Гиперссылка** (Hyperlink) подойдет, если бы хотите создать ссылку на Web-страницу, файл или адрес электронной почты, срабатывающие по щелчку кнопки мыши. Вы можете в одной таблице создавать любые комбинации этих трех видов указателей.

В **Режиме таблицы** Access обрабатывает гиперссылки немного иначе. Когда вы вводите текст в поле типа Гиперссылка, он окрашивается в синий цвет и подчеркивается. И когда вы щелкаете ссылку кнопкой мыши, Access открывает ее в вашем Web-обозревателе (рис. 2.15).

Примечание

Программа Access не мешает вам вводить в поле с типом данных **Гиперссылка** значения, не являющиеся гиперссылками. Эта особенность может создать проблему, когда вы щелкнете кнопкой мыши ложную гиперссылку. Если вы поместите текст "saggy balloons" (сдувшиеся шарики) в поле типа **Гиперссылка** и щелкните его кнопкой мыши, Access попытается отправить Web-обозреватель по адресу **http://saggy balloons,** который на самом деле не существует.

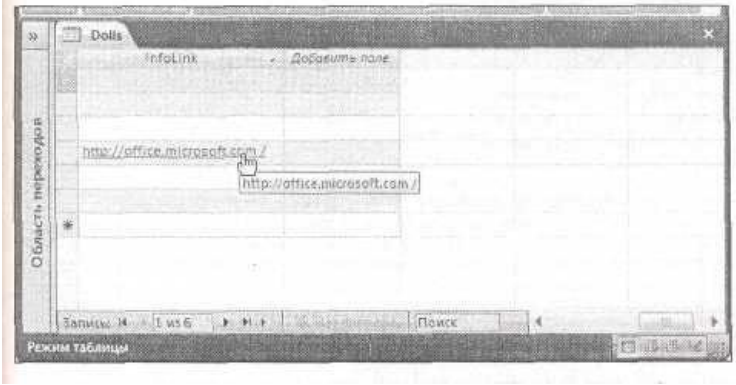


Рис. 2.15. Щелкните кнопкой мыши эту гиперссылку и попадете прямо на

доброжелательный Web-сайт Office Online

Одно свойство поля типа **Гиперссылка** сразу не очень понятно. На самом деле такие поля хранят несколько порций данных. Каждая гиперссылка включает три компонента:

■ текст, который вы видите в ячейке;

■ адрес, на который вы переходите при щелчке кнопкой мыши ячейки (URL или полное имя файла);

■ текст, который вы видите при наведении указателя мыши на ссылку (пояснительная надпись).

Когда вы вводите гиперссылку на листе данных, все три компонента получают одно и то же значение - то, что вы только что ввели. Другими словами, когда вы набираете http://www.FantasyPharmacologists.com, текст, который вы видите, URL ссылки и пояснительная надпись содержат одну и ту же информацию - URL - http://www.FantasyPharmacologists.com.

В большинстве случаев этот подход хорош, т. к. позволяет быстро просмотреть ссылку. Но это не единственно возможная стратегия. Если вы хотите трем описанным компонентам присвоить разные значения, перейдите в ячейку с набранным значением и нажмите сочетание клавиш <Ctrl>+<K> для того, чтобы раскрыть окно **Изменение гиперссылки** (Edit Hyperlink) - рис. 2.16. Или щелкните значение правой кнопкой мыши и выберите последовательность команд **Гиперссылка → Изменить гиперссылку** (Hyperlink → Edit Hyperlink).

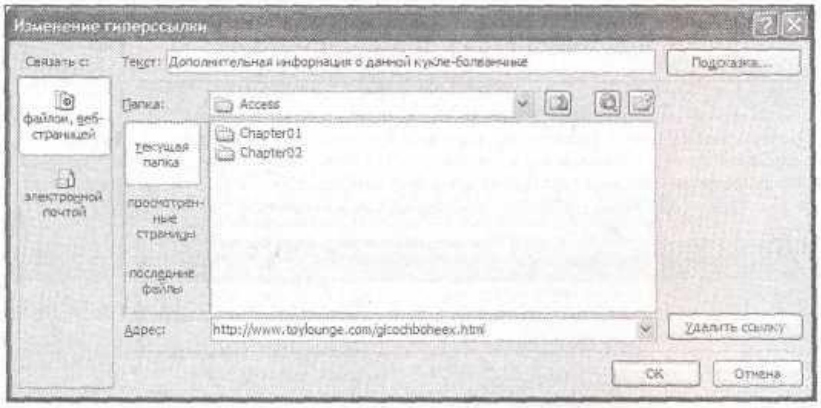


Рис. 2.16. С помощью окна **Изменение гиперссылки** можно изменить текст, появляющийся в ячейке (в верхней части окна), и страницу, которую откроет Access, если вы щелкните ссылку кнопкой мыши (в нижней части окна). Вы также можете создавать ссылки, включающие адреса электронной почты (в этом случае Access откроет программу электронной почты, установленную на вашем компьютере) или ссылки на полное имя файла (с использованием области просмотра папки для выбора нужного файла)

2.3.8. *Вложение*

Тип данных **Вложение** (Attachment) - это новый тип, появившийся в программе Access 2007. Он позволяет вставлять файлы в запись БД почти так же, как вы вкладываете файлы в ваши сообщения электронной почты. Access хранит файлы, вставленные в поле типа Вложение как часть вашей таблицы, встроенную в файл вашей БД.

Тип данных **Вложение** хорошо подходит для вставки в запись изображения, короткого звукового файла или документа из другого приложения пакета Office, такого как Word или Excel. Вы можете создать таблицу **People** (люди) с изображением каждого человека, включенного в список контактов, или каталог изделий с изображением товаров, которые вы пролаете. В этом случае у данных типа **Вложение** - очевидные преимущества, поскольку они хранятся в файле вашей БД, и вы никогда не потеряете их след.

Но данные типа Вложение не так привлекательны в случае больших файлов или файлов, требующих частой корректировки. Если вы поместите часто корректируемый документ в БД Access, он не будет доступен для быстрого редактирования, печати и поиска. Вам придется запустить программу Access и найти соответствующую запись, прежде чем вы сможете открыть ваш документ. Если же нужно внести изменения, вы должны оставить программу Access открытой, чтобы она могла забрать измененный файл и вставить его снова в БД.

Предупреждение

Дважды подумайте, прежде чем связываться с вложенными файлами. Как вы уже знаете, объем, который может занимать БД Access, ограничен двумя гигабайтами. Если вы начнете сохранять большие файлы в ваших таблицах, то можете просто превысить его. Лучше хранить большие документы в отдельных файлах, а затем записывать имя файла в текстовое поле или поле с типом данных **Гиперссылка** (Hyperlink).

Применяя тип данных **Вложение**, убедитесь в том, что задано свойство поля **Подпись** (Caption), определяющее текст, который появляется в заголовке столбца для этого поля. (Часто для заголовка используется имя файла.) Если свойство не задано, в заголовке столбца отображается скрепка, но без текста.

На листе данных поле с типом данных **Вложение** легко узнать, т. к. рядом с ним расположена пиктограмма скрепки (рис. 2.17).

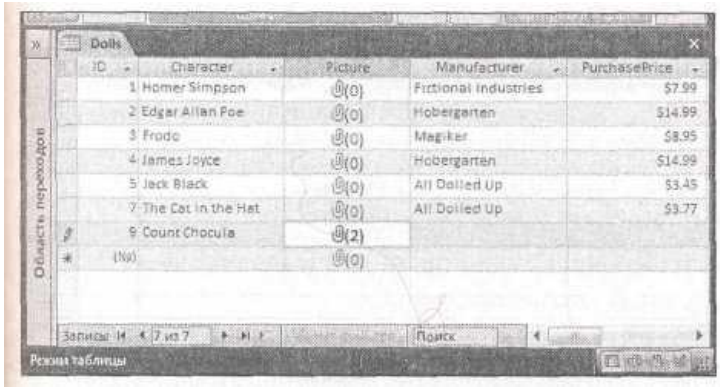


Рис. 2.17. Вложения помечаются пиктограммой скрепки и числом в скобках, сообщающим о количестве вложенных файлов. В данном примере все значения в поле **Picture** с типом данных **Вложение** пустые за исключением Count Chocula, у которого оно равно двум

Для вложения файла или просмотра списка вложенных файлов дважды щелкните кнопкой мыши пиктограмму скрепки. Вы увидите диалоговое окно **Вложения** (Attachments) - рис.2.18.

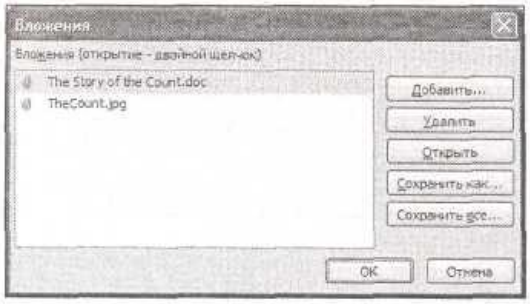


Рис. 2.18. В диалоговом окне Вложения показаны все файлы, связанные с вашим полем

Далее перечислены действия, которые можно выполнить с помощью окна **Вложения** (Attachments).

■ *Вставить новый вложенный файл.* Щелкните мышью кнопку **Добавить** (Add). Затем найдите и укажите новый файл и нажмите кнопку **ОК**. Вы увидите новый файл в конце списка файлов.

■ *Удалить вложение файла.* Выберите в списке нужный файл и щелкните мышью кнопку **Удалить** (Remove).

Сохранить копию вложенного файла. Выберите нужный вложенный файл, щелкните мышью кнопку **Сохранить как** (Save As) и затем укажите место на вашем компьютере для сохранения копии. Или щелкните мышью кнопку **Сохранить все** (Save All) для сохранения копий всех вложенных в это поле файлов. Если вы меняете данные копии, содержимое вложенного файла в вашей БД не меняется.

■ *Редактировать и просматривать вложенный файл.* Выберите вложенный файл и щелкните мышью кнопку **Открыть** (Open). Программа Access скопирует вложенный файл во временную папку па вашем компьютере, ту, в которой сохраняется кэшируемая интернет-информация. Если вы сохраняете файл, Access отслеживает изменения, автоматически обновляет вложенный файл и

затем удаляет временный файл. Если вы закроете окно **Вложения** (Attachments) до того, как закрыли файл, то Access предупреждает о том, что ваши корректировки не будут отражены в вашей БД. На рис. 2.19 показано, что происходит.

К сожалению, у типа данных **Вложение** мало параметров управления. Далее перечислены некоторые ограничения этого типа данных.

- Вы не можете ограничить количество разрешенных вложений файлов в поле типа Вложение. У всех полей этого типа практически нет ограничения на количество вложенных файлов (хотя вы не можете вложить два файла с одним и тем же именем).
- Вы также не можете ограничить типы файлов, предназначенных для вложения.
- Вы не можете ограничить и размер файлов, предназначенных для вложения.

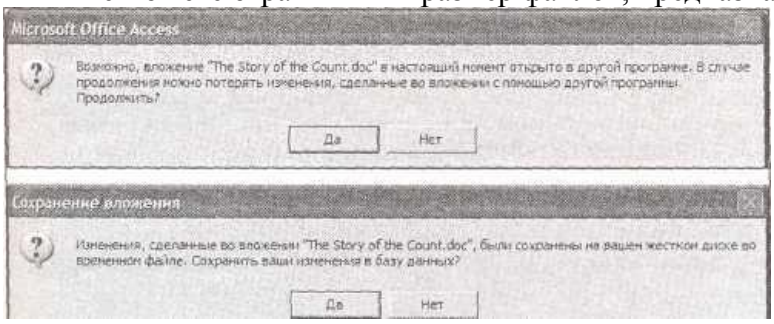


Рис. 2.19. *Вверху:* в данном примере файл "The Story of the Count.doc" все еще открыт.

Если вы продолжите, то все изменения, которые вы вносите (или любые изменения, которые вы внесли к данному моменту *и* не сохранили), не будут отражены в БД.

Внизу: если программа Access замечает, что вы сохраняли ваш файл с тех пор, как открыли его впервые, она спрашивает вас о том, хотите ли вы обновить БД последней сохраненной версией файла. (Для того чтобы избежать подобных треволнений, вкладывайте только те файлы, которые вы не собираетесь редактировать.)

2.3.9. *Счетчик*

Счетчик (AutoNumber) - это специальный тип данных. В отличие от всех других знакомых вам типов данных, в поле типа **Счетчик** нельзя ввести значение. Программа Access делает это автоматически, когда вы вставляете новую запись. Access гарантирует, что значение счетчика уникально - другими словами, программа никогда не присвоит двум записям одно и то же значение типа **Счетчик.**

Примечание

У каждой таблицы может быть не более одного поля **Счетчик.**

Обычно поле типа **Счетчик** выглядит как последовательность чисел - Access стремится дать первой записи значение 1, второй записи значение 2 и т. д. Но истина не так проста. Иногда программа Access пропускает числа. Такой пропуск возможен, когда несколько пользователей одновременно работают с БД, или когда вы начинаете вставлять новую запись, а затем отменяете это действие, нажав клавишу <Esc>. Вы также можете удалить существующую запись, в этом случае Access никогда повторно не использует значение типа **Счетчик** из удаленной записи. В итоге, если вы вставляете новую запись и видите, что ей присвоено значение типа **Счетчик,** равное 401, то не можете с уверенностью сказать, что в таблице уже есть 400 записей. Реальное их количество, возможно, меньше.

Бесспорно, значение **Счетчик** не отображает ничего реального, и, возможно, вы не захотите тратить много времени на его рассмотрение. Единственная задача поля с типом данных **Счетчик** - гарантировать наличие у каждой записи вашей таблицы уникального указателя. Обычно ваше поле типа **Счетчик** служит также и первичным ключом для вашей таблицы, как объясняется в *разд. "Первичный ключ " далее в этой главе.*

Применение поля типа *Счетчик* без раскрытия реального размера вашей таблицы

У значений **Счетчик** есть маленький недостаток: они предоставляют сведения о количестве записей в таблице. Быть может, вы не хотите, чтобы клиент знал, что ваша торговая марка, новая компания, торгующая скульптурными фигурками из масла в духе народных ремесел (Better Butter

Sculptures), "не одурачила" и 12 заказчиков. Поэтому вас смутит необходимость признаться ему в том, что его номер, ID, всего 6.

Лучше всего начать отсчет с большего числа. Вы можете обмануть программу Access, заставив генерировать числа типа **Счетчик,** начиная с заданного минимума. Например, вместо создания номеров клиентов 1, 2 и 3 вы можете создать ID-значения 11001, 11002, 11003. Такой подход также гарантирует наличие у ваших идентификаторов одинакового количества цифр и позволяет разделить ID в разных таблицах, начиная их формирование с различных минимальных значений. К сожалению, для того чтобы реализовать эту хитрость, вам надо обмануть Access с помощью специально разработанного запроса, который вы увидите в *разд. "Получение начальных значений типа Счетчик, отличных от 1" главы 8.*

С другой стороны, вы можете заставить программу генерировать значения типа **Счетчик** иным способом. Есть два варианта.

■ *Случайное значение типа* ***Счетчик.*** Для того чтобы воспользоваться случайными числами, измените свойство поля **Новые значения** (New Values) со значения *Последовательные* (Increment) на значение *Случайные* (Randome). Теперь вы получите длинные номера для каждой записи, такие как 212125691, 1671255778 и -1388883525. Вы можете использовать случайные числа типа **Счетчик** для формирования значений, которые другие люди не смогут угадать. (Например, если у вас есть таблица **Orders** (заказы), в которой применяются случайные числа *в* поле **OrderlD** (идентификатор заказа), их можно использовать как подтверждающие номера (confirmation numbers).) Но в мире Access случайные числа типа **Счетчик** применяются редко.

■ *Коды репликации.* Коды репликаций (Replication ID) - это длинные непонятные коды, например, 38A94E7B-2F95-4E7D-8AF1-DB5B35F9700C, гарантированно уникальные с точки зрения теории вероятностей. Для их применения измените значение свойства **Размер поля** с *Длинного целого* на *Код репликации.* Этот вариант действительно используется только в одном случае - если у вас есть отдельные копии БД и вам в будущем придется объединить данные из них. В следующем разделе объясняется этот сценарий.

Оба этих варианта несколько затуманивают простую **и** понятную концепцию типа данных **Счетчик,** поэтому, прежде чем использовать их в своих таблицах, серьезно оцените необходимость их применения.

Применение типа *Код репликации*

Представьте себе, что вы работаете в компании с несколькими региональными отделами продаж, каждый из которых имеет собственную БД, отслеживающую заказы. Если применять обычное поле типа **Счетчик,** в конце концов, вы получите несколько клиентов с одинаковыми ID, но в разных филиалах. Если вы когда-нибудь захотите сравнить данные, то быстро запутаетесь. И не сможете в будущем объединить данные в общей БД для дальнейшего анализа.

Программа Access предлагает вам другую возможность - *Код репликации.* Код репликации - странное творение - очень длинный идентификатор (всего 16 байтов), представленный строкой цифр и букв, которая выглядит примерно следующим образом:

38A94E7B-2F95-4E7D-8AF1-DB5B35F9700C

Такой идентификатор - более громоздкий по сравнению с обычным целым числом. Помимо всего прочего, гораздо легче поблагодарить кого-либо за отправку заказа Order 4657, чем заказа Order 38A94E7B-2F95-4E7D-8AF1-DB5B35F9700C. Другими словами, если значение типа **Счетчик** применяется для отслеживания и бухгалтерского учета, *Код репликации* использовать для этой цели не стоит.

Но эти коды помогают решить описанную ранее проблему, если многочисленные копии одной и той же БД используются в разных местах. Дело в том, что *Код репликации* гарантирует вероятностную уникальность значений. Другими словами, возможных значений типа *Код репликации* так много, что практически невероятно, что вы создадите одно и то же значение дважды. Следовательно, если у вас даже десятки отдельных копий вашей БД и они управляются сотнями клиентов, вы можете быть уверены в том, что у каждого клиента уникальный идентификатор. Более того, вы можете периодически объединять отдельные таблицы в одной главной БД. (Этот процесс называется *репликацией* и служит причиной появления термина "код репликации". Вы узнаете больше о передаче данных из одной БД в другую в *главе 19.)*

Примечание

Код репликации также называют GUID (globally unique identifier, глобально уникальный идентификатор). В теории вероятность того, что два GUID идентичны, равна 1/2128, величина достаточно маленькая для того, чтобы вы могли заставить работать один биллион сотрудников, создающих не более одного биллиона GUID в год, и все же быть уверенным в отсутствии дубликатов в течение десятилетия или двух. На практике реальным ограничением служит качество используемого в программе Access генератора случайных чисел.

На рис. 2.20 показана таблица, в которой используются коды репликаций.

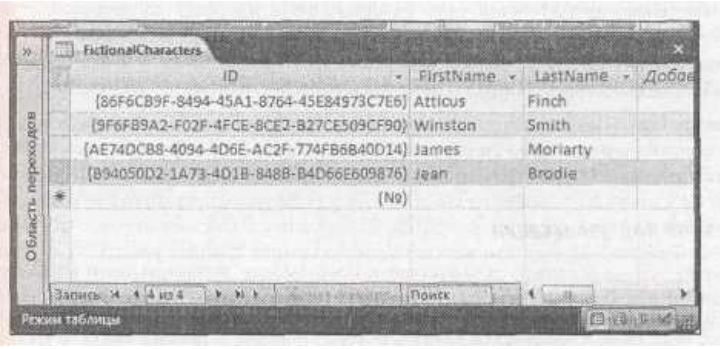


Рис. 2.20. В таблице **FictionalCharacters** показаны 10 записей, каждая с уникальным с вероятностной точки зрения значением типа **Счетчик**

2.4. *Первичный ключ*

В **Конструкторе** можно задать *первичный ключ* (primary key) таблицы, представляющий собой поле (или комбинацию полей), уникальное для каждой записи. У всех таблиц должен быть первичный ключ. Для того чтобы понять важность роли первичного ключа, нужно знать немного больше о принципах работы БД. В *примечании "На профессиональном уровне. Как Access предотвращает дублирование записей "* вы найдете подробный рассказ об этом.

На профессиональном уровне.
Как Access предотвращает дублирование записей

Для того чтобы функционировать корректно, система управления базами данных, такая как Access, должна уметь определять разницу между *каждой* и *любой* записью в вашей таблице. Другими словами, вы не можете вставить две записи с полностью идентичными данными. Чистоплюйство БД общеизвестно, они не терпят такого рода мусора.

Проблема устранения дубликатов не так проста, как кажется. Программа Access спроектирована в расчете на максимальную скорость обработки, и она не может позволить себе перепроверять вашу новую запись, сравнивая ее со всеми другими записями в таблице для выявления дубликата. Вместо этого она полагается на первичный ключ. До тех пор пока у каждой записи в таблице есть уникальный, никогда не повторяющийся первичный ключ, у вас. не будет двух идентичных записей. (В худшем случае, они могут быть почти идентичны, с одинаковыми данными во всех остальных полях, но с разными первичными ключами. Это вполне приемлемо для программы Access.)

В таблице **Employees** (сотрудники) номер социального обеспечения (Social Security number) мог бы служить первичным ключом. Этот метод хорошо работает, поскольку, когда вы вставляете новую запись, Access может проверить наличие дублирования, пробежав список этих номеров, что гораздо быстрее, чем просмотр целой таблицы.

Выбор первичного ключа - непростая задача. Представьте, что у вас есть список друзей (и их контактная информация) в таблице, названной **People** (люди). Рассуждая логически, вы можете решить, что можно создать первичный ключ как комбинацию имени и фамилии.

К сожалению, как раз так и не следует делать - в конце концов, есть масса адресных книг с двумя Шонами- Смитами (Sean Smith).

Лучше всего вставить новую порцию данных. Вы можете пометить каждого человека в вашем списке контактов с помощью уникального ID-номера. Самое лучшее, что вы можете сделать, - дать возможность программе Access создать такой номер для вас (и быть уверенным

в том, что ни у каких двух людей из списка не будет одинаковых номеров) и больше не думать об этом. В этом случае, если у вас есть два Шона Смита, у каждого из них будет свой ID. И даже если Феррис Вил (чертово колесо) Симпсон решит сменить имя, ID останется прежним.

Именно так действует программа Access, когда вы создаете таблицу в **Режиме таблицы**. Рассмотрим таблицу **Dolls**, созданную в *главе 1*. В ней есть поле, названное **Код** (ID) и автоматически заполняемое Access. Вы не можете вставить в таблицу значение поля **Код** или изменить значение в имеющейся записи. Программа Access полностью контролирует это поле, гарантируя уникальный номер для каждой куклы-болванчика. Такой подход в большинстве случаев - как раз то, что вам нужно, поэтому не пытайтесь изменить его или удалить поле **Код.**

Но есть одно исключение. Если вы создаете таблицу в **Конструкторе,** выбрав на ленте **Создание → Таблицы →Конструктор таблиц** (Create → Tables → Table Design), Access считает, что вы знаете, что делаете, и не создает для вас поле **Код.** Вы должны вставить поле Код (или что-то подобное).

2.4.1. *Создание поля для вашего собственного первичного ключа*

Если в вашей БД нет поля **Код** (возможно, вы создали ее, выбрав **Создание → Таблицы → Конструктор таблиц),** ваша задача - создать его и установить первичный ключ. Вот как это делается.

1. Создайте новое поле, вставив его имя в столбец **Имя поля** (Field Name).

Для автоматической генерации значений лучше всего подходит имя **Код** (ID). Некоторые пользователи предпочитают более информативное имя (например, BobbleheadID, CustomerlD и т. д.), но это необязательно.

2. В столбце **Тип данных** (Data Type) выберите тип данных **Счетчик** (Currency).

Выбрав этот тип данных, вы можете быть уверены в том, что программа Access создаст уникальное значение ID для каждой вставляемой вами новой записи. Если этот подход вас не устраивает, можно выбрать что-то другое (например, текстовый тип данных или числовой). В этом случае вы отвечаете за ввод собственного уникального значения для каждой записи, что потребует больше работы, чем кажется на первый взгляд.

3. Щелкните поле правой кнопкой мыши и выберите команду **Ключевое поле** (Primary Key).

Этот выбор помечает поле как первичный ключ для вашей таблицы. Программа Access не допустит повторяющихся значений в этом поле.

Примечание

Если вы хотите включить в первичный ключ несколько полей, вам придется использовать несколько иной подход. Сначала щелкните кнопкой мыши отступ на странице рядом с именем поля, а затем переместите мышь с нажатой кнопкой, чтобы выделить несколько полей. Затем нажмите и удерживайте нажатой клавишу <Shift> и щелкните правой кнопкой выделенные поля. Теперь можно выбрать команду **Ключевое поле** (Primary Key).

На профессиональном уровне.

Почему так важна уникальность

Вы не поймете до конца важность наличия уникального ID-номера у каждой записи, пока не поработаете с более сложными примерами последующих глав. Но одну из причин можно назвать - другие программы, использующие вашу БД, должны однозначно идентифицировать запись.

Для того чтобы понять, в чем проблема, представьте, что вы создали программу для редактирования таблицы **Dolls.** Эта программа начинает с извлечения списка всех ваших кукол-болванчиков, включенных в таблицу. Она выводит на экран этот список для своего пользователя и предлагает ему внести изменения. В этом вся загвоздка - если произведена корректировка, программа должна иметь возможность внести изменения в соответствующую запись в БД.

А для того чтобы это сделать, ей нужна уникальная порция данных, которую можно использовать для определения местонахождения записи. Если вы соблюдали все практические рекомендации по проектированию, описанные ранее, уникальный "адрес" - поле **Код** куклы-болванчика.

2.5. *Шесть правил проектирования БД*

Чем больше власти, тем больше ответственности. Как проектировщик БД вы должны разработать набор таблиц с подходящей структурой. Если вы сделаете это как следует, то сэкономите массу времени и сил в будущем. Хорошо спроектированные БД легко усовершенствовать, с ними легче работать, они приводят к гораздо меньшему числу трудно разрешимых проблем в случае извлечения информации.

Увы, нет рецепта создания идеальной БД. Но ряд рекомендаций может направить вас в нужное русло. В этом разделе вы познакомитесь с наиболее важными из них.

Совет

Разработка хорошей БД - это искусство, требующее опыта. Для достижения наилучших результатов прочтите следующие рекомендации и пробуйте создавать собственные тестовые БД.

2.5.1. *Правило 1. Выбирайте подходящие имена полей*

В программе Access не установлено особых правил для выбора используемых имен полей. Она позволяет включить *64* символа в создаваемое вами имя. Тем не менее имена полей важны. Вы снова и снова вынуждены ссылаться на одни и те же имена, когда создаете формы, формируете отчеты и даже когда пишете код. Поэтому важно выбрать подходящее имя с самого начала.

Далее приведено несколько советов.

■ *Сделайте его простым и коротким.* Имя поля должно быть настолько коротко, насколько это возможно. Длинные имена утомительно набирать, в них больше шансов сделать ошибку, их труднее втиснуть в формы и отчеты.

■ *Пользуйтесь заглавными буквами.* Нельзя сказать, что это "железное" правило, но большинство приверженцев Access делают заглавной первую букву каждого слова (называют этот прием **CamelCase** (контур верблюда)) и затем сливают слова вместе для формирования имени поля. Примерами могут быть **UnitsInStock** (единиц на складе) и **DateOfExpiration** (годен до).

■ *Избегайте пробелов.* В программе Access разрешается в именах полей применять пробелы, но они могут вызывать проблемы. В языке SQL (Structured Query Language, язык структурированных запросов) (язык для работы с БД, которым вы будете пользоваться для поиска данных) пробелы запрещены. Это означает, что вам придется использовать квадратные скобки при упоминании имен полей, содержащих пробелы (например, [Number Of Guests]), а это быстро надоест. Если без пробелов не обойтись, рассмотрите возможность их замены знаком подчеркивания (_).

■ *Будьте последовательны.* Вы можете выбрать одно из двух имен поля **ProductPrice** (цена_товара) и **ProductPrice.** Любой выбор вполне оправдан. Однако не следует смешивать оба подхода к заданию имен в одной БД, т. к. это верный шаг к созданию путаницы

Точно так же, если у вас несколько таблиц с аналогичной информацией (например, поле **FirstName** (имя) в таблице **Employees** (сотрудники) и в таблице **Customers** (клиенты)), используйте для этих полей одно и то же имя.

■*Не включайте в имя поля имя таблицы.* Если у вас есть поле **Country** (страна) в таблице **Customers,** совершенно ясно, что вы имеете в виду страну (Country), в которой живет клиент. Имя поля **CustomerCountry** было бы избыточным.

■*Не используйте для имени поля слово "Name".* Помимо того, что это труднопроизносимое имя, слово "Name" - ключевое в программе Access. Вместо него применяйте такие имена, как **ProductName, CategoryName, ClassName** и т. д. (Это как раз тот случай, когда следует нарушить правило о включении имени таблицы в имя поля.)

Вы также должны, как следует, подумать над именами ваших таблиц. И снова последовательность достойна королевского звания. Например, фанаты БД проводят часы в спорах о том, надо ли использовать множественное число в именовании таблиц (например, что лучше: Клиент или Клиенты). И то и другое хорошо, но постарайтесь выдержать имена всех таблиц в одном ключе.

2.5.2. *Правило 2. Разбивайте ваши данные*

Следите за тем, чтобы не включить слишком большую порцию данных в одно поле. Вы должны хранить в каждом поле элементарную порцию данных.

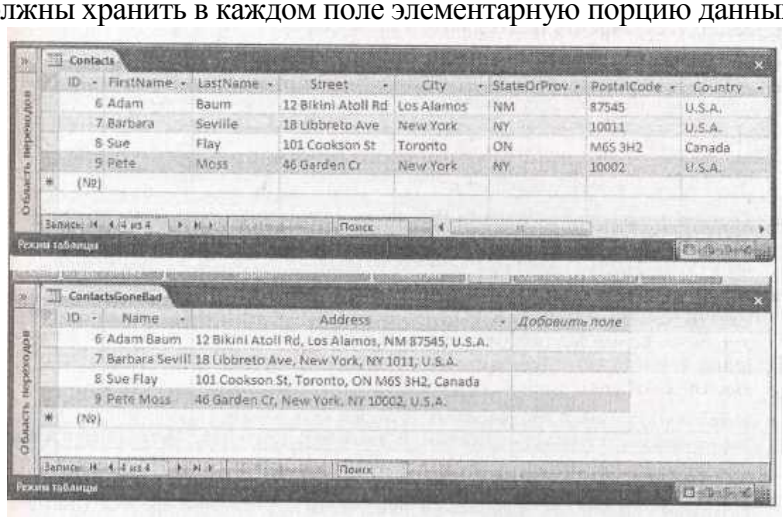


Рис. 2.21. Данный пример демонстрирует правильный способ разделения данных *(вверху)* в таблице **Contacts** (контакты) и неверный способ *(внизу)*. Учтите, что технически все еще возможно разделить данные позже - теоретически сведения об уличном адресе можно разделить на **StreetNumber** (номер дома), **StreetName** (название улицы) и **StreetType** (тип улицы). Но такой подход создает массу сложностей, не давая ничего взамен, поэтому гуру БД редко соглашаются на дополнительные неприятности

Вместо того чтобы создать одно поле **Name** в таблице о контактах, гораздо разумнее вставить два поля: **FirstName** (имя) и **LastName** (фамилия).

Существует множество оснований для разделения информации на отдельные поля. Прежде всего, это устраняет некоторые типы ошибок. В поле **Name** имя можно ввести несколькими разными способами (например, "Фамилия, Имя" или "Имя Фамилия"). Разбиение имени устраняет эти проблемы, способные создать затруднения при попытке использовать данные в автоматизированной задаче какого-либо вида (например, объединение сообщений электронной почты). Но гораздо важнее то, что значительно легче работать с данными, которые разделены на маленькие порции. После разделения поля **Name** на поля **FirstName** и **LastName** вы можете сортировать и искать информацию, основываясь на одной порции этой информации, чего нельзя было бы сделать в противном случае. Аналогично следует разделить сведения об адресе на несколько столбцов, таких как **Street** (улица), **City** (город), **State** (штат) и **Country** (страна) - в этом случае вы гораздо легче найдете всех, кто живет в Нантуките (Nantucket).

На рис. 2.21 показан пример надлежащего разбиения. На рис. 2.21 (внизу) отображена опасная ошибка - попытка хранить несколько порций данных в одном поле.

2.5.3. *Правило 3. Храните все детали в одном месте*

Часто одна таблица применяется в решении многих задач. Вы можете использовать таблицу **Dolls** для выявления дубликатов (и избежать повторной покупки одной и той же куклы-болванчика), для поиска самых старых экземпляров вашей коллекции и для определения общей суммы, потраченной в этом году (для расчета величины налога). Для решения каждой из них нужен слегка отличающийся набор данных. Когда вы подсчитываете потраченную сумму, вам не нужно поле **Character** (персонаж), идентифицирующее куклу. Когда вы ищете дубликаты, вам не нужна информация о дате покупки **(DateAcquired)** или покупной цене **(PurchasePrice).**

Несмотря па то, что вам далеко не всегда нужны все эти поля, совершенно ясно, что есть смысл поместить их все в одну таблицу. Однако, если вы создаете более подробные таблицы, это может быть не столь очевидно. Не трудно представить себе версию таблицы **Dolls,** содержащую 30 или 40 полей данных. Некоторые из них вы можете использовать только от случая к случаю. Но все равно следует все их включить в одну таблицу. Все, написанное в этой книге, доказывает, что вы легко можете отфильтровать всю ненужную вам информацию на листе данных, а также в ваших формах и напечатанных отчетах.

2.5.4. *Правило 4. Избегайте дублирования данных*

Когда вы начинаете заполнять таблицу полями, иногда возникает желание включить в нее несвязанную информацию. Такое включение создает нескончаемые проблемы и в подобную ловушку попасть на удивление легко. На рис. 2.22 показана эта проблема в действии на примере таблицы, пытающейся делать лишнюю работу.

Дублирование данных, подобное показанному на рис 2.22, неэффективно. Легко вообразить таблицу с сотнями похожих записей, бесполезно расходующих дисковое пространство и повторяющих одни и те же значения снова и снова. Но эта неприятность - мелочь, по сравнению с затратами на обновление подобной информации и возможностью возникновения противоречивости данных. Что произойдет, если вы на основании новых научных данных решите изменить сведения о средней продолжительности жизни слона? В соответствии с имеющимся проектом таблицы вам нужно изменить все записи с одними и теми же данными.

Еще хуже то, что очень легко внести изменения в одни записи и оставить нетронутыми другие. Окончательный итог - противоречивые данные - несогласованная информация и нескольких местах таблицы - что делает невозможным извлечение корректных данных.

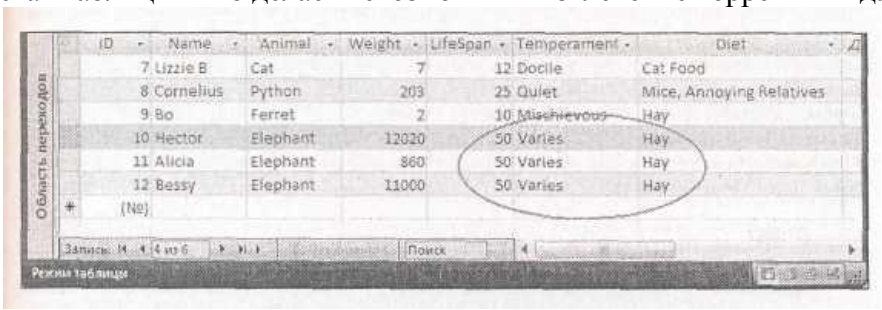


Рис. 2.22. Данная таблица содержит список имеющихся в наличии домашних питомцев у человека, занимающегося разведением редких животных. В ней также приведена некоторая полезная информация о средней продолжительности жизни, характере и пищевых предпочтениях каждого вида животных. Сначала такой проект кажется вполне разумным. Но проблема возникает, как только у вас появляется несколько животных одного вида (в данном случае три слона). Теперь все касающиеся слонов подробности повторяются три раза

Проблема возникает, потому что не вся информация в таблице **Pets** (домашние животные) связана между собой. Для того чтобы понять - почему, нужно погрузиться поглубже в анализ БД.

Как правило, таблица в БД хранит один объект. В таблице **Pets** - это домашние питомцы. Каждое поле в таблице - это порция данных об этом объекте.

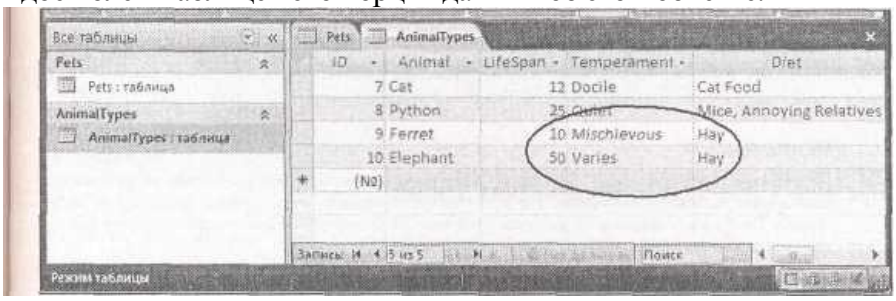


Рис. 2.23. Теперь относящаяся к животным информация хранится в одном месте, без дублирования. Потребуется чуть-чуть больше времени для получения всей необходимой вам информации о животном - например, для того чтобы выяснить продолжительность жизни Беатрис, вам придется проверить запись Elephant (слон) в таблице **AnimalTypes,** но в итоге проект стал более логичным

В таблице **Pets** все поля, такие как **Name** (имя), **Animal** (вид животного) и **Weight** (вес), имеют смысл. Они описывают конкретную особь. Поля же **LifeSpan** (продолжительность жизни), **Temperament** (характер) и **Diet** (рацион) не совсем уместны. Они не описывают конкретного домашнего питомца. В них содержатся стандарты для животных этого вида. Другими словами, эти поля основываются не на вашем животном (как должны были бы) - они основываются на биологическом виде животного. Единственный способ решения проблемы - создание двух таблиц: **Pets** и **AnimalTypes** (виды животных) (рис. 2.23).

Нужен опыт для поиска полей, не связанных с другими полями. В некоторых случаях

дробление таблицы на все более мелкие части не стоит затраченных усилий. Теоретически вы можете извлечь информацию об адресе (содержащую такие поля, как Street, City, Country, PostalCode) из таблицы Customers и поместить ее в таблицу Addresses (адреса). Но два клиента проживают по одному адресу довольно редко, поэтому эта дополнительная работа вряд ли окупится. Мы рассмотрим способы определения связей между таблицами, такими как Pets и AnimalTypes, в *главе 5*.

Совет

Многие специалисты по проектированию БД считают лучшим методом планирования БД - применение учетных карточек (index cards). Для этого запишите все различные типы информации, необходимые для вашей БД. Затем отложите учетную карточку для каждой таблицы, которую планируете использовать. Наконец, возьмите поля из черновика и записывайте их по очереди на соответствующую учетную карточку до тех пор, пока они не разделятся на четкие связанные группы.

2.5.5. *Правило 5. Избегайте избыточной информации*

Другой тип несвязанной информации - избыточные данные - информация, которая уже есть где-то в БД или даже в той же таблице, иногда в слегка иной форме. Как и в случае дублирования данных, такая избыточность может порождать противоречивость данных.

Вычисляемые данные - самый распространенный тип избыточной информации. Примером может служить поле **AverageOrderCost** (средняя стоимость заказа) в таблице Customers. Проблема в данном случае состоит в том, что вы можете определить среднюю стоимость заказа, просмотрев в таблице **Orders** (заказы) все записи для данного клиента, и усреднить их. Вводя поле **AverageOrderCost**, вы создаете возможность хранения в нем некорректных данных (возможно, его значение не будет соответствовать реальным записям о заказах). Кроме того, вы усложняете жизнь, поскольку каждый раз, когда клиент вставляет заказ, нужно пересчитывать среднее значение и обновлять запись клиента.

Примечание

Небожители, проектирующие БД, иногда действительно используют вычисляемые данные для повышения производительности. Но этот тип оптимизации очень редко встречается в БД Access. Он больше характерен для размещенных на серверной стороне коммерческих БД, которыми управляют большие компании или Web-сайты.

Далее перечислены еще несколько примеров избыточной информации.

■ *Поля **Age** (возраст) и **DateOfBirth** (дата рождения) (в таблице **People**).* Обычно вы включаете только поле **DateOfBirth**. Если же у вас их два, то в поле **Age** содержится избыточная информация. Но если у вас только поле **Age**, то вы в опасности - если вы не сможете отслеживать дни рождения и тщательно редактировать каждую запись, ваши данные скоро станут некорректными.

■ *Поле **DiscountPrice** (цена со скидкой) (в таблице **Products**).* Вы должны иметь возможность вычислять цену со скидкой как положено, основываясь на заданных процентах. В обычном деловом мире надбавки и скидки часто меняются. Если вы вычислите скидки, равные 10%, и сохраните измененные цены в вашей БД, вас ждет много работы в случае снижения скидки до 9%.

2.5.6. *Правило 6. Включайте поле Код*

Как вы уже знаете, программа Access автоматически создает поле **Код** (ID), когда вы разрабатываете таблицу в **Режиме таблицы,** и назначает его первичным ключом вашей таблицы. Но даже теперь, когда вы изучили **Конструктор,** все равно вставляйте поле **Код** во все ваши таблицы. Убедитесь, что в нем используется тип данных **Счетчик,** в этом случае Access автоматически заполнит его числами и отведет ему роль первичного ключа.

В некоторых ситуациях в вашу таблицу может быть включено уникальное поле, которое можно использовать как первичный ключ. Не поддавайтесь искушению. Вставляя поле **Код,** вы всегда увеличиваете степень свободы ваших действий. Поле **Код** не придется менять никогда. Другая

информация, даже имена и номера социального обеспечения могут измениться. И если вы используете связи между таблицами, программа Access копирует первичный ключ в другие таблицы. Если первичный ключ меняется, вы вынуждены искать его значение и разных местах БД.

Примечание

Хорошо бы сделать привычкой включение полей Код во все ваши таблицы. В *главе 5* вы поймете преимущества такого подхода, когда начнете устанавливать связи между таблицами.

3. Глава 3. Обработка листа данных: сортировка, поиск, фильтрация и другие действия

В *главе 1* вы впервые познакомились с листом данных - простым и понятным средством для просмотра и редактирования данных таблицы. Как вы уже узнали, лист данных - не лучший инструмент для создания таблицы. (**Конструктор** - более удобное средство для всевозможного управления БД.) Но лист данных - великолепный инструмент для просмотра записей таблицы, внесения изменений и добавления новых данных.

Имея опыт создания таблицы Dolls *(см. разд. "Создание простой таблицы" главы 1),* вы, возможно, чувствуете себя излишне самонадеянно в отношениях с листом данных. Однако большинство таблиц гораздо больше виденных вами до сих пор примеров. В конце концов, если вам необходимо вести учет только дюжины кукол-болванчиков, на самом деле никакая БД не нужна - вы будете вполне удовлетворены, набросав краткий список в старой электронной таблице, документе текстового процессора или на клочке бумаги.

С другой стороны, если вы планируете создать маленькую империю кукол-болванчиков (готовую для демонстрации на международных выставках), придется заполнить вашу таблицу сотнями или тысячами записей. В этом случае не так легко пересмотреть массу данных, чтобы найти то, что нужно. Неожиданно лист данных покажется, мягко говоря, необъятным.

К счастью, у программы Access есть чудодейственные средства управления листом данных, способные сделать жизнь проще. Прочитав эту главу, вы станете знатоком средств, видоизменяющих листы данных, досконально освоив такие методы, как сортировка, поиск и фильтрация. Вы также узнаете, как легко напечатать моментальную копию ваших табличных данных.

Примечание

Время, затраченное на работу с листами данных, - исключительно ваш выбор. Некоторые знатоки программы Access предпочитают создавать формы для всех своих таблиц (как описано в *части IV).* С помощью форм можно создать полностью настраиваемое окно для ввода данных, Проектирование форм требует больше усилий, но ваше творческое эго будет довольно.

3.1. *Настройка листа данных*

Бас утомил тусклый лист данных с монотонно тянущимися рядами столбцов и строк? Вы можете кое-что сделать с ним. Программа Access позволяет настроить внешний вид и структуру листа данных, чтобы сделать его более удобным (или отвечающим вашему чувству стиля). Некоторые из этих настроек, например, изменение шрифта на листе данных, - постыдные излишества. Другие параметры, например скрытие или закрепление столбцов, могут действительно облегчить работу с большими таблицами.

Примечание

Access сохраняет изменения форматирования не сразу (в отличие от корректировок записей, которые программа сохраняет, как только они произведены). Вместо этого Access предлагает сохранить изменения при следующем закрытии листа данных. Вы можете выбрать **Да** для сохранения ваших настроек или **Нет** для возвращения к предыдущему внешнему виду листа данных (что не повлияет на любые внесенные вами в таблицу корректировки данных).

3.1.1. *Форматирование листа данных*

Программа Access позволяет выбрать для листа данных яркие цвета и выразительные шрифты. Повлияют ли эти изменения на функционирование листа данных? Конечно, нет. Но если рабочий стол вашего компьютера больше похож на фестиваль духовного возрождения в стиле 60-х, а не на обычный офисный терминал, возможно, это доставит вам удовольствие.

Параметры форматирования можно найти на вкладке ленты Главная (Home) в группе Шрифт (Font) - рис. 3.1.

Рис. 3.1. Группа **Шрифт** на вкладке **Главная** позволяет изменить шрифт и цвета на всем листе данных. Наиболее полезное средство - возможность отключения некоторых или всех линий сетки и применение чередующихся цветов для выделения соседних строк

Любое внесенное вами изменение форматирования влияет на всю таблицу. Вероятно, вы считаете отличной идеей возможность по-разному форматировать различные столбцы, но Access не предоставляет такой возможности. Если вас это огорчает, вы, безусловно, обрадуетесь, познакомившись с формами и отчетами. Они более трудны в настройке, но обладают более развитыми средствами форматирования.

Примечание

Есть и другой способ использования группы **Шрифт** (Font), расположенной на вкладке ленты **Главная** (Home). Если у вас есть поле с типом данных **Поле MEMO** (Memo) и вы настроили его на хранение форматированного текста (rich text) *(см, разд. "Форматированный текст" главы 2),* можно выделить фрагмент текста в вашем поле и изменить его формат с помощью ленты.

Малоизвестная или недооцененная возможность.

Настройка всех листов данных

Программа Access разрешает форматировать таблицы поочередно. Поэтому, если вы обнаружили настройку, которая вам действительно нравится, вам придется применять ее по очереди в каждой таблице вашей БД.

Но, настроив программу Access, вы можете установить параметры форматирования таким образом, что они автоматически будут применяться к каждой таблице всех БД. Для того чтобы воспользоваться этим приемом, выполните следующие действия:

1.Нажмите кнопку **Office,** а затем кнопку **Параметры Access** (Access Options) для вывода на экран одноименного окна.

*2.*В списке слева выберите **Таблица** (Datasheet).

3.Справа появятся стандартный шрифт, цвет, сетка и ширина колонки - параметры, которые можно менять, как вам захочется.

Когда вы корректируете установочные параметры форматирования в окне **Параметры Access** (Access Options), меняются значения по умолчанию, используемые программой Access. Эти установки определяют форматирование, используемое программой в новых таблицах и в любых таблицах, которые не переформатированы пользователем. Когда вы изменяете параметры форматирования, вы переопределяете установки по умолчанию, независимо от их конкретных значений.

Если вы задали в программе Access использование шрифта красного цвета, но переформатировали конкретную таблицу для вывода текста зеленого цвета, у зеленого цвета более высокий приоритет. Но если вы установили желтый фон в окне **Параметры Access** и не изменяли эту характеристику в вашей таблице, она автоматически получает стандартный желтый фон.

3.1.2. *Реорганизация столбцов*

Поля на листе данных располагаются слева направо в том порядке, в каком вы их создали. Часто выясняется, что этот порядок следования столбцов не самый удобный для ввода данных.

Предположим, что вы создали таблицу Customers (Клиенты) для компании по производству новинок макаронных изделий. Когда на вашей стойке завершается регистрация нового заказчика, выясняется, что регистрационная форма начинается с имени и адреса, а затем уже идут вкусы заказчика, касающиеся пасты. К сожалению, поля на листе данных следуют в совершенно другом порядке. Слева направо они расположены таким образом: ID (код), FreshPastaPreference (предпочитаемый сорт новой пасты), DriedPastaPreference (предпочитаемый сорт сухой пасты), FirstName (имя), LastName (фамилия), Street (улица), City (город), State (штат), Country (страна). (Подобная организация не так глупа, как кажется, - она помогает людям, выполняющим заказы на пасту, быстро находить нужную информацию.) Но из-за такой организации вы вынуждены перемещаться вперед и назад для того, чтобы ввести данные одной регистрации.

К счастью, вы можете решить эту проблему без перепроектирования таблицы. Переместите столбцы, которые вы хотите передвинуть, на новые позиции, как показано на рис. 3.2.

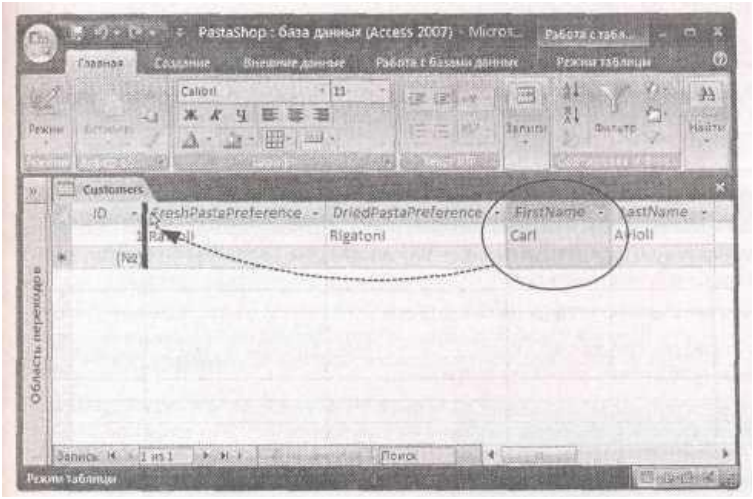


Рис. 3.2. Для переноса столбца щелкните один раз кнопкой мыши его заголовок, чтобы выбрать столбец. Затем с нажатой кнопкой мыши передвиньте столбец на новое место. В данном примере поле **FirstName** предполагается переместить так, чтобы оно располагалось слева от поля **FreshPastaPreference**

Лучшее в таком подходе - отсутствие необходимости изменять реальную структуру БД. Если после перемещения нескольких столбцов вы перейдете в **Конструктор,** то увидите, что порядок полей не изменился. Другими словами, у вас есть возможность, сохраняя физический порядок следования полей (в файле вашей БД), реорганизовать порядок их отображения в **Режиме таблицы.**

Совет

Реорганизация столбцов - довольно незначительное изменение. Смело передвигайте столбцы для того, чтобы удобнее было редактировать, и затем возвращайте их обратно после внесения изменений. Ваши действия не повлияют на данные в БД. Если определенный порядок следования столбцов нужен для одноразовой работы, просто не сохраняйте этот порядок, когда будете закрывать лист данных.

3.1.3. *Изменение размеров столбцов и строк*

По мере заполнения таблицы данными ваш лист данных становится все шире и шире. Часто вас будет огорчать то, что одни столбцы съедают больше пространства, чем им нужно, а другие невозможно узки.

Как вы догадываетесь, программа Access позволяет изменять ширину столбцов. Но, возможно, вы не знаете, что для этого существует множество различных способов.

■ *Изменение ширины одного столбца.* Поместите указатель мыши на правый край столбца. Передвиньте мышь с нажатой левой кнопкой влево (для сжатия столбца) или вправо (для его расширения).

■ *Изменение ширины столбца в соответствии с объемом содержащихся в нем данных.* Щелкните дважды кнопкой мыши край столбца. Программа Access увеличит ширину столбца настолько, чтобы в него поместилось имя поля или самое большое значение (в зависимости от того, что длиннее). Но при этом столбец не выйдет за границы окна программы.

■ *Изменение ширины нескольких смежных столбцов.* Переместите мышь с нажатой лесой кнопкой вдоль заголовков нужных столбцов, чтобы выделить их все. Затем, не отпуская левой кнопки мыши, переместите ее влево или вправо. Все выделенные столбцы сожмутся или расширятся, используя равные доли имеющегося свободного пространства.

■ *Точное задание ширины столбца.* Щелкните правой кнопкой мыши заголовок столбца и выберите команду **Ширина столбца** (Column Width). Вы увидите одноименное диалоговое окно, в котором можно задать точное числовое значение ширины (рис. 3.3).

Рис. 3.3. В окне **Ширина столбца** можно задать точное числовое значение ширины. (У числа в действительности нет конкретного значения - предполагается, что это ширина в символах, но поскольку современная программа Access использует пропорциональные шрифты, у разных символов разная ширина.) Вы также можете установить флажок **Стандартная** для того, чтобы вернуть стандартную ширину столбцу, или щелкнуть мышью кнопку **По ширине данных** для расширения столбца в соответствии с его содержимым (аналогично двойному щелчку мышью края столбца)

Примечание

Имейте в виду, что столбец не должен быть широк настолько, чтобы отобразить все содержащиеся в нем данные сразу. Вы можете прокрутить длинное текстовое поле с помощью клавиш со стрелками, а если это слишком утомительно, использовать сочетание клавиш <Shift>+<F2> для отображения всей информации текущего поля в окне **Область ввода** (Zoom).

Также как ширину столбцов, можно менять и высоту строк. Отличие заключается в том, что программа Access формирует все строки одного размера. Поэтому когда вы делаете одну строку выше или ниже, Access изменяет все остальные строки до соответствующей высоты.

Сжать строки хочется главным образом для того, чтобы вывести больше строк на экран. Увеличивают высоту строк чаще всего для того, чтобы вывести несколько строк текста в каждом текстовом поле (рис. 3.4).

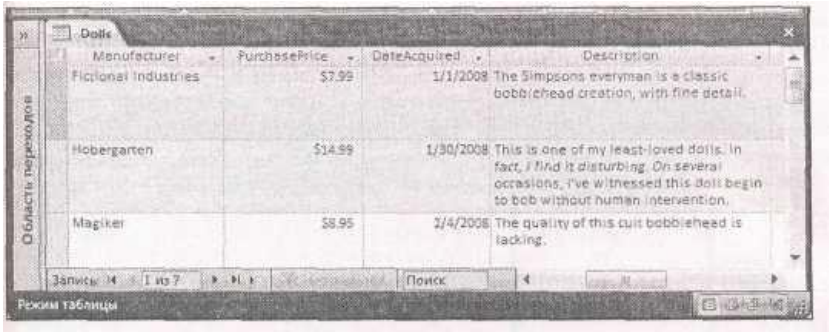


Рис. 3.4. Если высота строки достаточна, программа Access разбивает текст внутри поля на несколько строк, как показано в столбце **Description**

3.1.4. *Скрытие столбцов*

У большинства таблиц так много столбцов, что вы не можете все их вывести на экран одновременно. Это один из недостатков листа данных, и зачастую вы вынуждены прокручивать страницу туда-сюда.

Но иногда вам не нужно видеть все столбцы сразу. В этом случае можно скрыть на время столбцы, которые вас не интересуют, чтобы сосредоточиться на важных для вас подробностях, не

отвлекая внимания. Первоначально все вставленные вами поля отображаются на экране открытыми.

Для того чтобы скрыть столбец, выделите его, щелкнув кнопкой мыши по заголовку. (Можно выбрать несколько смежных столбцов, щелкнув кнопкой мыши первый заголовок и с нажатой кнопкой проведя мышью по заголовкам всех остальных.) Затем щелкните правой кнопкой выделенные столбцы и выберите команду меню **Скрыть столбцы** (Hide Columns). Столбец тут же исчезнет с листа данных. (Это внезапное исчезновение может слегка обескуражить новичков.)

К счастью, с полем и его данными ничего не случится. Для того чтобы вернуть столбец на экран, щелкните правой кнопкой мыши заголовок любого столбца и выберите команду **Отобразить столбцы** (Unhide Columns). Программа Access выведет на экран диалоговое окно Отображение **столбцов** (Unhide Columns) - рис. 3.5.

Примечание

Под списком полей есть элемент, названный **Добавить поле** (Add New Field). На самом деле это не настоящее поле, а заполнитель, появляющийся справа от последнего поля на листе данных, который можно использовать для вставки новых полей *(см. разд. "Организация и описание ваших полей" главы 2).* Если вы привыкли вставлять поля в **Конструкторе** *(см. разд. "Создание простой таблицы" главы 1),* то можете скрыть этот заполнитель и получить дополнительное свободное пространство.

Если вы вставляете новую запись, когда столбцы скрыты, то не можете вставить значения в такие поля. Значение остается пустым или заданным по умолчанию (если вы определили такое для данного поля, как описано в *разд. "Задание значений по умолчанию" главы* 4). Если вы скрыли обязательное поле *(см. разд. "Запрет незаполненных полей" главы 4),* при попытке вставить запись выводится сообщение об ошибке. Вам ничего не остается, как отобразить соответствующий столбец, а затем заполнить его пропущенными данными.

Рис. 3.5. С помощью этого окна можно вернуть скрытые столбцы и (как не удивительно) скрыть те, что отображаются в настоящий момент. Все столбцы с установленным флажком, расположенным рядом, видимы, а все со сброшенным флажком скрыты. Как только вы измените видимость столбцов, программа Access обновит внешний вид листа данных. Добившись удовлетворяющего вас результата, щелкните мышью кнопку **Закрыть** для того, чтобы вернуться на лист данных

3.1.5. *Закрепленные столбцы*

Даже имея возможность скрывать столбцы или изменять их ширину, в типичной таблице вы иногда вынуждены пользоваться прокруткой. В такой ситуации легко потерять ориентацию. Вы можете прокрутить таблицу **Contacts** (Контакты), чтобы увидеть дополнительную информацию и забыть, какого именно человека вы проверяете. У программы Access есть еще одно средство, помогающее обеспечить постоянное отображение важной информации, - *закрепленные столбцы.*

Закрепленный столбец всегда остается зафиксированным в левой части окна Access. Даже если вы прокручиваете таблицу вправо, все закрепленные столбцы остаются видимыми (рис. 3.6). Для того чтобы закрепить столбец (или столбцы), выделите их, щелкните правой кнопкой мыши заголовок столбца и выберите команду **Закрепить столбцы** (Freeze Columns).

Совет

Если вы хотите закрепить несколько несмежных столбцов, начинайте с самого левого. Затем повторите процесс закрепления для столбца, расположенного справа от первого, и т. д.

Закрепленные столбцы всегда должны располагаться с левой стороны таблицы. Если вы закрепляете столбец, размещенный в произвольном месте таблицы, программа Access переносит его к левому краю и закрепляет. Вы можете вернуть его на место после снятия закрепления, используя метод реорганизации столбцов, описанный в *разд. "Реорганизация столбцов" ранее в этой главе.* Помните о том, что пока столбец закреплен, вы не можете с помощью мыши переместить его в другое место.

Для освобождения столбцов щелкните правой кнопкой мыши заголовок столбца и выберите команду **Освободить все столбцы** (Unfreeze All Columns).

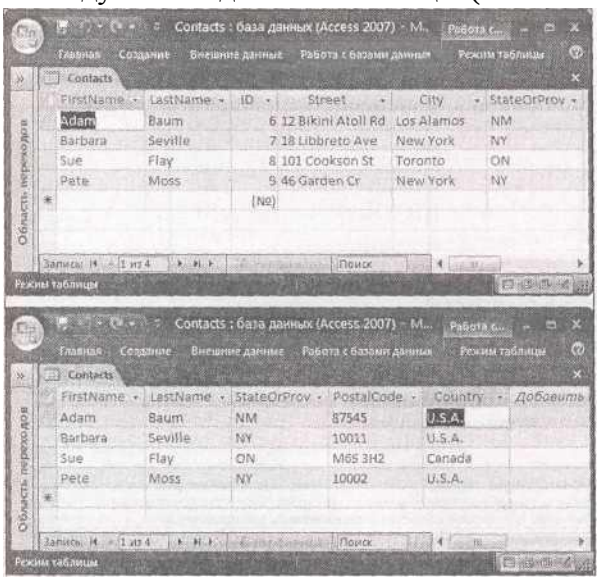


Рис. 3.6. Вверху: в данном примере **поля FirstName** и **LastName** закреплены. Они отображаются в исходном положении слева. (На этом рисунке для получения дополнительного свободного пространства лента свернута.) *Внизу:* когда вы прокручиваете таблицу в горизонтальном направлении, чтобы увидеть больше информации, столбцы **FirstName** и **LastName** сохраняют свое положение

Примечание

В конце концов, вы обнаружите, что предлагаемых настроек листа данных недостаточно или что вам необходимо настроить одну и ту же таблицу по-разному для разных людей. Это значит, что надо переходить к формам, более совершенному варианту отображения, описанному в *части IV*.

3.2. *Перемещение в таблице*

В *главе 1* вы познакомились с основными способами перемещения в таблице. С помощью мыши и нескольких клавиш выделения можно многого добиться (см. табл. 1.1, в которой представлен список клавиш, которые можно использовать для перехода с места на место и выполнения корректировок).

Но несколько приемов вы все еще не видели. Один из них - сберегающие время кнопки переходов от записи к записи, расположенные в нижней части листа данных (рис. 3.7).

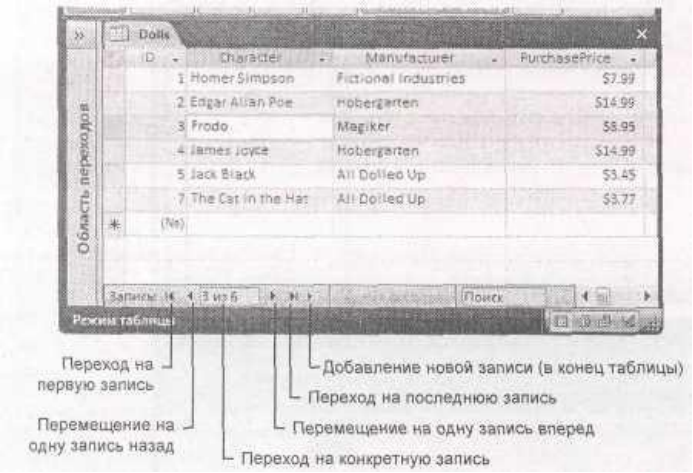


Рис. 3.7. Эти малозаметные кнопки помогают переходить к началу и концу таблицы или, что гораздо интереснее, прямо к записи с заданной позицией. Для этого введите номер записи (например 4) в поле (в котором в данном примере стоит строка "3 из 6") и затем нажмите клавишу <Enter>. Конечно, этот прием работает, только если вы представляете, где а вашей таблице находится запись

Несколько дополнительных средств помогут вам ориентироваться при обработке больших объемов данных. К ним относятся сортировка (которая упорядочивает записи так, что вы можете видеть нужную информацию), фильтрация (которая ограничивает вывод на экран данных, отображая только интересующие вас записи) и поиск (который извлекает только определенные записи из огромной массы данных). Вы опробуете все эти средства в следующих разделах.

3.2.1. *Сортировка*

В некоторых случаях легче составить представление о большом объеме данных, если их упорядочить. Можно систематизировать список клиентов в соответствии с их фамилиями, каталог изделий в зависимости от их цены, а список свадебных гостей по возрасту и т. д.

Для сортировки записей выберите столбец, который хотите использовать для упорядочивания записей. Щелкните кнопкой мыши стрелку, направленную вниз, у правого края заголовка столбца и выберите один из вариантов сортировки в верхней части меню (рис. 3.8).

Как объясняется в табл. 3.1, предлагаемые варианты сортировки зависят от типа данных поля. (Вы также можете применить те же самые варианты сортировки, используя команды на вкладке лепты Главная (Home) в группе **Сортировка и фильтр** (Sort & Filter).)

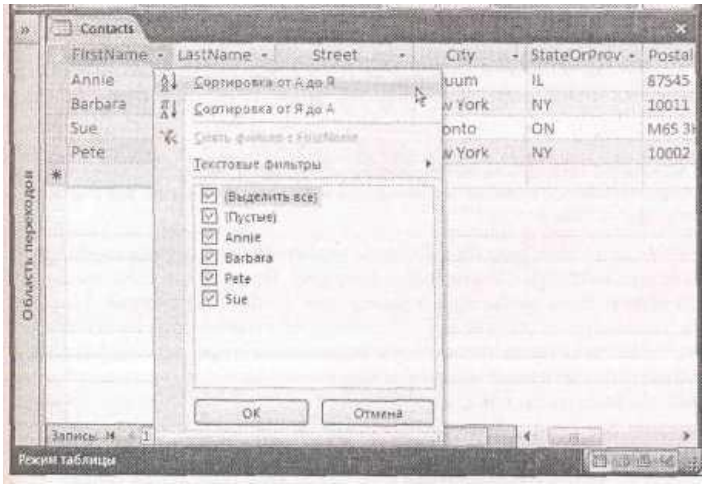


Рис. 3.8. Это текстовое поле можно отсортировать в алфавитном порядке от начала к концу алфавита (от А до Я) или от конца к началу (от Я до А). В меню также есть варианты фильтрации, которые описаны далее

Таблица 3.1. варианты сортировки для различных типов данных

Тип данных	*Варианты сортировки*	*Описание*
Текстовый, Поле MEMO и Гиперссылка	Сортировка от А до Я, сортировка от Я до А	Выполняет сортировку в алфавитном порядке (как в словаре), упорядочивая букву за буквой. Сортировка не зависит от регистра, поэтому "чепуха" и "Чепуха" считаются одинаковыми словами
Числовой, Денежный и Счетчик	Сортировка по возрастанию, от самого маленького значения к самому большому. Сортировка по убыванию, от самого большого значения к самому маленькому	Выполняется числовая сортировка, помещающая меньшие числа в начало или в конец
Дата/время	Сортировка по возрастанию от самой ранней даты к самой поздней.	Выполняется временная сортировка, разделяющая более ранние даты (которые наступили первыми) от более поздних

	Сортировка по убыванию от самой поздней даты к самой ранней	
Логический	Сортировка от установленных к сброшенным. Сортировка от сброшенных к установленным	Отделяет установленные флажки от сброшенных

В неотсортированной таблице записи упорядочены в соответствии со временем их создания, поэтому самые старые записи находятся в верхней части листа данных, а самые новые - в нижней его части. Сортировка не влияет на способ хранения записей программой Access, но действительно изменяет способ их отображения.

Совет

Для того чтобы вернуть таблицу к первоначальному неотсортированному состоянию, воспользуйтесь последовательностью **Главная → Сортировка и фильтр → Очистить все сортировки** (Home → Sort & Filter → Clear All Sorts).

Сортировка - это одноразовое действие. Если вы редактируете отсортированный столбец, программа Access не выполняет повторную сортировку. Представьте себе, что вы сортируете список по имени. Если затем вы исправите имя в одной из записей, например Фрэнки на Чен, Access не переместит запись в группу записей с именами, начинающимися с буквы "Ч". Измененная строка останется на своем исходном месте до тех пор, пока вы не пересортируете таблицу. Аналогичным образом любые новые записи, которые вы вставляете, остаются в конце таблицы до следующей сортировки (или следующего открытия таблицы). В таком поведении есть смысл. Если бы программа Access изменяла местоположение строки, как только вы внесли в нее изменения, это очень быстро нас дезориентировало бы.

Примечание

Порядок сортировки - одна из подробностей, которую программа Access сохраняет **в** файле БД. Когда вы в следующий раз откроете таблицу в **Режиме таблицы,** Access автоматически применяет заданные вами параметры сортировки.

На профессиональном уровне.

Числа и специальные символы в текстовых полях

Результат сортировки текста иногда противоречит интуитивным предположениям, особенно если в вашем текстовом поле содержатся числовые данные.

Обычно, когда сортируются два числа (например, 153 и 49), они упорядочиваются от меньшего к большему (49, 153). Но текстовая сортировка действует иначе. Когда программа Access сортирует текст, она проверяет его символ за символом, что означает сортировку чисел по первой цифре. Если первые цифры одинаковы, проверяется вторая цифра и т. д. В результате если числа 49 и 153 сортируются в алфавитном порядке, то вы получите 153, 49, поскольку 4 (первая цифра числа 49) больше 1 (первой цифры числа 153).

Жизнь становится еще интереснее, если в эту мешанину добавить знаки пунктуации и другие специальные символы. Далее приведен порядок, в котором программа Access сортирует любые символы (при стандартной сортировке по возрастанию от А до Я или от A до Z):

1. Пустые значения.
2. Пробелы.
3. Специальные символы (например, знаки пунктуации).
4. Буквы.
5. Числа.

Сортировка по нескольким полям

Если при сортировке обнаруживаются два одинаковых значения, нет способа определить порядок их следования (по отношению друг к другу). Если вы сортируете список клиентов, в котором есть два Вана Хаузера (Van Hauser), вы можете быть уверены, что сортировка по фамилии выведет их друг за другом, но неизвестно, кто будет первым, а кто вторым.

Если вы хотите узнать больше о том, как Access интерпретирует дубликаты, нужно выбрать сортировку, основанную на нескольких столбцах. Отличным примером может служить телефонная книга, в которой люди отсортированы по фамилиям, а затем по именам. Таким образом, люди с одинаковой фамилией группируются вместе и упорядочиваются в соответствии с их именами, например, следующим образом:

…
Smith, Star
Smith, Susan
Smith, Sy
Smith, Tanis
…

В таблице результаты сортировок накапливаются, что означает возможность одновременной сортировки, базирующейся на нескольких полях. Единственная сложность - правильно задать порядок сортировки. Далее перечислены необходимые действия.

1, Выберите на ленте **Главная → Сортировка и фильтр → Очистить все сортировки** (Home → Sort & Filter →Clear All Sorts).

Программа Access вернет таблицу к исходному не отсортированному состоянию.

2. Воспользуйтесь раскрывающимся контекстным меню столбца для применения к дубликатам дополнительной сортировки.

Если нужно отсортировать телефонную книгу (абоненты упорядочены по фамилиям, а затем по именам), надо включить сортировку по полю **FirstName** (имя). В табл. 3.1 поясняется сортировка, которую вы видите в зависимости от типа данных.

3, Используйте контекстное раскрывающееся меню для применения сортировки первого уровня.

В случае телефонной книги - это поле **LastName** (фамилия).

Вы можете повторить эти шаги для выполнения сортировок по нескольким полям. Представьте, что у вас чудовищно большой набор имен, включающий людей с одинаковыми фамилиями и именами. В этом случае вы могли бы добавить третью сортировку по первой букве второго имени (или отчеству). Для применения этой сортировки следует включить сортировку в следующем порядке: **Middlelnitial** (первая буква второго имени), **FirstName** (имя), **LastName** (фамилия). Вы получите следующий результат:

…
Smith, Star
Smith, Susan K
Smith, Susan P
Smith, Sy
…

3.2.2. *Фильтрация*

В таблице с сотнями или тысячами записей прокрутка таблицы вперед и назад на листе данных так же успокаивает, как звук пневматической дрели в 3 часа утра. Иногда вам даже не нужно видеть все записи сразу - они лишь вызывают усталость пальцев и отвлекают от тех данных, которые вас действительно интересуют. В этом случае вам следует урезать лист данных с помощью *фильтрации,* оставив только интересующие вас записи.

Для фильтрации записей вы задаете условие, которому должна удовлетворять запись для того, чтобы быть включенной в лист данных. Например, интернет-магазин может выбирать пищевые продукты из полного каталога товаров, компания по доставке может искать заказы, сделанные на прошлой неделе, а служба свиданий (dating service) может выискивать холостяков, живущих отдельно от своих родителей. Когда применяется условие фильтра, в конце концов, скрываются все записи, не удовлетворяющие вашим требованиям. Они по-прежнему остаются в таблице, но ловко скрыты от глаз.

Программа Access может применять фильтры несколькими методами. В следующих разделах вы начнете знакомство с простейшего способа и будете постепенно осваивать более сложные варианты.

Обычный фильтр

Обычный фильтр (quick filter) позволяет выбрать те значения, которые вы хотите включить, и те - которые нужно скрыть, основываясь на текущих данных вашей таблицы. Для применения обычного фильтра выберите столбец, который вы хотите использовать, и затем щелкните мышью направленную вниз стрелку у правого края заголовка столбца. Вы увидите список всех конкретных значений в данном столбце. Первоначально флажок, расположенный у каждого значения, установлен. Сбросьте флажок у тех записей, которые вы хотите скрыть. На рис. 3.9 показан пример, в котором сортировка и фильтр применяются одновременно.

Примечание

Для удаления фильтров из столбца (и отображения всех записей на листе данных) щелкните мышью раскрывающуюся кнопку у правого края заголовка столбца и выберите команду **Снять фильтр** (Clear Filter).

Не все типы данных поддерживают фильтрацию. К типам данных, ее поддерживающим, относятся **Числовой, Денежный, Счетчик, Текстовый, Гиперссылка, Дата/время** и **Логический. Поля MEMO** не поддерживают обычные фильтры (поскольку, как правило, их значения слишком велики и не помещаются в раскрывающемся списке), но другие типы фильтров они поддерживают.

Обычные фильтры можно применять в нескольких столбцах. Порядок применения фильтров не важен, поскольку все фильтры кумулятивные, т. е. вы увидите только те записи, которые удовлетворяют условиям всех установленных фильтров. Обычные фильтры можно использовать и в комбинации с другими вариантами фильтрации, описанными в следующих разделах. Для удаления фильтров выберите на ленте **Главная → Сортировка и фильтр → Удалить фильтр** (Home → Sort & Filter → Remove Filter).

Совет

Обычные фильтры работают лучше всего, если у вас относительно немного различных значений. Отбор людей в зависимости от штата их проживания - прекрасный выбор, как и в зависимости от политической партии, которую они поддерживают, или их любимого цвета. Фильтр будет работать не так эффективно, если вы решите отфильтровать список, основываясь на дате рождения, росте или весе, поскольку существует гигантский диапазон возможных значений этих характеристик. (Это не значит, что вы должны совсем отказаться от фильтрации - просто нужно использовать другой тип фильтра.)

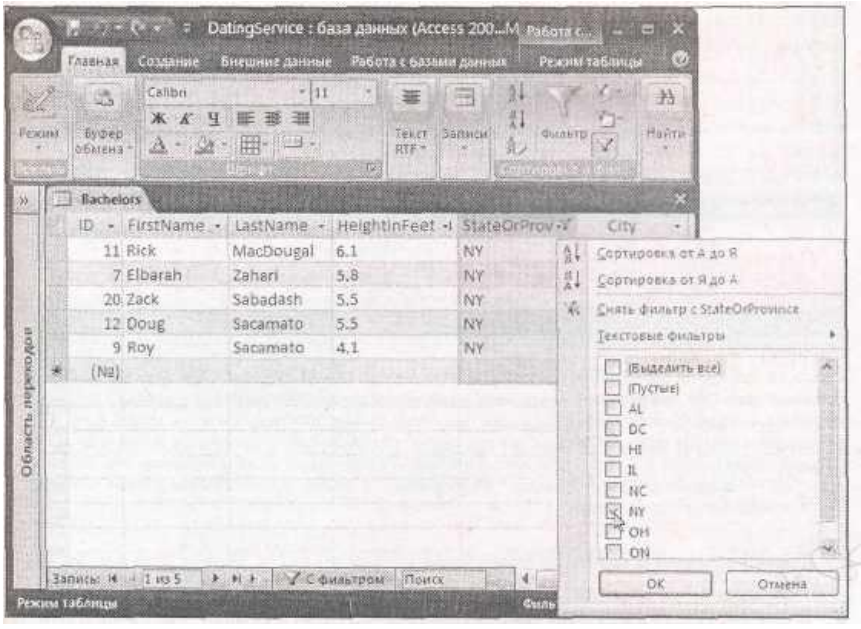


Рис. 3.9. Этот список подходящих холостяков сначала сортируется по росту (в порядке убывания) и затем отфильтрован с сохранением только перспективных кандидатов, проживающих в штате Нью-Йорк. Установленный флажок

свидетельствует о том, что записи с данным значением будут включены в лист данных. Остальные записи скрыты от глаз

Фильтр по выделенному

Фильтр по выделенному (filter by selection) позволяет применять фильтр, базирующийся на любом значении в вашей таблице. Этот тип фильтра очень удобен, если вы нашли точный тип записи, который хотите включить в выборку или исключить из нее. Применяя фильтр по выделенному, вы можете превратить текущее значение в фильтр без прочесывания списка фильтра.

Вот как это делается. Сначала найдите значение, которое хотите использовать для фильтрации листа данных. Щелкните его правой кнопкой мыши и выберите один из вариантов фильтрации в нижней части меню (рис. 3.10).

Все типы данных, поддерживающие фильтрацию, позволяют отбирать точные совпадения. Но многие из них предоставляют дополнительные варианты фильтрации, отображаемые в контекстном меню, которое вызывается щелчком правой кнопки мыши.

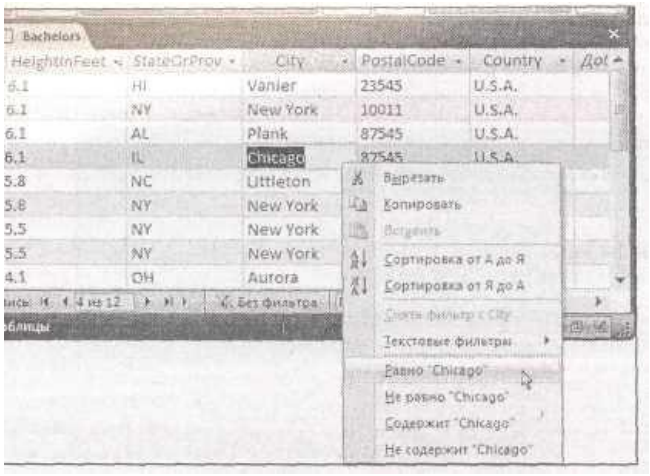


Рис. 3.10. Варианты фильтрации для разных типов данных слегка отличаются. В случае текстового поля (как поле **City,** показанного на рисунке) у вас есть возможность включить в выборку только те записи, которые соответствуют текущему значению **(Равно "Chicago"),** или те, которые не совпадают с ним **(Не равно "Chicago").** У вас также появляются дополнительные возможности, расширяющие отбор с помощью обычного фильтра, - а именно вы можете включить или исключить поля, просто содержащие текст "Chicago". Условию такого фильтра удовлетворяют такие значения, как "Chicago land" и "Little Chicago"

Далее перечислены варианты фильтрации для разных типов данных, которые вы можете увидеть.

- *Текстовые типы данных.* Вы можете отбирать значения с точным совпадением или значения, содержащие фрагмент текста.
- *Числовые типы данных.* Вы можете отфильтровать точно совпадающие значения или числа, которые меньше или больше текущего значения.
- *Типы данных **Дата/время.*** Вы можете отфильтровать точно совпадающие значения или даты, более ранние или более поздние по сравнению с текущей датой.

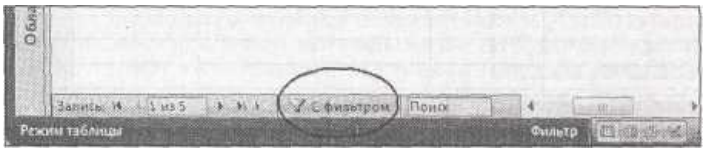


Рис. 3.11. Рядом с элементами управления в нижней части листа данных есть индикатор **С фильтром/Без фильтра,** оповещающий о применении фильтра. Его можно также использовать для быстрого включения или отключения вашего фильтра - одинарный щелчок мышью по индикатору удаляет все фильтры, а повторный щелчок применяет еще раз самый последний набор фильтров

Наконец, для получения дополнительных навыков можно создать условие фильтрации,

использующее только часть значения. Если у вас есть значение "Great at darts" (великолепно играет в дартс) в поле **Description** (описание) вашей таблицы, можно выделить текст "darts" и затем щелкнуть правой кнопкой мыши только этот текст. Теперь вы можете найти другие записи с нолями, содержащими слово "darts". Благодаря этой возможности фильтр по выделенному получил свое название.

Программа Access моментально включает или отключает фильтрацию, как показано на рис. 3.11.

Расширенный фильтр

Пока применяемые вами фильтры брали текущие значения в вашей таблице как отправную точку. Если вы чувствуете себя уверенно при работе с фильтрами, то можете попробовать расширенный вариант: *фильтрацию по условию.* Применяя расширенный фильтр, можно точно определить нужную вам фильтрацию.

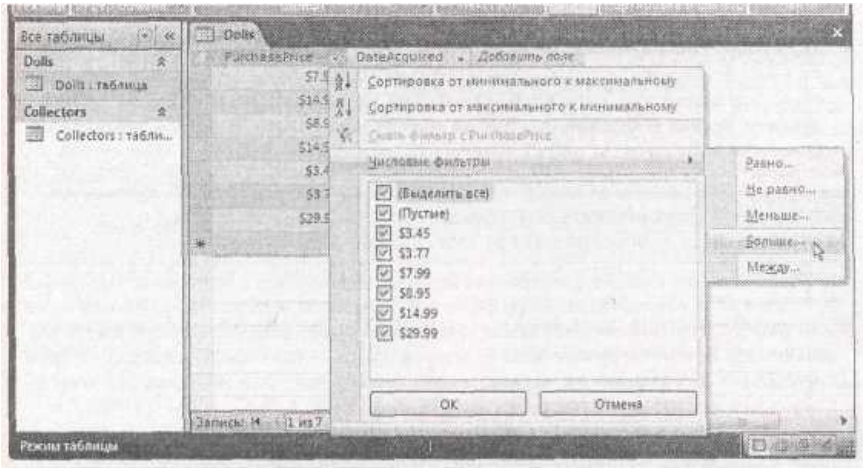
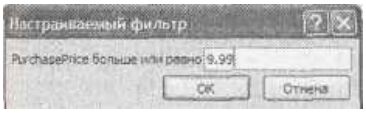


Рис. 3.12. *Вверху:* для числового поля, такого как **PurchasePrice** (покупная цена), фильтрация по условию позволяет найти значения, превышающие определенный минимум. *Внизу:* после выбора нужного вам типа фильтра следует задать информацию для фильтрации. Если вы выбрали условие **Больше,** нужно задать минимальное число. Записи со значениями, равными или большими минимума, отображаются на листе данных

Допустим, что вы хотите найти в погребе все редкие марочные вина с ценой, не превышающей 85 долларов. В случае фильтра по выделенному вам придется начать с поиска вина с ценой 85 долларов, который можно использовать для создания условия. Но что произойдет, если в вашем перечне нет вина с ценой точно 85 долларов или вам кажется, что его невозможно найти? Быстрее задать условие фильтрации вручную.



Вот как это делается. Сначала щелкните направленную вниз стрелку у правого края заголовка столбца. Но вместо выбора одного из вариантов обычного фильтра найдите подменю с вариантами фильтрации. Это меню называется в соответствии с выбранными данными, так для текстовых полей включен вариант **Текстовые фильтры** (Text Filters), для числовых полей - **Числовые фильтры** (Number Filters) и т. д. На рис. 3.12 показан пример.

Далее приведен краткий обзор, описывающий дополнительные варианты, которые можно использовать в расширенном фильтре в зависимости от выбранного типа данных.

■ *Текстовые типы данных.* Те же варианты, что и у фильтра по выделенному, вдобавок вы можете найти значения, начинающиеся с конкретного текста или заканчивающиеся определенным текстом.

■ *Числовые типы данных.* Те же варианты, что и у фильтра по выделенному, а также вы можете выбрать значения, лежащие в диапазоне, т. е. все значения, большие заданного минимума, но меньшие заданного максимума.

■ *Типы данных **Дата/время.*** Те же варианты, что и у фильтра по выделенному, вдобавок вы можете найти даты, попадающие в диапазон, и выбрать из огромного перечня встроенных вариантов, таких как **Вчера** (Yesterday) **На прошлой неделе** (Last Week), **В следующем месяце**

(Next Month), **В этом** году (Year to Date), **В первом квартале** (First Quarter) и т. д.

Практические занятия для опытных пользователей.
Фильтры в противоположность запросам

Если вы часто применяете фильтры, то наверняка столкнулись с проблемой. Программа Access сохраняет только один набор фильтров - фильтры, которые применяются в данный момент. Другими словами, после применения отличающегося фильтра ваш исходный фильтр исчезает и нужно снова формировать его с нуля в следующий раз, когда он понадобится. В большинстве случаев задать фильтр повторно нетрудно. Но если вы приложили значительные усилия, формируя наилучший набор условий фильтрации, и знаете, что захотите использовать их позже, их уничтожение огорчает.

Если вы оказались в подобной ситуации, значит, вы чрезмерно увлеклись фильтрами. Вместо того чтобы полагаться на фильтры для отображения интересующей вас информации, лучше создать отдельный запрос многократного использования. Как и фильтры, запросы позволяют увидеть подмножество ваших данных, сформированное на основании определенных условий. В отличие от фильтров запросы могут содержать более сложную логику, они могут не включать ненужные вам столбцы, и программа Access сохраняет их как отдельные объекты БД, поэтому вы всегда можете повторно использовать их позже. Вы начнете применять запросы в *главе 6.*

3.3. *Поиск*

Access предоставляет также средство *быстрого поиска* (quick search), позволяющее проверить лист данных на наличие заданной информации. В то время как фильтрация помогает извлечь важные записи, поиск больше всего подходит для обнаружения единственной детали, спрятанной в грудах данных. Фильтрация изменяет внешний вид листа данных, скрывая некоторые записи, а поиск оставляет все как есть. Он привлекает ваше внимание к данным, которые вы хотели видеть.

Самый быстрый вариант поиска - поиск с помощью поля, расположенного рядом с элементами управления для переходов между записями (рис. 3.13). Просто введите в него текст, который хотите найти. Пока вы вводите, в таблице автоматически высвечивается первое совпадение. Можно нажать клавишу <Enter> для поиска следующих совпадений.

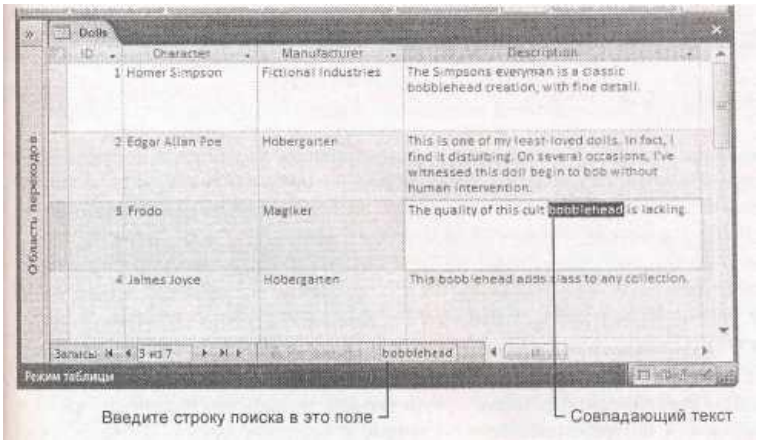


Рис. 3.13. Приведен пример поиска слова "bobblehead" (кукла-болванчик). Если найдено совпадение, можно продолжить поиск - просто нажать клавишу <Enter> для перехода к следующему совпадению. В данном примере нажатие клавиши <Enter> отправляет программу Access к полю **Description** следующей записи

Выполняя поиск, программа Access просматривает таблицу, начиная с первого ноля первой записи. Затем она перемещается слева направо, исследуя каждое поле текущей записи. Если достигнут конец и не найдено ни одно совпадение, поиск продолжается в следующей записи, проверяются все ее поля и т. д. Когда достигнут конец таблицы, поиск прекращается.

Если вы хотите изменить алгоритм поиска в программе Access, следует использовать команду **Найти** (Find).

1. Выберите на ленте **Главная → Сортировка и фильтр →Найти** (Home → Sort & Filter → Find) или просто используйте сочетание клавиш <Ctrl>+<F>.

На экран выводится диалоговое окно **Поиск и замена** (Find and Replace) (рис. 3.14).

2. Наберите искомый текст в поле **Образец** (Find What) и затем задайте остальные параметры поиска, которые хотите использовать.

- **Образец** (Find What) - текст, который вы ищите.
- **Поиск в** (Look In) - позволяет выбрать между поиском во всей таблице или в одном поле.

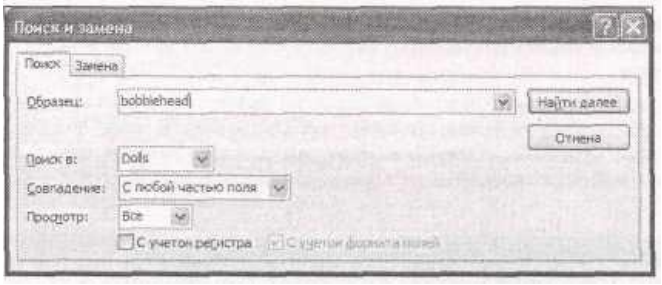


Рис. 3.14. Диалоговое окно **Поиск и замена** - отличное средство охоты за потерянной информацией

- **Совпадение** (Match) - позволяет определить, должны ли значения полностью совпадать с образцом. Используйте значение **Поле целиком** (Whole Field) для задания точного совпадения. Используйте значение **С начала поля** (Start of Field), если вы хотите найти совпадения с начальной частью поля (например, "bowl" (шар) и "bowling" (боулинг)), или значение **С любой частью поля** (Any Part of Field), если хотите найти искомый текст в любом месте поля (в этом случае "bowl" совпадает с "League of extraordinary bowlers" (лига выдающихся игроков в боулинг)).
- **Просмотр** (Search) - задает направление просмотра записей программой Access: **Вверх (Up), Вниз** (Down), **Все** (All).
- **С учетом регистра** (Match Case) - если флажок установлен, находятся только те совпадения, в которых совпадают заглавные и строчные буквы. Поэтому строка "банан" не совпадает со строкой "БАНАН".
- **С учетом формата полей** (Search Fields as Formatted) - установка этого флажка означает, что программа Access ищет значение, совпадающее с форматом значения (образца) на листе данных. Например, число 44 может отображаться в поле с денежным типом данных как $44.00. Если вы ищете 44, то всегда найдете его. Но если вы ищете форматированное представление $44.00, то найдете совпадение только если установлен флажок **С учетом формата полей.** В очень больших таблицах (с тысячами записей) поиск может идти быстрее, если сбросить данный флажок.

Примечание

Если вы сбрасываете флажок **С учетом формата полей,** следует выбрать поиск в одном поле в параметре **Поиск в** (Look In). Если же поиск ведется во всей таблице, нужно искать форматированные значения.

3. Щелкните мышью кнопку **Найти** далее (Find Next).

Программа Access начнет поиск с текущей позиции. Если вы применяете стандартное направление поиска **(Вниз),** Access перемещается слева направо в текущей записи, а затем переходит от записи к записи от начала таблицы к концу, пока не найдет совпадение.

Когда программа Access находит совпадение, она выделяет его цветом. Вы можете щелкнуть мышью кнопку **Найти** далее (Find Next) для поиска следующего совпадения или кнопку **Отмена** (Cancel) для прекращения поиска.

Малоизвестная или недооцененная возможность.
Поиск и замена

Это средство поиска, которое действует как мощное средство (но иногда опасное) корректировки записей.

При первоначальном выводе на экран диалоговое окно **Поиск и замена** (Find and Replace) отображает вкладку **Поиск** (Find). Но вы можете щелкнуть кнопкой мыши вкладку **Замена**

(Replace), чтобы иметь возможность найти конкретные значения и заменить их другим текстом. У операции замены те же параметры, что и у операции поиска за исключением дополнительного поля **Заменить на** (Replace With), предназначенного для ввода замещающего текста.

Самый безопасный способ замены - щелкнуть мышью кнопку **Найти** далее (Find Next) и перейти к следующему найденному совпадению, В этот момент вы можете посмотреть найденное совпадение и убедиться в том, что вы действительно хотите откорректировать его, а затем щелкнуть мышью кнопку **Заменить** (Replace) для корректировки найденного совпадения и перехода к следующему. Повторите эту операцию и внимательно проверьте всю таблицу.

Если вы похожи на неистового экстремала, обожающего затяжные прыжки с парашютом, и предпочитаете ходить по лезвию бритвы, можете использовать кнопку **Заменить** все (Replace All) для одномоментной замены всех найденных в таблице совпадений. Хотя эта процедура и до смешного быстра, но слегка рискованна. Операцию замены нельзя отменить (и команда **Отмена** (Undo) здесь не поможет, т. к. она отменяет только замену в одной записи), поэтому, если вы перестарались с заменой, легко отказаться от нее вам не удастся. Если вас все еще привлекает легкость операции **Заменить все,** создайте резервную копию вашей БД *(см. разд. "Создание резервных копий" главы 1),* прежде чем идти дальше.

3.4. *Усовершенствованное редактирование*

В *главе 1* вы познакомились с основными приемами редактирования, включая вставку, удаление и корректировку записей. Но у программы Access есть несколько более замечательных инструментов, которых вы еще не видели. В следующих разделах вы, как следует, освоите два замечательных и удобных средства программы Access - блок проверки орфографии и блок автозамены - и научитесь вставлять специальные символы в ваши поля.

3.4.1. *Проверка орфографии*

Проверка орфографии в программе Access почти точно такая же, как в других программах пакета Office, например, в Word, - в ней применяется тот же словарь, вылавливаются те же типы ошибок, и предоставляется возможность игнорировать нераспознанные элементы или добавлять их в словарь.

Разница заключается в том, что когда вы проверяете орфографию в Access, программа просматривает только текстовые поля и поля MEMO. Числа, даты и все остальное пропускаются. Конечно, в ваших таблицах наверняка есть текст, который вы не хотите проверять, например, имена, географические названия или названия товаров. У вас есть две возможности: можно проверить орфографию в одном поле, игнорируя все остальные, или начать проверку орфографии всего листа данных, но при этом пропускать определенные поля по ходу проверки.

Вот как это делается.

1. Перейдите в поле, с которого вы хотите начать проверку орфографии.

Если вы хотите провести проверку всего листа данных, от начала до конца, перейдите в первое поле первой записи.

Если вы хотите проверить фрагмент листа данных, перейдите в то место, с которого хотите начать проверку орфографии. Помните о том, что когда программа Access дойдет до конца листа данных, она снова вернется в начало листа **и** продолжит проверку до тех пор, пока не просмотрит каждое поле во всех записях. (Конечно, есть возможность в любой момент отменить проверку орфографии.)

Если вы хотите проверить только одно поле, выберите его перед началом проверки, щелкнув кнопкой мыши заголовок столбца.

2. Выберите на ленте **Главная → Записи → Орфография** (Home → Records → Spelling) или просто нажмите клавишу <F7>.

Кнопка **Орфография** (Check Spelling) выглядит как маленький значок (галочка) с буквами ABC над ним.

Если вы проверяете в пределах листа данных, программа Access просматривает текущую запись и переходит от поля к полю слева направо. Когда запись проверена, программа переходит к следующей записи и повторяет процесс. Если вы выбрали один столбец, Access просматривает значения только в этом поле сверху вниз.

После завершения проверки орфографии диалоговое окно сообщает о том, что все ваши данные

проверены. Если ваша таблица прошла успешно проверку орфографии, это диалоговое окно - единственное получаемое вами известие о выполненной операции. С другой стороны, если программа Access обнаруживает возможные орфографические ошибки в процессе проверки, она выводит на экран окно **Орфография** (Spelling) (рис. 3.15), отображающее слово, ставшее камнем преткновения, и список вариантов замены.

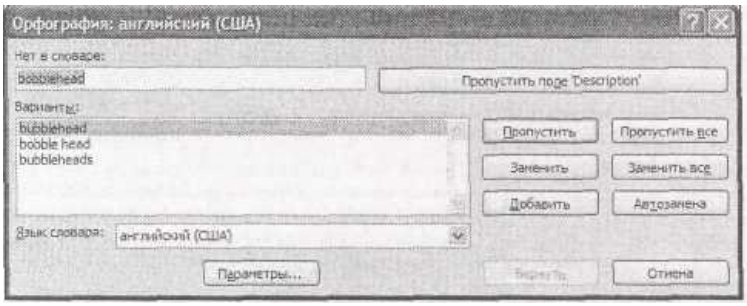


Рис. 3.15. Когда программа Access обнаруживает слово, как ей кажется написанное с ошибкой, то выделяет его цветом. В отличие от программы Word, Access не разрешает редактировать файл БД, пока активно окно **Орфография.** Вы можете либо щелкнуть кнопкой мыши один из вариантов замены в окне **Орфография -** например, щелкнуть мышью кнопку **Заменить** для замены слова с ошибкой на первый предлагаемый для замены вариант - либо отменить проверку орфографии

Окно **Орфография** содержит много вариантов корректировки. Если блок проверки орфографии жалуется на слово, в котором действительно допущена ошибка, у вас есть три возможных варианта.

■ *Исправить его немедленно.* Щелкните кнопкой мыши одно из слов в списке возможных вариантов и затем щелкните мышью кнопку **Заменить** (Change) для замены слова с ошибкой правильно написанным словом. Вы также можете дважды щелкнуть кнопкой мыши выбранный вариант, эффект будет такой же.

■ *Исправить его везде.* Щелкните кнопкой мыши одно из слов в списке возможных вариантов и затем щелкните мышью кнопку **Заменить все** (Change All) для замены слова с ошибкой правильно написанным словом. Если программа Access во время проверки орфографии найдет ту же ошибку где-нибудь еще на вашем листе данных, она автоматически повторит замену, не предупреждая вас о возникшей проблеме.

■ *Исправлять его всегда.* Щелкните кнопкой мыши одно из слов в списке возможных вариантов и затем щелкните мышью кнопку **Автозамена** (AutoCorrect). Access вносит изменение в это поле и во все другие вхождения слова с такой же ошибкой. Кроме того, программа добавит сведения о замене в список автозамены (описанный в *разд. "Автозамена" далее в этой главе).* Если вы введете это нераспознанное слово в другую запись (или даже в другую таблицу), Access автоматически откорректирует ваш ввод. Этот вариант очень полезен, если обнаружилась ошибка, которую вы часто допускаете.

С другой стороны, если блок проверки орфографии жалуется на слово, которое вы не хотите заменять, у вас есть дополнительные возможности для выбора следующих вариантов.

■Кнопка **Пропустить** (Ignore) пропускает данное слово, и проверка орфографии продолжается. Если программа Access найдет то же слово где-то еще в вашей электронной таблице, она снова предложит вам исправить его.

■Кнопка **Пропустить все** (Ignore All) пропускает данное слово, и проверка орфографии продолжается. Если Access найдет это таинственное слово где-то еще в вашей таблице, оно будет пропущено. Кнопку **Пропустить все** можно использовать для того, чтобы заставить программу игнорировать слова, которые вы не хотите исправлять, например, имя и фамилию человека.

■Кнопка **Пропустить поле** (Ignore Field) игнорирует любые ошибки в заданном поле до конца процесса проверки орфографии. Этот путь удобен для исключения из проверки полей, содержащих множество имен, географических названий или названий для того, чтобы не терять время на повторяющиеся просмотры искусственных вариантов замены, предлагаемых блоком проверки орфографии.

■Кнопка **Добавить** (Add) добавляет слово в вспомогательный орфографический словарь. Этот шаг важно совершить, если вы планируете продолжать использовать слово на этом листе данных и на многих других. (Название компании - это важный вклад во вспомогательный

орфографический словарь.) Программа Access не только пропустит любые вхождения этого слова, но и при обнаружении в поле похожего, но слегка отличающегося слова включит слово из вспомогательного словаря в свой список вариантов для замены, позволяя вам быстро устранить мелкие опечатки.

■ Кнопка **Отмена** (Cancel) полностью прекращает процесс проверки орфографии. После этого вы можете откорректировать поле и продолжить проверку позже.

Примечание

Все приложения пакета Office, установленные на вашем компьютере, совместно используют один и тот же вспомогательный орфографический словарь. Если вы добавите слово в программе Access, а затем выполните проверку орфографии в программе Word, это слово успешно пройдет проверку. Это удобство, экономящее время до тех пор, пока вам не изменяет чувство меры и вы добавляете слова, которые действительно имеют отношение к делу.

Параметры проверки орфографии

Вы можете управлять процессом проверки орфографии с помощью нескольких понятных параметров. Для задания этих параметров (или их просмотра) выберите кнопку **Office,** а затем кнопку **Параметры Access** (Access Options) для вывода на экран одноименного диалогового окна. Далее выберите строку **Правописание** (Proofing) в списке слева (рис. 3.16). Эту же страницу параметров можно найти, если щелкнуть мышью кнопку **Параметры** (Options) в окне **Орфография** (Spelling) во время проверки орфографии.

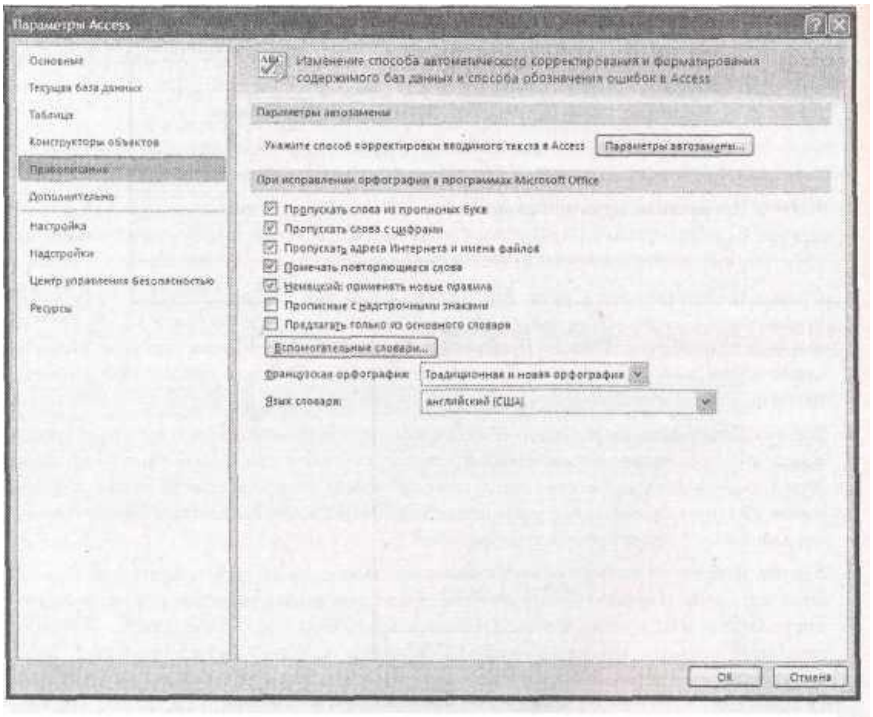


Рис. 3.16. Параметры проверки орфографии позволяют задать язык и несколько других установок. Все они зависят от выбранного для проверки языка; в последнем раскрывающемся списке окна показан язык, используемый в настоящий момент

Далее приведены самые распространенные параметры проверки орфографии. **Пропускать слова из прописных букв** (Ignore words in UPPERCASE). Если этот флажок установлен, программа Access не проверяет слова, состоящие из прописных букв (что полезно, если ваш текст содержит много акронимов).

Пропускать слова с цифрами (Ignore words that contain numbers). Если установить этот флажок, Access не проверяет слова, содержащие цифровые символы, например, Sales43 или Н31Ю. Если вы не установили этот флажок, программа проверяет эти данные и помечает их как содержащие ошибки до тех пор, пока вы не добавите их по вспомогательный словарь.

Пропускать адреса Интернета и имена файлов (Ignore Internet and file addresses). Если установлен этот флажок, программа Access игнорирует слова, кажущиеся именами файлов (например, c:\Documents and Settings) или адресами Web-сайтов (например, http://FreeSweatSocks.com).

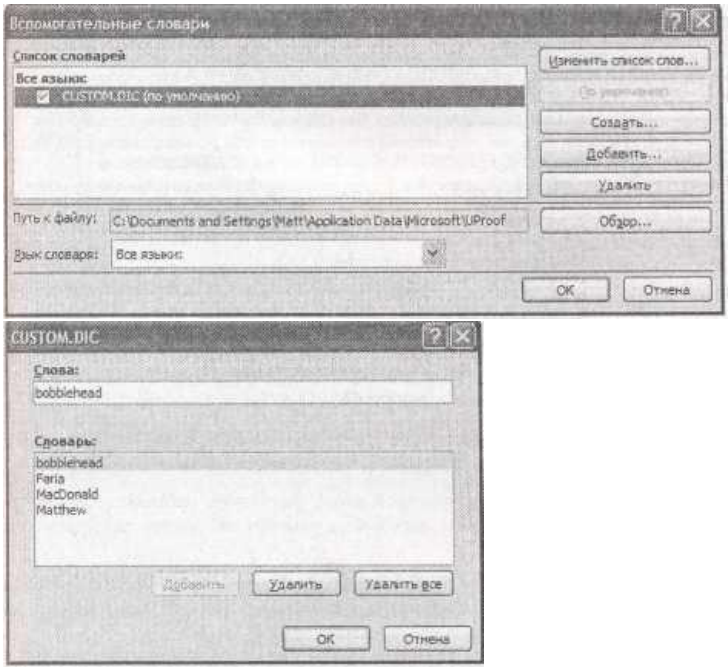


Рис. 3.17. *Вверху:* с помощью диалогового окна **Вспомогательные словари** вы можете удалить текущий вспомогательный орфографический словарь, добавить новые или отредактировать список слов вручную. *Внизу:* если щелкнуть мышью кнопку **Изменить список слов,** на экран выводятся все слова из файла custom.dic. Вы можете вставить новые слова или удалить те, которые больше не используются

■ **Помечать повторяющиеся слова** (Flag repeated words). Этот параметр находит ошибки, если вы случайно повторили одно и то же слово дважды, например "это это".

■ **Прописные с надстрочными знаками** (Enforce accented uppercase in French). Заставляет применять во французских словах диакритические знаки, которые они должны иметь, даже в случае прописных букв (у которых они выглядят странно). Англоговорящим (как и русскоговорящим) людям не стоит беспокоиться о задании этого параметра.

■ **Предлагать только из основного словаря** (Suggest from main dictionary only). Когда установлен этот флажок, блок проверки орфографии не использует слова из вспомогательного словаря как варианты для замены, если находит незнакомое слово. Но он принимает слова, совпадающие со словами, включенными во вспомогательный словарь.

Вы также можете задать файл, который программа Access применяет для хранения слов вспомогательного словаря - незнакомых программе слов, которые вы добавляете в словарь во время проверки орфографии, Для этого щелкните мышью кнопку **Вспомогательные словари** (Custom Dictionaries), которая выводит на экран одноименное диалоговое окно (рис. 3.17).

Примечание

Вспомогательные словари хранятся на вашем жестком диске в разделе, отведенном для каждой учетной записи. Например, если вы зарегистрировались в системе под именем пользователя Dan_Quayle (Дэн_Кейл), то, вероятно, найдете вспомогательный словарь в папке C:\Documents and Settings\Dan_Quayle\Application Data\Microsoft\UProof. Один из побочных эффектов этой системы состоит в том, что вспомогательные словари не могут использовать два человека, зарегистрировавшиеся с разными именами на одном компьютере (пока вы не добавите вручную словарь другого пользователя в диалоговое окно **Вспомогательные словари).**

3.4.2. *Автозамена*

Когда вы вводите текст в поле, операция автозамены (AutoCorrect) подчищает за вами, исправляя строчные буквы на прописные и обычные орфографические ошибки. Автозамена -

настолько умное средство, что вы даже можете не знать о ее неусыпном контроле каждого вашего движения. Для того чтобы оценить это волшебство, проанализируйте поведение, похожее на следующее.

■ Если вы набираете *ПРивет,* автозамена изменяет набор на *Привет.*

■ Если вы набираете *friday* (пятница), автозамена изменяет слово на *Friday.*

■ Если предложение начинается со строчной буквы, автозамена заменяет ее прописной.

■ Если вы перепутали местами буквы в обычном слове (например, набрали *этто* вместо *этот* или слово *teh* вместо артикля *the),* автозамена заменит набранное слово корректным вариантом.

■ Если вы случайно нажали клавишу <Caps Lock> и затем набрали *дЖОН сМИТ,* хотя на самом деле хотели набрать *Джон Смит,* программа Access не только исправит ошибку, но и отключит режим Caps Lock.

По большей части автозамена безвредна, а иногда и полезна, т. к. избавляет вас от мелких опечаток в большом отчете. Но если вам нужно ввести слова с необычным сочетанием строчных и прописных букв или у вас возникло заурядное желание восстать против стандартов английского языка, некоторые или все действия автозамены можно отключить.

Для установки параметров автозамены выберите кнопку **Office,** а затем кнопку Параметры Access (Access Options) для вывода на экран одноименного диалогового окна. Далее выберите строку **Правописание** (Proofing) в списке слева. На странице с параметрами в правой части окна щелкните мышью кнопку **Параметры автозамены** (AutoCorrect Options).

Большинство установочных параметров содержит пояснительную информацию, и их можно отключить, сбросив флажки. На рис. 3.18 поясняется использование параметра **Заменять при вводе** (Replace text as you type), который предназначен не только для исправления ошибок.

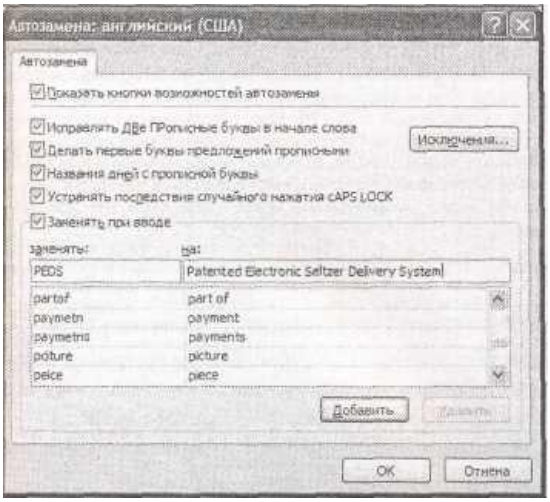


Рис. 3.18. Под флажком **Заменять при вводе** располагается длинный список символов и обычно набираемых с опечатками слов (левый столбец), которые программа Access автоматически заменяет другим текстом (представленным в правом столбце). Что если вы хотите, чтобы символ авторского права был представлен как буква в скобках? Можно удалять отдельные корректировки (выбрать нужную и щелкнуть мышью кнопку **Удалить)** или откорректировать замещающий текст. Вы также можете добавить собственные правила. Возможно, вы хотите заставить программу Access при наборе символов "PEDS" вставлять текст "Patented Electronic Seltzer Delivery System" (Запатентованная электронная система подачи газированной воды). Просто наберите текст в полях **заменять** и **на,** как показано на рисунке, и щелкните мышью кнопку **Добавить**

Совет

Параметры более тонкой настройки можно использовать, нажав кнопку **Исключения** (Exceptions) для задания случаев отказа программы Access от применения автозамены. При щелчке мышью этой кнопки на экран выводится диалоговое окно **Исключения при автозамене** (AutoCorrect Exceptions) со списком исключений. В этот список включены аббревиатуры, содержащие точку, но не требующие применения прописных букв (например, стр.), и слова, в которых разрешается смешивать строчные и прописные буквы (например, WordPerfect).

3.4.3. *Специальные символы*

Текст порой состоит не только из букв, цифр и знаков пунктуации. Существуют специальные символы, которые нельзя просто набрать на вашей клавиатуре. Примером может служить знак авторского права (©), который вы вставляете, набрав текст (С) и предоставив возможность автозамене выполнить свою работу. Другие символы, такие как греческая буква тэта (Ѳ), не так легко доступны. Для использования такого символа вам понадобится утилита Таблица символов (Character Map).

Таблица символов, часто игнорируемое средство, позволяет увидеть все символы, входящие в шрифт. Это большое подспорье для выискивания необычной буквы "е" с надстрочным знаком и других неанглийских символов.

Примечание

Другие приложения пакета Office, такие как Word и Excel, предлагают гораздо больше специальных символов для использования. Они поддерживают все разновидности шрифтов, включая стильный шрифт Wingdings, поставляемый с ОС Windows и содержащий пиктограммы. У программы Access более строгий стиль работы. Программа принимает только обычные стандартные символы, поддерживаемые в любом шрифте. БД хранят неформатированные данные и, следовательно, текстовые поля не содержат подробностей, касающихся шрифта и форматирования. Исключение составляет редко применяемый в полях Memo текст RTF *(см. разд. "Форматированный текст" главы 2).*

Далее объясняется, как применять утилиту Таблица символов для вставки специальных символов.

1. Щелкните мышью кнопку меню Пуск (Start) и выберите команду **Выполнить** (Run).

Утилита Таблица символов - это составная часть ОС Windows, а не программы Access. В результате запускать ее нужно вне Access.

2. В диалоговом окне **Выполнить** введите charmap и щелкните мышью кнопку ОК. На экране появится окно **Таблица символов** (рис. 3.19).
3. В списке **Шрифт** (Font) выберите шрифт Calibri.

Нет необходимости использовать экзотический шрифт, поскольку программа Access не поддерживает его. Но, даже применяя любой обычный шрифт, включая Arial, Times и Tahoma. можно найти поддерживаемые специальные символы. Calibri - это стандартный шрифт, который программа Access использует для отображения информации на листе данных, если вы не изменяли настройку (как описано в *примечании "Малоизвестная или недооцененная возможность. Настройка всех листов данных" в разд. "Форматирование листа данных "ранее в этой главе).*

4. Прокручивайте набор символов до тех пор, пока не найдете тот, который нужен вам. Если вам требуется буква из другого языка, ищите настойчивее - вы почти наверняка найдете ее. Если вам хочется чего-то более экзотического и вы не можете найти это, вероятно, удача от вас отвернулась. И вам придется использовать обычный текст.
5. Дважды щелкните кнопкой мыши найденный символ.

Он появится в поле **Для копирования** (Characters to copy) в нижней части окна **Таблица символов.** Вы можете повторять шаги 4 и 5 столько раз, сколько нужно для копирования разных символов в строку.

6. Щелкните мышью кнопку **Копировать** (Copy).

ОС Windows копирует символы из поля **Для копирования** в буфер обмена.

7. Перейдите снова в окно программы Access.

Если вы находитесь не в том поле, в которое хотите вставить копируемый текст, перейдите в нужное поле сейчас. Если вы хотите поместить символ между двумя уже имеющимися, убедитесь в том, что курсор стоит в нужном месте поля.

Нажмите сочетание клавиш <Ctrl>+<V> для вставки символа.

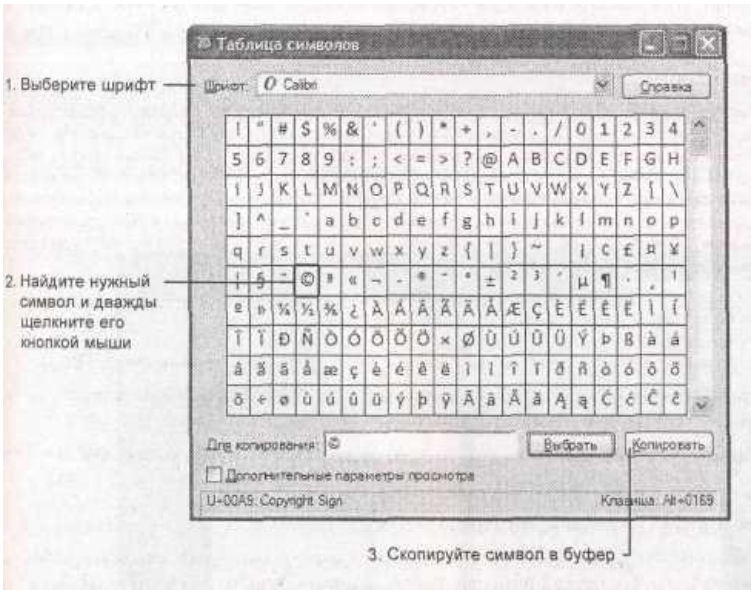


Рис. 3.19. В данном примере с помощью Таблицы символов копируется символ авторского права

3.5. *Печать листа данных*

Если вы хотите просмотреть ваши данные за обеденным столом (и вас не беспокоят возможные конфликты с теми, кто не относится к числу любителей программы Access), нет ничего лучше твердой копии ваших данных. Вы можете организовать быстрый вывод на принтер, выбрав **Office → Печать** (Office → Print) в тот момент, когда лист данных виден на экране. Но результаты, вероятно, разочаруют вас, особенно если у вас большой стол.

Основная проблема состоит в том, что программа Access не беспокоится о таблицах, слишком широких для размещения на одной печатной странице. Программа разбивает распечатку на отдельные страницы. Если у вас большой стол, и вы распечатали таблицу, используя стандартные установочные параметры, то вполне можете получить в результате распечатку из четырех страниц в ширину и стольких же страниц в длину. Для того чтобы добиться лучших результатов при выводе на печать, крайне важно выполнить предварительный просмотр таблицы перед печатью, как описано в следующем разделе.

3.5.1. *Предварительный просмотр страницы*

Средство предварительного просмотра в программе Access дает возможность отрегулировать поля, ориентацию страницы и т. д. прежде, чем вы отправите вашу таблицу на принтер. Благодаря этому средству, вы можете гарантировать большую пригодность вашей финальной распечатки. Для предварительного просмотра таблицы откройте ее (или выберите в области переходов) и затем выберите последовательность: кнопка **Office → Печать → Предварительный просмотр** (Office → Print → Print Preview).

Предварительный просмотр воспроизводит внешний вид ваших данных при выводе их на бумагу, В отличие от листа данных в предварительном просмотре печатной версии таблица разбивается на страницы (рис. 3.20). Вы видите, что именно помещается на каждой странице и сколько страниц требуется для распечатки вашей таблицы (и какое содержимое отображается на каждой странице).

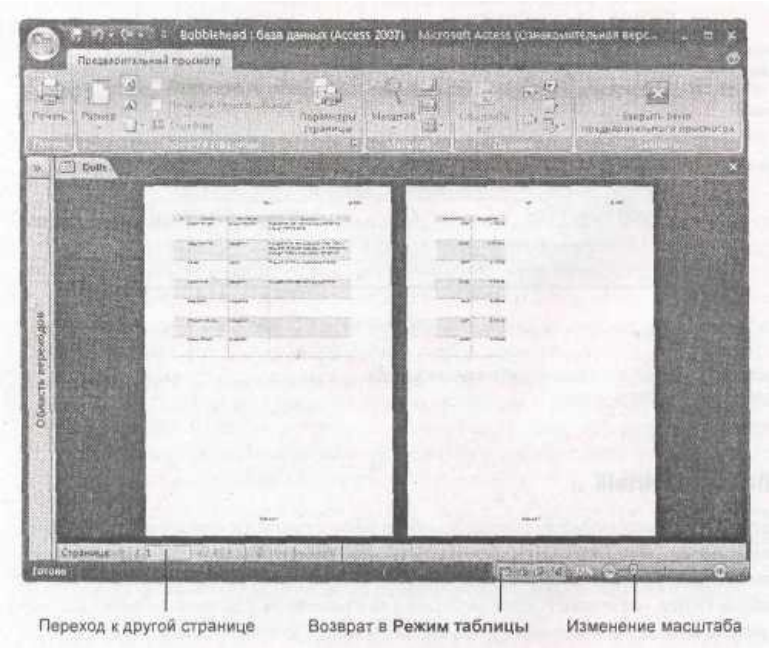


Рис. 3.20. Эта таблица слишком широка для вывода на одну бумажную страницу, поэтому некоторые столбцы перенесены на вторую страницу

Если вы довольны тем, что видите, можете выводить вашу распечатку, выбрав на ленте **Предварительный просмотр → Печать → Печать** (Print Preview → Print → Print). Эта последовательность открывает диалоговое окно, подобное окну **Печать** ОС Windows, в котором можно выбрать принтер и завершить дело.

Завершив просмотр в окне предварительного просмотра, выберите на ленте **Предварительный просмотр → Закрыть → Закрыть окно предварительного просмотра** (Print Preview → Close Preview → Close Print Preview) или щелкните мышью одну из кнопок режима в правом нижнем углу окна программы Access для перехода в **Режим таблицы** или **Конструктор.**

Перемещение в окне предварительного просмотра

Вы не можете ничего менять в окне предварительного просмотра. Но у вас есть возможность просмотреть страницы вашей виртуальной распечатки и увидеть, устраивает ли она вас.

Далее описаны способы перемещения в окне предварительного просмотра.

■Используйте кнопки прокрутки для перехода от страницы к странице. **Эти** кнопки выглядят так же, как элементы управления переходами между записями на листе данных, но они служат для перемещения между страницами, а не записями.

■Для перехода от страницы к странице можно использовать полосу прокрутки или клавиши <Page Up> и <Page Down>.

■Для того чтобы поближе рассмотреть страницу, щелкните кнопкой мыши в любом месте страницы предварительного просмотра (вы заметите, что указатель мыши превратился и лупу). Этот щелчок мышью увеличивает масштаб отображения листа до 100%, поэтому яснее виден текст и другие детали. Для того чтобы вернуться снова к отображению полной страницы, еще раз щелкните кнопкой мыши страницу.

■Для большего точного масштабирования используйте скользящий ползунок (zoom slider), расположенный в строке состояния в правом нижнем углу. Сместите его влево для уменьшения масштаба (и увидите больший фрагмент на экране) или вправо для увеличения масштаба (и сосредоточьтесь на меньшей порции вашей страницы),

■Для одновременного отображения двух страниц выберите на лепте **Предварительный просмотр → Масштаб → Две страницы** (Print Preview → Zoom → Two Pages). Для вывода большего числа страниц выберите **Предварительный просмотр → Масштаб → Другие страницы** (Print Preview → Zoom → More Pages) и укажите в списке число страниц, которое вы хотите видеть одновременно.

Изменение макета страницы

Программа Access предоставляет небольшой набор параметров страницы, которые можно регулировать в окне предварительного просмотра с помощью группы **Разметка страницы** (Page Layout) на вкладке ленты **Предварительный просмотр.** Далее перечислены эти параметры.

■**Размер** (Size). Позволяет использовать страницы разного размера. Если вам надоели таблицы, которые не помещаются на страницу, может быть, стоит инвестировать в бумаги большего размера (например, стандартные листы размером 216x355 мм).

■**Книжная** и **Альбомная** (Portrait, Landscape). Позволяют выбрать ориентацию страницы. Программа Access, как и другие программы пакета Office, полагает, что вы хотите распечатать текст, применяя стандартную книжную ориентацию. При книжной ориентации страницы располагаются вертикально, так что длинная сторона становится боковой, а короткая - верхней. Она прекрасно подходит для резюме и заметок, но совершенно не годится для широкой таблицы, поскольку, по крайней мере, несколько столбцов будут бездумно обрезаны и перенесены на другие страницы. Альбомная ориентация в этой ситуации гораздо больше подходит, поскольку она поворачивает страницу на 90°, сокращая количество строк на странице, но размещая больше столбцов на ней.

■ **Поля** (Margins). Позволяет выбрать величину свободного пространства между таблицей и краями страницы. Поле представляет собой раскрывающуюся кнопку, и когда вы щелкаете ее мышью, то видите меню с несколькими стандартными вариантами размера полей **(Обычное** (Normal), Узкое (Narrow) и **Широкое** (Wide)). Если ни один из них не удовлетворяет вашим требованиям, щелкните мышью кнопку **Параметры страницы** (Page Setup), выводящую на экран одноименное диалоговое окно, в котором можно задать точные значения полей со всех сторон страницы.

3.5.2. *Тонкая настройка распечатки*

Исходя из ограниченного числа параметров настройки макета страницы, вы можете решить, что мало, что можно сделать для настройки распечатки. На самом деле у вас больше возможностей, чем вы думаете. Многие параметры форматирования, о которых вы узнали в данной главе, влияют и на вашу распечатку. Применяя правильное форматирование, можно создать более удачную печатную копию.

Далее приведены советы, касающиеся печати и объясняющие, как разные параметры форматирования влияют на ваши распечатки.

■ *Шрифт.* При выводе таблиц на печать используются гарнитура и размер шрифта, применяемые в вашей таблице. Уменьшите размер, и вы сможете разместить больше данных на ограниченном пространстве.

■ *Порядок следования столбцов и скрытие некоторых из них.* Измените порядок следования столбцов перед выводом на печать, добиваясь нужного отображения страницы. Еще лучше использовать скрытие столбцов *(см. разд. "Скрытие столбцов "ранее в этой главе)* для того, чтобы спрятать поля, которые не важны.

■ *Ширина столбцов и высота строк.* Программа Access использует точные значения ширины и высоты, которые вы задали в таблице. Сожмите некоторые столбцы, чтобы поместилось больше данных, и увеличьте высоту строк, если у вас есть поля с большими текстовыми фрагментами и вы хотите выводить их в нескольких строках.

■ *Закрепленные столбцы.* Если таблица слишком широка и не помещается на одной странице, закрепленный столбец будет печататься на всех страницах. Например, если закрепить поле **FirstName** (имя), вы увидите его на всех напечатанных страницах, поэтому вам не придется совмещать страницы для того, чтобы определить, кто есть кто.

■ *Параметры сортировки.* Они помогают легко ориентироваться в информации, размещенной на листе данных, и могут сделать то же самое в ваших распечатках. Задайте их перед выводом на печать.

■ *Параметры фильтрации.* Это невоспетые герои вывода на печать в программе Access. Примените их для отбора только важных строк. В этом случае на вашей распечатке будет именно то, что нужно.

Единственная проблема, с которой вы столкнетесь при использовании этих параметров, заключается в том, что их нельзя задать в окне предварительного просмотра распечатки. Но это

можно сделать на листе данных, перейти в окно предварительного просмотра, чтобы увидеть результат, снова вернуться на лист данных и еще подкорректировать их, а затем еще раз перейти в окно предварительного просмотра и т. д. Правда, этот процесс очень быстро может утомить.

Совет

Не тратьте много времени на корректировку параметров форматирования для создания совершенной распечатки. Если у вас большая таблица, которую не удается разместить удачно на странице, возможно, вам следует использовать отчеты, которые описаны в *части III*. Отчеты предоставляют гораздо больше возможностей форматирования, включая разбиение полей на несколько строк, вставку рамок вокруг записей и возможность отведения дополнительного пространства для больших значений за счет мягкого перемещения остальной информации на другие страницы.

4. Глава 4. Блокировка неправильных данных

Даже лучшие проектировщики БД проводят бессонные ночи, тревожась об ошибках, способных проникнуть в их БД. Неверные данные - печально известная проблема, эти данные проникают в БД, прячутся там с течение месяцев и проявляют себя, только когда вы отправили по электронной почте счет клиенту "Blank Blank" ("Пробел пробел") или продали мешок арахиса за -4.99 долларов.

Лучший способ борьбы с подобными проблемами - прежде всего, помешать некорректным данным проникать в вашу БД. Другими словами, нужно задать правила проверки, которые отвергают подозрительные данные в момент их ввода.

После того как неверные данные попали в БД, найти их труднее, чем иголку в стоге сена.

В этой главе описан основной набор средств проверки, имеющийся у программы Access:

1) ***основные,*** включая совпадения, обязательные поля и значения по умолчанию;
2) ***маски ввода,*** форматирующие, с помощью образцов обычный текст, такой как почтовые коды и телефонные номера;
3) ***правила верификации*** (validation rules), устанавливающие строгие правила для полей, не подчиняющихся никаким законам;
4) ***подстановки*** (lookups), ограничивающие возможные значения списком заранее заданных вариантов.

4.1. *О целостности данных*

Все средства проверки программы Access реализованы в режиме Конструктора, с которым вы познакомились в *главе 2.* Для их применения вы выбираете поле и настраиваете его свойства. Единственная сложность - знать, какие свойства наиболее полезны. Некоторые из них описаны в *главе 2,* но в следующих разделах вы найдете дополнительные подробности.

Совет

Программа Access предоставляет три варианта переключения в режим **Конструктора.** Можно щелкнуть правой кнопкой мыши на заголовке вкладки таблицы и выбрать из меню команду **Конструктор,** использовать кнопку **Режим** на вкладке ленты **Главная** или воспользоваться крошечными кнопками режима в правом нижнем углу окна программы Access. Если вы очень нетерпеливы, то можете даже не открывать вашу таблицу, а просто найти ее в области переходов, щелкнуть ее правой кнопкой мыши и выбрать команду **Конструктор.**

4.1.1. *Запрет незаполненных полей*

Для того чтобы каждая запись имела какой-то смысл, в ней должен быть хотя бы абсолютный минимум информации. Без вашей помощи программа Access не может отделить важную информацию от необязательных деталей. По этой причине все поля в новой таблице определены как необязательные, за исключением поля первичного ключа (которое обычно содержит идентификационный номер записи). Убедитесь в этом с помощью таблицы **Dolls** из *главы 1;* вы быстро обнаружите, что можете вставлять записи, фактически не содержащие данных.

Этот недостаток легко исправить. Просто выберите в Конструкторе поле, которое обязательно должно быть заполнено и задайте в свойстве Обязательное **поле** (Required) значение *Да* (рис. 4.1).

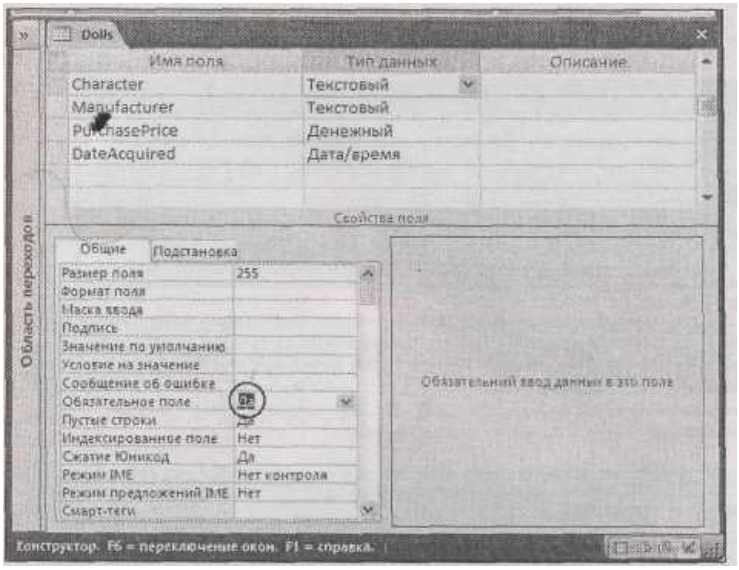


Рис. 4.1. Свойство **Обязательное поле** сообщает программе Access о запрете пропущенных значений (именуемых на профессиональном жаргоне null)

Access проверяет свойство **Обязательное поле** при каждом добавлении новой записи или исправлении поля в уже имеющейся записи. Но если в вашей таблице уже есть данные, нет гарантии, что они соответствуют установленным правилам.

Представьте, что вы внесли в таблицу **Dolls** сведения о нескольких куклах-болванчиках до того, как решили, что в поле **Character** обязательно должно быть значение. Вы переключаетесь в **Конструктор,** выбираете поле **Character** и заменяете значение свойства Обязательное **поле** на *Да.* Когда вы сохраняете таблицу (возвращаясь в **Режим таблицы** или закрывая таблицу), программа Access дает вам возможность проверить записи о куклах-болванчиках, уже внесенных в таблицу (рис. 4.2). Если вы выбираете выполнение проверки и Access обнаруживает проблему, программа позволяет вам отменить внесенные корректировки (рис. 4.3).

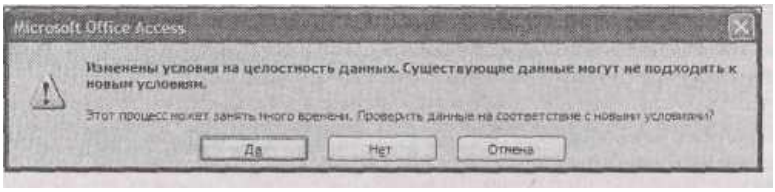


Рис. 4.2. Это хорошая идея проверить таблицу на соответствие новым требованиям, которые вы установили. В противном случае некорректные данные могут остаться в БД. Не дайте этому сообщению запугать вас - пока у вас нет десятков тысяч записей, такая проверка не займет много времени

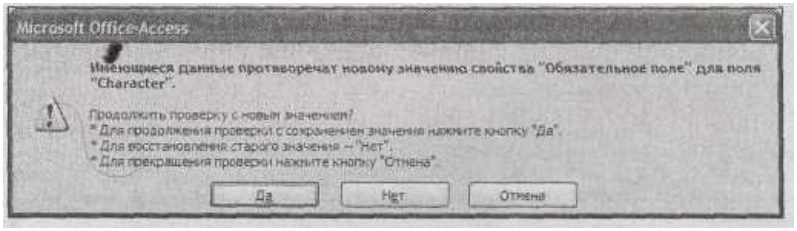


Рис. 4.3. Если программа Access находит пропущенное значение, она останавливает поиск и спрашивает вас, что делать. Вы можете сохранить изменения (даже если они конфликтуют хотя бы с одной записью) - в итоге, по крайней мере, новые записи не будут порождать подобную проблему. Другой возможный вариант - вернуть прежнее, более терпимое значение свойству поля. В любом случае вы можете найти пропущенные данные, отсортировав данные с помощью вопроса, выводящего незаданные значения в верхние строки таблицы

Для тех, кто понимает.

Не требуйте слишком многого

Вы должны хорошенько подумать, какой минимум данных вам нужен для создания записи.

Например, компания, продающая костюмы Элвиса, возможно, не захочет вставлять новый комплект одежды в свою таблицу **Products** (изделия), пока он не будет полностью готов. Свойство **Обязательное поле** в данном случае хорошее подспорье, поскольку не даст включить в каталог незавершенные изделия.

С другой стороны, такая строгость не годится в таблице **Customers** (клиенты) той же компании. Отделу продаж нужна гибкость при добавлении новых предполагаемых клиентов, даже если предоставлена только частичная информация. Потенциальный клиент может позвонить и оставить только почтовый адрес (без номера счета, номера телефона, адреса электронной почты и т. п.). Даже в этом случае, не имея полной информации о клиенте, вы должны включить его в таблицу **Customers,** для того чтобы он или она получали ежемесячный информационный бюллетень.

Примите за правило применение необязательного поля в том случае, когда данные для него необязательны или недоступны в момент ввода записи.

Пропущенные значения и пустые строки

Программа Access поддерживает свойство **Обязательное поле** (Required) для всех типов данных. Но, возможно, для некоторых типов данных понадобятся дополнительные проверки. Это объясняется тем, что свойство **Обязательное поле** запрещает только незаполненные поля - поля, в которых нет совсем никаких данных. Но программа Access, что кажется несколько странным, различает пропущенные значения и пустые строки (empty text).

Пропущенное значение (null) означает отсутствие данных. *Пустая строка* свидетельствует о том, что значение поля было введено, но оказалось пустым. Все еще недоумеваете? Разница существует, т. к. БД, такие как Access, должны распознавать пропущенные данные. Пропущенное значение может означать оплошность - возможно, кто-то просто забыл ввести значение, С другой стороны, пустая строка означает сознательное решение исключить данную информацию.

Примечание

Для того чтобы проверить эту разницу в своей таблице, создайте текстовое поле со значением свойства **Обязательное поле,** равным *Да.* Попытайтесь вставить новую запись и оставить ее пустой. (Access хладнокровно остановит вас.) Теперь попробуйте вставить новую запись, но поместите единственный пробел в поле. Происходит странная вещь: Access автоматически обрезает пробелы и, делая это, превращает ваш единственный пробел в пустую строку. Но вы не получите сообщения об ошибке, поскольку пустая строка - это не то же самое, что пропущенное значение.

К счастью, если вы сочтете это различие сбивающим с толку, можно запретить и пропущенные значения, и пустые строки. Просто установите в свойстве **Обязательное поле** (Required) значение *Да* для запрета пропущенных значений и в свойстве **Пустые строки** (Allow Zero Length) значение *Нет* для запрета пустых строк.

Примечание

Такое же различие существует и у данных числового типа. Даже если установить свойство **Обязательное поле** равным *Да,* вы все равно можете вставить значение 0. Если вы хотите помешать этому, нужно применить правила верификации, описанные в *разд. "Правила верификации или условия на значения" далее в этой главе.*

4.1.2. *Задание значений по умолчанию*

До сих пор поля в ваших таблицах заполнялись явно человеком, вставлявшим запись или пропускавшим ее. Но есть еще одна возможность - вы можете определить значение по умолчанию. Теперь, если кто-то вставляет запись и пропускает поле, программа Access использует в нем значение по умолчанию.

Задается значение по умолчанию в свойстве поля **Значение по умолчанию** (Default Value).

Для поля **AddedCost** (добавленная стоимость) числового типа вы могли бы оставить его равным 0. В текстовом поле **Country** (страна) можно использовать строку "U.S.A." как значение по умолчанию. (Все текстовые значения, используемые как значения по умолчанию, должны быть заключены в кавычки.)

Программа Access выводит вес значения по умолчанию в строке, подготовленной для ввода новой записи, в нижней части таблицы (рис. 4.4). Она также автоматически вставляет значения по умолчанию в любые скрытые столбцы *(см. разд. "Скрытие столбцов " главы 3).*

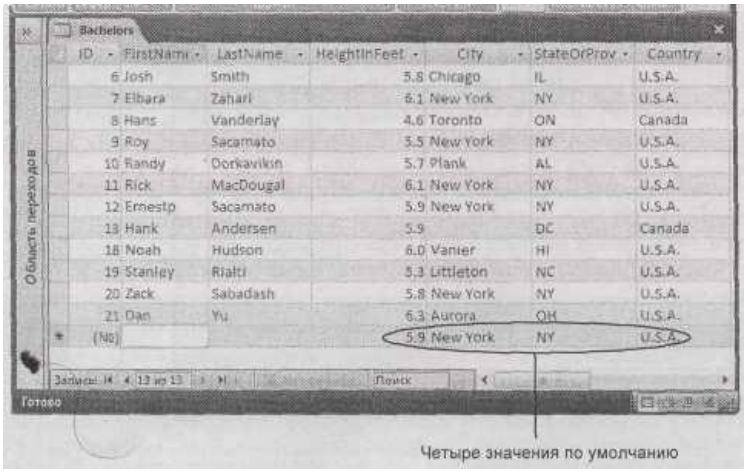


Рис. 4.4. Эта служба знакомств использует четыре значения по умолчанию: стандартный рост (5.9), город по умолчанию (New York), штат по умолчанию (тоже New York - NY) и страну по умолчанию (U.S.A.). Такая система хороша, поскольку в большинстве новых записей содержится именно эта информация. С другой стороны, нет оснований предлагать значения по умолчанию для полей, содержащих имя и фамилию

Access вставляет значение по умолчанию, когда вы создаете новую запись (но вы всегда можете изменить это значение). Вы также можете во время редактирования поля вернуться к значению по умолчанию с помощью сочетания клавиш <Ctrl>+<Alt>+<Пробел>.

Совет

Очень удобно использовать значение по умолчанию как отправную точку для новой записи. Например, когда создается новая запись в таблице, можно редактировать значение по умолчанию, а не заменять его полностью другим значением.

Вы можете создать и более развитые динамические значения по умолчанию. Программа Access оценивает их, когда вы вводите новую запись, что означает зависимость выбранного значения по умолчанию от других данных записи. Динамические значения по умолчанию используют выражения (специальные формулы БД), способные выполнять вычисления или извлекать другие подробности. Одна полезная функция Date ( ) извлекает текущую дату, установленную на вашем компьютере. Если применить эту функцию как значение по умолчанию для поля с датой (как показано на рис. 4.5), программа Access автоматически вставляет текущую дату при вводе новой записи.

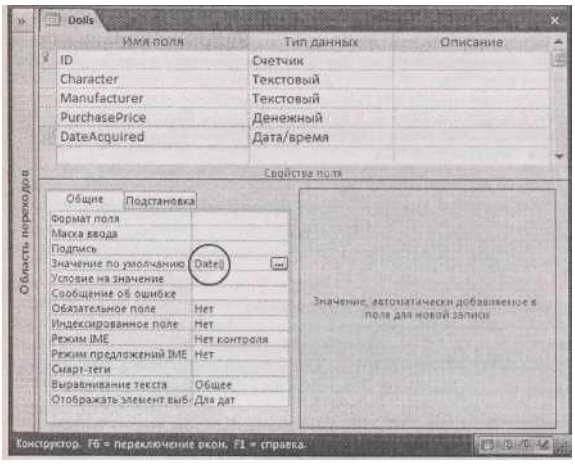


Рис. 4.5. Если вы применяете функцию Date ( ), как значение по умолчанию в поле **DateAcquired** в таблице с куклами-болванчиками, то при каждой вставке новой записи о кукле программа Access вставляет текущую дату. Вы решаете - оставить ее или заменить другим значением

Примечание
Вы узнаете больше о выражениях языка SQL (Structured Query Language, язык структурированных запросов) в *части II.*

4.1.3. *Предотвращение дублирования значений с помощью индексов*

Первое правило любой таблицы - каждая включенная в нее запись должна быть уникальна. Для соблюдения этого требования вам нужно выбрать первичный ключ *(см. разд. "Первичный ключ"главы 2),* одно или несколько полей, которые не должны дублироваться в разных записях.

Но здесь есть подводный камень. Как вы узнали из *главы 2,* самый надежный способ - создание идентификационного поля (поля Код) для первичного ключа. До сих пор во все таблицы, которые вы видели, включалось такое поле. Но что, если вам нужно, чтобы другие поля тоже были уникальны? Представьте себе, что вы создаете таблицу **Employees** (сотрудники). Вы следуете правильным принципам проектирования БД и идентифицируете каждую запись автоматически генерируемым идентификационным номером. Но вы также хотите быть уверены в том, что в таблице нет двух сотрудников с одинаковыми номерами социального обеспечения (Social Security number, SSN), и, тем самым, желаете предупредить возможные ошибки - такие, как случайный повторный ввод данных об одном и том же сотруднике.

Примечание
Для того чтобы вспомнить, почему так важны идентификационные поля, еще раз прочитайте *примечание "На профессиональном уровне. Как Access предотвращает дублирование записей" в разд. "Первичный ключ" главы 2.* В таблице **Employees** вы конечно можете выбрать SSN в качестве первичного ключа, но ситуация будет далека от идеальной, когда вы начнете связывать таблицы друг с другом *(см. главу 5),* и возникнут проблемы, если позже понадобится изменить номер социального обеспечения (например, из-за ошибки) или ввести информацию о сотруднике до того, как вы получите SSN.

Вы можете заставить поле требовать уникальных значений с помощью *индекса.* Индекс БД похож на предметный указатель в книге - это список значений (из поля) с перекрестной ссылкой, которая указывает на соответствующий раздел (полную запись). Если индексировать поле **SocialSecurityNumber,** программа Access создаст список, подобный приведенному в табл. 4.1 и хранящийся в файле вашей БД.

Таблица 4.1. Список, хранящийся в БД после создания индекса

SocialSecurityNumber	Location of Full Record
001-01-3455	…
001-02-0434	…
001-02-9558	…
002-40-3200	…

С помощью этого списка программа Access может быстро определить, не дублируется ли в новой записи уже имеющийся SSN. Если это опасение подтверждается, Access не разрешит вставить такое значение.

На профессиональном уровне.
Как работают индексы
Важно то, что список номеров социального обеспечения отсортирован. Сортировка означает, что номер 001-01-3455 всегда предшествует в индексе номеру 002-40-3200, независимо от физического размещения записи в БД. Такая сортировка важна, т. к. она позволяет программе

Access быстро проверять наличие дубликатов. Если вы вводите номер 001-02-4300, Access достаточно прочитать только первую часть списка. Как только программа обнаружит следующий "больший" SSN (тот, который включен в список позже в результате сортировки, например 001-02-501), она уже знает, что в оставшейся части индекса нет дубликата. На практике все БД используют множество алгоритмов оптимизации для того, чтобы сделать этот процесс стремительным. Но существует один ключевой принцип - без применения индекса программа Access должна проверять всю таблицу. В БД хранятся несортированные таблицы, поэтому программа не может быть уверена в том, что данного SSN нет в таблице до тех пор, пока не проверит каждую запись.

Итак, как применить индекс с полем? Хитрость заключается в применении свойства Индексированное **поле** (Indexed), которое доступно для данных всех типов за исключением типа **Вложение** и типа **Объект OLE.** Когда вы добавляете поле, у его свойства **Индексированное поле** указано значение *Нет.* Для вставки индекса и предупреждения дублирования значений вы можете изменить в **Конструкторе** значение свойства **Индексированное поле** на *Да (Совпадения не допускаются).* При выборе третьего варианта - *Да (Допускаются совпадения)* - создается индекс, но разрешается нескольким записям иметь одинаковые значения поля. Этот вариант не поможет вам поймать повторяющиеся записи, но его можно применить для ускорения поиска (для получения дополнительной информации см. *примечание "Практические занятия для опытных пользователей. Как индексы ускоряют поиск" в разд. "Получение заданного количества первых записей" главы 6).*

Примечание

Как вы знаете из *главы 2,* первичные ключи также предотвращают дублирование записей с помощью аналогичного метода. Когда вы определяете первичный ключ, программа Access создает индекс в поле.

Когда вы закрываете **Конструктор** после изменения свойства **Индексированное поле,** программа Access напоминает о необходимости сохранить ваши корректировки. В этот момент она создает любые нужные ей новые индексы. Вы не можете создать индекс, запрещающий совпадения, если в вашей таблице уже есть дублирующаяся информация. В данной ситуации Access выводит сообщение об ошибке, когда закрывается **Конструктор** и программа пытается добавить индекс.

Часто задаваемый вопрос.

Индексы и производительность

Индексы - это средство предотвращения ввода неверных данных или средство, повышающее производительность?

Индексы не только препятствуют дублированию значений. Они также незаменимы, когда нужно увеличить скорость обычного поиска. Программа Access может использовать индекс для поиска нужной ей записи во многом так же, как вы применяете предметный указатель в конце книги для поиска конкретной темы или термина.

Если вы выполняете поиск для удаления из таблицы **Employees** сотрудника с заданным номером социального обеспечения (SSN), Access может применить индекс. С его помощью программа найдет совпадающее значение гораздо быстрее и просто перейдет по указателю к полной записи.

Дополнительную информацию о том, как индексы могут ускорить поиск, вы найдете в *примечании "Практические занятия для опытных пользователей. Как индексы ускоряют поиск" в разд. "Получение заданного количества первых записей" главы. 6.* Но важно знать, что индексы улучшают производительность только в случае очень больших и сложных таблиц. Если вы храните несколько сотен записей, в каждой из которых горстка полей, вам на самом деле не нужен индекс - Access и так выполнит поиск с ошеломляющей скоростью.

Индексы для нескольких полей

Вы также можете применять индексы для предотвращения повторений комбинации значений. Представьте себе, что вы создаете таблицу **People** (люди) для хранения списка ваших друзей и их контактной информации. Вам могут встретиться одинаковые имена и фамилии.

Но, возможно, вы хотите помешать включению в две записи одинаковых и имени, и фамилии. Такой запрет избавит вас от случайного ввода сведений об одном и том же человеке дважды.

Примечание

Данный пример может привести к нескончаемым проблемам, если у вас действительно есть двое друзей с одинаковыми именем и фамилией. В этом случае вам придется удалить индекс до ввода этих имени и фамилии. Прежде чем создавать любые индексы, следует серьезно подумать о законных основаниях возможного дублирования значений.

Для обеспечения уникальности комбинации полей необходимо создать составной индекс, в котором объединяется информация из нескольких полей. Далее описаны необходимые для этого действия. 1. В **Конструкторе** выберите на ленте **Работа с таблицами | Конструктор → Показать или скрыть → Индексы** (Table Tools | Design → Show/Hide → Indexes).

На экране появится окно **Индексы** (Indexes) (рис. 4.6). С его помощью можно просмотреть уже созданные индексы и создать новые.

Рис. 4.6. В окне **Индексы** показаны все индексы, которые определены в таблице. В нем приведен простой индекс для поля ID (создаваемого программой Access автоматически) и составной индекс, который в данный момент создается

2. Выберите имя для вашего индекса. Введите его в первую пустую строку в столбце **Индекс** (Index Name)

Имя индекса не важно - программа Access использует его для хранения индекса в БД, но вы не увидите его во время работы с таблицей. Обычно для этого используются имена одного или нескольких полей, которые индексируются (например, LastName+FirstName).

3. Выберите первое поле в столбце **Имя поля** (Field Name) в той же строке (например, LastName).

Какое поле вы укажете первым, не имеет значения. В любом случае индекс сможет помешать дублированию значений. Но порядок играет роль в случае использования индекса в поиске для повышения производительности. Вы узнаете об этом больше в *примечании "Практические занятия для опытных пользователей. Как индексы ускоряют поиск" в разд. "Получение заданного количества первых записей" главы 6.*

4. В нижней части окна установите значение свойства **Уникальный индекс** (Unique) равным *Да.* В этом случае создается индекс, запрещающий совпадения значений (в отличие от индекса, применяемого только для увеличения скорости поиска).

Вы также можете задать значение Да для свойства **Пропуск пустых полей** (Ignore Nulls), если хотите, чтобы программа Access разрешила дублирование пропущенных (незаданных) значений. Например, вы хотите сделать поле номера социального обеспечения (SSN) необязательным. Но если уж SSN вводится вы хотите быть уверенным в том, что он не совпадает ни с каким другим значением, в данном случае вам следует задать значение *Да* в свойстве **Пропуск пустых полей.** Если значение этого свойства равно *Нет,* программа Access разрешит только одной записи иметь пропущенное (неопределенное) значение в поле SSN, что, возможно, вас не устроит.

Примечание

Можно запретить пропущенные значения вместе с применением свойства **Обязательное поле,** как описывается в *разд. "Запрет незаполненных полей" ранее в этой главе.*

Пропустите свойство **Ключевое поле** (Primary), которое задает индекс первичного ключа.

5. Перейдите на одну строку ниже. Оставьте поле **Индекс** пустым (это говорит программе Access о том, что это часть предыдущего индекса), но выберите другое поле в столбце **Имя поля** (например, FirstName).

Если вы хотите создать составной индекс из нескольких полей (больше двух), просто повторяйте этот шаг до тех пор, пока не вставите все нужные поля. На рис. 4.7 показано, как выглядит сформированный индекс. Теперь можно закрыть окно **Индексы.**

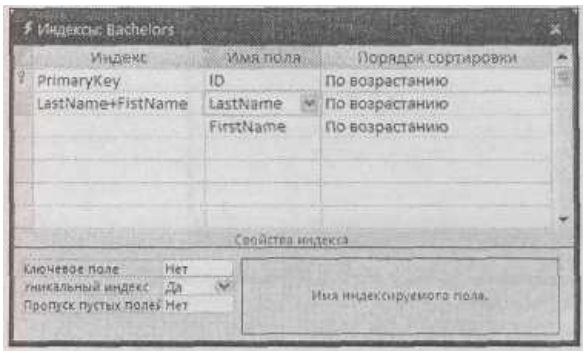


Рис. 4.7. Здесь представлен составной индекс, препятствующий включению в таблицу двух людей с одинаковыми именем и фамилией

4.2. *Маски ввода*

Как вы уже знаете, БД ценят непротиворечивость данных. Если у вас есть поле **Height** (рост), лучше использовать в нем значения, заданные в одних и тех же единицах измерения, в противном случае ваши данные и ломаного гроша стоить не будут. Аналогично, если у вас есть поле PhoneNumber (номер телефона), лучше убедиться в том, что у всех номеров один и тот же формат. Если одни телефонные номера записаны с дефисами, пробелами и скобками (например, (844) 547-1123), в то время как другие несколько отличаются (скажем, 847-547-1123), а третьи вообще пропускают междугородний код (547-1123), у вас появится небольшая проблема. Из-за недостатка согласованности вам будет трудно обрабатывать такие данные (например, искать конкретный телефонный номер или отсортировать телефонные номера в зависимости от междугороднего кода).

Рис, 4.8. *Вверху:* показано поле PhoneNumber с маской, готовое принимать данные. К этому моменту пользователь, вводящий данные, еще не ввел ни одного символа. В поле PhoneNumber автоматически выводится текст, содержащий символы-заполнители. *Внизу:* маска форматирует

номера по мере их ввода. Если вы введете 1234567890 в данную маску телефонного номера, то увидите следующий текст: (123) 456-7890.

В файле БД данные хранятся как 1234567890, а на листе данных представлены с привлекательным внешним оформлением. Это оформление и есть маска

Для облегчения обработки значений, имеющих фиксированный шаблон, - например, телефонных номеров - вы можете воспользоваться *маской ввода.* Маска ввода (или маска для краткости) предоставляет возможность сообщить программе Access, какой шаблон или образец должны использовать ваши данные. Основываясь на этом образце, Access изменяет способ ввода и редактирования значений, делая их более понятными и менее подверженными ошибкам. На рис. 4.8 показано, как маска позволяет программе Access форматировать последовательность символов в процессе их ввода в поле.

Вы можете добавить маску для любого поля с текстовым типом данных. По сравнению с обычным текстом маски обладают рядом достоинств,

■ *Маски управляют элементом ввода.* Будучи пустым, шаблон маски отображает символы-заполнители, на место которых должны попасть значения. В пустой маске телефонного номера отображается текст (_ _ _)_ _ _ - _ _ _ , ясно обозначающий вид необходимых данных.

■*Маски помогают понять смысл данных.* Гораздо легче читать множество значений, представленных определенным образом. Большинство людей быстрее найдут нужные номера социального обеспечения, если они будут представлены как (012-86-7180), не как (012867180).

■*Маски предупреждают ошибки.* Они отбрасывают символы, не соответствующие шаблону. Если вы пользуетесь маской для ввода номеров телефонов, то не сможете ввести буквы.

■*Маски устраняют путаницу.* Одни и те же данные многих типов можно представить несколькими способами. Вы можете ввести номера телефонов с междугородним кодом и без него. Используя маску с символами-заполнителями для междугороднего кода, выдаете понять, что эта информация обязательна (а также показываете, где она должна располагаться). Очевидно, что вам не нужно набирать скобки или дефисы для разделения номеров, поскольку эти детали уже стоят в нужном месте. Такие же преимущества маски дают при вводе дат, которые можно ввести разнообразными способами (Год/Месяц/День, Месяц-День-Год и т. д.).

Маски подходят как нельзя лучше для сортировки числовой информации в текстовом поле. Этот сценарий реализуется с самыми разными данными, включая номера кредитных карт, почтовые индексы и номера телефонов. Данные этих типов не следует хранить в числовых полях, поскольку они не должны интерпретироваться как единый номер. Их следует воспринимать как последовательность цифр. (Если вы допустите ошибку и сохраните номер телефона в числовом поле, то обнаружите, что пользователи могут ввести совершенно бессмысленные номера, такие как 0 или -14, поскольку это примеры корректных чисел, несмотря на то, что их нельзя считать допустимыми номерами телефонов. Маска в текстовом поле вылавливает с легкостью подобные ошибки.)

Маска не помогает бороться с более сложными проблемами, такими как значения с переменной длиной или хитроумными шаблонами. Например, маска не поможет определить неправильный адрес электронной почты.

Примечание

Маски поддерживают только типы данных **Текстовый и Дата/время.**

4.2.1. *Применение готовых масок*

Легче всего начать использование масок с применения одной из разнообразных подготовленных масок, предлагаемых программой Access. Это замечательный способ, т. к. не требует изучения таинственного искусства создания масок.

Далее перечислены действия, необходимые для выбора встроенной маски.

1. В **Конструкторе** выберите текстовое поле, в котором вы хотите применить маску. В данном примере используйте поле **PhoneNumber.**
2. Найдите свойство поля **Маска ввода** (Input Mask), щелкните кнопкой мыши в поле этого

свойства.

В этот момент у правого края поля появится маленькая кнопка (...) со скругленными углами, показанная на рис. 4.9.

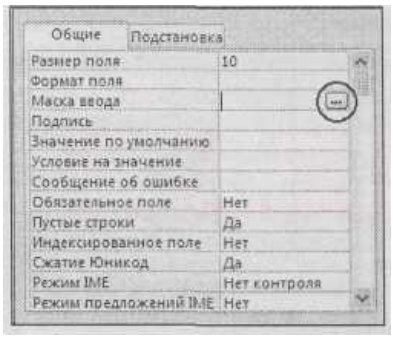


Рис. 4.9. С помощью кнопки (...) со скругленными углами программа Access сообщает о том, что вы не должны вводить значение вручную. Вместо этого можно щелкнуть мышью кнопку и вывести на экран программу-мастер (например, Мастер создания масок ввода) или другую разновидность полезного диалогового окна

3. Щелкните мышью эту кнопку.

Запустится Мастер создания масок ввода (рис. 4.10).

4. Выберите нужную вам маску из списка возможных вариантов.

В данном случае выберите первый элемент списка **(Phone Number).**

Примечание

Не нравится то, что получается? Тогда вам нужно создать собственную маску, пользуясь советами из *разд. "Создание собственной маски" далее в этой главе.* Если вы нашли близкую, но не идеальную готовую маску, выберите ее. Вы сможете откорректировать ее во втором окне данного мастера.

5. Щелкните мышью кнопку Далее (Next).

На экране появится второе окно мастера (рис. 4.11).

6. Если хотите, можно изменить маску или символ-заполнитель в ней.

Для изменения маски вам придется узнать, что означает каждый символ маски. Вы найдете их описание в табл. 4.2.

Символы-заполнители применяются для обозначения пустых позиций, в которые вводится информация. Стандартный заполнитель - символ подчеркивания (_). Иногда можно использовать пробел, дефис, звездочку или любой другой символ, введя его в иоле **Заполнитель** (Placeholder character).

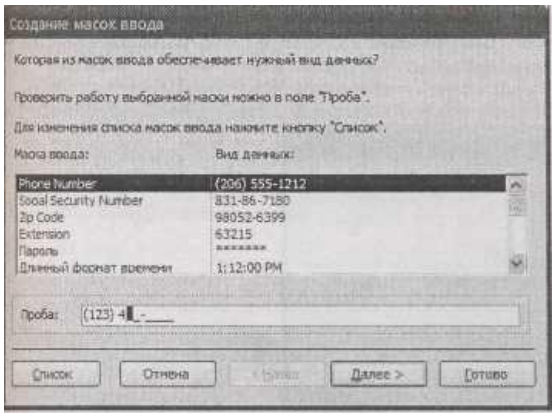


Рис. 4.10. Мастер создания масок ввода выводит на экран список общеупотребительных масок. Рядом с каждой маской приведен пример отформатированного маской значения. После выбора маски можно опробовать ее в текстовом поле **Проба,** в котором вы увидите, как будет вести себя ваше поле, если в нем применить маску

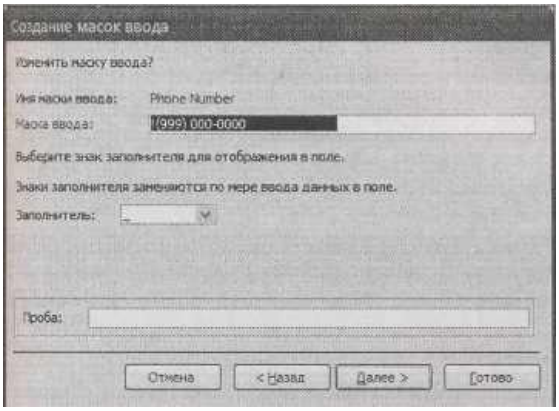


Рис. 4.11. Для телефонного номера применяется маска !(999) 000-000. Каждая цифра 9 обозначает необязательную цифру от 0 до 9. Каждая цифра 0 представляет обязательную цифру от 0 до 9. Итак, в соответствии с данной маской (123) 456-7890 - правильный номер телефона, как и 123-4567, а номер (123) 456 - неправильный

7. Щелкните мышью кнопку Далее.

Если вы вставляете маску в текстовое поле, на экран выводится завершающее окно мастера (рис. 4.12).

Если вы включаете маску в поле с датами, программе Access не нужно знать, как вы будете хранить информацию в поле - она уже знает это. В этом случае вы можете перейти к пункту 9 и щелкнуть мышью кнопку **Готово** (Finish).

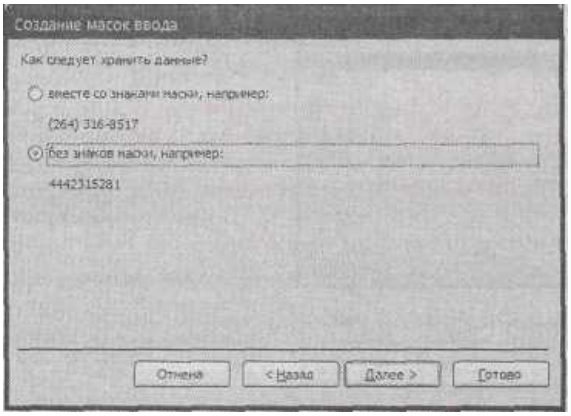


Рис. 4.12. В завершающем окне мастера можно выбрать способ хранения данных в вашем поле - с символами маски или без них

8. Укажите, как вы хотите хранить значение в данном поле.

Обычно хранятся только вводимые символы (другими словами, все, что вы вводите в поле). Если вы выбираете этот вариант, символы-заполнители в поле не включаются. Например, телефонный номер (416) 123-4567 хранится как 4161234567. Этот вариант экономит немного дискового пространства и позволяет вам изменить маску в дальнейшем для несколько иного представления данных.

Вы могли бы сохранить маску полностью со всеми дополнительными символами. В этом случае номер телефона хранился бы с дефисами, знаками подчеркивания и пробелами, например, (416) 123-4567. Это менее гибкий подход, поскольку не позволяет изменять маску впоследствии.

9. Щелкните мышью кнопку **Готово.**

Окончательная маска выводится в поле свойства **Маска ввода** (Input Mask). Прежде чем двигаться дальше, возможно, вы хотите убедиться в том, что зарезервированная длина поля соответствует маске. В примере с номером телефона **Размер поля** должен быть 10, если выбрано хранение неформатированного значения (поскольку в номере 10 цифр), или 14, если выбрано хранение полной маски вместе с заполнителями (один дефис, один пробел и две скобки).

10. Перейдите снова в Режим таблицы и щелкните мышью кнопку Да, когда программа Access предложит сохранить изменения. Теперь ваша маска ввода подготовлена.

Примечание

Программа Access использует информацию маски ввода для управления способом ввода данных в таблицу. Но маску можно перехитрить и ввести данные по-другому. Вы могли бы создать форму (как описано в *части IV)* и отключить маску. Маска не обеспечивает стопроцентной защиты от некорректных данных, если вам нужна полная гарантия, вместо маски нужно применять правило верификации (validation rule).

4.2.2. *Создание собственной маски*

Мастер создания масок ввода предоставляет очень ограниченный набор вариантов масок. Если вы хотите применять маску для ваших данных особого вида (например, специальный код клиента, применяемый на вашем предприятии), вы должны создать собственную маску.

Создать маску очень легко, но придется потратить немного времени, прежде чем вы добьетесь желаемого результата. У вас есть два основных варианта:

■наберите или отредактируете маску непосредственно в поле свойства Маска ввода;

■запустите Мастер создания масок ввода, выберите одну из масок как отправную точку (как описано в предыдущем разделе) и затем перейдите во второе окно мастера. Достоинство этого варианта в возможности тестирования вашей маски в поле Проба до того, как вы сохраните ее как часть своей таблицы.

В любой маске есть три типа символов:

■*заполнители* указывают вам, куда вводить символ;

■*специальные символы* сообщают программе Access о способе интерпретации части маски;

■*литералы* и любые другие символы служат элементами оформления, которое облегчает трактовку значения.

В предыдущем примере маска номера телефона - !(999) 000-000. Символы 9 и 0 - заполнители: они указывают, куда вводить цифры номера телефона. Скобки, пробел и дефис - просто средства форматирования - литералы. И всего один специальный символ - восклицательный знак. Он сообщает Access о том, что символы должны вводиться в маску слева направо, стандартный и единственный имеющий смысл в случае телефонного номера вариант.

Для того чтобы разложить все по полочкам, обратите внимание на следующие таблицы, В табл. 4.2 приведены все заполнители, которые можно использовать в масках ввода. В табл. 4.3 перечислены другие специальные символы. Все остальное автоматически относится к литералам.

Таблица 4.2. Символы-заполнители для масок ввода

Символ	*Описание*
0	Обязательная цифра (от 0 до 9)
9	Необязательная цифра (от 0 до 9)
#	Необязательная цифра, знак плюс (+) или знак минус (-)
L	Обязательная буква
?	Необязательная буква
A	Обязательные буква или цифра
a	Необязательные буква или цифра
&	Обязательный символ любого типа (включая буквы, цифры, знаки пунктуации и т. д.)
C	Необязательный символ любого типа (включая буквы, цифры знаки пунктуации и т. д.)

Таблица 4.3. Специальные символы для масок ввода

Символ	*Описание*
!	Обозначает направление заполнения маски слева направо при вводе. Это направление выбрано по умолчанию, поэтому данный символ не требуется (но во встроенные маски он включен)
<	Преобразует все следующие за ним символы в строчные
>	Преобразует все следующие за ним символы в прописные
\	Указывает на то, что следующий символ надо интерпретировать как литерал. Например, у символа # специальное назначение в масках. Поэтому если вы

	хотите обычный символ # включить в маску, следует ввести \#. Иногда этот символ применяется перед заполнителем, даже когда он не нужен. Вы можете встретить маску телефонного номера, содержащую последовательность \- вместо просто знака -. Оба эти варианта равнозначны
Пароль (Password)	Создает поле пароля. Любой символ, который вы вводите в поле, хранится как символ, но отображается как звездочка (*). С помощью варианта можно вставить в маску и что-то другое

Далее приведено несколько примеров масок, чтобы помочь вам взяться за дело.

■ (000) 000-000. В телефонный номер обязательно должны быть включены цифры междугороднего кода. Эта маска отличается от маски телефонного номера предлагаемой Мастером создания масок. В последней первые три 0 заменены 9, что делает междугородний код необязательным.

■ 00000-9999. Американский почтовый индекс, который состоит из пяти обязательных цифр, за которыми следуют дефис и (иногда) четыре дополнительные цифры.

■ L0L 0L0. Британский или канадский почтовый код, который формируется из шести символов с чередованием букв и цифр, например, M6S 3H2.

■ 99:00:00 >LL. Маска для ввода времени в поле типа **Дата/время**. Она формируется из двух цифр для часов и двух цифр для минут. Последние два символа (благодаря наличию символа >) всегда отображаются как прописные и предназначены для обозначения половины суток AM или PM. (Технически эта маска не препятствует вводу в эти позиции других символов. Но если вы введете время, такое как 12:30 GM, программа Access

пожалуется на то, что не может преобразовать ваше значение в тип данных **Дата/время**, как того требует поле.)

■099.099.099.099. IP-адрес, идентифицирующий компьютер в сети. Он записан как четыре значения, разделенные точками. В каждой части адреса должна быть, как минимум, одна цифра, а как максимум - три. Такой шаблон в маске отображается комбинацией 099 (одна обязательная цифра, за которой следуют две необязательные).

■Пароль (Password). Маска, допускающая ввод обычного текста разной длины с одной лишь разницей, все символы отображаются звездочками (*) и скрыты от любопытных глаз.

Маски могут заканчиваться двумя необязательными элементами, разделенными точкой с запятой (;).

Вторая составляющая маски - число, сообщающее программе Access, должна ли она сохранять литеральные символы маски в записи БД. (Это последний вопрос, который задает Мастер создания масок.) Если этот фрагмент маски пропустить или использовать цифру 1, Access сохраняет только символы, которые вводит пользователь. Если же вы примените цифру 0, программа сохранит весь текст вместе с литералами.

В третьей составляющей маски содержится символ-заполнитель. Если этот компонент маски пропустить, программа Access применяет знакомый знак подчеркивания.

Далее приведена маска, в которую включены оба дополнительных компонента:

(000) 000-000;1;#

Во второй части стоит 1, а в третей - #. Маска предназначена для ввода телефонных номеров и сохранения их в БД вместе с литералами маски (в данном случае двумя скобками, пробелом и дефисом), в ней вместо знака подчеркивания в качестве заполнителя используется знак номера (#).

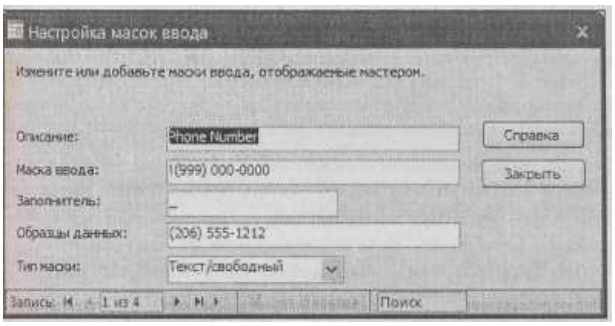


Рис. 4.13. Для добавления вашей собственной маски используйте кнопки переходов между записями (расположенные у нижнего края этого окна) для перехода в конец. Это

окно можно использовать и для изменения маски. Например, встроенная маска телефонного номера не требует обязательного включения междугороднего кода. Если это свобода, которой вы не хотите пользоваться, замените маску более строгой версией (000) 000-0000

Практические занятия для опытных пользователей.

Вставка вашей маски в список масок программы

Иногда удается создать маску, которая невероятно полезна, и хочется ее использовать в разных таблицах вашей БД (а может быть, и в разных БД). Несмотря па то, что можно скопировать маску в каждое поле, которое нуждается в ней, у программы Access есть более удачное средство - можно хранить вашу маску в списке масок программы. В этом случае маска будет появляться, как только вы запустите Мастер создания масок, рядом с другими стандартными масками ввода программы Access.

Для вставки маски в список перейдите к свойству поля **Маска ввода** (любого поля) и щелкните мышью кнопку со скругленными углами для запуска Мастера создания масок. Затем щелкните мышью кнопку **Список** (Edit List), раскрывающую удобное окно, в котором можно редактировать маски, предоставляемые Access, и вставить собственную (рис. 4.13).

4.2.3. *Правила верификации или условия на значения*

Маски ввода - замечательное средство, но они применяются лишь с информацией нескольких определенных типов, обычно с тестом фиксированной длины, имеющим единственный неизменный шаблон. Для создания действительно "пуленепробиваемой" таблицы следует применять более сложные ограничения, такие как гарантия попадания числа в определенный диапазон, проверка еще не наступивших дат или первой буквы текста. *Правила верификации* или *условия на значения* могут помочь вам сформировать все эти ограничения, используя всю мощь языка SQL (Structured Query Language, язык структурированных запросов).

Примечание

Более полное введение в язык SQL начнется с *главы 6.* К счастью, для написания правила верификации вам понадобится лишь частичка SQL. Ключевая составляющая правила - выражение проверки правильности данных (validation expression) вы увидите несколько практических примеров таких выражений, которые можно сразу поместить в ваши таблицы.

Суть правила верификации проста. Вы задаете ограничение, сообщающее программе Access, какие значения разрешены в поле, а какие нельзя считать правильными. Когда кто-нибудь добавляет новую запись или редактирует имеющуюся, Access проверяет, удовлетворяют ли данные вашим условиям на значения. Если нет, программа выводит сообщение об ошибке и заставляет вас откорректировать ошибочные данные и попробовать еще раз.

4.2.4. *Применение условия на значение поля*

У каждого поля может быть одно условие на значение или правило верификации. Далее приведены действия, необходимые для задания такого правила. Начнем с простого условия, запрещающего вводить в числовое поле 0 или любое отрицательное число (а в следующих разделах вы отшлифуете навыки создания правил верификации настолько, что сможете защитить и данные других типов).

Для вставки вашего условия на значение выполните следующие действия.

1. В **Конструкторе** выберите поле, к которому хотите применить условие.

Данные всех типов, кроме **Поле MEMO, Счетчик** и **Объект OLE,** поддерживают условие на значение. В данном примере правило верификации обрабатывает числовые данные **(Числового** или **Денежного** типа).

2. В свойстве поля **Условие на значение** (Validation Rule) введите проверочное выражение (рис. 4.14).

Выражение представляет собой фрагмент на языке SQL, выполняющий проверку введенных вами данных. Программа Access проверяет данные на значение, когда вы ввели порцию данных и собираетесь переходить к другому полю или другой записи. Например, условие >0 - это правило верификации, требующее ввода в иоле только положительных значений. В следующих разделах вы познакомитесь с другими условиями на значение.

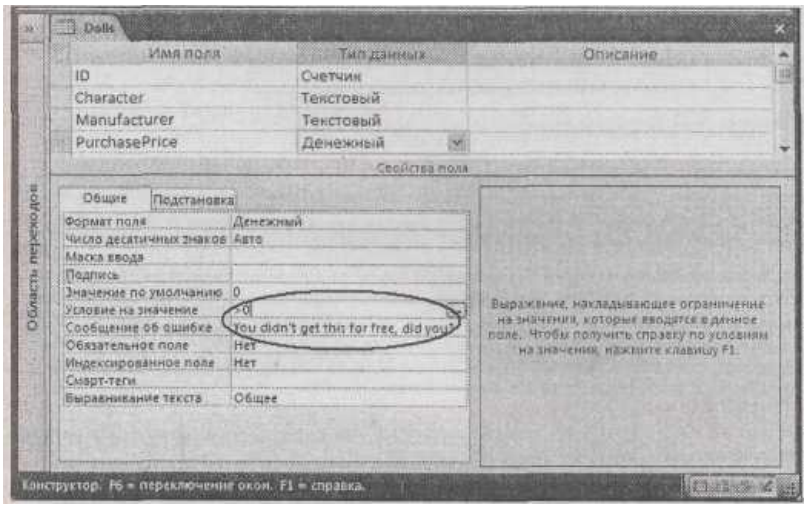


Рис. 4.14. В данном примере свойство **Условие на значение** препятствует вводу недопустимых цен, а свойство **Сообщение об ошибке** содержит текст сообщения

3. Введите текст сообщения в свойство поля **Сообщение об ошибке** (Validation Text).

Если вы введете значение, не прошедшее проверку, программа Access отвергнет его и выведет этот текст сообщения об ошибке в диалоговом окне. Если вы не предложите никакого текста, программа отобразит условие на значение для данного поля (которое вы ввели в пункте 2), что вызывает у простых смертных нечто большее, чем легкое недоумение.

4. Щелкните правой кнопкой мыши заголовок вкладки и выберите **Режим таблицы.**

Если в вашей таблице есть записи, программа Access дает вам возможность проверить их на соответствие заданному правилу верификации. Вам решать - выполнить такую проверку или полностью пропустить ее.

После перехода в **Режим таблицы** вы готовы к тестированию вашего условия на значение (рис. 4.15).

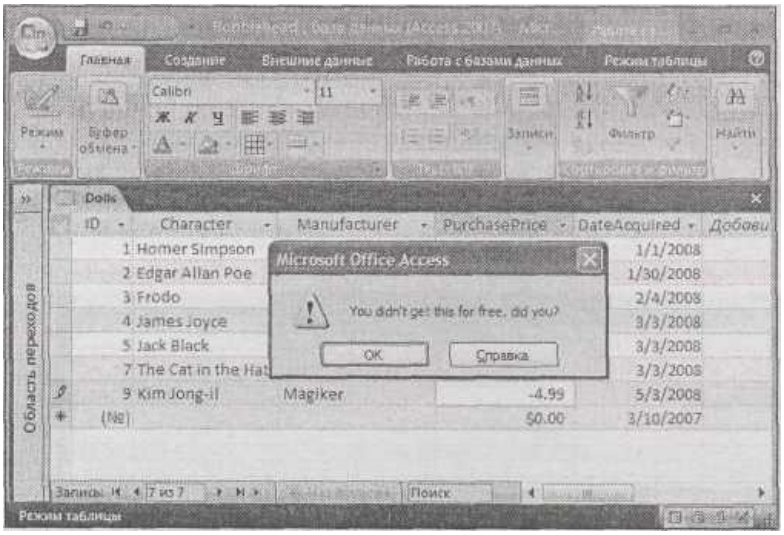


Рис. 4.15. В данном примере условие на значение >0 препятствует вводу отрицательных чисел в поле **Price.** Когда вы вводите отрицательное число, Access выводит окно сообщения с текстом об ошибке, который вы задали ("You didn’t get this for free, did you?" - "Вы ведь не получили ее даром, не так ли?"). После щелчка мышью по кнопке ОК вы возвращаетесь в поле, которое остается в режиме редактирования. У вас есть возможность изменить значение на положительное или нажать клавишу <Esc> для отказа от вставки или редактирования записи

Примечание

Наличие у вашей таблицы правил верификации или условий на значения вовсе не означает, что данные в этой таблице следуют заданным правилам. Отклонения могут возникнуть, если вы ввели данные прежде, чем правила верификации начали действовать. (Вы уже знаете об аналогичной потенциальной проблеме, связанной с обязательными полями, описанной в *разд. "Запрет незаполненных полей" ранее в этой главе*)

4.2.5. *Запись условия на значение поля*

Как видите, применять условие на значение к полю достаточно легко. Но формирование правила требует больших умственных усилий. Для получения желаемого результата вы должны сделать первый шаг в порой причудливый мир языка SQL.

Несмотря на то, что условия на значения ограничены только вашим воображением, разработчики, профессионально работающие о программе Access, возвращаются к нескольким основным шаблонам снова и снова. В следующих разделах представлены базовые знания для быстрой и легкой разработки правил верификации данных разных типов.

Примечание

Программа Access применяет правило верификации, только если в поле есть данные. Если поле остается пустым, программа принимает его без всякой проверки. Если вам не нравится такой подход, задайте в свойстве **Обязательное поле** значение *Да,* чтобы добиться обязательного заполнения поля, как описано в *разд. "Запрет незаполненных полей" ранее в этой главе.*

Проверка допустимости числовых значений

Для числовых данных самый распространенный вариант проверки - принадлежность введенного значения определенному диапазону. Другими словами, вы проверяете, больше или меньше введенное число другого значения. Ваши инструменты в этом случае - знаки операций сравнения < и >. В табл. 4.4 приведено несколько часто используемых примеров.

Таблица 4.4. Условия на значение для чисел

Сравнение	*Пример условия*	*Описание*
Меньше чем	<100	Значение должно быть меньше 100
Больше чем	>0	Значение должно быть больше 0
Не равно	< >42	Значение может быть любым, но не равным 42
Меньше или равно	<=100	Значение должно быть не больше 100
Больше или равно		Значение должно быть не меньше 0
Равно	=42	Значение должно быть 42. (Много ли смысла в необходимости ввода этого значения кем-то?)
Между	Between 0 and 100	Значение должно быть 0,100 или любое промежуточное значение

Проверка допустимости дат

Как и в случае числовых данных, проверка допустимости дат, как правило, включает проверку принадлежности даты определенному диапазону. Ваша задача - убедиться в том, что у вашей даты формат, подходящий для условия на значение. Если вы используете условие >Jan 30, 2007 (> 30 Янв, 2007), программа Access приходит в крайнее замешательство, т. к. не понимает, что текст (Jan 30, 2 007) предназначается для представления даты. Точно так же, если вы проверяете условие >1/30/07, Access предполагает, что числа справа от знака сравнения - часть выражения с последовательными операциями деления.

Для решения этой проблемы используйте универсальную синтаксическую форму представления дат программы Access, которая выглядит следующим образом: #1/30/2007#

В универсальную синтаксическую запись для представления дат компоненты включаются в порядке *месяц/день/год* и обрамляются с обеих сторон символами #. С помощью этого синтаксиса вы можете использовать условие, такое как >#1/30/2007#, требующее, чтобы

вводимая дата была больше (наступала позже), чем January 30, 2007 (30 января 2007 January 31, 2007 отвечает данному требованию, а любая дата в 2006 г. - нет.

Универсальная синтаксическая запись может включать и время, например: #1/30/2007 5:30PM#

Примечание

При сравнении двух дат программа Access принимает во внимание сведения о времени. Дата #1/30/2007# не содержит данных о времени, поэтому она интерпретируется как наступившая в самую первую секунду суток. В результате Access считает, что значение #1/30/2007 8:00 AM# больше, поскольку наступает на 8 часов позже.

Теперь, зная об универсальной синтаксической записи для дат, вы можете использовать любые операции сравнения, которые применяются для сравнения чисел. Можно применять и следующие удобные функции для получения информации о текущих дате и времени:

■ Date ( ) - вычисляет текущую дату (без какой-либо информации о времени, поэтому она вычисляется как первая секунда текущего дня);

■ Now ( ) - вычисляет текущий момент времени, включая дату и время.

Примечание

Функция - это встроенная процедура, выполняющая какую-либо задачу, например считывание текущей даты с компьютерных часов. В *разд. "Функции для обработки дат" главы 7* вы познакомитесь со многими функциями обработки дат, которые позволят вам выполнить более сложные задачи, например, определять день недели для конкретной даты.

В табл. 4.5 приведено несколько примеров.

Таблица 4.5. Условия на значения для дат

Сравнение	*Пример условия*	*Описание*
Меньше чем	<#1/30/2007#	Дата до 30 января 2007 г.
Больше чем	>#1/30/2007 5:30 PM#	Любая дата после 30 января 2007 г., или 30 января 2007 г. после 17:30
Меньше или равна	<=#1/30/2007#	Дата до 30 января 2007 г. или первая секунда 30 января 2007 г.
Больше или равна	>=#1/30/2007#	30 января 2007 г. или любая более поздняя дата
Больше текущей даты	>Date()	Сегодня или более поздняя дата
Меньше текущей даты	<Date()	Вчера или более ранняя дата
Больше текущей даты (и времени)	>Now()	Сегодня после текущего времени или любая дата в будущем
Меньше текущей даты (и времени)	<Now ()	Сегодня до настоящего момента или любая дата в прошлом

Проверка допустимости текста

В случае текста условие на значение позволяет задать начальный или конечный символ текста или наличие определенных символов в строке. Все эти задачи решаются с помощью оператора Like, сравнивающего введенный текст с образцом.

Следующее условие требует начинать поле с буквы "R":

Like "R*"

Звездочка обозначает отсутствие символов или присутствие нескольких символов. Таким образом, полное условие заставляет программу Access проверять, начинается ли строка с буквы "R" (или "r"), за которой могут следовать символы или нет.

Очень похожее условие можно применять для проверки завершающих символов фрагмента

текста:

Like "*ed"

Это условие считает корректными значения talked, walked и 34z%($)#ed и не пропускает значения talking, walkable или 34z%($)#.

Не столь распространенный прием - использование нескольких звездочек. В следующем выражении требуется наличие букв "a" и "b" (именно в таком порядке, но не обязательно сразу друг за другом) в любом месте строки текста:

Like "*a*b*"

Наряду со звездочкой в операторе Like могут использоваться и некоторые другие символы. Можно применять ?, соответствующий единичному символу, что очень удобно, если известна длина текста или позиция определенной буквы в тексте. Далее приведено условие на значение для восьмисимвольного кода изделия, заканчивающегося 0ZB:

Like "?????0ZB"

У символа # аналогичная роль, но он представляет цифру. Таким образом, следующее правило верификации определяет код изделия, заканчивающийся комбинацией символов 0ZB, которой предшествует пять цифр:

Like "#####0ZB"

И наконец, вы можете ограничить значение любого символа набором определенных букв или символов. Для этого нужно заключить допустимые символы в квадратные скобки.

Предположим, что ваша компания использует восьмисимвольный код изделия, который всегда начинается с "A" или "E". Далее приведено необходимое условие на значение:

Like "[AE]???????"

Обратите внимание на то, что фрагмент [AE] представляет один символ, который может принимать значение A или E. Если вы хотите разрешить символы A, B, C, D, следует написать в условии [ABCD] или воспользоваться удобной сокращенной формой [A-D], означающей разрешение любого символа от A до D, включая A и D.

Далее приведено условие на значение, разрешающее ввод семибуквенного слова и запрещающего цифры и другие символы. Оно формируется семикратным повторением кода [A- Z] (который разрешает любую букву).

Like [A-Z] [A-Z] [A-Z] [A-Z] [A-Z] [A-Z] [A-Z]

Как видите, условия на текстовые значения не всегда приятны для глаз. Они не только могут разрастаться до неимоверных размеров, но и не могут задать многих ограничений. Например, невозможно задать допустимое изменение длины текста между заданными максимумом и минимумом. Невозможно также различать строчные и прописные буквы.

Примечание

Множество подобных ограничений можно обойти, применяя некоторые функции, предоставляемые программой Access. В *разд. "Текстовые функции" главы 7* вы узнаете, как с помощью функций вырезать фрагменты текста, проверять длину строки, использовать прописные буквы в тексте и многое другое.

Комбинирование условий на значения

Независимо от типа данных вы можете комбинировать ваши условия двумя способами. Используя ключевое слово And, можно создать правило верификации, содержащее два условия. Этот прием очень полезен, т. к. у поля может быть только одно условие на значение.

Для использования ключевого слова And просто запишите два правила верификации и вставьте между ними слово And. Неважно, какое правило вы укажете первым. Далее приведено условие на значение даты, которая должна была наступить до текущей даты, но после 1 января 2000 г.:

<Date( ) And >#l/l/2000#

Можно также использовать ключевое слово Or для принятия значения, удовлетворяющего одному из условий. В следующем правиле разрешены числа, большие 1000 или меньшие-1000:

>1000 Or < -1000

4.2.6. *Создание условия на значение для таблицы*

Условия на значения всегда применяются к отдельному полю. Но проектировщики БД часто нуждаются в средствах сравнения значений разных полей. Предположим, что у вас есть таблица **Orders** (заказы), в которой регистрируются покупки в вашем магазине по продаже фирменных носков с монограммой. В таблице **Orders** вы используете два поля: DateOrdered (дата заказа) и **DateShipped** (дата доставки). Для того чтобы все было как следует, необходимо, чтобы дата из поля **DateOrdered** была более ранней, чем дата из поля DateShipped. Помимо всего прочего, как доставить изделие, прежде чем кто-то его закажет?

Поскольку это правило верификации включает в себя два поля, единственный способ вставить его - создать условие на значение для всей таблицы. Табличные правила верификации могут применять все приемы, уже известные вам, и удалять значения из любого поля текущей записи.

Далее приведен алгоритм создания условия на значение для таблицы.

1. В Конструкторе выберите на ленте **Работа с таблицами | Конструктор → Показать или скрыть → Страница свойств** (Table Tools | Design → Show/Hide → Property Sheet).

Справа в окне программы появляется страница с дополнительными параметрами (рис. 4.16).

Примечание

Для таблицы можно создать только одно правило верификации. Это ограничение может показаться проблемой, но ее легко обойти с помощью ключевого слова And *(см. разд. "Комбинирование условий на значения" ранее в этой главе)* и объединения того количества условий, которое вам нужно. Правило верификации может быть трудным для чтения, но при этом работать без сбоев.

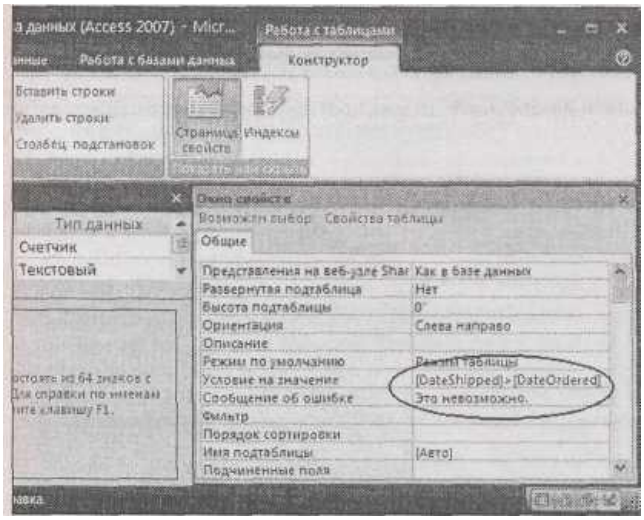


Рис. 4.16. В **Окне свойств** отображена некоторая информация о таблице в целом, включая параметры сортировки и фильтрации, примененные вами на листе данных, а также условие на значение. В данном примере правило не допускает доставку заказов до того, как они будут заказаны

2. На вкладке **Свойства таблицы** задайте **Условие на значение.**

В условии на значение для таблицы можно использовать все уже известные вам ключевые слова. Обычно в условии для таблицы сравнивается несколько полей. Условие на значение [DateOrdered] < [DateShipped] гарантирует, что в ноле **DateOrdered** более ранняя дата, чем используемая в поле **DateShipped.**

При ссылке на поле в условии на значение для таблицы имена полей следует заключать в квадратные скобки. Таким образом, программа Access может установить разницу межу полями и функциями (например, функцией Date ( ), о которой вы узнали в *разд. "Задание значений по умолчанию "далее в этой главе*).

3. Задайте текст **Сообщения об ошибке.**

Это сообщение об ошибке выводится на экран, если условие не выполняется. Оно аналогично сообщению об ошибке для условия на значение поля.

Когда вставляется новая запись программа Access сначала проверяет условия на значения

поля. Если данные успешно проходят проверку (и у них правильные типы), Access проверяет условие на значение для таблицы.

Подсказка

После вставки условия на значение для таблицы вы, возможно, захотите закрыть **Страницу** свойств, чтобы увеличить свободное пространство в окне **Конструктора,** для этого выберите на ленте **Работа с таблицами | Конструктор → Показать или скрыть → Страница свойств.**

4.3. *Подстановки*

В БД даже незначительные вариации могут создавать большие неприятности. Допустим, вы управляете компанией International Cinnamon, пекарней многонациональной сети, выпускающей булочки с корицей и имеющей сотни заказов в день.

В таблице Orders у вас есть следующие записи:

Quantity (количество)	***Product (изделие)***
10	Frosted Cinnamon Buns
24	Cinnamon Buns with Icing
16	Buns, Cinnamon (Frosted)
120	FCBs
…	

(В данном примере другие поля, такие как столбец **Код** и сведения о клиенте, сделавшем заказ, опущены.)

Все заказы, приведенные в таблице, означают одно и то же: различные количества вкусных, покрытых сахарной глазурью булочек с корицей, Но текст в столбце **Product** слегка отличается. Эти отличия не создают проблем для простых смертных (например, вы без труда выполните эти заказы), если вы захотите в дальнейшем проанализировать характеристику реализации (sales performance), возникнут неприятности. У вас не будет возможности сообщить программе Access о том, что Frosted Cinnamon Bun и FCB - одно и то же, булочка с корицей, программа считает их разными изделиями. Если вы попытаетесь подсчитать наиболее популярные изделия или проверить долгосрочные тренды объемов продаж, у вас ничего не выйдет.

Примечание

Данный пример акцентирует внимание на том, что вы узнали раньше. А именно, программы управления БД придирчивы и серьезны, они не терпят мельчайших несоответствий. Если вы хотите, чтобы ваши БД приносили пользу, то должны быть уверены в том, что в них содержится первоклассная информация.

Подстановки - еще одно средство, позволяющее стандартизировать ваши данные.

Подстановки, в первую очередь, позволяют вставить значение в поле из подготовленного списка возможных вариантов. При надлежащем применении это средство решает проблему, возникшую в таблице **Orders, -** вам просто нужна подстановка, включающая все виды изделий, которые вы продаете. В этом случае вместо набора вручную названия изделия вы можете выбрать из списка Frosted Cinnamon Buns (глазированные булочки с корицей). Вы не только экономите время, но и избегаете таких названий, как FCB, тем самым гарантируя непротиворечивость списка заказов.

У программы Access два основных типа списков подстановок: списки с набором фиксированных значений, заданных вами, и списки, полученные из связанной таблицы. В следующем разделе вы узнаете, как создавать список первого типа. В *главе 5* вы перейдете ко второму типу.

Примечание

Подстановки не поддерживают следующие типы данных: **Поле MEMO, Дата/время, Денежный, Счетчик, Логический, Объект OLE, Гиперссылка и Вложение.**

4.3.1. *Создание простого списка подстановок, состоящего из констант*

Простые списки подстановок имеют смысл, если у вас короткий простой список, не нуждающийся в частых корректировках. Отличным примером может служить список префиксов в

адресе, обозначающих штаты. В данном случае это набор 50 двухбуквенных сокращений (AL, AK, AZ и т. д.).

Для опробования процесса создания списка, состоящего из перечисленных далее действий, можно воспользоваться таблицей **Bachelors** (холостяки), входящей в примеры к данной главе, размещенные в Интернете (см. файл БД DatingService.accdb). Можно перейти непосредственно к конечному результату, просмотрев файл DatingServiceLookup.accdb.

1. Откройте таблицу в **Конструкторе.**

Если вы используете файл DatingService.accdb, откройте таблицу **Bachelors.**

2.Найдите поле, в которое нужно вставить список подстановок.

В таблице **Bachelors** это поле **State.**

3.Убедитесь в том, что у поля корректный тип данных.

Подстановки применяются чаще всего для данных **Текстового** и **Числового** типов.

4.Выберите в списке типов данных **Мастер подстановок.** Это действие на самом деле не изменяет заданный тип данных. Оно лишь сообщает программе Access о том, что вы хотите запустить мастер Создание подстановки, базирующийся на текущем типе данных. Как только выбран описанный вариант, на экране появляется окно мастера **Создание подстановки** (рис. 4.17).

5.Выберите переключатель **Будет введен фиксированный набор значений (I** will type in the values that I want). В *разд. "Поиск в связанных таблицах" главы* 5 описывается другой вариант: список подстановок из другой таблицы.

6.Щелкните мышью кнопку **Далее.**

В следующем окне **Создание подстановки** вы можете ввести список значений, которые следует использовать по одному в каждой строке (рис. 4.18). В данном случае вводится список сокращенных названий 50 американских штатов.

Как вы могли заметить, в списке можно заполнить несколько столбцов данных. Пока ограничимся одним столбцом. Вы узнаете, зачем использовать несколько, *разд. "Поиск в связанных таблицах" главы 5.*

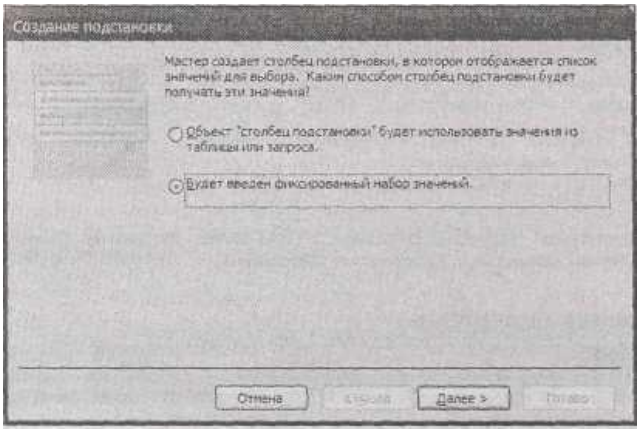


Рис. 4.17. Сначала вы выбираете источник для ваших подстановок: константы или данные из другой таблицы

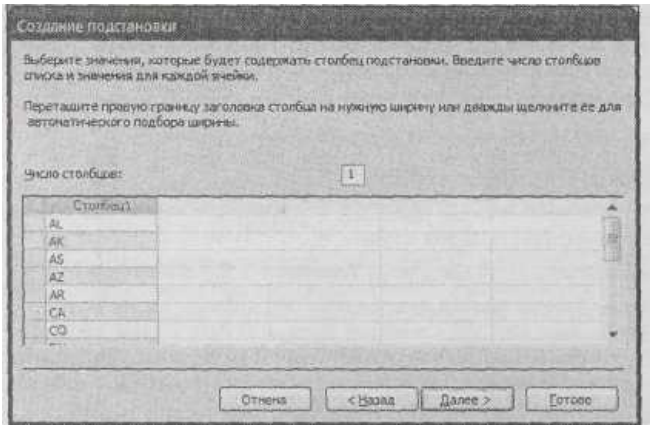


Рис. 4.18. Этот список подстановок содержит сокращенные названия всех американских штатов. Он вряд ли будет меняться в ближайшем будущем, поэтому безопаснее хранить его как набор

констант, а не в другой таблице

7. Щелкните мышью кнопку Далее.

На экране появляется последнее окно мастера **Создание подстановки.**

8. Укажите, можно ли хранить в столбце подстановки множественные значения.

Если вы разрешили наличие множественных значений, список подстановок отобразит флажок рядом с каждым элементом. Вы сможете выбрать несколько значений для одной записи, установив несколько флажков.

В поле State нет смысла разрешать множественные значения - помимо всего прочего физически человек может обитать только в одном штате (не принимая во внимание квантовую телепортацию). Но можно придумать примеры, в которых наличие множественных значений вполне оправданно. Например, в таблице **Products,** используемой компанией International Cinnamon, подстановка множественных значений позволила бы нам сформировать заказ нескольких изделий. (Вы узнаете больше о выборе нескольких значений **и** связях между таблицами в *главе* 5.)

9. Щелкните мышью кнопку **Готово.**

Перейдите в **Режим таблицы** (щелкните правой кнопкой мыши заголовок вкладки и выберите **Режим таблицы)** и сохраните изменения. На рис. 4.19 показана подстановка в действии.

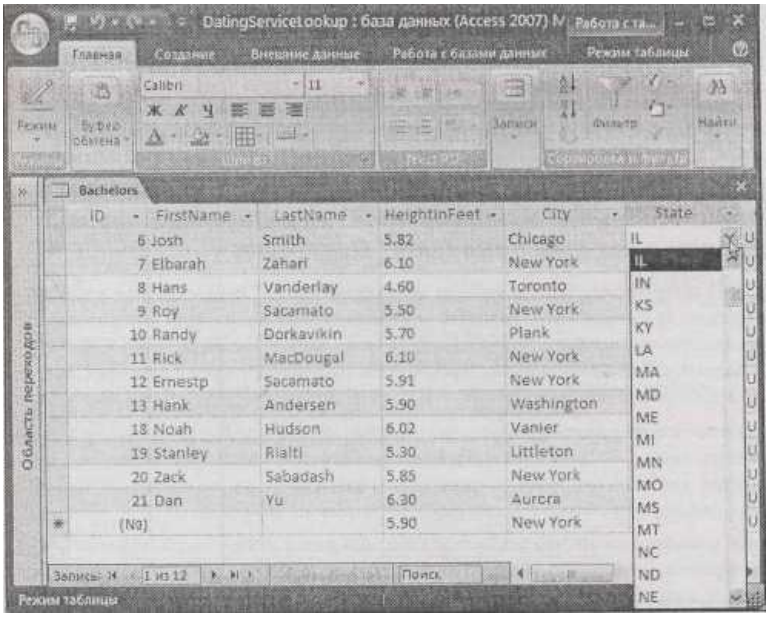


Рис. 4.19. Когда вы перейдете в поле со списком подстановок, то справа увидите стрелку, направленную вниз. Щелкните ее кнопкой мыши, и на экране появится раскрывающийся список со всеми введенными вами вариантами. Выберите один из них для вставки в поле

На профессиональном уровне.

Создание списка подстановки, использующего другую таблицу

В предыдущем примере (см. *начало данного раздела)* был создан список подстановок, сохранявшийся как один из параметров поля. Это хороший подход, но не лучшее решение. Гораздо эффективнее хранить список подстановок в отдельной таблице.

Далее приведено несколько доводов в пользу применения отдельной таблицы.

• *Можно вставлять, редактировать и удалять элементы, просто корректируя таблицу подстановок.* Даже если вам кажется, что у вас фиксированный неизменный набор значений, есть смысл подумать об отдельной таблице. Например, набор сокращенных названий штатов в предыдущем разделе, казалось бы, не требует изменений, но что, если служба знакомств выйдет на международный уровень и вам придется добавить канадские провинции в список?

• *Можно многократно использовать один и тот же список подстановок в нескольких разных полях (как в одной таблице, так и в разных).* Такой подход устраняет бесконечные операции копирования и вставки.

• *Можно хранить дополнительную информацию.* Например, вы можете хранить сокращенные названия штатов (для почтовых адресов), а отображать полные названия (для облегчения ввода данных). В *разд. "Поиск в связанных таблицах" главы* 5 вы узнаете, как это сделать.

Списки подстановок в виде таблиц немного сложнее, поскольку они включают связь таблицы - ссылку, объединяющую две таблицы вместе и (иногда) порождающую новые ограничения. *Глава 5* полностью посвящена связям таблиц, служащих ключевым компонентом любой применяемой на практике БД.

4.3.2. *Добавление новых значений в ваш список подстановок*

Когда создается подстановка, использующая константы, список предоставляет лишь перечень предложений. Вы можете игнорировать список подстановок и ввести совершенно другое значение (например, префикс штата ZI), даже если его нет в списке. Такой подход позволяет применять список подстановок как удобное экономящее время средство, которое при этом не ограничивает ваш выбор.

В большинстве случаев такая гибкость даже мешает. В таблице **Bachelors** вы, возможно, хотите помешать людям вводить в поле **State** что-либо, отличное от предлагаемого списка. В этой ситуации хотелось бы, чтобы список подстановок был так же средством проверки ошибок и верификации, препятствующим вводу посторонних значений.

К счастью, несмотря на то, что в Мастере создания подстановки такая возможность по непонятным причинам отсутствует, достаточно легко ее добавить после создания списка. Выполните следующие действия.

1. В **Конструкторе** перейдите в поле, содержащее список подстановок.
2. В области **Свойства поля** щелкните кнопкой мыши вкладку **Подстановка.**

На вкладке Подстановка представлены параметры для тонкой настройки вашего списка подстановки, большинство из которых легче задать в Мастере создания подстановки. В поле **Источник строк** (Row Source), например, можно откорректировать список предлагаемых вами значений. (Все значения расположены в одной строке, заключены в кавычки и отделяются друг от друга точкой с запятой.)

3. Задайте значение *Да* в поле **Ограничиться списком** (Limit to List). Это действие защитит вас от ввода значений, не включенных в список.
1. Можете выбрать значение *Да* в поле **Разрешить изменение списка значений** (Value List Edits).

Это действие позволит корректировать значения списка в любое время. Если в списке подстановок что-то пропущено, вы можете вставить новое значение на лету (рис. 4.20).

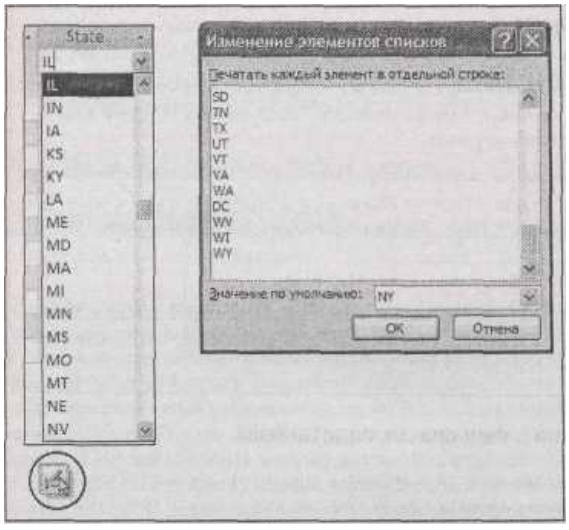


Рис. 4.20. Если задать значение поля **Разрешить изменение списка значений,** равным *Да,* во время применения списка подстановок появится пиктограмма *(слева).* Щелкните ее кнопкой мыши, и на экране откроется диалоговое окно **Изменение элементов списков** *(справа),* в котором можно откорректировать значения из списка подстановок и изменить значение по умолчанию

5. Глава 5. Связывание таблиц с помощью отношений

Таблицы, которые вы видели до сих пор, "жили" уединенно, независимо друг от друга. В реальных БД не найти подобной изоляции. В практических БД таблицы соединены друг с другом паутиной связей.

Допустим, вы намерены создать БД, способную управлять продажами в вашем магазине изделий из бисера, сделанных на заказ. Первая составляющая достаточно проста - таблица **Products** (изделия), в которой перечислены ваши товары. Но для продолжения вам придется собрать множество дополнительной информации. Проданные изделия из таблицы **Products** учитываются в таблице **Orders** (заказы). Товары из таблицы **Orders** посылаются по почте и фиксируются в таблице **Shipments** (поставки). Люди из таблицы **Customers** (клиенты) регистрируются в таблице **Invoices** (счета). Во всех этих таблицах - **Products, Orders, Shipments, Customers** и **Invoices** - содержатся порции связанной информации. В результате, если вы хотите получить ответ на обычный вопрос (например, "Сколько должна Джейн Мэлон (Jane Malone)?" или "Сколько париков из бисера продано на прошлой неделе?"), придется заглянуть в несколько таблиц.

На основании полученных вами знаний вы уже можете разработать проект такой БД. Но связи порождают возможность противоречивой информации, А если несоответствие проникнет в БД, вы уже не сможете доверять ей, как прежде.

В данной главе вы узнаете о том, как явно устанавливать связи между таблицами. Этот процесс позволит избежать распространенных ошибок, таких как противоречивость данных. Кроме того, он предоставляет мощное средство просмотра связанной информации в нескольких таблицах.

5.1. *Основы отношений между таблицами*

Одна из основных целей любой БД - разбить информацию на четкие управляемые порции. В хорошо спроектированной БД процесс завершается созданием большого числа таблиц. Несмотря на то, что в каждой из них записано что-то свое, вам часто придется путешествовать из одной таблицы в другую для того, чтобы получить всю нужную информацию.

Для того чтобы лучше понять отношения (отнюдь не романтический их вариант), рассмотрим пример. В следующем разделе представлены два способа добавления данных в БД о куклах-болванчиках: в одном есть риск избыточных данных, а другой решает эту проблему за счет надлежащего использования отношения или связи.

5.1.1. *Избыточные данные в противоположность связанным*

Вернемся к таблице **Dolls,** созданной в *главе 1* для хранения списка кукол-болванчиков. Одна из порций информации данной таблицы - поле **Manufacturer** (изготовитель), в котором записано имя компании, изготовившей каждую куклу. Несмотря на то, что это достаточно простая деталь, может оказаться, что для точной оценки стоимости куклы-болванчика вам понадобится немного больше информации о процессе изготовления. Возможно, вы захотите узнать, где находится компания-изготовитель, как долго она существует и вынуждена ли отбиваться от судебных исков разъяренных покупателей.

Если вы ленивы, то можете вставить всю эту информацию в таблицу **Dolls** так, как показано в табл. 5.1 (затемненные столбцы - новые).

Таблица 5.1. Сведения об изготовителе

ID	*Character*	*Manufacturer*	*Manufacturer-Location*	*Manufacturer-OpeningYear*	*Manu-facturer-Lawsuits*	*Purchase-Price*
342	Yoda	MagicPlastic	China	2003	No	$8.99

В первый момент вас, возможно, обеспокоит загроможденность таблицы всеми этими полями. Не волнуйтесь - это жизнь, в таблицы должны быть включены все важные детали, поэтому они порой сильно разрастаются. (Это правило проектирования БД, описанное в *разд. "Правило 3. Храните все детали в одном месте" главы 2.)* Пусть громоздкость вас не беспокоит. Можно воспользоваться такими средствами, как скрытие столбцов *(см. разд. "Скрытие столбцов " главы 3)* для устранения полей, которые вас не интересуют.

Несмотря на то что, что обилие столбцов - не повод для беспокойства, в этом примере кроется другая проблема - избыточные данные.

Подобная ситуация кажется невинной, но если добавить несколько новых строк, все окажется не столь безобидно (табл. 5.2).

Таблица 5.2. Сведения о производителе после добавления строк

ID	*Character*	*Manufacturer*	*Manu-acturer-Location*	*Manufacturer-OpeningYear*	*Manufactur er-Lawsuits*	*Purchase-Price*
342	Yoda	MagicPlastic	China	2003	No	S8.99
343	Dick Cheney	Rebobbiicans	Taiwan	2005	No	S28.75
344	Tiger Woods	MagicPlastic	China	2003	No	$2.99

Как только у вас появятся две куклы-болванчика, изготовленные одной компанией (в данном случае MagicPlastic), вы введете дублирующиеся данные - беда всех плохих БД. (Их можно распознать как нарушение правила № 4 хорошего проекта БД, описанного в *разд. "Правило 4. Избегайте дублирования данных" главы 2.)* Потенциальные проблемы не-скончаемы.

■ Если компания MagicPlastic переводит свои предприятия из Китая и Южную Корею, вам нужно обновлять целую группу записей о куклах-болванчиках. Если вы использовали две таблицы со связанной информацией (как вы увидите далее), вам придется бороться только с одной записью.

■ Крайне легко обновить информацию об изготовителе в одной записи о кукле-болванчике и пропустить ее в другой. Если вы допустили такую ошибку, то столкнетесь с противоречивыми данными в вашей таблице, которые еще хуже, чем дублирующаяся информация.
Существенно то, что ваши сведения об изготовителе обесценятся, поскольку неизвестно, в какой записи верные данные, и поэтому нельзя ничему верить.

« Если вы хотите хранить в вашей БД еще больше сведений, касающихся изготовителя (например, контактный телефон), вам придется обновлять вашу таблицу **Dolls** и корректировать каждую отдельную запись. Ваша семья может вас не увидеть в течение нескольких недель.

■ Если же понадобится информация об изготовителях (а не о куклах), вы потерпите неудачу. Например, вам не удастся распечатать список всех изготовителей кукол-болванчиков в Китае (по крайней мере, не так легко, как хотелось бы).

Проблема вполне очевидна. Из-за желания хранить слишком много подробностей в одном месте в одной таблице объединяется информация, которую лучше всего хранить в двух отдельных таблицах. Для исправления этого проекта нужно создать две таблицы со связанными данными. Например, можно сформировать таблицу **Dolls,** как показано в табл. 5.3, и отдельную таблицу **Manufacturers** с подробными данными об изготовителях (табл. 5.4).

Таблица 5.3. Данные в таблице Dolls

ID	Character	Manufacturer	PurchasePrice
342	Yoda	MagicPlastic	$8.99
343	Dick Cheney	Rebobblicans	$28.75
344	Tiger Woods	MagicPlastic	$2.99

Таблица 5.4. Данные в таблице Manufacturers

ID	Manufacturer	Location	OpeningYear	Lawsuits
1	MagicPlastic	China	2003	**No**
2	Rebobbitcans	Taiwan	2005	**No**

Этот проект позволяет легко работать отдельно с двумя типами информации (куклы и изготовители). Он также устраняет риск дублирования. В данном простом примере пре-имущества незначительны, но в таблице с сотнями или тысячами записей о куклах-болванчиках (и намного меньшим числом изготовителей) разница весьма существенна.

Теперь если компания MagicPlastic переезжает в Южную Корею, вам нужно обновить данные поля **Location** только в одной записи, вместо множества экземпляров в перегруженной таблице **Dolls.** Вы также гораздо легче сможете формировать запросы *(см. главу 6),* объединяющие данные подходящими и удобными способами. (Например, вы сможете подсчитать, сколько потратили на всех кукол от компании MagicPlastic, и сравнить эту сумму с затратами на кукол, созданных другими компаниями.)

Примечание

В программу Access включено средство, пытающееся отследить дублирующиеся данные в таблице и помочь вам разделить поля на две связанные таблицы. (Для знакомства с ним выберите **Работа с базами данных → Анализ → Анализ таблицы** (Database Tools → Analyze → AnalyzeTable)) Несмотря на то, что теоретически это хорошая идея, данное средство не слишком полезно. Гораздо полезнее, если вам понятна проблема дублирования данных, выявления дублирующихся данных и с самого начала создание хорошо спроектированных БД.

5.1.2. *Совпадающие поля: связующее звено отношения*

Данная БД о куклах-болванчиках - пример отношения. Верный его признак - две таблицы с одинаковыми полями. В данном случае такой объект - поле **Manufacturer,** существующее и в таблице **Dolls,** и в таблице **Manufacturers.**

Примечание

В приведенном примере у полей, связывающих две таблицы, одно и то же имя **Manufacturer** в обеих таблицах. Но это не обязательно. У таких полей могут быть разные имена при условии, что у них один тип данных.

С помощью этих связанных полей можно начать с записи в одной таблице и затем найти связанную информацию в другой таблице. Далее объясняется принцип действия.

■ Начиная с таблицы **Dolls,** выберите куклу, которая вас интересует (скажем, Yoda (Йода)). Вы получите дополнительную информацию о производителе куклы Yoda, найдя название компании "MagicPlastic" в таблице **Manufacturers.**

■ Начиная с таблицы **Manufacturers,** выберите изготовителя (скажем, компания Rebobblicans). Вы можете найти все изделия этого производителя с помощью поиска названия "Rebobblicans" в таблице **Dolls.**

Другими словами, взаимосвязь таблиц предоставляет больше гибкости, позволяя задавать больше вопросов, касающихся ваших данных, и получать более полные ответы.

5.1.3. *Связывание с помощью столбца Код (ID)*

В предыдущем примере таблицы **Dolls** и **Manufacturers** связаны друг с другом полем **Manufacturer,** в котором хранится имя компании-изготовителя. Такой выбор кажется оправданным в данном проекте - до тех пор, пока вы не задумаетесь на пару минут о возможных ошибках. Специалисты по проектированию БД известны тем, что тратят недели на обдумывание неотвратимых бедствий.

В данном случае есть две потенциальные проблемы.

■ У двух изготовителей одно и то же название компании. Как узнать, какая из компаний сделала куклу?

■ Изготовитель куплен другой компанией, и у него изменилось название. Внезапно появляется длинный список записей из таблицы **Dolls,** которые нуждаются в корректировке.

Возможно, вы узнали эти проблемы, поскольку они аналогичны тем, с которыми вы сталкивались, когда речь шла о выборе первичного ключа *(см. разд. "Первичный ключ "главы 2).* Как вы уже знаете, трудно найти данные, гарантированно уникальные и неизменные. Вместо риска возникновения проблем надежнее положиться на поле с типом данных **Счетчик,** содержащее код (ID number), генерируемый программой Access.

Интересно, что тот же подход применяется при связывании таблиц. Для ссылки на запись в другой таблице не следует использовать любую порцию данных - вместо этого нужно применять уникальный код, указывающий на нужную запись. В табл. 5.5 приведена переконструированная таблица **Dolls,** ставшая более корректной за счет замены поля **Manufacturer** полем **ManufacturerID.**

Таблица 5.5. Данные в таблице Dolls после ее реконструирования

ID	Character	ManufacturerlD	PurchasePrice
342	Yoda	**1**	$8.99
343	Dick Cheney	**2**	$28.75
344	Tiger Woods	**1**	$2.99

Если вы вернетесь назад и посмотрите на таблицу **Manufacturers** (см. табл. 5.4), то быстро обнаружите, что изготовитель с кодом или идентификационным номером 1 - это компания MagicPlastic.

Этот проект - универсальный стандарт БД. Но у него есть два явных недостатка:

■ пользователь, вставляющий записи в таблицу **Dolls,** возможно, не знает кодов всех изготовителей;

- когда смотришь на таблицу **Dolls,** нельзя сказать, кем изготовлена каждая кукла.

Для решения обеих проблем воспользуйтесь подстановкой. Подстановки отображают соответствующие сведения об изготовителе в таблице **Dolls,** а также позволяют выбрать изготовителя из списка в тот момент, когда добавляется запись или корректируется поле **ManufacturerlD.** (Вы узнали, как использовать подстановки со списками констант в *разд. "Создание простого списка подстановок, состоящего из констант" главы 4.* О том, как применять подстановки, сведя вместе связанные таблицы, такие как **Dolls** и **Manufacturers,** будет рассказано в *разд. "Поиск в связанных таблицах" далее в этой главе.*)

Подсказка

Еще больше возможностей предоставляет запрос-объединение (join query) *(см. разд. "Запросы и связанные таблицы" главы 6).* Он может предоставить все подробные сведения об изготовителе наряду с данными о кукле, поэтому можно отобразить их бок о бок.

5.1.4. *Отношение типа "родитель - потомок"*

Нет-нет, это не окольный путь в психологию положительных эмоций доктора Фила (Dr. Phil). Фанаты БД применяют обозначения "родитель" и "потомок" для идентификации двух таблиц в отношении и определения каждой из них.

Вот чем объясняется такая аналогия. Как вы, вне всякого сомнения, знаете, в реальной жизни у родителя может быть любое количество детей. А у ребенка только один "комплект" родителей. Это же правило действует и в БД. В БД о куклах-болванчиках единственная запись изготовителя может быть связана с любым количеством записей о куклах. Но каждая запись о кукле ссылается на единственного изготовителя. Итак, в соответствии со странными социальными законами в мире БД **Manufacturers** - родительская таблица или таблица-родитель, а **Dolls** - дочерняя таблица или таблица-потомок. Обе таблицы связаны отношением "родитель-потомок".

Подсказка

Не относитесь слишком серьезно к аналогии "родитель - потомок". Она неидеально соответствует биологической реальности. Например, в БД о куклах-болванчиках можно создать изготовителя, не связанного ни с какой куклой {другими словами, родитель без детей). Вы все равно назовете такую запись родительской, поскольку она - часть таблицы-родителя.

Важно понять, что нельзя поменять местами таблицу-родитель и таблицу-потомок без изменения отношения между ними. Неправильно считать таблицу **Dolls** родительской, а таблицу **Manufacturers** - дочерней. Вы можете заметить, что такое допущение разрушает аналогию "родитель - потомок": у любой куклы не может быть более одного изготовителя и производитель не ограничивается созданием одной куклы. Во избежание проблем и всякого рода туманных измышлений необходимо четко представлять, какая таблица - родитель, а какая - потомок.

Подсказка

Если возникают трудности при определении родительской таблицы, существует простое пра-

вило, направляющее в нужное русло. В дочерней таблице всегда содержится порция идентифицирующей информации из родительской таблицы. В БД о куклах-болванчиках таблица **Dolls** содержит поле **ManufacturerlD.** С другой стороны, в таблице **Manufacturer** нет никаких данных о куклах.

Если у вас есть друзья, хорошо разбирающиеся в БД, термин "отношение родитель-потомок" вы слышите нечасто. Этот же тип связи называют отношением "один-ко-многим" (где один представляет родителя, а многие - детей, поскольку единственная родительская запись в одной таблице может быть связана с несколькими дочерними записями в другой). Это самое распространенное отношение, но не единственное - о двух других типах отношений вы узнаете в *разд. "Отношение "один-к-одному""* и *"Отношение "многие-ко-многим"" далее в этой главе.*

Примечание

Отношения настолько распространены в современных БД, что программы, подобные Access, часто называют *системами управления реляционной базой данных* (СУРБД, RDBMS). БД без отношений так же часто встречается, как морской курорт в Огайо.

5.2. *Применение отношений*

Отношение, или связь между **Dolls** и **Manufacturers,** подразумеваемая или неявная, - это означает, что вы знаете о существовании связи, а программа Access нет. Подобная организация не устраивает проектировщиков БД. Вместо этого они почти всегда определяют устанавливаемые связи явно. Когда создается явная связь, вы явно сообщаете Access о том, как взаимосвязаны две таблицы. Программа сохраняет сведения об этой связи в файле БД,

У вас есть серьезные основания открыть ваши связи. После того как программа Access узнает о них, она может лучше отслеживать ошибки. Она также может предоставить удобные средства просмотра связанных данных и корректировки связанных полей. Вы познакомитесь со всеми этими средствами в следующих разделах. Но сначала необходимо научиться определять связь, или отношение.

5.2.1. *Определение отношения*

Приведенные далее действия можно выполнить, используя файл Bobblehead.accdb, включенный в примеры к данной главе, размещенные в Интернете. Файл содержит таблицы **Dolls** и **Manufacturers** в первоначальной форме (без определенных отношений). В файле БД BobbleheadRelationships.accdb представлен окончательный продукт: две таблицы с корректным отношением.

Далее перечислены действия, необходимые для определения связи или отношения.

1. Любое отношение связывает два поля, находящиеся в разных таблицах. Сначала нужно определить в родительской таблице поле, которое следует использовать.

В хорошо спроектированной БД в родительской таблице применяется поле первичного ключа *(см, разд. "Первичный ключ" главы 2).* Например, в таблице **Manufacturers** используется поле ID, однозначно определяющее каждого изготовителя.

2. Откройте дочернюю таблицу в **Конструкторе.** (Самый быстрый способ - щелкнуть правой кнопкой мыши таблицу в **Области переходов** и выбрать строку **Конструктор.)**

В данном примере дочерняя таблица - **Dolls.**

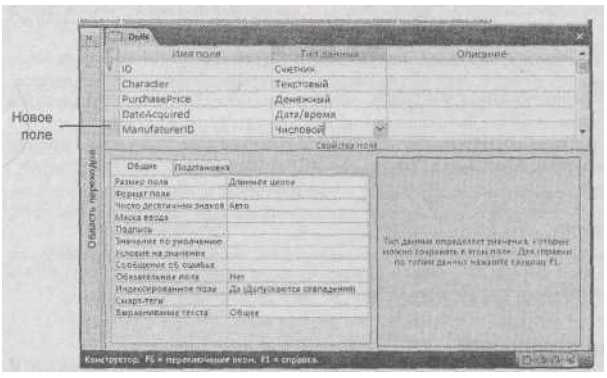


Рис. 5.1. В таблице **Dolls** должно быть поле, определяющее изготовителя данной куклы. Имеет смысл вставить новое поле **ManufacturerlD**. Задайте тип данных **Числовой** и размер поля Длинное целое, чтобы поле соответствовало полю **ID** в таблице **Manufacturers**. После вставки этого поля следует заполнить его правильными данными. (В записи для каждой куклы следует указать идентификационный номер соответствующего изготовителя.)

3. Создайте нужное вам поле в таблице-потомке, если его еще нет.

Каждая дочерняя запись создает ссылку, сохраняя порцию данных, указывающую на запись в таблице-родителе. Вы должны вставить новое поле для хранения этой информации, как показано на рис. 5.1.

Примечание

У полей, которые вы связываете в родительской и дочерней таблицах, должны быть совместимые типы данных. Но существует небольшое затруднение. В поле родительской таблицы используется тип данных **Счетчик,** а поле дочерней таблицы должно быть **Числового** типа (с размером поля *Длинное целое).* На диске типы **Счетчик** и *Длинное целое в* действительности хранят одинаковые данные. Но тип **Счетчик,** когда вы создаете запись, сообщает программе Access о необходимости заполнения поля новым автоматически генерируемым значением. Это поведение, очевидно, не годится при заполнении поля **ManufacturerlD** в таблице **Dolls.**

4. Закройте обе таблицы.

Программа Access предложит сохранить внесенные изменения. Теперь ваши таблицы готовы к установке связи.

5. Выберите **Работа с базами данных → Показать или скрыть → Схема данных** (Database Tools → Show/Hide → Relationships).

Программа Access откроет на ленте новую вкладку **Работа со связями** (Relationships). Это специализированное окно, в котором можно определить связи между всеми таблицами вашей БД. В данном примере вы создадите единственную связь, но эту вкладку можно применять для формирования множества связей.

Прежде чем программа Access разрешит вам работать на вкладке **Работа со связями,** она выведет на экран диалоговое окно **Добавление таблицы** (Show Table), запрашивая необходимые вам для работы таблицы (рис. 5.2).

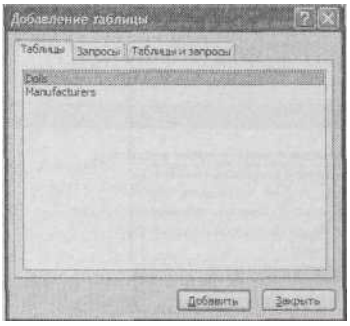


Рис. 5.2. На вкладке **Работа со связями** можно отобразить сколько угодно таблиц. Следите за тем, чтобы не вставить одну и ту же таблицу дважды (это не нужно и вводит в заблуждение)

6. Вставьте в рабочую область таблицу-родитель и таблицу-потомок.

Порядок указания таблиц не важен. Для выбора таблицы укажите ее в списке и щелкните мышью кнопку **Добавить** (или просто дважды щелкните кнопкой мыши имя таблицы).

Любая таблица на вкладке **Работа со связями** отображается в виде небольшого прямоугольника, в котором перечислены все поля таблицы. Если между таблицами уже определены связи, они отображаются как соединительные линии.

7. Щелкните мышью кнопку **Закрыть** (Close).

Теперь можно разместить таблицы на вкладке **Работа со связями** (рис. 5.3). На этой вкладке отображается схема данных - это холст, на котором вы "рисуете" устанавливаемые связи

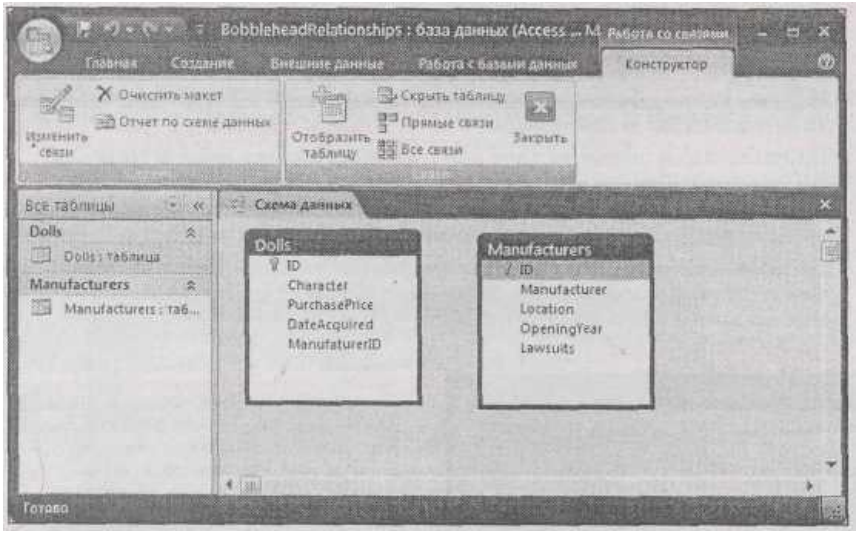


Рис. 5.3. Вставленные таблицы можно переместить с помощью мыши в любое место окна. Если в вашей БД много связей, данная возможность позволит упорядочить таблицы таким образом, чтобы были четко видны все связи. Для удаления таблицы из схемы данных щелкните ее правой кнопкой мыши и выберите команду **Скрыть таблицу** (Hide Table). Для вставки другой таблицы щелкните правой кнопкой мыши свободное пространство и выберите команду **Добавить таблицу** для вывода на экран диалогового окна **Добавление таблицы**

Подсказка

Программа Access позволяет быстро изменить структуру таблицы, открытую на вкладке **Работа со связями.** Просто щелкните правой кнопкой мыши прямоугольник нужной таблицы и выберите команду **Конструктор таблиц.**

8. Для определения связи найдите поле, которое используется в таблице-родителе. Перетащите это поле на поле в таблице-потомке, с которым вы хотите его связать.

В данном случае поле **ManufacturerID** в таблице **Dolls** (потомок) связывается с полем ГО в таблице **Manufacturers** (родитель). Итак, переместите поле ID (в прямоугольнике **Manufacturers)** на поле **ManufacturerID** (в прямоугольнике **Dolls).**

Подсказка

Можно перемещать и по-другому (из потомка в родителя). В любом случае программа Access создает одинаковую связь, или отношение.

Когда вы отпустите кнопку мыши, на экране появится диалоговое окно **Изменение связей** (Edit Relationships) (рис. 5.4).

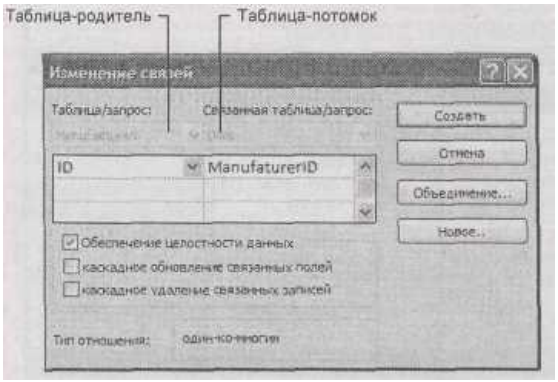


Рис. 5.4. Программа Access способна правильно определить таблицу-родитель (показанную в поле **Таблица/запрос** (Table/Query)) и таблицу-потомок (показанную в поле **Связанная таблица/запрос** (Related Table/Query)), когда вы соединяете два поля. Она идентифицирует поле в родительской таблице, поскольку у нее есть первичный ключ или уникальный индекс. Если в диалоговом окне **Изменение связей** что-то не так, прежде чем продолжить, вы можете поменять местами таблицы или заменить поля, используемые для

создания связи

9. Если вы хотите предотвратить возможные ошибки, установите флажок **Обеспечение целостности данных** (Enforce Referential Integrity). (Это всегда хорошая мысль.)

Этот параметр включает улучшенную проверку ошибок, запрещающую внесение изменений, способных нарушить правила связи (например, включение в таблицу куклы, ссылающейся на несуществующего изготовителя). Вы узнаете больше о целостности и двух параметрах для каскадных изменений в *следующем разделе.* Пока лучше установить флажок **Обеспечение целостности данных,** а остальные флажки оставить сброшенными.

10. Щелкните мышью кнопку **Создать** (Create).

Это действие создаст связь, соединяющую две таблицы. Она появляется на схеме в виде линии (рис. 5.5).

Подсказка

Если вы установите флажок **Обеспечение целостности данных** (см. пункт 9), программа Access проверит все имеющиеся в таблице данные на соблюдение правил связи. Если обнаружатся данные, их нарушающие, программа предупредит вас о проблеме и откажется продолжать. В этот момент лучшая стратегия - создать связь без обеспечения целостности, откорректировать неверные данные и позже отредактировать связь, установив флажок **Обеспечение целостности данных.**

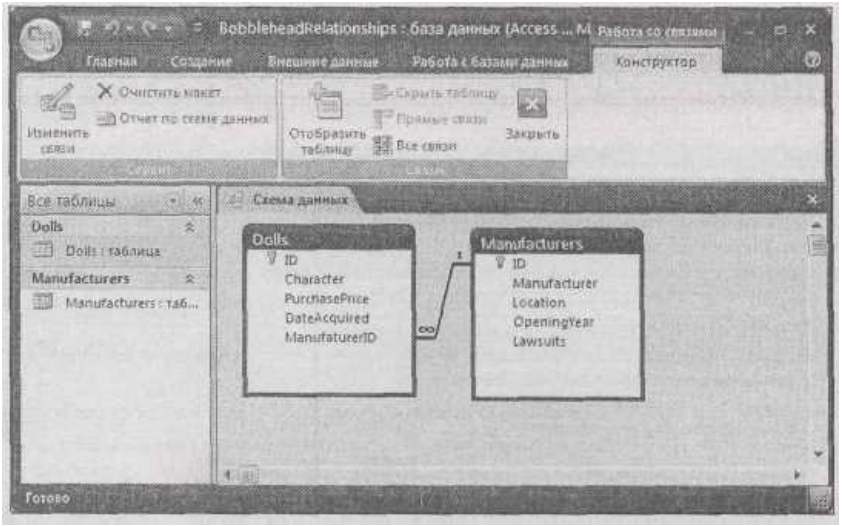


Рис. 5.5. Программа Access соединяет линией связанные поля на вкладке **Работа со связями,** маленькие значки 1 и бесконечность (∞) дают возможность в данном отношении "один-ко-многим" различить стороны "один" и "многие". Для редактирования связи дважды щелкните кнопкой мыши линию связи. Для того чтобы полностью удалить ее, щелкните дважды кнопкой мыши линию связи и выберите команду **Удалить**

11. Закройте вкладку **Работа со связями.** (Вы можете щелкнуть мышью кнопку в правом верхнем углу вкладки или выбрать **Работа со связями | Конструктор → Связи → Закрыть.**)

Программа Access спросит, хотите ли вы сохранить макет вкладки **Работа со связями.** На самом деле речь идет о сохранении схемы данных, созданной вами. Неважно, что вы выберите, связь останется в БД, и вы сможете использовать ее тем же способом. Единственное различие заключается в возможности быстрого просмотра или редактирования связи на вкладке **Работа со связями.**

Если вы выберете сохранение схемы данных, в следующий раз при переходе на вкладку **Работа со связями** (при выборе **Работа с базами данных → Показать или скрыть → Схема данных)** вы увидите ту же структуру таблиц. Это функциональная возможность удобна.

Если вы откажетесь от сохранения схемы данных, в следующий раз вашей задачей будет повторное создание схемы, включая добавление нужных вам таблиц и организацию их в окне (хотя вам не придется повторно переопределять связи). Этот процесс немного более трудоемкий.

Подсказка

Многие профессиональные проектировщики БД предпочитают сохранять схему данных, по-

скольку хотят видеть все свои связи на вкладке **Работа со связями** такими, какими они их оставили. Но реальные БД часто обрастают запутанной паутиной связей. В такой ситуации вы можете отказаться от сохранения всей схемы и сможете сосредоточить внимание на нескольких таблицах.

5.2.2. *Редактирование связей*

Когда вы в следующий раз захотите изменить или добавить связи, вы должны тем же способом вызвать на экран окно **Работа со связями** (выберите **Работа с базами данных → Показать или скрыть → Схема данных).**

Если вы решили сохранить схему данных (в пункте 11 предыдущего раздела), вставленные вами таблицы появятся автоматически в том виде, в каком вы их оставили. Если вы хотите поработать с таблицами, пока не включенными ни в какие связи, их можно добавить в схему, щелкнув правой кнопкой мыши где-нибудь на свободном пространстве и выбрав команду **Добавить таблицу.**

Если выбрано сохранение схемы данных, вы можете использовать несколько приемов быстрого возвращения ваших таблиц на экран:

■ перетащите ваши таблицы прямо из **Области переходов** на вкладку **Работа со связями;**

■ выберите **Работа со связями | Конструктор → Связи → Все связи** (Relationship Tools | Design → Relationships → All Relationships) для отображения всех таблиц, включающих связи, созданные вами ранее;

■ добавьте таблицу на схему, отметьте ее и затем выберите **Работа со связями | Конструктор → Связи → Прямые связи** (Relationship Tools ⎪ Design → Relationships → Direct Relationships) для вывода на экран таблиц, связанных с данной таблицей.

Как вы уже знаете, для создания новых связей можно использовать вкладку ленты **Работа со связями.** Созданные связи можно редактировать. Для этого щелкните правой кнопкой мыши линию, представляющую связь, и выберите команду **Изменить связь** (Edit Relationship). (Это делается легким щелчком пальца но кнопке мыши. Если в меню нет команды **Изменить связь,** вы просто не попали на линию связи.) Для удаления связи щелкните кнопкой мыши линию связи и выберите команду **Удалить.**

Примечание

Обычно редактирование связи заключается в изменении параметров, относящихся к целостности данных на уровне ссылок (referential integrity) или ссылочной целостности, о которой вы узнаете в *следующем разделе.*

5.2.3. *Целостность на уровне ссылок*

Теперь, когда вы собрались определять связи, самое время узнать, какие преимущества вы приобретаете. Как и в реальном мире, отношения налагают некоторые ограничения. В мире БД эти правила называются *целостностью данных на уровне ссылок.* Все вместе они гарантируют постоянную непротиворечивость связанных данных.

Примечание

Целостность на уровне ссылок вступает в действие, только если у вашей связи установлен флажок **Обеспечение целостности данных** (см. рис. 5.4). Без этой детали вы можете безумствовать и вводить противоречивые данные.

В примере с куклами-болванчиками целостность данных требует, чтобы все изготовители, на которых вы ссылаетесь в таблице **Dolls,** были включены в таблицу **Manufacturers.** Другими словами, ни при каких обстоятельствах не должно быть записи о кукле-болванчике, ссылающейся на несуществующего изготовителя. Этот сорт ошибок мог бы вызвать тяжелейшие сбои программы управления БД.

Для усиления влияния этого правила программа Access запрещает следующие три действия:

■ добавление куклы-болванчика, ссылающейся на несуществующего изготовителя;

■ удаление изготовителя, на которого ссылаются одна или несколько записей о куклах-болванчиках (если такую запись удалить, вы останетесь с куклой, ссылающейся на несуществующего изготовителя);

■ корректировку записи об изготовителе, заключающуюся в изменении его кода (идентификационного номера), в результате чего он перестает совпадать с кодом изготовителя в связанных записях о куклах-болванчиках. (Такая модификация не проблема, если вы используете поле типа **Счетчик,** поскольку вы не сможете изменить значения этого типа после создания записи.)

Примечание

Если нужно вставить сведения о новой кукле, сделанной новым изготовителем, следует сначала добавить запись об изготовителе и потом запись о кукле. Никаких проблем не возникает, если вы добавляете записи о компаниях-изготовителях, для которых нет соответствующих записей о куклах - в итоге вполне резонно вставить запись об изготовителе, даже если у вас нет пока сделанных им кукол.

Наряду с этими ограничениями программа Access также не разрешит удалить таблицу, участвующую в связи. Нужно сначала удалить связь (используя окно **Работа со связями),** а затем - таблицу.

Пропущенные значения в несвязанных записях

Важно понимать, что есть одна операция, которую вы можете выполнить, не нарушая целостности данных: создание записи о кукле, не ссылающейся ни на какого изготовителя. Это произойдет, если поле **ManufacturerlD** останется пустым (педанты БД называют такое значение неопределенным (null value)). Единственная причина, по которой это поле может остаться пустым, - отсутствие записи об изготовителе в БД или отсутствие приемлемой информации. Может быть, кукла-болванчик была создана не каким-то изготовителем, а высокоразвитой инопланетной цивилизацией и оставлена на этой планете для изучения вами.

Если эта лазейка из пропущенных значений вас нервирует, ее можно устранить. Просто задайте в таблице **Dolls** свойство поля **Обязательное** для ноля **ManufacturerlD.** Эта установка обеспечит каждую куклу-болванчика в вашей таблице **Dolls** законным изготовителем. Подобный прием важен, если связанная информация обязательна. Торгующая компания не сможет поместить заказ или создать счет без ссылки на клиента, сделавшего заказ.

Часто задаваемый вопрос.

Отключение обеспечения целостности данных

Бывают ли такие ситуации, когда не следует требовать целостности на уровне ссылок?

В большинстве случаев целостность данных - это окончательная проверка безопасности БД, и никто не захочет от нее отказаться - особенно, если в БД включена информация для решения критически важных задач вашего бизнеса. Напоминаю, что целостность на уровне ссылок препятствует проникновению противоречивых данных. Она пропускает пустые поля, если нет связанной записи, на которую вы хотите сослаться.

Единственная ситуация, в которой вы можете решить увильнуть от соблюдения правил целостности на уровне ссылок, - использование частичных копий вашей БД. Обычно это происходит на крупных предприятиях, использующих одну и ту же БД на разных площадках.

Рассмотрим очень успешную компанию, торгующую выпечкой на шести площадках. Когда клиент делает заказ на площадке в центре города, вы добавляете запись в таблицу **Orders** и заполняете поле **CustomerlD** (код клиента) (которое ссылается на полную запись в таблице **Customers).** Вот тут-то и возникает проблема. Полной записи о клиенте может не быть в вашей копии БД - вместо этого она может храниться в одной из БД на другой площадке или в главном управлении компании. Несмотря на то, что связь в таблице **Orders** правильная, программа Access считает, что вы допустили ошибку, поскольку она не может найти соответствующую запись о клиенте.

В этом случае вы можете отключить обеспечение целостности данных, чтобы иметь возможность вставить запись. Если вы сделаете это, будьте особенно внимательны при вводе связанного значения (в данном примере **CustomerID),** чтобы избежать ошибок в дальнейшем.

Каскадное удаление

Правила целостности решительно останавливают вас при попытке удалить родительскую

запись (такую как сведения об изготовителе), на которую ссылаются дочерние записи. Но есть и другой вариант - гораздо более радикальный. Вы можете выбрать удаление всех связанных дочерних записей при удалении записи-родителя. Например, это позволит вам удалить изготовителя и ликвидировать всех кукол, произведенных им.

Предупреждение

Каскадные удаления рискованны. Слишком легко удалить больше записей, чем было намечено, а если это сделать - пути назад нет. Еще хуже то, что команда **Отменить** (Undo) не сможет вам помочь отменить это изменение. Поэтому действуйте с осторожностью.

Для включения этого варианта при создании связи вы должны установить флажок **каскадное удаление связанных записей** (Cascade Delete Related Records) (см. рис. 5.4). Обновить связь можно и позже, установив данный флажок.

После установки этого флажка режим можно опробовать, удалив изготовителя, как показано на рис. 5.6.

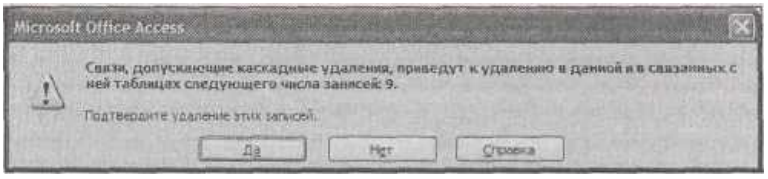


Рис. 5.6. В данном примере отношение Dolls-Manufacturers использует установленный флажок каскадное удаление связанных записей. Когда удаляется изготовитель, программа Access предупреждает о том, что в действительности вы удалите девять записей

Для тех, кто понимает.

Пользуйтесь каскадным удалением с осторожностью

Средство каскадное удаление связанных записей - ядерное оружие БД, поэтому хорошенько подумайте, необходимо ли оно вам. Этот режим позволяет очень легко удалить записи, которые на самом деле нуждались всего лишь в корректировке.

Если вы выбрасываете клиента из БД, нет смысла удалять сведения о его выплатах, которые нужны для вычисления вашей общей прибыли. Гораздо лучше изменить запись о клиенте, пометив ее как неиспользуемую в дальнейшем. Можно добавить в запись о клиенте поле **Active** (действующий) логического типа и задать в нем значение *Нет* для того, чтобы пометить расчеты клиента как не используемые в настоящий момент вместо удаления записи. Вы также должны помнить, что каскадные удаления - всего лишь удобное средство. Они не содержат новых функциональных возможностей. Если не устанавливать флажок **каскадное удаление связанных записей,** вы все равно сможете удалять связанные записи до тех пор, пока не нарушите правильный порядок действий. Если нужно удалить изготовителя, начинайте с удаления всех связанных с ним кукол-болванчиков или измените записи о куклах, указав для них другого изготовителя (или вообще удалив изготовителя) с помощью корректировки значений **ManufacturerID.** После выполнения этого шага вы сможете без проблем удалить запись об изготовителе.

Каскадные обновления

Программа Access также позволяет задать каскадное обновление. Если включить этот режим (установив флажок **каскадное обновление связанных записей** (Cascade Update Related Records) в диалоговом окне **Изменение связей),** Access копирует любое изменение, сделанное вами в связанном поле родительской записи, во все дочерние.

В БД кукол-болванчиков каскадное обновление позволяет изменить ID одного из изготовителей. Как только вы изменили ID, Access автоматически включает новое значение в поле **ManufacturerID** всех связанных записей таблицы **Dolls.** Без каскадного обновления вы не сможете изменить ID изготовителя, если есть связанные с этим значением записи о куклах.

Каскадные обновления безопаснее каскадных удалений, но они редко нужны. Поскольку, если вы следуете правилам хорошего проектирования БД, вы устанавливаете связь, используя столбец ID с типом данных Счетчик *(см. разд. "Счетчик" главы 2).* Программа Access не разрешает

корректировать значение типа Счетчик, и каскадное обновление вам никогда не понадобится. (Счетчик однозначно идентифицирует запись и не связан ни с каким реальным объектом.)

С другой стороны, каскадные обновления очень пригодятся, если вы работаете с таблицей, в которой не предусмотрено применение для связи значений Счетчик. Если таблицы Dolls и Manufacturers связаны именем изготовителя, вам нужны каскадные обновления - они гарантируют согласование значений дочерних записей при изменении имени изготовителя. Каскадные обновления также полезны, если записи связываются с помощью номера социального обеспечения, шифра компонента, серийного номера или других кодов, которые не генерируются автоматически и могут быть объектами корректировок.

5.2.4. *Переходы в отношении*

Отношения не только помогают вылавливать ошибки. Они облегчают просмотр связанных данных. В *разд. "Запросы и связанные таблицы" главы 6* вы узнаете, как создавать процедуры поиска, собирающие вместе информацию из связанных таблиц. Но даже без этих инструментов Access демонстрирует магию связей на листе данных.

Вот как она действует. Если вы просматриваете таблицу-родитель на листе данных, то можете найти все связанные дочерние записи, щелкнув кнопкой мыши квадратик со знаком "плюс", расположенный у левого края строки (рис. 5.7).

Рис. 5.7. Хотите узнать, какие у вас есть куклы от MagicPlastic? Просто щелкните кнопкой мыши квадратик со знаком "плюс" (обведенный)

Этот щелчок раскрывает подтаблицу, в которой отображаются только связанные записи (рис. 5.8). Подтаблицу можно использовать для редактирования записей о куклах прямо в этом окне так, будто вы работаете на листе данных с полной таблицей Dolls. Можно даже добавлять новые записи.

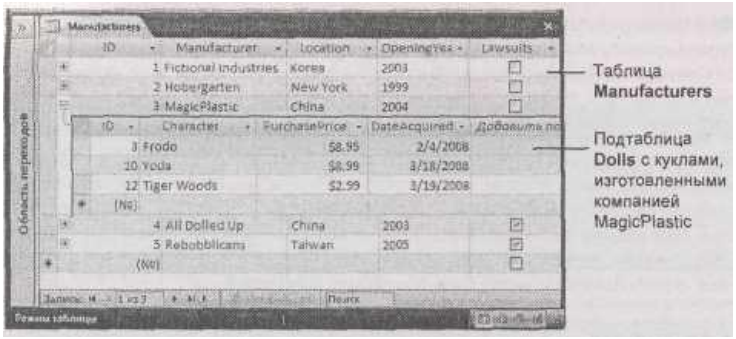


Рис. 5.8. На самом деле подтаблица - это отфильтрованная версия обычной таблицы **Dolls.** В ней отображаются только записи, связанные с выбранным изготовителем. У подтаблицы те же параметры форматирования (шрифт, цвета, порядок столбцов), как и у листа данных связанной таблицы

Примечание

Вы можете открыть одновременно столько подтаблиц, сколько захотите. Единственное ограничение - записи подтаблицы не выводятся при печати листа данных *(см. разд. "Печать листа*

данных" главы 3).

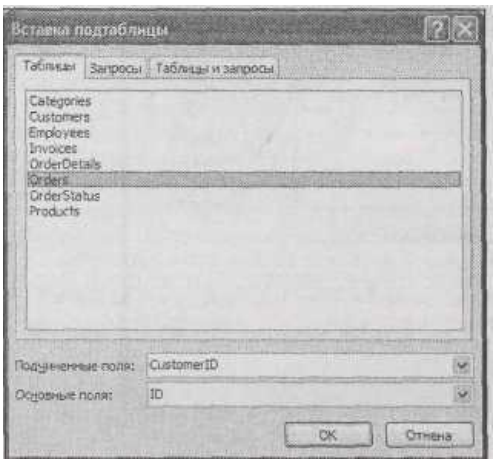


Рис. 5.9. Когда программа Access не знает, какую таблицу использовать как подтаблицу, она разрешает вам выбрать ее из списка всех ваших таблиц. В данном случае только два варианта имеют смысл. Выберите **Orders** для того, чтобы увидеть заказы клиентов, или **Invoices** для того, чтобы вывести счета клиентов. Когда в списке выбрана нужная таблица, Access автоматически заполняет связанными полями области в нижней части окна. Теперь для продолжения можно щелкнуть мышью кнопку ОК

Таблица-родитель может быть связана с несколькими таблицами-потомками. В этом случае программа Access предоставляет возможность выбора таблицы, которую вы хотите использовать, щелкнув кнопкой мыши квадратик со знаком "плюс". Допустим, вы создали таблицу **Customers,** которая связана с таблицей-потомком с заказами клиентов **(Orders)** и с таблицей, содержащей сведения о счетах **(Invoices).** Когда вы щелкаете кнопкой мыши квадратик со знаком "плюс", программа Access не знает, какую таблицу выбрать, поэтому ' она спрашивает вас (рис. 5.9).

Примечание

Вы должны выбирать только один раз подтаблицу, которую хотите использовать. Программа Access запоминает ваш выбор и использует с этого момента одну и ту же подтаблицу. Если вы передумали впоследствии, придется настраивать табличные параметры, как описано в *примечании "Практические занятия для опытных пользователей. Изменение параметров подтаблицы" далее в этом разделе.*

Когда вы создадите более детально проработанные БД, то обнаружите, что ваши таблицы связаны друг с другом цепочкой связей. Одна таблица-родитель может быть связана с таблицей-потомком, которая сама служит родителем для другой таблицы, и т. д. Это сложность не тревожит программу Access - она разрешает вам перемещаться по всей цепочке отношений (рис. 5.10).

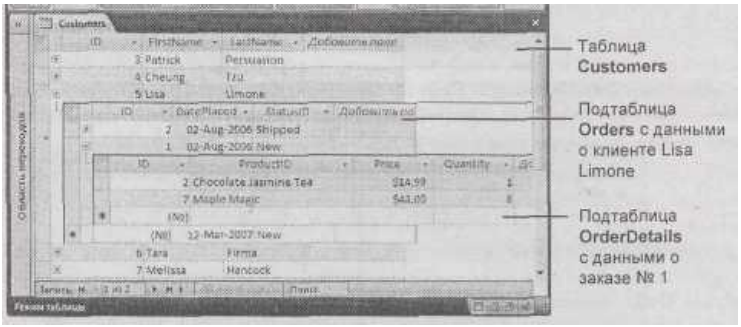


Рис. 5.10. Здесь показаны две действующие связи. Таблица Customers - родитель для таблицы Orders (в которой перечислены все заказы, сделанные клиентом). Таблица Orders - родитель для таблицы OrderDetails (в которой приведены конкретные компоненты каждого заказа). Переходя от уровня к уровню, можно увидеть, что именно купил каждый клиент

Практические занятия для опытных пользователей.

Изменение параметров подтаблицы

Вы можете отрегулировать несколько дополнительных параметров, влияющих на способ отображения в таблице ваших подтаблиц. Для вывода на экран этих параметров переведите таблицу в **Конструктор.** Затем выберите на ленте **Работа с таблицами │ Конструктор → Показать или скрыть → Страница свойств** (конечно, если эта страница не видна в данный момент). Страница свойств отображается в правой части окна.

На ней есть набор разнообразных параметров, которые применяются к таблице в целом. Далее перечислены те из них, которые относятся к подтаблицам.

• **Имя подтаблицы** (Subdatasheet Name). Связанная таблица, применяемая в качестве подтаблицы. Если у вас несколько связанных таблиц, можно выбрать ту, с которой вы хотите работать. Или установить значение параметра *(Auto),* которое заставит программу Access спросить у вас имя подтаблицы в следующий раз, когда вы щелкните кнопкой мыши квадратик со знаком плюс, как показано на рис. 5.9.

• Высота подтаблицы (Subdatasheet Height). Задает высоту в дюймах, отводимую подтаблице для отображения данных. Если все связанные строки не помещаются в отведенное пространство, вам придется пользоваться полосой прокрутки. Стандартное значение этого параметра - 0, позволяющее подтаблице занять столько места, сколько ей нужно.

• Развернутая подтаблица (Subdatasheet Expanded). Позволяет выбрать вывод свернутых подтаблиц до тех пор, пока вы не щелкните кнопкой мыши по квадратику с плюсом (значение по умолчанию), или задать автоматическое раскрытие подтаблицы при открытии основной таблицы (для этого надо выбрать значение *Да*).

5.2.5. *Поиск в связанных таблицах*

Итак, вы увидели, как связи облегчают просмотр и редактирование ваших записей. А как они помогают при первоначальном добавлении записи? Связи обычно основываются на бесполезном значении типа Счетчик. Когда вы создаете новую запись о кукле, то, возможно, не знаете, что компании Bobelle House O'Dolls соответствует код 3408. Программа Access не даст вам ввести идентификационный номер изготовителя, не связанный ни с одной компанией-изготовителем, но не поможет выбрать нужный номер.

К счастью, у Access есть средство, способное помочь вам. В предыдущей главе вы узнали о подстановках *(см. разд. "Создание простою списка подстановок, состоящего из констант" главы 4),* функциональной возможности, снабжающей вас списком доступных значений столбца. При создании подстановки можно представить список констант или предложить список значений из другой таблицы. Вы могли бы создать подстановку для поля **ManufacturerlD** в таблице **Dolls,** использующую список значений ID, взятых из таблицы **Manufacturers.** Такой тип подстановки немного помогает - он предлагает список всех значений, которые можно использовать - но не решает главную проблему. А именно озадаченные пользователи, применяющие вашу БД, понятия не имеют о том, какой идентификационный номер принадлежит какой компании. Вам все-таки нужен способ отображения в списке подстановок имени изготовителя.

У *списков подстановок,* к нашей общей радости, есть такая возможность. Решение - создание подстановки, у которой несколько столбцов. Один столбец содержит информацию (в данном случае имя изготовителя), которая выводится для пользователя вашей БД. В другом столбце находятся данные, которые вы хотите использовать при выборе значения (в данном случае идентификационный номер изготовителя).

Примечание

Программа Access становится немного странной при переходе к подстановкам. Она ждет, что вы добавите список подстановок, а потом связь. (В действительности, когда создается подстановка, использующая таблицу, Access создает связь *автоматически.)* Таким образом, если вы самостоятельно выполняете практические задания, используя предложенные примеры, удалите связь между таблицами **Dolls** и **Manufacturers** (как описано в *разд. "Редактирование связей" ранее в этой главе*), прежде чем двигаться дальше.

Далее приведены действия, необходимые для создания списка подстановок, связывающего две таблицы - **Dolls** и **Manufacturers.**

1. Откройте таблицу-потомок в **Конструкторе.**

В данном примере это таблица **Dolls.**

2. Выберите поле, связывающее ее с таблицей-родителем, в столбце **Тип данных** выберите вариант **Мастер подстановок.**

В предлагаемом примере поле, которое вам нужно, - **ManufacturerID.**

3. Выберите переключатель **Объект "столбец подстановки" будет использовать значения из таблицы или запроса ("I** want the lookup column to look up the values in a table or query") и щелкните мышью кнопку **Далее.**

На экране появится список всех таблиц вашей БД за исключением текущей таблицы.

4. Выберите таблицу-родитель и нажмите кнопку **Далее.**

В данном случае вам нужна таблица **Manufacturers.** После того как вы ее выбрали и перешли к следующему окну мастера, вы увидите на экране список всех полей этой таблицы.

5. Добавьте поле, которое используется для связи, и еще одно, более информативное поле в список **Выбранные поля** (Selected Fields) (рис. 5.11). Для продолжения щелкните мышью кнопку **Далее.**

В данном примере вам нужно добавить поля **ID** и **Manufacturer.**

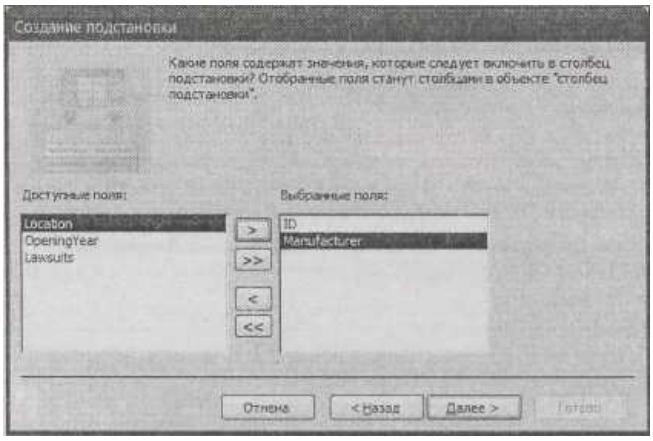


Рис. 5.11. Секрет хорошей подстановки - выбор двух порций информации, первичного ключа (в данном случае поля **ID**) и более информативного значения (названия компании-изготовителя). Данные из поля **ID** вы должны сохранить в записи о кукле, а значение поля **Manufacturer** вы отобразите в списке подстановки для того, чтобы облегчить правильный выбор компании-изготовителя

Подсказка

Иногда может понадобиться несколько полей для описательной информации. Например, можно использовать поля FirstName и LastName из таблицы FamilyRelatives (члены семьи). Но не включайте слишком много информации, иначе список подстановки станет необъемным из-за включений в него всех этих сведений. Это выглядит неестественно.

6. Выберите поле, применяемое для сортировки списка подстановки (рис. 5.12), и щелкните мышью кнопку **Далее.**

В нашем примере список подстановки лучше всего отсортировать в соответствии со значениями поля **Manufacturer.**

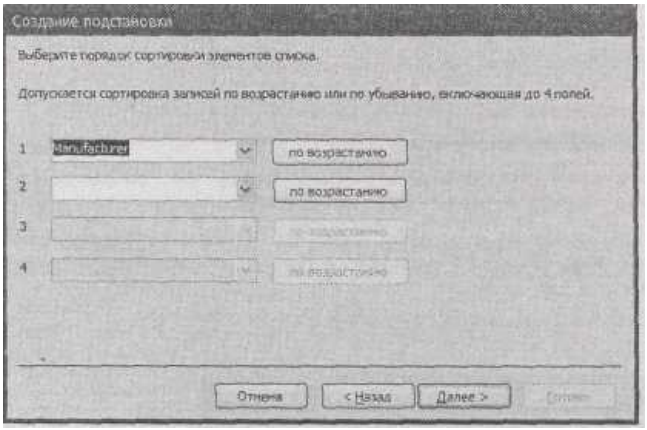


Рис. 5.12. Отсортировать список подстановки очень важно для того, чтобы пользователь мог быстро найти нужное значение

7. В следующем окне мастера показано предварительное представление вашего списка подстановки (рис. 5.13). Убедитесь в том, что установлен флажок **Скрыть ключевой столбец** (Hide key column (recommended)), и затем щелкните мышью кнопку **Далее.** Несмотря на то, что у поля первичного ключа есть значение, связывающее вместе две таблицы, для пользователя, работающего с БД, оно значит не слишком много. Ему гораздо важнее другое, описательное поле.

8. Выберите название столбца подстановки.

Обычно естественней всего сохранить название поля, использующего подстановку (в данном случае **ManufacturerID).**

На последнем этапе вы можете также выбрать режим, называемый **Разрешить несколько значений** (Allow Multiple Values). Если установить этот флажок, в списке отображается флажок рядом с каждым значением, поэтому можно одновременно выбрать несколько элементов списка. (В этом примере можно создать запись о кукле с несколькими изготовителями.) Вы узнаете больше о варианте **Разрешить несколько значений** в *разд. "Многозначные поля "далее в этой главе.*

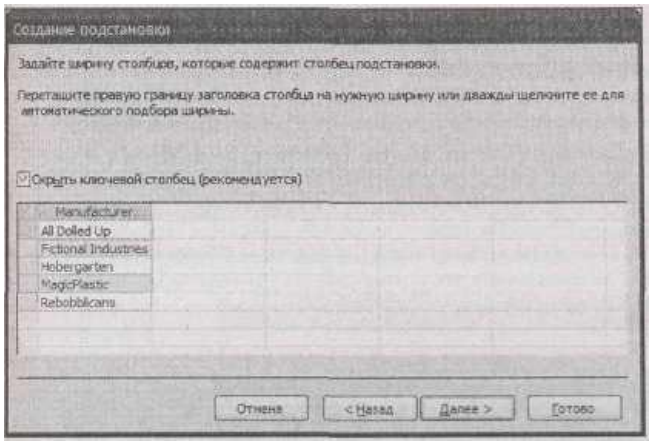


Рис. 5.13. Здесь показан список подстановки, содержащий имя изготовителя (поле **Manufacturer)** и скрывающий его идентификатор (поле **ID**)

9. Щелкните мышью кнопку Готово (Finish).

Теперь программа Access формирует список подстановки для поля и предлагает сохранить таблицу. После этого Access создает связь между двумя таблицами, связанными вашим столбцом подстановки. В данном случае программа устанавливает отношение "родитель-потомок" между таблицами Manufacturers и Dolls, так же как вы делали это самостоятельно *(см. разд. "Определение отношения "ранее в этой главе).*

Примечание

Созданная программой Access связь не обеспечивает ссылочной целостности, поскольку программа не знает, соответствуют ли ваши записи этому жесткому стандарту. В таблице может быть кукла, указывающая на несуществующего изготовителя. Если такая возможность кажется

угрожающе нестрогой, можно отредактировать связь с помощью вкладки Схема **данных** (как описано в разд. *"Редактирование связей" ранее в этой главе).* Начните с добавления на вкладку обеих таблиц **Dolls** и **Manufacturers.** Затем щелкните правой кнопкой мыши линию связи между ними и выберите команду **Изменить связь.** В заключение установите флажок **Обеспечение целостности данных** и щелкните мышью кнопку **ОК**.

Теперь, если отобразить таблицу Dolls в Режиме таблицы, можно использовать список подстановки во время редактирования и вставки записей (рис.5.14).

Часто задаваемый вопрос.
Обновление списка

Я только что добавил запись, но она не появилась в моем списке подстановки. Почему?

Программа Access заполняет ваши списки подстановок, когда таблица открывается впервые. Например, когда открывается таблица Dolls, Access получает готовый к использованию список изготовителей. Но иногда вы должны открыть одновременно как таблицу, использующую список подстановки, так и таблицу, предоставляющую данные для этого списка.

В этой ситуации изменения, вносимые вами в таблицу, предоставляющую данные для списка подстановки, не появятся в таблице, использующей этот список.

Для того чтобы понять механизм действия, откройте одновременно таблицы **Dolls** и **Manufacturers.** (Они открываются на разных вкладках.) В таблицу **Manufacturers** добавьте новую компанию-изготовителя. Затем вернитесь в таблицу **Dolls** и попробуйте воспользоваться списком подстановки в поле **ManufacturerID.** Вы увидите, что новой записи нет в списке подстановки.

К счастью, найти решение легко. Вы можете попросить программу Access обновлять список подстановки в любое время, выбрав **Главная → Записи → Обновить все** (Home → Records → Refresh All). Выполните эту последовательность из таблицы **Dolls** и увидите в списке подстановки обновленный список изготовителей.

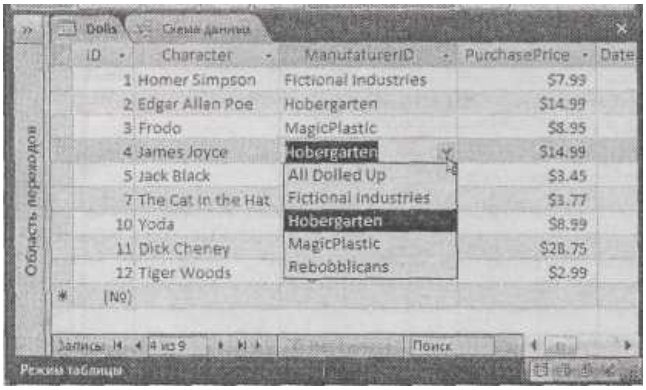


Рис. 5.14. Несмотря на то, что за кадром таблица **Dolls** хранит значение **ID** в поле **ManufacturerlD**, нет способа отобразить его на вашем листе данных. Вместо данного поля вы видите связанное с ним название изготовителя (как на экране, так и на любой сделанной вами распечатке). Более того, если нужно добавить новую запись или изменить изготовителя в имеющейся, вы можете выбрать изготовителя из списка по имени

5.3. *Более экзотические связи*

Как вы узнали из *разд. "Отношение типа „родитель-потомок" "ранее в этой главе,* отношение или связь "один-ко-многим" (известная также под именем родитель-потомок), связывающая единственную запись одной таблицы с одной или несколькими записями другой таблицы, - наиболее распространенный тип отношения. Один изготовитель может быть связан с одной куклой, несколькими или не связан ни с одной куклой вообще.

Наряду со связями "один-ко-многим" существуют еще два несколько иных типа связей: отношение "один-к-одному" и отношение "многие-ко-многим". В следующих разделах вы познакомитесь с обоими типами.

5.3.1. *Отношение "один-к-одному"*

Отношение или связь "один-к-одному" связывает одну запись таблицы с одной или не связывает ни с одной записью другой таблицы. Иногда этот тип отношения применяется для разбиения таблицы с большим количеством полей на две или несколько меньших таблиц.

Таблица **Products** (изделия) может содержать подробную информацию, описывающую изделие и его цену, и дополнительные сведения об особенностях его производства. Эти сведения интересны только сотрудникам инженерно-технических подразделений, поэтому их можно перенести в отдельную таблицу (названную, например, **ProductsEngineering** (технические характеристики изделия). Это та информация, которая не должна интересовать продавцов при оформлении заказов. В другой ситуации можно разбить таблицу на две, просто потому что она слишком велика. (Программа Access не разрешает таблице иметь более 255 полей.)

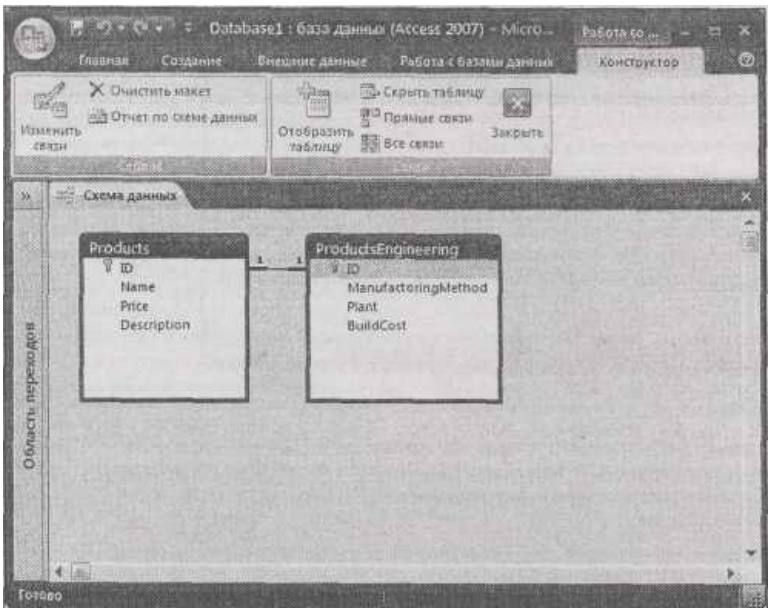


Рис. 5.15. Когда связываются два поля, в которых не допускаются дублирующиеся данные (и флажок **Обеспечение целостности данных** установлен), Access считает, что создается связь "один-к-одному". Программа помещает цифру 1 на концах линии связи для того, чтобы отличать ее от других типов связей. В этом примере столбец **ID** в таблице **Products** и столбец **ID** в таблице **ProductsEngineering** - первичные ключи соответствующих таблиц, поэтому невозможно связать несколько записей таблицы **ProductsEngineering** с одной и той же записью таблицы **Products**

Отношение "один-к-одному" создается так же, как отношение "один-ко-многим" - перетаскиванием с помощью мыши полей на вкладке **Схема данных** (рис. 5.15). Единственная разница состоит в том, что в связанных полях обеих таблиц нужно задать запрет совпадений. В этом случае запись одной таблицы может (как максимум) быть связана с единственной записью в другой таблице.

Примечание

В поле запрещены совпадения, если оно является первичным ключом таблицы *(см. разд. "Первичный ключ" главы 2)* или если у поля есть индекс, препятствующий появлению дублирующейся информации *(см. разд. "Предотвращение дублирования значений с помощью индексов" главы 4)*.

Для тех, кто понимает.

Применяйте связи "один-к-одному" с осторожностью

Отношения "один-к-одному" крайне редко применяются в программе Access. Обычно гораздо удобнее использовать скрытие столбцов *(см. разд. "Скрытие столбцов" главы 3)* и запросы *(см. главу 6),* если вы хотите видеть только отдельные поля таблицы.

Разделение таблицы на две части усложняет проект вашей БД и обычно это делается, только если есть другие причины для разбиения таблицы. Примерами могут служить следующие варианты,

- Две части таблицы необходимо поместить в отдельные БД *(см. разд. "Что такое разделенная*

БД" главы 18) для того, чтобы разные люди могли копировать их на разные компьютеры и редактировать независимо.

• Вы хотите защитить от любопытных глаз уязвимые данные. Один из возможных способов - поместить информацию, которую нужно защитить, в отдельную таблицу и сохранить эту таблицу в другой, более защищенный файл БД.

• У вас есть таблица с огромным объемом данных, таких как поля типа Вложение *(см. разд. "Вложение" главы 2)* с большими документами. В этом случае можно повысить производительность, если разделить таблицу. Вы даже можете решить, что лучше поместить половину таблицы в отдельную БД *(см, разд. "Что такое разделенная БД" главы 18).*

• Некоторые данные вашей таблицы необязательны. Вместо того чтобы включать большое количество незаполненных полей, можно выделить их в отдельную таблицу. Когда не нужно включать эту информацию, вам не придется добавлять запись в связанную таблицу.

Если у вас нет таких ситуаций, вы больше выиграете от создания одной большой таблицы.

5.3.2. *Отношение "многие-ко-многим"*

Отношение или связь *"многие-ко-многим"* связывает одну или несколько записей одной таблицы с одной или несколькими записями в другой таблице. Рассмотрим БД, в которой в отдельных таблицах хранятся данные об авторах и книгах. Авторы бестселлеров не останавливаются на одной книге (поэтому вы должны иметь возможность связать одного автора с несколькими книгами). Однако иногда авторы объединяются в команду под одним заглавием (поэтому вы должны иметь возможность связать одну книгу с несколькими авторами). Аналогичная ситуация возникает, если нужно распределить студентов по курсам, сотрудников по комитетам или ингредиенты по рецептам. Можно даже представить подобную ситуацию и в случае БД с куклами-болванчиками, если несколько изготовителей решат объединиться для изготовления одной куклы-болванчика.

Связи "многие-ко-многим" довольно распространены, и программа Access предоставляет два способа их обработки.

Связующие таблицы

Связующие таблицы - традиционный метод обработки связей "многие-ко-многим", и их используют повсеместно в мире БД (включая и программное обеспечение промышленного уровня, такое как Microsoft SQL Server). Основная идея состоит в том, что вы создаете дополнительную таблицу, у которой единственное назначение - связывание двух таблиц.

Каждая запись в связующей таблице представляет связь, которая соединяет вместе запись каждой таблицы в отношение. В БД с книгами и авторами единственная запись в связующей таблице сопоставляет одного автора с одной книгой. Если один и тот же автор написал три книги, вы должны добавить три записи в связующую таблицу. Если два автора работают над одной книгой, вам потребуется дополнительная запись для связи с каждым новым автором.

Предположим, что в вашей таблице **Authors** хранятся записи, представленные в табл. 5.6.

Таблица 5.6. Данные таблицы Authors

ID	FirstName	LastName
10	Alf	Abet
11	Cody	Pendant
12	Moe	DeLawn

В таблице **Books** содержатся записи, показанные в табл. 5.7.

Таблица 5.7. Данные таблицы Books

ID	Title	Published
402	Fun with Letters	January 1, 2007
403	How to Save Money by Living with Your Parents	February 24, 2008
404	Unleash Your Guilt	May 5, 2007

В табл. 5.8 приведена таблица **Authors_Books,** связывающая обе таблицы.

Таблица 5.8. Данные таблицы Authors_Books

ID	AuthorlD	BookID
1	**10**	402
2	**11**	403
3	**12**	403
4	**11**	404

AuthorsBooks - связующая таблица, определяющая четыре связи. Первая запись указывает на то, что автор № 10 (Alf Abet) написал книгу № 402 (Fun with Letters). Если вы просмотрите остальную часть таблицы, то обнаружите, что Cody Pendant принимал участие в написании двух книг, и два автора работали над одной и той же книгой (How to Save Money by Living with Your Parents).

Подсказка

Имя связующей таблицы часто состоит из имен двух таблиц, которые она связывает, например **Authors Books.**

Суть связующей таблицы заключается в том, что она формирует два отношения "один-ко-многим", определенные в программе Access. Другими словами, связующая таблица - это таблица-потомок, у которой два родителя. У таблицы **Authors** отношение "один-ко-многим" с таблицей **Authors_Books,** в котором таблица **Authors** выступает как родитель. У таблицы **Books** также отношение "один-ко-многим" с таблицей **Authors_Books,** в котором таблица **Books -** родитель. Вы можете определить эти два отношения на вкладке **Схема данных,** убедившись в том, что заданы правила целостности данных (рис. 5.16).

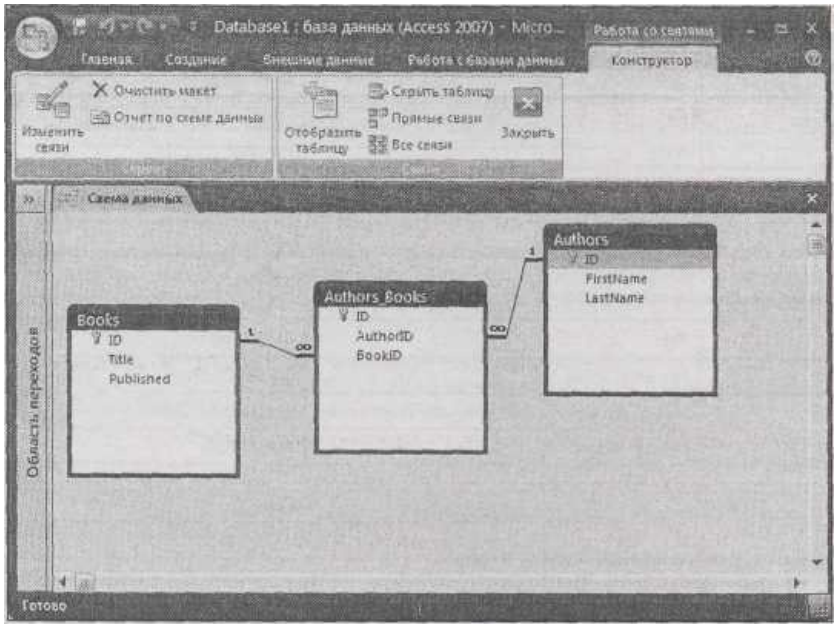


Рис. 5.16. На самом деле отношение "многие-ко-многим" между таблицами **Authors** и **Books -** это два отношения "один-ко-многим", включающие таблицу **Authors_Books.** После определения этих отношений вы не сможете связать автора или книгу, которые не существуют, и удалить автора или книгу, у которых есть запись в таблице **Authors_Books**

Хотя на первый взгляд связующие таблицы производят странное впечатление, большинство фанатов БД быстро привыкают к ним. Как и в случае связей "один-ко-многим", которыми вы пользовались ранее, можно создавать подстановки *(см. разд. "Поиск в связанных таблицах" ранее в этой главе)* для полей **AuthorID** и **BookID** таблицы **Authors_Books.** Но вам придется всегда вставлять вручную запись в таблицу Authors_Books для того, чтобы связать автора с книгой.

Многозначные поля

До появления программы Access 2007 связующие таблицы были единственным средством создания связей "многие-ко-многим". Но для поддержки средств интеграции *(см. главу 21)* сервисов SharePoint в Access 2007 включена новая функциональная возможность - *многозначные поля.*

Как следует из названия, многозначное поле может хранить более одного значения. Эта возможность очень удачно решает проблему связей "многие-ко-многим". Идея состоит в настройке связующего поля в таблице-потомке как многозначного поля. Вернемся к примеру с авторами и книгами. При отсутствии связующей таблицы вам нужно вставить столбец **AuthorID** в таблицу с записями о книгах для обозначения каждого автора, написавшего данную книгу (табл. 5.9).

Таблица 5.9. Данные таблицы Books, в которую добавлен столбец AuthorID, содержащий дублирующие значения

ID	Title	Published	AuthorID
402	Fun with Letters	January 1, 2006	**10**
403	How to Save Money by Living with Your Parents	February 24, 2005	**11**
404	Unleash Your Guilt	May 5, 2006	**11**

Но обычное поле хранит единственное значение. Таким образом, в этой таблице можно указать только одного из двух авторов книги № 403.

Если же разрешить хранение нескольких значений в поле **AuthorID,** можно ввести список авторов, подобный приведенному в табл. 5.10.

Таблица 5.10. Данные таблицы Books, в которую добавлен столбец AuthorlD, хранящий несколько значений

ID	Title	Published	AuthorID
403	How **to** Save Money by Living with Your Parents	February 24, 2005	11, 12

За кадром многозначное поле в действительности использует связующую таблицу. Но программа Access скрывает эту подробность от вас, что существенно облегчает объединение связанных записей.

Для создания поля с несколькими значениями следует использовать подстановку. Как вы уже знаете (см. рис. 5.14), эта функциональная возможность выбирается на последней странице мастера Создание подстановки. С другой стороны, если у вас уже есть подстановка в поле, необходимо внести небольшое изменение. Откройте таблицу в Конструкторе, выберите поле с подстановкой (например, **ManufacturerID)** и затем в области **Свойства поля** щелкните кнопкой мыши вкладку **Подстановка** (Lookup). Найдите свойство **Разрешение нескольких значений** (Allow Multiple Values) и измените его значение с *Нет* на *Да.*

Примечание

Как только вы задали в поле поддержку множественных значений, *вы* не сможете вернуться к варианту поддержки одного значения.

На рис. 5.17 показан многозначный список подстановки в действии.

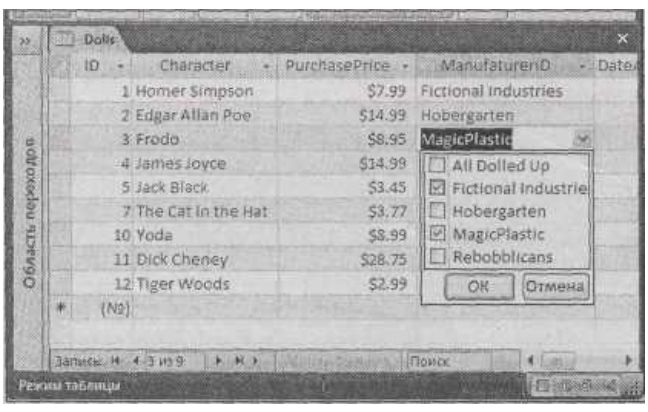


Рис. 5.17. В данном списке подстановки применяются флажки, поскольку он предназначен для многозначного поля. В одной записи можно выбрать несколько значений, установив флажки нескольких элементов списка. Тем самым вы указываете, что одна кукла была создана в результате партнерства двух компаний-изготовителей

Многозначные поля доступны, только если применяется БД нового формата с расширением accdb *(см. примечание "Для тех, кто понимает. Использование Access БД, созданных в более ранних версиях программы" в разд. "Создание новой базы данных" главы 1).* В файле с расширением mdb (БД, созданной программой Access 2003 и еще не преобразованной) вы не сможете их использовать.

Поля с множественными значениями вызывают проблемы при переносе вашей БД на SQL Server (как описано в *главе 20),* поскольку SQL Server не поддерживает их. Следовательно, если есть вероятность совместного использования вашей БД многими пользователями (скажем, в большой компании) и вы можете в какой-то момент перенести ваши данные в БД более мощной программы SQL Server, избегайте полей с множественными значениями.

Примечание

Поля с множественными значениями не создают проблем при переносе вашей БД на Share-Point Server (как описано в *главе 21).*

Часто задаваемый вопрос.

Работа со связями "многие-ко-многим"

Какой подход лучше: связующие таблицы или поля с множественными значениями?

Большинство фанатов БД в ближайшие годы будут приверженцами связующим таблицам. Такие таблицы приняты, укоренились и не скрывают внутреннего функционирования вашей БД. Связующие таблицы особенно удобны, если вы хотите добавить дополнительную информацию о связи между двумя конкретными таблицами. Предположим, что вы создаете таблицу **Students_Classes** для учета учебных курсов, которые все студенты слушают в популярном учебном заведении. В таблицу **Students_Classes** можно включить дополнительные поля, такие как **EnrollmentDate** (дата записи на курс), **Con-firmationLetterSentDate** (дата отправки подтверждающего письма) и **Prerequi-sitesChecked** (необходимые условия приема проверены).

С другой стороны, у связующих таблиц есть недостатки - с ними трудно работать на листе данных. Если в вашей БД применяется связующая таблица **Authors_Books,** для вставки новой книги в вашу систему придется редактировать, по крайней мере, две таблицы. Сначала необходимо вставить запись в таблицу **Books.** Затем следует открыть таблицу **Authors_Books** и вставить в нее новую запись, которая свяжет книгу с автором. (Для облегчения этого процесса можно использовать подстановки в таблице **Authors_Books,** но все равно для этого требуется отдельный шаг.) Если же в таблице **Books** содержится поле **Authors** с множественными значениями, можно добавить книгу и присвоить ей авторов за один шаг, что гораздо удобнее.

Если вы решили остановиться на связующих таблицах и хотите облегчить свою жизнь, программа Access предлагает отличное решение. Можно создать настраиваемую форму, умеющую работать сразу с несколькими таблицами. Можно сконструировать форму, позволяющую человеку, работающему с БД, вставлять запись одновременно и в таблицу **Books,** и в таблицу **Authors_Books.** И главное - ваша форма может выглядеть так, как будто она использует только одну таблицу. Вы узнаете, как применять этот прием в *части IV.*

5.4. *Практическое применение связей*

Каждому проектировщику БД необходимо видеть мир как совокупность таблиц и связей. Смышленые разработчики БД могут быстро оценить информацию и увидеть, как она связана. Благодаря этой способности они могут создать корректную БД в любой области.

В последующих разделах представлены два сценария, которые могут помочь вам приобрести опыт создания более реалистических отношений. Обе БД, используемые в данных сценариях, включены в примеры к данной главе, мы вернемся к ним снова в следующих главах, когда начнем создавать более сложные объекты БД, такие как запросы, отчеты и формы.

5.4.1. *Музыкальная школа*

Cacophone Studios управляет музыкальной школой среднего размера. Школа предлагает определенный набор курсов и имеет штатное расписание преподавателей, способных вести большинство из них. Есть также довольно длинный список бывших и потенциальных клиентов. В прошлом году случилась катастрофа местного масштаба, когда 273 студента были втиснуты на один и тот же курс и им не назначили преподавателя. (Соседний курс из 14 студентов почему-то получил трех преподавателей) Руководители надеются, что программа Access поможет им избежать подобного конфуза в настоящем и будущем.

Подсказка

Хотите поработать с Cacophone Studios? Попытайтесь выделить возможные таблицы и их связи, прежде чем читать дальше.

Определение таблиц

У каждого бизнеса есть особенности, и необходим долгий подробный анализ для создания наилучшей структуры таблиц для Cacophone Studios. Но, даже не имея особенно глубоких знаний, можно выделить несколько наиболее очевидных кандидатов:

- **Teachers** - таблица для хранения списка всех преподавателей из штатного расписания, дополненная контактной информацией;
- **Students** - таблица для хранения всех учеников, прошлых, настоящих и будущих. Вам не нужно разделять эти группы людей в таблице **Students** - вместо этого вы сможете выбрать нынешних студентов из таблицы, найдя связанную информацию (а именно их запись на курс). Таким образом, можно не усложнять таблицу **Students** и хранить в ней только имя и фамилию и контактную информацию;
- **Classes** - таблица для хранения курсов, предлагаемых компанией Cacophone Studios. В эту таблицу следует включить название учебного курса, дату начала и окончания занятий, максимальный номер принятого студента и другую важную информацию.

Примечание

Условия, необходимые для приема студента на курс, хранятся в поле **PreviousClassRequirements** (необходимые предыдущие курсы) с множественными значениями и подстановкой. Это поле содержит идентификационные номера всех прослушанных курсов. (Другими словами, у каждой записи в таблице **Classes** есть ловко реализованная возможность указать на другие курсы в той же самой таблице.)

Конечно, компании Cacophone Studios очень скоро понадобится гораздо больше таблиц. Но для начала перечисленных таблиц достаточно.

Определение связей

Выделить необходимые связи очень легко. Студенты записываются на курсы. Преподаватели ведут курсы. Эта ситуация предполагает две связи: одна между таблицами **Students** и **Classes,** а другая между таблицами **Teachers** и **Classes.**

Но тут есть одна загвоздка. Компания Cacophone Studios конечно же не хочет мешать одному студенту заниматься на нескольких курсах, поэтому между этими двумя таблицами необходима связь "многие-ко-многим". Несмотря на то, что Cacophone Studios планирует иметь одного преподавателя для ведения каждого курса, но они хотят сохранить возможность кооперации преподавателей для проведения занятий на одном курсе. Следовательно, таблицы **Teachers** и **Classes** вовлечены в более сложное отношение "многие-ко-многим". Для поддержки этих двух связей можно создать две связующие таблицы, названные **Students_Classes** и **Teachers_Classes** (соответственно).

На рис. 5.18 показана описанная организация таблиц.

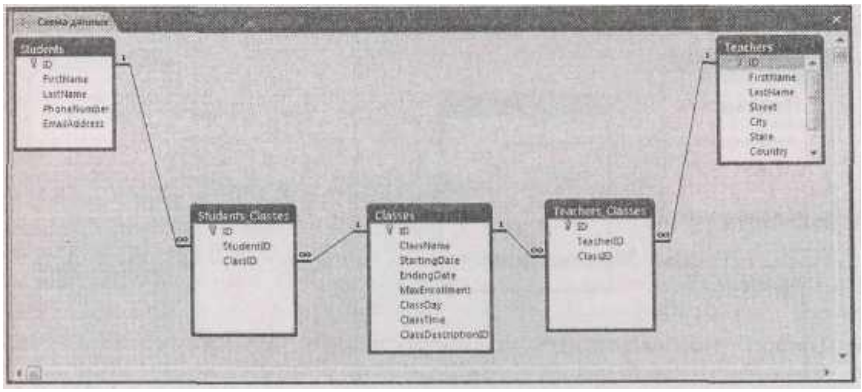


Рис. 5.18. Две связи "многие-ко-многим" формируют основу схемы данных БД музыкальной школы Cacophone Studios

Примечание

Каждая запись в таблице **Students_Classes** представляет сведения о приеме студента на курс. В таблицу можно добавить дополнительные поля, такие как дата записи студента на курс, предложенная вами скидка для принятых студентов, сделавших заказ заранее, и т. д.

Дополнительные подробности

Компания Cacophone Studios начала движение в нужном направлении, но им еще нужно подумать о многом. Прежде всего, каждый раз, когда предлагается курс, необходимо создать отдельную запись в таблице Classes. Это разумный подход, но у него есть потенциальная проблема. Это связано с тем, что когда курс (например, электроакустический в стиле гамелан (Electro-Acoustic Gamelan)) заканчивается, он снова предлагается как новый курс с новыми студентами. Несмотря на то, что это полностью новый курс, у него есть информация, общая с предыдущим курсом, например, описание, стоимость курса, предъявляемые требования и т. д.

Для учета этой особенности нужно создать еще одну таблицу ClassDescriptions (описания курсов). В записи этой таблицы должна содержаться вся описательная информация о курсе. В записи таблицы Classes представлены сведения об одном предусмотренным расписанием конкретном учебном курсе. Таким образом, школа может предлагать без помех один и тот же курс многократно.

Для реализации этой части проекта каждая запись таблицы Classes связывается с единственной записью таблицы ClassDescriptions. Между этими таблицами существует связь "один-ко-многим" (рис. 5.19).

Компания Cacophone Studios также должна думать о неприятной финансовой стороне вещей. Каждый раз, когда студент записывается на курс, нужно взять с него установленную плату за обучение. А при каждом назначении преподавателя для ведения курса ему нужно своевременно выплачивать зарплату.

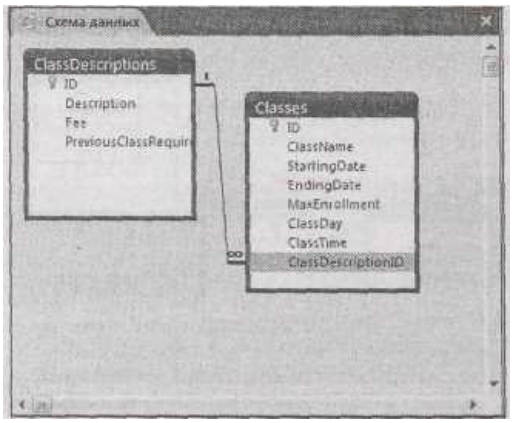


Рис. 5.19. Благодаря таблице ClassDescriptions можно использовать одно и то же описание для нескольких курсов, тем самым избегая избыточности данных

Этими подробностями можно заполнить две таблицы: **TeacherPayments** (плата преподавателям) и **StudentCharges** (плата за обучение студентов). Очевидно, что для этих таблиц надо установить связи, но, возможно, не такие, как вы ожидали. Вы можете решить, что запись таблицы **StudentCharges** следует связать напрямую с записями таблицы **Students.** Такая связь не лишена смысла, поскольку необходимо знать, кто из студентов заплатил деньги, Но так же важно знать, за что заплачены деньги - за какой именно курс платит студент. Другими словами, каждая запись таблицы **StudentCharges** должна быть связана и с таблицей **Students,** и с таблицей **Classes.**

Однако есть более легкий способ. Вы можете сберечь силы, связав таблицу **StudentCharges** непосредственно с таблицей **StudentsClasses.** Если помните, в каждой записи таблицы **Students_Classes** содержатся сведения о студентах и курсе для одного учебного курса. Каждый раз, когда добавляется запись в таблицу **Students_Classes,** необходимо включить соответствующую сумму в таблицу **StudentCharges,** Аналогичное отношение существует между таблицами **Teachers_Classes** и **TeacherPayments.** На рис. 5.20 показано все сооружение (не включена только таблица ClassDescriptions, представленная на рис. 5.19).

Примечание

Напоминаю, что для создания отношения "один-к-одному" следует использовать первичный ключ или индекс, не допускающий совпадений *(см.' разд. "Предотвращение дублирования значений с помощью индексов" главы 4).* В данном примере нужно создать не допускающий совпадений индекс для поля Student_ClasseslD в таблице StudentCharges и поля Teacher_ClasseslD в таблице TeacherPayments. Этот индекс гарантирует, что студенты заплатят только один раз за каждый выбранный ими курс, а преподаватели получат зарплату только один раз за каждый курс, который они провели.

Эта БД очень быстро станет достаточно сложной. А компания Cacophone Studios, возможно, все еще не закончила ее формирование. (Например, очень вероятно, что ей понадобится таблица о платежах студентов.) Как и в большинстве реальных БД, вы будете продолжать добавлять новые таблицы и связи бесконечно.

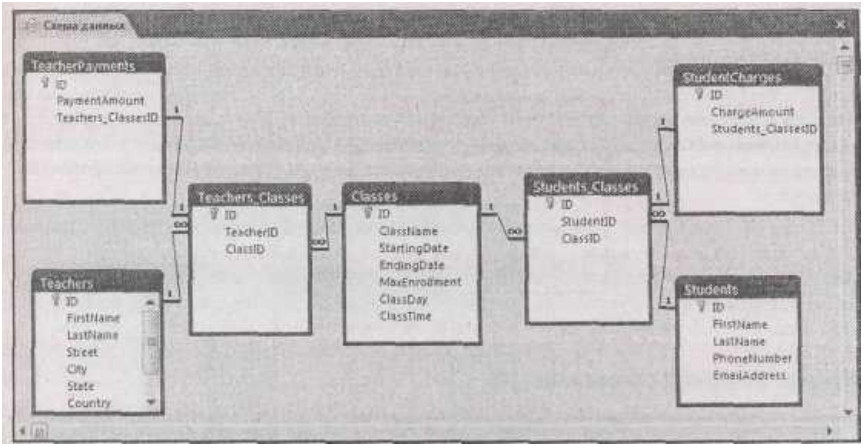


Рис. 5.20. Каждый назначенный курс приводит к появлению платежа в таблице **TeacherPayments** (вверху слева). Каждый прием студента в курс формирует запись об оплате в таблице **StudentCharges** (вверху справа). Несмотря на то, что на первый взгляд эта картинка слегка устрашающая, вы должны уметь прокладывать путь последовательно через все таблицы и связи. Начинать создание БД легче всего с нескольких таблиц, постепенно добавляя новые

Часто задаваемый вопрос.

Печать ваших отношений

Почему последовательность ***Office → Печать*** *(Office button → Print.) становится недоступной, когда я просматриваю вкладку* ***Схема данных?***

После создания связей между вашими таблицами, возможно, вам захочется быстро напечатать эту структуру. Напечатать непосредственно вкладку **Схема данных** нельзя, но ее можно преобразовать в отчет, представляющий собой специализированный объект БД и позволяющий

создать распечатку в любое время. (Вы узнаете, как создавать отчеты, в *части III.)*

Для создания отчета о связях ваших таблиц сначала расположите все выбранные по нашему вкусу таблицы на вкладке **Схема данных.** Затем выберите **Работа со связями | Конструктор → Сервис → Отчет по схеме данных** (Relationship Tools | Design → Tools → Relationship Report). На экран выводится окно предварительного просмотра, которое похоже в той или иной степени на текущее содержимое вкладки **Схема данных.** Для того чтобы вывести его на печатающее устройство, можно выбрать последовательность **Office → Печать.**

После закрытия отчета со связями программа Access предложит сохранить его в вашей БД. Обычно вы с этим не связываетесь, потому что можете легко восстановить его в любое время. Но если у вас сложная БД и вы хотите напечатать несколько разных схем (на каждой из которых представлены разные группы связей), у вас может возникнуть желание сохранить ваш отчет о связях для последующего применения. Вы узнаете больше об отчетах в *главе 10.*

5.4.2. *Магазин шоколадных изделий*

БД о продажах, в которой хранятся товары, клиенты и заказы компании, торгующей чем-либо, - наиболее распространенная разновидность БД. На самом деле этот шаблон появляется так часто, что стоит рассмотреть его в кратком примере. Как будет видно, существует несколько основных правил, применяемых к любому зависящему от продаж (торговому) предприятию, будь то продажа коллекционируемых книг или уцененных фармацевтических изделий.

В данном примере вы встретитесь с принимающей заказы по почте компанией Boutique Fudge, которая удовлетворяет декадентские изыски большой аудитории помешанных на шоколаде клиентов. Ее бесстрашные руководители склонны к постоянным инновациям и хотят как можно лучше управлять постоянно растущим каталогом своих высококачественных изделий. Они также нуждаются в способе учета клиентов и сделанных ими заказов.

Каталог изделий и список клиентов

Даже зная не слишком много о компании Boutique Fudge, вы можете придумать несколько основных нужных им таблиц. Для того чтобы выставить что-то на продажу, вам потребуются следующие таблицы.

■ Таблица **Products,** в которой перечислены соблазнительные шоколадные деликатесы, предлагаемые для продажи. В таблице записаны: название, описание и цена каждого предлагаемого изделия. Имеет смысл включить в нее несколько необязательных подробностей - например, почему бы не учесть текущий запас с помощью двух числовых полей **(UnitsInStock** (количество единиц на складе) и **UnitsOnOrder** (количество заказанных единиц)), а также логическое поле (названное **Discontinued** (больше не продающиеся)) для обозначения изделий, которые больше не поступают в продажу.

Примечание

Во многих БД нельзя удалять старые данные. Компания вроде Boutique Fudge не может просто исключить старые изделия из своих каталогов, поскольку они могут быть связаны со старыми заказами. Кроме того, есть смысл хранить исторические сведения для возможного анализа данных. (Boutique Fudge могла бы применить запрос для поиска самых хорошо продаваемых товаров в 1999 г. и выяснить, есть ли связь между снижением содержания какао и сокращением продаж.) Именно поэтому вам нужно поле **Discontinued.** Когда перечисляются изделия для продажи, все больше не поступающие в продажу изделия можно исключить с помощью средств фильтрации, с которыми вы познакомились в *разд. "Фильтрация" главы 3.*

■ Таблица **ProductCategories** делит изделия на несколько описательных групп. Это позволит клиентам просматривать изделия только в той категории, которая им нужна (будь то напитки (Beverages), конфеты (Candies), шоколад (Chocolate) или сделанная на заказ одежда с надписями, подчеркивающими пристрастие хозяина к шоколаду (Personalized Choco-wear).

■ Таблица **Customers** содержит список шокоголиков, подписавшихся на регулярные заказы. В этой таблице вам нужна вся информация о клиентах, например, имена и фамилии, сведения, необходимые для доставки, информация о счетах.

Примечание

Многие компании разрешают своим клиентам предоставлять несколько адресов доставки и кредитных карт. Если вы согласитесь на такое разнообразие, вам понадобится (не удивляйтесь) больше таблиц. Можно создать таблицу **CustomerCreditCards.** Затем каждую запись из таблицы **Customers** можно связать с одной или несколькими записями в таблице **CustomerCreditCards.** BoutiqueFudge выбрала легкий путь и хранит номер кредитной карты клиента и его адрес непосредственно в таблице **Customers.**

Пока существует одна действующая связь - связь типа "один-ко-многим" между таблицами **ProductCategories** и **Products.** Этот проект показан на рис. 5.21.

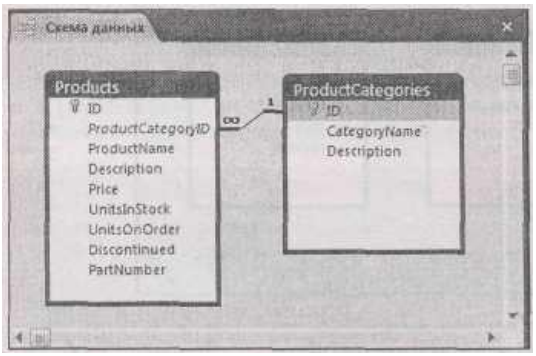


Рис. 5.21. Товар (например, Chocolate Jasmine Tea (шоколадный жасминовый чай)) можно поместить в одну категорию (скажем, напитки), но в отдельную категорию входит много товаров

Заказ товаров

Как бы искусно не была спроектирована ваша БД о продажах, если клиенты не смогут заказывать интересующие их товары, компания Boutique Fudge быстро разорится.

Новички в разработке БД часто допускают ошибку, считая, что сведения о заказах можно хранить в одной таблице. На самом деле вам нужны две.

■ В таблицу **Orders** записывается каждый заказ, сделанный клиентом. Она связана с клиентом, сделавшим заказ, и включает информацию, такую как дата размещения заказа.

■ В таблице **OrderDetails** перечислены отдельные элементы заказа. Каждая запись в таблице **OrderDetails** включает код (ID) заказанного товара, количество единиц товара в заказе и цену, по которой они заказаны.

Поскольку в среднем заказ содержит несколько видов изделий, отдельная запись в таблице **Orders** обычно связана с несколькими записями таблицы **OrderDetails** (как показано на рис. 5.22). Это утверждение может показаться нелепым (т. к. оно означает, что вам требуется создать группу новых записей для всего лишь одного заказа), но процесс не потребует от вас больших усилий. У программы Access есть два средства, которые выручат: подтаблицы (рис. 5.23) и формы *(см. главу 12).*

Обратите внимание на то, что запись **OrderDetails** хранит цену каждого заказанного вида товара. Может показаться, что такая система порождает избыточность данных по отношению к таблице Products. Но цены товаров меняются и компании предлагают скидки. По этим причинам очень важно отслеживать цену товара, когда его заказывают. В противном случае вам придется гадать о том, сколько должен вам каждый клиент.

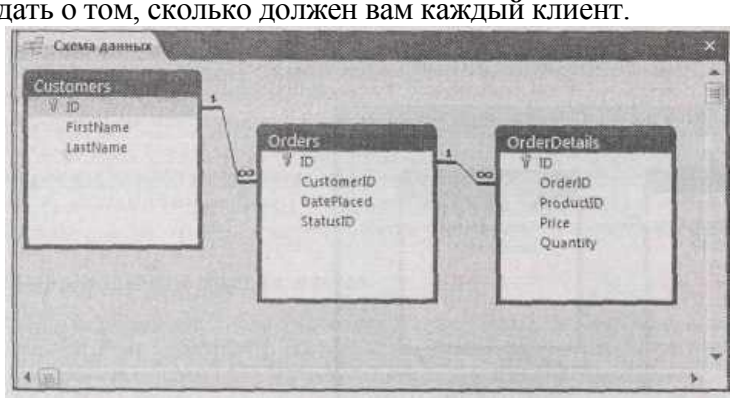


Рис. 5.22. У каждого заказа может быть неограниченное количество заказанных видов товаров.

Такая возможность неизменно радует компанию Boutique Fudge

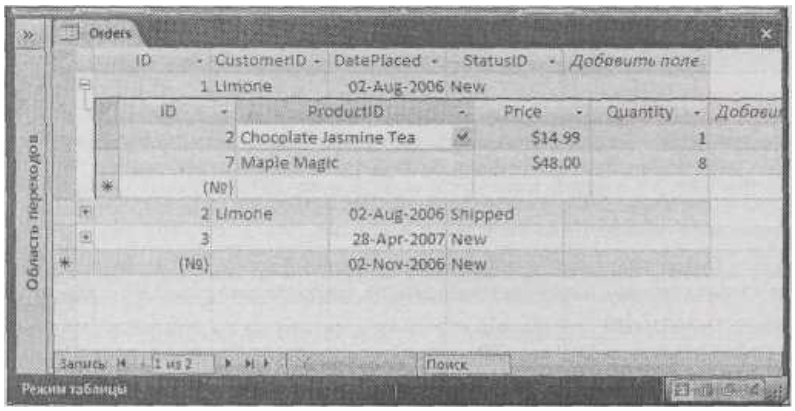


Рис. 5.23. Благодаря наличию подтаблицы можно добавлять в одном месте запись о заказе и связанные с ним виды товаров

Примечание

Профессиональные разработчики БД называют информацию такого сорта сиюминутными или текущими данными (point-in-time data), поскольку они меняются со временем.

Следует отметить, что запись таблицы **Order** не хранит общую стоимость заказа, т. к. общая стоимость - это просто сумма стоимостей всех заказанных товаров. Если хранить общую стоимость, то возникает возможность появления противоречивых данных - иными словами, у вас появится проблема, если сохраняемая общая стоимость не совпадет со стоимостью всех товаров, включенных в заказ.

Вам придется еще как следует поработать, прежде чем Boutique Fudge превратится в компанию, которая управляется БД. Например, возможно, придется создать таблицу **Shipments** (доставки), в которой учтены заказы, отправленные по почте, и таблицу **Payments** (платежи), чтобы быть уверенными в том, что клиенты расплатились полностью. Концептуально в этом нет ничего нового, но чем больше таблиц добавляется, тем сложнее становятся БД. Теперь, зная основы отношений и правила создания таблиц хорошей структуры, вы сумеете сохранять хладнокровие в стрессовой ситуации.

Часть II

Обработка данных с помощью запросов

6. Глава 6. Запросы, выбирающие записи

В типичной БД с тысячами или миллионами записей поиск нужной информации может оказаться трудной и неприятной работой. В *главе 3* вы узнали, как отправляться на охоту, вооружившись средствами, предлагаемыми на листе данных, включая фильтрацию, поиск и сортировку. На первый взгляд эти средства могут показаться отличным способом добывания крупиц глубоко спрятанной информации. Но, к сожалению, средства листа данных временны.

Для того чтобы понять, о чем идет речь, представьте себе, что вы создаете БД в программе Access для компании Boutique Fudge, принимающей по почте заказы на пищевые продукты. С помощью фильтрации, сортировки и скрытия столбцов можно сократить таблицу **Orders,** так чтобы в ней отображались только самые дорогие заказы, сделанные в прошлом месяце. (Эти сведения хороши для выявления транжир или ведения кампании маркетинга самых кодовых товаров (hot marketing).) Затем можно применить другой набор параметров, чтобы определить клиентов, заказывающих более пяти фунтов (2 кг) сливочной помадки каждую субботу. (Вы могли бы использовать эти данные для более детального исследования рынка сбыта или для передачи Министерству здравоохранения.) Но каждый раз, применяя на листе данных новый набор параметров, вы теряете предыдущие результаты. Если нужно перейти от одной выборки к другой, придется скрупулезно переопределять все ваши параметры. Если на создание удачного представления ваших данных было затрачено какое-то время, этот процесс добавит вам много ненужной дополнительной работы.

Решить описанную проблему можно с помощью запросов: заранее подготовленных процедур поиска, которые хранятся в вашей БД. Несмотря на то, что у компании Boutique Fudge только одна таблица **Orders,** у нее могут быть десятки (и больше) запросов с разными параметрами фильтрации и сортировки каждый. Если вы ищете самые дорогостоящие заказы, вам не нужно применять фильтрацию и сортировку вручную - вместо этого вы можете просто запустить запрос **MostExpensive Orders LastMonth** (самые дорогостоящие заказы за последний месяц) и он извлечет только нужную вам информацию. Аналогичным образом, если нужно найти страстных любителей сливочной помадки, можно выполнить запрос **LargeRepeatFudgeOrders** (большие, повторяющиеся заказы сливочной помадки).

Запросы - основной элемент проекта БД. В данной главе вы узнаете все, что нужно для разработки и тонкой настройки базовых запросов.

6.1. *Основные сведения о запросах*

Как следует из названия, запросы позволяют сформировать вопросы о ваших данных, например, какие продукты приносят больше всего денег, где живет большинство клиентов и кто заказал разукрашенную зубную щетку? Программа Access сохраняет каждый запрос в вашей БД, как любой другой ее объект *(см. разд. "Что такое базы данных Access" главы 1).* Сохранив запрос, вы можете выполнить его в любое время, когда захотите взглянуть на реальные данные, отвечающие заданным вами критериям.

Основное достоинство запросов заключается в их способности многократно выполнять тяжелую работу за вас. Кроме того, запросы открывают новые функциональные возможности, которых вы лишены при использовании только листа данных.

■ *Запросы могут объединять связанные таблицы.* Такая возможность невероятно полезна, т. к. позволяет при поиске принимать в расчет связанные данные. В примере с компанией Boutique Fudge благодаря этой способности можно создать запросы, которые находят заказы конкретных продуктов или заказы клиентов, живущих в определенных городах. Оба эти поиска должны использовать связи, т. к. они выходят за пределы таблицы **Orders** и включают данные из других таблиц (например, **Products** (товары) и **Customers** (клиенты)). Как действуют такие запросы, вы узнаете в *разд. "Запросы и связанные таблицы "далее в этой главе.*

■ *Запросы могут выполнять вычисления.* В таблице **Products** БД Boutique Fudge приведены сведения о ценах наряду с данными о количестве товаров на складе. Запрос может перемножить эти данные, а затем вставить столбец, в котором представлена вычисленная стоимость товара, находящегося у вас под рукой. В *главе 7* вы попробуете выполнить такой подсчет.

■ *Запросы могут подсчитывать итоги.* Для анализа больших массивов данных вы можете сгруппировать строки с подобными данными. Можно сгруппировать вместе заказы одного клиента, чтобы узнать его максимальные затраты. Или вы можете сгруппировать заказы по продуктам, чтобы на лету построчно сравнить объем продаж товара ThermoNutcular Fudge с объемом продаж продукта Vanilla Bean Dream. С этим методом вы познакомитесь в *главе 7.*

■ *Запросы могут автоматизировать внесение изменений.* Если нужно найти все заказы, сделанные определенным человеком, и снизить стоимость каждого на 10%, запрос можно применить сразу к группе записей. Это действие требует применения запроса другого типа, запроса на изменение (action query), с которым вы познакомитесь в *главе 8.*

В данной главе рассматривается простейший и самый распространенный тип запроса: *запрос на выборку* (select query), который извлекает подмножество данных из таблицы. После получения этого подмножества вы можете напечатать или отредактировать его с помощью листа данных так же, как таблицу.

6.2. Создание запросов

Программа Access предлагает три способа создания запросов.

■ *Мастер запросов* предоставляет самый легкий способ построения простого запроса. Но этот метод обладает минимальным набором средств управления.

Примечание

Если вы решите использовать Мастер запросов для формирования вашего запроса, возможно, впоследствии вам придется переопределить этот запрос с помощью **Конструктора.**

■ ***Конструктор*** предлагает самый общий метод построения запросов. Он обладает удобным графическим инструментом, который можно применять для улучшения вашего запроса.

■ В ***Режиме SQL*** вы можете увидеть скрытую команду запроса, представляющую собой текстовый фрагмент (состоящий из одной строки или десятка строк), который задает конкретные действия программе Access. Многие профессиональные разработчики творят именно в **Режиме SQL,** и хотя на первый взгляд он кажется мудреным, на самом деле в нем не так трудно разобраться.

6.2.1. Создание запроса в Конструкторе

Лучшая стартовая точка для создания запроса - режим **Конструктора.** Далее перечислены необходимые действия. (Для того чтобы самостоятельно попробовать создать запрос, можно использовать базу данных BoutiqueFudge.accdb, включенную в примеры к данной главе, загружаемые из Интернета.) Окончательный результат - запрос, получающий данные за 2007 г. - показан на рис. 6.6.

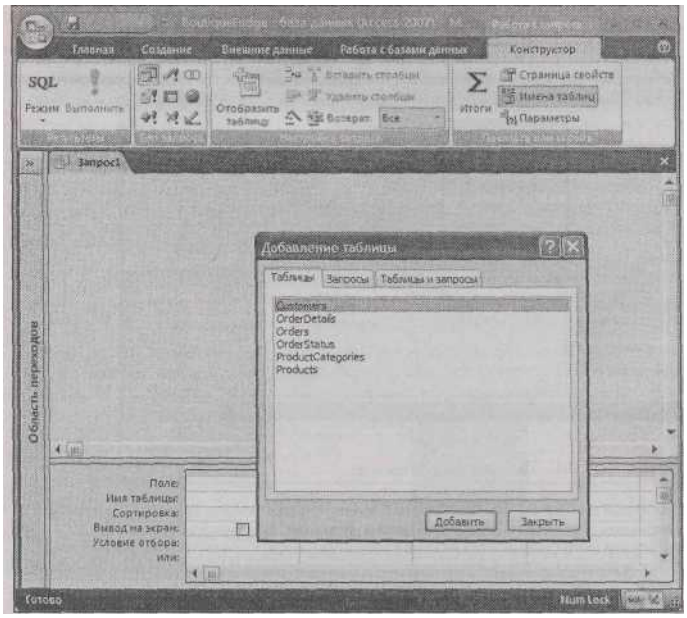


Рис. 6.1. Вы уже видели окно **Добавление таблицы -** с его помощью вы вставляли таблицы в схему данных в *главе 5*

Далее описано, что следует сделать.

1. Выберите **Создание → Другие → Конструктор запросов** (Create → Other → Query Design).

Па экране появится новое окно **Конструктора,** в котором вы сможете создать вага запрос. Но сначала программа Access распахивает диалоговое окно **Добавление таблицы** (Show Table), в котором можно выбрать таблицы для обработки (рис. 6.1).

2. Выберите таблицу, содержащую нужные вам данные, и щелкните мышью кнопку **Добавить** (Add) (или дважды щелкните таблицу кнопкой мыши).

В примере с БД Boutique Fudge вам нужна таблица **Orders.**

Access вставляет прямоугольник, представляющий таблицу в окне **Конструктора.** Вы можете повторить этот шаг и вставить несколько связанных таблиц, но пока остановимся на одной.

3. Щелкните мышью кнопку **Закрыть** (Close).

Диалоговое окно **Добавление таблицы** исчезает, открывая доступ в **Конструктор** для формирования запроса.

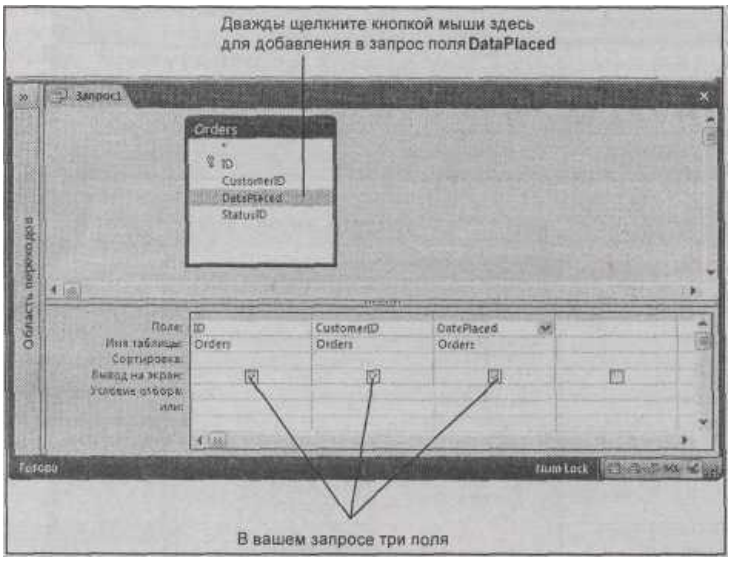


Рис. 6.2. Каждый двойной щелчок мышью поля в прямоугольнике таблицы заставляет программу Access добавлять это поле к списку полей в нижней части окна. Вы задаете условия отбора и сортировку для этого столбца. Если не хотите возвращать мышь в прямоугольник таблицы, можно добавить поле прямо из списка столбца, выбрав его имя из раскрывающегося списка в строке **Поле**

4. Выберите поля, которые хотите включить в ваш запрос.

Для выбора поля в прямоугольнике таблицы щелкните поле дважды кнопкой мыши (рис. 6.2). Не включайте одно и то же поле дважды, иначе столбец будет отображаться два раза. Если вы пользуетесь примером Boutique Fudge, обязательно выберите, по крайней мере, поля **ID, DatePlaced** и **CustomerID.**

Для того чтобы выбрать все поля из таблицы, можно щелкнуть дважды кнопкой мыши звездочку (*). Но в большинстве случаев лучше добавлять каждое поле отдельно. Такой способ не только помогает видеть, каков ваш запрос, но и позволяет выбрать порядок столбцов о запросе и использовать поле для сортировки и фильтрации.

Примечание

Хороший запрос содержит только самые нужные поля. Чем меньше полей в запросе, тем легче сконцентрироваться на важной информации (и легче разместить распечатку на странице).

5. Расположите поля слева направо в том порядке, в каком вы хотите, чтобы они появились на экране результатов запроса.

При выполнении запроса столбцы появляются в том же порядке, в каком они перечислены в списке столбцов в **Конструкторе.** (Обычно это означает, что столбцы выводятся слева направо в

том порядке, в каком вы их добавляете.) Если вы хотите изменить порядок, то нужно переместить столбцы с помощью мыши (как показано на рис. 6.3).

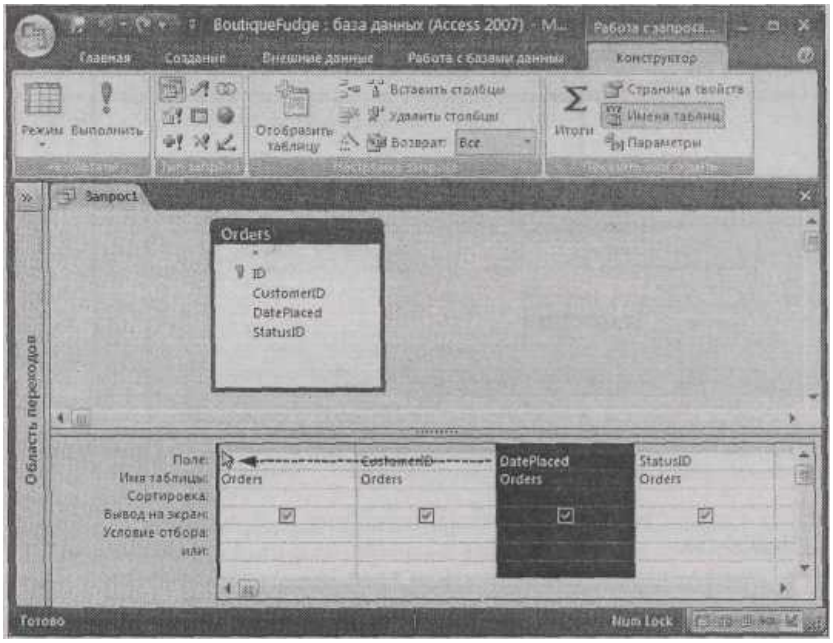


Рис. 6.3. Для реорганизации столбцов перетащите с нажатой кнопкой мыши серую полоску на вершине столбца, который вы хотите перенести на новое место. Такой же способ применяется для упорядочивания столбцов на листе данных. В рассматриваемом примере поле DatePlaced перемещается в крайнее левое положение

6. Если вы хотите скрыть один или несколько столбцов, сбросьте у них флажок **Вывод на экран** (Show).

Как правило, программа Access отображает все столбцы, добавленные в список столбцов. Но в некоторых ситуациях вам нужен столбец при обработке запроса, но отображать его данные нет никакой необходимости. Обычно так бывает, если значения столбца применяются для сортировки или фильтрации.

7. Выберите порядок сортировки.

Если вы не зададите порядок сортировки, то получите записи прямо из БД в том порядке, в каком они там хранятся. Это правило обычно (но не всегда) означает, что самые ранние записи появятся первыми, в верхней части таблицы. Для явной сортировки таблицы выберите поле, которое вы хотите использовать для сортировки результатов, и затем в соответствующем поле **Сортировка** (Sort) задайте вариант упорядочивания. В данном примере таблица сортируется по дате в порядке убывания, поэтому самые последние заказы оказываются первыми в списке (рис. 6.4).

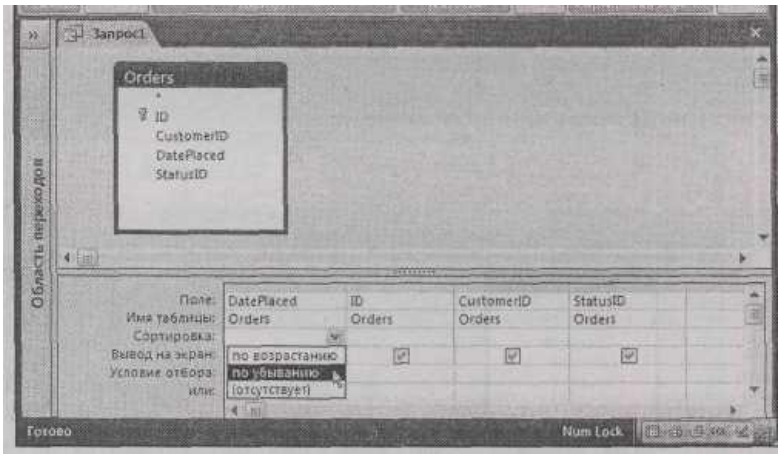


Рис. 6.4. Выберите вариант **по возрастанию,** если хотите отсортировать текстовое поле от А до Я, а числовое поле от меньшего значения к большему или поле даты от самой давней к самой свежей дате. Выберите вариант по убыванию для обратного порядка

Подсказка

Вы можете сортировать по нескольким полям. Единственная хитрость заключается в том, что столбцы должны быть упорядочены таким образом, что первый сортируемый столбец выводится первым (слева) в списке столбцов. Для получения корректных результатов воспользуйтесь методом переупорядочивания столбцов, описанным в пункте 5.

8 Задайте условие фильтрации или отбора.

Фильтрация *(см. разд. "Фильтрация" главы 3)* - это средство, позволяющее акцентировать внимание только на интересующих вас записях и игнорировать все остальные.

Фильтрация или отбор урезает большой пласт данных до нужной вам информации и является сутью множества запросов. (Вы узнаете больше о создании условий фильтрации в следующем разделе.)

Если вы сформировали нужное условие фильтрации, поместите его в поле **Условие отбора** (Criteria) соответствующего поля (рис. 6.5). В данном примере можно поместить это условие в **Условие отбора** поля **DatePlaced** таблицы для того, чтобы выбрать заказы, сделанные в течение первых трех месяцев года: >=#1/1/2007# And <=#3/31/2007#

Вы можете не ограничиваться одним фильтром - на самом деле можно вставить собственное условие отбора в каждое поле. Если вы хотите использовать поле для фильтрации, но не желаете выводить его на экран в окне результата, сбросьте флажок **Вывод** на **экран** для этого ноля.

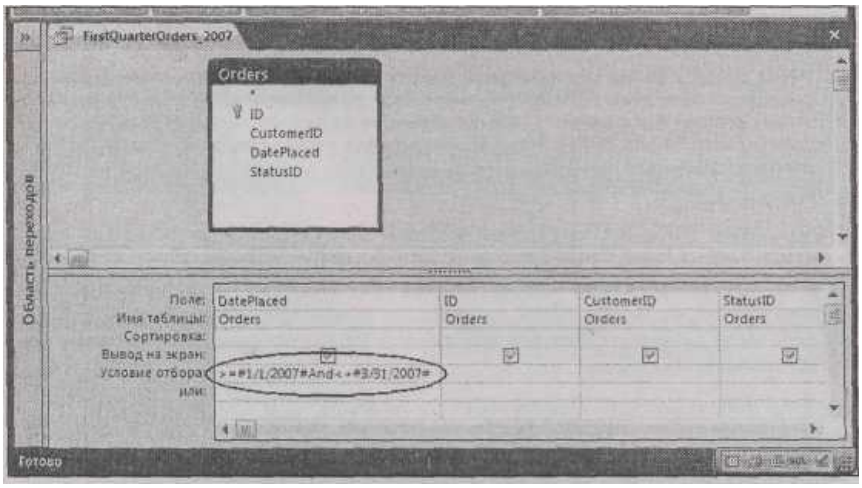


Рис. 6.5. Здесь показан фильтр, определяющий заказы, сделанные в заданном диапазоне дат (с 1 января по 1 марта в 2007 году). Учтите, что когда вы используете реальную жестко закодированную дату как часть условия (например, 1 января 2007 г. в данном примере), ее следует обрамлять символами #

9, Выберите **Работа с запросами | Конструктор → Результаты → Выполнить** (Query Tools | Design → Results → Run).

Теперь создание запроса закончено и он готов к выполнению. Когда вы запустите запрос, то увидите результаты, представленные на листе данных (дополненные подстановками в связанных полях) и напоминающие таблицу в режиме редактирования. (На рис. 6.6 показан результат запроса к таблице **Orders.)**

Вернуться в **Конструктор** можно, щелкнув правой кнопкой мыши заголовок вкладки и выбрав команду **Конструктор** (Design View).

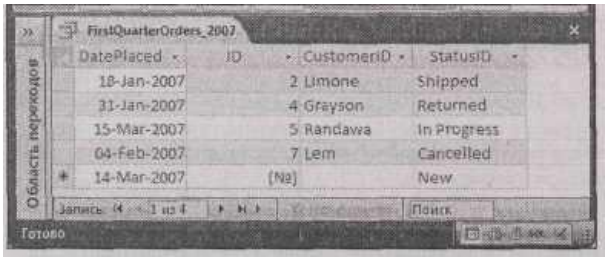


Рис. 6.6. Здесь представлены результаты запроса, отображающего заказы,

сделанные в течение заданного временного периода. Окно листа данных можно использовать для просмотра и вывода на печать результатов или редактирования информации, так же как данных таблицы, отображенной на листе данных

Примечание

Лист данных с вашим запросом приобретает те же параметры форматирования, которые вы задали на листе данных с базовой таблицей. Если вы применили ярко-розовый фон и наклонный шрифт на листе данных с таблицей **Orders,** те же параметры будут у всех запросов, использующих таблицу **Orders.** Но вы можете изменить оформление вашего запроса точно так же, как и в случае таблицы.

10. Сохраните запрос.

Вы можете сохранить ваш запрос в любое время с помощью сочетания клавиш <Ctrl>+<S>. Если вы этого не сделаете, программа Access автоматически сохранит его, когда вы закроете вкладку запроса (или всю вашу БД). Конечно, вы не обязаны сохранять ваш запрос. Иногда запрос создается для конкретной решаемой один раз задачи. Если вы не планируете повторно использовать запрос, нет смысла загромождать вашу БД лишними объектами.

При первом сохранении запроса программа Access запрашивает его имя. Применяйте те же правила именования, которым вы следуете при задании имен таблиц - воздержитесь от использования пробелов и специальных символов и делайте заглавной первую букву каждого слова. Удачное имя запроса описывает представление данных, которое он формирует. Хороший выбор **FirstQuarterOrders_2007** (заказы первого квартала 2007) показан на рис. 6.6.

Примечание

Помните о том, что, сохраняя запрос, вы сохраняете не результаты, а структуру запроса со всеми его параметрами. В этом случае вы можете выполнить запрос в любое время и получить реальные результаты, соответствующие вашим условиям отбора.

После создания запроса вы увидите его в области переходов вашей БД (рис. 6.7). Если использовать стандартный режим отображения **Все таблицы (All** Tables), запрос появится под таблицей, которую он использует. Если запрос использует несколько таблиц, он появится в нескольких группах области переходов.

Вы можете запустить запрос в любой момент, дважды щелкнув его кнопкой мыши. Предположим, что вы создали запрос с именем TopProducts, который захватывает все дорогие продукты из таблицы **Products** (с помощью условия фильтра >50 в поле **Price).** Если нужно просмотреть, отредактировать или напечатать информацию о дорогих товарах, вы выполняете запрос **TopProducts.** Для тонкой настройки параметров запроса щелкните его правой кнопкой мыши в области переходов и затем выберите режим Конструктор.

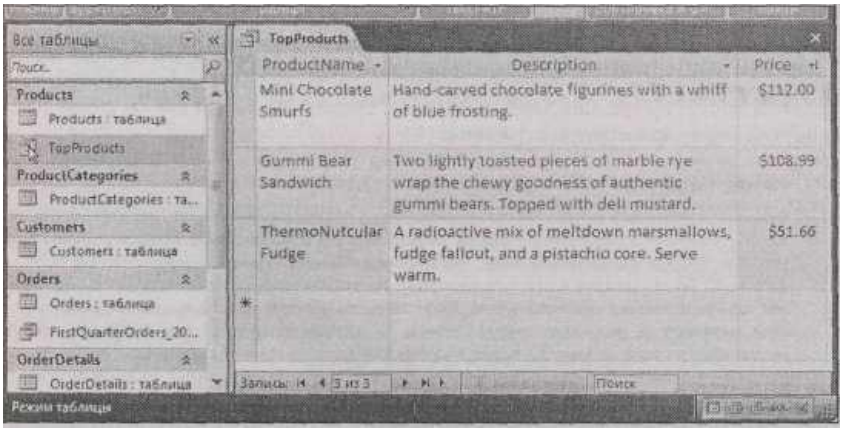


Рис. 6.7. По умолчанию в области переходов ваши запросы выводятся сразу под таблицами, которые они используют. Например, запрос **TopProducts** (показанный здесь) появляется под таблицей **Products**

Программа Access разрешает открывать одновременно таблицу и любые запросы, ее ис-

пользующие. (Все они отображаются на отдельных вкладках.) Но вы не сможете изменить структуру таблицы, пока не закроете все запросы на базе этой таблицы.

Если вы добавляете в таблицу записи, когда открыт запрос, новые записи не будут автоматически появляться в запросе. Вам придется повторно выполнить ваш запрос. Самый быстрый способ - выбрать последовательность **Главная → Записи → Обновить → Обновить** все (Home → Records → Refresh → Refresh All). Можно также закрывать запрос и снова открывать его, поскольку Access каждый раз выполняет запрос, когда вы открываете его в **Режиме таблицы.**

Примечание

Напоминаю, запрос - это представление некоторой части данных вашей таблицы. Когда вы редактируете результаты вашего запроса, программа Access изменяет данные в базовой таблице. С другой стороны, совершенно безопасно переименовывать, изменять и удалять запросы - в конце концов, они существуют для облегчения вашей жизни.

Построение условий отбора

Секрет хорошего запроса - извлечение только нужной вам информации и ничего больше. Для того чтобы сообщить программе Access, какие записи следует взять (а какие нужно игнорировать), вам понадобится условие фильтрации или отбора.

Условие отбора определяет интересующие вас записи. Если нужно найти осе заказы, сделанные клиентом с номером 1032, можно применить следующее условие отбора:

=1032

Для того чтобы заставить это условие действовать, необходимо поместить его в поле Условие отбора (Criteria) под полем **CustomerID.**

В этом поле можно написать просто 1032 вместо =1032, но лучше придерживаться второй формы, поскольку этот шаблон применяется в более сложных условиях фильтрации. Они начинаются с оператора (в данном случае знака равенства), определяющего способ сравнения данных программой Access, за которым следует значение (в данном случае 1032), которое вы хотите применять для сравнения.

Примечание

Если вы используете многозначное поле *(см. разд. "Многозначные поля" главы 5),* программа Access включает в результаты запроса запись, хотя бы одно значение которой соответствует условию отбора. Представьте себе, что таблица **Classes** содержит многозначное поле **InstructorID** (указывающее на то, что несколько преподавателей могут объединиться для ведения одного и того же учебного курса). Если написать условие =1032 в поле **InstructorID,** Access включает в результат любую запись, в которой преподаватель 1032 ведет класс независимо от того, назначены ли для ведения этого класса другие преподаватели.

Для тех, кто понимает.

Не бойтесь подстановок

Как вы знаете, подстановки изменяют способ отображения значений на листе данных. Если добавить подстановку к полю **CustomerID** в таблице **Orders,** вы не увидите зашифрованные числа, такие как 1032. Вместо этого на экран выводятся информативные данные, например фамилия и имя Hancock, John (Хэнкок Джои).

Но при создании условия отбора или фильтрации следует помнить, какие данные на самом деле хранятся в поле. Условие отбора -1032 для поля **CustomerID** действует корректно, а условие =Hancock, John - нет, потому что имя и фамилия хранятся отдельно. (Они содержатся в таблице **Customers,** а не в таблице **Orders.)**

Порой требуется создать условие отбора, использующее связанную информацию. Например, если вы хотите найти записи в таблице **Orders,** используя имя и фамилию клиента вместо его идентификационного номера, поскольку этого номера у вас под рукой нет. Для этого есть две возможности:

- найти нужное значение кода (ID) в таблице **Customers** заранее. После этого вы можете его использовать при построении запроса для таблицы **Orders;**
- применить запрос на объединение для получения имени и фамилии из таблицы **Customers** и вывести их рядом с остальными подробностями заказа. Как воспользоваться этим приемом, вы

узнаете в *разд. "Запросы и связанные таблицы" далее в этой главе.*

Если сопоставляется текст, необходимо значение заключить в кавычки. Иначе программа Access не будет знать, где начинается и заканчивается текстовый фрагмент.

="Harrington Red"

Вместо поиска точного совпадения можно использовать диапазон. Добавьте следующее условие отбора в поле **OrderTotal** для поиска всех заказов, стоящих больше 10 и меньше 50 долларов:

<50 And >10

В этом выражении на самом деле два условия (меньше 50 и больше 10), которые объединены могущественным ключевым словом And *(см. разд. "Комбинирование условий на значения" главы 4).* Как альтернативу можно применять ключевое слово Or, если нужны результаты, которые удовлетворяют одному из заданных вами условий. В *главе 7* вы рассмотрите более мощные инструменты для построения выражений.

Особенно полезны условия для дат. Но не забывайте обрамлять жестко фиксированные даты знаками # *(см. разд. "Проверка допустимости дат" главы 4).* Если поместить следующее условие отбора в поле **DatePlaced,** будут найдены асе заказы, сделанные в 2007 г.:

<#1/1/2008# And >#12/31/2006#

Это выражение отбирает все даты до 1 января 2008 г., но после 31 декабря 2006 г.

Подсказка

Поработав чуть больше, можно создать условие фильтрации, отбирающее заказы за первые три месяца текущего года, независимо от того, какой год на дворе. Это условие требует применения функций, предоставляемых программой Access для дат.

На профессиональном уровне.

Синтаксис фильтра

Если фильтры кажутся хорошо знакомыми, для этого есть основания. У них тот же синтаксис, что и у правил верификации или условий на значение, которые использовались для защиты от некорректных данных *(см, разд. "Правила верификации или условия па значения" главы 4).* Единственное отличие - способ интерпретации условия программой Access. Например, условие на значение <50 And >10 сообщает Access о том, что значение не должно приниматься, если оно не попадает в заданный диапазон (от 10 до 50). Если же такое условие помещается в поле отбора, оно уведомляет программу Access о том, что вас не интересует отображение записей, не попавших в заданный диапазон. Благодаря такому подобию вы можете использовать все условия на значение, представленные в *разд. "Запись условия па значение поля" главы 4.*

В *главе 7* вы узнаете, как усилить условия фильтрации или отбора с помощью функций Access.

Получение заданного количества первых записей

Когда выполняется обычный запрос, вы видите все результаты, удовлетворяющие условиям отбора. Если их больше, чем вы ожидали, можно воспользоваться условиями фильтрации для сокращения списка.

В некоторых случаях фильтры требуют немного больше работы, чем следовало бы. Представьте себе, что вы хотите увидеть 10 самых дорогостоящих продуктов. С помощью условия отбора легко можно получить продукты с ценами, превышающими заданное пороговое значение. Используя сортировку, можно также добиться того, что наиболее дорогие компоненты попадут в верхнюю часть таблицы. Но вы не сможете с легкостью сообщить Access о том, что нужно получить 10 записей и остановиться. В этой ситуации у режима **Конструктора** запросов есть инструмент, способный помочь вам выйти из затруднительного положения. Далее описан его принцип работы.

1. Откройте запрос в **Конструкторе** (или создайте новый запрос и добавьте поля, которые хотите использовать).

В данном примере используется таблица **Products** и добавляются поля **ProductName** и **Price.**

2. Отсортируйте таблицу так, чтобы наиболее интересные для вас записи оказались в верхней части таблицы.

Если вы хотите найти самые дорогостоящие продукты, вставьте сортировку по убыванию в

поле **Price.**

3. В поле **Работа с запросами | Конструктор → Настройка запроса → Возврат** (Query Tools | Design → Query Setup → Return) выберите другой вариант (рис. 6.8).
Стандартный вариант для этого поля - **Все** (All), получение всех соответствующих условию записей. Но можно выбрать 5, 25 или **100** для получения 5, 25 или 100 первых записей соответственно. Можно также задать значение в процентах, например, 25% для получения первой четверти всех отобранных записей.

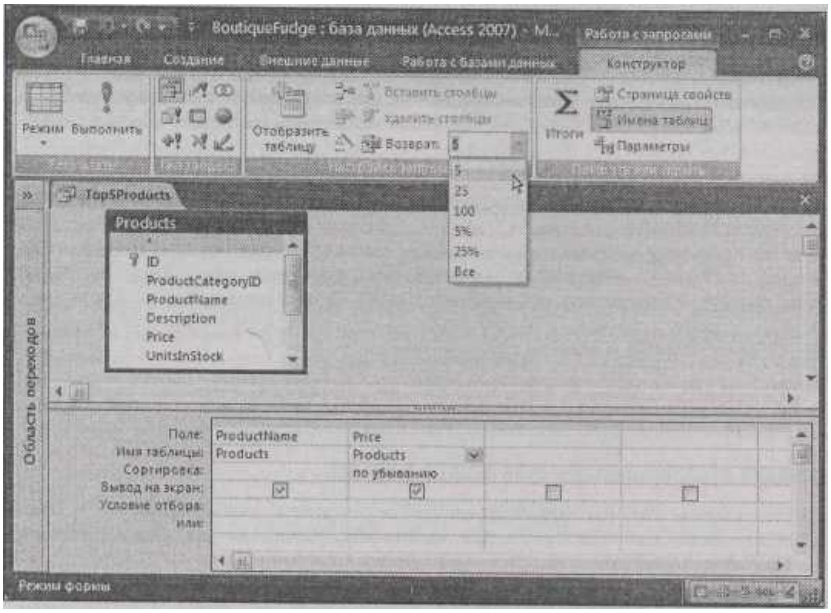


Рис. 6.8. Если нужного вам количества записей нет в списке, просто введите его в поле **Возврат** собственноручно. Ничего не помешает вам отобрать 27 наиболее дорогостоящих продуктов

Примечание

Для того чтобы поле **Работа с запросами | Конструктор → Настройка запроса → Возврат** нормально функционировало, следует выбрать подходящий порядок сортировки. Важность этого условия будет понятна, если узнать немного больше о принципе работы данного инструмента. Если задать программе Access извлечение только пяти записей, она на самом деле выполнит обычный запрос, отберет все записи, удовлетворяющие заданным условиям, и упорядочит их в соответствии с вашим порядком сортировки. Затем программа отбросит все кроме первых пяти записей в списке. Если отсортировать ваш список так, что первыми будут идти самые дорогостоящие продукты (как в данном примере), в результате вы получите пять самых опустошительных для бюджета продуктов.

4. Выполните ваш запрос для отображения результатов (рис. 6.9).

Рис. 6.9. Пять самых дорогостоящих продуктов

Практические занятия для опытных пользователей.

Как индексы ускоряют поиск

В *главе 4* вы познакомились с табличными индексами и научились создавать их. (Индекс - это перечень всех значений в одном поле в отсортированном порядке. Рядом с каждым значением

хранится указатель на полную запись в таблице.) У индексов два назначения. Во-первых, они препятствуют возникновению дублирующихся значений *(см. разд. "Предотвращение дублирования значений с помощью индексов" главы 4).* Во-вторых, они помогают программе Access выполнять поиск с более высокой скоростью. Зачастую Access может искать, пользуясь индексом, быстрее, чем просматривая целую таблицу. Не только потому, что индекс меньше физически (поскольку он содержит значения только одного поля), но и потому, что он отсортирован, и программа может быстрее перейти в нужное место.

Для того чтобы понять разницу, предположим, что вы с помощью программы Access хотите найти запись "Bavarian Tart" в таблице **Products.** Если у вас есть индекс для поля **ProductName,** Access может просматривать раздел для буквы "B", дока не найдет нужное значение, и затем перейти к полному набору деталей. Если же индекса нет, программе придется просмотреть всю таблицу, запись за записью. Таблица не отсортирована, поэтому нельзя сказать, сколько пройдет времени до того, как Access случайно натолкнется на нужную запись.

На первый взгляд индексы кажутся невероятно полезными, и вы готовы попытаться создать их для всех полей вашей таблицы. Но у индексов есть и недостатки. Чем больше индексов создано, тем больше работы приходится выполнять программе Access при добавлении и обновлении записей. Кроме того, каждый индекс занимает какое-то место. В действительности индексы расходуют ресурсы, как бы они не повышали производительность поиска.

Далее перечислены ситуации, в которых следует рассматривать возможность применения индекса для ускорения поиска.

- *У вас БД большого объема.* Если у вас несколько сотен записей, Access почти всегда благодаря принципам работы жесткого диска может просмотреть быстрее всю таблицу, чем применять индекс. Даже если у вас тысячи записей, программа Access часто может загрузить весь набор в оперативную память вашего компьютера, поэтому ей не придется ждать отклика жесткого диска, и все ваши запросы становятся молниеносными.
- *Ваш поиск выполняется медленно.* Нет смысла улучшать запрос, если он и так работает с максимальной скоростью. Большинство приверженцев программы Access могут искать в гигантских БД день за днем, не тратя времени на ожидание.
- *Поле, которое вы хотите индексировать, используется в поиске.* Не индексируйте поле, если вы не применяете его в условии отбора. Если вы часто ищете отдельного конкретного клиента, применяя подстановку его фамилии, добавьте индекс в поле **LastName** (фамилия).
- *Поле, которое вы хотите индексировать, содержит уникальные (или почти уникальные) значения.* Есть смысл добавить индекс к полю **ProductName** в таблице **Products,** поскольку лишь у нескольких продуктов (если такие есть вообще) одинаковое название. С другой стороны, не стоит индексировать поле **City** в таблице **Customers,** поскольку множество клиентов живет в одном и том же городе, в результате индекс в поле **City** будет неэффективен и, возможно, программа Access вообще им не воспользуется.

6.2.2. *Создание простого запроса с помощью Мастера запросов*

Как правило, начинать создание запроса лучше всего в **Конструкторе,** но это не единственная возможность. Можно применить Мастер запроса как отправную точку, а затем переопределить ваш запрос в **Конструкторе.**

В процессе выполнения Мастер запроса задает серию вопросов и затем формирует запрос, отвечающий вашим требованиям. В отличие от множества других мастеров Access и других приложений пакета Office, Мастер запроса довольно слабый. Он хорош как отправная точка для новичков, но не для оперативного исполнителя.

Далее описаны действия, необходимые для запуска и выполнения Мастера запроса.

1. Выберите **Создание → Другие → Мастер запроса** (Create → Other → Query Wizard). Программа Access позволяет выбрать из нескольких разных мастеров (рис. 6.10).
2. Выберите тип запроса. Сейчас лучше всего выбрать мастер **Простой запрос** (Simple Query).

В Мастер запроса включено несколько распространенных типов запросов. За исключением перекрестного запроса у всех остальных нет ничего необычного.

Вы научитесь создавать все эти типы запросов с помощью **Конструктора.**

- Мастер **Создание простых запросов** (Simple Query Wizard) позволяет вам создать

обычный запрос, отображающий подмножество данных таблицы. Этот тип запроса вы создали в предыдущем разделе.

- Мастер **Создание перекрестных запросов** (Crosstab Query Wizard) создает перекрестный запрос, который позволяет анализировать большие объемы данных с помощью разных вычислений. Один такой запрос рассматривается в *разд. "О перекрестных запросах" главы 9.*
- Мастер запросов **Поиск повторяющихся записей** (Find Duplicates Query Wizard) похож на мастер **Создание простых запросов,** за исключением того, что он включает условие отбора, отображающее только те записи, в которых используются совпадающие значения. Если вы забыли создать первичный ключ или создать уникальный индекс в вашей таблице *(см. разд. "Предотвращение дублирования значений с помощью индексов "главы 4),* такой запрос поможет удалить возникший беспорядок.
- Мастер запросов **Поиск записей, не имеющих подчиненных** (Find Unmatched Query Wizard) похож на мастер **Создание простых запросов,** за исключением того, что он содержит условие отбора, извлекающее несвязанные записи из подчиненных таблиц. Его можно применить для поиска заказа, который не связан ни с одним конкретным клиентом. Вы узнаете, как он работает в *разд. "Поиск несвязанных записей" далее в этой главе.*

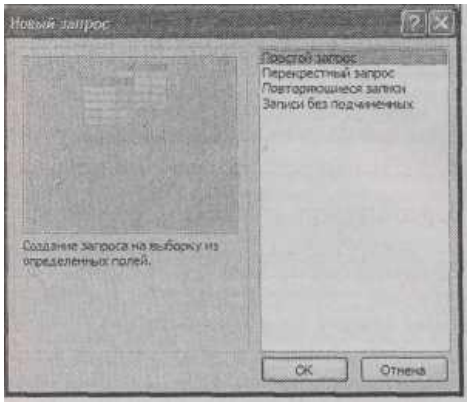


Рис. 6.10. На первом этапе выполнения Мастера запроса вы выбираете из небольшого набора основных типов запросов

3. Щелкните мышью кнопку ОК.

На экране появляется первое окно мастера запросов.

4. В. раскрывающемся списке **Таблицы и запросы** (Tables/Queries) выберите таблицу, содержащую нужные вам данные. Затем добавьте поля, которые вы хотите видеть в окне результатов запроса, как показано на рис. 6.11.

Лучше добавлять поля поочередно. Вставляйте их слева направо в том порядке, в каком они должны появиться на экране результатов.

Можно добавлять поля из нескольких таблиц. Для этого сначала выберите одну таблицу и добавьте поля, которые нужны, затем выберите вторую таблицу и повторите процесс. Такой выбор имеет смысл, только если таблицы связаны. Вы узнаете больше об этом в *разд. "Запросы и связанные таблицы" далее в этой главе.*

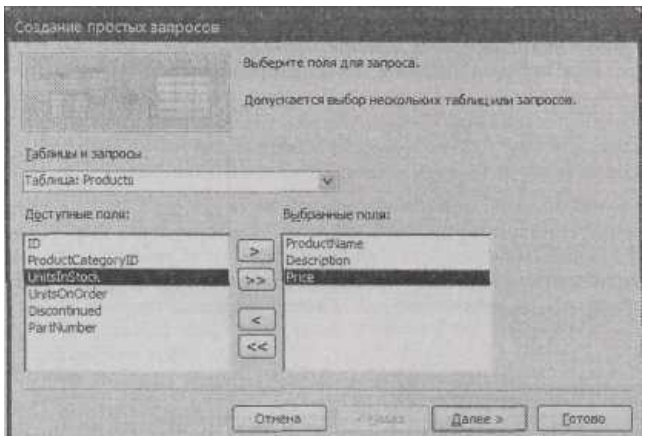


Рис. 6.11. Для добавления поля выберите его в списке **Доступные поля** и щелкните мышью кнопку со стрелкой > (или дважды щелкните его мышью). Можно добавить все поля сразу, если щелкнуть мышью кнопку с двойной стрелкой » и удалить поля, выбрав их в списке **Выбранные поля** и щелкнув мышью кнопку <. В данном примере в запрос включены три поля

5. Щелкните мышью кнопку **Далее** (Next).

Если в вашем запросе есть числовое поле, Мастер запроса предложит создать итоговый запрос, объединяющий строки в группы и вычисляющий итоги или средние значения. Вы узнаете больше об итоговых запросах в *главе 7.* Сейчас, если у вас есть такой выбор, отметьте переключатель **подробный (вывод каждого поля каждой записи)** (Detail), а затем нажмите кнопку **Далее** (Next).

На экране появляется завершающее окно Мастера запроса (рис. 6.12).

6. Введите имя запроса в поле **Задайте имя запроса** (What title do you want for your query?).

7. Если вы хотите подкорректировать запрос, выберите переключатель **Изменить макет запроса** (Modify the query design). Если же вы довольны тем, что получилось, для выполнения запроса выберите переключатель **Открыть запрос для просмотра данных** (Open the query to view information).

Одна из причин, по которой вам может понадобиться переход в режим **Конструктора** - вставка условий отбора или фильтрации *(см. разд. "Фильтрация" главы 3)* для извлечения определенных строк. К сожалению, вы не можете задать условия отбора в Мастере запроса.

8. Щелкните мышью кнопку **Готово** (Finish).

Ваш запрос откроется в **Конструкторе** или **Режиме таблицы** в зависимости от выбора, сделанного вами в пункте 7. Выполнить запрос можно с помощью последовательности **Работа с запросами | Конструктор → Результаты → Выполнить** (Query Tools | Design → Results → Run).

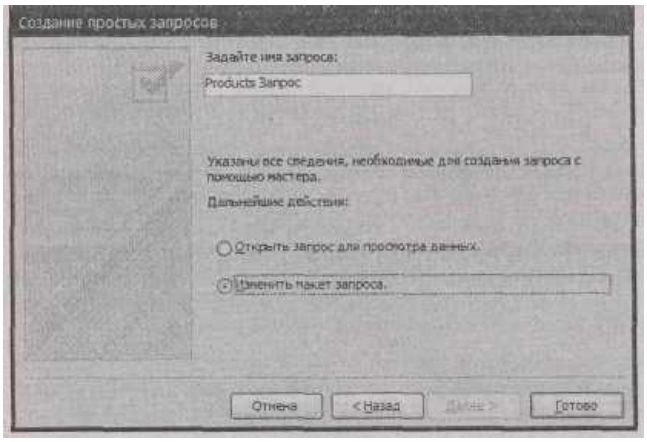


Рис. 6.12. На последнем этапе вы выбираете имя для вашего запроса и немедленный вывод результатов или дальнейшее усовершенствование запроса в Конструкторе

Малоизвестная или недооцененная возможность.

Запросы на базе запросов

В примерах этой главы предполагается, что вы создаете запрос на базе таблицы из вашей БД. Но внимательные читатели могли заметить и другой возможный выбор - а именно возможность создания запроса, отбирающего результаты другого запроса. Если вы создаете запрос в окне **Конструктора,** нужно просто использовать вкладку **Запросы** в диалоговом окне **Добавление таблицы** (вместо вкладки **Таблицы).** Если же запрос создается с помощью мастера, все ваши запросы выводятся вместе с таблицами в раскрывающемся списке **Таблицы и запросы** в первом окне мастера.

Чаще всего запрос строится на другом запросе, если вы хотите повторно использовать плоды вашего напряженного труда и упростить сложные запросы. Например, вы хотите создать запрос к БД Boutique Fudge, который находит клиентов, поместивших заказ в текущем месяце, и извлекает всю информацию об этих клиентах. На основе этого запроса, возможно, вам захочется создать более специализированный итоговый запрос *(см. разд. "Итоговые данные" главы 7),* который объединяет клиентов в группы с учетом города, в котором они живут, и подсчитывает, сколько у

вас недавних покупателей в каждом регионе.

Можно создать один запрос, выполняющий оба этапа. Но разделив логику на две части, вы сможете легко повторно использовать первый запрос (недавние клиенты) для создания множества связанных запросов.

6.2.3. *Режим SQL*

За кадром каждый запрос в действительности - текстовая команда, написанная на экзотическом языке, именуемом SQL (Structured Query Language, язык структурированных запросов). Язык SQL- один из главных компонентов мира БД, он поддерживается всеми основными программными продуктами для управления БД, хотя и с незначительными вариациями и индивидуальными отличительными особенностями.

Примечание

Гуру БД все еще спорят о том, как произносить название языка: "Эс-ку-эль" (что исторически корректно) или "Сиквэл" (именно это название применяется в программном обеспечении корпорации Microsoft SQL Sever). В этой книге мы полагаем, что вы пользуетесь более продвинутым вариантом "Сиквэл".

Когда вы создаете запрос в **Конструкторе** (или с помощью Мастера запроса), программа Access генерирует соответствующую команду SQL. Когда вы сохраняете запрос, Access просто сохраняет в вашей БД текст этой команды. Это все, что нужно программе для выполнения запроса в дальнейшем.

Чаще всего вы не будете тратить много времени на обдумывание SQL, прячущегося за вашими запросами. Но иногда нужно посмотреть на него повнимательнее. Далее перечислены возможные причины.

■ Вы хотите выполнить действие, которое поддерживается языком SQL, но не доступно в **Конструкторе** запросов. Конечно, для редактирования команды нужно знать о языке больше, чем самая малость. Далее в этой главе вы узнаете, как с помощью **Режима SQL** создать запрос на объединение, содержащий в окне результатов данные двух похожих таблиц.

■ Вы хотите выучить язык SQL. Это неплохая идея, если вы хотите делать карьеру администратора БД, но это лишнее, если вы привязаны к программе Access.

■ Вы собираетесь перенести команду в другую программу управления БД. Вы можете находиться в состоянии переноса БД из Access в более мощную БД Oracle. Это работа, требующая усилий, и можно столкнуться с тем, что несмотря на возможность переноса данных в другое хранилище, перемещение других объектов, таких как запросы, невозможно. Вам придется познакомиться поближе с лежащим в основе запросов языком SQL, который можно использовать для реконструкции запроса в новой БД.

■ Вы просто любознательны без всякой задней мысли. Изучение команд SQL для ваших запросов снимает налет таинственности с принципов работы программы Access.

■ Вы - суперспециалист по написанию кода на языке SQL, и **Конструктор** запросов только тормозит вашу работу.

Для просмотра команды SQL для вашего запроса щелкните заголовок вкладки правой кнопкой мыши и выберите команду **Режим SQL** (SQL View). На рис. 6.13 показана команда, которую вы увидите.

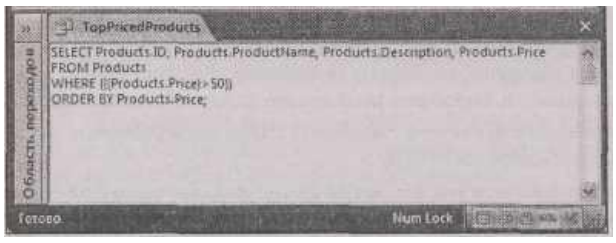


Рис. 6.13. На экране команда SQL для запроса **TopProducts,** который находит товары, стоящие больше 50 долларов. Если вас испугал этот режим, в любой момент можно вернуться в другой, щелкнув правой кнопкой мыши заголовок вкладки и выбрав **Конструктор** или **Режим таблицы**

Анализ запроса

Несмотря на то, что язык SQL на первый взгляд кажется сложным, все запросы готовятся из одних и тех же ингредиентов. Рассмотрим запрос для поиска дорогостоящих заказов, которые приведены далее (каждая строка пронумерована для облегчения ссылок), сводятся по сути к одним и тем же ингредиентам:

```
1.SELECT Products.ID, Products.ProductName, Products.Price
2.FROM Products
3.WHERE (((Products.Price)>50))
4.ORDER BY Products.Price;
```

Проанализируем первые две строки.

■ *Строка 1* начинается со слова SELECT, означающего, что перед вами запрос, который выбирает записи (как и все запросы, с которыми вы имели дело в этой главе).

За словом SELECT следует разделенный запятыми список полей, которые вы хотите видеть. Каждое поле записано в длинном формате *ИмяТаблицы. ИмяПоля,* на случай если вы решите создать запрос, использующий несколько таблиц.

■ *Строка 2* начинается со слова FROM, указывающего на таблицу (таблицы), которую вы исследуете. В данном случае нужные вам записи есть в таблице **Products. В** этих двух строках представлен законченный действующий запрос. Но часто в вашей команде будут дополнительные строки, задающие параметры фильтрации и сортировки.

■ *Строка 3* начинается со слова WHERE, указывающего на начало ваших условий отбора. В данном случае есть только одно условие - цена продукта должна быть больше 50 долларов. Если вы задали несколько условий отбора в разных полях, здесь будут представлены все они, объединенные с помощью оператора AND.

Примечание

Программа Access несколько странным образом применяет скобки в условиях отбора. Предложение языка SQL WHERE (((Products . Price) >50)) можно упростить до следующего WHERE Products. Price>50. Access использует скобки, поскольку они облегчают анализ сложных запросов с множественными условиями отбора.

■ *Строка 4* начинается со слов ORDER BY, которые определяют порядок сортировки. В данном случае записи отсортированы по возрастанию значений в поле **Price** (цена). Если задана сортировка по убыванию, вы увидите сокращение DESC после имени поля. Если сортируется несколько полей, вы увидите разделенный запятыми список полей.

Команда заканчивается завершающей точкой с запятой (;). Программе Access эта деталь не нужна, но таковы соглашения в мире языка SQL.

Резюме приведенного урока состоит в том, что любой запрос, который вы создаете, формируется из нескольких общих ингредиентов, представленных разделами SELECT, FROM, WHERE и ORDER BY.

Программа Access следит за синхронизацией разных режимов представления запроса. Если вы вносите изменение в текст SQL, а затем возвращаетесь в режим **Конструктора,** то увидите только что откорректированную версию запроса (пока вы не допустили ошибку, при возникновении которой Access выводит на экран сообщение об ошибке).

Для проверки этого свойства можно изменить текст SQL так, чтобы выбирался дополнительный столбец и сортировка выполнялась по двум полям таким образом, чтобы продукты с одинаковой ценой выводились в алфавитном порядке (новые текстовые фрагменты выделены жирным шрифтом):

```
SELECT Products.ID, Products.ProductName, Products.Price, Products.Description
FROM Products
WHERE (((Products.Price)>100))
ORDER BY Products.Price, Products.ProductName;
```

Щелкните правой кнопкой мыши заголовок вкладки, затем выберите **Конструктор** для того, чтобы увидеть, как внесенные изменения отражены в режиме **Конструктора.**

Создание запроса на объединение

Конструктор не распознает некоторые редкие методы языка SQL. Их можно применить,

только откорректировав команду SQL в **Режиме SQL,** и после внесения этих изменений вы больше не сможете просмотреть ваш запрос в **Конструкторе** (пока позже не удалите неподдерживаемое изменение).

Запрос на объединение (union query) - один из примеров запросов, временами очень полезных, но не поддерживаемых в **Конструкторе** запросов. Запрос на объединение объединяет результаты из нескольких таблиц и затем представляет их на общем листе данных.

По сути, запрос на объединение составляется из двух (или нескольких) отдельных запросов на выборку. Тонкость заключается в том, что структура результатов всех запросов на выборку должна быть одинаковой. Таким образом, следует извлечь аналогичные столбцы из каждой таблицы в одном **и** том же порядке. Если вы выполнили все перечисленные требования, остается только вставить слово UNION между двумя запросами.

Далее приведен запрос на объединение, который представляет список имен и фамилий, полученный из двух таблиц - **Customers** и **Employees:**

```
SELECT Customers.FirstName,   Customers.LastName
FROM Customers
 UNION
SELECT Employees . FirstName,   Employees . LastName
FROM Employees
```

Этот запрос функционирует, несмотря на то, у таблиц **Customers** и **Employees** разная структура. Но гораздо важнее то, что структура результатов запросов к обеим таблицам, в данном случае поля **FirstName** и **LastName,** совпадает.

Примечание

Создать запрос на объединение можно, даже если имена столбцов отличаются - если в таблице **Employees** содержатся столбцы с именами **F_Name** и **L_Name,** запрос все равно будет выполняться. Программа Access просто использует имена столбцов из первого запроса при выводе результатов на лист данных.

В данном примере, когда вы просматриваете результаты запроса, на экран выводится список имен и фамилий клиентов, за которым следует список имен и фамилий сотрудников, хотя вы не сможете с уверенностью определить, где заканчивается одна таблица и начинается другая. Вы также не сможете редактировать данные - запросы на объединение предназначены исключительно для просмотра сведений, а не для их изменения. Программа Access не позволит вам редактировать запросы на объединение в **Конструкторе** запросов. Если вы щелкнете правой кнопкой мыши заголовок вкладки и выберете **Конструктор,** вместо конструктора вы попадете в **Режим SQL.**

Программа помещает запросы па объединение в группу Несвязанные объекты (Unrelated Objects) в области переходов и применяет для их обозначения пиктограмму, отличающуюся от пиктограммы обычного запроса (рис. 6.14).

Примечание

Если в результатах запроса на объединение выявляются совпадения, на экран выводится одна копия. Это поведение можно изменить, если заменить слово UNION словосочетанием UNION ALL. В предыдущем примере этот шаг вызовет повторное отображение в объединенных результатах человека, являющегося и клиентом, и сотрудником.

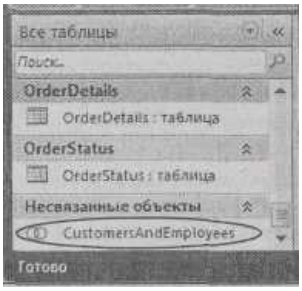


Рис. 6.14. Запросы на объединение появляются в области переходов с другой пиктограммой. Две

пересекающиеся окружности обозначают несколько наборов результатов, отображаемых совместно

Запросы на объединение - это хороший способ соединения двух аналогичных таблиц, которые были разделены из соображений производительности, безопасности или способа распространения. *(См. в разд. "Подготовка вашей базы данных" главы 18* различные причины деления одного набора данных на несколько разных таблиц.) Эти запросы неудобны для обработки отношений "родитель - потомок". Для этой задачи вам нужны запросы на выборку с объединением таблиц (join queries), описанные в следующем разделе.

Для тех, кто понимает.

Подумайте дважды, прежде чем изменять структуру таблиц

Программа Access проявляет удивительную смекалку при отслеживании связей запросов с конкретными таблицами. Это особенность становится актуальной, когда вы распахиваете таблицу в **Конструкторе** для изменения ее структуры.

Предположим, что вы переименовываете таблицу **Orders** (заказы) в таблицу **Sales** (продажи) и поле **DatePlaced** (дата размещения) в поле **OrderDate** (дата заказа). В следующий раз, когда вы запустите запрос **FirstQuarterOrders_2007** (см. рис. 6.6), то с удивлением обнаружите, что он все еще действует. Программа Access знает о том, что запрос **FirstQuarterOrders_2007** зависит от таблицы **Orders.** Когда вы изменяете имена в таблице, программа соответствующим образом корректирует запрос.

Access содержит отличное средство, способное проверить любой выбранный вами объект БД и сообщить обо всех других объектах, которые от него зависят. Это средство можно применить для определения запросов, форм и отчетов, использующих таблицу **Orders,** прежде чем изменять ее. Для применения этого средства выполните следующие действия:

1. Выберите **Работа с базами данных → Показать или скрыть → Зависимости объектов** (Database Tools → Show/Hide → Object Dependencies). В правой части окна программы Access появляется область **Зависимости объектов.** (Для того чтобы скрыть ее, выберите ту же последовательность еще раз.)
2. В области переходов выберите объект БД, который вы хотите исследовать.
3. В области **Зависимости объектов** выберите переключатель **Объекты, зависящие от данного** (Objects that depend on me), чтобы увидеть объекты, использующие данный, или переключатель **Объекты, от которых зависит данный** (Objects that I depend on), чтобы увидеть все объекты, которые использует данный объект.
4. В верхней части области **Зависимости объектов** щелкните кнопкой мыши ссылку **Обновить** (Refresh). В области **Зависимости объектов** выводятся все соответствующие объекты, разделенные на категории в зависимости от их типа (рис. 6.15).

Программа Access не может отследить все зависимости, например, если вам нужно проникнуть в Режим **SQL** для формирования запроса, который нельзя создать в режиме **Конструктора.** Если создается запрос на объединение (как в предыдущем примере), у Access не хватает сообразительности для того, чтобы выяснить, от каких таблиц зависит ваш запрос. Если вы изменили структуру этих таблиц, то при следующем выполнении запроса получите сообщение об ошибке, говорящее о том, что программа не может найти нужное поле или таблицу. (Для исправления ошибки необходимо снова открыть запрос в **Режиме SQL** и заменить имена полей или таблиц их новыми значениями.)

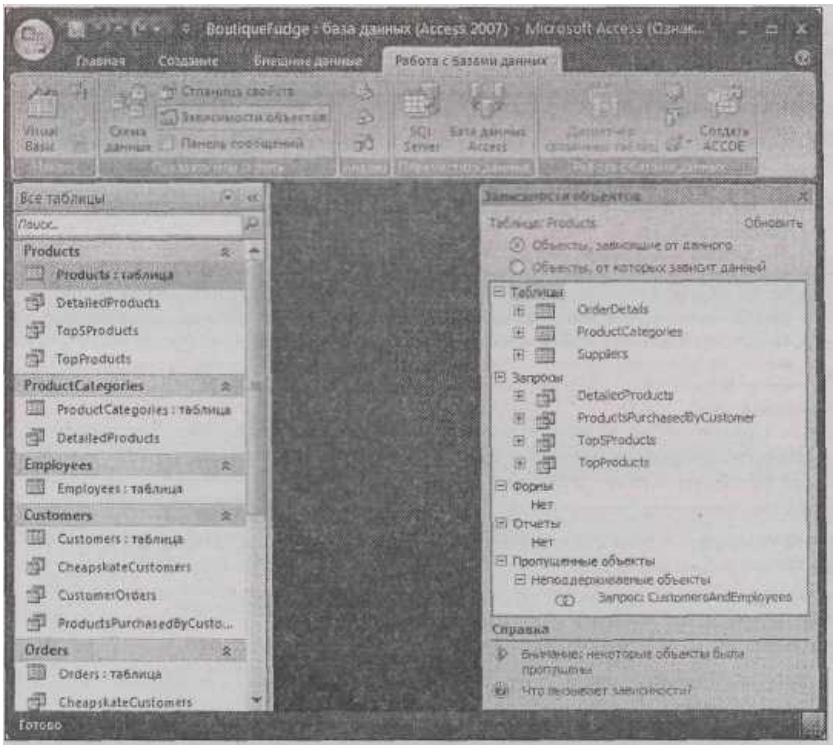


Рис. 6.15. На рисунке в области **Зависимости объектов** анализируется таблица **Products.** Отображены три таблицы, связанные с **Products,** и четыре запроса, использующие таблицу **Products.** В любой объект можно углубиться, щелкнув кнопкой мыши квадратик со знаком "плюс" (+), расположенный рядом с именем объекта. (Щелкните кнопкой мыши + рядом с именем **TopProducts,** чтобы выяснить, используют ли другие объекты БД данный запрос.) В конце списка находится раздел **Пропущенные объекты.** В нем отображен запрос на объединение **CustomersAndEmployees,** и это свидетельствует о том, что у программы Access нет данных о его зависимостях

6.2.4. *Запросы и связанные таблицы*

В *главе 5* вы узнали, как делить данные на базовые фрагменты и сохранять их в отдельных хорошо организованных таблицах. У такого проекта есть лишь одна проблема - гораздо труднее представить общую картину, если связанные данные хранятся в разных местах. К счастью, Access обладает чудесным средством - вы можете снова соединить таблицы при выводе на экран с помощью операции объединения (join).

Объединение - операция запроса, извлекающая столбцы из двух таблиц и соединяющая их на листе результатов. Объединение применяется для усиления подчиненных таблиц данными из таблицы-родителя.

Далее приведено несколько примеров.

В БД кукол-болванчиков можно отобразить список кукол (извлеченный из таблицы **Dolls)** совместно с данными об изготовителе каждой куклы-болванчика (из таблицы-родителя **Manufacturers).**

В БД школы Cacophone music можно получить список учебных классов, снабженный информацией о преподавателях.

Из БД Boutique Fudge можно извлечь список заказов, дополнив его сведениями о клиенте, сделавшем заказ.

Примечание

Вы уже научились создавать таблицы подстановок для отображений части информации из связанной таблицы. Подстановка может вывести название категории изделия из поля **ProductID** вместо кода изделия. Но запрос с использованием операции объединения гораздо мощнее. Он может выбрать массу сведений из связанной таблицы - гораздо больше, чем может вместить одно поле.

На рис. 6.16 показано объединение таблиц.

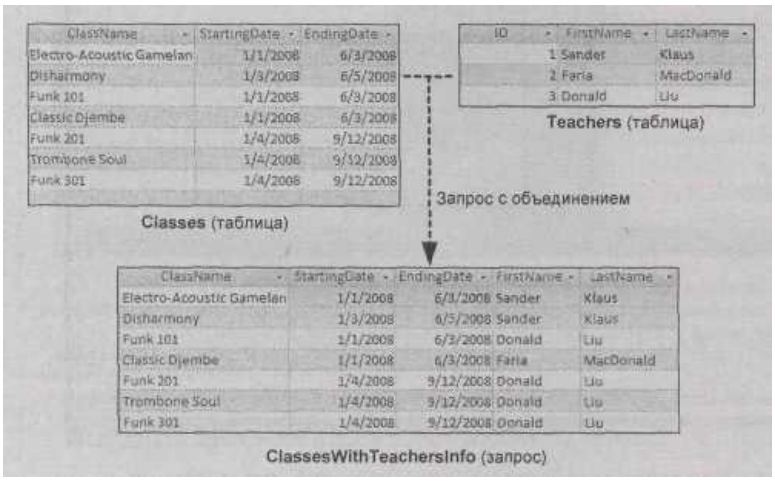


Рис. 6.16. Сама по себе таблица **Classes** содержит данные о каждом классе, но она предоставляет только идентификационный номер назначенного преподавателя. Соедините эту таблицу с таблицей **Teachers,** и вы получите любую интересующую вас информацию из связанной записи о преподавателе - включая его имя и фамилию

6.2.4.1. *Объединение таблиц в запросе*

Access делает удивительно легким объединение двух таблиц. Первый шаг - добавление обеих таблиц в ваш запрос, с помощью диалогового окна **Добавление таблицы** (Show Table).

Если в **Конструкторе** создается новый запрос, это окно появляется немедленно. Если вы работаете над уже созданным запросом, убедитесь, что вы в режиме **Конструктора,** щелкните правой кнопкой мыши окно и выберите в меню строку **Добавление таблицы.**

Если связь между таблицами уже определена (с помощью схемы данных, как описано в *разд. "Определение отношения" главы* 5, или созданием подстановки, как описано в *разд. "Поиск в связанных таблицах" главы* 5), программа Access использует эту связь для автоматического выполнения операции объединения. Вы увидите на схеме линию, соединяющую соответствующие поля, как показано на рис. 6.17.

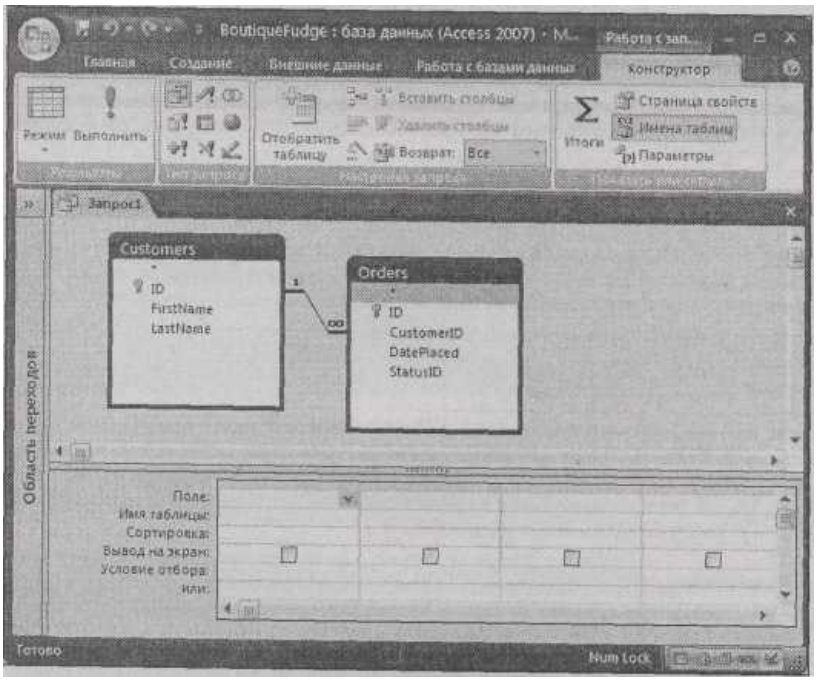


Рис. 6.17. Access автоматически соединяет поле **CustomerlD** в таблице **Orders** с полем ID в таблице **Customers,** основываясь на связи, определенной в БД

Если связь между двумя связанными таблицами еще не определена, возможно, это следует сделать до того, как создавать запрос (см. подробные инструкции в *главе* 5). Но если вы по

каким-то непонятным причинам решили не создавать связь (может быть, проект БД был введен в эксплуатацию другим, менее сообразительным специалистом Access), вы можете задать объединение вручную в окне запроса. Для этого просто перетащите мышью связанное поле одной таблицы на совпадающее поле в другой. Можно также удалить объединение, щелкнув правой кнопкой мыши линию объединения и выбрав команду **Удалить** (Delete).

Примечание

Если вы добавляете две несвязанные таблицы, программа Access пытается угадать связь, чтобы помочь вам. Если она находит поле с одинаковыми типами данных и именем в обеих таблицах, она добавляет связь для этих полей. Подобное действие зачастую совсем не то, что вам нужно - например, у многих таблиц есть общее поле **Код** (ID). Однако если вы строго соблюдаете правила проектирования БД, приведенные в *разд. "Шесть правил проектирования БД" главы 2,* у связанных полей имена в разных таблицах слегка отличаются, например **ID** и **CustomerID.** Если программа все же предлагает несуществующую связь, просто удалите ее, прежде чем выполнять нужное объединение.

На профессиональном уровне.

Сравнение: отношения и объединения

Важно понимать различия между отношением или связью и запросом с операцией объединения.

• *Отношение* - постоянная связь между двумя таблицами, хранящаяся в БД. Когда создается отношение в БД, есть возможность включить режим обеспечения целостности данных: набора правил, препятствующих проникновению противоречивых данных в связанные таблицы *(см. разд. "Целостность на уровне ссылок " главы 5).*

• *Объединение* - операция запроса, позволяющая соединять связанные данные из двух таблиц в одном результирующем наборе. Объединение не влияет на то, как вы вводите и редактируете объединенные данные в базовых таблицах.

Если отношение установлено, Access считает, что вы хотите использовать операцию объединения для связи этих таблиц в запросе - единственное действие, имеющее смысл.

После того как вы поместили две таблицы в окно Конструктора запросов и определили операцию объединения, можно выбирать нужные вам поля из обеих таблиц. Можно также добавить условия отбора записей и задать порядок их сортировки, как в запросе любого другого типа. На рис. 6.18 показан пример запроса, использующего операцию объединения, а на рис. 6.19 - результат выполнения этого запроса.

Примечание

Если у вас две таблицы, легко забыть, что вы выводите на экран. Если объединяются таблицы **Orders** и **Customers,** а затем из каждой выбираются поля, что вы получите в результате: список заказов или список клиентов? Ответ прост - вы получаете список заказов, дополненный информацией о клиентах. Запросы со связанными таблицами всегда обрабатывают таблицу-потомок и берут дополнительную информацию из таблицы-родителя.

Примечание

При выполнении объединения вы видите повторяющуюся информацию. Если объединяются таблицы **Customers** и **Orders,** вы видите имя и фамилию самого активного клиента рядом с несколькими заказами. Но при этом правило запрета дублирующихся данных не нарушается. Несмотря на то, что сведения о клиенте появляются в нескольких местах на листе данных с результатами запроса, в таблице **Customers** они сохраняются всего один раз.

Когда вы связываете родительскую и дочернюю таблицы с помощью запроса с операцией объединения, то выполняете запрос, получающий все записи из дочерней таблицы, и затем добавляете дополнительную информацию из родительской таблицы. Страница в конце

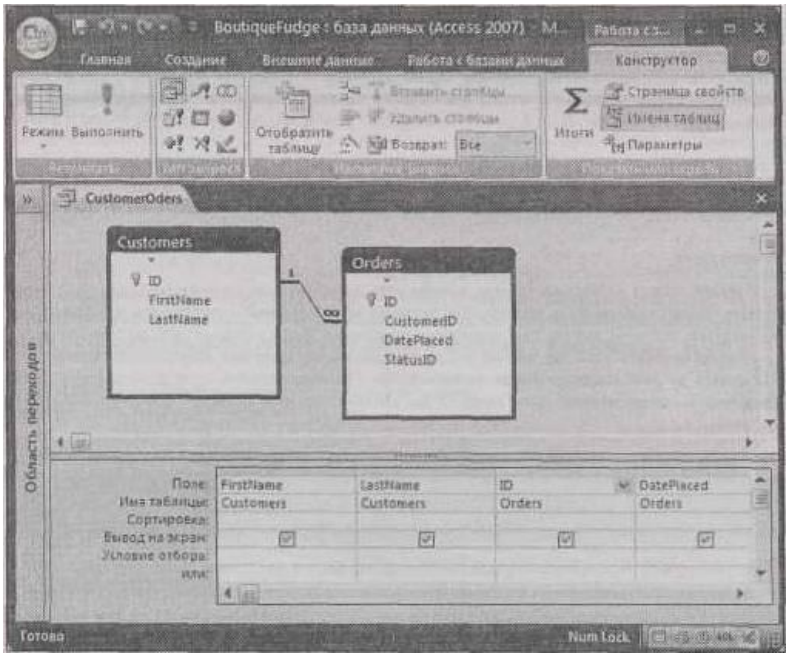


Рис.6.18. Этот запрос отображает данные из таблиц **Orders** и **Customers**. Неважно, из какой таблицы первое поле – в любом случае вы создаете список заказов с дополнительными сведениями о клиентах. Обратите внимание на строку **Имя таблицы** (Table) (под строкой **Поле**), указывающую на таблицу, из которой в запрос попадает каждое поле

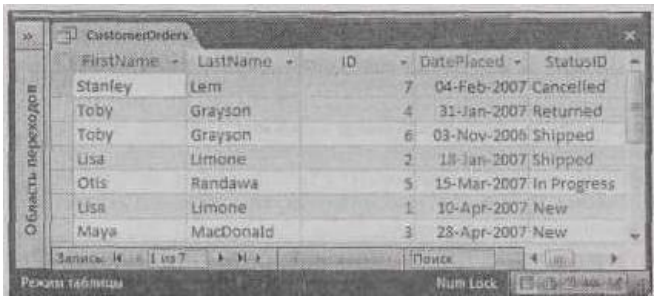


Рис.6.19. Вы легко можете при беглом просмотре увидеть, кто что заказал. Столбец **ID** содержит идентификационный номер заказа (хотя вы могли бы отобразить идентификационные номера как таблицы **Customers,** так и таблицы **Orders**)

Например, можно применить запрос с объединением для получения списка заказов (из дочерней таблицы) и дополнения каждой записи информацией о клиенте, сделавшем заказ. Независимо от способа объединения вы никогда не получите список клиентов с присоединенной информацией о заказе - в этом нет смысла, поскольку каждый клиент может сделать много заказов.

Операции объединения - одно из самых мощных средств в наборе инструментов любого разработчика запросов. Они позволяют отображать одну таблицу со всей нужной вам информацией.

Примечание

Когда используется несколько таблиц, всегда есть риск наличия одноименных полей в обеих таблицах. Такая возможность не страшна, если вы не собираетесь отображать эти поля в вашем запросе, но если это не так, возможны проблемы. Единственный способ различить поля - переименовать одно из них на листе данных с результатами запроса. Из *примечания "Малоизвестная или недооцененная возможность, Переименование поля в запросе" в разд. "Определение вычисляемого поля" главы 7* вы узнаете, как это сделать с помощью вычисляемого поля.

На профессиональном уровне.

Изменение данных при использовании запроса с объединением

Следует соблюдать осторожность при изменении данных в запросе, применяющем операцию объединения. Проблем не возникнет, если вы захотите изменить детали из таблицы-потомка. В

примере на рис. 6.19 достаточно легко изменить поля **DatePlaced** или **StatusID** в записи о заказе.

Но что произойдет, если вы измените одно из значений таблицы-родителя, например, имя или фамилию клиента? Очевидно, что сведения об одном и том же клиенте могут выводиться в запросе несколько раз. (Например, в запросе на рис. 6.19 отображены два заказа клиента с именем Toby.) Если изменить имя в одном месте, программа Access автоматически изменит информацию в таблице **Customers** и затем обновит весь запрос. Таким образом, если вы замените "Toby" на "Topy" на рис. 6.19, Access обновит вторую и третью строки на листе данных.

Проблема возникает, когда вы хотите изменить связь между записью о заказе и записью клиента. Например, вы хотите отредактировать заказ, сделанный Toby, так, чтобы в БД было записано, что этот заказ сделан клиентом Lisa. Но этого сделать невозможно с помощью редактирования в запросе полей **FirstName** и **LastName.** (Если вы отредактируете имя и фамилию, то просто измените запись о Toby в таблице **Customers.)** Вместо этого вам нужно откорректировать поле **CustomerID** в таблице **Orders** так, чтобы оно указывало на нужного клиента. Но в запрос, показанный на рис. 6.19, иоле **CustomerID** не включено, поэтому изменить связь невозможно.

6.2.4.2. *Внешние объединения*

В запросах, которые вы видели в предыдущем примере, используются операции, которые специалисты БД называют *внутренним объединением* (inner join). Внутренние объединения выводят только связанные записи - другими словами, записи, встречающиеся в обеих таблицах. Если выполнить запрос к таблицам **Customers** и **Orders,** вы не увидите клиентов, не сделавших ни одного заказа. Вы также не увидите заказов, не связанных с конкретным клиентом (пропущено значение в поле **CustomerID)** или не связанных с корректной записью (они могут содержать значение в поле **CustomerID,** не соответствующее ни одной записи в таблице **Customers).**

Внешние объединения (outer join) более емкие - они включают все результаты внутреннего объединения плюс оставшиеся в одной из таблиц (по вашему выбору) несвязанные записи. Очевидно, что эти несвязанные записи выводятся в окне результатов запроса с некоторыми пропущенными значениями, соответствующими пропущенным данным, которые должна была бы предоставить таблица.

Предположим, что вы выполняете внешнее объединение таблиц **Orders** и **Customers,** а затем настраиваете его так, чтобы выводились все записи о заказах. Все заказы, не связанные с записью о клиенте, появятся в нижней части списка и будут иметь незаполненные значения во всех полях со сведениями о клиенте (таких как **FirstName** и **LastName) -** табл. 6.1.

Таблица 6.1. Результат внешнего объединения таблиц Orders и Customers: все записи о заказах

FirstName	LastName	ID	DatePlaced	StatusID
Stanley	Lem	7	13-Jun-07	Cancelled
Toby	Grayson	4	03-NOV-06	Returned
Toby	Grayson	6	03-Nov-06	Shipped
		18	01-Jan-08	In Progress
		19	01 -Jan-08	In Progress

В данном конкретном примере нет смысла в существовании заказов, не связанных с конкретным клиентом. (На самом деле, скорее всего, это некорректно введенный заказ.) Если вы подозреваете наличие проблемы, внешнее объединение поможет устранить ее.

Подсказка

Вы можете помешать появлению записей-сирот с данными о заказе, сделав поле **CustomerID** обязательным *(см. разд. "Запрет незаполненных полей" главы 4)* и обеспечив ссылочную целостность *(см. разд. "Целостность на уровне ссылок" главы 5).*

Вы также можете выполнить внешнее объединение таблиц **Orders** и **Customers,** которое отображает все записи о клиентах. В этом случае в конце результатов запроса вы увидите все несвязанные записи о клиентах с соответствующими пустыми полями, которые должны были бы содержать сведения о заказах (табл. 6.2).

Таблица 6.2. Результат внешнего объединения таблиц Orders и Customers: все записи о клиентах

FirstName	LastName	ID	DatePlaced	StatusID
Stanley	Lem	7	13-Jun-07	Cancelled
Toby	Grayson	4	03-NOV-06	Returned
Toby	Grayson	6	03-Nov-06	Shipped
Ben	Samatara			
Goosey	Mason			
Tabasoum	Khan			

В данном случае запрос с внешним объединением выбрал трех отстающих. Хотите знать, как добавить внешнее объединение в ваш запрос? Начните с внутреннего объединения (которое программа Access добавляет автоматически, *см. разд. "Объединение таблиц в запросе" ранее в этой главе),* а затем преобразуйте его во внешнее. Для этого просто щелкните правой кнопкой мыши линию объединения, связывающую две таблицы в окне **Конструктора,** и выберите команду **Параметры объединения (Join** Properties) (или дважды щелкните кнопкой мыши эту линию). На экране появится одноименное диалоговое окно (рис. 6.20), позволяющее изменить тип используемого вами объединения.

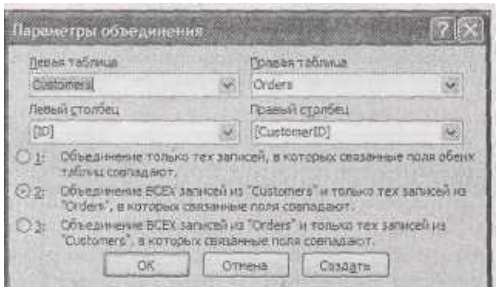


Рис. 6.20. Выбор первого переключателя **Объединение только тех записей, в которых связанные поля обеих таблиц совпадают** приводит к выполнению операции стандартного внутреннего объединения. Два оставшихся переключателя позволяют создать внешнее объединение, включающее все несвязанные строки одной из двух таблиц

6.2.4.3. *Поиск несвязанных записей*

Внутренние объединения - гораздо более распространенный тип объединений. Но внешние объединения позволяют создать, по крайней мере, один важный тип запроса: запрос, отслеживающий несвязанные записи (unmatched records).

Вы уже видели, как внешнее объединение дает возможность увидеть список всех ваших заказов плюс клиентов, не сделавших ни одного заказа. Эта комбинация не так уж полезна. Но отдел по маркетингу очень заинтересован во второй части этого сочетания - списке людей, до сих пор ничего не купивших. Эта информация поможет сотрудникам отдела провести кампанию поощрения клиентов, сделавших первую покупку.

Для выполнения такого запроса начните с запроса с внешним объединением, включающего все записи о клиентах. Затем добавьте один ингредиент: условие отбора, выбирающее записи без кода (ID) заказа. Технически это неопределенные (null) или пустые значения.

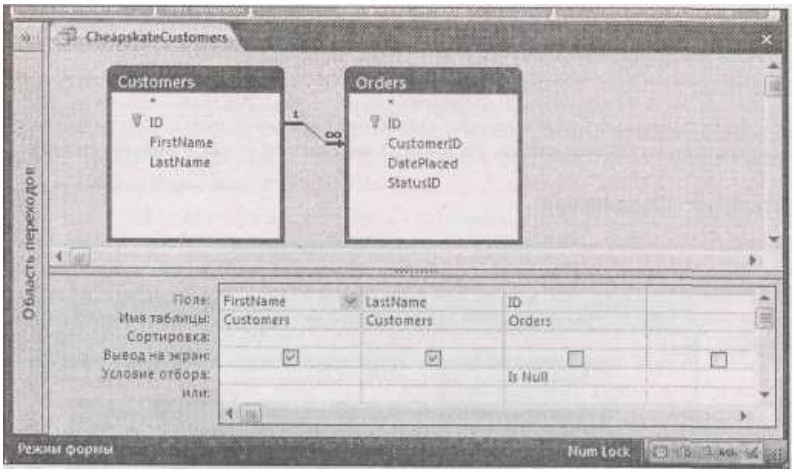


Рис. 6.21. В этом запросе сочетаются внешнее объединение и условие отбора, которому соответствуют только несвязанные записи о клиентах. Обратите внимание на то, что флажок **Вывод на экран** сброшен. Это сделано потому, что поле **ID** применяется в условии отбора, но в его присутствии на листе данных с результатами нет никакой необходимости

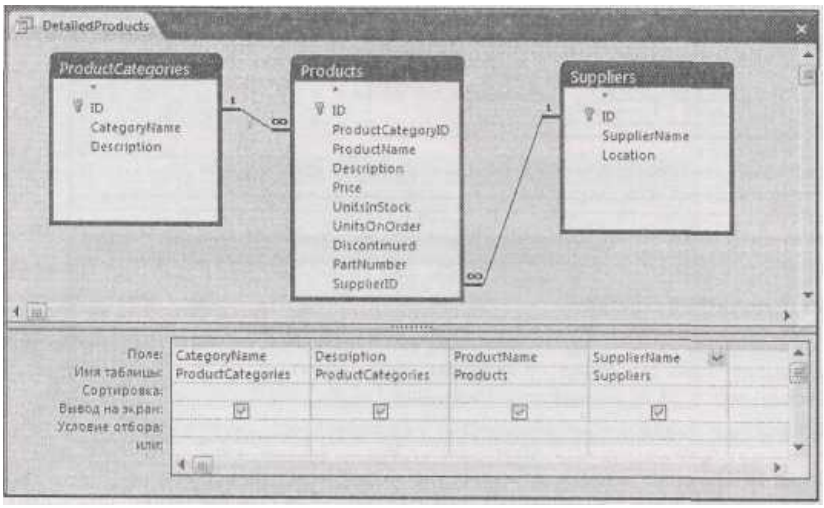


Рис. 6.22. В этом примере список продуктов усилен данными о категории продукта и сведениями о поставщике. Таблица **Products -** потомок как таблицы **ProductCategories,** так и таблицы **Suppliers** (поставщики), таким образом, данный запрос без усилий использует обе эти таблицы

Далее приведено нужное условие фильтрации, которое следует поместить в строке Условие отбора поля **ID** таблицы **Orders**:

IS Null

Теперь, когда программа Access выполнит запрос, она включит только записи клиентов, не связанные с записями в таблице заказов. На рис. 6.21 показан этот запрос в Конструкторе.

6.2.4.4. *Множественные объединения*

Как только вы освоились с внутренними и внешними объединениями, Access подбрасывает вам новую функциональную возможность. Многие запросы не ограничиваются одним объединением. Они используют три, четыре и больше объединений для соединения данных из многих связанных таблиц. Несмотря на то, что поначалу это кажется устрашающим, ничего сложного в этом нет.

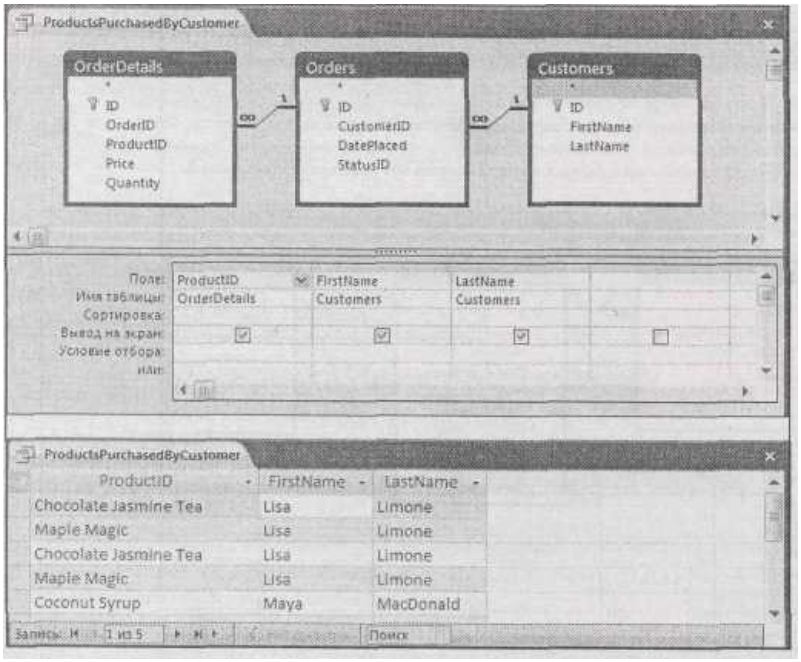


Рис. 6.23. Если вы хотите выяснить, кто заказал каждый продукт, вам понадобится связанная таблица Orders, а затем придется перейти к связанной таблице Customers Даже если вы не хотите отображать какие-либо данные из таблицы Orders, вам все равно не обойтись без этого двухшагового процесса. На верхнем рисунке показан запрос, реализующий этот процесс, а на нижнем - результат, который вы получите после выполнения запроса

Множественные объединения - всего лишь способ включения дополнительной связанной информации в ваш запрос. При наличии нескольких объединений каждое из них выполняется точно так же, как если бы оно было одним в запросе. Для использования множественных объединений добавьте в запрос все нужные вам таблицы с помощью диалогового окна **Добавление таблицы,** убедитесь, что между таблицами появились линии объединения и затем выберите, какие хотите, поля. Программе Access почти всегда хватает интеллекта, чтобы понять, что вы собираетесь делать.

На рис. 6.22 показан пример, в котором у таблицы-потомка два родителя, способных внести некоторую дополнительную информацию.

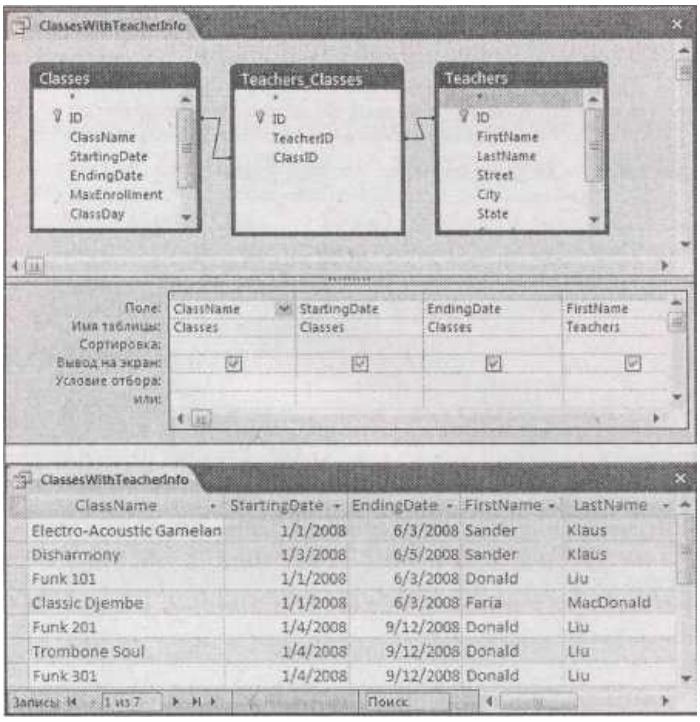


Рис. 6.24. Вы видите, как определить список курсов, содержащий рядом с каждым курсом имя и фамилию преподавателя, ведущего курс. На верхнем рисунке показана нужная вам структура

запроса, а на нижнем - результат

Иногда нужная вам информация находится в таблице, которая непосредственно не связана с основной таблицей запроса. Обратите внимание на таблицу **OrderDetails,** которую компания Boutique Fudge использует для перечисления всех товаров в заказе клиента. Сама по себе таблица **OrderDetaits** связана не с клиентом, заказавшим товары, а со связанной записью о заказе *(см. разд. "Заказ товаров" главы 5,* в котором обсуждается этот проект БД). Если вы хотите получить сведения о том, кто заказал каждый товар, следует добавить в запрос таблицы **OrderDetails, Orders** и **Customers,** как показано на рис. 6.23.

Множественные объединения незаменимы, если у вас есть отношение "многие-ко-многим" со связующей таблицей *(см. разд. "Отношение "многие-ко-многим"" главы 5),* как между преподавателями и курсами. Как вы помните из *главы 5,* музыкальная школа Cacophone Studios использует промежуточную таблицу для отслеживания назначений преподавателей для ведения конкретных курсов. Если вы хотите получить список учебных курсов, дополненный именем и фамилией преподавателя, ведущего курс, придется создать запрос с тремя таблицами: **Classes, Teachers** и **Teachers_Classes** (рис. 6.24).

7. Глава 7. Основные хитрости, применяемые в запросах

Все специалисты Access хранят в своих БД несколько (или несколько десятков) полезных запросов, упрощающих решение повседневных задач. В предыдущей главе вы узнали, как создавать запросы, обрабатывающие огромные объемы информации и предлагающие вам именно то, что вы хотите увидеть. Но знатоки программы Access знают, что гораздо больше мощи прячется за пределами окна конструктора запросов.

В этой главе вы проникните в магические тайны создания запросов, которые, конечно же, произведут впечатление на вашего шефа, коллег и партнеров. Вы узнаете, как выполнять вычисления в запросе и подводить итоги, сжимающие колонки цифр до лаконичных итоговых сумм. Вы также научитесь писать сверхинтеллектуальные условия отбора и создавать динамические запросы, запрашивающие информацию при каждом выполнении. Эти методы - неотъемлемая часть набора рабочих средств любого настоящего фанатика запросов.

7.1. *Вычисляемые поля*

Когда вы начинали проектировать таблицы, то узнали, что в мире БД считается преступлением включение информации, основанной на данных другого поля или другой таблицы. Примером такой ошибки может служить таблица **Products,** в которой есть и поле **Price** (цена), и поле **PriceWithTax** (цена с включенным налогом). Проблема в том, что поле **PriceWithTax** вычисляется на основании поля **Price.** Хранение обоих полей - это избыточное расходование дискового пространства. Еще хуже, если налоговая ставка изменится, тогда вы останетесь с множеством записей, нуждающихся в обновлении, и возможностью появления противоречивых данных (например, когда цена с налогом окажется ниже цены без налога).

Даже зная, что не следует создавать поля, такие как **PriceWithTax,** иногда вы вынуждены отображать в программе Access вычисляемые данные. Прежде чем компания Boutique Fudge напечатает список для одного из своих наименее любимых розничных продавцов, она хочет установить для цены надбавку 10%. Для этого компании необходимо откорректировать информацию о цене до вывода данных на печать. Если продавец увидит более низкую цену без надбавки, компания будет вынуждена запросить ее.

Запросы предлагают отличное решение такого рода проблем, поскольку они содержат универсальные методы математической обработки данных. Хитрость состоит в добавлении вычисляемого поля: поля, определенного в вашем запросе, но не существующего в таблице. Программа Access вычисляет значение этого поля, основываясь на одном или нескольких других полях таблицы. Значения вычисляемых полей никогда не хранятся в БД - программа генерирует их при каждом выполнении запроса.

7.1.1. *Определение вычисляемого поля*

Для создания вычисляемого поля следует задать два компонента: имя поля и выражение, определяющее вычисление, которые должна выполнить программа Access. Вычисляемые поля определяются с помощью следующего состоящего из двух частей шаблона:

ИмяВычисляемогоПоля: Выражение

Например, поле цены с налогом, **PriceWithTax,** определяется следующим образом:

```
PriceWithTax: [Price] * 1.10
```

По сути, это выражение сообщает программе Access о том, что нужно взять поле **Price** и умножить его на 1.10 (что эквивалентно повышению цены на 10%). Access повторяет это вычисление для каждой записи, входящей в результаты запроса. Для того чтобы это вычисление выполнялось, в таблице должно существовать поле **Price.** Но вовсе необязательно отображать отдельно это поле в окне результатов запроса.

Можно сослаться на поле **Price,** используя его полное имя, состоящее из имени таблицы с последующей точкой, за которой указано имя поля:

```
PriceWithTax: [Products].[Price] * 1.10
```

Такая синтаксическая запись нужна, если в ваш запрос включено несколько таблиц (например, использование запроса с операцией объединения (query join), описанного в *разд. "Запросы и связанные таблицы" главы* 6), и одно и то же поле есть в обеих таблицах. В этой ситуации следует

применять полное имя для того, чтобы избежать неоднозначности. (Если не сделать этого, Access выдаст сообщение об ошибке при попытке выполнить запрос.)

Примечание

Пользователи предыдущих версий программы Access иногда вместо точки используют восклицательный знак (например, [Products] ! [Price]), что равнозначно.

Для добавления вычисляемого поля **PriceWithTax** вам понадобится **Конструктор.** Сначала найдите столбец, в который вы хотите вставить вычисляемое поле. (Обычно оно добавляется и конец, в первый свободный столбец, хотя можно раздвинуть существующие столбцы и освободить для него место.) Далее в ячейке **Поле** введите полное определение поля (рис. 7.1).

Теперь вы готовы к выполнению запроса. Когда вы выполните его, вычисляемые данные появятся рядом с другими столбцами (рис. 7.2). Если вы не довольны тем, что вычисляемые данные несколько иначе отформатированы - больше знаков в дробной части и нет символа валюты - это можно исправить с помощью округления *(см. разд. "Применение функций " далее в этой главе)* и форматирования *(см.разд. "Форматирование чисел "далее в этой главе).*

У вычисляемых полей есть одно ограничение - поскольку информация не сохраняется в вашей таблице, вы не можете их редактировать. Если нужно изменить цену, необходимо отредактировать базовое поле **Price -** попытка корректировать поле **PriceWithTax** привела бы программу Access в полное замешательство.

Примечание

Выражение в каждый конкретный момент времени обрабатывает отдельную запись. Если вы хотите объединить информацию из некоторых записей для вычисления итогов или средних значений, необходимо использовать свойства группировки, описанные в *разд. "Итоговые данные" далее в этой главе.*

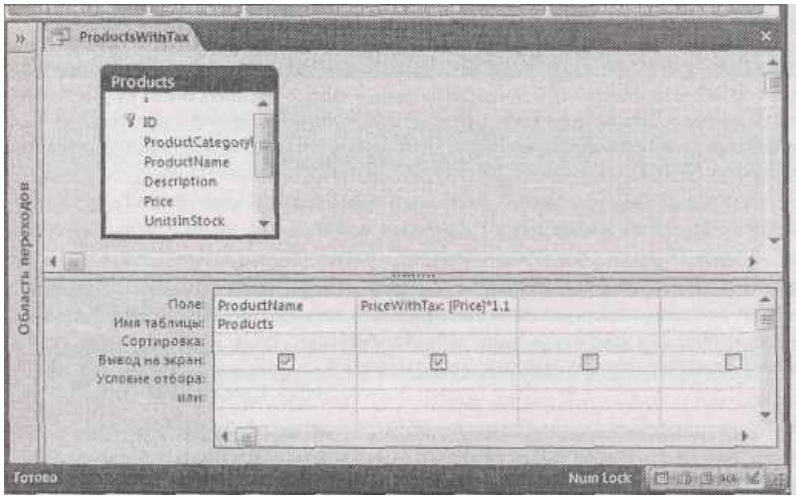


Рис. 7.1. Этот запрос отображает два поля непосредственно из БД (**ID** и **Name**) и вставляет вычисляемое поле **PriceWithTax**. Обычное поле **Price**, которое программа **Access** применяет для вычисления поля **PriceWithTax**, вообще не отображается

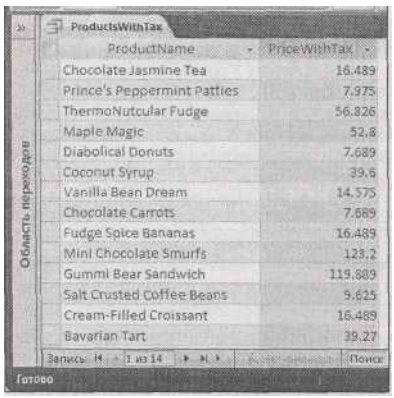


Рис. 7.2. Результаты запроса отображают поле **PriceWithTax** с надбавкой 10%. Главное состоит в том, что вычисляемая информация теперь доступна постоянно, несмотря на то, что она не хранится в БД. Попробуйте проверить это с помощью карманного калькулятора

На профессиональном уровне.

Синхронизация запросов

Можно опробовать интересный прием. Выполните запрос **ProductsWithTax** и оставьте его открытым, отображающим свои результаты. Теперь откройте таблицу **Products,** содержащую реальные данные, и измените цену любого продукта. Вернитесь снова в запрос **Products WithTax.** Изменилось значение в этом запросе?

Если вы не знаете, что произойдет, не бойтесь - **PriceWithTax** автоматически обновляется для отображения новой цены. Программа Access автоматически синхронизирует представления запросов с реальными данными в вашей таблице. Access отслеживает изменение записи и немедленно обновляет окно результатов запроса.

Есть лишь несколько исключений из этого правила.

• Access не отслеживает добавление новой записи после запуска запроса - для того чтобы она учитывалась в результатах вашего запроса, необходимо обновить результаты.

• Если вы изменили запись так, что она больше не должна попадать в результаты запроса, она не удаляется автоматически из окна результатов. Если в вашем запросе отображаются все продукты, стоящие больше 100 долларов, и вы уменьшили цену одного из них до 50 долларов, этот продукт останется в результирующем перечне вашего запроса (с новой ценой) до тех пор, пока вы не обновите результаты.

• Аналогично, если вы изменили запись так, что она должна попасть в результирующий набор, она не появится в нем, пока вы не выполните запрос еще раз.

• Если несколько пользователей на разных компьютерах редактируют БД (как описано в *главе 18),* вы не увидите немедленно изменения, внесенные другими пользователями.

Для получения самых свежих результатов можно обновить отдельные записи или весь запрос целиком. Для обновления одной записи выберите **Главная → Записи → Обновить → Обновить запись** (Home → Records → Refresh → Refresh Record). Для повторного выполнения запроса и полного обновления выберите **Главная → Записи → Обновить → Обновить все** (Home → Records → Refresh → Refresh All). Это действие также выводит на экран любые новые записи и скрывает те записи, которые после внесения изменений больше не удовлетворяют вашим условиям отбора.

Прежде чем двигаться дальше, есть смысл рассмотреть правила создания и использования вычисляемых полей. Далее приведено несколько рекомендаций.

■ *Всегда выбирайте уникальное имя.* Выражение Price: [Price] * 1.10 создает циклическую ссылку, поскольку имя используемого вами поля такое же, как имя создаваемого поля. Программа Access не допускает подобных проделок.

■ *Формируйте выражения из полей, чисел и математических операций.* Наиболее распространенные вычисляемые поля содержат одно или несколько существующих полей или числовые константы и соединяет их друг с другом с помощью хорошо знакомых знаков математических операций, таких как сложение (+), вычитание (-), умножение (*) или деление (/).

■ *Не удивляйтесь присутствию квадратных скобок.* Выражение PriceWithTax: [Price] * 1.10 эквивалентно выражению PriceWithTax: Price * 1.10 (единственное отличие - квадратные скобки вокруг имени поля **Price).** Технически скобки нужны только, если в имени ноля есть пробелы или специальные символы. Но если в **Конструкторе** вы вводите в запрос выражения без квадратных скобок, программа Access автоматически добавляет их, просто чтобы обезопасить себя.

Малоизвестная или недооцененная возможность.

Переименование поля в запросе

Устали от длинных имен полей в окне результатов ваших запросов? Используя толькочто полученные знания, касающиеся выражений, можно безболезненно переименовать поле в окне результатов вашего запроса. Все что вам требуется - это вычисляемое поле. Хитрость заключается в создании (с помощью выражения) вычисляемого поля, совпадающего с одним из

существующих полей и присвоении ему нового имени. Технически в этом поле не выполняются никакие вычисления, но оно все равно корректно действует. Далее приведен пример вычисляемого поля, которое переименовывает **DateCustomerPlacedPurchaseOrder** в **Date:** Date: DateCustomerPlacedPurchaseOrder Новое имя (в данном примере **Date)** называют *псевдонимом* (alias).

Используя этот прием, помните о том, что исходное иоле (в данном случае **DateCustomerPlacedPurchaseOrder)** не надо включать в ваш запрос. Нужная вам информация отображается в вычисляемом поле **(Date).**

7.1.2. *Простая математическая обработка числовых полей*

Многие вычисляемые поля полагаются на обычные арифметические операции. В табл. 7.1 представлен краткий обзор основных вариантов комбинирования чисел.

Таблица 7.1. Арифметические операции

Операция	Название	Пример	Результат
+	Сложение	1+1	**2**
-	Вычитание	**1-1**	**0**
	Умножение	**2*2**	**4**
	Возведение в степень	**2^3**	**8**
/	Деление	**5/2**	**2.5**
\	Деление нацело (возвращает наименьшее целое число и отбрасывает остаток)	**5\2**	**2**
Mod	Остаток от деления (возвращает остаток, полученный в результате деления нацело)	**5** Mod **2**	**1**

Для создания выражения можно использовать произвольное количество полей и операций. Рассмотрим таблицу **Products** с полем **QuantityInStock** (количество на складе), в котором записано количество единиц товара, находящихся в вашем хранилище. Для определения стоимости данного товара, имеющегося у вас под рукой, введите следующее выражение, применяющее два поля:

ValueInStock: [UnitsInStock] * [Price]

Подсказка

При выполнении математической операции над полем может произойти сбой, если в поле пропущено значение. Для устранения этой проблемы вам необходима функция Nz ( ), описанная в *разд. "Обработка пропущенных или неопределенных значений" далее в этой главе.*

Поля с датами

Операции сложения и вычитания можно применять к полям, содержащим даты. (Можно применять и умножение, деление и все что угодно, но реального смысла такие значения иметь не будут.)

Применяя операцию сложения, вы можете добавить обычное число к полю с датой. Это число сдвигает дату вперед на указанное число дней. Далее приведен пример, добавляющий дополнительные две недели к сроку платежа, установленному компанией:

ExtendedDeadline: [DueDate] + 14

Если это выражение применить к дате 10 января 2007 г., новая дата будет.24 января 2007 г.

Используя операцию вычитания, можно найти число дней между двумя датами. Далее показано, как вычисляется период между временем помещения заказа и временем доставки:

ShippingLag: [ShipDate] - [OrderDate]

Если доставка произошла через 12 дней после того, как сделан заказ, вы увидите значение 12.

Примечание

Поля типа **Дата/время** включают информацию о времени суток. В вычислениях данные о времени представляются как дробная часть числа. Если вы вычли одну дату из другой и получили число 12.25, оно представляет период 12 дней и 6 часов (поскольку 6 часов равно 25% суток).

Помните, что если вы хотите включить фиксированные даты в ваши запросы, их необходимо обрамлять символами # и использовать формат *Месяц/День/Год* (Month/Day/Year). Далее приведен пример, использующий такой подход для вычисления количества дней между датой своевременного предоставления задания (20 марта 2007 г.) и датой действительного его предоставления: LateDays: [DateSubmitted] - #03/20/07#

Положительное значение указывает на то, что значение в поле **DateSubmitted** больше (более поздняя дата), чем предельный срок сдачи - другими словами, студент опоздал. Значение 4 указывает, что студент на 4 дня задержался, в то время как -4 означает, что студент сдал работу на четыре дня раньше назначенного в расписании срока.

Порядок выполнения операций

Если о вашем выражении длинная строка вычислений, программа Access следует строгим правилам определения старшинства операций или, говоря математическим языком, учитывает, какое вычисление выполняется первым при наличии нескольких вычислений в выражении. Итак, если у вас длинное выражение, Access не просто обрабатывает его слева направо. Вместо этого программа оценивает выражение фрагмент за фрагментом в следующем порядке:

1. Скобки (любые вычисления, заключенные в скобки, Access всегда выполняет первыми).
2. Проценты.
3. Возведение в степень.
4. Деление и умножение.
5. Сложение и вычитание.

Предположим, что вы хотите использовать поля **QuantityInStock** (количество на складе) и **QuantityOnOrder** (количество в заказах) для подсчета всех товаров, имеющихся в наличии и находящихся на пути к клиенту. Если вы не знаете правил старшинства операций, то можете сформировать следующее выражение:

TotalValue: [UnitsInStock] + [UnitsOnOrder] * [Price]

Проблема состоит в том, что программа Access перемножит поля **QuantityOnOrder** и **Price** и добавит полученное значение к значению ноля **QuantityInStock.** Для устранения этой оплошности нужно применить скобки следующим образом:

TotalValue: ([UnitsInStock] + [UnitsOnOrder]) * [Price]

Теперь поля **QuantitylnStock** и **QuantityOnOrde**r суммируются, а затем умножаются на поле **Price** для получения общей суммы.

Подсказка

Вам нужно больше свободного места для набора длинного выражения? Можно расширить любой столбец в **Конструкторе** запросов для увеличения видимой зоны, но в случае сложных вычислений этого все равно не хватит. Лучше щелкнуть кнопкой мыши в области **Поле** и нажать сочетание клавиш <Shift>+<F2>. Это действие распахивает на экране диалоговое окно **Область ввода** (Zoom), отображающее все содержимое в большом текстовом поле, разделенном на столько строк, сколько вам нужно. Когда просмотр или редактирование выражения завершены, щелкните мышью кнопку **ОК,** чтобы закрыть окно и сохранить изменения, или кнопку **Отмена** для отказа от них.

7.1.3. *Выражения с текстовыми значениями*

Несмотря на то, что обычно вычисляемые поля имеют дело с числовыми данными, но так бывает не всегда, существуют действительно удобные способы обработки текста.

Если у вас есть текстовые данные, конечно нельзя применять сложение, вычитание и другие арифметические операции. Но можно слить текст. Есть возможность связать несколько полей с адресной информацией и отображать их все в одном поле, экономя пространство (и, возможно, облегчая экспорт этих данных и другую программу).

Для слияния текста применяется оператор амперсанд (&). Далее показано, как создать поле **FullName** (ФИО), в котором собрана информация из полей **FirstName** и **LastName:**

FullName: [FirstName] & [LastName]

Это выражение выглядит достаточно корректным, но на самом деле у него есть один недостаток. Поскольку вы не вставили никаких пробелов, имя и фамилия в результате прижаты друг к другу, например, так: BenJenks. Лучше слить три фрагмента текста: имя, пробел и

фамилию. Вот исправленная версия: FullName: [FirstName] & " " & [LastName]

Приведенное выражение создаст значение: Ben Jenks. Можно также поменять местами имя и фамилию и отделить их запятой, если вы предпочитаете указывать фамилию первой (например, Jenks, Ben) для облегчения сортировки: FullName: [LastName] & ", " & [FirstName]

Примечание

В программе Access есть два типа текстовых значений: те, которые вы извлекаете из других полей, и те, которые вы вводите непосредственно (или фиксированные). Когда вы вводите текстовые константы, такие как запятая или пробел в предыдущем примере, их следует заключать в кавычки, чтобы программа Access знала, где начинается и заканчивается текст.

Вы даже можете применять амперсанд для сцепления текста с числовыми значениями. Если вы хотите, чтобы слегка бесполезный текст "The price is" появлялся перед каждым значением цены, примените следующее вычисляемое поле:

Price: "The price is: " & [Price]

7.2. *Функции запросов*

Теперь, возможно, вам приходит в голову мысль о том, что манипулировать числами и текстом можно более искусными способами - способами, которые предоставляют гораздо больше возможностей, чем простые арифметические операции. Вам может потребоваться округление чисел или преобразование строчных букв в прописные. Программа Access действительно предоставляет средства, называемые *функциями* и позволяющие поднять ваши выражения на более высокий уровень.

Функция - это реализация встроенного алгоритма, принимающая ваши данные, выполняющая вычисления и возвращающая результат. Разница между функциями и математическими операциями, как вы уже убедились на деле, заключается в том, что функции могут выполнять более сложные действия. У программы Access есть каталог с десятками разных функций, многие из которых реализуют такие приемы, которые вы даже не надеялись выполнить самостоятельно.

Функции пригодятся для всех видов обработки данных в программе Access.

Вы можете примять их в:

■ *вычисляемых полях* для включения данных в результаты ваших запросов;

■ *условиях отбора* для задания записей, отображаемых в запросе;

■ *коде на языке Visual Basic,* многоцелевой расширяемой системе для программы Access, с которой вы познакомитесь в *части V.*

Изучив как следует мир функций, вы поймете, что многие из них хорошо подходят для вычисляемых полей, но не для условий отбора. В следующих разделах вы увидите, где именно разумнее всего использовать каждую из них.

Примечание

Функции - это встроенная часть версии языка SQL *(см. разд. "Режим SQL" главы 6),* применяемой в программе Access для обработки данных.

7.2.1. *Применение функций*

Применяете ли вы простейшую или сложнейшую функцию, синтаксис - правила применения функции в выражении - остается неизменным. Для использования функции просто введите ее имя с последующими скобками. Затем внутрь скобок поместите все данные, которые нужны функции для выполнения ее вычислений (если это необходимо).

Для примера рассмотрим очень полезную функцию Round (), которая принимает число с дробной частью и убирает любые нежелательные позиции в дробной части. Эта функция - удобный способ приведения в порядок отображаемых значений вычисляемого поля. Вы оцените полезность функции Round (), если создадите выражение, подобное приведенному далее и вычисляющее цены, сниженные на 5%:

SalePrice: [Price] * 0.95

Примените это выражение к цене $43.97 и получите в результате 41.7715, что вряд ли будет выглядеть уместно на ценнике товара. Здесь пригодится функция Round (). Передайте ей

неокругленное число и число знаков в дробной части, которое нужно сохранить:

SalePrice: Round([Price] * 0.95, 2)

Технически функции Round ( ) требуется две порции информации или два аргумента. Первый - это число, которое округляется (в данном случае это результат вычисления Price * 0. 95), а второй - количество цифр, которое вы хотите оставить справа от десятичной точки (2). В результате получается значение, округленное до двух десятичных знаков, или 41.77.

Примечание

Большинству функций, подобных Round (), требуются два или три аргумента. Но некоторые функции могут принять гораздо больше, в то время, как нескольким функциям вообще не нужны аргументы

Часто задаваемый вопрос

Банковское округление

Похоже, программа Access округляет числа неправильно. Как быть?

Вас может удивить то, что Access округляет число 21.985 до 21.98. Если вас учили округлять до большего числа, числа заканчивающиеся цифрой 5, то вы считаете, что результат должен быть 21.99. Этот способ называют *арифметическим округлением.* Программа Access не применяет арифметическое округление - она выбирает *банковское округление,* которое лучше в некоторых случаях.

Разница между арифметическим и банковским округлением заключается в трактовке цифры 5. Поскольку число 21.985 находится точно в середине, между числами 21.98 и 21.99, не просто решить, что с ним делать. При постоянном округлении числа с 5 на конце до большего числа вносится систематическое отклонение в итоги и средние значения. Поскольку вы округляете до большего чаще, чем до меньшего, любые итоги или среднее, которые вы вычисляете, получаются чуть больше, чем следовало бы.

Банковское округление решает эту проблему округлением 5 в одних случаях до большего числа, а в других до меньшего, в зависимости от соседней четной или нечетной цифры.

Число 21.985 округляется до меньшего числа 21.98, а число 21.995 - до большего, 22. Это не единственный способ борьбы с систематическим отклонением (можно решать случайным образом, когда округлять, а когда нет), но это общепринятая практика в бухгалтерских расчетах и статистике.

Вложенные функции

В вычисляемом поле или условии отбора можно применять несколько функций. Этот прием известен как *вложенные функции:* специальный термин, обозначающий вставку одной функции внутрь другой. Например, в программе Access есть встроенная функция определения абсолютного значения числа Abs ( ), преобразующая отрицательные числа в положительные (и оставляющая положительные числа без изменения). Далее приведен пример деления одного поля на другое и получения в результате заведомо положительного значения:

Speed: Abs ([DistanceTravelled] / [TimeTaken])

Если полученный результат нужно округлить, можно поместить целиком все выражение внутрь скобок функции Round ( ):

Speed: Round (Abs([DistanceTravelled] / [TimeTaken]), 2)

Вычисляя выражение с вложенными функциями, программа Access сначала вычисляет результат самой внутренней функции. В данном примере, прежде всего, определяется абсолютное значение, а затем результат округляется. В приведенном далее примере порядок вычислений изменен на обратный без изменения результата:

Speed: Abs(Round([DistanceTravelled] / [TimeTaken], 2))

Во многих случаях порядок вложения функций имеет значение, и разный порядок приводит к различным результатам.

Вложенные функции быстро превращают выражение в потенциально опасное. Даже в сравнительно простом примере вычисления скорости трудно предсказать результат без по-шагового вычисления выражения. Одна не на месте стоящая или пропущенная скобка может привести к сбою вычисления. Если вы применяете вложенные функции, не пишите сразу выражение целиком - включайте функции в выражение поочередно и после вставки очередной

функции выполняйте вычисление.

7.2.2. *Построитель выражений*

Функции - замечательное нововведение, но в программе Access может оказаться слишком много замечательных вещей. Программа предлагает перечень из десятков различных функций, связанных с решением разнообразных задач, некоторые из этих функций предназначены для выполнения специализированных математических и статистических вычислений.

Примечание

В этой книге описана далеко не каждая функция {если бы это было так, вам пришлось бы, сидя над книгой, бороться со сном). Но в следующих разделах вы познакомитесь с наиболее полезными функциями для работы с числами, тестом и датами. Для поиска дополнительных функций используйте **Построитель выражений** (Expression Builder) Если же вы предпочитаете интерактивный режим обучения, проверьте информативный Web-ресурс **www.techonthenet.com/access/functions.**

Для быстрого поиска нужных вам функций Access предлагает компонент, именуемый **Построителем выражений.** Для его запуска выполните следующие действия:

1. Откройте запрос в **Конструкторе.**
2. Щелкните правой кнопкой мыши поле, в которое вы хотите вставить выражение, и выберите команду **Построить** (Build).

Если вы создаете вычисляемое поле, нужно щелкнуть правой кнопкой мыши в ячейке **Поле,** Если создается условие отбора, следует щелкнуть правой кнопкой мыши в ячейке **Условие отбора.**

После выбора команды **Построить** на экране появляется окно **Построителя выражений,** отображающее текущее содержимое поля (рис. 7.3).

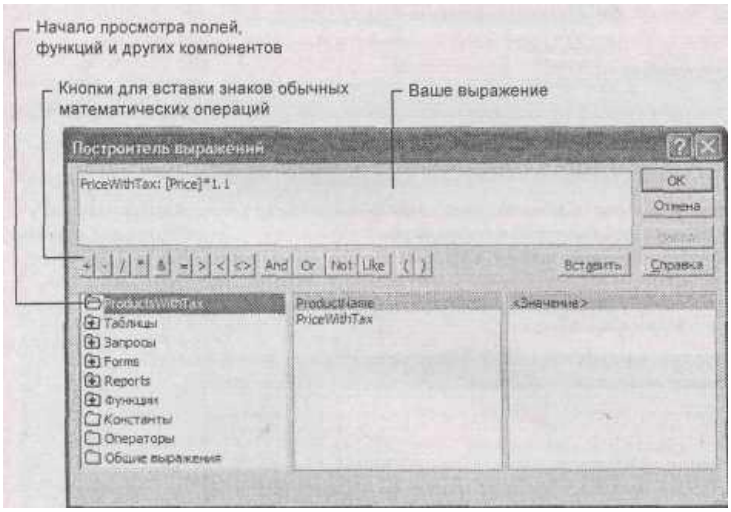


Рис. 7.3. Построитель выражений состоит из текстового поля в верхней части окна, в котором можно редактировать выражение, кнопок быстрой вставки знаков обычных операций (таких как +, - , / и *, если почему-либо вы не можете их найти на клавиатуре), и трехпанельного обозревателя в нижней части окна, который поможет найти нужные поля и функции

3. Вставьте или отредактируйте выражение.

У **Построителя выражений** есть два ускоряющих приема работы, которые вы, может быть, захотите испробовать. Можно вставлять имя без ввода с клавиатуры (рис. 7.4) и можно найти функцию с помощью обзора (рис. 7.5).

Примечание

Построитель выражений - универсальное средство создания выражений в вычисляемых полях и условиях отбора. Некоторые параметры имеют смысл только в одном из его назначений. Логические операторы, такие как символ равенства (=), And, Or, Not и Like, удобны для задания условий фильтрации *(см. разд. "Построение условий отбора" главы 6),* но бесполезны в вычисляемых полях.

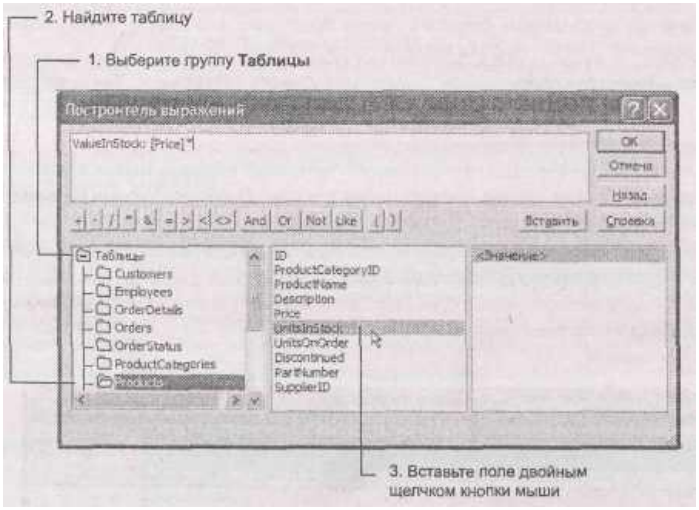


Рис. 7.4. Для вставки имени поля щелкните дважды кнопкой мыши папку **Таблицы** в самом левом списке. Затем щелкните мышью вложенную папку, соответствующую нужной вам таблице. И, наконец, дважды щелкните кнопкой мыши имя поля в среднем списке для добавления его в ваше выражение. Этот прием рекомендуется только тем, кто любит щелкать кнопкой мыши

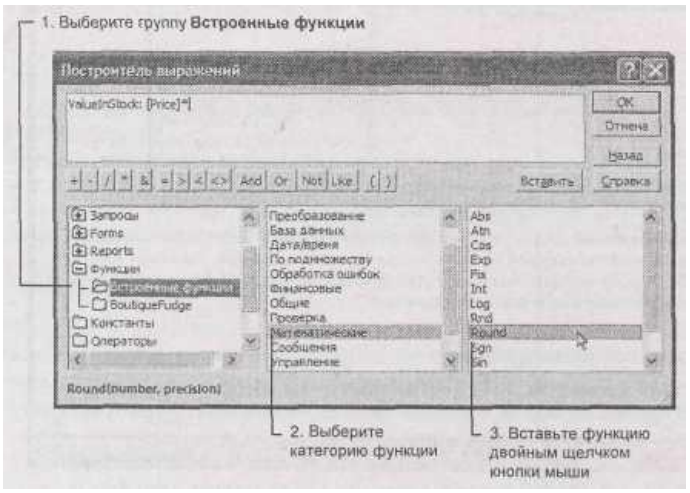


Рис. 7.5. Для поиска функции начните с двойного щелчка кнопкой мыши папки **Функции** в левом списке. Затем выберите вложенную папку **Встроенные функции.** (Другие варианты отображают любые пользовательские функции, которые вы добавили в вашу БД с помощью пользовательского кода VBA.) Далее выберите категорию функции в среднем списке. В правом списке показаны все функции в выбранной категории. Вы можете дважды щелкнуть кнопкой мыши функцию для вставки ее в выражение

Примечание

Когда вы вставляете имена полей в **Построителе выражений,** они записываются в более длинном формате с обязательным указанием имени таблицы. Вы увидите [Products] ! [Price] вместо просто [Price]. Не волнуйтесь - для программы Access это одно и то же.

4. Щелкните мышью кнопку **ОК**.

Программа Access скопирует ваше новое выражение в ячейку **Поле или Условие отбора.**

Примечание

Когда в **Построителе выражений** вставляется функция, программа добавляет заполнители (например, <number> и <precision>), на место которых нужно ввести аргументы. Замените этот текст нужными вам значениями.

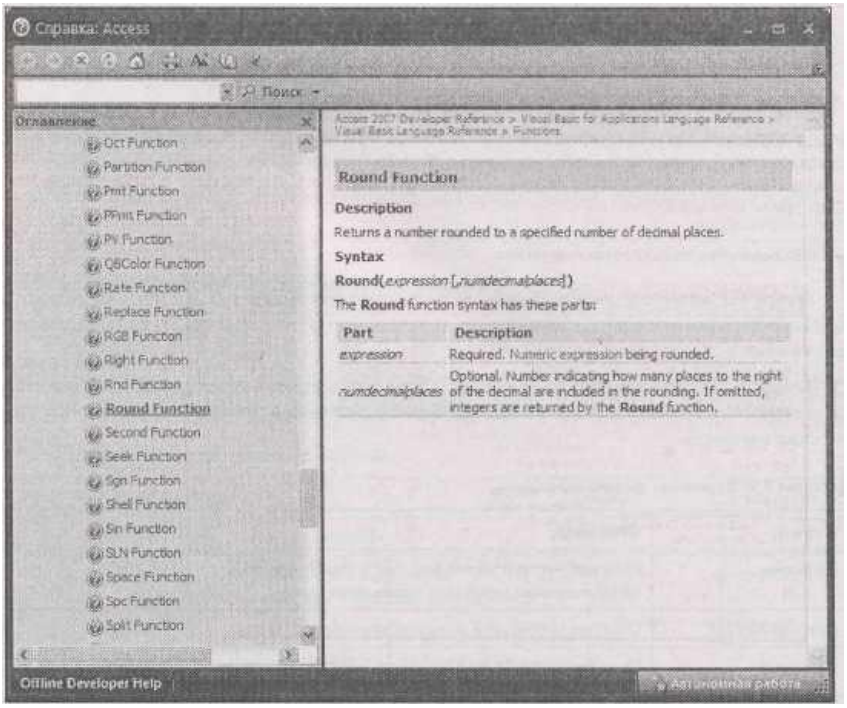


Рис. 7.6. В справке функции Round () описываются ее назначение и два параметра. Второй параметр - количество знаков в дробной части - заключен в квадратные скобки, что означает необязательное значение. Пропустите его, и программа Access округлит значение до ближайшего целого числа. Слева приведен перечень функций, который позволяет просмотреть любую другую функцию Access и прочесть ее описание

Большинство специалистов Access считают **Построитель выражений** слишком тяжеловесным, чтобы использовать его в работе. Но несмотря на то, что это не самое эффективное средство написания выражений, построитель предоставляет прекрасный способ знакомства с новыми загадочными функциями, благодаря своей справке для встроенных функций. Если вы нашли функцию, которая кажется многообещающей, но хотите получить дополнительную информацию, выберите ее в списке и щелкните мышью кнопку **Справка** (Help). Вы будете вознаграждены кратким резюме, объясняющим назначение функции и описывающим ее параметры, которые нужно задать, как показано на рис. 7.6.

7.2.3. *Форматирование чисел*

Format ( ) - интересная математическая функция, преобразующая числа в текст. Она интересна, потому что создаваемый текст можно отформатировать несколькими способами, управляя, таким образом, представлением чисел.

Для того чтобы понять разницу, вернемся к выражению, использованному ранее для снижения цены:

```
SalePrice:  [Price]  *  0.95
```

Даже если у поля **Price** не **Денежный** тип, вычисленные значения в поле **SalePrice** (продажная цена) выводятся как обычные числа (без знака валюты, разделителя тысяч и т. д.). Вы увидите значение 43.2 вместо желаемого $43.20.

Решить эту проблему можно, применив функцию Format ( ) для задания денежного формата вывода:

```
SalePrice:  Format([Price]  /  0.95,  "Currency")
```

Теперь вычисленные значения содержат знак валюты. Более того, поскольку денежные суммы отображаются с двумя знаками после точки, вам не нужно больше применять функцию Round ( ).

Хитрость применения функции Format () состоит в выборе текста, задаваемого в качестве второго аргумента для получения желаемого результата. В табл. 7.2. приведены возможные варианты.

Таблица 7.2. Варианты форматирования

Формат	Описание	Пример
Денежный	Выводит число с двумя знаками в дробной части, разделителями для тысяч и знаком валюты	$1 433.20
Фиксированный	Отображает число с двумя десятичными знаками	1433.20
Основной	Выводит на экран число с двумя десятичными знаками и разделителями тысяч	1 433.20
Процентный	Отображает процентное значение (число, умноженное на 100, и со знаком процента). Выводит две цифры справа от десятичной точки	143320.00%
Экспоненциальный	Отображает число в научной нотации с двумя десятичными знаками	1.43E+03
Да/нет	Отображает *Нет,* если число равно 0, и Да, если число отлично от 0. Можно использовать аналогичные типы формата *Истина/Ложь* и *Вкл/Выкл*	*Да*

Практические занятия для опытных пользователей.

Улучшенные числовые форматы

Истинные педанты не будут довольны вариантами, перечисленными в табл. 7.2. Им нужен полный контроль количества десятичных знаков в числе. Один из возможных вариантов - использовать функции FormatCurrency( ), FormatPercent ( ) и FormatNumber ( ) (в зависимости от необходимости вывода значения как денежного, процентного или обычного числового). В этих функциях в качестве первого аргумента задается число, которое нужно отформатировать, а в качестве второго - число десятичных знаков, которое нужно сохранить.

Для более полного контроля можно определить собственный формат, точно описывающий то, что вы хотите получить, и затем применить его в функции Format (). В данной книге не рассматриваются пользовательские числовые форматы, по вы можете посмотреть дополнительную информацию в справочной системе программы Access (см. рис. 7.6).

7.2.4. *Дополнительные математические функции*

Математическим функциям в программе Access не уделяется должного внимания, потому что потребность в них возникает крайне редко. Вы уже видели функции Round ( ) и Format ( ) - самые полезные в этой категории - но есть еще несколько других (табл. 7.3), к которым знатоки Access обращаются время от времени в вычисляемых полях.

Таблица 7.3. Функции для числовых данных

Функция	Описание	Пример	Результат
Sqr ( )	Извлекает квадратный корень	Sqr(9)	3
Abs()	Возвращает положительное значение (отрицательные числа становятся положительными)	Abs(-6)	6
Round ()	Округляет число до заданного числа десятичных знаков	Round(8.89, 1)	3.9
Fix ( )	Возвращает целую часть числа, отбрасывая любую дробную часть	Fix(8.89)	8
Int( )	То же что функция Fix (), но отрицательные числа округляются до ближайшего меньшего целого числа, а не большего	Int(-8.89)	-9
Rnd()	Генерирует случайное дробное число в диапазоне от 0 до 1	Int((6)* Rnd+1)	Случайное целое от 1 до 6
Val ()	Преобразует числовые данные в текстовом поле в настоящее число так, что вы можете использовать его в вычислении. Останавливается, как только находит нецифровой символ, и возвращает 0, если не найдено ни одной цифры	Val ("315 Crossland St")	315
Format()	Преобразует число в форматированную текстовую строку в соответствии с выбранными вами параметрами	Format(243.6 , Currency)	$243.60

Малоизвестная или недооцененная возможность.

Использование случайных чисел для сортировки в случайном порядке

Функцией Rnd ( ) пользуются редко - в конце концов, кому нужны столбцы, заполненные искусственно сгенерированными данными? Однако инициативные гуру Access предложили одно интригующее применение для функции Rnd ( ). Они применяют ее для сортировки таблицы, таким образом, чтобы записи выводились в случайном порядке.

По существу вы добавляете вычисляемое поле, содержащее случайное число. Можно использовать в поле выражение, такое как Random: Rnd ( ). Если посмотреть результаты вашего запроса, то можно увидеть случайное значение в диапазоне от 0 до 1 (например, 0.7045, 0.2344 и т. д.) рядом с каждой записью.

Теперь вернитесь в Конструктор и сбросьте флажок **Вывод на экран** для того, чтобы поле **Random** не выводилось на лист данных. Далее выберите порядок в ячейке **Сортировка** по возрастанию или по убыванию (что на самом деле не имеет значения) и снова выполните запрос. Ву-а-ля! Каждое выполнение запроса выводит на экран записи в разном порядке, в соответствии со случайными числами, которые программа Access генерирует на лету.

7.2.5. *Текстовые функции*

Все функции, которые вы видели до этого момента, работали с числовыми данными. Но с текстом тоже можно делать многое. В целом есть три способа обработки текста.

- *Слияние текста.* Вы можете соединить несколько текстовых полей в одном. Для этого способа не нужна функция - достаточно оператора &, описанного в *разд. "Выражения с текстовыми значениями "ранее в этой главе.*
- *Извлечение подстроки из текстовой строки.* Может быть, вам потребуется первое слово из заголовка или первые 100 символов в описании.
- *Замена строчных букв прописными и наоборот.* Возможно, вы захотите отобразить строчные буквы прописными или наоборот.

В табл. 7.4 перечислены функции, наиболее часто применяемые для обработки текста.

Таблица 7.4. Функции для работы с текстом

Функция	*Описание*	*Пример*	*Результат*
UCase( )	Выводит текст прописными буквами	UCase("Hi There")	HI THERE
LCase( )	Выводит тест строчными буквами	LCase("Hi There")	hi there
Left ( )	Выводит заданное вами число символов, начиная от левого края строки	Left("Hi There", 2)	Hi
Right 0	Выводит заданное вами число символов, начиная от правого края строки	Right ("Hi There", 5)	There
Mid ( )	Выводит часть строки, начиная с заданной позиции, и заданное число символов	Mid ("Hi There", 4, 2)	Th
Trim ( )	Удаляет пробелы с обеих сторон (или используйте LTrim ( ) и RTrim( ) для удаления пробелов только в начале или в конце строки)	Trim(" Hi There ")	Hi There
Len ( )	Подсчитывает количество символов в текстовой строке	Len("Hi There")	8

С помощью этих функций вы можете создать вычисляемое поле, которое отображает фрагмент длинной текстовой строки или изменяет вид отображения (строчные или прописные буквы). Применение этих функций в условиях отбора не столь очевидно. Можно создать условие фильтрации, задающее совпадение с частью текстовой строки, а не со всей строкой. Далее приведен пример условия отбора, выбирающего записи, начинающиеся с "Choco":

```
Left([ProductName], 5) = "Choco"
```

На рис. 7.7 показано, как ввести это условие отбора.

Функция Len ( ) - особый случай. Она проверяет текстовое значение и возвращает числовую информацию (в данном случае количество символов в строке, включая все пробелы, буквы, цифры и специальные символы). Эта функция не слишком полезна в простых вычисляемых

выражениях, т. к. вас редко будет интересовать количество букв в текстовой строке. Но она позволяет создавать интересные условия отбора, включая, например, такое, которое отбирает все записи с полем Description короче 15 символов.

```
Len{Description)  <  15
```

Практические занятия для опытных пользователей.

Как извлечь первое слово из текстовой строки

Функции обработки строк созданы с ориентацией на символы. Они умеют подсчитывать символы, но не понимают, что такое слово пли предложение.

Единственный способ обойти это ограничение - применить функцию Instr ( ), которая ищет один или несколько символов в текстовой строке. (Название Instr ( ) - это сокращение от "in string", поскольку вы ищите конкретные символы внутри текстовой строки.) Для поиска символов "he" с строке "Hi There" нужно применить функцию Instr ( ) следующим образом:

```
Instr("Hi  There",   "he")
```

Результат равен 5, потому что текст "he" начинается с пятой символьной позиции. Если программа Access не находит совпадения, функция Instr ( ) возвращает 0. Если же есть множественные совпадения, Instr () возвращает позицию первого.

Сама по себе эта функция не очень полезна в условиях отбора и вычисляемых полях. Но ее можно использовать в сочетании с другими функциями, такими как Mid ( ) и Left ( ), для вырезания части строки, расположенной рядом с какой-либо буквой. Можно применить функцию Instr ( ) для поиска первого пробела и вырезать весь текст *до* этого пробела. Таким образом, вы извлечете целое слово.

Далее приведено слегка ошеломляющее вычисляемое поле, которое получает первое слово из поля **ProductName** с помощью вложенных функций *(см. разд. "Вложенные функции" ранее в этой главе).* Оно разбито на несколько строк для того, чтобы разместить его на странице книги. Когда будете его набирать, поместите все выражение в одну строку.

```
FirstWordProduct:
Left([ProductName],   Instr([ProductName],    " " -  1 ) )
```

Это выражение переводится следующим образом: "Найди позицию первого пробела, вычти единицу и извлеки все символы слева от пробела". Вычислите это поле для значения Banana Cream Fudge и вы получите вырезанный текст Banana, что выглядит как впечатляющий яркий трюк.

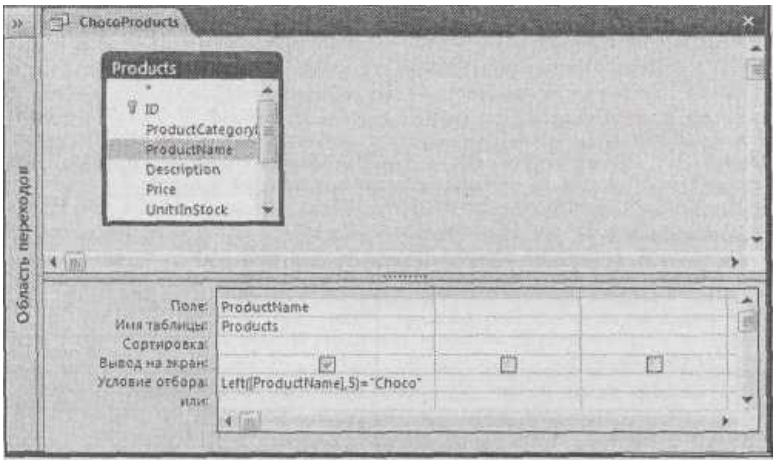


Рис. 7.7. Функции Left ( ), Right ( ) и Mid ( ) действуют во многом аналогично ключевому слову Like, помогая найти совпадения кусочков длинной текстовой строки

7.2.6. *Функции для обработки дат*

Вы уже видели, как можно использовать операции сложения и вычитания при работе с датами *(см. разд. "Поля с датами "ранее в этой главе).* Но вы можете выполнить гораздо больше действий с помощью некоторых функций Access для работы с датами.

Несомненно, многие применяют функции Now () и Date (), с которыми вы познакомились в

главе 4. Эти функции извлекают текущие дату и время или только текущую дату. Их можно применять в запросах, работающих с заказами, принесшими доход в текущем году.

Вот условие для выбора просроченных проектов:

=<Date ( )

Вставьте его в ячейку **Условие отбора** поля **DueDate** (срок платежа) и вы увидите только те записи, в которых поле **DueDate** содержит дату, наступившую ранее нынешнего дня.

Анализ дат может быть более сложным в сочетании с функцией DatePart (), которая извлекает часть информации из даты. DatePart () может определить номер месяца или год, позволяя игнорировать другие подробности (такие как число или время). С помощью DatePart () и Date () можно легко написать условие фильтрации, отбирающее заказы, сделанные в текущем месяце.

DatePart("m", [DatePlaced])=DatePart("m", Date())

And DatePart("yyyy", [DatePlaced])=DatePart("yyyy", Date ())

Это довольно длинное выражение на самом деле представляет собой комбинацию двух условий, соединенных ключевым словом And. Первое условие сравнивает номер месяца текущей даты с датой, хранящейся в поле **DatePlaced:**

DatePart("m", [DatePlaced])=DatePart ("m", Date( ))

Приведенное выражение устанавливает, что у обеих дат один и тот же календарный месяц, но вы должны также убедиться в том, что год у них тоже совпадает:

DatePart("yyyy", [DatePlaced])=DatePart("yyyy", Date ( ))

Сложность применения функции DatePart ( ) (и некоторых других функций для дат) заключается в понимании идеи компонентов, составляющих дату. Применяя символ m в функции DatePart ( ), вы получите номер месяца, а используя текст yyyy, извлечете чете-рехсимвольный номер года. В табл. 7.5 приведены все возможные варианты.

Таблица 7.5. Компоненты даты

Компонент	*Описание*	*Значение на 20 февраля, 2006 г. 1:30 PM*
yyyy	Год в четырехсимвольном формате	2006
q	Квартал от 1 до 4	1
т	Месяц от 1 до 12	2
y	День в году, от 1 до 365 (обычно)	51
d	День в месяце от 1 до 31	20
w	День недели, от 1 до 7	2
ww	Неделя в году, от 1 до 52	8
h	Час, от 1 до 24	13
n	Минута, от 1 до 60	30
s	Секунда, от 1 до 60	0

Для тех, кто понимает.

Вычисления для дат и времени

Используя функции для дат, всегда следует помнить о датах, содержащих информацию о времени. (Напоминаю, все даты могут содержать данные о времени суток. Но, выбирая подходящий формат для поля с датами, вы сообщаете программе Access о том, нужно ли отображать временной компонент даты и разрешать пользователям вводить его, как объясняется в *разд. "Дата/время " главы 2.* Чаще всего вы будете пользоваться форматом, скрывающим любую информацию о времени суток.)

Проблема: функция Date ( ) возвращает текущую дату со значением времени суток, равным 0. Другими словами, если текущая дата - 4 июля 2008 г., то функция Date ( ) возвращает первую секунду 4 июля 2008 г. - момент, когда часы показывают полночь (12:00 a.m.)

Если вы не храните значения времени суток, эта проблема не важна, поскольку у всех ваших дат время суток равно 0. Но что произойдет, если вы используете *Полный формат даты* (см. табл. 2.3) в поле **DueDate,** разрешающий пользователям вводить и дату, и время. Теперь у условия отбора =<Date ( ) несколько иной смысл - вы сообщаете Access о необходимости отобрать, как совпадающие, все поля с датами, наступившими до первой секунды текущего дня. Это условие не выберет запись со сроком платежа, назначенным на 16:00 текущего дня.

В данной ситуации, возможно, нужно изменить условие отбора на следующее: <(Date()+l)

Date () +1 - это завтра. Другими словами, это условие отбирает любые записи со сроком платежа, истекшим до первой секунды завтрашнего дня.

Между прочим, у программы Access есть функция Now (), возвращающая текущую дату и время. Таким образом, следующее условие фильтрации отбирает все записи, соответствующие текущему моменту (текущего дня) или любому моменту времени во все предшествующие дни:

=<Now( )

Компоненты дат применяются в нескольких функциях обработки дат, включая функции DatePart ( ), DateAdd ( ) и DateDiff ( ). В табл. 7.6 приведены эти и дополнительные полезные функции, относящиеся к датам.

Таблица 7.6. Функции обработки дат

Функция	*Описание*	*Пример*	*Результат*
Date ()	Получает текущую дату	Date ()	1/20/2006
Now ()	Получает текущую дату и время	Now ( )	1/20/2006 10:16:26 PM
DatePart ()	Извлекает часть даты (например, год, месяц или день в месяце)	DatePart(#1/20/2006#, "d")	20
DateSerial()	Преобразует год, месяц и день в значение даты Access	DateSerial(2006, 5, 4)	5/4/2006
DateAdd ( )	Сдвигает дату на заданный интервал	DateAdd("yyyy", 2, #22/ll/2006#)	22/11/2008
DateDiff ( )	Определяет интервал между двумя датами	DateDiff("w", #10/l5/2006#, #l/ll/2007#)	12
MonthName ( )	Получает название, соответствующее номеру месяца (от 1 до 12)	MonthName (1)	"January"
WeekdayName ( )	Получает название, соответствующее номеру дня в неделе (от 1 до 7)	WeekdayName (1)	"Sunday"
Format ()	Преобразует дату в форматированный текст (используя любой формат даты, описанный в табл. 2.3)	Format(#27/04/2008#, "Long Date")	"April 27, 2008"

Подсказка

У программы Access есть другие функции обработки дат, выполняющие часть алгоритма функции DatePart ( ). Примером может служить функция Month ( ), извлекающая номер месяца из даты. К другим аналогичным функциям относятся Year(), Day ( ), Hour ( ), Minute ( ) и Second ( ). Эти функции не дают никаких преимуществ, но вы можете встретить их в запросах других пользователей, применяющих их для получения аналогичных результатов.

7.2.7. *Обработка пропущенных или неопределенных значений*

В БД есть два типа полей: обязательные и необязательные. Как правило, в БД поля необязательные (как обсуждалось в *разд. "Запрет незаполненных полей" главы 4),* что означает для неаккуратного пользователя возможность пропуска большого числа значений. Эти пропущенные значения называют *неопределенными* (null), и их следует тщательно обрабатывать.

Если вы хотите написать условие отбора, вылавливающее неопределенные значения, просто введите в ячейку Условие отбора следующий текст: Is Null

Это условие выберет все записи с пропущенными значениями. Воспользуйтесь им в поле CustomerID таблицы Orders для поиска всех заказов, не связанных с клиентом. Или игнорируйте несвязанные записи, заменив условие отбора на обратное:

Is Not Null

Иногда вам не нужно специально искать (или игнорировать) неопределенные значения. Вместо этого вам надо заменить их для рассматриваемой задачи чем-то более информативным. К счастью, как раз для этого есть функция со странным названием Nz ( ).

У функции Nz () два аргумента. Первое значение (как правило, поле запроса) может содержать неопределенное значение. Второй параметр - это значение, которое вы хотите отобразить в результатах запроса, если программа Access найдет неопределенное значение. Далее приведен пример, использующий функцию Nz ( ) для преобразования в 0 неопределенных значений в поле **Quantity:**

Nz([Quantity], 0)

Преобразование в 0 - стандартное поведение для функции Nz ( ), поэтому можно опустить второй параметр, если это как раз то, что вам нужно:

Nz([Quantity])

В данный момент, возможно, вас не сильно впечатляет перспектива замены пропущенных значений нулями. Но эта функция жизненно важна, если нужно создать вычисляемые поля, обрабатывающие значения, которые могут быть неопределенными. Рассмотрим кажущийся безобидным пример:

OrderItemCost: [Quantity] * [Price]

Это выражение приведет к ошибке, если значение поля **Quantity** будет неопределенным. Неопределенные значения странным образом распространяются наподобие инвазивного грибка. Если у одного из операндов в вычислении неопределенное значение, результат автоматически становится неопределенным. В данном примере это означает, что иоле **OrderItemCost** для данной записи становится неопределенным. Хуже того, если **OrderItemCost** ввести в другое вычисление или промежуточный итог, они тоже станут неопределенными. Прежде чем вы об этом узнаете, ваши значимые данные запроса превратятся в кучу ячеек с неопределенными значениями.

Для устранения этой проблемы очистите необязательные ноля от неопределенных значений с помощью функции Nz ( ): OrderItemCost: Nz([Quantity]) * Nz([Price])

Наконец, функцию Nz ( ) можно использовать для замены всех неопределенных значений другой величиной, В текстовое поле можно ввести что-то более информативное. Далее приведен пример, отображающий текст ("Not Entered" - не введено) рядом с каждой записью, не содержащей имени и фамилии:

Name: Nz([FirstName] & [LastName], "[Not Entered]")

7.3. *Итоговые данные*

Все запросы, которые вы применяли до этого момента, имели дело с отдельными записями. Если вы выбирали 143 записи из таблицы **Orders,** то видели все 143 записи в ваших результатах. Но вы также можете группировать записи для получения промежуточных и общих итогов. В этом случае легче просматривать большие объемы информации и делать важные далеко идущие выводы.

Далее приведены примеры полезных подводящих итоги запросов:

- подсчет студентов в каждом классе;
- подсчет количества заказов, сделанных каждым клиентом;
- сумма, потраченная на один продукт;
- общая сумма долга или платежа клиента;
- подсчет среднего объема заказа, сделанного каждым клиентом;
- поиск самого дорогостоящего или самого дешевого заказа, сделанного клиентом.

Эти операции - подсчет, суммирование, определение среднего и поиск максимального и минимального значений - основные варианты в итоговом запросе (totals query). *Итоговый запрос* - это вид запроса, который должен перелопатить большое количество записей и выдать лаконичные итоги.

Для создания итогового запроса выполните следующие действия:

1.Создайте новый запрос, выбрав **Создание → Другие → Конструктор запросов** (Create → Other → Query Design).

2.Добавьте нужные вам таблицы с помощью диалогового окна **Добавление таблицы** (Show Table) и щелкните мышью кнопку **Закрыть** (Close).

В приведенном далее примере используется таблица **Products** из БД Boutique

Fudge.

3. Вставьте поля, которые хотите использовать.

В этом примере используется поле **Price,** но хитрым образом: поле **Price** вставляется три раза для того, чтобы в запросе отображались результаты трех разных вычислений.

4. Выберите **Работа с запросами → Показать или скрыть → Итоги** (Query Tools → Show/Hide → Totals).

Программа Access вставляет ячейку **Групповая операция (**Total) для каждого поля сразу под ячейкой **Таблица.**

5. Для каждого поля задайте вариант из списка **Групповая операция.** Этот вариант определяет использование поля для вычисления итога или для группировки.

Итоговый запрос немного отличается от обычного запроса. Каждое поле должно попадать в одну из следующих категорий.

■ *Поле используется в итоговом вычислении* (таком как определение среднего, подсчет количества и т. д.). Тип нужного вычисления выбирается с помощью ячейки **Групповая операция.** В табл. 7.7 перечислены все варианты из ячейки **Групповая операция.**

■ *Поле применяется для группировки.* Обычно итоговые запросы соединяют в большой общий итог. Но вы можете разбить результаты на более мелкие промежуточные итоги, как описано в следующем разделе.

■ *Поле используется для фильтрации или отбора.* В этом случае в ячейке **Групповая операция** нужно выбрать Условие (WHERE). (Фанаты БД, возможно, помнят, что WHERE - это ключевое слово, применяющееся для определения условия в языке SQL, как было описано в *разд. "Анализ запроса" главы 6).* Нужно также сбросить флажок **Вывод на экран,** поскольку программа Access не может выводить отдельные значения в итоговых сводках.

Примечание

Если вы попытаетесь вставить в итоговый запрос поле, которое не используется для вычисления или группировки и не скрыто, то получите ошибку при попытке выполнить запрос.

В этом примере (рис. 7.8) в поле Price применяются три разные групповые операции: Max, Min и Avg.

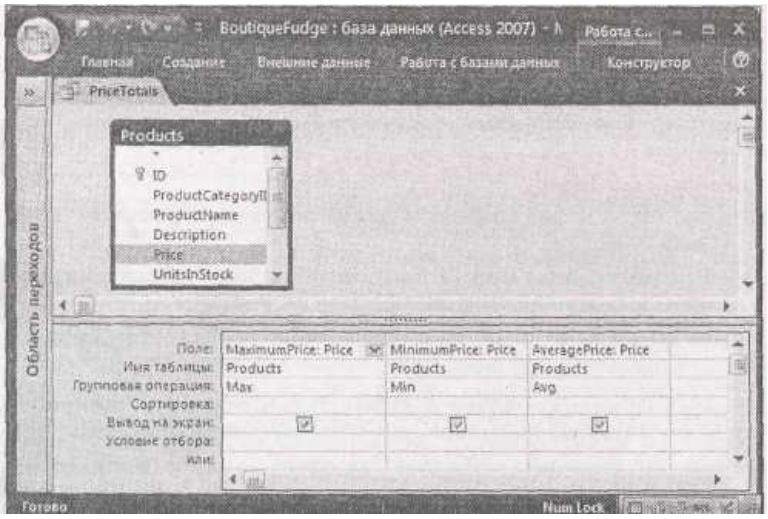
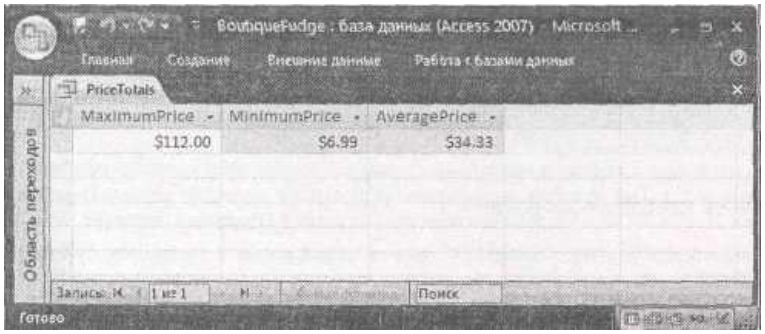


Рис. 7.8. *Вверху:* в данный итоговый запрос поле **Price** включено трижды и использует три разных вычисления. Обратите внимание на то, что в каждое поле применяется выражение, в котором дано более информативное название поля. *Внизу:* результаты отображают одну запись с максимальной ценой, минимальной ценой и средней ценой продуктов, проданных компанией Boutique Fudge



Примечание

В табл. 7.7 не указаны два варианта, предназначенные для статистиков - **StDev** и **Var** - которые вычисляют стандартное отклонение и дисперсию ряда чисел.

Таблица 7.7. Варианты получения итоговых данных

Значение в ячейке ***Групповая операция***	**Описание**
Группировка	Группирует записи, основываясь на значениях в этом поле
Sum	Суммирует значения этого поля
Avg	Находит среднее для значений этого поля
Min	Возвращает наименьшее значение в этом поле
Max	Возвращает наибольшее значение в этом поле
Count	Подсчитывает количество записей (независимо от того, какое поле вы используете)
First	Возвращает первое значение в этом поле
Last	Возвращает последнее значение в этом поле

Разрабатывая итоговые запросы, можно использовать все приобретенные ранее в этой главе навыки написания запросов. Если вы хотите суммировать только продукты в конкретной категории, можно использовать в поле **CategoryID,** например, такое условие отбора:

=3

Это условие отбирает записи, у которых **CategoryID** равно 3 (что означает их включение в категорию Candies (конфеты)).

Примечание

Если вы хотите выполнить фильтрацию в поле, которое не используется в вычислении или группировке, убедитесь, что в ячейке **Групповая операция** выбран вариант **Условие** и сброшен флажок **Вывод на экран.**

7.3.1. *Группировка в итоговом запросе*

Наипростейший итоговый запрос суммирует все выбранные записи в одну строку результатов, как показано на рис. 7.8. В более сложном итоговом запросе применяется группировка для вычисления промежуточных итогов.

Для корректного применения группировки следует помнить о том, что поле, которое вы используете, должно содержать много повторяющихся значений. Например, хорошо группировать клиентов по штатам, в которых они живут. Поскольку в каждом штате много клиентов, у вас получатся осмысленные промежуточные итоги. Глупо группировать клиентов по номерам социального обеспечения, поскольку в результате получится столько групп, сколько у вас клиентов. На рис. 7.9 показан итоговый запрос с применением группировки.

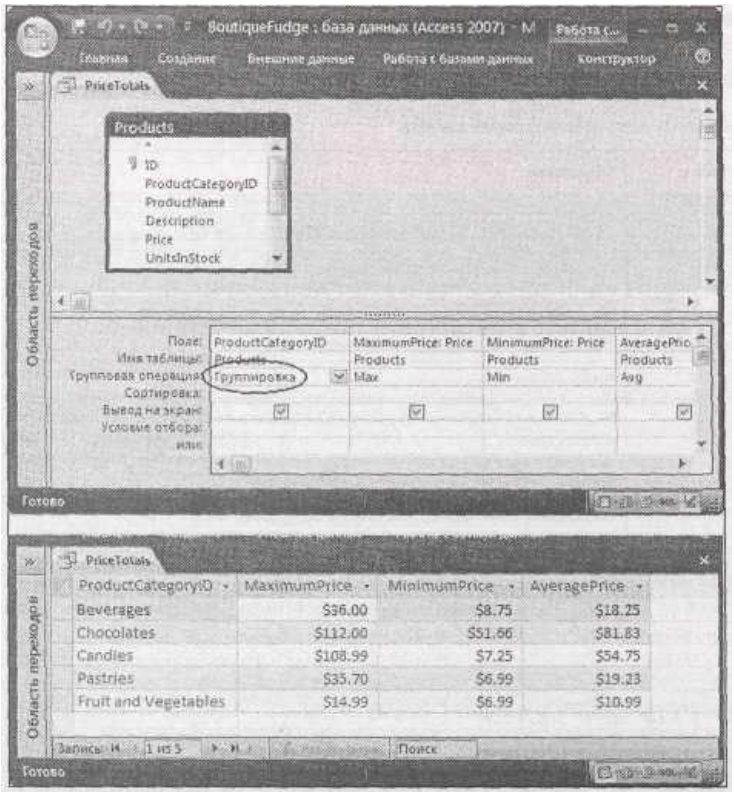


Рис. 7.9. *Вверху:* продукты сгруппированы по категории продукта. *Внизу:* результат - отдельная строка с итогами по каждой категории продуктов

В итоговом запросе можно использовать *многоуровневую* группировку, вставив несколько полей со значением **Группировка** в ячейку **Групповая операция.** Но результаты могут оказаться неожиданными. Предположим, что вы группируете длинный список записей о продажах по наименованиям товаров и по именам клиентов. В результате вы получите отдельную группу для каждой комбинации "клиент - товар". В табл. 7.8 приведена часть результатов запроса, подобного описанному, в котором группируются записи из таблицы **OrderDetails** БД Boutique Fudge, а затем они сортируются по **CustomerID.**

Таблица 7.8. Результаты запроса с многоуровневой группировкой

CustomerID	ProductID	TotalSales
10	108	$432.12
10	134	$16.79
10	210	$53.30
14	144	$18.99
18	112	$107.04
18	210	$12.02

Из этой таблицы видно, что клиент с номером 10 потратил в целом $432.12 на товар с номером 108 во всех заказах. Этот же клиент потратил $16.79 на товар с номером 134, $53.30 - на товар с номером 210 и т. д. (Вы можете взять эти данные и отсортировать их по коду товара **ProductID,** чтобы увидеть объемы продаж каждого товара для разных клиентов. У вас все та же информация, но анализировать ее можно по-разному.)

Это результат, который вы хотели получить. Но ему недостает хороших промежуточных итогов. Было бы полезно узнать, сколько клиент с номером 10 потратил на каждый тип продукта и сколько он потратил всего. Но из-за жесткой табличной структуры итогового запроса получить такой результат невозможно.

Если вы хотите увидеть эту разбитую на подгруппы информацию с промежуточными итогами, у вас есть две возможности. Можно использовать перекрестный запрос или запрос к сводной таблице - два улучшенных варианта подведения итогов, которые описаны в *главе 9.* Если же вас на самом деле интересует вывод вашей информации на печать, можно создать отчет, включающий

многоуровневую группировку и итоги, как описано в *части III.*

7.3.2. *Объединения в итоговом запросе*

Итоговые запросы невероятно полезны, когда они комбинируются с объединениями *(см. разд. "Запросы и связанные таблицы" главы 6)* для получения связанной информации из нескольких таблиц. В БД Boutique Fudge таблица **OrderDetails** хранит отдельные компоненты каждого заказа. Вы можете сгруппировать эту информацию (как показано в предыдущем разделе) для поиска самых ходовых товаров или самых активных клиентов. Но, к сожалению, вы увидите только значения кодов клиентов и товаров, которые малоинформативны.

Примечание

Если у вас есть подстановка, определенная для полей **ProductID** и **CustomerID**, вы увидите описание из поля подстановки (например, наименование товара, имя и фамилию клиента). Эта информация немного полезнее, но, возможно, вы хотите извлечь дополнительные сведения - например, адрес клиента, описание товара и т. д. - находящиеся вне связанной таблицы.

Если вставить в запрос объединение или два, можно извлечь подчиненную информацию из связанных таблиц (таких как **Customers, Products** и **Orders)** и добавить ее в ваши результаты. На рис. 7.10 показан пример вычисления общей стоимости каждого заказа. Затем результаты отсортированы по коду клиента **CustomerID.**

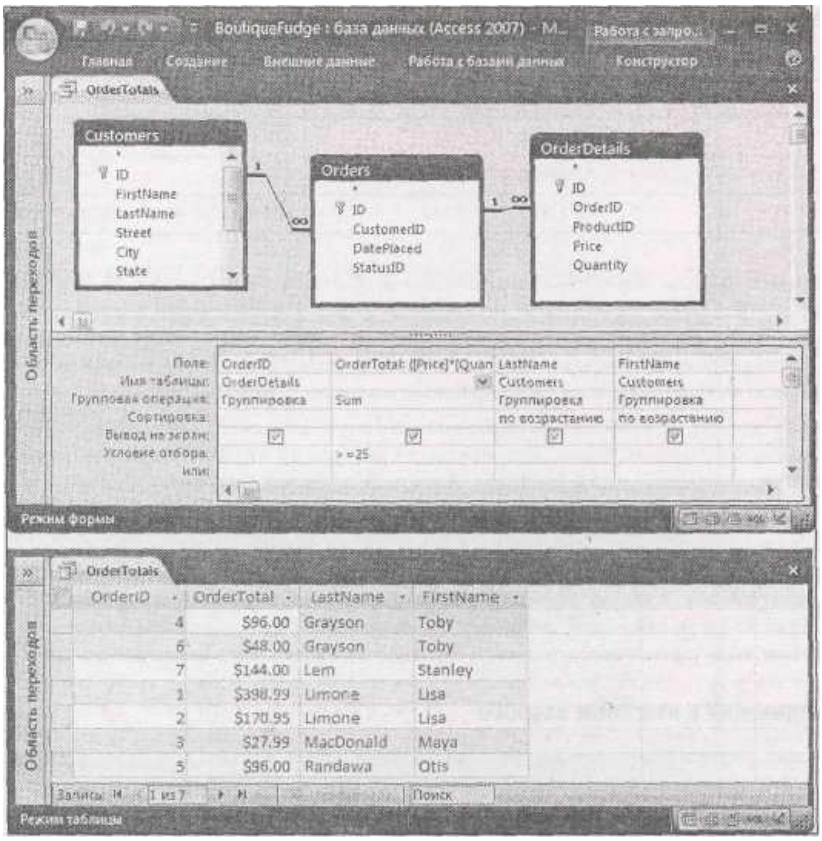


Рис. 7.10. *Вверху:* итоговый запрос можно усовершенствовать, дополнив его информацией из трех связанных таблиц: **Customers, Orders** и **OrderDetails,** для отображения списка сумм заказов, упорядоченных по клиентам. В запросе игнорируются заказы стоимостью меньше $25. Можно вставить условие отбора в поле **DatePlaced,** чтобы узнать, сколько потратили клиенты до нынешнего дня текущего года, сколько они потратили в прошлом году, на прошлой неделе и т. д. *Внизу:* результаты сгруппированы по **OrderID** и отсортированы по **LastName** и **FirstName**, что сохраняет хороший уровень детализации

Вы уже знаете достаточно для того, чтобы построить запрос, показанный на рис. 7.10. Выполните следующие действии:

1. Создайте новый запрос, выбрав **Создание → Другие → Конструктор запросов** (Create → Other → Query Design).

2. Вставьте нужные вам таблицы с помощью диалогового окна **Добавление таблицы** (Show Table) и затем щелкните мышью кнопку **Закрыть** (Close).

В примере на рис. 7.10 используются таблицы **Customers, Orders** и **OrderDetails.** После добавления этих таблиц программа Access вставит линии объединения между ними на основе связей, установленных в вашей БД.

3. Выберите **Работа с запросами | Конструктор → Показать или скрыть → Итоги** (Choose Query Tools | Design → Show/Hide → Totals).

У каждого поля появится ячейка **Групповая операция.**

4. Добавьте поля, которые хотите использовать, и затем в ячейке **Групповая операция** выберите для каждого поля подходящие группировку или итоговое вычисление.

Поля можно выбирать из любых связанных таблиц. В данном примере применяются следующие поля.

- **OrderID -** это поле используется для группировки результатов. Другими словами, вы хотите подытожить все записи в таблице **OrderDetails** с одинаковым значением поля **OrderID.** Для выполнения этой работы в ячейке **Групповая операция** выберите вариант **Группировка.** (Между прочим, неважно, какое поле вы выберите - **OrderID** в таблице **OrderDetails** или **ГО** в таблице **Orders -** они связаны.)
- **OrderTotal -** это вычисляемое поле, использующее выражение [Price] * [Quantity] для перемножения двух полей из таблицы **OrderDetails.** В результате получится итог для отдельной строки заказа. Программа Access суммирует эти строчные итоги для получения общего итога, поэтому задайте в ячейке **Групповая операция** вариант **Sum.** В поле **OrderTotal** включено условие отбора >=25, скрывающее любые заказы с общей стоимостью ниже $25.
- **LastName** и **FirstName -** эти поля идентифицируют клиента, сделавшего заказ. Но здесь есть хитрость. Для отображения в итоговом запросе любого поля нужно вставить для него вычисление (как для поля **OrderTotal)** или использовать его для группировки (как поле **OrderID).** Это означает, что вы должны установить в ячейке **Групповая операция** вариант **Группировка** для обоих полей. Эта установка на самом деле не будет оказывать никакого влияния, поскольку каждый заказ делается *всегда* одним клиентом. (Другими словами, вы никогда не найдете в таблице **OrderDetails** группу записей из одного заказа, но для разных клиентов. Это просто невозможно.) В результате программа Access не выполнит никакой группировки полей **LastName** и **FirstName,** они просто будут отображаться рядом с каждым заказом.

Примечание

Этот трюк с группировкой немного странный, но широко применяется в итоговых запросах. Просто запомните, что программа Access создает самые маленькие доступные ей группы. Если вы хотите сгруппировать только по клиентам (так можно посмотреть, сколько потратил каждый клиент), нужно только удалить группировку в поле **OrderID** и установить ее в поле **CustomerID.** Или если вы хотите подсчитать объем продаж конкретного товара, удалите всю информацию о клиенте, сгруппируйте данные по полю **ProductID** и затем вставьте дополнительные поля из таблицы **Products,** которые вы хотите видеть на экране (например, **ProductName** и **Description).**

5. Теперь можно выполнить запрос.

7.4. *Параметры запроса*

Параметры запроса - секретное оружие программы Access. Они позволяют создавать сверхгибкие запросы за счет умышленного пропуска одной или нескольких порций информации. При каждом запуске запроса Access запрашивает у вас пропущенные значения. Эти недостающие значения и называют *параметрами запроса.*

Обычно параметры запроса применяют в условиях отбора. Допустим, вы хотите вывести на экран список клиентов, живущих в конкретном штате. Можно создать целый набор запросов, таких как **New York Customers, CaliforniaCustomers, Ohio Customers** и т. д. Если вас

на самом деле интересуют только несколько штатов, такой подход имеет смысл. Но если нужно работать с любым и каждым штатом, лучше создать один запрос, использующий параметр

для задания штата. Когда запрос выполняется, вы в определенный момент вводите нужный штат.

Для создания запроса с параметрами выполните следующие действия:

1. Создайте новый запрос, выбрав на ленте **Создание → Другие → Конструктор запросов** (Create → Other → Query Design).

2. Из диалогового окна **Добавление таблицы** (Show Table) вставьте нужные вам таблицы и щелкните мышью кнопку **Закрыть** (Close).

Б данном примере используется таблица **Customers.**

3. Выберите Работа с запросами | Конструктор → Показать или скрыть → Параметры (Choose Query Tools | Design → Show/Hide → Parameters).

На экране появится диалоговое окно **Параметры запроса** (Query Parameters).

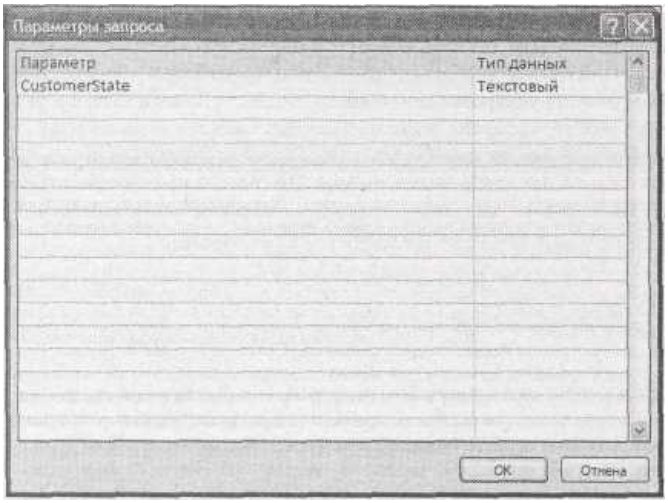


Рис. 7.11. Можно определить столько параметров, сколько нужно. В окне задан один параметр, названный **CustomerState** и содержащий текст

4. Выберите имя и тип данных для вашего параметра (рис. 7.11).

Вы можете использовать любое понравившееся вам имя (но не применяйте имя, которое используется для обозначения поля в вашем запросе). Тип данных должен соответствовать типу данных поля, для которого используется параметр. Тип данных задается выбором одного из вариантов в раскрывающемся списке. Самые распространенные варианты: **Текстовый, Целый, Денежный** и **Дата/время.**

5. Щелкните мышью кнопку **ОК** для закрытия окна **Параметры запроса.**

Теперь можно ссылаться на параметр по имени так же, как вы ссылаетесь на поле в своем запросе. Например, можно добавить следующее условие отбора для поля **State:**

[CustomerState]

Убедитесь, что вы не забыли вставить квадратные скобки, чтобы программа Access знала, что вы не пытаетесь ввести фрагмент текста.

Во время выполнения запроса Access откроет диалоговое окно **Введите значение параметра** (Enter Parameter Value) для ввода конкретного значения (рис. 7.12). Введите интересующий вас штат и щелкните мышью кнопку **ОК**. Программа использует ваше значение для отбора в поле **State.**

Рис. 7.12. При каждом выполнении запроса можно ввести другой штат. В данном случае будут отобраны клиенты из штата Нью-Йорк

Подсказка

Несмотря на то, что такая возможность есть, лучше не применять несколько параметров в одном запросе. Когда запрос выполняется, программа Access отображает отдельное диалоговое окно **Введите значение параметра** для каждого значения. Если у вас несколько параметров, вам

придется сопровождать выполнение запроса щелчками мышью в назойливых диалоговых окнах.

Практических решений, использующих запросы с параметрами, великое множество. Вы можете приспособить запрос о годовом объеме продаж для выполнения с выбранным вами годом. Можно сотворить аналогичное чудо и создать один запрос для отображения объема продаж за любой месяц.

Однако параметры запроса не следует применять для решения повседневных задач, касающихся ввода данных (таких как обновление единственной записи о клиенте). Формы, которые вы начнете создавать в *части IV,* предоставляют гораздо больше возможностей для просмотра и корректировки информации.

8. Глава 8. Запросы, обновляющие записи

Запросы больше всего известны своей способностью отображать небольшие подмножества больших объемов информации. Этот тип запроса известен как *запрос на выборку,* и именно его вы изучали в предыдущих двух главах.

Многие приверженцы Access не знают, что у запросов есть другое назначение. Их можно использовать не только для поиска информации, но и для *изменения* данных. Запросы, оказывающие более сильное действие, будь то удаление, обновление или добавление записей, называют *запросами на изменение* (action query).

8.1. *О запросах на изменение*

Запросы на изменение не так полезны как запросы на выборку, поскольку в них гораздо меньше гибкости. Идеальный запрос создается единожды и повторно используется снова и снова. Запросы на выборку соответствуют этому определению, поскольку часто требуется обзор данных одного и того же сорта (заказы прошлой недели, самые ходовые товары, размеры классов и т. д.). Запросы на изменение коварней, поскольку они вносят *необратимые* изменения.

В большинстве случаев, изменение - одноразовое действие, поэтому нет оснований связываться с запросом, который повторно вносит одно и то же изменение повсеместно. И даже если какие-то данные нужно изменять регулярно (например, цены товаров или уровни складских запасов), задаваемые каждый раз реальные значения не одни и те же. В результате невозможно создать запрос на изменение, который бы вносил это изменение в автоматическом режиме.

Прежде чем вы отложите эту главу и займетесь чем-то более интересным, важно рассмотреть некоторые ситуации, в которых применение запросов на изменение удивительно удобно. Запросы на изменение хороши, если у вас есть следующие типы задач.

■ *Пакетные задачи, которые вы хотите выполнять многократно.* Некоторые задачи могут повторяться регулярно. Если нужно копировать большое число записей из одной таблицы в другую, удалять порцию старых данных или обновлять поле статуса в группе записей, и такие задачи вы вынуждены выполнять снова и снова, запросы на изменение - замечательное, экономящее время средство.

■ *Сложные или трудоемкие задачи, влияющие на большое количество записей.* Любая таблица время от времени нуждается в незначительной реорганизации. Вы можете решить, что пора поднимать цены на 15% или вы установили, что все записи, связанные с клиентом 403, на самом деле должны указывать на клиента 404. Это одноразовые задачи, но они воздействуют на большое количество записей. Для того чтобы расправиться с ними, вам придется потратить много времени на исправление листа данных - или же можно создать новый запрос на изменение, который внесет исправление гораздо эффективнее. Когда запрос сделан, решите, удалять его или сохранить на случай, если вы захотите откорректировать его и повторно применить позже.

■ *Задачи, зависящие от единственной порции информации, которую вы предоставляете при каждом выполнении запроса.* Можно создать запрос на изменение, также содержащий параметры и позволяющий задавать важные значения каждый раз, когда выполняется запрос. (О параметрах запроса *см. разд. "Параметры запроса " главы 7,)* Используя параметры запроса, можно превратить относительно жесткий запрос (который удаляет конкретную запись) в более гибкий (удаляющий любую выбранную запись).

8.1.1. *Тестирование запросов на изменение (с осторожностью)*

В плохих руках запросы на изменение - не что иное, как высокотехнологичный способ навредить себе. Они фиксируют изменения (обычно во множестве записей), и после применения изменений вы не можете их отменить. Некоторые поклонники БД вообще избегают запросов на изменение.

Если вы все же решили применять запросы на изменение (и есть множество полезных трюков, которые молено в них использовать), следует принять должные меры предосторожности. Важнее всего перед применением запроса на изменение сделать резервное копирование БД! Этот шаг особенно важен при создании нового запроса на изменение, потому что он не всегда формирует результат, который вы ждете. Для создания резервной копии можно скопировать ваш файл с

расширением accdb (как любой другой файл; один из способов- щелчок по нему правой кнопкой мыши и выбор команды **Копировать).** Если же вы не хотите связываться с Проводником Windows, можно создать резервную копию, не покидая программы Access, с помощью последовательности **Office → Управление → Резервная копия базы данных** (Office → Manage → Back Up Database) *(см. разд. "Создание резервных копий" главы 1).*

Подсказка

Всегда легче сделать резервную копию, чем устранять последствия изменений, оставленных разбушевавшимся запросом на изменение.

Резервные копии незаменимы при устранении неисправностей, но неплохо не допускать ошибок с самого начала. Один из безопасных способов - начать с запроса на выборку. В этом случае вы можете убедиться в том, что запрос отбирает нужные записи, прежде чем сделать следующий шаг и преобразовать его в запрос на изменение (выбрав один из типов запросов на изменение в группе на ленте **Работа с запросами │ Конструктор → Тип запроса** (Query Tools │ Design → Query Type)).

Семейство запросов на изменение

В программе Access есть четыре типа запросов на изменение:

■ *запрос на обновление* изменяет значения в одной или нескольких записях;

■ *запрос на добавление* выбирает одну или несколько записей и вставляет их в существующую таблицу;

■ *запрос на создание таблицы* выбирает одну или несколько записей и создаст для них новую таблицу;

■ *запрос на удаление* удаляет одну или несколько записей.

В следующих разделах мы попробуем создать запросы всех этих типов.

8.2. *Запросы на обновление*

Запрос на обновление находит некоторые записи и затем изменяет их. Обычно изменения ограничиваются одним полем, но программа Access разрешает корректировать столько полей, сколько нужно. У вас также есть некоторая свобода в способе реализации обновления. Простейший вариант - ввести совершенно новое значение в поле. Можно создать запрос, который перемещает все товары из одной категории в другую с помощью ввода нового значения в поле **CategoryID.** Другой вариант - изменение текущих значений в поле с помощью выражения (специальная формула БД, способная выполнять разнообразные вычисления). Можно повысить цены на 10% или добавить неделю к сроку завершения для всех невыполненных проектов.

Подсказка

Если вам нужно выполнить очевидное одноразовое обновление, может быть, предпочтительней воспользоваться поиском и заменой на листе данных *(см. разд. "Поиск" главы 2).* Этот подход предоставляет возможность просмотреть найденные совпадения и решить, заменять каждое из них или нет.

В приведенном далее примере используются таблицы **Products и Products Categories** из БД Boutique Fudge (которая описана в *разд. "Магазин шоколадных изделий" главы* 5). Запрос обновляет все товары в категории Beverages (напитки), повышая цены товаров на 10%. Вы можете самостоятельно выполнить этот пример, загрузив примеры к этой главе со страницы "Missing CD" на Web-сайте **www.missingmanuals.com.**

Для создания запроса на обновление выполните следующие действия.

1. Создайте новый запрос, выбрав **Создание → Другие → Конструктор запросов** (Create → Other → Query Design).

На экране появится диалоговое окно **Добавление таблицы** (Show Table).

2. Добавьте все таблицы, которые вы хотите включить в ваш запрос, выбрав каждую и щелкнув мышью кнопку **Добавить** (Add) (точно так же, как вы делали, создавая запрос на выборку). По завершении щелкните мышью кнопку **Закрыть** (Close).

Обычно в запросе на обновление используется одна таблица, но если нужна информация из

нескольких связанных таблиц, добавьте их все. Включение в запрос нескольких таблиц создает объединение *(см, разд. "Запросы и связанные таблицы" главы 6).* Операция объединения в запросе на изменение действует так же, как в запросе на выборку - она извлекает информацию из таблицы-родителя и отображает ее рядом с записями из дочерней таблицы.

В данном примере вам потребуются таблицы **Products** и **ProductCategories.**

3. Измените тип запроса на запрос на обновление, выбрав **Работа с запросами | Конструктор → Тип запроса → Тип запроса: обновление** (Query Tools | Design → Query Type →Update).

Столбец со списком свойств полей в нижней части окна изменится, отражая новый тип запроса. Строки **Сортировка** (Sort) и **Вывод на экран** (Show) исчезнут (поскольку они не имеют смысла в запросах на обновление) и для каждого поля, включенного в запрос, появится строка **Обновление** (Update To).

4. Добавьте поле (или поля), которое вы хотите использовать для отбора и задайте для каждого свойство **Условие отбора** (Criteria).

Условия отбора определяют, какие записи отберет программа Access. Поскольку данный запрос - это запрос на обновление, отобранные записи - это записи, в которые будут вноситься изменения.

В данном примере следует использовать поле **CategoryID** или поле **CategoryName.** Если используется поле **CategoryID,** нужно задать значение кода (ID) для вашей категории. Если применяется поле **CategoryName,** можно искать соответствия с помощью названия категории.

Для добавления поля дважды щелкните его кнопкой мыши на схеме в прямоугольнике таблицы так же, как вы делали это в запросе на выборку. Затем задайте условие отбора для значения, с которым вы хотите найти совпадения, как показано на рис. 8.1. Если вы хотите обновить все записи в таблице, то никакого условия отбора не нужно.

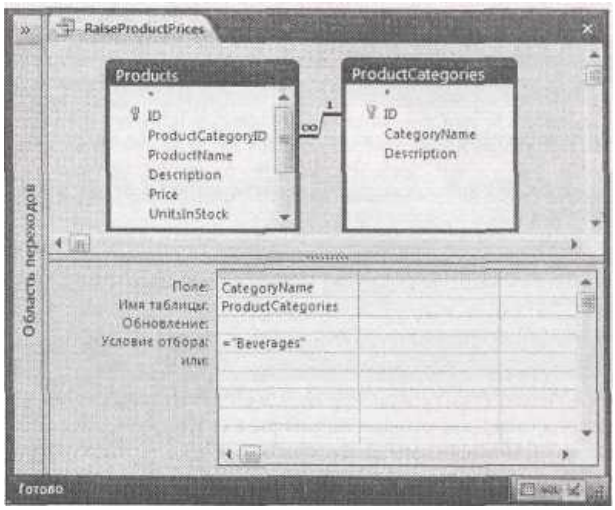


Рис. 8.1. Этот запрос ищет продукты в категории Beverages

5. Добавьте поле (или поля), которое хотите изменить.

В данном примере следует добавить поле **Price,** таким образом, вы сможете изменить цены продуктов.

6. В строке **Обновление** задайте новое значение, которое ваш запрос поместит в каждое поле.

Существуют два способа обновления поля. Можно задать фиксированное значение, введя его в строке **Обновление.** Если выбрать этот подход, программа Access вставит в каждую отобранную вами запись именно это значение.

Можно также применить выражение, которое берет одно или несколько значений из существующих полей и использует их для вычисления нового значения. Вы можете применять все операции и функции, описанные в *главе 7* и предназначенные для обработки текста, чисел и дат. Например, можно использовать следующее выражение в поле **Price** для повышения цен товаров на 10%:

```
[Price]*1.10
```

Подсказка

В выражении обновления может использоваться один или несколько параметров *(см. разд. "Параметры запроса" главы 7).* В этом случае Access запрашивает у пользователя, выполняющего запрос, важную информацию (например, каково процентное изменение цены).

7. Добавьте любые другие поля, которые хотите использовать для подтверждения правильности отбора записей.

Прежде чем выполнять ваш запрос и вносить изменения, выполните предварительный просмотр, который выводит все строки, отбираемые вашим запросом на обновление (и, таким образом, все записи, которые изменятся, когда вы выполните запрос). Для того чтобы убедиться, что ваш запрос отобрал нужные записи, возможно, на листе данных понадобится некоторая дополнительная идентифицирующая информация, например, **ProductName.**

Для того чтобы заставить работать этот предварительный просмотр, нужно применить один формальный прием. Программа Access игнорирует поля, которые вы не собираетесь обновлять. Поэтому если вы хотите добиться вывода на листе данных поля **ProductName,** следует задать что-то в строке **Обновление.** В данном случае используйте значение [ProductName]. Этот шаг заставит программу Access заменить значение в поле **ProductName** текущим значением поля **ProductName.** Другими словами, Access на самом деле ничего менять не будет, но отобразит поле **ProductName** па листе данных в окне предварительного просмотра.

На рис. 8.2 показан законченный запрос на обновление.

8. Щелкните правой кнопкой мыши заголовок вкладки и выберите команду **Режим таблицы** (Datasheet View) для просмотра записей, на которые повлияет ваш запрос (рис. 8.3).

Этот шаг позволит просмотреть строки, которые вы собираетесь изменить, прежде чем запустите выполнение запроса. На листе данных вы увидите все записи, удовлетворяющие вашим условиям отбора, - иначе говоря, все записи, которые вы измените при выполнении запроса, но вы не увидите изменений, которые хотите внести.

Примечание

В обычном запросе на выборку просмотр листа данных и выполнение запроса - равнозначные действия. В запросе на изменение отображение листа данных показывает строки, которые будут изменены, но на самом деле не изменяет их. Выполнение запроса изменяет данные, но не показывает измененные записи.

9. Теперь вернитесь в **Конструктор** (щелкните правой кнопкой мыши заголовок вкладки и выберите Конструктор (Design View)). Если вы уверены в том, что запрос действует правильно, выберите **Работа с запросами | Конструктор → Результаты → Выполнить** (Query Tools | Design → Results → Run) для запуска запроса на обновление и внесения заданных изменений.

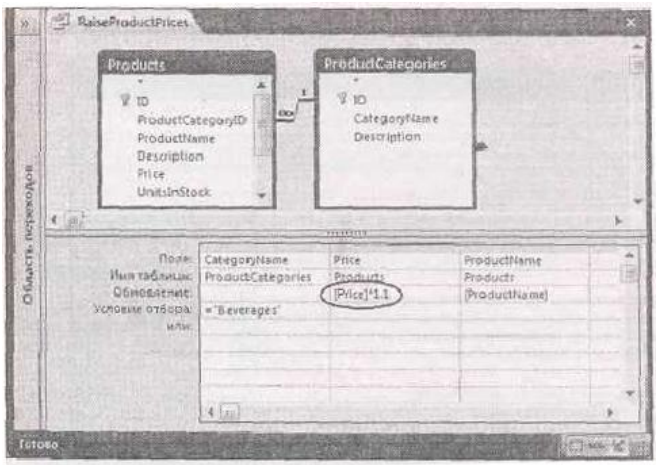


Рис. 8.2. Этот запрос отбирает все товары в заданной категории и повышает их цену на 10%

Рис. 8.3. Здесь показан предварительный просмотр. В нем отображаются все товары в категории Beverages с текущими ценами. Когда вы выполните запрос, именно эти записи изменятся

Помните: перед выполнением этого шага рекомендуется сделать резервное копирование вашей БД.

Когда вы выполняете запрос на изменение, программа Access предупреждает о том, что собирается выполнить изменение БД (рис. 8.4). Щелкните мышью кнопку **Да** (Yes) для внесения изменений.

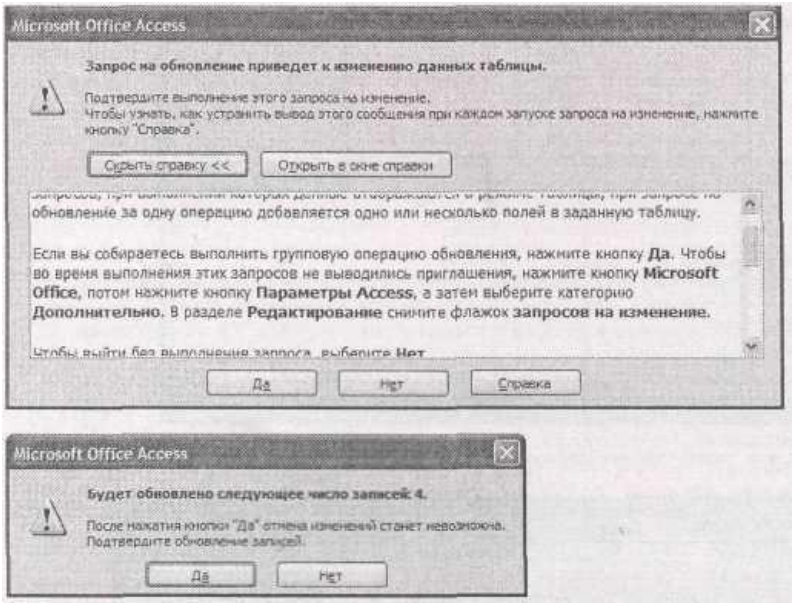


Рис. 8.4. *Вверху:* при каждом выполнении запроса на изменение Access предупреждает о том, что запрос изменит БД. Если вам не нужно это напоминание, выполните действия, перечисленные в этом окне для открытия диалогового окна **Параметры Access** и отключения вывода на экран этого предупреждения. (Сначала нужно щелкнуть мышью кнопку **Нет** для закрытия диалогового окна). *Внизу:* далее Access сообщает о количестве изменяемых записей и дает вам последний шанс отказаться от изменений. Программа всегда предоставляет эту информацию, даже если отключен вывод стандартных предупреждений. Если сейчас щелкнуть мышью кнопку **Да,** Access обновит таблицу

К сожалению, программа Access не показывает измененные записи - она вообще ничего не показывает. Если вас интересует, что же произошло, и вы хотите просмотреть только что измененные записи, у вас один вариант - снова вывести на экран окно предварительного просмотра записей, которые вы только что изменили (щелкнув правой кнопкой мыши заголовок вкладки и выбрав **Режим таблицы).** Этот способ действует до тех пор, пока вы не изменили записи таким образом, что они больше не соответствуют условиям отбора. (Если это произошло,

следует создать новый запрос или просмотреть таблицу для двойной проверки ваших данных.)

10. Для сохранения запроса нажмите комбинацию клавиш <Ctrl>+<S> (или закроите вкладку запроса). При этом придется задать имя запроса.

Используйте имя запроса, четко указывающее на то, что это запрос на изменение. Можно, например, задать имя **UpdateProductPrices** (изменение цен товаров). Запросы на изменение отображаются в области переходов с пиктограммой восклицательного знака. У каждого типа запроса на изменение слегка отличающаяся пиктограмма - для запросов на обновление применяется пиктограмма с карандашом и восклицательным знаком за ним (рис. 8.5).

Если вы не собираетесь повторно использовать свой запрос, может быть, стоит его удалить. Удаление запроса защитит от случайного выполнения вами (или кем-то еще) запроса и внесения нежелательных изменений.

Рис. 8.5. Помните о том, что двойной щелчок по запросу в области переходов запускает его на выполнение. Если вы щелкнули мышью запрос на изменение, например, такой, как выделенный на этом рисунке, то можете изменить или удалить важные данные. (Для открытия запроса на изменение без его запуска щелкните по его имени правой кнопкой мыши и выберите команду **Конструктор)**

Аварийная ситуация.

Когда Access блокирует ваше обновление

Рассмотрим рабочий аспект стратегии программы Access: что происходит, когда вы нажимаете мышью кнопку **Выполнить** (как описано в пункте 9 предыдущего алгоритма), и пет никакой реакции? Не появляются ни предупреждение, ни окно сообщения или ошибки, объясняющие причину сбоя. Лишь в строке состояния, в нижней части окна Access выводится таинственное сообщение, которое любезно информирует о том, что "действие или событие заблокировано режимом отключения" ("The action or event has been blocked by Disabled Mode"). Что все это значит?

Access - по-настоящему параноидальная программа. Она не разрешает вам выполнять некоторые действия до тех пор, пока вы явно не подтвердите, что все в порядке.

Как вы узнали из *разд. "Открытие БД" главы 1,* каждый раз, когда вы открываете вашу БД, Access отображает панель сообщений с сообщением системы безопасности. Вам решать, что делать с этим сообщением. Можно щелкнуть мышью кнопку ? в правом верхнем углу, чтобы полностью скрыть эту панель. В этом случае ваша БД остается в отчасти заблокированном состоянии. Вы можете создавать, изменять и удалять объекты БД, принадлежащие вам, но не можете запускать никакой код или запросы на изменение. (Для повторного вывода на экран панели сообщений и просмотра сообщения системы безопасности выберите на ленте **Работа с таблицами → Показать или скрыть → Панель сообщений** (Database Tools → Show/Hide → Message Bar).)

Другой способ - щелкнуть мышью кнопку **Параметры** (Options) на панели сообщений для отображения диалогового окна **Параметры безопасности Microsoft Office**

(Microsoft Office Security Options). Дальше следует выбрать переключатель Включить **это содержимое** (Enable this content) и щелкнуть мышью кнопку **ОК**. Этот шаг предоставляет программе Access непоколебимую гарантию безопасности вашей БД - другими словами, БД разработана не жующим чипсы хакером в подвале родительского дома. После того как вы

предприняли этот шаг, Access разрешает выполнять запросы на изменение (по крайней мере, до тех пор, пока вы не закрыли БД, не открыли ее снова и не увидели на экране повторно отображенное сообщение центра безопасности).

Если вы устали от многоразового включения БД при каждом ее использовании, есть другое решение. Можно заставить программу Access доверять всем БД в конкретной папке на вашем жестком диске. Этот метод описан в *разд. "Задание надежного расположения " главы 15.*

8.3. *Запросы на добавление*

Запрос на добавление выбирает записи из таблицы и вставляет их в другую таблицу. (С технической точки зрения, добавление - это процесс вставки записей в конец таблицы.)

Создать запрос на добавление можно по ряду причин, но обычно это делается для переноса записей из одной таблицы в другую. Этот метод удобен, если у вас есть повторяющиеся таблицы в разных БД (возможно, разным людям приходится использовать БД на разных компьютерах).

Примечание

После завершения копирования записей в новую таблицу можно продолжить работу с помощью запроса на удаление *(см. разд. "Запросы на удаление" далее в этой главе)* и удалить старые версии.

Запросы на добавление также имеют смысл, если вы работаете со сверхчувствительной базой. В этом случае вы можете вводить данные во временную таблицу для того, чтобы кто-то просмотрел их позже. Когда проверка закончена, можно применить запрос на добавление для переноса записей в реальную таблицу.

Запросы на добавление жестче других типов запросов на изменение. При переносе записей нужно быть уверенным в полной согласованности таблиц.

■ *Типы данных должны быть совместимы.* У выбранных вами полей (в исходной таблице) и заменяемых полей (в конечной таблице) должны быть совместимые типы данных. Но имена полей могут не совпадать. Вы можете сформировать ваш запрос так, что данные из поля **FirstName** помещаются в поле **F_Name** при условии, что у обоих полей текстовый тип данных.

■ *Некоторые поля можно пропускать.* Если в исходной таблице есть поля, которых нет в конечной таблице, не включайте их в свой запрос. Если в конечной таблице есть поля, которых нет в исходной таблице, программа Access оставит их незаполненными или использует значения по умолчанию *(см. разд. "Задание значений по умолчанию" главы 4).* Но если вы пропустите обязательное поле (поле, у которого свойство **Обязательное поле** (Required) имеет значение *Да,* как объясняется в *разд. "Пропущенные значения и пустые строки "главы 4),* вы получите сообщение об ошибке.

■ *Программа Access применяет все обычные правила при добавлении записи.* Вы не можете вставить данные, нарушающие условие на значение *(см. разд. "Целостность на уровне ссылок" главы 5)* и добавить дублирующиеся значения в поле с первичным ключом или уникальным индексом *(см. разд. "Предотвращение дублирования значений с помощью индексов " главы 4).*

■ *Если в конечной таблице есть поле с типом данных* ***Счетчик,*** *не задавайте значение для этого поля.* Access автоматически сгенерирует значение для каждой добавляемой записи.

Примечание

Вы не можете копировать значения типа **Счетчик** в запросе на добавление. Если для идентификационных полей (кодов) применяется тип данных **Счетчик,** у вновь скопированных записей будут значения кодов (ID), отличающиеся от оригиналов.

Программа Access предоставляет и другую возможность, аналогичную запросу на добавление; *запрос па создание таблицы,* который ничем не отличается от запроса на добавление за исключением одного: запрос на создание таблицы *создает* конечную таблицу и затем копирует в нее записи.

8.3.1. *Создание запроса на добавление (или на создание таблицы)*

Приведенные далее действия описывают процесс создания запроса на добавление или на

создание таблицы. Вы переносите записи из таблицы **Contacts** (контакты) БД Marketing.accdb в таблицу **PotentialClients** (потенциальные клиенты) БД Sales.accdb. (Вы можете найти обе БД на странице "Missing CD" на Web-сайте **www.missingmanuals.com.)**

1. Откройте БД-источник.

В данном примере это БД Marketing.accdb, содержащая контактную информацию.

2. Создайте новый запрос, выбрав на ленте **Создание → Другие → Конструктор запросов** (Create → Other → Query Design).

На экране появится диалоговое окно **Добавление таблицы** (Show Table).

3. С помощью этого окна добавьте таблицу-источник, содержащую записи, которые вы хотите скопировать. Затем для закрытия окна щелкните мышью кнопку **Закрыть** (Close).

В данном примере используется таблица **Contacts.**

4. Измените тип запроса на запрос на добавление, выбрав на ленте **Работа с запросами | Конструктор → Тип запроса → Тип запроса: добавление** (Query Tools | Design → Query Type → Append) (или выберите **Работа с запросами | Конструктор → Тип запроса → Тип запроса: создание таблицы** (Query Tools | Design → Query Type → Make Table) для превращения его в запрос на создание таблицы).

Конечная таблица (таблица **PotentialClients** в БД Sales.accdb) уже существует. По этой причине применяется запрос на добавление вместо запроса на создание таблицы.

Когда вы измените тип запроса на запрос на добавление или на создание таблицы, программа Access попросит указать конечную таблицу (место, куда вы будете копировать записи), как показано на рис. 8.6.

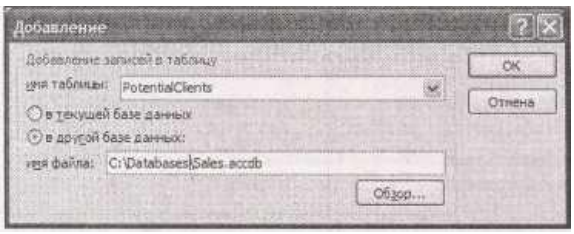


Рис. 8.6. Программа Access хочет знать, куда вы собираетесь перенести копируемые записи. Таблицу можно выбрать из удобного раскрывающегося списка. Если вы копируете данные из одной БД в другую, выберите переключатель **в другой базе данных,** щелкните мышью кнопку **Обзор...** для выбора файла БД и затем кнопку ОК

5. Если вы хотите переместить записи в другую БД, выберите переключатель **в другой базе данных,** затем нажмите кнопку **Обзор...** Укажите файл вашей БД и нажмите кнопку ОК, чтобы подтвердить ваш выбор.

Вы перемещаете записи в БД Sales.accdb.

Если вы планируете повторное использование нового запроса, не меняйте место хранения конечной БД. Если конечный файл переместить в другую папку или на другое устройство (или переименовать его), программа Access не сможет найти его во время выполнения запроса и выдаст сообщение об ошибке.

6. В поле **имя** таблицы (Table Name) укажите имя таблицы, в которую вы хотите перенести записи.

Если создается запрос на добавление, выбранная таблица должна где-то храниться - либо в файле БД, либо в другом доступном вам месте. Ее можно выбрать из раскрывающегося списка **имя таблицы.**

Если вы формируете запрос на создание таблицы, нужно ввести имя таблицы для новой таблицы, и программа Access создаст ее во время выполнения запроса. В данном примере вы переносите записи в таблицу **PotentialCIients.**

7.Щелкните мышью кнопку ОК для того, чтобы закрыть диалоговое окно **Добавление** или **Создание таблицы.**

8. Теперь добавьте поле (или поля), которое вы хотите скопировать из таблицы-источника.

Напоминаю о том, что вы не должны копировать все поля. В данном примере нужно добавить только поля **FirstName** и **LastName.**

9. Если создается запрос на добавление, вставьте имена полей конечной таблицы в строку **Добавление** (Append To).

В этом примере задайте в поле **Добавление** для **FirstName** имя **F_Name.** В этом случае программа Access скопирует информацию из поля **FirstName в** таблице-источнике в поле **F_Name** конечной таблицы (рис. 8.7). Аналогично задайте в поле **LastName** для добавления поле **L_Name.**

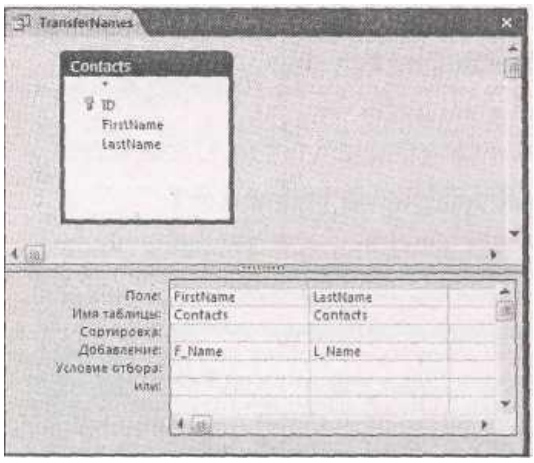


Рис. 8.7. Данный запрос на добавление переносит информацию из таблицы **Contacts** в БД Marketing в таблицу **PotentialClients** БД Sales. Поскольку в обеих таблицах используются поля ID с типом данных **Счетчик,** номера ID в скопированных записях будут отличаться от номеров ID в исходных записях. (Если вас это не устраивает, нужно скопировать номера ID типа **Счетчик** таблицы **Contacts** в обычный числовой столбец таблицы **PotentialClients -** такой, в котором не используется тип данных **Счетчик.)**

10. Если вы хотите скопировать только некоторые записи из таблицы-источника, задайте необходимые условия отбора.

Как и в любых других секциях программы Access, эти условия отбора определяют, какие записи копируются из таблицы-источника. Для задания условия заполните строку **Условие отбора** соответствующего поля таблицы.

Если вы добавляете условие отбора в запрос на добавление, но не хотите копировать значение этого поля в конечную таблицу, оставьте пустым поле **Добавление.**

Если поле с условием отбора применяется в запросе на создание таблицы, но вы не хотите копировать его значение в новую таблицу, сбросьте флажок **Вывод на экран** для этого поля.

11. Щелкните правой кнопкой мыши заголовок вкладки и затем выберите команду **Режим таблицы** для просмотра строк, на которые воздействует ваш запрос.

Этот шаг позволяет просмотреть строки, которые вы собираетесь копировать.

12. Если вы убедились в том, что все верно, вернитесь в **Конструктор** и выберите на ленте **Работа с запросами j Конструктор → Результаты → Выполнить** для переноса ваших записей (Query Tools │ Design → Results → Run).

Программа Access предупредит вас об изменении, которое собирается сделать. Щелкните мышью кнопку Да для копирования записей. Access не выведет на экран скопированные записи - для того, чтобы проверить их, нужно просмотреть лист данных с конечной таблицей.

В настоящий момент у вас одни и те же записи в двух местах - в таблице-источнике и в конечной таблице. Мы сможете продолжить работу и с помощью запроса на удаление очистить таблицу-источник, как описано в *разд. "Запросы на удаление" далее в этой главе,*

13.Для сохранения запроса нажмите комбинацию клавиш <Ctrl>+<S> (или закройте вкладку запроса). Вам нужно задать имя запроса.

Если вы не собираетесь повторно использовать запрос, подумайте о его удалении.

8.3.2. *Получение начальных значений типа Счетчик, отличных от 1*

Ведущие специалисты Access применяют запросы в одном из самых изощренных искусственных приемов: замене в поле таблицы с типом **Счетчик** начального значения числом, отличающимся от 1.

Как вы узнали в *главе 2,* программа Access всегда генерирует значения типа **Счетчик,** начиная с 1. (Единственное исключение - применение случайных чисел или кодов репликаций, два редких варианта, описанных в *разд. "Применение поля типа Счетчик без раскрытия реального размера вашей таблицы" главы 2.)* Но существует множество причин, вызывающих желание изменить такое поведение программы. Например, компании Boutique Fudge хочется начать нумерацию своих клиентов с 1000, а номеров товаров - с 5000, или начать нумеровать свои заказы с 10 000. Эти схемы нумерации часто облегчают бухгалтерский учет. Они позволяют сохранять постоянным количество цифр в значениях типа **Счетчик,** помогают разделить коды в двух разных таблицах и не смущаться, сообщая клиенту о том, что он сделал заказ номер 1.

К счастью, существует (немного неуклюжий) способ обмануть систему и заставить Access начать отсчет с любого нужного вам числа. Для того чтобы сделать то, что вы не можете сделать сами, применяется запрос на добавление. Просто вставляется запись с заданным значением типа **Счетчик.** После того как запись создана, программа Access наращивает значения, начиная со вставленного вами значения. Таким образом, если вы добавили запись типа **Счетчик** со значением 999, Access присвоит следующей записи значение 1000 и т. д.

Вот как это делается.

1. Создайте новую таблицу **(Создание → Таблицы → Конструктор** (Create → Tables → Table Design)).

Эта таблица будет храниться всего несколько минут.

2. Добавьте одно поле. Присвойте ему то же имя, что и у поля с типом данных Счетчик в таблице, которую вы пытаетесь изменить.

Обычно у него имя **Код** (ID).

3. Измените тип данных поля на **Числовой** (вместо **Счетчик)** и убедитесь в том, что размер поля - *Длинное целое* (Long Integer) (стандартный выбор).

4. Щелкните правой кнопкой мыши заголовок таблицы и выберите **Режим таблицы.**

Сохраните таблицу, когда программа Access напомнит об этом, но не беспокойтесь о ее имени, имя **Табляца1** вполне подходит. Когда Access предложит создать первичный ключ, щелкните мышью кнопку **Нет.**

5. В **Режиме таблицы** введите в поле с типом данных **Числовой** временной таблицы значение, на 1 меньшее того, которое вы хотите использовать в качестве начального в поле с типом данных **Счетчик.**

Если вы хотите начать со значения 100 в поле с типом **Счетчик,** введите в поле с типом **Числовой** значение 99. Закройте таблицу.

6. Создайте новый запрос **(Создание → Другие → Конструктор запросов** (Create → Other → Query Design)).

В появившемся диалоговом окне **Добавление таблицы** (Show Table) выберите созданную вами временную таблицу **(Таблица!.)** и щелкните мышью кнопку **Закрыть** (Close).

7. Выберите на ленте **Работа с запросами J Конструктор → Тип запроса → Тип запроса:**

добавление (Query Tools | Design → Query Type → Append) для изменения типа запроса на запрос на добавление.

Когда программа Access запрашивает, в какую таблицу вы хотите добавить запись, выберите таблицу с полем типа **Счетчик,** значения которого вы хотите попробовать изменить.

8. Дважды щелкните кнопкой мыши поле, которое вы добавили в вашу таблицу (например Код (ID)).

Программа Access задаст в строке **Добавление** то же имя, это как раз то, что надо.

9. Выберите на ленте **Работа с запросами | Конструктор → Результаты → Выполнить** (Query Tools | Design → Results →Run).

Щелкните мышью кнопку **Да,** когда Access предупредит вас о том, что собирается добавить запись.

10. Откройте таблицу, которую вы только что обновили, и удалите только что вставленную запись.

Начиная с этого момента и далее, значения типа **Счетчик** будут увеличиваться, начиная с добавленного значения.

11. Удалите временную таблицу, созданную в пункте 1, поскольку она вам больше не нужна.

У этого метода есть несколько ограничений. А именно, если у вашей таблицы строгие правила верификации - например, у одного или нескольких полей в свойстве **Обязательное поле** задано

значение *Да* - Access не разрешит вам вставить новую запись с помощью запроса на добавление. В этой ситуации нужно либо отключить правила верификации (временно установив для всех полей в свойстве **Обязательное поле** значение *Нет),* либо добавить обязательные поля с корректными значениями в вашу временную таблицу.

8.4. *Запросы на удаление*

Запросы на удаление - самые простые и самые опасные из всех типов запросов на изменение. Запрос на удаление действует во многом так же, как запрос на выборку: вы задаете ряд условий отбора, и затем программа Access находит соответствующие записи в таблице. Но запросы на удаление не просто отображают записи, а *удаляют* их из вашей БД.

Примечание
Дважды подумайте, прежде чем удалять что бы то ни было. Старая информация вам может понадобиться для отчетов или анализа. В *разд. "Редактирование таблицы" главы 1* объясняется, почему.

Запросы на удаление незаменимы при одновременном удалении большого количества записей после завершения перемещения их в другую таблицу. В примере с запросом на добавление, описанном ранее в этой главе, вам, возможно, понадобится способ удаления исходных записей после копирования их в новую таблицу. Запрос на удаление отлично подходит для этого.

Для создания запроса на удаление выполните следующие действия.

1. Создайте новый запрос **(Создание → Другие → Конструктор запросов** (Create → Other → Query Design)).

2. В появившемся диалоговом окне **Добавление таблицы** выберите таблицу, содержащую записи, которые вы хотите удалить. Затем щелкните мышью кнопку **Закрыть** для закрытия окна.

3. Измените тип вашего запроса на запрос на удаление, выбрав **Работа с запросами | Конструктор → Тип запроса → Тип запроса: удаление** (Query Tools | Design → Query Type → Delete).

В списке свойств полей исчезнут строки **Сортировка** и **Вывод на экран** и появится поле Удаление (Delete).

4. Добавьте поля, которые вы хотите использовать для отбора, и задайте условия отбора.

Ваши условия отбора определяют, какие записи удаляются, поэтому задавайте их очень аккуратно. Если вы не включите никаких условий отбора, Access удалит все записи при выполнении вашего запроса.

5. Добавьте любые другие поля, с помощью которых вы хотите проверить при предварительном просмотре на листе данных правильность отбора записей.

Очень важно убедиться в том, что вы удаляете только те записи, которые хотели удалить, У запросов на удаление есть чудесное свойство, которое поможет вам идентифицировать каждую запись, прежде чем вы выполните реальную операцию удаления. Для его применения щелкните дважды кнопкой мыши звездочку (*) в списке полей таблицы. Значение в строке **Удаление** автоматически изменится на **Из** (From), означающее, что данная информация не используется как часть условия отбора - напротив, она применяется для отображения списка предназначенных для удаления записей в ваших окнах предварительного просмотра.

На рис. 8.8 показан окончательный вариант запроса на удаление.

6. Щелкните правой кнопкой мыши заголовок вкладки и затем выберите **Режим таблицы** для того, чтобы увидеть строки, на которые повлияет ваш запрос.

Этот шаг позволит предварительно просмотреть строки, которые вы собираетесь удалить. Когда применяется звездочка (*), на экран выводится вся информация, относящаяся к каждой записи.

7. Если вы уверены, что получена корректная информация, вернитесь в **Конструктор** и затем выберите на ленте **Работа с запросами | Конструктор → Результаты → Выполнить** (Query Tools | Design → Results → Run) для удаления записей.

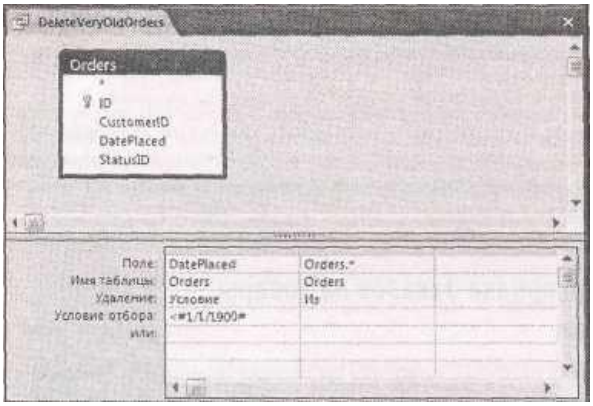


Рис. 8.8. Этот запрос удаляет записи со старыми заказами, первое поле в запросе определяет условие отбора (заказы с датами в поле DatePlaced (дата размещения) наступившими до 1900 г.). Второе поле (*) - сокращенная ссылка, позволяющая увидеть в окне предварительного просмотра все поля для того, чтобы можно было внимательно изучить данные, которые вы собираетесь удалять

Программа Access предупредит вас об изменении, которое собирается сделать. Щелкните мышью кнопку ОК, если хотите навсегда удалить записи.

8. Если хотите сохранить запрос, нажмите комбинацию клавиш <Ctrl>+<S> (или закройте вкладку запроса). Вы должны задать имя вашего запроса.

Если вы не собираетесь использовать запрос на удаление повторно, не сохраняйте его. Это опасное средство, которое не стоит оставлять под рукой.

Малоизвестная или недооцененная возможность.
Скрытие запроса

Если вы хотите сохранить запрос под рукой для последующего использования, по считаете это слишком опасным, программа Access предоставляет более безопасную возможность. Вы можете скрыть запрос так, что он не будет выводиться в области переходов. В этом случае вы не воспользуетесь им по небрежности. Тот, кому он понадобится, вынужден будет отыскать его.

Для того чтобы скрыть запрос, щелкните его в области переходов правой кнопкой мыши и выберите команду **Скрыть в данной группе** (Hide in this Group). Запрос незаметно исчезнет из поля зрения.

Единственный способ вернуть в область переходов скрытый объект БД - щелкнуть правой кнопкой мыши заголовок области переходов (что-нибудь похожее на **Все таблицы)** и выбрать команду **Параметры переходов** (Navigation Options). Затем можно установить флажок **Показывать скрытые объекты** (Show Hidden Objects). Когда этот флажок установлен, скрытые объекты видны в области переходов, но они отображаются светло-серым цветом. Для возврата объекту нормального состояния видимости, щелкните его правой кнопкой мыши и выберите команду **Показать в этой группе** (Unhide in this Group).

Убедитесь в том, что вы не злоупотребили скрытием. Если это так, вы вынудите других включить режим Показывать скрытые объекты, который сделает видимыми и пригодными к использованию все запросы.

Если вас все еще беспокоит присутствие опасного запроса в вашей БД, рассмотрите возможность переноса запроса на изменение в совершенно отдельный файл БД и не разрешайте другим пользователям открывать этот файл. В *разд. "Подготовка вашей базы данных" главы 18* приведена дополнительная информация о разделении БД на несколько файлов.

8.5. *Учебный пример: маркировка заказов на товары, которых нет в наличии*

У компании Boutique Fudge есть проблема. Компания производит свои товары небольшими партиями, и они быстро распродаются. Например, если их источник импортного дуриана иссякает, то же происходит и со всемирно известным продуктом Mocha Malaysian Espresso Milk (малазийский кофе Мокко эспрессо с молоком).

Однако активные покупатели продолжают заказывать товары, которых нет на складе. В

конечном счете они получают их, но заказ продукта, которого нет на складе, может быть более долгим, одиноким и забытым в БД на недели. Компания Boutique Fudge могла бы уберечь клиента от неразберихи (не говоря уже о жажде), если бы смогла учесть клиентов, заказавших товары, которых нет в наличии, и предупредила их о необходимости ожидания.

Разработчики БД в компании Boutique Fudge подумали над этой проблемой и решили, что им нужно поле в таблице **Orders,** позволяющее пометить заказы, находящиеся в состоянии ожидания из-за отсутствия на складе ингредиентов. Решено было использовать поле с Логическим типом данных, названное **OnHold** (в ожидании). В этом случае, когда рабочие склада подготавливают заказ, они могут сэкономить время, игнорируя заказы, находящиеся в состоянии ожидания. А отдел обслуживания клиентов может отследить клиентов, поместивших эти заказы, и объяснить им причину задержки.

Пока в этом примере нет ничего нового. Но в нем есть одна хитрость: компания Boutique Fudge хочет автоматизировать процесс задания значений в поле **OnHold.** Она рассчитывает выполнить запрос, который проверит поле **UnitsInStock** (единиц на складе) в таблице **Products** и затем установит значение *Да* в поле **OnHold** для всех находящихся в работе заказов, включающих товары, которых "нет в наличии" (out-of-stock). Теперь, когда вы научились создавать запросы на изменение, вы можете рассмотреть эту хитроумную головоломку.

Как и многие проблемы в программе Access, эту задачу можно решить шаг за шагом. В данном примере вы решите ее созданием двух отдельных запросов:

- запроса на выборку, отбирающего заказы, содержащие продукты, которых нет в наличии;
- запроса на изменение, обновляющего поле **OnHold** для продуктов, которых нет в наличии.

8.5.1. *Поиск продуктов, которых нет в наличии*

Первый шаг - поиск всех заказов, включающих продукты, которых в данный момент нет на складе.

Для этого нужен запрос, содержащий две таблицы:

- **Products**, т. к. в ней есть поля с уровнями запасов;
- **OrderDetails,** потому что она сообщает, в какие заказы входят конкретные продукты.

В данном случае таблица **OrderDetails -** дочерняя таблица, а **Products -** родительская. В результате, когда вы выполните этот запрос, то действительно получите список записей таблицы **OrderDetails,** снабженных информацией о продукте.

После того как создан запрос с нужными таблицами, необходимо добавить в него подходящие поля.

- **UnitsInStock** (поле из таблицы **Products).** Это поле сообщает о наличии продукта на складе. Для поиска отсутствующих компонентов заказа задайте свойство поля **Условие отбора** равным 0;
- **OrderID** (поле из таблицы **OrderDetails).** Это поле идентифицирует заказы с отсутствующими ингредиентами.

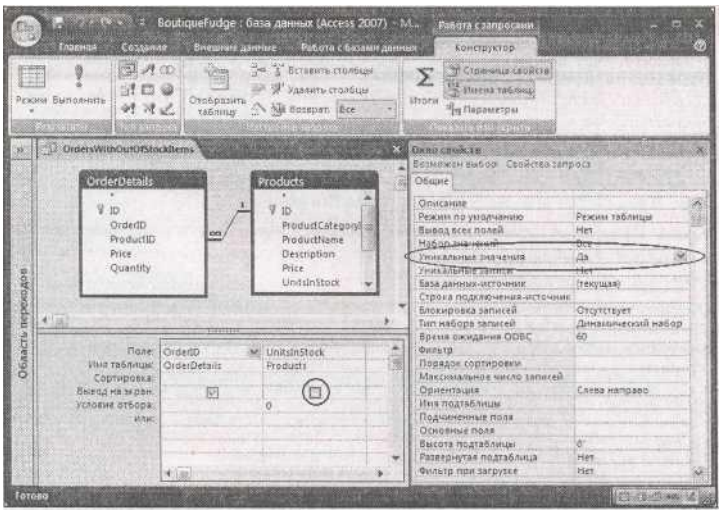


Рис. 8.9. Этот запрос (названный **OrdersWithOutOfStockItems** (заказы с отсутствующими на складе ингредиентами)) генерирует список кодов продуктов, которых нет в наличии. Поле

UnitsInStock (единиц на складе) используется в нем для отбора, но не включено в результат запроса (обведенный флажок **Вывод на экран** сброшен). Для исключения повторения одних и тех же заказов (если в них содержится несколько продуктов, которых нет в наличии) свойству запроса **Уникальные значения** (также обведенному) присвоено значение *Да*

Одна проблема все еще остается. Когда этот запрос выполняется, один и тот же ID может выводиться много раз, поскольку извлекается список отсутствующих на складе продуктов, а в одном заказе их может быть несколько. (Вы, конечно же, не хотите, чтобы сотрудники отдела по обслуживанию клиентов звонили клиенту несколько раз, не так ли?) Самый легкий способ решения этой проблемы - сообщить программе Access о необходимости игнорировать дубликаты в вашем запросе, выполнив следующие действия.

1. Выберите на ленте **Работа с запросами | Конструктор → Показать или скрыть → Страница свойств** (Query Tools | Design → Show/Hide → Property Sheet).

В правой части окна программы Access появится область **Окно свойств** (Property Sheet) с низкоуровневыми параметрами запроса.

2. Щелкните кнопкой мыши на пустом месте в зоне **Конструктора запросов** (например, рядом с одним из прямоугольников таблиц). В верхней части области **Окно свойств** появится строка: **Возможен выбор: Свойства запроса** (Selection Type: Query Properties).

3. В области **Окно свойств** измените значение параметра **Уникальные значения** (Unique Values) с *Нет* на *Да.*

Теперь в результатах запроса каждый заказ будет появляться только один раз. На рис. 8.9 показан законченный запрос.

8.5.2. *Перевод заказов в режим ожидания*

Далее нужно выполнить запрос, который изменяет все вызывающие проблемы заказы. Этот запрос должен отыскать все записи заказов, найденные запросом **OrdersWithOutOfStockltems,** и изменить их.

Какое решение? Конечно же запрос на обновление, подобный описанному ранее в этой главе. В нем следует использовать таблицу **Orders** и два поля:

- поле **ID** применяется для поиска записей заказов, которые вы хотите откорректировать;
- поле **OnHoId** меняется на *Да* для перевода заказа в режим ожидания.

Вы уже знаете достаточно для того, чтобы добавить оба поля в запрос и заполнить свойство **Обновление** поля **OnHold** (значением Да). Труднее всего найти нужные записи. Ясно, что необходимо найти заказы, содержащие одно из значений **ID**, которые вы отыскали в запросе **OrdersWithOutOfStockltems.** Но как использовать этот запрос в запросе на обновление?

Для решения этой задачи необходимо применить пару новых хитростей в условии отбора. Во-первых, нужно использовать ключевое слово In, проверяющее, попадает ли значение в заданный список значений. Далее приведен пример действующего условия с ключевым словом In:

```
In  (14,15,16)
```

Это условие фильтрации отбирает любые записи с кодами 14,15 или 16-

Ясно, что вводить вручную все значения ID очень трудоемко. Гораздо разумнее еще раз выполнить работу, проделанную во время создания запроса **OrdersWithOutOfStockltems.**

Для воплощения этой идеи вам придется воспользоваться еще одним необычным средством: подзапросом.

Подзапрос - это запрос, встроенный внутрь другого запроса. При написании подзапроса вам придется использовать язык SQL, с которым вы познакомились в *главе 6.* Начать следует со слова SELECT, затем перечислить поля, которые вы хотите получить, за ними вставить слово FROM и завершить все именем таблицы или запроса, которые используются. Далее приведена команда на SQL для запроса на выборку, извлекающего все **ID** (коды) заказов из запроса **OrdersWithOutOfStockltems:**

```
SELECT OrderID  FROM OrdersWithOutOfStockltems
```

Теперь, когда у вас есть оба нужных вам компонента, следует соединить их вместе в одном суперэлегантном условии отбора. Далее приведено окончательное выражение:

```
In  (SELECT OrderID FROM OrdersWithOutOfStockltems)
```

Поместите это условие отбора в поле **ID**. Оно получит все коды (**ID**) проблемных заказов с помощью запроса **OrdersWithOutOfStockltems** и затем сравнит их с полным набором записей в

таблице **Orders.** Окончательный запрос на изменение показан на рис. 8.10.

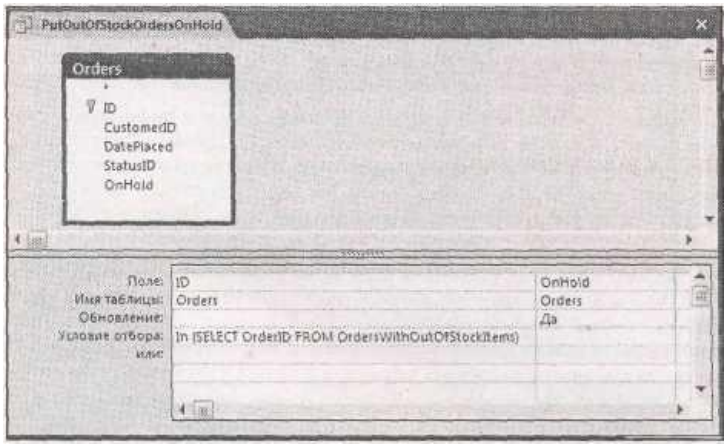


Рис. 8.10. Этот запрос на обновление (**PutOutOfStockOrdersOnHold** (перевод заказов отсутствующими продуктами а режим ожидания)) гарантирует, что клиенты компании Boutique Fudge будут довольны в будущем. Он выбирает заказы с отсутствующими на складе продуктами и изменяет поле **OnHold**. Теперь вам достаточно убедиться в том, что представители отдела обслуживания клиентов вежливы

Если вы создали похожий запрос, возможно, вам захочется включить в БД и запрос, выполняющий обратное действие и возвращающий заказы из режима ожидания в действующие при пополнении запаса нужных продуктов. Применяя знания, полученные в данном разделе, вы без проблем разработаете необходимый запрос.

9. Глава 9. Анализ данных с помощью перекрестных запросов и сводных таблиц

Программа Access приспособлена и нацелена на сохранение всех подробностей в вашей БД. Но иногда вам не нужно знать все детали - взамен вы хотите получить общее представление. Необходим способ получения необработанных данных, которые могут включать сотни и тысячи записей и подведения итогов каким-либо осмысленным образом.

Вы уже познакомились с одним методом анализа больших объемов информации с помощью итогового запроса *(см. разд. "Итоговые данные" главы 7)*. Применяя итоговый запрос, можно взять огромную подборку строк и сократить ее для нескольких искусно сгруппированных промежуточных итогов. В этой главе вы познакомитесь с двумя более специализированными вариантами обработки чисел: перекрестными запросами и сводными таблицами.

Перекрестные запросы и сводные таблицы играют ту же роль, что и уже знакомые вам итоговые запросы. Но они представляют данные несколько иначе. В перекрестных запросах применяются дополнительные столбцы для размещения информации в крайне сжатой таблице. Сводные таблицы используют интерфейс перемещений, позволяющий реорганизовать ваши итоги на лету для выявления различных тенденций и связей. Оба эти средства интенсивно используются в наборе средств любого специалиста Access.

Примечание

Для опробования перекрестных запросов и сводных таблиц вам нужны данные - большой объем данных. В БД, используемых в качестве примеров в предыдущих главах, нет достаточного количества исходных данных. В примерах этой главы используются некоторые таблицы огромной БД AdventureWorks - примера, предлагаемого корпорацией Microsoft и содержащего каталог товаров и сведения о продажах вымышленного производителя велосипедов. Найдите Web-страницу "Missing CD" для данной книги (на сайте **www.missingmanuals.com)** для загрузки нужной вам информации.

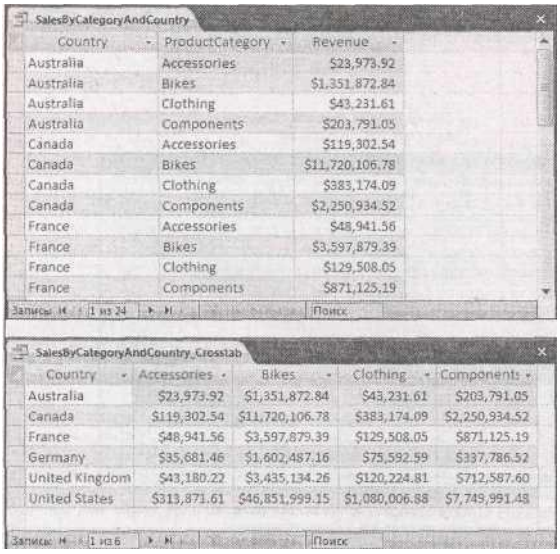
9.1. *О перекрестных запросах*

Перекрестный запрос - это мощное средство подведения итогов, исследующее большие объемы данных и применяющее их для вычисления промежуточных итогов и средних значений. Это определение может показаться знакомым, поскольку вы уже именно для этой цели использовали итоговые запросы в *главе 7*.

Как и итоговые запросы, перекрестные применяют два основных компонента: группировку и функции подведения итога. Группировка применяется для объединения строк в небольшие подмножества. Функция подведения итога используется для вычисления единого значения для каждой группы.

За кадром перекрестные и итоговые запросы функционируют почти одинаково. Принимают большие количества записей и сокращают их до итогов, средних, минимальных или максимальных значений и т. д. Но есть два важных отличия.

Первое отличие заключается в том, в перекрестных запросах всегда применяется двух-уровневая группировка. Например, в типичных итоговых запросах можно сгруппировать записи по товарам и увидеть самые ходовые из них или размер приносимого ими дохода. В перекрестном запросе можно проанализировать данные о продажах в зависимости от страны и категории товара. С помощью такого анализа вы сможете быстро определить, какие категории товаров особенно популярны в конкретных странах.



SalesByCategoryAndCountry

Country	ProductCategory	Revenue
Australia	Accessories	$23,973.92
Australia	Bikes	$1,351,872.84
Australia	Clothing	$43,231.61
Australia	Components	$203,791.05
Canada	Accessories	$119,302.54
Canada	Bikes	$11,720,106.78
Canada	Clothing	$383,174.09
Canada	Components	$2,250,934.52
France	Accessories	$48,941.56
France	Bikes	$3,597,879.39
France	Clothing	$129,508.05
France	Components	$871,125.19

Записи: 1 из 24 Поиск

SalesByCategoryAndCountry_Crosstab

Country	Accessories	Bikes	Clothing	Components
Australia	$23,973.92	$1,351,872.84	$43,231.61	$203,791.05
Canada	$119,302.54	$11,720,106.78	$383,174.09	$2,250,934.52
France	$48,941.56	$3,597,879.39	$129,508.05	$871,125.19
Germany	$35,681.46	$1,602,487.16	$75,592.59	$337,786.52
United Kingdom	$43,180.22	$3,435,134.26	$120,224.81	$712,587.60
United States	$313,871.61	$46,851,999.15	$1,080,006.88	$7,749,991.48

Записи: 1 из 6 Поиск

Рис. 9.1. *Вверху:* в итоговом запросе каждая группа занимает отдельную строку, отображая объемы продаж отдельной категории товаров в одной стране. В общем получается 24 группы, и в результате формируется длинный узкий список. *Внизу:* в перекрестном запросе программа Access применяет первый уровень группировки (в данном случае страну) для разделения данных на строки и следующий уровень (категорию товара) для распределения каждой строки по столбцам. Числа, которые вы видите, те же самые, что и на верхнем рисунке, но теперь у вас всего 6 строк с четырьмя категориями товаров в каждой

Другое отличие перекрестных запросов от итоговых заключается в способе представления программой Access результатов. Итоговый запрос создает отдельную строку для каждой группы. Например, если вы анализируете продажи в разных странах различных категорий товаров, итоговый запрос выведет строку для каждой комбинации страны и категории товаров, как показано в верхней части рис. 9.1. Перекрестный запрос действует несколько иначе; он использует ту же информацию, но располагает ее в отдельных столбцах, создавая более сжатое представление (нижняя часть рис. 9.1).

На рис. 9.1 внизу показано, как выглядят данные с двумя уровнями группировки: страны и товары. Но если хотите, в перекрестных запросах можно применять и больше уровней. (Большее число уровней группировки полезно для выполнения более детального анализа - например, для определения, какие товары особенно популярны в конкретных странах, штатах и городах.) В этом случае последний уровень группировки используется для разделения строки на столбцы. Все остальные уровни применяются для разбиения результатов на большее число строк. Если создать перекрестный запрос, группирующий объемы продаж по категориям товаров, названиям товаров и странам, вы увидите результат, показанный на рис. 9.2.

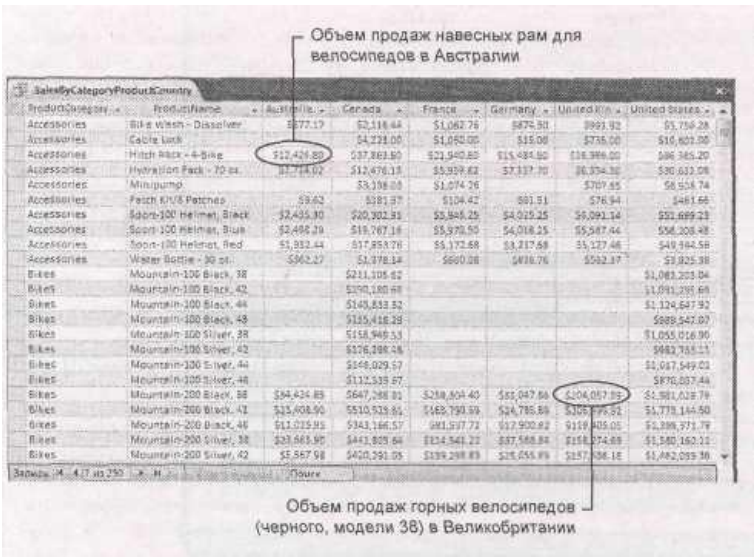


Рис. 9.2. В данном примере записи сгруппированы в строки по категориям товаров и затем разделены на отдельные товары. Затем данные по каждому товару дополнительно делятся на столбцы в зависимости от страны

Примечание

Помните о том, что при использовании нескольких уровней группировки последний уровень (используемый для формирования столбцов) не должен быть связан с другими уровнями. В то время как другие уровни группировки могут быть связаны между собой. Пример на рис. 9.2 работает, потому что следует этому правилу (группировка по категории, товару и затем стране). Если эти же данные сгруппировать иначе (например, по категории, стране и товару), результат будет далеко не так хорош.

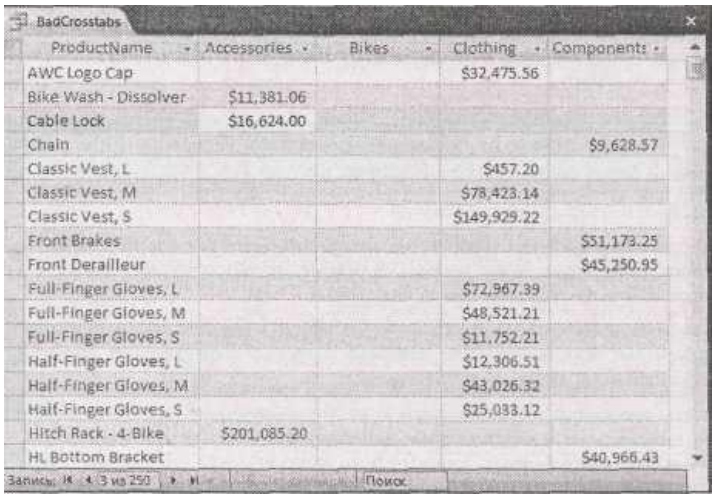
Часто задаваемый вопрос.
Итоговый проигрыш; итоговый запрос против перекрестного

Что лучше: итоговый или перекрестный запрос?

Все зависит от типа информации, которую вы хотите анализировать, и от способа ее структурирования. Далее приведены несколько рекомендаций, которые помогут решить, какой вариант больше подходит для ваших данных.

- *Если вы хотите сгруппировать в зависимости от одного поля (как показано, например, на верхнем рис. 9.1), используйте итоговый запрос.* У перекрестных запросов всегда, по меньшей мере, два уровня группировки.
- *Если вы хотите выполнять несколько типов вычислений (например, находить среднее и итоги или минимальные и максимальные значения), применяйте итоговый запрос.* Из-за своего компактного формата перекрестные запросы могут отображать только одно вычисляемое значение в каждой группе. Итоговые запросы выводят на экран столько вычисляемых значений, сколько вам нужно, поскольку каждое из них помещается в отдельный столбец.
- *Если вы хотите сравнить одну группу с другой, применяйте перекрестный запрос.* В этом у перекрестных запросов нет равных. Они размещают подгруппы в одной строке, поэтому тенденции видны с первого взгляда. Примером может служить рис. 9.1. В перекрестном запросе легко установить, какие приспособления принесли меньше всего денег независимо от страны, на которую вы смотрите. В итоговом запросе для проведения такого же сравнения вашим глазам пришлось бы перебегать со строчки на строчку вверх и вниз.
- *Если ваши условия группировки в результате приводят к большому числу групп, подумайте о применении итогового запроса.* Перекрестный запрос может читаться с трудом, если в него включено много столбцов. (Другая возможность - использование фильтрации для сокращения числа групп.)
- *Если применяются два независимых уровня группировки, используйте перекрестный запрос.* Например, категория товаров и страна клиента - полностью независимые критерии отбора. Вы не можете узнать, любят ли в конкретных странах определенные категории, пока не пороетесь в числах. Для подобной организации информации очень подходит перекрестный запрос. С другой стороны, категория товаров и название изделия связаны друг с другом. Каждый товар попадает в заданную категорию и ни один не может оказаться в нескольких категориях. Если такой способ группировки применить в перекрестном запросе, пропадет много свободного пространства, как показано на рис. 9.3.

Во многих ситуациях можно попробовать оба подхода - создать и итоговый запрос, и перекрестный - а затем сравнить их, чтобы понять, какое представление информации вам больше нравится.



BadCrosstabs

ProductName	Accessories	Bikes	Clothing	Components
AWC Logo Cap			$32,475.56	
Bike Wash - Dissolver	$11,381.06			
Cable Lock	$16,624.00			
Chain				$9,628.57
Classic Vest, L			$457.20	
Classic Vest, M			$78,423.14	
Classic Vest, S			$149,929.22	
Front Brakes				$51,173.25
Front Derailleur				$45,250.95
Full-Finger Gloves, L			$72,967.39	
Full-Finger Gloves, M			$48,521.21	
Full-Finger Gloves, S			$11,752.21	
Half-Finger Gloves, L			$12,306.51	
Half-Finger Gloves, M			$43,026.32	
Half-Finger Gloves, S			$25,033.12	
Hitch Rack - 4-Bike	$201,085.20			
HL Bottom Bracket				$40,966.43

Записи: 1 из 250 Поиск

Рис. 9.3. Считайте это предостережением: не применяйте для группировки в перекрестном запросе связанные поля. В данном примере строки сгруппированы по названию товара, а столбцы по категории товаров. Проблема состоит в том, что каждый товар включен только в одну категорию, поэтому в каждой строке данные есть лишь в одном столбце - столбце с категорией данного товара. Для решения проблемы и формирования более наглядной сводки можно использовать три уровня группировки, как показано на рис. 9.2

9.2. *Создание перекрестных запросов*

Программа Access предоставляет два способа создания перекрестного запроса: можно воспользоваться мастером создания перекрестного запроса или построить его вручную. Большинство приверженцев Access предпочитают для начала использовать мастер, а затем совершенствовать свой запрос в **Конструкторе,** добавляя разные детали, такие как условие отбора.

В следующих разделах вы попробуете силы в приготовлении перекрестного запроса обоими способами.

Для тех, кто понимает.
Создание запроса с объединением для лучшей группировки

Когда применяется любом тип запроса, использующий группировку, часто требуется извлечь информацию из нескольких разных таблиц. Например, если рассматривается таблица с объемами продаж, возможно, понадобится дополнительная информация о проданных товарах, клиентах, купивших их, месте покупки и т. д. Чаще всего приходится сводить воедино информацию из нескольких связанных таблиц.

Самый легкий способ создания сводного запроса (такого как перекрестный) - формирование еще одного запроса, в котором есть вся нужная вам информация.

В этом новом запросе используются операции объединения *(см. разд. "Запросы и связанные таблицы" главы 6)* для соединения всех таблиц с необходимыми данными. Затем можно использовать этот запрос для построения сводного. Такой подход особенно полезен в случае перекрестных запросов, поскольку мастер создания перекрестного запроса способен использовать только одну таблицу или запрос. Он самостоятельно не может объединить таблицы.

В БД AdventureWorks есть запрос **OrderedItems,** который формирует основу для всех перекрестных запросов, которые вы видели до сих пор. Запрос **OrderedItems** получает все товары, приобретенные во всех когда-либо сделанных заказах (из таблицы **SalesOrderDetails),** и затем использует объединения для извлечения дополнительной информации из таблиц **SalesOrderHeader** (представляющей заказ целиком), **Customers, Products, Store** и **ShipMethod.** Вам придется выполнить несколько переходов для получения данных об адресе клиента, которые позволят проследить, как объемы продаж распределены в разных городах, штатах и странах. (Этот запрос можно изучить, загрузив БД AdventureWorks со страницы "Missing CD" Web-сайта **www.missingmanuals.com.)**

9.2.1. *Создание перекрестного запроса с помощью мастера*

Легче всего построить перекрестный запрос с помощью мастера создания перекрестного запроса. Если вы хотите сделать это самостоятельно, выполните следующие действия, пользуясь БД AdventureWorks.

1. Если нужно собрать информацию из связанных таблиц, начните с создания запроса с объединением (join query).

В данном примере используется уже созданный запрос **Orderedltem** с объединением таблиц, заимствующий массу данных о компонентах заказов, соответствующих товарах, клиентах, месте их проживания и т. д. Дополнительные сведения, необходимые для самостоятельного создания запроса с объединением, *см. в разд. "Запросы и связанные таблицы" главы 6.*

Если вы считаете, что можете получить все необходимое из одной таблицы, пропустите этот пункт.

2. Выберите на ленте **Создание → Другие → Мастер запросов** (Create → Other → Query Wizard).

Теперь начинает действовать чудесный мастер. На экране появляется окно **Создание запроса** (New Query) со списком запросов разных типов, которые может создать мастер.

3. Выберите **Перекрестный запрос** (Crosstab Query Wizard) и щелкните мышью кнопку **ОК**.

Сначала мастер попросит выбрать таблицу или запрос (рис. 9.4). Выберите один из переключателей в области **Показать** (View).

4. Выберите нужную таблицу. Если хотите выбрать запрос, щелкните кнопкой мыши переключатель **Запросы,** а затем выберите ваш запрос. Щелкните мышью кнопку **Далее.**

В данном примере следует щелкнуть мышью переключатель **Запросы** и затем выбрать запрос **OrderedItems.**

На следующем этапе нужно задать критерии группировки, которые будут применяться для объединения данных в строки (рис. 9.5).

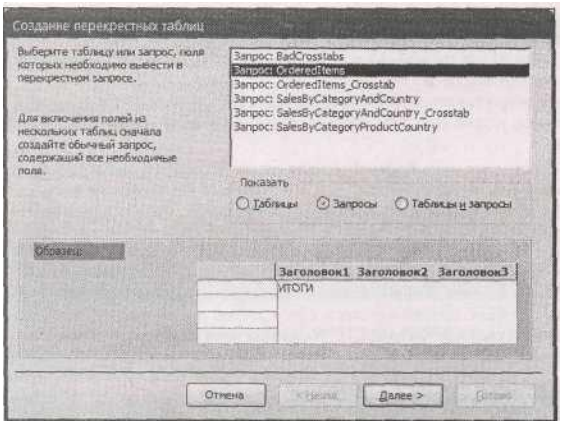


Рис. 9.4. Для просмотра таблиц вашей БД щелкните кнопкой мыши переключатель **Таблицы,** а для просмотра запросов - **Запросы**

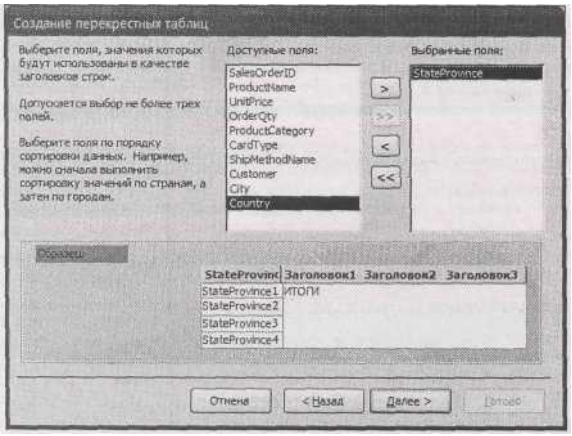


Рис. 9.5. Для применения поля в качестве заголовка строки выберите его в списке **Доступные поля** и затем щелкните мышью забавную кнопку > для переноса поля в список **Выбранные поля**

Если создается простой двухуровневый перекрестный запрос, выберите один критерий для строк и один для столбцов (на следующем шаге). Но есть возможность задать до трех уровней группировки строк. Этот вариант больше всего подходит для разных уровней, связанных между собой. Например, вы можете выбрать группировку по стране проживания клиента, в каждой стране подгруппу для городов и в каждом городе подгруппу на основе идентификационного номера (ГО) клиента. Пример хорошо сгруппированного перекрестного запроса см. на рис. 9.2.

5. Включите поля, которые хотите использовать, в список **Выбранные поля** и затем щелкните мышью кнопку Далее.

В примере с запросом **Orderedltems** строки группируются по полю **State Province.** После того как вы опробуете свой запрос, группировку можно легко изменить в окне **Конструктора.** Например, если хотите, можно поменять поле **StateProvince** на поле **Country.** Как изменить перекрестный запрос, рассказывается в *разд. "Создание перекрестного запроса с нуля " далее в этой главе.*

Теперь следует задать критерий группировки, применяемый для разделения ваших строк на столбцы (рис. 9.6). В этот момент можно выбрать одно поле.

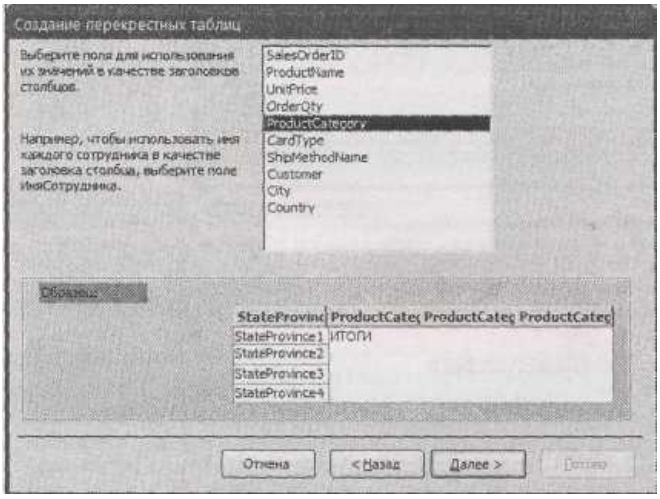


Рис. 9.6. По мере выполнения мастера программа Access выводит в нижней части его окна мини-окно предварительного просмотра структуры вашего находящегося в процессе создания перекрестного запроса. В данном примере строки группируются по полю **StateProvince,** а столбцы - по полю **ProductCategory**

6. Выберите поле для группировки столбцов и щелкните мышью кнопку Далее. В данном примере это поле **ProductCategory.**

На последнем шаге вы должны подобрать вычисление, которое хотите выполнять для получения итогов.

Выберите поле для вычисления и затем функцию для подсчета сводных данных (рис. 9.7). Например, можно найти самую дешевую продажу, заказ с наибольшим числом проданных товаров, среднюю цену товара и т. д. В данном примере для подсчета количества проданных товаров используется поле **OrderQty.**

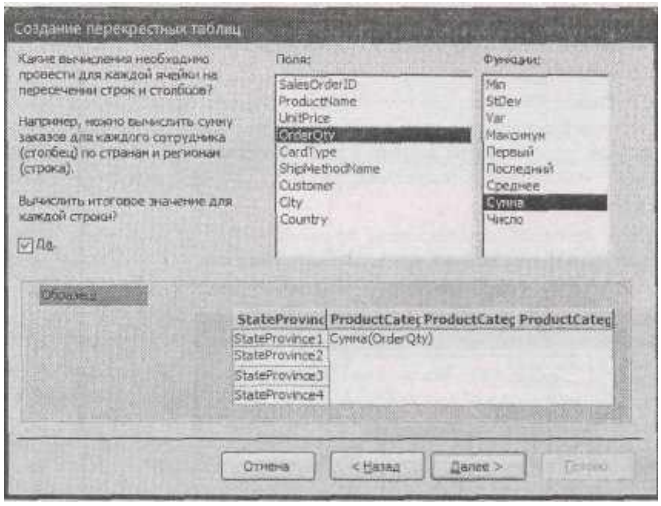


Рис. 9.7. В этом примере функция Sum суммирует значения поля **OrderQty** из всех записей. Например, данный запрос сообщает о том, что вы продали в целом 53 товара из категории Bike (велосипеды) клиентам из Алабамы. Если нужно посчитать, сколько заказов сделали ваши клиенты (вместо количества доставленных товаров), необходим немного другой запрос - в этом случае следует использовать функцию Count для подсчета различных значений поля **SalesOrderID**

На профессиональном уровне.

Правильный выбор групп

Пытаетесь решить, какое поле использовать для группировки в строках, а какое для группировки в столбцах? Если эти два поля независимы (а они должны быть таковыми), есть смысл применить поле, создающее меньше всего групп для группировки в столбцах. Таблицы с множеством строк и несколькими столбцами легче читать (и печатать), чем таблицы с множеством столбцов и несколькими строками.

Например, если группировать по названию товара и стране, можно биться об заклад, что вы в результате получите больше групп товаров, чем стран. (У вас могут быть клиенты в восьми разных странах, а каталог товаров с 480 видами товаров.) Итак, примените группировку по товарам для строк, а по странам - для столбцов.

8. Если нужно показать промежуточный итог для каждой строки, установите флажок **Вычислить итоговое значение для каждой строки? Да** (Yes, include row sums).

Промежуточный итог по строке отображается в самом первом столбце. Например, если установлен этот флажок в запросе со штатами и категориями товаров, общие объемы продаж для каждого штата отображаются в первом столбце, за которым следует разбиение продаж по категориям (рис. 9.8).

OrderedItems_Crosstab

StateProvince	Total Of OrderQty	Accessories	Bikes	Clothing	Components
Alabama	87		53	12	22
Alberta	4410	582	1025	1644	1159
Arizona	4116	429	1257	1193	1237
Bayern	901	240	164	373	124
Brandenburg	218	7	132	58	21
British Columbia	8943	1137	3414	2635	1757
Brunswick	1605	387	327	677	214
California	24302	2701	9292	7190	5119
Colorado	6808	950	2524	2189	1145
Connecticut	2421	62	1223	476	660
England	13193	1849	4082	4201	3061
Essonne	381		105	44	232
Florida	6473	925	2135	2268	1145
Garonne (Haute)	578		368	26	184
Georgia	2601	236	964	921	480
Hamburg	1240	276	285	542	137
Hauts de Seine	2369	396	685	835	453
Hessen	1143	246	356	348	193
Idaho	465	33	250	22	160

Записи: 2 из 64 Поиск

Рис. 9.8. Заключительный перекрестный запрос показывает связь между штатами и типами товаров, покупаемыми его жителями (слева дается итог для каждого штата или провинции по всем категориям товаров)

9. Щелкните мышью кнопку Далее.

На заключительном шаге придется задать имя запроса. Затем можно выбрать запуск запроса и просмотр полученных результатов или продолжить его редактирование в **Конструкторе.** Если вам необходима фильтрация, перейдите в **Конструктор.** В противном случае самое время увидеть плод вашего труда.

10 . Щелкните мышью кнопку **Готово.**

9.2.2. *Создание перекрестного запроса с нуля*

Как и запрос любого другого типа, перекрестный запрос можно тонко настроить в **Конструкторе.** Вы также можете создать новый перекрестный запрос с нуля, выполнив следующие действия.

1. Выберите на ленте **Создание → Другие → Конструктор запросов** (Create → Other → Query Design). Программа Access создаст новый пустой запрос и откроет его в окне **Конструктора.**

2. С помощью окна **Добавление таблицы** (Show Table) добавьте таблицу или запрос, которые вы хотите использовать, и щелкните мышью кнопку **Закрыть.**

Если вы используете БД AdventureWorks, легче всего выбрать вкладку **Запросы (Queries)** в окне **Добавление таблицы** и добавить запрос **Orderedltems.**

Подсказка

Вы также можете закрыть диалоговое окно **Добавление таблицы** и просто перетащить таблицы, которые вам нужны, из области переходов на поверхность рабочего поля **Конструктора запросов.**

3. Выберите на ленте **Работа с запросами | Конструктор → Тип запроса → Тип запроса: перекрестный** (Query Tools | Design → Query Type → Crosstab),

Программа Access преобразует ваш запрос в перекрестный. Перекрестные запросы выглядят как итоговые с одной лишь разницей. В списке полей в нижней части окна вы найдете дополнительную строку - **Перекрестная таблица** (Crosstab) (рис. 9.9).

Рис. 9.9. Как и у итоговых запросов, у перекрестных есть свойство **Групповая операция,** в котором задается способ применения поля: для группировки, фильтрации или вычисления итога. В отличие от итоговых запросов перекрестные включают также свойство **Перекрестная таблица,** в котором задается размещение полей в строке, в столбце, в виде значений или полное скрытие (в этом случае вы, вероятно, применяете поле для сортировки или фильтрации)

4. Выберите поля, которые хотите использовать в своем перекрестном запросе. Каждое поле в перекрестном запросе играет одну из следующих ролей.

□ *Поле используется для группировки по строкам.* В данном случае задайте в свойстве **Групповая операция** значение **Группировка** и значение **Заголовки строк** (Row Heading) в свойстве **Перекрестная таблица.**

Несмотря на то, что мастер создания перекрестного запроса ограничивает вас тремя полями для группировки в строках, вы на самом деле можете добавить практически неограниченное число полей для группировки в столбцах. Убедитесь в том, что столбцы размещены надлежащим образом. Например, если у вас два поля для группировки строк, поле слева используется для группировки первым, а затем группы разбиваются с помощью следующего поля.

□ *Поле применяется для группировки по столбцам.* В этом случае задайте в свойстве **Групповая операция** значение **Группировка** и значение **Заголовки столбцов** (Column Heading) в свойстве **Перекрестная таблица.**

Для этой цели вы должны использовать только одно поле. Помните о том, что группировка по столбцам выполняется после применения группировки строк.

Поле отображается как значение в таблице. В этом случае задайте в свойстве **Групповая операция** итоговую функцию, которую хотите использовать (такую как Sum, Count, Avg и т. д.), и вариант **Значение** (Value) в свойстве **Перекрестная таблица.**

Для этой цели вы должны использовать только одно поле. Но вы можете применить выражение, выполняющее вычисления, базирующиеся на значениях нескольких полей. Например, перекрестные запросы, показанные на рис. 9.1 и 9.2, используют выражение Revenue: [UnitPrice] * [OrderQty] для вычисления общей выручки для каждой строки заказа.

Подсказка

Возможно, вы помните, что мастер создания перекрестного запроса предоставляет возможность показать итог для каждой строки в отдельном столбце. На рис. 9.10 показано, как создать

подобный эффект самостоятельно.

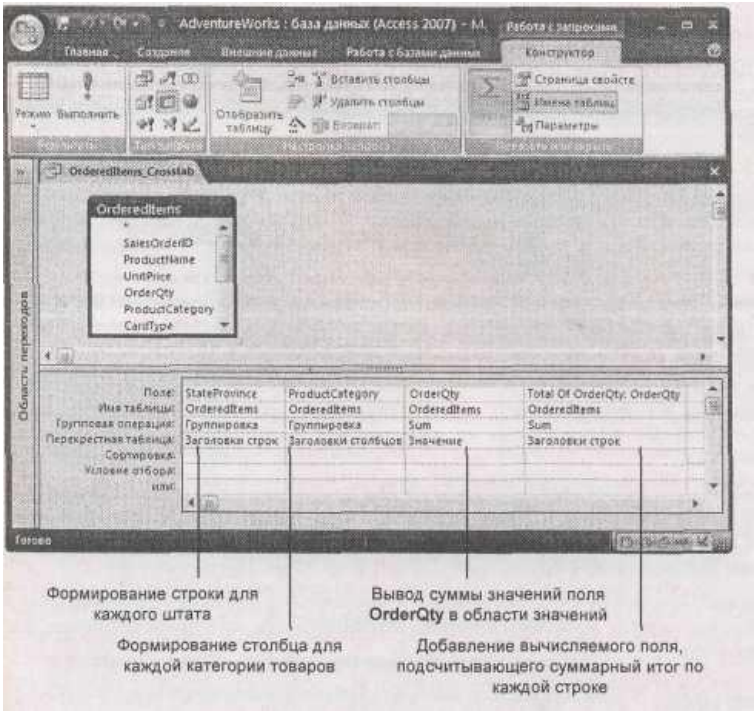


Рис. 9.10. Обратите внимание на то, что поле **OrderQty** появляется дважды. В первый раз оно определено как значение, отображаемое в сетке таблицы. Во второй раз оно определяется как заголовок строки, которая создается в дополнительном столбце с итогом для каждой строки. С помощью псевдонима дополнительный столбец переименован в **Total Of OrderQty** (общее количество в заказе) во избежание путаницы

□ *Поле применяется для фильтрации или отбора.* В этом случае задайте в свойстве Групповая операция значение Условие (Where) и вариант (не отображается) (not shown) в свойстве Перекрестная таблица. Затем вставьте условие в поле Условие отбора. (См. выражения для условий отбора в *разд. "Построение условий отбора" главы* 6.)

Примечание

К сожалению, в вычисляемом поле нельзя применить фильтрацию или сортировку. Это означает, что если создается запрос, подсчитывающий количество продаж, нельзя отобрать только строки с большими количествами продаж. Но этот трюк можно проделать в сводной таблице, как описано в следующем разделе.

На рис. 9.10 показано определение запроса, подобного созданному с помощью мастера в предыдущем разделе (см. рис. 9.8).

9.3. *Сводные таблицы*

Если итоговые и перекрестные запросы не потрясли вас в достаточной степени, у программы Access есть еще одно мощное средство для подытоживания ваших данных. *Сводная таблица* - это специальная таблица, выполняющая те же трюки, что и перекрестный запрос - группировку по строкам и столбцам - но обладающая большими функциональными возможностями. Далее перечислены некоторые из них.

■ *Сводные таблицы можно перестроить в любой момент.* Быстрым перемещением с помощью мыши вы можете превратить итоги продаж по странам в сетку с продажами в зависимости от возраста клиента. Сводные таблицы - незаменимое средство *исследования* (добычи) данных, в ходе которого вы пытаетесь выудить скрытые тенденции и связи из массы необработанных сведений.

■ *Сводные таблицы поддерживают неограниченное число уровней группировки.* Вы не ограничены одним уровнем группировки по столбцам, как в перекрестном запросе. Вместо этого

можно разбить строки и столбцы на группы более мелкие.

■ *Сводные таблицы сворачиваются.* Можно скрыть группы строк и столбцов, которые не интересуют вас в данный момент, и углубиться в группу для просмотра каждой записи, содержащейся в ней. Просматривая данные подобным способом, легче понять, что с ними происходит.

■ *Сводные таблицы поддерживают неограниченное количество вычислений.* Перекрестные запросы способны выполнять одно вычисление, повторяющееся для каждой группы. Сводная таблица может выполнить столько вычислений, сколько вам нужно, и поместить их все в одну и ту же ячейку.

■ *Сводные таблицы поддерживают сортировку вычисляемых значений.* Например, если в вашу сводную таблицу добавлены итоги по объемам продаж, лучшие исполнители поднимаются вверх.

Примечание

Многие приверженцы программы Access живут долго и счастливо, даже не сталкиваясь со сводными таблицами. И все потому, что это узкоспециализированное средство, и многие специалисты предпочитают анализировать информацию в других программах (например, в Microsoft Excel). Но свойства сводных таблиц заслуживают внимания, поскольку могут оказаться полезными в следующий раз, когда вам потребуется сделать общие выводы о производительности вашей компании, выпекающей изделия, посвященные знаменитостям.

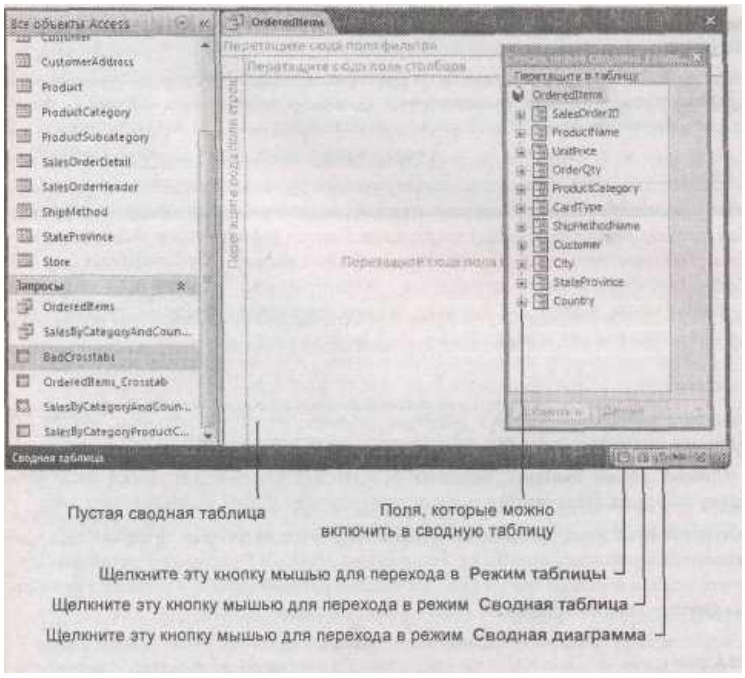
9.3.1. *Построение сводной таблицы*

Программа Access включает в состав сводные таблицы несколько странным образом. В отличие от итоговых и перекрестных запросов сводные таблицы - это не специальный тип запроса. Access обеспечивает сводную таблицу режимом отображения, который можно применять с любой таблицей или запросом.

Примечание

Причина выбора этого кажущегося странным дизайна состоит в необходимости обеспечить чрезвычайную гибкость таблиц. С помощью нескольких щелчков мыши вы можете реорганизовать категории или углубиться в анализ, перейдя от отображения сводных данных к отдельным записям. Для того чтобы это стало возможным, сводным таблицам нужен легкодоступный полный набор записей.

Для применения режима **Сводная таблица** (Pivot Table) откройте таблицу или запрос, которые хотите использовать, и затем выберите на ленте **Главная → Режимы → Режим → Сводная таблица** (Home → Views → View → PivotTable View).



Сначала сводная таблица отображается пустой (рис. 9.11).

Рис. 9.11. Этот пример отображает запрос **Orderedltems** в режиме **Сводная таблица.** Пока еще не на что смотреть, потому что вы не построили сводную таблицу. Справа открыто окно **Список полей сводной таблицы,** включающее все поля вашей таблицы или запроса

Примечание

Сводные таблицы одновременно работают только с одной таблицей или запросом. Поэтому стоит создать запрос, объединяющий все нужные вам таблицы, так же как вы делали при построении перекрестного запроса. Можно также воспользоваться запросом для создания дополнительных вычисляемых полей (например, поля, перемножающего стоимость товара и количество единиц товара).

Для создания сводной таблицы нужно сообщить программе Access о том, какие поля использовать в каждой части таблицы. Все сводные таблицы сформируются из пяти компонентов:

■ *поля строк* применяются для группировки записей в строках;

■ *поля столбцов* используются для группировки записей в столбцах;

■ *поля итогов* применяются для вычисления итоговых значений в каждой группе;

■ в *полях деталей* или *подробностей* выводятся отдельные значения для каждой записи в группе. При желании вы можете выводить на экран итоговую информацию (в этом случае поле деталей действует как поле итогов);

■ *поля фильтра* применяются для сокращения списка записей, используемых для создания сводной таблицы, на основании заданного критерия.

Примечание

Структура сводной таблицы очень похожа на структуру перекрестного запроса - ключевое отличие состоит в том, что многие из ограничений, сужающих перекрестные запросы, не применяются в сводных таблицах.

Самый легкий способ освоить сводные таблицы и множество их функциональных возможностей - попытаться построить хотя бы одну своими руками. Приведенные далее действия подробно описывают процесс создания простой сводной таблицы, которая отображает итоги объемов продаж, сгруппированных по странам и категориям товаров. Если вы хотите действовать в соответствии с ними, используйте запрос **OrderedItems** в БД AdventureWorks, которую можно загрузить с Web-страницы "Missing CD" на сайте **www.missingmanuals.com.** Затем вы сможете усовершенствовать сводную таблицу, воспользовавшись ее богатыми возможностями.

Примечание

Предпочитаете визуальный подход к изучению сводных таблиц? На странице "Missing CD" можно найти и изобразительный ряд, интерактивное анимационное руководство.

1. Из окна **Список полей сводной таблицы** перетащите с помощью мыши поле **ProductCategory** в область **Перетащите сюда поля строк** (Drop Row Fields Here). Когда вы переместите поле, программа Access выведет все категории товаров сверху донизу в алфавитном порядке (рис. 9.12). Если нужен обратный порядок сортировки, просто щелкните правой кнопкой мыши одно из значений и выберите последовательно **Сортировка → Сортировка по убыванию** (Sort →Sort Descending).

Подсказка

Если вы случайно закрыли окно со списком полей сводной таблицы, его можно вернуть выбрав на ленте **Работа со сводными таблицами I Конструктор -> Показать или скрыть -» Список полей** (PivotTable Tools I Design -> Show/Hide -> Field List).

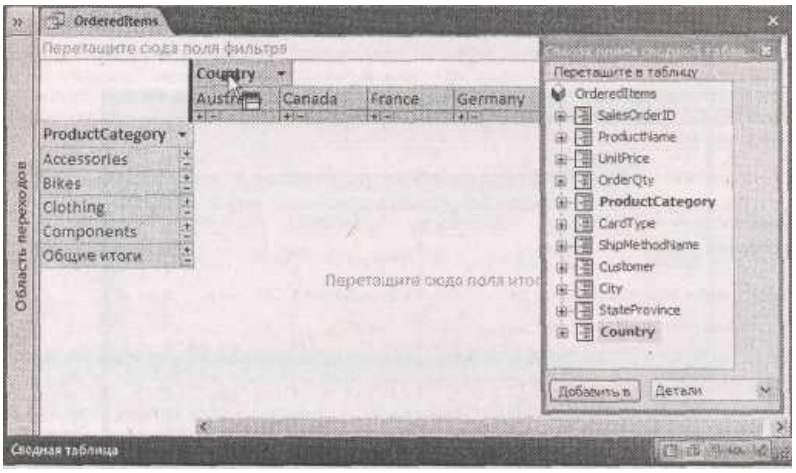


Рис. 9.12. В данном примере в область строк добавлен список товаров, а элементы второй группировки (список стран) переносится в область столбцов. Обратите внимание, как только поле выбрано, его имя отображается жирным шрифтом в окне **Список полей сводной таблицы**

2. Из списка полей сводной таблицы перетащите поле **Country** в область **Перетащите сюда поля столбцов** (Drop Column Fields Here).

Как только вы перетащите поле, программа Access заполнит его слева направо в алфавитном порядке названиями всех стран из списка. Другими словами, каждая страна приведена в своем столбце.

Подсказка

Если перемещение с помощью мыши утомительно, есть другой способ создания макета сводной таблицы. В окне **Список полей сводной таблицы** просто выберите поле, которое хотите добавить в сводную таблицу, и затем в раскрывающемся списке в нижней части окна выберите область, в которую нужно поместить поле. В завершение для вставки поля щелкните мышью кнопку **Добавить в** (Add to) (расположенную под списком).

3. Теперь нужно выбрать данные для анализа. Переместите поле **OrderQty** в область **Перетащите сюда поля итогов или деталей** (Drop Totals or Details Fields Here).

Этот шаг заполняет сводную таблицу данными (программе Access может понадобиться несколько мгновений для группировки всех данных, если у вас таблица большого объема).

Поле **OrderQty** добавлено как подробность, т. е. вы увидите все записи в таблице (или запросе) в соответствующих группах (рис. 9.13).

4. Пришло время добавить итоговые вычисления. Щелкните правой кнопкой мыши в сводной таблице поле **OrderQty** (подойдет любое поле **OrderQty)** и выберите вариант из подменю **Автовычисления** (AutoCalc).

Можно выполнить все групповые функции, с которыми вы знакомы, включая суммирование, подсчет количества **и** поиск среднего арифметического. Например, выберите

Автовычисления → Сумма, если хотите найти общее количество товаров, проданных в данной категории.

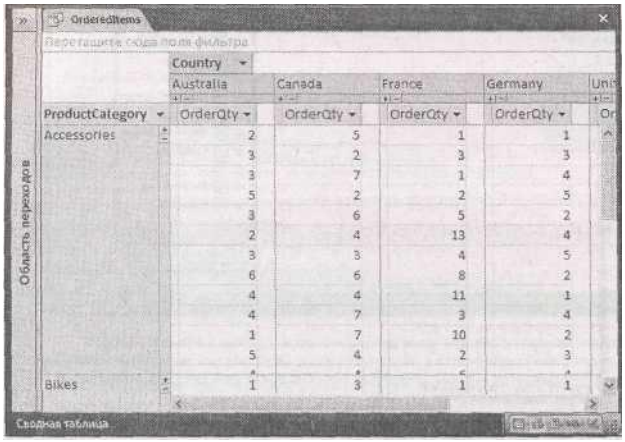


Рис. 9.13. В этой сводной таблице показаны все значения поля **OrderQty** из запроса **Orderedltems,** но их немного трудно анализировать. Для того чтобы составить общее представление, необходимо скрыть некоторые из этих деталей и взглянуть на итоговую информацию

Все сводные данные, которые создаются с помощью подменю **Автовычисления,** называют *итогами.* Они добавляются в группу **Итоги** (Totals) в верхней части окна **Список полей сводной таблицы.** (Щелкните кнопкой мыши квадратик со знаком +/-, расположенный рядом со словом **Итоги** для вывода на экран содержимого группы.) Для удаления итога щелкните его в списке правой кнопкой мыши и выберите команду **Удалить** (Delete).

5. Для скрытия подробной информации и отображения только сводных данных снова щелкните правой кнопкой мыши поле **OrderQty** и выберите команду Без **подробностей** (Hide Details).

После того как подробности будут скрыты, вы получите приемлемый результат, напоминающий перекрестный запрос (рис. 9.14).

Подсказка

Если вы знаете наперед, что никогда не захотите отображать подробности, можно добавить итоги сразу же. Для этого выберите поле (в данном случае **OrderQty)** в окне **Список полей сводной таблицы** и затем в раскрывающемся списке в нижней части этого окна выберите вариант **Данные** (Data Area). Далее щелкните мышью кнопку **Добавить в.** Этот подход удобен при работе с громадными таблицами, содержащими тысячи записей. В такой ситуации быстрее добавить итог, а не подробные сведения из каждой записи.

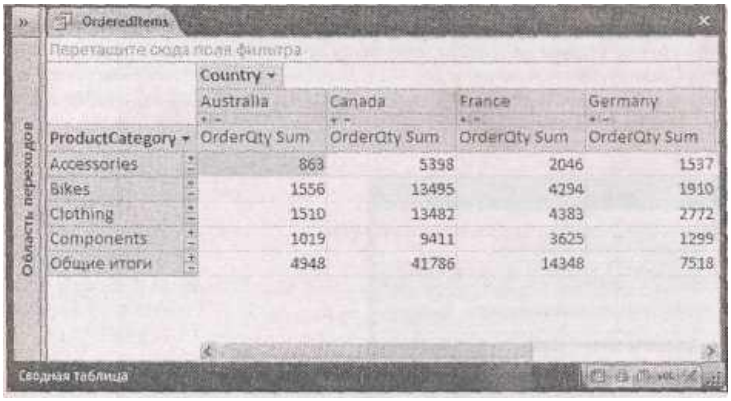


Рис. 9.14. Если убрать с дороги подробности, можно сразу увидеть признанные и не пользующиеся спросом группы. Итог для каждой строки выводится в конце строки (здесь не показан), а итог для каждого столбца отображается в строке **Общие итоги** в нижней части столбца

9.3.2. *Манипуляции сводной таблицей*

Вот где начинается веселье. Одно из ключевых достоинств сводных таблиц - их гибкость. Можно сколько угодно раз перемещать поля, вводить неограниченное число уровней группировки и выполнять множество вычислений сразу.

Далее приведено несколько способов быстрого изменения сводной таблицы.

■*Для удаления поля* щелкните его правой кнопкой мыши и выберите команду **Удалить** (Remove). Или перетащите *его* за пределы окна программы Access (так, чтобы указатель мыши превратился в стрелку со знаком X) и отпустите.

■*Для перемещения поля из одной позиции в другую* просто перетащите имя поля в нужное место. Например, можно изменить приведенный ранее пример, перетащив поле столбцов **(Country)** в область строк, а поле строк **(ProductName)** в область столбцов.

■*Для упорядочивания группы* щелкните правой кнопкой мыши внутри столбца этой группы и выберите последовательность команд **Сортировка → Сортировка по возрастанию** (Sort → Sort Ascending) или **Сортировка → Сортировка по убыванию** (Sort → Sort Descending). Вы можете использовать этот прием для поиска стран и категорий с максимальным объемом продаж (или приносящих максимальный доход).

■*Для добавления элементов данных* перетащите дополнительные поля из окна **Список полей сводной таблицы** в таблицу. Например, можно вычислить общее количество заказанных товаров и среднюю цену товара. Бы можете даже вставить одно и то же поле несколько раз для выполнения разных итоговых вычислений. Просто перетащите то же поле в таблицу еще раз, щелкните его правой кнопкой мыши и выберите вариант из подменю команды **Автовычисления.**

■*Для включения дополнительных уровней группировки* перетащите дополнительные поля из окна **Список полей сводной таблицы** в область строк или столбцов. Хитрость заключается в размещении поля в нужном месте иерархии группировок. Например, если вы хотите разбить группу страны на группы штатов, нужно поместить поле **StateProvince** справа от поля **Country,** как показано на рис. 9.15. Если же вы хотите разбить группы категорий товаров на группы отдельных товаров, нужно поместить поле **ProductName** справа от поля **ProductCategory.**

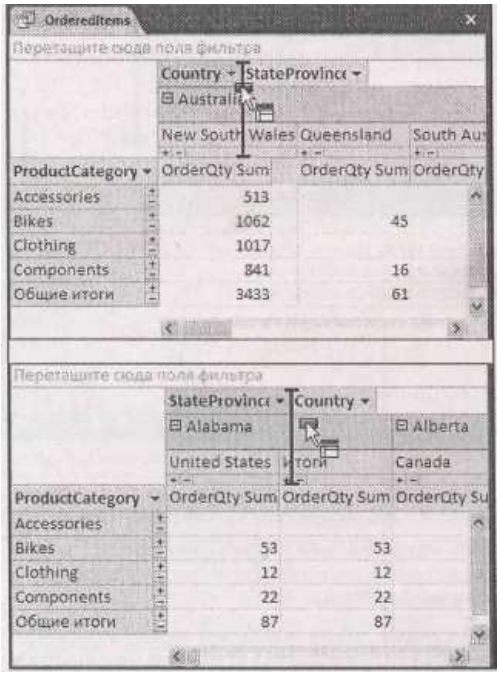


Рис. 9.15. *Вверху:* поле **StateProvince** располагается справа от поля **Country.** Теперь столбцы будут группироваться по странам (поле **Country)** и внутри стран по штатам (поле **StateProvince) -** как раз то, что нужно. Программа Access отображает толстую голубую линию, обозначая место расположения столбца после того, как вы отпустите левую кнопку мыши. *Внизу:* поле **StateProvince** располагается слева от поля **Country.** Теперь столбцы группируются сначала по штатам, а затем по странам. Программа разрешает сделать это, хотя в подобном действии мало смысла. Каждый штат принадлежит только одной стране, и в группе будет только одна подгруппа, которая бесполезна

Каждый раз, когда вы изменяете структуру сводной таблицы, программа просматривает вашу таблицу и полностью перестраивает сводную. Если вы изменяете данные в открытой сводной таблице, можно выбрать на ленте **Работа со сводными таблицами | Конструктор → Данные → Обновить** (PivotTabie Tools | Design → Data → Refresh Pivot), чтобы заставить программу Access сразу перестроить сводную таблицу.

9.3.3. *Создание вычисляемого поля*

Для получения самой интересной информации из сводной таблицы зачастую необходимо соединить несколько полей в выражении. Классический пример (который вы уже видели раньше в этой главе при обсуждении перекрестного запроса) - умножение количества заказанных изделий на цену изделия для определения итоговой выручки. Можно также умножить цену товара на количество единиц товара на складе для определения стоимости запасов, имеющихся у вас под рукой.

Этот метод действует **и в** сводной таблице, но для его реализации потребуется немного больше

работы.

1. В режиме **Сводная таблица** выберите на ленте **Работа со сводными таблицами | Конструктор → Сервис → Формулы → Создание вычисляемого поля сведений** (PivotTable Tools | Design → Tools → Formulas → Create Calculated Detail Field).

На экране появится окно **Свойства** (Properties) с несколькими вкладками, в данный момент отображена вкладка **Вычисление (Calculation) (рис. 9.16).**

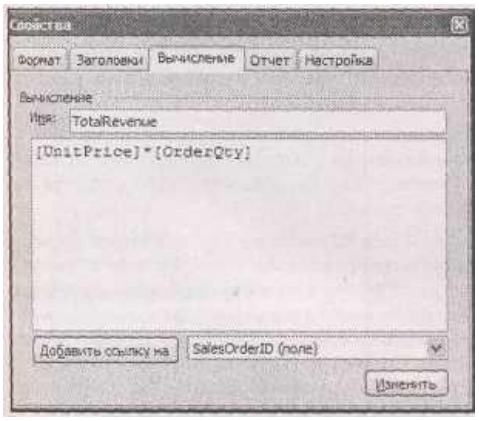


Рис. 9.16. Это вычисляемое поле перемножает значения двух полей

2. В поле **Имя** (Name) введите имя вычисляемого поля.
Например, можно ввести имя **TotalRevenue.**

3. В большое поле, расположенное под полем **Имя,** введите выражение, выполняющее вычисление.

Можно ввести, к примеру, [UnitPrice] * [OrderQty].

Вы можете применять любую комбинацию функций Access и полей исходной таблицы. (Сведения о создании выражений для вычисляемых полей можно найти в *разд. "Вычисляемые поля "главы 7.)* Если вы забыли имя поля, можно воспользоваться раскрывающимся списком, расположенным под текстовым полем. Просто выберите поле и щелкните мышью кнопку **Добавить ссылку на** (Insert Reference To).

4. Используя другие вкладки, измените, как считаете нужным, параметры форматирования вашего поля.

Другие вкладки позволяют управлять тем, как ваше вычисляемое поле решает иные задачи, связанные со сводной таблицей (например, отбор записей) и параметрами форматирования поля. Самые полезные параметры находятся на вкладке **Формат** (Format), позволяющей задать шрифт, цвет и (что особенно важно) формат вывода **Денежный** для поля **TotalRevenue,** таким образом, оно отображается со знаком валюты в соответствии с настройками вашего компьютера, запятыми и двумя знаками в дробной части.

5. Щелкните кнопкой мыши вкладку Вычисление (если она не выбрана в настоящий момент) и щелкните мышью кнопку **Изменить** (Change) для вставки вычисляемого поля в сводную таблицу.

Если вы воспользовались кнопкой **Без подробностей** (Hide Details) для сворачивания сводной таблицы и отображения только итоговых данных, то ничего не увидите в ней. Только что добавленное вами вычисляемое поле - это поле сведений или подробностей. Для того чтобы увидеть полный список значений всех записей, выберите на ленте **Работа со сводными таблицами | Конструктор → Показать или скрыть → С подробностями** (PivotTable Tools | Design → Show/Hide → Show Details), прежде чем продолжить. Ваше вычисляемое поле также появится в окне **Список полей сводной таблицы.** Если позже захотите от него избавиться, щелкните поле в этом списке правой кнопкой мыши и выберите команду **Удалить.** В следующем пункте добавляется полезный итог к вашему полю сведений.

6. Щелкните правой кнопкой мыши вычисляемое поле, выберите команду **Автовычисления** и задайте вариант групповой операции (например, Сумма), затем можно щелкнуть поле правой кнопкой мыши и выбрать команду **Без подробностей**, чтобы вернуться к более компактному сводному представлению данных.

Ваше итоговое поле появится в списке полей сводной таблицы в группе **Итоги,** расположенной

в верхней части списка. Для удаления этого поля щелкните его правой кнопкой мыши и выберите команду **Удалить.** Для удаления поля из сводной таблицы, но сохранения его под рукой для применения в дальнейшем, щелкните мышью поле в таблице и выберите команду **Удалить.** Если вам не нравится длинное имя итогового поля (которое обычно выглядит как "Sum of TotalRevenue" (Сумма TotalRevenue)), щелкните поле правой кнопкой мыши и выберите команду **Свойства,** чтобы открыть одноименное окно. Укоротить имя можно на вкладке **Заголовки** (Caption) в текстовом поле **Заголовок** (Name).

На рис. 9.17 показан окончательный вариант примера.

Малоизвестная или недооцененная возможность.

Помещение сводных таблиц в их собственные формы

Как вы уже поняли, сводная таблица - это режим необычного представления таблицы (или запроса). Когда вы закрываете окно после построения сводной таблицы, программа Access предлагает сохранить сделанные вами "изменения макета" (layout changes). В этот момент у вас есть два варианта: выбрать **Да** для сохранения структуры сводной таблицы с основной таблицей или запросом для просмотра в дальнейшем или выбрать **Нет** для того, чтобы отбросить все сделанное. Если вы выбрали **Нет**, то при переключении в режим **Сводная таблица** в следующий раз вы начнете снова с чистого листа и придется собирать сводную таблицу с пуля.

		Country ▾	StateProvince ▾				
		⊟ United States					
		Arizona		California		Colorado	
ProductCategory ▾	ProductName ▾	OrderQty Sum	TotalRevenue	OrderQty Sum	TotalRevenue	OrderQty	
⊟ Accessories	Bike Wash - Dissolver	46	$210.44	197	$931.74		
	Cable Lock	13	$195.00	162	$2,417.00		
	Hitch Rack - 4-Bike	47	$3,230.40	251	$17,634.00		
	Hydration Pack - 70 oz.	38	$1,198.78	148	$4,867.71		
	Minipump	18	$215.89	161	$1,925.84		
	Patch Kit/8 Patches	19	$26.11	32	$43.97		
	Sport-100 Helmet, Black	60	$1,154.13	525	$10,525.97		
	Sport-100 Helmet, Blue	69	$1,353.17	536	$10,687.82		
	Sport-100 Helmet, Red	83	$1,578.48	478	$9,588.62		
	Water Bottle - 30 oz.	36	$105.39	211	$630.54		
	Итоги	429	$9,267.78	2701	$59,253.21		
⊟ Bikes	Mountain-100 Black, 38			100	$178,874.47		
	Mountain-100 Black, 42			109	$192,374.43		
	Mountain-100 Black, 44			98	$174,824.48		
	Mountain-100 Black, 48			102	$181,743.21		
	Mountain-100 Silver, 38			95	$174,759.49		
	Mountain-100 Silver, 42			86	$156,399.54		
	Mountain-100 Silver, 44			95	$177,513.48		
	Mountain-100 Silver, 48			62	$118,149.65		
	Mountain-200 Black, 38	69	$89,111.18	211	$269,907.21		
	Mountain-200 Black, 42	55	$70,423.41	202	$257,288.67		
	Mountain-200 Black, 46	60	$77,983.76	125	$159,132.97		
	Mountain-200 Silver, 38	61	$80,288.23	156	$200,447.14		
	Mountain-200 Silver, 42	35	$45,438.66	152	$197,414.58		

Рис. 9.17. В представленной сводной таблице группы стран подразделяются на штаты, а группы категорий товаров - на отдельные изделия. В сводной таблице также отображаются два итоговых поля: общий доход **(TotalRevenu)** и общее количество проданных товаров **(OrderQty Sum)**

Если вы истинный ценитель сводных таблиц, то, возможно, вам захочется сохранить для одной и той же таблицы или запроса два разных варианта сводной таблицы. В этом случае вы смогли бы легко взглянуть на ваши данные под разными углами. К сожалению, каждая таблица или запрос могут вмещать только одну сводную таблицу. Что же делать предприимчивому разработчику БД?

Ответ - создать отдельную форму для вашей сводной таблицы. Формы - это настраиваемые окна, которые создаются для облегчения ввода и просмотра данных. Вы изучите их подробно в *части IV*. Но сейчас важно понять, что можно взять единственную таблицу и создать для нее бесконечное число форм сводной таблицы. Если вы планируете работать со сводными таблицами долго, отделение сводной таблицы от ваших данных - бесценная возможность.

Для создания формы сводной таблицы выполните следующие действия.

1. Выберите в области переходов таблицу или запрос, данные которых вы хотите использовать в сводной таблице.
2. Выберите на ленте **Создание → Формы → Другие формы → Сводная таблица** (Create → Forms → More Forms → PivotTable).
3. На экране появится стандартное окно конструктора сводной таблицы.

4. Перетащите поля для создания сводной таблицы.

5. Выберите кнопку **Office → Сохранить,** когда будете готовы сохранить вашу сводную таблицу (или просто закройте форму, а программа Access предложит сохранить ее). В любом случае вы должны задать имя формы.

6. Выбирайте такое имя, которое ясно показывает, что форма - сводная таблица, например, **SalesPivotTable.** Позже ее можно открыть в области переходов, дважды щелкнув форму кнопкой мыши.

9.3.4. *Скрытие и отображение подробностей*

Как вы уже видели, сводные таблицы - весьма полезное средство для создания подробных итоговых таблиц. Единственная проблема состоит в том, что порой сводные таблицы перегружены подробностями, даже их итоги почти также детальны, как и исходная таблица.

Рассмотрим сводную таблицу, показанную на рис. 9.17. Когда создается сводная таблица, вы видите информацию о каждом товаре и каждом географическом регионе. Но, что если вы хотите отобразить только конкретный товар, товар в определенной категории или товары в конкретной стране или штате? В данном случае хитрость заключается в скрытии всех категорий, которые вы не хотите видеть, с помощью свертывания таблицы.

Самый легкий способ раскрыть или свернуть данные - использование кнопок +/-, которые выводятся рядом с заголовками строк и столбцов (рис. 9.18). Этот метод позволяет раскрыть или свернуть все группы в конкретной строке или столбце.

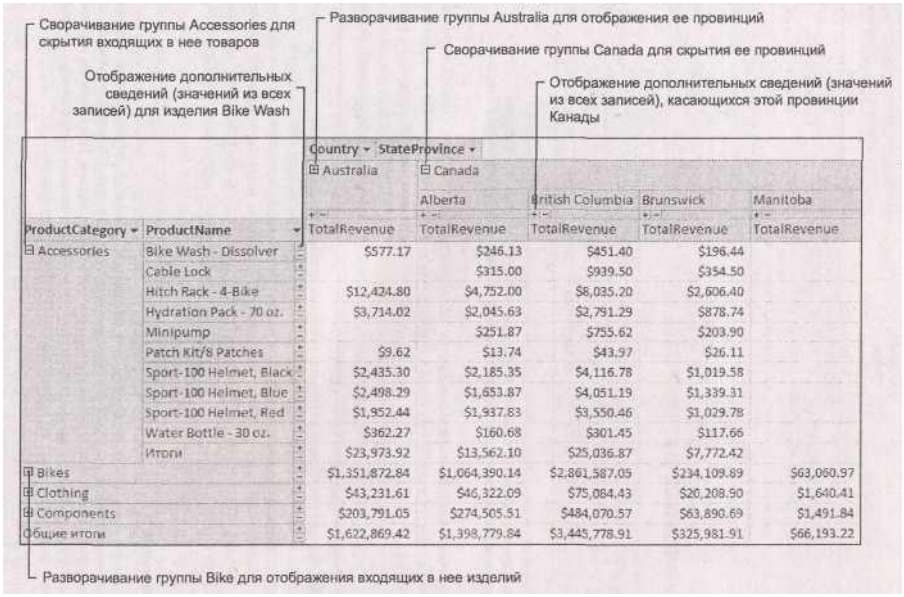
Если вы хотите сосредоточиться на конкретных данных, можно раскрыть единственную ячейку. В данном случае просто щелкните ячейку правой кнопкой мыши и выберите команду **С подробностями** (Show Details). Например, используя этот метод, можно раскрыть ячейку, в которой отображаются продажи одежды в Австралии (вместо объема продаж одежды во всех странах или объема всех продаж в Австралии).

9.3.5. *Фильтрация в сводных таблицах*

Другой способ упрощения сводных таблиц - исключение некоторых данных, участвовавших в их формировании. Для этого применяется фильтрация сводной таблицы, во многом похожая на фильтрацию на листе данных, - вы сообщаете программе Access о том, какие записи вы хотите использовать, а какие вас не интересуют.

Существует несколько способов фильтрации. Два самых быстрых варианта фильтрации - выбор элементов, которые вы хотите видеть, из списка. Далее перечислены эти варианты.

■ Можно отобрать записи, используя поля, которые группируют строки и столбцы вашей сводной таблицы. Например, с помощью такого варианта фильтрации можно скрыть страны или категории товаров, которые вас не интересуют. Для применения фильтрации просто щелкните мышью стрелку раскрывающегося списка справа от заголовка соответствующего поля (рис. 9.19). Далее сбросьте флажок, расположенный рядом с каждым элементом, который вы не хотите



		Country	Australia	Canada			
		StateProvince		Alberta	British Columbia	Brunswick	Manitoba
ProductCategory	ProductName		TotalRevenue	TotalRevenue	TotalRevenue	TotalRevenue	TotalRevenue
Accessories	Bike Wash - Dissolver		$577.17	$246.13	$451.40	$196.44	
	Cable Lock			$315.00	$939.50	$354.50	
	Hitch Rack - 4-Bike		$12,424.80	$4,752.00	$8,035.20	$2,606.40	
	Hydration Pack - 70 oz.		$3,714.02	$2,045.63	$2,791.29	$878.74	
	Minipump			$251.87	$755.62	$203.90	
	Patch Kit/8 Patches		$9.62	$13.74	$43.97	$26.11	
	Sport-100 Helmet, Black		$2,435.30	$2,185.35	$4,116.78	$1,019.58	
	Sport-100 Helmet, Blue		$2,498.29	$1,653.87	$4,051.19	$1,339.31	
	Sport-100 Helmet, Red		$1,952.44	$1,937.83	$3,550.46	$1,029.78	
	Water Bottle - 30 oz.		$362.27	$160.68	$301.45	$117.66	
	Итоги		$23,973.92	$13,562.10	$25,036.87	$7,772.42	
Bikes			$1,351,872.84	$1,064,390.14	$2,861,587.05	$234,109.89	$63,060.97
Clothing			$43,231.61	$46,322.09	$75,084.43	$20,208.90	$1,640.41
Components			$203,791.05	$274,505.51	$484,070.57	$63,890.69	$1,491.84
Общие итоги			$1,622,869.42	$1,398,779.84	$3,445,778.91	$325,981.91	$66,193.22

включать в сводную таблицу. Это похоже на сворачивание частей сводной таблицы, с одной лишь разницей - вся отфильтрованная информация исчезает бесследно, не остается даже итогов.

Рис. 9.18. Используйте кнопку со знаком "плюс" (+) для отображения подробностей свернутой группы и кнопку со знаком "минус" (-) для свертывания развернутой группы. В данной сводной таблице все группы товаров свернуты за исключением группы Accessories. Также свернута страна Австралия, поэтому отображаются только итоги (а не распределение по регионам)

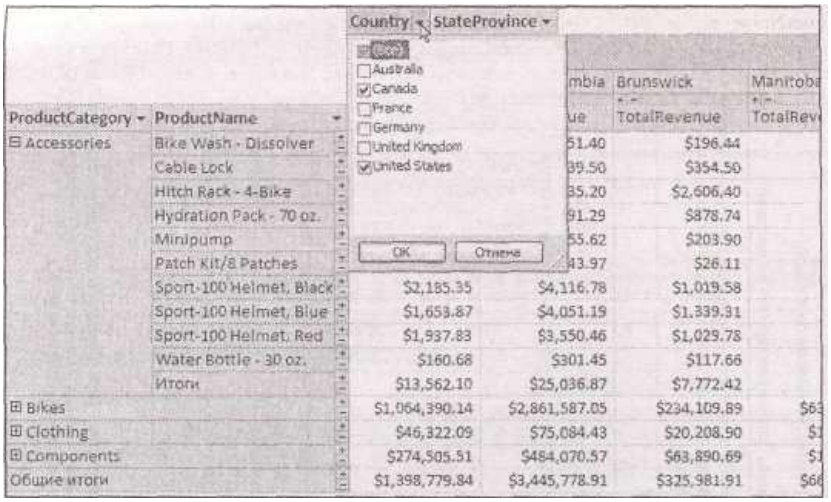


Рис. 9.19. Вариант быстрой фильтрации позволяет скрыть определенные элементы, не желательные в вашей сводной таблице. Когда применяется этот сорт фильтрации, стрелка раскрывающегося списка для соответствующего поля меняет цвет с черного на синий

■ Можно отбирать записи, используя другие поля в исходной таблице. Просто перетащите их из списка полей сводной таблицы в область над сводной таблицей **Перетащите сюда поля фильтра** (Drop Filter Fields Here). После того как поле фильтра вставлено, рядом с его заголовком появляется раскрывающийся список. Щелкните кнопкой мыши стрелку для отображения списка всех значений и сбросьте флажок, расположенный рядом со значениями, которые вы не хотите видеть.

Программа Access позволяет отобрать максимальные и минимальные значения вычисляемых величин в любой группе. Например, можно применить этот метод для того, чтобы скрыть плохо продаваемые товары. Для этого щелкните правой кнопкой мыши заголовок поля **ProductName** и выберите вариант из подменю **Показать верхние и нижние элементы** (Show Top/Bottom Items). Возможно, вы хотите увидеть конкретное число товаров (наилучшие или наихудшие 1, 10, 25 и т. д.) или процентное выражение (наилучший 1%, наилучшие 10% и т. д.). На рис. 9.20 показан пример.

Примечание

Когда задана фильтрация верхних/нижних элементов, справа от заголовка поля появляется пиктограмма **Автофильтр** (AutoFilter) (она выглядит как воронка). Переместите указатель мыши над пиктограммой, чтобы установить, какая фильтрация задана. Для удаления установленной фильтрации щелкните мышью пиктограмму и выберите команду **Автофильтр**. Для восстановления фильтрации в любой момент выберите эту команду еще раз.

Фильтрацию верхних/нижних элементов легко применять, но если у вас несколько уровней группировки, нужно тщательно выбирать место применения фильтрации. Например, рассмотрим сводную таблицу, показанную на рис. 9.20, в которой продажи разделены по категориям и названиям товаров. Если применить фильтр "верхние/нижние" к полю

ProductName, вы увидите 1% наилучших товаров из всех изделий. Но если однопроцентный фильтр применить к полю ProductCategory, вы увидите наилучший 1% из всех категорий. Другими словами, вы сосредоточитесь на категориях, имеющих максимальные объемы продаж, а

не на самых ходовых товарах.

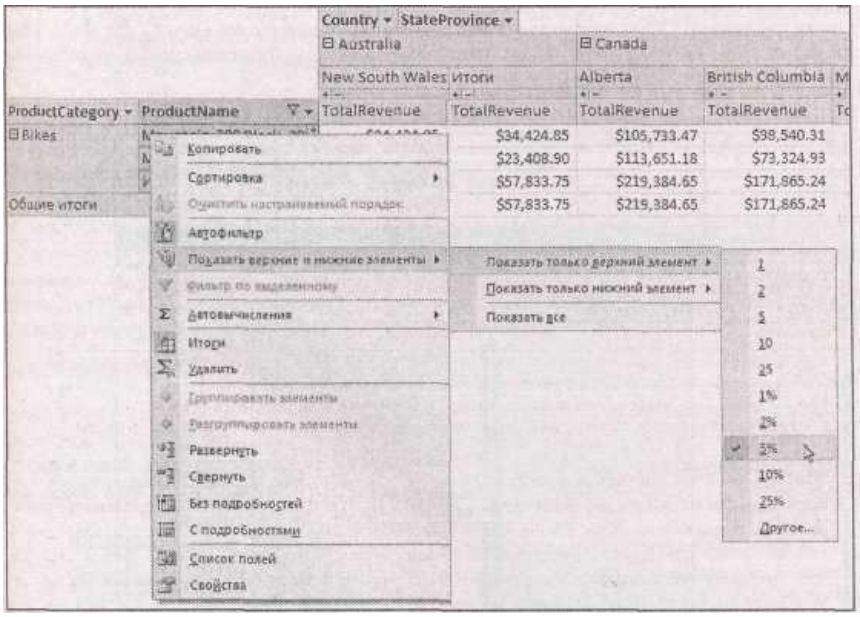


Рис. 9.20. Сводная таблица сокращена до пяти процентов самых популярных товаров. Если есть категории, не содержащие товары из этого диапазона, эти категории не будут отображаться вообще

Для того чтобы понять разницу, представьте, что произойдет, если в категорию Components входит большое количество медленно продаваемых изделий, которые, будучи просуммированы, представляют внушительную величину. Когда вы примените фильтр к полю **ProductCategory,** то увидите все товары в этой категории с высокой торговой эффективностью. Если же применить фильтр к полю **ProductName,** вы сосредоточитесь на самых популярных товарах и категориях, их содержащих. В данном случае центром внимания станет категория Clothing с несколькими самыми ходовыми товарами.

Подсказка

Есть опасность затянуть на себе узел, применив слишком много условий фильтрации одновременно. Если вы забыли, какие условия отбора задали, их можно отключить все сразу, выбрав на ленте **Работа со сводными таблицами | Конструктор → Фильтр и сортировка → Автофильтр (**PivotTable Tools | Design → Filter & Sort → AutoFilter).

9.4. *Сводные диаграммы*

Программа Access позволяет создавать диаграммы, основанные на данных сводной таблицы. На самом деле с каждым представлением сводной таблицы связано представление сводной диаграммы. Для переключения из режима сводной таблицы на диаграмму, отображающую ваши результаты в графической форме, выберите на лепте **Работа со сводными таблицами | Конструктор → Режим → Сводная диаграмма** (PivotTable Tools | Design → View → PivotChart View) или воспользуйтесь кнопками режимов в правом нижнем углу окна программы.

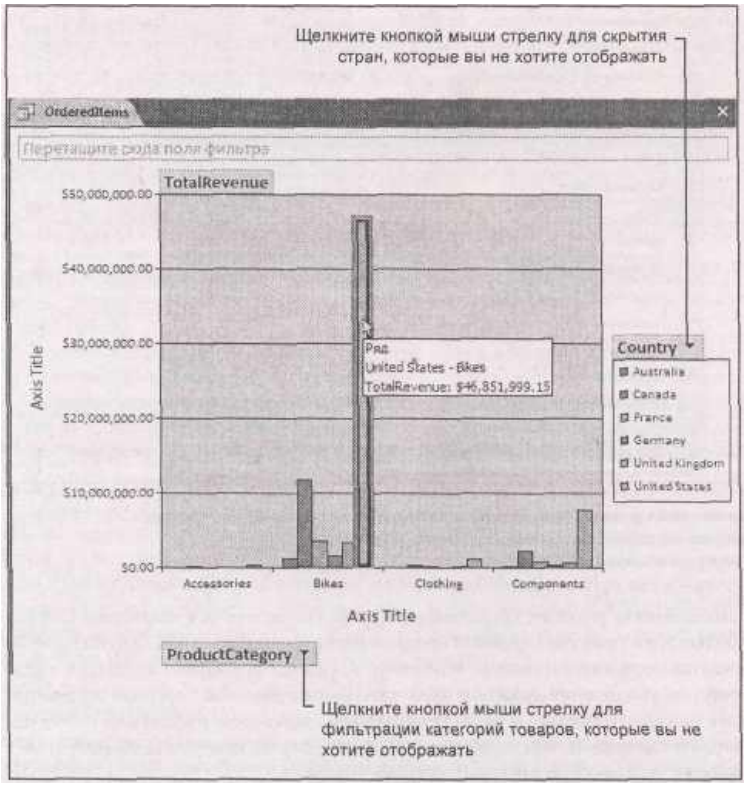


Рис. 9.21. На этой сводной диаграмме показана сводная таблица, разделенная по категориям товаров в строках и на страны в столбцах. Каждая категория строк отображается как группа расположенных рядом столбцов. Можно переместить указатель мыши над столбцом и увидеть всплывающую подсказку с дополнительными сведениями о нем. В этом примере выбранный в данный момент столбец (самый высокий), отображающий объем продаж велосипедов в США, превышает объемы продаж во всех группах

В случае сводной таблицы товаров из примера, описанного ранее в этой главе, сводная диаграмма позволяет легко увидеть группы, выделяющиеся объемами продаж. Нужно просто найти самые высокие столбцы на диаграмме, показанной на рис. 9.21.

Подсказка

Выберите на ленте **Работа со сводными таблицами | Конструктор → Показать или скрыть → Легенда** (PivotChart Tools | Design → Show/Hide → Legend) для вывода на экран блока с условными обозначениями ваших групп (legend box).

Сводные диаграммы интерактивны, как и сводные таблицы. Если посмотреть внимательно на сводную диаграмму, то можно увидеть заголовки полей, выбранные для элементов строк, столбцов и данных, которые отображаются непосредственно на диаграмме. Вы можете использовать заголовки полей для изменения данных, которые выводятся на экран, реорганизации уровней группировок или применения фильтрации без выхода из режима сводной диаграммы. Например, на рис. 9.21, если нужно отобразить меньше стран, просто щелкните кнопкой мыши заголовок поля Country в правой части диаграммы. На экране появится список стран с установленными флажками рядом с каждой страной, которая выбрана для вывода на экран. Если сбросить флажок, страна исчезнет из сводной диаграммы и лежащей в основе сводной таблицы.

Сводные диаграммы не так полезны, как кажется на первый взгляд. Одна из проблем состоит в том, что подробные сводные данные не всегда можно эффективно отобразить на чертеже. Если у вас большое число групп (например, данные сгруппированы по названию товара или по городу клиента, как в приведенных ранее примерах), то в результате вы получите десятки столбцов, тесно прижатых друг к другу, и не сможете прочесть блок условных обозначений, чтобы понять, какой столбец какую группу представляет.

Подсказка

Перед созданием сводной диаграммы часто полезно ограничить количество информации в

сводной таблице. Слишком большое количество данных может привести к большой загруженности диаграммы и трудности ее восприятия. Легче всего скрыть данные, если не применять слишком много уровней группировок и ограничиться только интересующими вас группами с помощью фильтрации, как описано в предыдущем разделе.

Выбор типа диаграммы

Еще одно ограничение, связанное со сводными диаграммами, - малое число вариантов визуализации данных. Изменить тип диаграммы можно, щелкнув диаграмму правой кнопкой мыши и выбрав команду **Изменить тип диаграммы** (Change Chart Type). На экране появляется коллекция разных вариантов. Но большинство диаграмм в этой коллекции, начиная от круговых и заканчивая графиками, не могут сформировать приличного отображения ваших данных с большим количеством групп. В действительности стоит опробовать только три приемлемых варианта.

■ *Гистограмма с накоплением* (stacked bar or column chart) формирует столбец для каждой группы и затем делит его для представления подгрупп (рис. 9.22).

■ *Нормированная гистограмма с накоплением* очень похожа на обычную гистограмму с накоплением, за исключением того, что каждый прямоугольник растянут на всю высоту диаграммы. В этом случае вы действительно сможете сравнивать подгруппы (рис. 9.23).

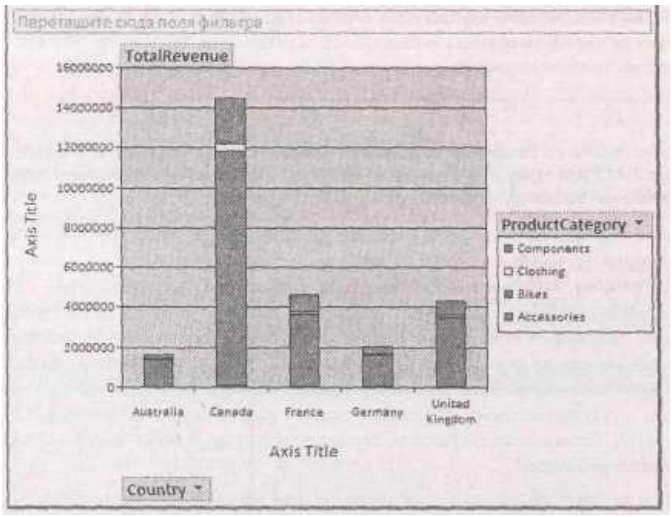


Рис. 9.22. В гистограмме с накоплением группа каждой строки - отдельный прямоугольник. Затем прямоугольник делится на группы столбцов таблицы. В данном примере это означает, что у вас есть один прямоугольник для каждой страны и отдельные области прямоугольника представляют объемы продаж в разных категориях товаров для этой страны. Гистограмма с накоплением облегчает сравнение разных категорий. Очевидно, что велосипеды лучше всего продаются во всех странах

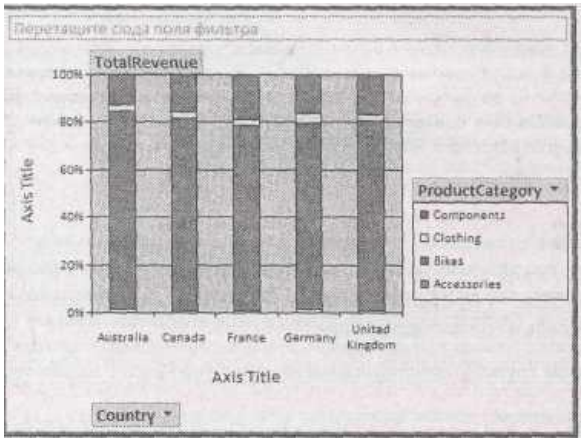


Рис. 9.23. В нормированной гистограмме с накоплением нельзя определить страну с максимальным объемом продаж, но можно сравнить распределение продаж по категориям. Например, можно установить, в какой стране максимальная выручка от продаж велосипедов. (В

данном примере, такой страной оказывается Австралия, но, как ни странно, в других странах похожая картина)

- *Объемная гистограмма в* основном похожа на

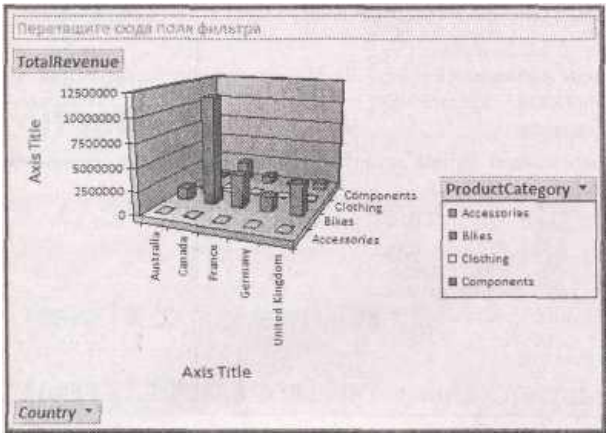


Рис. 9.24. На объемной гистограмме страны упорядочены слева направо, а все категории товаров помещаются от переднего края диаграммы к заднему. К сожалению, нельзя выбрать, какую категорию поместить на переднем плане, а какой закончить на заднем плане - порядок расположения алфавитный

Подсказка

Для поворота объемной диаграммы щелкните кнопкой мыши в свободном месте диаграммы. Затем выберите на ленте **Работа со сводными таблицами | Конструктор → Сервис → Страница свойств** для отображения окна **Свойства** (Properties). Перейдите на вкладку **Объем** (3D View), содержащую множество ползунков, которые можно передвигать для получения разных ракурсов представления данных.

9.4.1. *Печать сводной диаграммы*

Если вы хотите напечатать сводную диаграмму, используйте последовательность команд **Office → Печать** (или кнопка **Office → Предварительный просмотр** для того, чтобы сначала внимательно посмотреть на то, как будут выглядеть ваши результаты).

Если у вас нет цветного принтера, визуальное разделение групп может оказаться трудной задачей. Можно выбрать определенные цвета для всех групп, но это потребует дополнительных усилий. Вот как это делается.

1. Щелкните кнопкой мыши конкретную группу где-нибудь на диаграмме (например, группу Bikes в столбце Australia).
2. Сделайте паузу и снова щелкните мышью эту группу для того, чтобы выделить ее во всех столбцах гистограммы. Например, если дважды щелкнуть кнопкой мыши группу Bikes в столбце Australia (Австралия), группа Bikes будет выделена во всех странах, а именно ее вы и хотите изменить.
3. Выберите на ленте **Работа со сводными таблицами | Конструктор → Сервис → Страница свойств** (PivotChart Tools | Design → Tools → Property Sheet) для вывода на экран окна **Свойства.**
4. Выберите вкладку **Границы и заливка** (Border/Fill). На ней вы найдете параметры, позволяющие задать толщину и цвет границ вокруг столбца и цвет (или узор) для заполнения внутренней области столбца.
5. Повторите этот процесс для каждой группы, которую хотите изменить, пока не получите приемлемый набор подходящих для вывода на печать цветов.

Часть III

Отчеты

Глава 10. Создание отчетов

Глава 11. Проектирование сложных отчетов

10. Глава 10. Создание отчетов

Для создания твердой копии ваших тщательно сохраняемых в Access данных есть множество причин. Имея хорошую распечатку, вы можете:

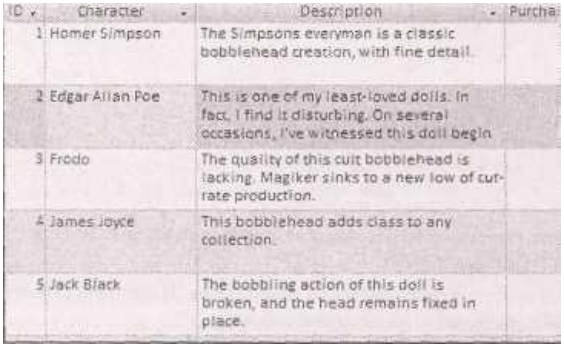
- брать с собой информацию без транспортировки своего компьютера. Например, можно захватить с собой инвентарный список, когда идете за покупками;
- показать свою информацию тем, кто не работает с программой Access. Например, можно раздать каталоги товаров, формы заказов и списки классов другим людям;
- просмотреть подробности вне офиса. Например, можно искать ошибки в пригородной электричке по дороге домой;
- произвести впечатление на начальство. Помимо всего прочего трудно аргументировать, имея 286 страниц необработанных данных.

В *главе 3* вы научились печатать исходные данные таблицы непосредственно с листа данных. Этот метод удобен, но у него всего лишь несколько параметров настройки. Вы лишены гибкости при обработке больших блоков информации, возможности тонкой настройки параметров форматирования разных полей и средств группировки и подведения итогов, облегчающих анализ информации. Как вы уже, вероятно, догадались, программа Access предлагает другое средство вывода на печать, восполняющее перечисленные пробелы. Оно называется *отчетом* и позволяет создавать полностью настроенную твердую копию, сообщающую программе Access точные параметры подготовки данных для вывода на печатающее устройство.

Отчеты - это специализированные объекты БД, похожие во многом на таблицы и запросы. В результате вы можете подготовить нужное вам количество отчетов и хранить их под рукой неограниченно долго. Жизнь не покажется медом, если ограничиваться только листом данных. Например, если вы пользуетесь БД о куклах-болванчиках, у вас может возникнуть желание напечатать инвентарный список кукол с указанием имени куклы и сведений об изготовителе и отдельный список с ценами для составления сметы расходов. Переключаясь многократно между этими двумя выводами на печать и листом данных, вы должны будете каждый раз вручную изменять порядок столбцов и скрывать их. У отчетов нет этих проблем, поскольку каждый из них сохраняется как отдельный объект БД. Итак, если вы хотите напечатать вашу инвентарную опись, то просто запустите отчет **DollInventory.** Если нужны подробности сметы, выполните отчет **DollPrices.**

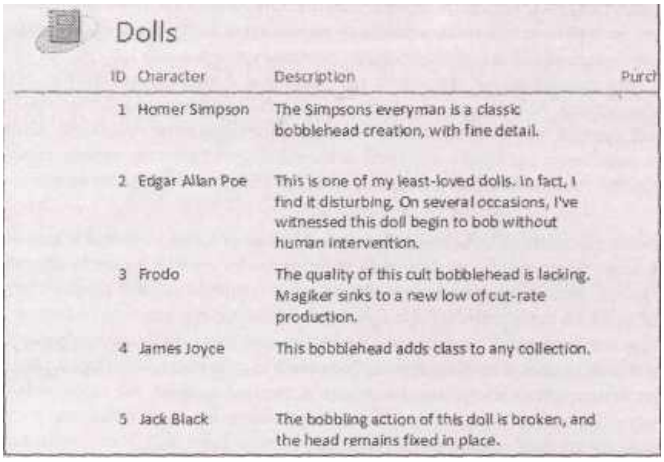
Примечание

Идеология та же, что и в случае запросов, с которыми вы познакомились в *главе 6.* Вместо одного набора параметров сортировки и фильтрации запросы позволяют приготовить все комбинации, которые вам когда-либо понадобятся, и затем сохранить каждую из них как отдельный объект БД.



ID	Character	Description	Purcha
1	Homer Simpson	The Simpsons everyman is a classic bobblehead creation, with fine detail.	
2	Edgar Allan Poe	This is one of my least-loved dolls. In fact, I find it disturbing. On several occasions, I've witnessed this doll begin	
3	Frodo	The quality of this cult bobblehead is lacking. Magiker sinks to a new low of cut-rate production.	
4	James Joyce	This bobblehead adds class to any collection.	
5	Jack Black	The bobbling action of this doll is broken, and the head remains fixed in place.	

Рис. 10.1. Обычные распечатки, как всем известно, плохи при обработке больших объемов данных в одном столбце. Посмотрите на поле **Description** в таблице **Dolls.** У всех записей под описание отводится область одного размера. Если объем данных больше доступного пространства (как в случае куклы Эдгара Алана По), информация в конце обрезается. Если данных меньше (как в случае куклы Джеймса Джойса), вы будете любоваться зияющими пустотами



Dolls

ID	Character	Description	Purch
1	Homer Simpson	The Simpsons everyman is a classic bobblehead creation, with fine detail.	
2	Edgar Allan Poe	This is one of my least-loved dolls. In fact, I find it disturbing. On several occasions, I've witnessed this doll begin to bob without human intervention.	
3	Frodo	The quality of this cult bobblehead is lacking. Magiker sinks to a new low of cut-rate production.	
4	James Joyce	This bobblehead adds class to any collection.	
5	Jack Black	The bobbling action of this doll is broken, and the head remains fixed in place.	

Рис. 10.2. В типичном отчете вы задаете ширину столбцов, а высота каждой строки зависит от количества информации в записи. Это означает, что каждая строка достаточно велика для того, чтобы вместить весь текст из поля **Description.** Самое замечательное в том, что не нужно задавать специальные параметры для обеспечения подобного поведения. Отчеты делают это автоматически

Для того чтобы увидеть невероятные преимущества отчетов по сравнению с обычными распечатками листа данных, сравните рис. 10.1 (на котором показана распечатка листа данных) и рис. 10.2 (на котором те же данные помещены в простой отчет). Обратите внимание на то, что на распечатке листа данных есть и потерянное пространство, и пропущенная информация.

10.1. *Базовые сведения об отчетах*

Есть несколько способов создания отчетов. Опытные разработчики отчетов (какими вы станете, прочитав эту главу) часто самостоятельно создают отчет, не прибегая к помощи мастера. Новички (к которым пока относитесь и вы) обычно быстро формируют отчет с помощью одного щелчка мыши. В данном разделе рассказывается о простейшем методе построения отчетов.

Примечание

Как вы увидите, простой способ создания отчета всегда помещает данные в табличную структуру (со строками и столбцами). Вы научитесь избавляться от этого дизайна в главе 11.

10.1.1. *Создание простого отчета*

Создание простого отчета требует двух шагов и еще несколько понадобится для его тонкой настройки. Если хотите опробовать этот метод самостоятельно, откройте БД Boutique Fudge (включенную в загружаемое содержимое для данной главы, описанное в *разд. "Примеры" во введении)* или собственную БД и выполните следующие действия.

1. В области переходов выберите таблицу, которую хотите использовать для своего отчета.

В данном примере применяется таблица **Products** из БД Boutique Fudge. Можно также создать отчет на базе запроса. Об этом приеме см. дополнительную информацию в *примечании. "На профессиональном уровне. Выполнение тяжелой работы с помощью запроса " далее в этом разделе).*

2. Выберите на ленте **Создание → Отчеты → Отчет** (Create → Reports → Report).

На экране появится новая вкладка с простым автоматически сгенерированным отчетом. Этот отчет организует информацию в виде таблицы, в которой каждое поле исходной таблицы (или запроса) занимает отдельный столбец. Отображение отчета похоже на лист данных за исключением улучшенного форматирования и более эффективного использования пространства, как показано на рис. 10.2.

Когда отчет создается впервые, поля располагаются в нем слева направо в том же порядке, что и в исходной таблице. На него не повлияет реорганизация столбцов на листе данных. Но любые столбцы, скрытые на листе данных, исчезнут из отчета.

Примечание

Вы можете точно определить данные в вашем отчете, удаляя ненужные столбцы и вставляя новые. В *разд. "Добавление и удаление полей" далее в этой главе* приведены дополнительные

сведения об этом приеме.

3. Измените ширину столбцов, увеличивая ее или уменьшая до тех пор, пока не добьетесь нужного баланса.

Для изменения ширины столбца сначала щелкните кнопкой мыши его заголовок, чтобы выбрать столбец. (Вокруг столбца появится точечная линия.) Далее переместите мышь к правому краю ячейки с заголовком столбца так, чтобы указатель превратился в двунаправленную стрелку. И, наконец, с нажатой кнопкой мыши перетащите границу столбца влево (для сужения) или вправо (для расширения). На рис. 10.3 показан этот процесс.

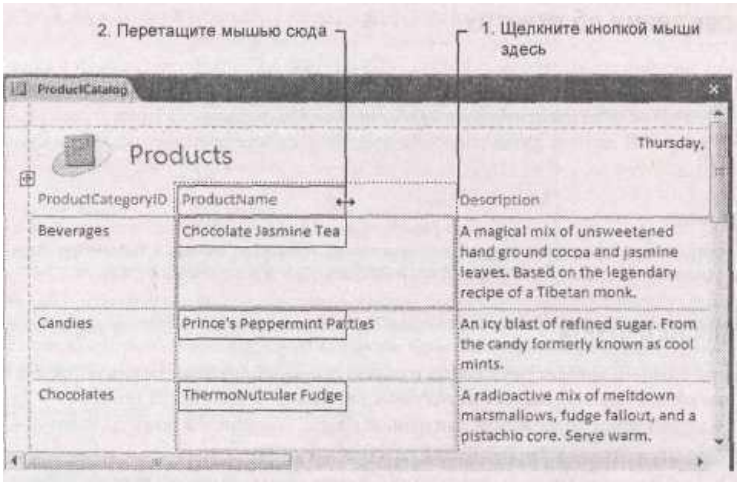


Рис. 10.3. Перетащите край столбца для получения нужной ширины. Черный прямоугольник показывает новую ширину. Когда вы отпустите кнопку мыши, программа Access изменит ширину столбца и, соответственно, передвинет все последующие столбцы. Для того чтобы последний столбец не ушел за пределы окна, возможно, после расширения некоторых столбцов другие придется немного сузить

Примечание

На правой стороне отчета может появиться точечная линия, обозначающая край страницы. Вы можете изменить ширину столбца так, что он выйдет за край страницы. Это сделать имеет смысл, если у вас десятки столбцов и единственный способ справиться с ними - распечатать таблицу, занимающую две страницы по ширине. Тем не менее обычно лучше разместить все поля на одной странице, повернув ее набок с помощью альбомной ориентации *(см. разд. "Изменение макета страницы" главы 3),* если нужно расположить дополнительные столбцы.

4. Задайте нужный порядок следования столбцов, перемещая их с помощью мыши на новое место.

Для перемещения столбца щелкните кнопкой мыши его заголовок и затем перетащите столбец с нажатой кнопкой мыши в новое место.

Подсказка

Столбцы можно передвигать и с помощью клавиатуры. Просто щелкните кнопкой мыши для выбора нужного столбца, а затем используйте клавиши <?> или <→> для перепрыгивания с одного места на другое.

5. При желании можно откорректировать форматирование, сменив шрифты, цвета и границы.

Самый быстрый способ изменения форматирования вашего отчета - выбор подходящей части (с помощью щелчка мышью) и использование кнопок на ленте **Работа с макетами отчетов | Формат → Шрифт** (Report Layout Tools | Formatting → Font). С помощью этого метода можно изменить способ отображения заголовков, имен столбцов и данных. Дополнительную информацию о нем можно найти в *разд. "Форматирование столбцов и заголовков столбцов" далее в этой главе.*

6. Добавьте завершающий штрих.

Сейчас самое время изменить заголовки, вставить эмблему и номера страниц. Вы узнаете как добавлять эти элементы в *разд. "Создание пустого отчета" далее в этой главе.*

7. Для печати отчета можно выбрать последовательность: кнопка **Office → Печать** (Office → Print).

Отрегулировать параметры печати можно и в режиме **Предварительный просмотр** (выберите **Office → Печать → Предварительный просмотр,** как описано в *разд. "Печать, предварительный просмотр и экспорт отчета" далее в этой главе).*

8. Сохраните ваш отчет для дальнейшего использования.

Сохранить отчет можно в любое время, нажав комбинацию клавиш <Ctrl>+<S>. Если закрыть вкладку отчета без сохранения, программа Access сообщит о необходимости сохранения отчета. В любом случае вам придется задать имя отчета.

Возможно создание отчетов с тем же именем, что и таблицы или другие объекты БД. Например, можно создать отчет **Products,** отображающий информацию из таблицы **Products.** Но на практике лучше выбирать для отчета более точное имя (например, **Products By Category** (товары по категориям), **ProductListForDealers** (список товаров для дилеров) и **Top50Products** (50 самых популярных товаров)). Отчет, приведенный на рис. 10.2, назван **ProductCatalog** (каталог товаров),

На профессиональном уровне.

Выполнение тяжелой работы с помощью запроса

Самый очевидный способ построения отчета - создание его на базе существующей таблицы. Но можно также создать отчет на базе запроса. Этот метод позволяет использовать мощные средства фильтрации и сортировки записей перед включением их в отчет. Применять его также имеет смысл при создании отчета, использующего информацию из нескольких таблиц.

Предположим, что вы решили создать список изделий, включающий дополнительные сведения из другой таблицы (такие как описание категории изделия из таблицы **ProductCategories).** Несмотря на то, что можно создать такой отчет с нуля, часто гораздо разумнее сначала структурировать данные с помощью запроса. В этом случае вы сможете повторно использовать запрос для различных целей (например, редактирования) и изменить его в любое время.

В данном примере сначала надо создать запрос, объединяющий **Categories** (категории) и таблицу **Products** *(см. разд. "Запросы и связанные таблицы" главы 6).*

Затем следует сохранить запрос, выделить его в области переходов и выбрать на ленте **Создание → Отчеты → Отчет** (Create → Reports → Report) для создания отчета на базе запроса. В дальнейшем усовершенствовать ваш отчет можно обычным способом.

10.1.2. *Компоновка отчета*

Вы уже знаете, как перемещать столбцы в отчете. Но передвигать можно не только столбцы. Можно раздвинуть строки (рис. 10.4) и отрегулировать настройку следующих элементов.

■ Эмблема (в левом верхнем углу). В новом отчете она выглядит как тетрадь с охватывающим кругом.

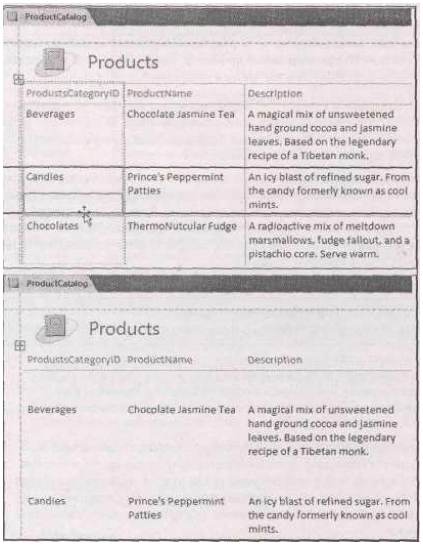


Рис. 10.4. *Вверху:* для того чтобы раздвинуть строки, щелкните кнопкой мыши значение в строке и перетащите его вниз. *Внизу:* все строки выровнены и у всех одинаковая разбивка

- *Заголовок отчета* (непосредственно рядом с эмблемой). Для начала им может стать имя таблицы или запроса, на базе которых создается отчет (например, **Products).**
- *Дата и время* (которые обновляются при каждом открытии отчета). Первоначально они выводятся в правом верхнем углу.
- *Номер страницы.* Он выводится внизу каждой страницы, в центре. В **Режиме макета** (Layout) программа Access считает, что все данные, включенные в отчет, занимают одну страницу, поэтому придется воспользоваться прокруткой, чтобы увидеть этот элемент в конце страницы.
- *Данные отчета* (после заголовка). Для изменения первоначального расположения таблицы на странице отчета щелкните кнопкой мыши один из заголовков столбцов и перетащите его вниз (для того чтобы отодвинуть заголовок отчета от данных) или вверх (для того чтобы удалить свободное пространство между ними).
- *Итоги* (внизу, в некоторых столбцах). Программа Access автоматически добавляет итоги для числовых полей. Например, когда впервые создается отчет **ProductCatalog.** Access вставляет итог в конец столбца **Price,** указывающий для всех видов изделий стоимость покупки одного изделия. (Это сомнительный итог - для его замены выберите столбец и затем укажите другой вариант групповой операции из меню **Работа с макетами отчетов | Формат → Группировка и итоги → Итоги** (Report Layout Tools | Formatting → Grouping & Totals → Totals).)

Подсказка

Большинство элементов отчета можно удалить, если выделить их и затем нажать клавишу <Delete>. Этим приемом удобно пользоваться, если не нужно выводить на экран номера страниц, даты и итоги.

10.1.3. *Добавление и удаление полей*

Если вы устали от простого перемещения столбцов, возможно, вам захочется добавить столбцы, еще не включенные в отчет, или удалить ненужные вам. Удалить поле легко: просто щелкните его кнопкой мыши для выделения и нажмите клавишу <Delete>. (Можно опробовать этот прием на поле **Discontinued** (снятые с производства) в отчете **ProductCatalog.)**

Когда создается простой отчет с помощью метода быстрого создания, описанного ранее в этой главе, в конечном итоге в отчет, как правило, включаются все необходимые поля. Но есть две причины, по которым может возникнуть необходимость вставки дополнительного поля, до сих пор не включенного в отчет:

- вы хотите добавить поле, скрытое в **Режиме таблицы** *(см. разд. "Скрытие столбцов" главы 3).* Когда создается новый отчет, скрытые поля пропускаются;
- вы хотите добавить поле со связанной информацией из подчиненной таблицы. Например, можно добавить поля из таблицы **ProductCategories** для отображения информации о категории, к которой принадлежит каждое изделие.

Для того чтобы вставить новое поле, понадобится помощь панели **Список** полей (Field List) (рис. 10.5). Для ее вывода выберите **Работа с макетами отчетов | Формат → Элементы управления → Добавить поля** (Report Layout Tools | Formatting → Controls → Add Existing Fields).

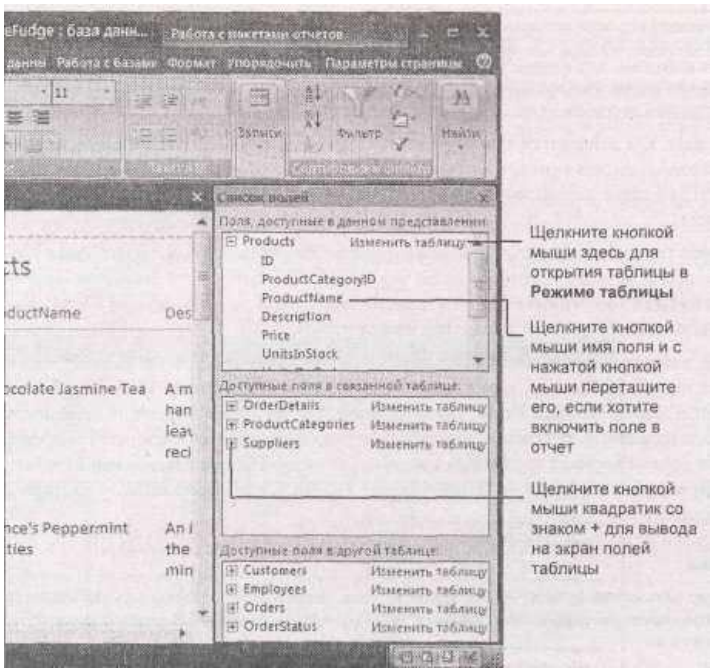


Рис. 10.5. В верхней части окна **Список полей** перечислены поля из таблицы (или запроса), на которой основан отчет. В средней части приведены поля из любых связанных таблиц и в нижней - несвязанные таблицы (которые вы, возможно, и не будете использовать). Для вставки поля перетащите его из окна **Список полей** и положите на свой отчет

Когда вы добавляете новое поле, программа Access использует имя поля для заголовка столбца, что не всегда совпадает с вашим желанием. Возможно, вы предпочтете **Product Name** (с пробелом между словами), а не **ProductName.** Или захотите укоротить имя **ProductCategoryID** до просто **Category.** В конце концов, в отчете выводится имя вместо числового идентификатора категории товара, поскольку в поле **ProductCategoryID** применяется подстановка *(см. разд. "Поиск в связанных таблицах" главы 5). К* счастью, переименовать столбцы очень легко. Просто щелкните дважды кнопкой мыши заголовок столбца для переключения в режим редактирования. Теперь можно откорректировать существующий текст или полностью заменить его.

Часто задаваемый вопрос.

Добавление изображений в отчеты.

Можно ли хранить изображения в таблицах и отображать их в отчетах?

Во многие таблицы включаются встроенные изображения с помощью типа данных **Вложение.**

Эту методику можно применять для хранения фотографий сотрудников, изображений изделий или эмблем поставщиков. Возможно, вам захочется включить изображения и в ваши распечатки, но это зависит от типа графического файла.

Ваши изображения в отчете можно выводить на экран и даже распечатывать при соблюдении следующих требований.

• Изображение хранится в поле типа **Вложение** (см. дополнительную информацию о типе данных **Вложение** в *разд. "Вложение" главы 2).*

• Ваше изображение хранится в стандартном графическом формате (скажем BMP, JPG, GIF, TIF, WMF и т. д.). Если в поле типа **Вложение** у вас файл другого форма та, вы увидите в вашем отчете пиктограмму связанного с ним приложения (например, приложения Microsoft Word для файла с расширением doc).

• Ваше изображение - первое вложение. Если в поле хранится несколько вложений, то при выборе строки в отчете над ней появляются крошечные кнопки со стрелками, которые можно использовать для перехода от одного изображения к другому. В этом случае вам придется проделать это со всеми записями, прежде чем вы напечатаете

отчет.

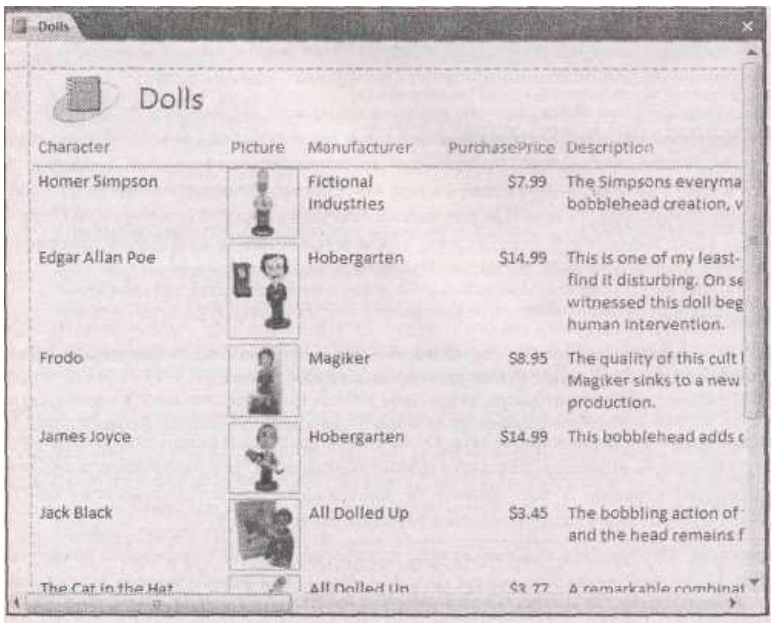


Рис. 10.6. Этот отчет можно увидеть в БД Bobblehead, приведенной как пример к данной главе

Таблица **Dolls** в БД о куклах-болванчиках удовлетворяет этим требованиям, что позволяет создать отчет, приведенный на рис. 10.6.

Альтернативный вариант - отображение в отчете имени файла или типа файла вложения. Для этого нужно использовать окно **Список полей** (см. рис. 10.5). Например, у вас есть поле типа **Вложение,** названное **Picture,** в окне **Список полей** рядом с ним расположена кнопка со знаком "плюс". Щелкните мышью эту кнопку, и вы увидите три относящихся к изображению характеристики, которые можно отобразить в отчете: **Picture.FileData** (содержимое вложения, представляющее собой изображение), **Picture.FileName** (имя файла) и **Picture.FileType** (тип файла). Если вы хотите включить в отчет эти сведения, просто перетащите их в область отчета с помощью мыши.

10.1.4. *Разные режимы отображения отчета*

Как и в случае таблиц и запросов, для изменения отчета можно использовать несколько различных режимов. Когда отчет формируется методом быстрого создания, описанным ранее в этой главе, вы начинаете с **Режима макета -** идеальной отправной точки для разработчиков отчетов. Но в зависимости от ближайшей задачи вы можете переключиться в другой режим. У вас есть четыре варианта.

■ *Режим макета.* Отображает представление отчета, предназначенное для печати и заполненное реальными данными из базовой таблицы. Этот режим можно использовать для форматирования и реорганизации основных структурных блоков отчета.

■ *Представление отчета.* Выглядит так же, как режим макета, но в нем не разрешается вносить изменения в отчет. Если дважды щелкнуть кнопкой мыши отчет в области переходов, программа Access откроет его в режиме **Представление отчета,** таким образом, вы сможете увидеть данные, содержащиеся в нем, но будете лишены возможности случайно изменить его структуру. Режим **Представление отчета** обычно применяется для копирования фрагментов отчета в буфер для того, чтобы затем вставить их в другие программы (например, в Microsoft Word). На рис. 10.7 показано, как это делается.

Примечание

Если необходимо перенести все содержимое отчета, следует воспользоваться возможностями экспорта, описанными в *разд. "Экспорт отчета" далее в этой главе.*

- *Предварительный просмотр.* Отображает реальное представление отчета так же, как **Режим**

макета или **Представление отчета.** Отличие заключается в разбиении на страницы, поэтому вы можете выяснить, сколько страниц потребуется для распечатки, и где окажутся разрывы страниц. У вас также есть возможность изменить параметры печати (например, ориентацию страниц) и экспортировать полный отчет.

■ *Конструктор.* Отображает шаблон отчета, в котором можно определить различные структурные части отчета. Этот режим не так очевиден, как **Режим макета,** но он на самом деле предоставляет неограниченные возможности для настройки отчета. Специалисты, работающие в программе Access, часто начинают создавать отчет в **Режиме макета,** а затем добавляют более причудливые эффекты в **Конструкторе.** Вы ближе познакомитесь с этим режимом в *главе 11.*

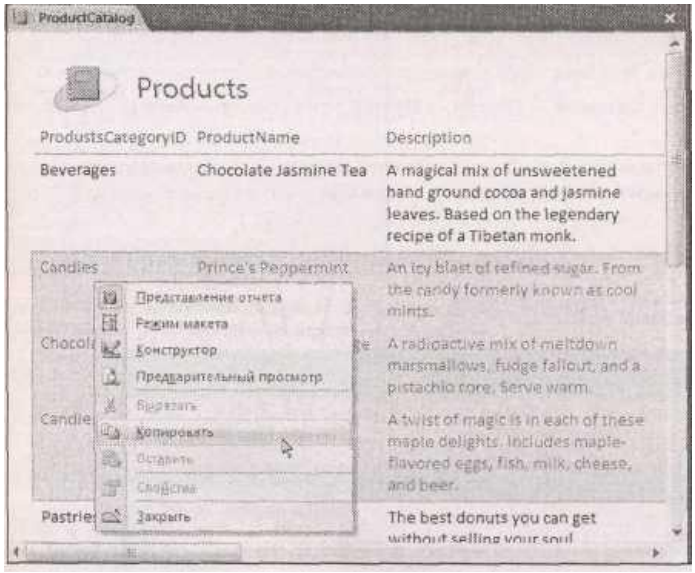


Рис. 10.7. Для выделения группы строк щелкните кнопкой мыши отступ слева от первой строки, которую хотите выделить, и затем переместите вниз мышь с нажатой кнопкой для выделения остальных строк, Далее щелкните правой кнопкой мыши выделенный фрагмент и выберите команду **Копировать** для переноса его в буфер, теперь он готов для вставки в другие приложения Windows

Примечание

Конструктор - это возврат к прежним версиям Access, в которых нет более наглядных **Режима макета** и **Представления отчета.** Он все еще полезен для решения некоторых задач, но перестал быть центральным местом формирования и форматирования отчета.

Переходить из одного режима в другой можно, щелкнув правой кнопкой мыши заголовок отчета и выбрав подходящий режим из всплывающего меню. (Можно также использовать последовательность **Главная → Режимы → Режим** или кнопки режимов в правом нижнем углу окна программы Access. Выбор зависит от личных предпочтений.)

После того как отчет закрыт, его можно открыть в выбранном вами режиме. Щелкните правой кнопкой мыши отчет в области переходов и выберите подходящий режим. Или дважды щелкните кнопкой мыши отчет в области переходов для того, чтобы открыть его в режиме **Представление отчета.**

10.1.5. *Создание пустого отчета*

Вы уже узнали, как быстро создать отчет на базе таблицы пли запроса. Но у вас есть и другая возможность - создать пустой отчет и добавить все нужные вам поля. Оба подхода правомерны. Вы можете предпочесть метод быстрого создания, когда хотите создать отчет, почти повторяющий структуру существующих таблицы или запроса. В противном случае, если вы собираетесь сформировать отчет, использующий лишь несколько полей из таблицы, возможно, легче начать с нуля.

Далее перечислены действия, необходимые для формирования отчета с чистого листа.

1. Выберите на ленте **Создание → Отчеты → Пустой отчет** (Create → Reports → Blank Report).

На экране появится новый пустой отчет в **Режиме макета.** Справа, в окне программы вы увидите **Список полей** со всеми таблицами вашей БД.

Рис. 10.8. Эмблема и заголовок обычно расположены в верхней части отчета. Для эмблемы можно использовать любое изображение, а для заголовка - любой текст. Для выбора представления даты (рис. 10.9) и номера страницы (рис. 10.10) программа Access предлагает больше вариантов

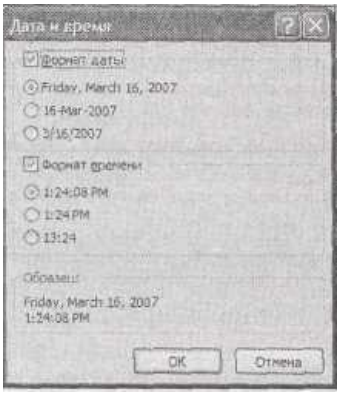


Рис. 10.9. При вставке даты можно выбрать включение даты, времени или обоих компонентов. Вы также можете выбрать формат их представления. После того как дата добавлена, можно изменить ее шрифт, границы и цвета, как и у любого другого элемента отчета

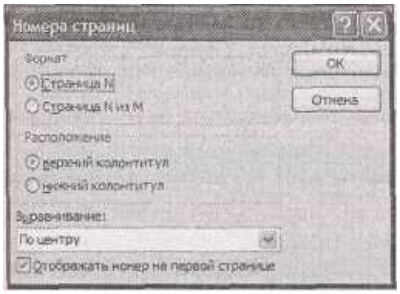


Рис. 10.10. В случае номеров страниц можно выбрать формат, местоположение и выравнивание. (Местоположение определяет вывод номера страницы над или под данными отчета. Вы можете переместить номера страниц после включения их в отчет, и программа Access сдвинет данные отчета для того, чтобы высвободить место для номеров страниц.)

2.Добавьте, какие хотите, поля из соответствующей таблицы, либо перетащив их с помощью мыши из **Списка полей** в область отчета, либо дважды щелкнув их кнопкой мыши. Можно также использовать поля из связанных таблиц. Например, вы можете создать отчет, в котором соединена информация о товаре и сведения о категории каждого товара. В этом случае отчет автоматически применяет запрос с объединением для получения результатов.

3.Отформатируйте столбцы.

Когда отчет создается с нуля, столбцы появляются без какого-либо форматирования. Вам придется воспользоваться способами форматирования, описанными в следующем разделе, для

применения цвета и средств визуального выделения.

4. Добавьте другие необходимые вам элементы, например эмблему, заголовок, номера страниц и дату.

Когда создается простой отчет, вы получаете все эти компоненты автоматически. К счастью, их также легко включить в отчет, и когда вы формируете его с пуля. Просто перейдите на ленте в группу **Работа с макетами отчетов | Формат → Элементы управления** (Report Layout Tools | Formatting → Controls) (рис. 10.8).

10.2. *Печать, предварительный просмотр и экспорт отчета*

После того как совершенный отчет создан, самое время представить его па суд зрителей. Чаще всего для этого выбирается печать.

Напечатать отчет легко - просто выберите кнопку **Office** и команду **Печать** (Print). Но прежде чем вы нечаянно отправите на печать 87-страничный список клиентов с неуклюжим шрифтом размером 24 пункта, неплохо было бы просмотреть конечный результат. Программа Access делает это с легкостью, благодаря встроенному средству **Предварительный просмотр** (Print Preview).

Подсказка

Для того чтобы напечатать отчет, его не нужно открывать. Просто выделите его в области переходов и затем выберите команду Печать из меню Office. Но учтите - когда применяется этот ускоряющий прием, вы лишены возможности предварительно просмотреть результат и должны убедиться в том, что он соответствует вашим ожиданиям до того, как он отправится на принтер.

10.2.1. *Предварительный просмотр отчета*

Для того чтобы увидеть, как будет выглядеть напечатанный отчет, щелкните правой кнопкой мыши заголовок вкладки с отчетом и выберите команду **Предварительный просмотр** или выберите **Office → Печать → Предварительный просмотр** (Office → Print → Print Preview). В режиме предварительного просмотра нельзя вносить изменения в отчет или выделять его фрагменты, Вы можете только уменьшать или увеличивать масштаб отображения отчета и переходить со страницы на страницу (рис. 10.11). Когда просмотр отчета будет закончен, выберите **Предварительный просмотр → Закрыть →Закрыть окно предварительного просмотра** (Print Preview → Close Preview → Close Print Preview).

В режиме **Предварительный просмотр** лента разительно меняется. Исчезают вкладки, которые вы освоили и полюбили, программа Access заменяет их единственной вкладкой, названной **Предварительный просмотр** (это та же самая вкладка **Предварительный просмотр,** которую вы видели во время просмотра подготовленного к печати листа данных в *главе 3).* Для работы в окне предварительного просмотра можно использовать приемы, с которыми вы познакомились в *разд. "Предварительный просмотр страницы" главы 3,* одновременно просматривать несколько страниц (что позволяет увидеть места разрывов страниц) и изменять поля страницы и ориентацию листа бумаги.

Например, кнопки Книжная (Portrait) и **Альбомная** (Landscape) позволяют переходить от стандартной книжной ориентации (которая располагает короткую сторону листа бумаги сверху) к альбомной (которая поворачивает страницу, помещая длинную сторону листа бумаги сверху). Книжная ориентация вмещает больше строк, а альбомная - больше столбцов. Как правило, книжная ориентация лучше, при условии, что она может вместить все ваши столбцы. Если же это не так, можно попробовать применить альбомную ориентацию, шрифт меньшего размера, более узкие поля или лист бумаги большего размера.

Примечание

В отчетах в момент их создания всегда применяется бумага стандартного размера (обычно 8.5x11 дюймов или Letter). Но если вы изменяете размер, лист нового размера сохраняется вместе с отчетом. Это означает, что, когда вы откроете отчет в следующий раз, у него будет выбранный вами размер бумаги. Такое же правило действует и в отношении ориентации страницы.

У программы Access есть два дополнительных параметра, которые не включены в стандартный предварительный просмотр листа данных.

■ Используйте кнопку **Печатать только данные** (Print Data Only) для создания потоковой распечатки, которая пропускает такие подробности, как названия столбцов или заголовки отчетов. Этот вариант редко бывает полезен, поскольку полученную в результате распечатку трудно читать.

■ Используйте кнопку Столбцы (Columns) для размещения большего количества данных на странице. Этот вариант действует, если ваш отчет гораздо уже ширины страницы. Например, если он занимает меньше половины ширины страницы, вы можете удвоить объем данных на странице, применяя две колонки. Вам понадобится вдвое меньше страниц. В *разд. "Мастер создания наклеек" главы 11* вы увидите, как применяется многоколоночный отчет для размещения почтовых этикеток на странице.

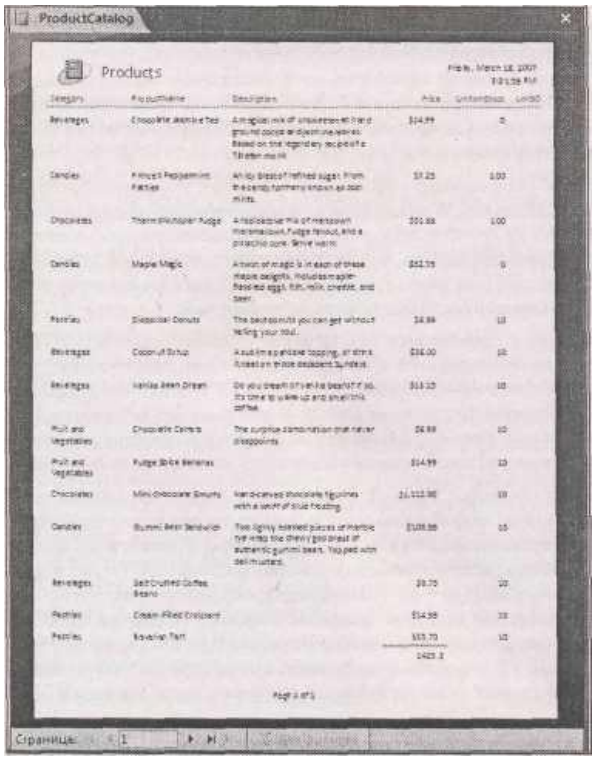


Рис. 10.11. Для отображения отчета крупным планом один раз щелкните кнопкой мыши. Щелкните мышью еще раз для того, чтобы вернуться назад к отображению всей страницы. Для перехода от одной страницы к другой можно использовать кнопки перехода между страницами в нижней части окна, а для более тонкого изменения масштаба - масштабный ползунок (не показан). Но самые полезные команды расположены на ленте, они позволяют настраивать параметры печати и экспортировать данные отчета в файл другого типа

Подсказка

Многие параметры макета страницы (такие как поля и ориентация страницы) можно изменить без перехода в режим **Предварительный просмотр.** Много аналогичных кнопок можно найти на вкладке ленты **Работа с макетами отчетов | Параметры страницы** (Report Layout Tools | Page Setup), которая появляется на экране, когда вы находитесь в **Режиме макета.**

10.2.2. *Экспорт отчета*

Вкладка **Предварительный просмотр -** немного странное творение, поскольку в нее включено несколько команд, не касающихся печати вашего отчета. Команды в группе **Предварительный просмотр → Данные** (Print Preview → Data) позволяют сделать моментальную копию данных текущего отчета и экспортировать их в файл другого типа, что позволит просмотреть отчет вне программы Access или работать с ним в другой программе. Это средство незаменимо, если вы хотите поделиться данными с другими людьми.

Несмотря на то, что Access поддерживает много разных форматов для экспорта отчетов, вам понадобится для отчета лишь несколько из них. (Другие гораздо полезнее при экспорте чистых данных из таблицы или запроса, как объясняется в *главе 19.)* К форматам, полезным для экспорта отчета, относятся следующие.

■ **Экспорт в файл RTF** (Word). Этот вариант преобразует ваш отчет в документ, который можно открыть в программе Microsoft Word. Однако применяемый программой Access формат неуклюж. В нем каждый столбец отделяется символами табуляции, а каждая строка - принудительным переводом строки, что затрудняет реорганизацию данных после переноса их в Word. Более удобный вариант экспорта должен был бы поместить данные отчета в таблицу Word, с которой было бы гораздо легче работать.

■ **Document HTML.** Этот вариант преобразует ваш отчет в форматированный HTML-документ, который годится для публикации в Web-пространстве или для чтения с вашего жесткого диска. Достоинство этого формата заключается в том, что для его просмотра вам нужен только Web-обозреватель (а у кого его нет?). Единственный недостаток формата - невозможность точной передачи форматирования, макета и разбиения на страницы вашего отчета, что станет помехой при желании распечатать экспортированный отчет.

■ **Средство просмотра снимков** (Snapshot Viewer). Этот вариант создает файл с расширением snp - копию экрана, которую любой пользователь может открыть для просмотра и печати полностью отформатированного отчета. Для того чтобы просмотреть снимок файла, вам понадобится свободно распространяемая программа Snapshot Viewer корпорации Microsoft. (Для загрузки ее из Интернета перейдите на Web-сайт **http://office.microsoft.com** и найдите поиском "Snapshot Viewer".) Несмотря на то, что программа Snapshot Viewer отлично работает, многие пользователи предпочитают применять более распространенный PDF-формат (следующий в этом перечне), обладающий теми же возможностями. (По правде сказать, Snapshot Viewer - часть наследия более ранних версий пакета Office.)

■ **PDF или XPS.** Этот вариант позволяет сохранить точное форматирование отчета (поэтому отчет можно напечатать), и пользователям, не имеющим программы Access (и, возможно, даже не имеющим ОС Windows), просмотреть ваш отчет. Единственный недостаток - эта возможность не включена в базовый пакет Access. Вам придется установить свободно распространяемый дополнительный модуль для ее использования (в следующем разделе вы узнаете, как это делается).

На профессиональном уровне.

Учитесь любить PDF-файлы

Возможно, вы слышали о PDF - популярном формате компании Adobe, предназначенном для коллективного использования отформатированных, готовых к выводу па печать документов.

Формат PDF применяется для распространения сопроводительных руководств к изделиям, буклетов и электронных документов всех сортов. В отличие от форматов документов, таких как XLSX, PDF-файлы разработаны для просмотра и печати, но не для редактирования.

Главное достоинство PDF-файлов состоит в том, что их можно просмотреть и напечатать на любом компьютере и под управлением операционной системы практически любого типа с помощью свободно распространяемой программы Adobe Reader. Вы можете загрузить эту программу со страницы **www.adobe.com/products/acrobat/readstep2.html,** но, возможно, это и не понадобится. На большинстве компьютеров Adobe Reader уже установлена, поскольку она приходит вместе с множеством других программ (обычно для того, чтобы вы могли просмотреть электронную документацию). Adobe Reader также широко применяется во Всемирной паутине.

PDF - не единственное детище в этом блоке. Корпорация Microsoft в новейшую операционную систему Windows Vista включила собственный формат для электронных документов, названный XPS (XML Paper Specification, бумажная спецификация на языке XML). По мере распространения формата XPS он, возможно, станет настоящим конкурентом PDF, но сейчас формат PDF несравнимо более популярен и шире распространен, поэтому именно он остается и центре внимания.

Неважно, какой формат вы используете, процесс экспорта по существу один и тот же.

1.Если вы не находитесь в режиме **Предварительный просмотр,** щелкните правой кнопкой мыши заголовок вкладки с отчетом и выберите **Предварительный просмотр.**

2.Щелкните мышью на ленте, в группе **Предварительный просмотр -> Данные** (Print Preview → Data) одну из кнопок в зависимости от выбранного вами формата для экспорта.

Например, выберите **Предварительный просмотр → Данные → Экспорт в файл RTF** (Print Preview → Data → Word) для копирования результирующего отчета в Word-совместимый

документ. Некоторые варианты хранятся в меню **Предварительный просмотр → Данные → Дополнительно** (Print Preview → Data → More), и вы не увидите вариант экспорта в PDF-формате, пока не установите дополнительный модуль для PDF (как описано в следующем разделе).

3. Выберите имя целевого файла (рис: 10.12).

Целевой файл - это место хранения экспортированных данных.

4. Если вы хотите открыть ваш экспортированный файл в соответствующей программе, установите флажок **Открыть целевой файл после завершения операции экспорта** (Open the destination file after the export operation is complete).

Предположим, что вы экспортируете в документ Word и установили этот флажок; программа Access экспортирует данные, запустит программу Word и загрузит в нее документ. Это хороший способ убедиться в том, что операция экспорта прошла успешно. Данный вариант сработает, только если на вашем компьютере установлена нужная программа.

5. Щелкните мышью кнопку ОК для выполнения экспорта.

Не обращайте внимания на два других флажка, окрашенных в серый цвет. Они действуют только в операциях экспорта, работающих с другими объектами БД.

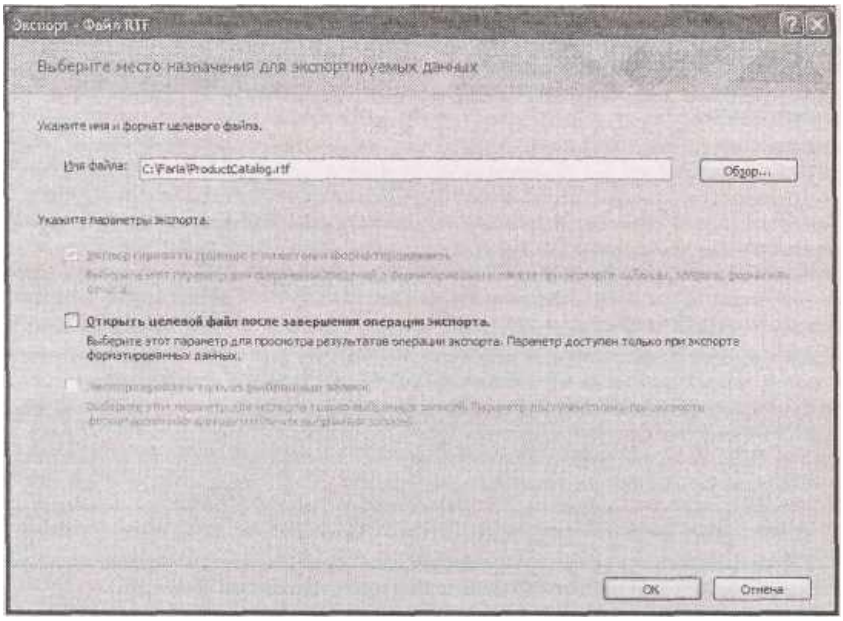


Рис. 10.12. Программа Access предполагает, что вы хотите использовать имя вашего отчета (например, ProductCatalog.rtf, если отчет **ProductCatalog** экспортируется в документ RTF-формата, который можно открыть в программе Word). Но вы можете заменить это имя любым другим

Примечание

Помните о том, что экспорт отчета подобен выводу отчета на печать. В вашем экспортируемом файле содержатся данные, имеющиеся в настоящий момент. Если неделей позже вы решите, что вам нужны более свежие данные, отчет придется экспортировать снова.

6. Выберите, хотите ли вы сохранить параметры экспорта.

Сохранив параметры экспорта, вы сможете быстро повторить эту операцию позже. Например, если вы экспортируете отчет в документ Word и сохраняете параметры экспорта, то сможете экспортировать данные отчета завтра, на следующей неделе или через год. Эта возможность описана в *главе 19,* которая рассматривает операции экспорта более подробно.

Подсказка

Для того чтобы экспортировать отчет, его необязательно открывать. Все нужные вам команды можно применять из области переходов. Просто щелкните правой кнопкой мыши название отчета и выберите команду **Экспорт** (Export) для вывода на экран меню со всеми вариантами операций экспорта, начиная PDF-файлами и заканчивая HTML-страницами. Вы также увидите несколько вариантов, не представленных на вкладке ленты **Экспорт,** включая варианты экспорта отчета в более старые уже почти забытые программы управления базами данных и электронные таблицы,

такие как dBase, Paradox и Lotus 1 -2-3.

10.2.3. *Получение дополнительного модуля "Save As PDF"*

Для экспорта отчета как PDF-файла вам понадобится дополнительный модуль "Save As PDF or XPS" (Сохранить как PDF или XPS). Для его получения перейдите на страницу www.microsoft.com/downloads и выполните поиск строки "PDF". Ссылки приведут вас на страницу, с которой можно загрузить дополнительный модуль и установить его парой щелчков кнопкой мыши.

После установки дополнительного модуля у ваших приложений пакета Office появится возможность экспорта их документов в PDF-формат. В отчете программы Access это таинство совершается при выборе **Предварительный просмотр** → Данные → **PDF** или **XPS** (Print Preview → Data → PDF or XPS), когда отчет отображен в режиме Предварительный просмотр. Или же можно щелкнуть правой кнопкой мыши отчет в области переходов и выбрать последовательность команд **Экспорт → PDF или XPS** (Export → PDF or XPS).

Когда отчет экспортируется в PDF-файл, вы получаете несколько дополнительных параметров настройки в диалоговом окне **Publish as PDF or XPS** (Публиковать как PDF или XPS). Можно экспортировать в PDF-файлы с разным разрешением и параметрами качества (которые в основном влияют на отчеты, содержащие изображения). Обычно применяются параметры, обеспечивающие высокое качество, если планируется печать PDF-файла, поскольку у принтеров более высокие разрешения, чем у компьютерных мониторов.

В диалоговом окне **Publish as PDF or XPS** помимо параметров качества есть элемент управления с вариантами **Optimize for** (Оптимизировать для). Если вы создаете копию в PDF-формате для того, чтобы другие пользователи могли просмотреть информацию в вашем отчете, выберите переключатель **Minimum size (publishing online)** (Минимальный размер (интерактивная публикация)) для сокращения размера файла. С другой стороны, если есть вероятность того, что пользователи, прочитав ваш отчет, захотят его напечатать, выберите переключатель **Standard (publishing online and printing)** (Стандартный (интерактивная публикация и вывод на принтер). Вы создадите PDF-файл несколько большего размера, который обеспечит распечатку более высокого качества.

Наконец, если вы хотите опубликовать только фрагмент вашего отчета в виде PDF-файла, щелкните мышью кнопку **Options...** (Параметры) для того, чтобы открыть диалоговое окно с несколькими дополнительными параметрами. Можно выбрать публикацию фиксированного числа страниц вместо полного отчета.

Подсказка

Получение дополнительного модуля "Save As PDF or XPS" сопряжено с некоторым неудобством, но он стоит затраченных усилий. В предыдущих версиях Access пользователи, желавшие создавать PDF-файлы, вынуждены были получать другой дополнительный модуль или покупать дорогую полную версию программы Adobe Acrobat. Первоначально средство "Save As PDF or XPS" предназначалось для включения в пакет Office (без необходимости в каких-либо дополнительных модулях), но антимонопольные отношения заставили сверхпредусмотрительную корпорацию Microsoft оставить его в стороне. Самое замечательное то, что этот модуль позволяет применять сохранение в PDF-файл в других приложениях Office, таких как Word, Excel и PowerPoint.

Часто задаваемый вопрос.

Разные способы экспорта данных

Что лучше экспортировать - результирующий отчет или все содержимое таблицы?

Существует несколько способов передачи данных за пределы программы Access. Вы можете взять данные непосредственно из таблицы или экспортировать результаты запроса или отчета. Какой же подход лучше?

Обычно самый легкий путь - получение данных из соответствующей таблицы (как описано в *главе 19).* Однако в некоторых случаях гораздо разумнее использовать отчет.

- Вы хотите использовать уникальное расположение столбцов, определенное в отчете. (Например, вы не хотите использовать всю таблицу **Products,** а отчет **ProductCatalog** выводит именно то, что нужно.)

• Вы хотите использовать параметры фильтрации, сортировки или группировки, которые уже применили в отчете. (Вы узнаете об этих возможностях в *главе 11.)*

• Вы хотите воспользоваться форматированием, которое применили в отчете. В зависимости от используемого варианта экспорта у вас может появиться возможность сохранить подробные настройки форматирования, такие как шрифты. Если вы экспортируете в PDF-файл, HTML-докумет или снимок отчета, все форматирование сохраняется. Если отчет экспортируется в приложение Office, например, Word или Excel, лишь некоторые характеристики форматирования сохраняются. Но если вы экспортируете таблицу или запрос, вы получаете только данные, и ваша задача - снова придать им привлекательный внешний вид.

В *главе 19* вы познакомитесь поближе с операцией экспорта таблиц и запросов.

10.3. *Форматирование отчета*

Вы уже научились создавать простые отчеты, отображающие всю нужную вам информацию в виде компактной таблицы. Единственная проблема этих отчетов - все они выглядят одинаково. Если вы работаете для транснациональной страховой компании в фирме, разделенной перегородками на рабочие места-кабинки, это унылое однообразие - возможно, хорошая вещь. Но более впечатлительным людям, быть может, хочется оживить свои отчеты, добавив в них границы, выразительные шрифты и яркие цветовые штрихи.

Самый быстрый способ форматирования - использование заранее подготовленных автоформатов (показанных на рис. 10.13) из коллекции, представленной на ленте **Работа с макетами отчетов | Формат → Автоформат → Автоформат** (Report Layout Tools | Formatting → AutoFormat → AutoFormat). Каждый вариант автоформата применяет комбинацию заданных шрифтов, цветов и параметров границы. Автоформат позволяет преобразовать представление отчета за один шаг, но не дает возможности скрупулезного управления, применения именно тех установок, которые вам нравятся.

Не все форматирование может сохраниться в HTML-документе. Об этом говорится ранее в описании форматов. - *Ред.*

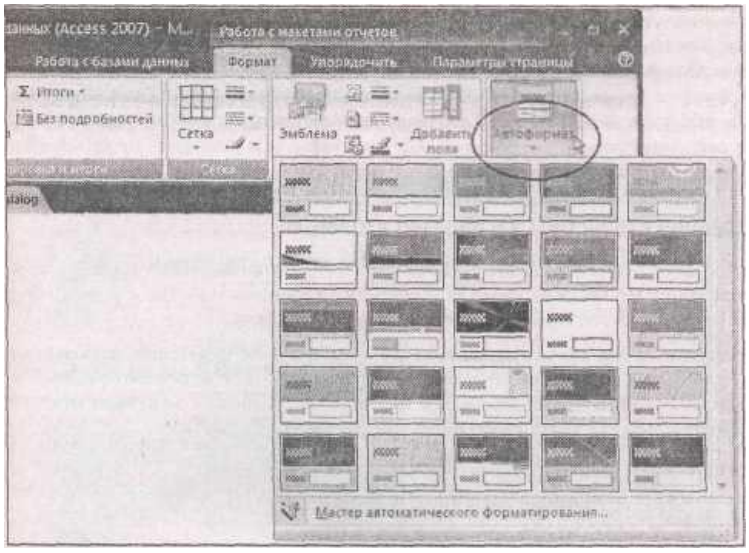


Рис. 10.13. Щелкните мышью направленную вниз стрелку (обведена) для того, чтобы увидеть все имеющиеся варианты автоформата. (Если у вас действительно большой монитор, варианты предварительного просмотра автоформатирования отображаются справа на ленте.) Каждая пиктограмма предварительного просмотра отображает цвета и фрагмент фона, применяемого в данном варианте форматирования, но для того чтобы увидеть, как оформление выглядит на самом деле, его нужно применить

Примечание

Напоминаю, для форматирования отчета нужно перейти в **Режим макета.** Если дважды щелкнуть название отчета в области переходов, он откроется в режиме **Представление отчета.** Щелкните правой кнопкой мыши заголовок вкладки и выберите команду **Режим макета** для

перехода в этот режим.

С помощью автоформата можно выполнить еще пару действий.

Для применения части параметров автоформатирования выберите на ленте **Работа с макетами отчетов | Формат → Автоформат → Автоформат → Мастер автоматического форматирования** (Report Layout Tools | Formatting → Auto Format → AutoFormat → AutoFormat Wizard). В диалоговом окне **Автоформат** (AutoFormat) выделите нужный вариант автоформата. Затем щелкните мышью кнопку **Параметры** (Options) для отображения трех флажков в нижней части диалогового окна: **шрифт** (Font), **цвет** (Color) и **границы** (Border). Сбросьте флажок того атрибута, который не хотите применять, и щелкните мышью кнопку **ОК**.

Для возврата к стандартному виду отчета без форматирования выберите **Работа с макетами отчетов | Формат → Автоформат → Автоформат → Мастер автоматического форматирования** для отображения диалогового окна **Автоформат.** Далее в списке вариантов автоформата выберите строку **Нет** (None) и щелкните мышью кнопку **ОК**.

■ Если вы использовали причудливый формат в вашем отчете и хотите сохранить его как пользовательский вариант автоформата, выберите на ленте **Работа с макетами отчетов | Формат → Автоформат → Автоформат → Мастер автоматического форматирования** для отображения диалогового окна **Автоформат.** Далее щелкните мышью кнопку **Настройка** и выберите переключатель **создание нового стиля па основе стиля объекта** (Create a new AutoFormat), задайте название автоформата и щелкните мышью кнопку ОК. Вы увидите созданный вариант в коллекции автоформатов.

10.3.1. *Форматирование столбцов и заголовков столбцов*

Автоформатирование - великая вещь, удобная для быстрого применения набора параметров форматирования. Но иногда хочется большего личного вмешательства и ручного форматирования некоторых частей отчета.

Для применения более узконаправленного форматирования необходим двухшаговый подход. Во-первых, выделите фрагмент отчета, который хотите отформатировать. Во-вторых, щелкните кнопкой мыши команду в группе ленты **Работа с макетами отчетов | Формат → Шрифт** (Report Layout Tools | Formatting → Font) (рис. 10.14).

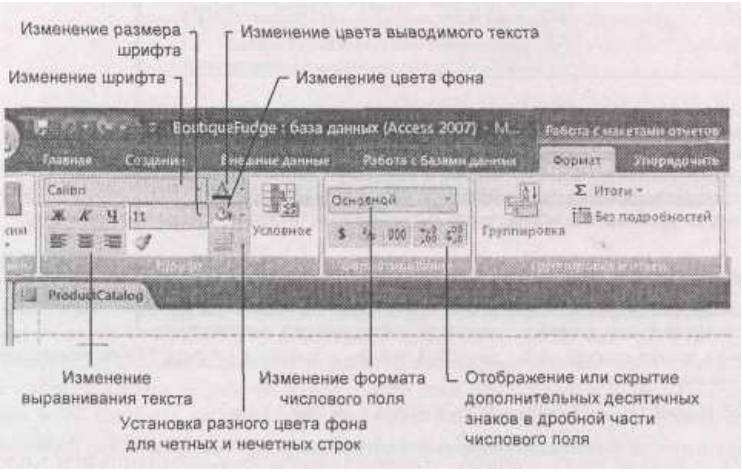


Рис. 10.14. Группа **Работа с макетами отчетов | Формат → Шрифт** включает основные средства форматирования

Группа **Работа с макетами отчетов | Формат → Шрифт** позволяет настроить все перечисленные далее параметры форматирования;

■ шрифт и его размер (шрифт Calibri размером 11 пунктов - стандарт, не утомляющий глаза при чтении);

■ выравнивание текста в строке (по левому краю, по центру, по правому краю);

■ цвет шрифта и цвет фона.

Несмотря на то, что можно форматировать разделы отчета с заголовком, датой или номером страницы, большую часть времени вы будете тратить на форматирование заголовков столбцов и

значений в столбцах. Для форматирования заголовка столбца щелкните его кнопкой мыши. Для форматирования значений в столбце щелкните кнопкой мыши любое из них. На рис. 10.15 приведен пример.

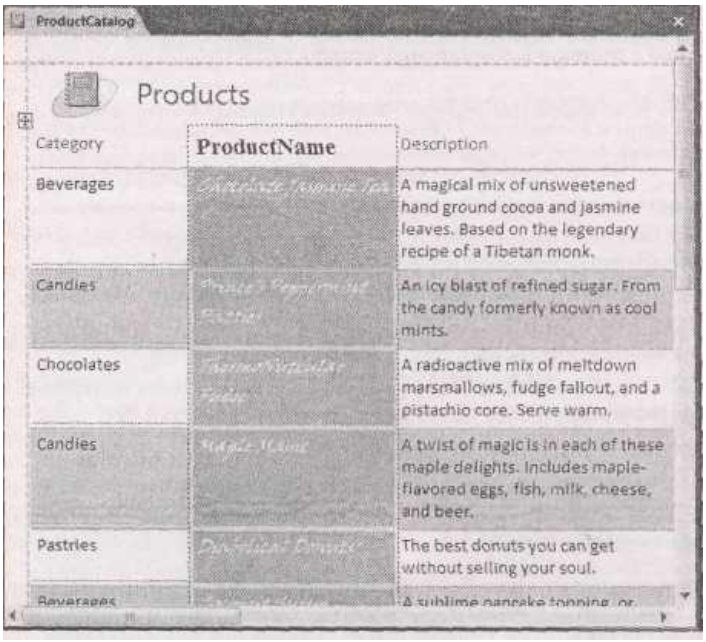


Рис. 10.15. Столбец **ProductName** выделен для задания особого форматирования. Несмотря на то, что выделенным кажется одно значение, программа Access применит изменения формата ко всему столбцу

Изменить форматирование отдельных значений в столбце нельзя. Это означает, что вы можете отформатировать столбец **ProductName** так, чтобы он выглядел иначе, чем столбец **Price,** но не сможете задать для значения Chocolate Jasmine Tea формат, отличный от формата значения Prince's Peppermint Patties. Это ограничение не лишено смысла - кроме всего прочего у вас могут быть тысячи записей, и отслеживание форматирования каждой из них было бы непосильной работой для программы Access.

Подсказка

Это ограничение можно обойти, если воспользоваться условным форматированием (см. разд. "Условное форматирование" далее в этой главе) для того, чтобы сообщить Access, когда следует добавить дополнительное форматирование, зависящее от значения в ячейке.

Малоизвестная или недооцененная возможность.

Формат по образцу.

Формат но образцу (Format Painter) - часто игнорируемый инструмент, копирующий форматирование из одного места отчета в другое. Суть заключается в том, что **Формат по образцу** копирует все параметры форматирования за один проход, включая цвет, шрифт и характеристики границы.

На самом деле **Формат по образцу** наиболее полезен в документоориентированных приложениях, таких как Word или Excel. Но в отчетах Access он тоже иногда удобен. Например, если вы только что закончили настройку форматирования одного столбца и хотите применить те же параметры в другом, **Формат по образцу** может быстро справиться с этой задачей.

Далее описаны действия, необходимые для применения **Формата по образцу.**

1. Щелкните кнопкой мыши область с параметрами форматирования, которые вы хотите скопировать.
2. Например, это может быть значение в отформатированном столбце.
3. Выберите на ленте **Работа с макетами отчетов | Формат → Шрифт → Формат по образцу** (Report Layout Tools j Formatting → Font → Format Painter).
4. Пиктограмма **Формата по образцу** выглядит как кисть для рисования.
5. Щелкните кнопкой мыши область, к которой вы хотите применить скопированные

параметры форматирования.

Например, это может быть другой столбец. Вот и готово: Access копирует ваш формат из первого столбца во второй.

10.3.1.1. *Форматирование числовых полей*

Вы можете использовать группу ленты **Работа с макетами отчетов | Формат → Форматирование** (Report Layout Tools | Formatting → Formating) для настройки вывода числовых полей (например, поля **Price** в отчете **ProductCatalog).** В группе есть раскрывающийся список, позволяющий выбрать разные варианты форматирования чисел.

■ **Основной** (General Number) выводит базовое обычное число. Программа Access оставляет в значении столько десятичных знаков, сколько требуется.

■ **Денежный** (Currency) гарантирует два десятичных знака в дробной части и символ валюты в соответствии с настройками вашего компьютера (в зависимости от географического местоположения). Большие числа получают запятые в качестве разделителя тысяч для отделения цифр, как в числе $1,111.99[2].

■ **Евро** (Euro) аналогичен формату **Денежный** за исключением отображения символа евро в качестве знака валюты.

■ **Фиксированный** (Fixed) выводит числа с одним и тем же количеством знаков в дробной части. (Первоначально это количество равно двум, но вы можете использовать для его изменения кнопки **Увеличить разрядность** (Increase Decimals) и **Уменьшить разрядность** (Decrease Decimals), показанные на рис. 10.16.) У больших чисел нет запятых-разделителей разрядов.

[2] Разделитель между целой и дробной частью, как и разделитель для тысяч, зависит от настроек в Панели управления. - *Ред.*

■ **С разделителями разрядов** (Standard) аналогичен **Фиксированному** за исключением запятых, разделителей для тысяч (как в 1,111.99).

■ **Процентный** (Percent) считает любое число частью целого, представленной в процентах, где 1.0 равно 100 процентам. Если у вас есть число 48, Access изменит его на 4800.00%. (Изменить число знаков в дробной части можно с помощью кнопок **Увеличить разрядность** и **Уменьшить разрядность.)**

■ **Экспоненциальный** (Scientific) отображает каждое число с помощью экспоненциального представления, поэтому 48 превращается в 4.80E+01 (причудливый способ сказать о том, что 4.8, умноженное на 101, дает значение, которое хранится в поле). Экспоненциальное представление применяется для отображения чисел из сильно отличающихся числовых диапазонов с почти одинаковым количеством цифр. Изменить число знаков в дробной части можно с помощью кнопок **Увеличить разрядность** и **Уменьшить разрядность.** Изменить количество знаков справа от десятичной точки можно, щелкнув мышью кнопки **Увеличить разрядность** (Increase Decimals) и **Уменьшить разрядность** (Decrease Decimals) в группе ленты **Работа с макетами отчетов | Формат → Форматирование.**

10.3.1.2. *Форматирование чередующихся строк*

Существует простой, но эффектный прием форматирования: добавление затененного фона в каждой второй строке. Форматирование чередующихся строк добавляет немного изысканности самому простому отчету, но оно служит и практическим целям. В отчетах с большим количеством данных затененные полосы помогают читателям различать смежные строки и переводить взгляд вдоль строки от столбца к столбцу.

Для применения форматирования чередующихся строк необходимо щелкнуть кнопкой мыши сразу слева от любой строки. В этой точке выделяется вся строка целиком и становится активной кнопка **Работа с макетами отчетов | Формат → Шрифт → Изменить цвет заливки/фона** (Report Layout Tools | Formatting → Font → Alternate Fill). (Эта кнопка выглядит как сетка. Она расположена сразу под кнопкой **Цвет заливки/фона** (Fill).) Щелкните ее, а затем выберите цвет.

Если щелкнуть кнопкой мыши одно из значений в строке, кнопка **Изменить цвет заливки/фона** не будет активна, и вы не сможете изменить цвет заполнения чередующихся строк.

10.3.1.3.*Линии сетки*

Когда создается новый отчет, данные помещаются в невидимую таблицу. У этой таблицы нет никакой сетки, поэтому распечатки выглядят приглаженными и легкими. Но если вы - скрытый любитель сетки, вас обрадует возможность добавить границы в таблицу отчета. Вам решать, вставлять ли их везде и тщательно распределить все данные по отдельным ячейкам или применять их разумно, выделяя только важные столбцы.

Подсказка

Сетка полезна в плотно заполненных отчетах, данные в которых в противном случае напоминают мешанину. Гуру программы Access знают, что меньше значит больше, и применение только нескольких линий сетки обычно лучше, чем добавление их между каждыми столбцом и строкой.

Применить линии сетки можно двумя способами. Простейший и самый распространенный вариант - сетка для всей таблицы. Для этого щелкните кнопкой мыши в любом месте таблицы с данными отчета и выберите один из вариантов сетки из списка на ленте **Работа с макетами отчетов | Формат → Сетка → Сетка** (Report Layout Tools | Formatting → Gridlines → Gridlines) (рис. 10.16). Далее используйте другие кнопки в группе **Работа с макетами отчетов | Формат → Сетка** для изменения толщины, цвета и стиля (штриховая, точечная, сплошная и т. д.) линий сетки. В *примечании "Практические занятия для опытных пользователей. Разные линии сетки "далее в этом разделе* объясняется второй способ применения сетки.

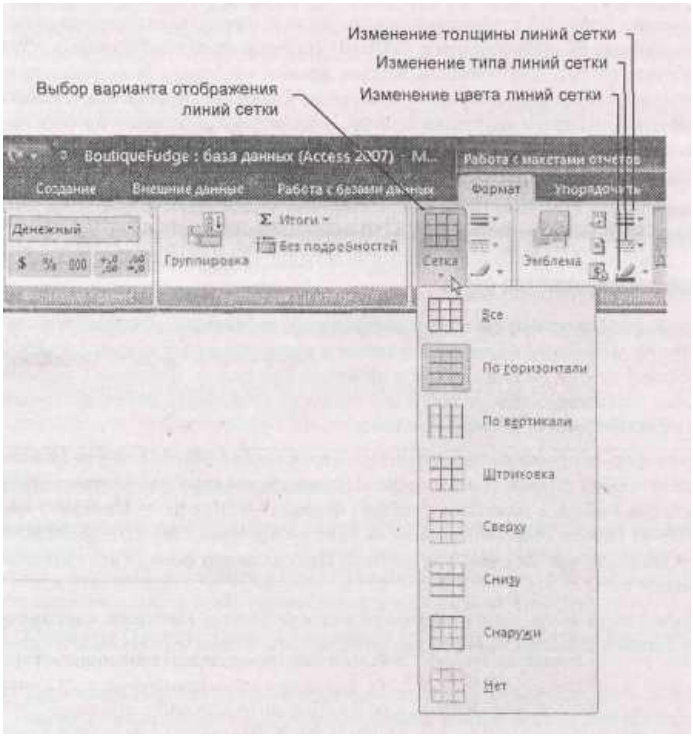


Рис. 10.16. Команды в группе **Сетка** на ленте позволяют использовать линии сетки самых распространенных образцов: только между столбцами, только между строками, вдоль внешнего края данных и т. д. Можно также выбрать стиль линии (сплошная, точечная, штриховая и т. д.), толщину и цвет

Примечание

У сетки есть одна особенность. К заголовкам столбцов можно применять линии сетки, отличающиеся от сетки остальной таблицы. Для задания линий сетки в шапке таблицы (секции заголовков столбцов) просто щелкните кнопкой мыши любой заголовок столбца и затем выберите на ленте нужный вариант линий сетки.

Практические занятия для опытных пользователей.
Разные линии сетки

Единственное ограничение способа применения сетки с помощью ленты - в вашем распоряжении один тип, цвет и одна толщина линии. Например, если вы добавляете штриховую

синюю границу для левой стороны столбца **ProductName,** вам придется использовать ту же синюю штриховую линию и для правой стороны. Но это ограничение можно обойти с помощью **Окна свойств.**

Вот как работает этот метод.

1. Щелкните правой кнопкой мыши один из столбцов и затем выберите команду **Свойства** (Properties).

Б левой части окна программы появится **Окно свойств** (Property Sheet). В нем показан длинный список вариантов, которые можно настраивать. Многие из них не слишком полезны и предназначены для использования в формах (которым посвящена *часть IV)* или коде на VBA *(см. часть V).*

2. В раскрывающемся списке в верхней части **Окна свойств** выберите столбец, с которым хотите работать.

Например, если выбран столбец **ProductName,** вы сможете настроить вид левой и правой линий сетки и линии между строками.

3. Выберите вкладку **Макет** (Format).

Перейдите в конец списка, к набору параметров, управляющих характеристиками левой, правой, верхней и нижней линий сетки.

4. Если необходимо, измените их.

Например, можно использовать параметры **Ширина линий сетки слева** (Gridline Width Left) и **Стиль линий сетки слева** (Gridline Style Left) для изменения толщины и стиля линии сетки слева от столбца. Вы увидите, что есть только один параметр, задающий цвет, - **Цвет линий сетки** (Gridline Color) - поскольку линии сетки со всех сторон столбца должны быть одинакового цвета.

10.3.1.4. *Границы*

Наряду с линиями сетки можно использовать аналогичный набор параметров линий границы. Разница между сеткой и границей заключается в том, что сетка применяется к таблице данных отчета, а границы могут быть связаны с любым компонентом вашего отчета.

В группе ленты **Работа с макетами отчетов | Формат → Элементы управления** (Report Layout Tools | Formatting → Controls) можно найти три кнопки (для выбора толщины, цвета и стиля границы). Границы нет смысла применять к значениям столбца, поскольку в результате вы получите рамку вокруг каждого значения. Гораздо полезнее устанавливать границы вокруг других элементов отчета, например его заголовка.

10.3.2. *Условное форматирование*

Средний отчет содержит много информации. Какая-то ее часть более важна, чем остальные сведения. Например, у вас могут быть веские причины для выделения товаров, отсутствующих па складе, заказов стоимостью более 100 долларов, кукол-болванчиков, купленных в прошлом году, или свадебных гостей, не приславших подарки. С помощью условного форматирования можно акцентировать внимание на этих порциях данных, задав для них другое форматирование.

Фундаментальная идея условного форматирования заключается в том, что вы определяете условие, которое, принимая значение *Истина,* сообщает программе Access о необходимости применить дополнительное форматирование к значению в столбце.

Для использования условного форматирования выполните следующие действия.

1. Выберите значение в столбце, к которому хотите применить условное форматирование, Например, если хотите выделить товары стоимостью более 100 долларов, щелкните кнопкой мыши одно из значений в столбце **Price.** Неважно, какое значение вы выберите - правило условного форматирования применяется ко всем значениям в столбце.

2. Выберите **Работа с макетами отчетов | Формат → Шрифт → Условное** (Report Layout Tools | Formatting → Font → Conditional).

Если кнопка **Условное** не активна, возможно, вы не выбрали подходящую порцию данных в отчете. Например, если выбрать заголовок столбца вместо значения в столбце, условное форматирование применить невозможно.

После щелчка мышью кнопки **Условное** на экране появится диалоговое окно **Условное форматирование** (Conditional Formatting) с одним условием. Это окно можно использовать для

задания от одного до трех условий в одном столбце, но чаще всего достаточно одного.

3. С помощью списка и текстовых полей задайте условие, которое должна проверить программа Access.

Первое поле с раскрывающимся списком позволяет выбрать, хотите ли вы проверять реальные данные, хранящиеся в поле **(Значение поля),** или вычисление, использующее эти данные **(Выражение).** Простейший и чаще всего применяемый вариант - условие, базирующееся на значении поля.

С помощью второго поля с раскрывающимся списком выберите тип сравнения, которое вы хотите выполнить. Например, можно проверить, равно ли значение в ячейке заданному числу, больше или меньше заданного или лежит в некотором диапазоне значений. Наконец, введите данные, которые программа Access должна использовать для сравнения. Например, если вы выбрали сравнение **меньше чем,** введите значение, с которым Access будет сравнивать. Если выбрана проверка принадлежности диапазону значений, введите два значения, задающие этот диапазон.

На рис. 10.17 показано заполненное окно **Условное форматирование,** проверяющее дорогостоящие товары в поле **Price.**

Примечание

Все сравнения "между" - охватывающие. Например, если задано условие "между 1 и 10", оно истинно для чисел 1, 10 и любых промежуточных значений. С другой стороны, если программа Access столкнется со значением 0.99, она не будет применять к нему условное форматирование.

4. Щелкните мышью кнопки **Формат** (Format) для задания параметров форматирования, которые программа должна применить, если условие истинно.

Рис. 10.17. В данном примере цена товара будет выводиться жирными красными символами желтом фоне, если она больше 50 долларов

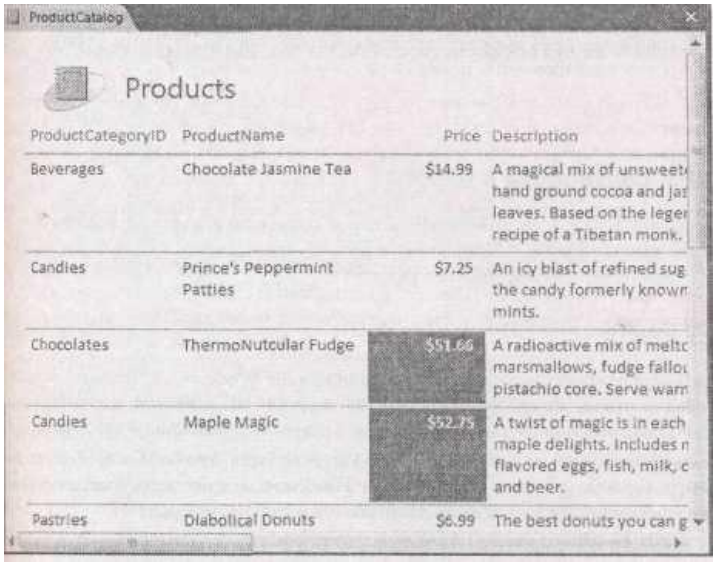


Рис. 10.18. Данный отчет не даст пропустить дорогостоящие товары

Этот вариант форматирования может изменить цвет шрифта или фона, применить жирное курсивное или подчеркнутое начертание шрифта. Но в условном форматировании нельзя изменить гарнитуру шрифта или его размер.

Под строкой с вашим условием вы увидите область предварительного просмотра для выбранных вами параметров форматирования.

Если хотите добавить второе и третье условия, щелкните мышью кнопку Добавить » (Add ») и вернитесь к пункту 3.

Например, можно задать шрифт синего цвета для пометки цен выше 50 долларов и шрифт красного цвета для выделения цен, превышающих 100 долларов. Если у вас несколько перекрывающих друг друга условий, программа Access применит только первое соответствующее.

Подсказка

Во избежание недоразумений тщательно структурируйте ваши условия, избегая их взаимного перекрытия. Например, используйте одно условие для отбора значений между 100 и 499, а другое - для значений не меньших 500.

Для удаления условий можно щелкнуть мышью кнопку **Удалить** (Delete). Access выведет на экран диалоговое окно с просьбой указать удаляемые условия. Вы можете установить флажок рядом с соответствующим условием и щелкнуть мышью кнопку **ОК.**

6. Щелкните мышью кнопку **ОК.**

После нажатия кнопки **ОК** программа Access вычисляет условие для каждого значения в столбце и устанавливает нужное форматирование. На рис. 10.18 показан окончательный вид отчета.

10.4. *Фильтрация и сортировка в отчете*

В отчетах предлагаются те же средства фильтрации и сортировки, которые вы научились применять к листу данных в *главе 3.* Кроме того, у вас есть возможность группировки и подсчета итогов, которые будут рассматриваться в *главе 11.*

10.4.1. *Фильтрация в отчете*

В отчете ProductCatalog представлены все записи из таблицы Products. Однако очень часто отчеты нуждаются в отборе важного подмножества данных. Например, у вас может появиться желание проанализировать объемы продаж товаров определенной категории или заказов, сделанных клиентами из конкретного города. Из отчета ProductCatalog было бы логично исключить товары, снятые с производства. Помимо всего прочего, компании Boutique Fudge не за чем рекламировать товары, которые она больше не продает.

Вы можете сократить результаты, включенные в отчет, двумя способами. Один вариант вы уже изучили: создание запроса, извлекающего нужные вам результаты, и последующее использование запроса для формирования отчета. Этот способ хорош, если у вас уже есть запрос, удовлетворяющий требованиям, или вы планируете использовать это подмножество данных для нескольких целей (отчеты, редактирование, другие запросы и т. д.).

Другой вариант - применить фильтрацию к данным отчета. Преимущество этого метода заключается в том, что вы можете изменять условия фильтрации быстро и многократно, Если планируется использование одного и того же отчета для печати нескольких выборок данных, это самый лучший подход. Например, можно отобрать товары одной категории, напечатать их, а затем изменить условия фильтрации для выбора товаров другой категории, которые тоже можно вывести на печать.

Фильтрация отчета выполняется так же, как и фильтрация листа данных. У вас есть два возможных варианта.

- Если вы хотите быстро сформировать условие отбора на основе имеющегося значения, щелкните правой кнопкой мыши это значение, как показано на рис. 10.19. Например, в поле **CategoryName** можно щелкнуть правой кнопкой мыши значение Beverages (напитки). Всплывающее меню содержит несколько вариантов условий отбора, основанных на текущем значении. В зависимости от выбранного варианта можно включить в отчет записи из категории Beverages, записи из других категорий, записи из категорий, в название которых входит слово Beverages (например, Alcoholic Beverages (алкогольные напитки)) и т. д.

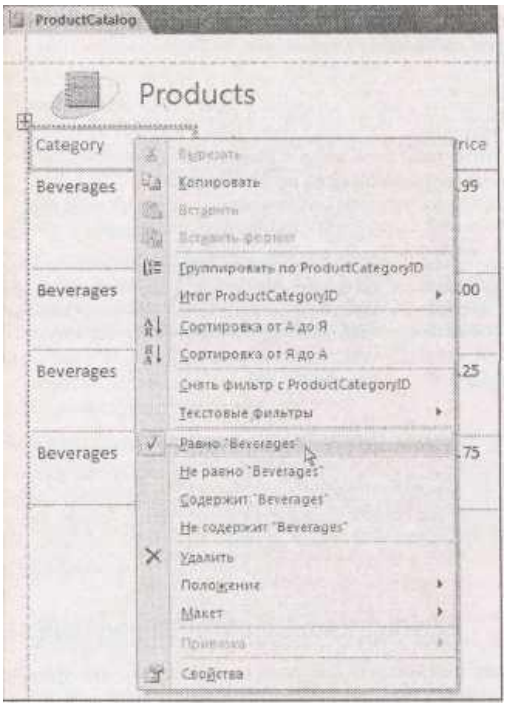


Рис. 10.19. Варианты быстрой фильтрации, которые вы видите, зависят от типа данных. В этом примере условия отбора позволяют задать разные фильтры, базирующиеся на слове "Beverage"

■ Если вы хотите создать выражение для большей гибкости условия, щелкните правой кнопкой мыши любое значение в столбце и найдите подменю фильтрации. Точное название меню зависит от типа данных. Например, если щелкнуть правой кнопкой мыши поле CategoryName, появится подменю **Текстовые фильтры.** Если щелкнуть правой кнопкой мыши поле **Price,** вы увидите подменю **Числовые фильтры.** Эти подменю включают набор вариантов фильтрации, позволяющих задать конкретные выражения. Все тонкие детали и вспомогательную информацию о создании разнообразных выражений фильтров *см. в разд. "Фильтрация" главы 3.*

Фильтры могут применяться в нескольких столбцах одновременно. Для удаления фильтра щелкните правой кнопкой мыши столбец и выберите команду **Снять фильтр с...** (Clear filter from...).

10.4.2. *Сортировка данных в отчете*

Первоначально в отчете существует тот же порядок записей, что и в источнике данных. Если отчет строится на базе запроса, порядок определяется порядком сортировки, использованным в запросе. Если отчет создается на базе таблицы, у записей нет определенного порядка, как правило, они выводятся в порядке их добавления в таблицу.

В любом случае сортировку можно применить непосредственно в отчете, во многом так же, как и на листе данных. Просто щелкните правой кнопкой мыши заголовок соответствующего столбца и найдите вариант сортировки. Варианты сортировки зависят от типа данных - например, вы можете упорядочить текстовые поля в алфавитном порядке, даты - в хронологическом, а числовые поля - по возрастанию или по убыванию.

Примечание

Одновременно можно сортировать по одному полю. Если вы хотите применить более сложную сортировку, использующую несколько столбцов (например, сортировку, разделяющую товары по категориям, отсортированным в алфавитном порядке, а затем упорядочивающую заказы в каждой категории в зависимости от цены товара), для такого отчета придется создать запрос.

11. Глава 11. Проектирование сложных отчетов

В предыдущей главе вы научились создавать простые отчеты - хорошо отформатированные распечатки, компонующие данные в одну таблицу. Простые отчеты - прекрасная вещь для создания твердой копии, более отшлифованной, чем простая распечатка листа данных. Как уже известно из предыдущей главы, простые отчеты предоставляют возможность детального форматирования, необходимого для выделения важных столбцов и значений, кроме того, в простых отчетах изящно обрабатываются большие текстовые поля без потери пространства на листе и без обрезания данных.

Простые отчеты - великолепное средство программы Access, но они статичны, строги и просты. Главное их ограничение - структура отчета. Независимо от того, как вы форматируете или упорядочиваете данные в простом отчете, программа Access всегда представляет их в виде таблицы. В реальной жизни может возникнуть необходимость представления отпечатанных данных в иной форме. Возможно, вам понадобится преобразовать данные в счета клиентов, списки посещаемости класса или почтовые этикетки. Все эти отчеты выполняют одну и ту же задачу - извлекают данные из таблицы и затем организуют их на печатной странице, но ни для одного из них не подойдет стандартный отчет с его простой табличной структурой.

В этой главе вы увидите, как создавать разнообразные, более специализированные отчеты, использующие идеи, изложенные в предыдущей главе, и дополненные несколькими новыми приемами. В процессе изучения вы обратитесь к режиму **Конструктора,** узнаете, как добавлять изображения и контуры, и научитесь устанавливать разрывы страниц в длинных распечатках. Вы также узнаете, как использовать группировку для анализа данных и вычисления промежуточных итогов.

11.1. *Улучшение отчетов в Конструкторе*

Конструктор - это режим, который освобождает ваши отчеты. Как вы узнали в предыдущей главе, **Конструктор** обеспечивает иное представление отчета. В отличие от **Режима макета** в **Конструкторе** вы не увидите никаких данных отчета. Вместо этого перед вами предстанет проект вашего отчета, сообщающий программе Access о том, как формировать отчет. С помощью этого представления можно делать то, что практически невозможно в любом другом режиме.

Рассмотрим простой отчет с перечнем товаров, созданный в предыдущей главе. Переключившись в режим **Конструктора,** можно увидеть, что делает отчет действующим (рис. 11.1). Для перехода в **Конструктор** щелкните правой кнопкой мыши заголовок вкладки и выберите режим **Конструктор.**

Рис. 11.1. Окно **Конструктора** разделено на пять разделов. Каждый из них сообщает программе Access о том, как конструировать фрагмент отчета. Раздел **Область данных** - самая важная часть. Когда запускается выполнение отчета (с помощью печати или отображения в **Режиме макета),** Access повторяет раздел **Область данных** для каждой строки. Программа заполняет поля в разделе **Область данных** значениями из соответствующей записи

Примечание

Странная сетка из линий и точек, которую вы видите в **Конструкторе,** предназначена для того, чтобы помочь вам выровнять разные части отчета. Как вы увидите, пользователи часто применяют **Конструктор** для размещения данных в точно заданной позиции, поскольку автоматического выравнивания разных фрагментов отчета не существует.

11.1.1. *Разделы в режиме конструктора*

Секрет освоения режима **Конструктор -** понимание назначения пяти его разделов. Несмотря на то, что некоторые из них можно оставить пустыми, в каждый отчет они включаются практически в неизменном порядке.

■ **Заголовок отчета** (Report Header)**.** Этот раздел выводится один раз в начале отчета, на первой странице. Именно в него вставляются заголовки, эмблемы и ваши имя и фамилия.

■ **Верхний колонтитул** (Page Header). Этот раздел появляется сразу под **Заголовком отчета** на первой странице и в верхней части каждой последующей страницы. Это место для вставки номеров страниц, а в простых табличных отчетах, таких как каталог товаров, он также применяется для размещения заголовков столбцов.

■ **Область данных** (Detail). Этот раздел отображается сразу после **Верхнего колонтитула** и является сердцем всех отчетов. Хитрость заключается в том, что **Область данных** выводится один раз для каждой записи вашего отчета. В простом табличном отчете этот раздел представляет единственную строку.

■ **Нижний колонтитул** (Page Footer). Этот раздел отображается в нижней части каждой страницы. Если вы не пользуетесь **Верхним колонтитулом** для вывода номеров страниц, этот раздел предоставляет вам еще одну возможность.

■Примечание отчета (Report Footer). Этот раздел выводится один раз в конце любого отчета. Его можно использовать для печати сводных данных, знаков авторского права, даты вывода на печать и других сведений.

Содержимое описанных разделов выглядит совершенно иначе чем то, что вы видели в других режимах, поскольку не отображает реальные данные. Вместо данных разделы содержат заполнители, в которые программа Access может каждый раз вставлять необходимую информацию, когда формируется отчет. Если вы создаете отчет о товарах, Access извлекает мнения из полей **ProductCategoryID, ProductName, Price и Description** и затем перемещает их в соответствующие поля.

Понадобится некоторое время для того, чтобы вы освоили манипулирование данными в режиме **Конструктор.** Прежде всего, нужно научиться изменять размер каждого раздела. Эта возможность полезна, т. к. в разных отчетах для каждой области отводятся различные по объему пространства. На рис. 11.2 показано, как изменять размер разделов.

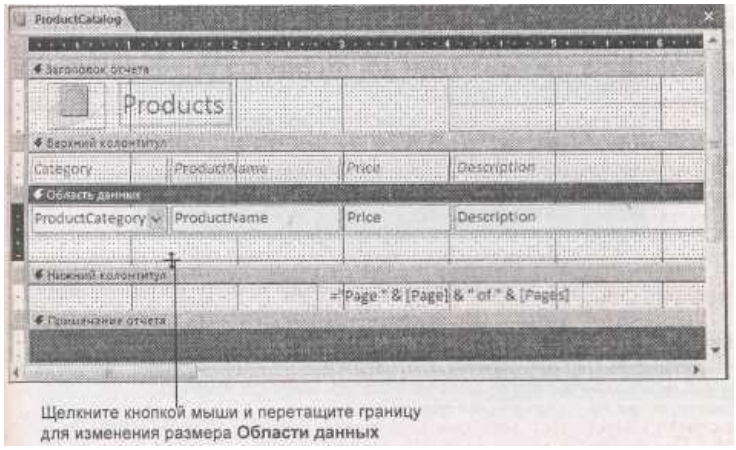


Рис. 11.2. Для изменения размера раздела переместите указатель мыши к границе, расположенной под разделом, размер которого нужно изменить. Перетащите с помощью мыши границу вниз (для увеличения его размера) или вверх (для сокращения отведенного пространства). Дальше ваша задача - разместить содержимое внутри раздела для того, чтобы заполнить доступное пространство. В данном примере раздел **Область данных**

расширяется

Примечание

Если вы не хотите использовать раздел, его размер можно свести практически к нулю. Посмотрите на раздел **Примечание отчета** на рис. 11.1. Он присутствует, но не будет виден, поскольку в этот отчет не включены примечания. Помимо этого можно скрыть разделы колонтитулов.

Просто щелкните правой кнопкой мыши отчет и выберите команду **Колонтитулы страницы** для того, чтобы удалить разделы колонтитулов в отчете. (Щелкните кнопкой мыши эти команды еще раз для того, чтобы снова отобразить эти разделы.)

11.1.2. *Об элементах управления*

Конструктор предлагает иное представление вашего отчета. Программа Access отображает все содержимое отчета с помощью элементов управления: графических объектов, содержащих текст, изображения и форматирование. Каждый элемент управления - это отдельный объект. Вы можете изменить его внешний вид или перетащить его в другое место (иногда может понадобиться сначала вытащить его из табличного макета).

На рис. 11.3 показаны элементы управления в отчете, представляющем каталог товаров. На рисунке видны не все элементы, поскольку не осе столбцы попали в поле зрения.

Рис. 11.3. В отчете о товарах есть элементы управления для отображения заголовка, номеров страниц, заголовков столбцов и данных. У каждого элемента управления есть тонкая граница черного цвета, которая помогает выделить элемент и изменить его размер в **Конструкторе.** На распечатке эти границы не видны

Примечание

Небольшая странность на рис. 11.3 может удивить вас. На рисунке есть несколько полей ввода, которые выглядят, скорее, как подписи. Подобное отображение объясняется содержимым этих элементов управления. Как вы узнаете чуть позже в этой главе, в отчетах элемент управления **Подпись** применяется для неизменного текста, а элемент управления **Поле -** для переменного содержимого (основанного на текущей дате, текущей записи, текущей странице и т. д.).

Для того чтобы чувствовать себя свободнее в Конструкторе, поэкспериментируйте с простым отчетом. (Если у вас под рукой нет отчета, можно загрузить из Интернета БД Boutique Fudge с примерами к данной главе; см. Web-страницу "Missing CD" на сайте **www.missingmanuals.com.)** Далее перечислены некоторые задания, которые полезно проделать самостоятельно.

■Перетащите с помощью мыши с одного места на другое элементы управления в разделах **Заголовок отчета** и **Нижний колонтитул.**

■Измените размер элемента управления в разделе **Заголовок отчета** или **Нижний колонтитул,** перетащив с помощью мыши черную границу, окружающую элемент.

■Измените порядок вывода столбцов, переместив с помощью мыши их заголовки в разделе

Верхний колонтитул или поля в разделе **Область данных.** Когда перемещается столбец, программа Access автоматически реорганизует остальные столбцы. Программа действует так же, как при изменении порядка отображения столбцов в Режиме макета.

■Выделите элемент управления и измените его форматирование с помощью группы ленты **Инструменты конструктора отчетов | Конструктор -* Шрифт** (Report Design Tools | Design → Font). Этот метод можно применять как к элементам в разделах верхнего и нижнего колонтитулов, так и к заголовкам столбцов и отдельным полям.

Когда вы завершите изменения, щелкните правой кнопкой мыши заголовок вкладки и выберите другой режим (например, **Представление отчета, Режим макета** или **Предварительный просмотр)** для того, чтобы увидеть, как будет выглядеть напечатанный отчет. Когда вы закроете отчет, программа Access предложит сохранить выполненные вами изменения.

Вы должны уметь делать в **Конструкторе** все, что вы делали для настройки отчета в Режиме **макета.** Конечно, удобнее применять **Режим макета** для выполнения большинства этих задач. Но как вы увидите в следующем разделе, **Конструктор** предоставляет больше возможностей для отказа от типичного табличного отчета и организации ваших данных любым понравившимся вам способом.

11.1.3. *Удаление полей из макета*

В простом отчете программа Access группирует все поля в нечто, именуемое *макетом.* На самом деле макет - это контейнер, который позволяет легко работать с группами полей. Эта новинка появилась в Access 2007; она предоставляет несомненные удобства, с которыми вы познакомились в *главе 10:*

■когда вы перемещаете заголовок столбца, данные этого столбца перемещаются вместе с ним и наоборот;

■когда вы переносите столбец в новое место, программа Access соответствующим образом реорганизует другие столбцы;

■когда вы расширяете столбец, Access отодвигает все следующие за ним стоящие на пути столбцы. Аналогично, когда вы сжимаете столбец, следующие столбцы сдвигаются для заполнения освободившегося пространства.

Без макета вы не смогли бы перемещать столбцы так быстро. Каждый раз, когда нужно было внести изменение в один столбец, пришлось бы кропотливо переносить все остальные столбцы. В отчете с несколькими десятками полей этот процесс доставил бы множество хлопот.

Несмотря на то, что макеты - это маленькие чудесные удобства, они становятся смирительной рубашкой, если вы хотите организовать данные иначе. Допустим, вы хотите взять отчет с каталогом товаров и сделать его похожим не на инвентарную опись, а на розничное издание, показанное на рис. 11.4. С помощью макета это сделать не удастся, поскольку поля всегда заключены в жесткую табличную структуру. Такого результата можно добиться, только если вытащить поля из табличного макета и затем скомпоновать их вручную.

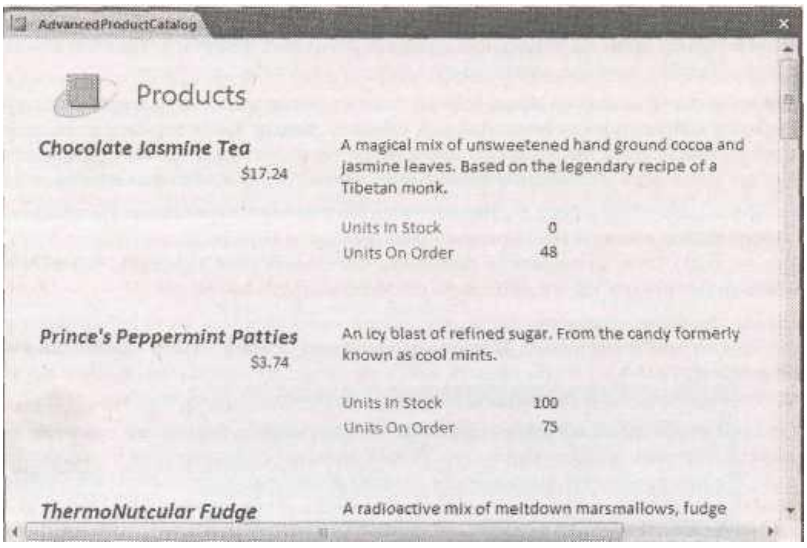


Рис. 11. 4. В этой версии отчета, содержащего каталог товаров, не используется табличный макет - все элементы управления "плавающие". Программа Access по-прежнему

создает отчет так же, как и в случае применения табличного макета - она выводит раздел **Область данных** для каждой строки. Единственное отличие- способ компоновки данных в разделе **Область данных**

Примечание

Не путайте макеты и **Режим макета.** Макет - это контейнер, компонующий набор элементов управления. **Режим макета -** это способ отображения вашего отчета и изменения различных его параметров. Вы можете применять **Режим макета,** даже если не используете макетные контейнеры (layout containers).

Далее приведены действия, необходимые для преобразования основанного на табличном макете каталога товаров в свободную от табличной структуры версию, показанную на рис. 11.4.

1. Перейдите в **Режим макета** (щелкните правой кнопкой мыши заголовок вкладки и выберите команду **Режим макета).**

Вы можете удалить поля в **Конструкторе,** но результаты будут не столь удачны. Когда поле удаляется в **Режиме макета,** программа Access автоматически убирает его в сторону и отводит ему немного собственного пространства. Когда же поле удаляется в **Конструкторе,** программа Access оставляет его на прежнем месте. Поскольку за полем все еще стоит табличный макет, в результате вы получаете груду элементов управления, лежащих друг на друге, что очень затрудняет их компоновку.

2. Найдите поле, которое хотите удалить из макета. Щелкните правой кнопкой мыши заголовок столбца и выберите **Макет → Удалить** (Layout → Remove).

Повторите этот пункт для удаления всех полей, которые хотите реорганизовать. Решите, хотите ли вы поместить одни поля в макет, а другие извлечь из макета, или же удалить все поля из табличного макета (как показано на рис. 11.4).

Для одновременного удаления нескольких полей держите нажатой клавишу <Shift>, когда щелкаете столбцы кнопкой мыши, и далее выберите **Макет → Удалить** (Layout → Remove). (Или для выбора всех столбцов щелкните кнопкой мыши крошечную пиктограмму из перекрещенных стрелок, появляющуюся в правом верхнем углу отчета.)

3. Теперь перед вами груда полей в вашем отчете. Щелкните правой кнопкой мыши заголовок вкладки и перейдите в **Конструктор.**

Можно размещать поля и в **Режиме макета,** но большинство пользователей находят это сложным, поскольку в **Режиме макета** одновременно отображается несколько записей. Возможно, их будет легче разместить в **Конструкторе,** применяя шаблоны разделов.

4. Увеличьте раздел **Область данных,** перетащив с помощью мыши его нижнюю границу (как показано на рис. 11.2).

В простом отчете вам требуется ровно одна строка для размещения записи. Но когда выполняется пользовательская компоновка, почти всегда требуется больше пространства.

5. Перетащите мышью элемент **Поле** для каждого поля записи в нужное место **Области данных** и затем задайте ему соответствующий размер.

Возможно, потребуется некоторая перекомпоновка, прежде чем вы разместите все поля в нужных местах. Поскольку данные теперь не в табличном макете, программа Access не убирает их автоматически с дороги. Вам придется все располагать вручную и следить за тем, чтобы поля не перекрывались.

6. В разделе **Верхний колонтитул** выберите заголовок столбца для одного из полей раздела **Область данных.** Либо нажмите клавишу <Delete> для его удаления, либо перетащите его с помощью мыши в **Область данных.**

Нет смысла помещать заголовок столбца в верхнюю часть страницы, если поле больше не является частью таблицы. Если данные говорят сами за себя, заголовок им и не требуется. Тем не менее вы можете перетащить с помощью мыши заголовок столбца в **Область данных** и поместить его рядом с соответствующими данными так, чтобы он служил заголовком для них. Отчет на рис. 11.4 содержит заголовки для полей **UnitsInStock** (единиц в запасе) и **UnitsOnOrder** (единиц в заказах).

7. Если вы еще не сделали этого, выделите все поля и задайте для них нужное форматирование.

В Конструкторе можно форматировать данные во многом так же, как и **Режиме макета.**

Просто выделите поле, а затем используйте группу ленты **Инструменты конструктора отчетов | Конструктор → Шрифт.** Держите нажатой клавишу <Shift>, если хотите выделить (а потом и отформатировать) несколько элементов управления одновременно.

Когда закончите, перейдите в **Режим макета** или в **Представление отчета** для просмотра внесенных изменений. На рис. 11.5 показана окончательная компоновка переделанного отчета с каталогом товаров, приведенного на рис. 11.4.

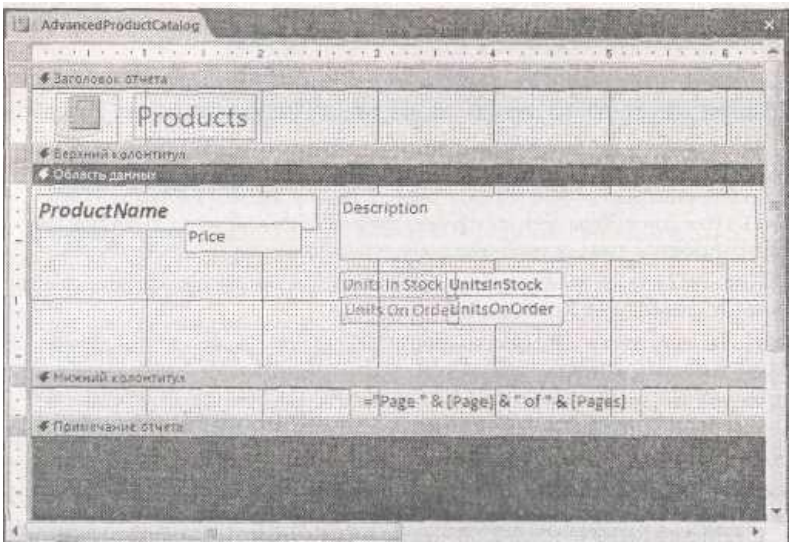


Рис. 11.5. После удаления полей из табличного макета их можно размещать как угодно - даже перекрывать один элемент управления другим для более сжатого вывода

11.1.4. *Добавление дополнительных элементов управления*

В предыдущем примере вы использовали полученные знания для освобождения полей вашего отчета. Но **Поля** со значениями полей таблиц - не единственный тип элементов управления, которые можно использовать. Отчеты Access также поддерживают **Подписи, Рисунки, Кнопки** и другие графические элементы, способные оживить самый унылый отчет. Вы можете вставлять дополнительные элементы управления по следующим причинам:

- для включения дополнительной текстовой информации, такой как подзаголовки, отказ от гарантийных обязательств, пояснительные заметки, название компании и т. д.;
- для вывода разделительных линий между областями в разделе **Область данных;**
- для выделения важной информации с помощью дополнительных рамок;
- для вывода эмблемы в верхнем или нижнем колонтитулах (автоматически отображаемые эмблемы, как вы узнали из *разд. "Создание пустого отчета" главы 10,* выводятся в разделе **Заголовок отчета).**

Добавить дополнительные элементы управления в отчет очень легко. Просто найдите нужную кнопку на ленте. Когда ваш отчет отображается в режиме **Конструктор,** комплексное обслуживание можно найти в группе ленты **Инструменты конструктора отчетов | Конструктор → Элементы управления** (Report Design Tools | Design → Controls), показанной на рис. 11.6.

Рис. 11.6. Применяя кнопки в левой секции группы **Элементы управления,** можно вставить несколько похожих компонентов, таких как заголовок отчета и номера страниц. Когда вы щелкаете

кнопкой мыши соответствующую кнопку, программа Access автоматически вставляет подходящий элемент в тот раздел, которому он принадлежит. Кнопки в средней части группы более специализированы. Для вставки одного из элементов нужно щелкнуть мышью кнопку и затем нарисовать мышью элемент управления в нужном месте области отчета. После вставки элементов управления кнопки в правой секции помогут нарисовать границы вокруг элементов

Некоторые из этих элементов управления - например, **Поля, Флажки** и другие элементы для редактирования - предназначены для применения в формах и нечасто используются в отчетах. Другие, например, кнопки **и** гиперссылки в сочетании с поддержкой макрокода (**как** вы увидите в *главе 15)* способны запускать полезные действия. Но сейчас вам понадобится только несколько элементов управления.

■ Элемент управления **Подпись** хранит маленькие или большие фрагменты неменяющегося текста. Например, все заголовки столбцов с именами полей - **Подписи.**

■ Элемент управления **Поле** содержит динамические выражения - иначе говоря, текст, который может меняться.

■ Элемент управления **Рисунок** содержит изображение.

■ Элемент управления **Линия** позволяет рисовать вертикальные, горизонтальные и наклонные линии. Он удобен для графического отделения содержимого и замены границ, не позволяющих добиться желаемого эффекта.

■ Элемент управления **Прямоугольник** позволяет рисовать отформатированные прямоугольники вокруг других элементов управления для выделения содержащейся в них информации.

■ Элемент управления **Вставить или удалить разрыв страницы** дает возможность разбить раздел **Область данных** на отдельные страницы - точно в том месте, которое вы укажете. Это очень полезно, если в **Области данных** хранится большой объем информации или если печатаются формы, которые должны выводиться на отдельных страницах (например, счета для разных клиентов).

Примечание

Несмотря на то, что в программе Access есть элементы управления для изображений, прямоугольников и линий, в ней отсутствует средство ClipArt, которое можно найти в других приложениях пакета Office. Поэтому не ищите изящных контуров и стилизованных текстов - их нет.

После выбора нужного элемента управления его можно вставить в отчет, как показано на рис. 11.7.

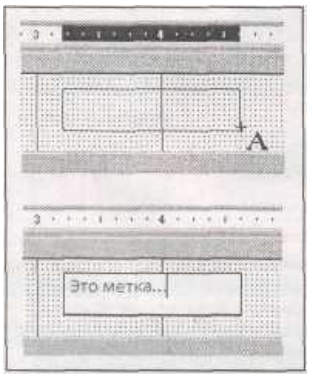


Рис. 11.7. *Вверху:* для добавления элемента управления в отчет щелкните его кнопкой мыши на панели инструментов. Затем перемещайте мышь с нажатой левой кнопкой в рабочей области отчета, пока прямоугольник не займет область, которую вы хотите ему отвести. *Внизу:* когда кнопка мыши будет отжата, в прямоугольнике отобразится элемент управления. Конечно, можно перемещать элемент управления и изменять его размер для получения желаемого эффекта. Если вставляется **Подпись** (как показано на рисунке), необходимо ввести в нее нужный текст

Если вы вставляете **Подпись,** нужно добавить текст, содержащийся в ней. После того как **Подпись** включена в отчет, курсор выводится внутри нее, и вы можете вводить текст. Если позже текст придется редактировать, щелкните один раз кнопкой мыши для выделения **Подписи** и затем подождите, пока указатель мыши не превратится в текстовый курсор (известный

профессионалам как I-образный курсор). Затем щелкните внутри Подписи кнопкой мыши еще раз и начинайте редактировать ее текст.

Примечание

Когда вы добавляете новую **Подпись,** программа Access может вывести предупреждающую пиктограмму с восклицательным знаком. Если провести мышью по этой пиктограмме, можно увидеть предупреждение, сообщающее о том, что ваша **Подпись** не связана ни с каким элементом управления (например, **Полем,** содержащим значение). Не беспокойтесь, если вы просто вставляете основной заголовок или неменяющийся текстовый фрагмент, эта ситуация - как раз то, что вам нужно.

Если вставляется **Прямоугольник,** возможно, вам захочется задать цвет контурной линии (с помощью группы **Инструменты конструктора отчетов | Конструктор → Шрифт**

(Report Design Tools | Design → Font)). Если поместить два элемента управления на одно и то же место, программа Access кладет последний вставляемый элемент управления поверх того элемента, который был добавлен первым. Для перемещения элемента управления вниз выделите его и выберите **Инструменты конструктора отчетов | Упорядочить → Положение → На задний план** (Report Design Toots | Arrange → Position → Send to Back).

11.1.5. *Создание отчета без помощи мастера*

К настоящему моменту вы уже попробовали изменить простой отчет с помощью Конструктора. Но если вы не хотите применять табличный макет, легче начать с режима Конструктор и создать свой отчет в нем. Когда отчет формируется в **Конструкторе,** программа Access не включает автоматически поля в табличный макет, как в **Режиме макета.**

Для создания отчета в **Конструкторе** нужно просто создать новый пустой отчет и затем добавить в него все нужные элементы управления в соответствующие разделы. Далее приведены действия, необходимые для создания отчета.

1. Выберите **Создание → Отчеты → Конструктор отчетов** (Create → Reports → Report Design).

Этот шаг создает новый пустой отчет и открывает его в Конструкторе.

2. Выберите **Инструменты конструктора отчетов | Конструктор → Сервис → Добавить поля** (Report Design Tools | Design → Tools → Add Existing Fields).

В правой части окна программы появляется панель **Список полей** (Field List) с перечнем таблиц и содержащихся в них полей.

3. Перетащите с помощью мыши поля, которые вы хотите отобразить в своем отчете, в раздел **Область данных.**

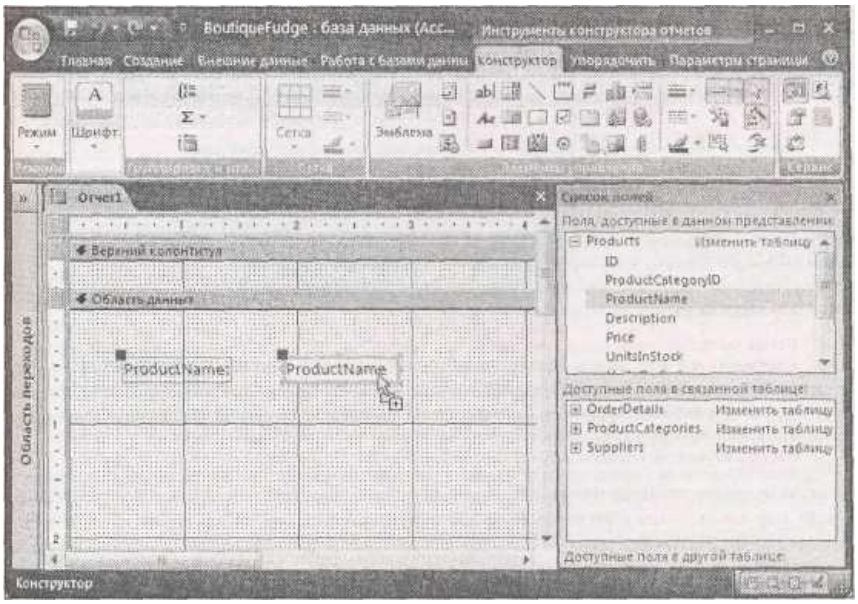


Рис. 11.8. В новый отчет добавлено поле ProductName. При вставке нужных полей придется потратить время на размещение их в рабочей области отчета, прежде чем отчет будет выглядеть как

следует

При каждом добавлении поля в отчет программа Access вставляет два элемента управления: **Подпись,** отображающую имя поля, и **Поле,** в которое выводятся данные поля (рис. 11.8).

Примечание

Программе Access для вывода значений поля нужен элемент управления **Поле,** а не **Подпись,** поскольку элемент управления **Подпись** хранит текстовую константу, неизменный текстовый фрагмент. Только **Поле** может получать реальные значения из поля таблицы или запроса.

Переместите поле в нужное место и затем подберите для него подходящий размер. Изменение размера сначала кажется непростым делом, поскольку вы работаете с двумя связанными элементами управления. Если вы перетащите любой из них, другой последует за ним. На рис. 11.9 показано, как передвигать только подпись к полю или только значение поля.

Если вам не нужна **Подпись,** просто выделите ее и нажмите клавишу <Delete>.

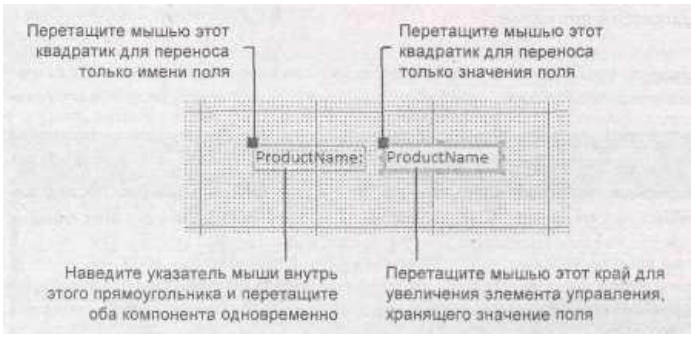


Рис. 11.9. В зависимости от того, где щелкнуть кнопкой мыши, вы сможете перемещать одновременно и подпись поля, и его значение или передвигать только один из этих компонентов. Можно изменить размер каждого из них, переместив с нажатой кнопкой мыши левый или правый край

5. С помощью группы ленты **Инструменты конструктора отчетов | Конструктор → Элементы управления** (Report Design Tools | Design → Tools → Controls) добавьте дополнительное содержимое (например, заголовок, номера страниц и различные текстовые фрагменты и изображения).

Группа **Элементы управления** позволяет вставлять большой набор различных элементов управления.

6. Отформатируйте ваши элементы управления, как вам хочется.

В группе ленты **Инструменты конструктора отчетов | Конструктор → Шрифт** есть команды, необходимые для изменения гарнитуры шрифта, его размера и цветов, в то время как в группе **Инструменты конструктора отчетов | Конструктор → Элементы управления** есть команды, нужные для добавления рамки вокруг элемента управления.

Значения полей и их имена можно форматировать отдельно - только убедитесь в том, что вы выбрали нужный компонент, прежде чем щелкать кнопкой мыши команду форматирования.

Подсказка

Для выделения определенных значений можно использовать условное форматирование точно так же, как вы делали в **Режиме макета** *(см. разд. "Условное форматирование" главы 10).*

7. Если хотите использовать **Заголовок отчета** или **Примечание отчета,** щелкните правой кнопкой мыши свободное место в рабочей области отчета и выберите Заголовок **отчета/Примечание отчета.**

Этот шаг выводит на экран разделы **Заголовок отчета** и **Примечание отчета.** Затем можно добавить элементы управления в них. Если вы решите, что они вам больше не нужны, повторите этот шаг еще раз, чтобы скрыть **Заголовок отчета** и **Примечание отчета.** Вы также можете

скрыть разделы верхнего и нижнего колонтитула, щелкнув правой кнопкой мыши рабочую область отчета и затем выбрав **Колонтитулы страницы.**

8. Измените размеры разделов отчета, чтобы устранить лишнее свободное пространство в нижней части.

Обычно требуется сжатие **Области данных,** поскольку вначале она отображается слишком большой. Если этого не сделать, то между записями вашего отчета будут значительные пустоты.

9. Сохраните отчет.

Сохранить отчет можно в любой момент, перейдя в меню **Office** и выбрав последовательность команд **Файл → Сохранить** (File → Save) или же закрыв ваш отчет, в этот момент программа Access предложит вам сохранить его.

11.2. *Мастер создания отчетов*

Создание отчета в **Конструкторе -** занятие на любителя. Добавление и компоновка необходимых элементов управления требует времени. Корпорация Microsoft решила добавить укороченный способ для быстрого создания отчетов разных типов. Этот сокращенный вариант и есть Мастер отчетов.

Примечание

С помощью Мастера отчетов легче создать отчет, не использующий табличный макет, при наличии заранее заданных параметров, предназначенных для организации элементов управления. Если вы хотите создать отчет на основе табличного макета (как вы делали это в *главе 10),* не следует применять Мастер отчетов - нужный отчет можно создать за один шаг.

Мастер отчетов задает несколько основных вопросов и затем формирует соответствующий отчет. Позже отчет можно дорабатывать в **Конструкторе** сколько душе угодно. Далее описаны действия, необходимые для применения мастера.

1. Выберите на ленте **Создание → Отчеты → Мастер отчетов** (Create → Reports → Report Wizard).

На экране появится первое окно мастера.

2. Из раскрывающегося списка выберите таблицу, которую хотите использовать.

В списке **Доступные поля** (Available Fields) отображаются все поля вашей таблицы.

Примечание

Возможно, вы узнали это окно, поскольку оно такое же, как окно, применяемое для создания запроса с помощью мастера создания запросов (см. рис. 6.11).

3. Добавьте поля, которые хотите включить в отчет, как показано на рис. 11.10. По завершении нажмите кнопку **Далее.**

Можно выбирать поля из нескольких таблиц при условии, что эти таблицы связаны.

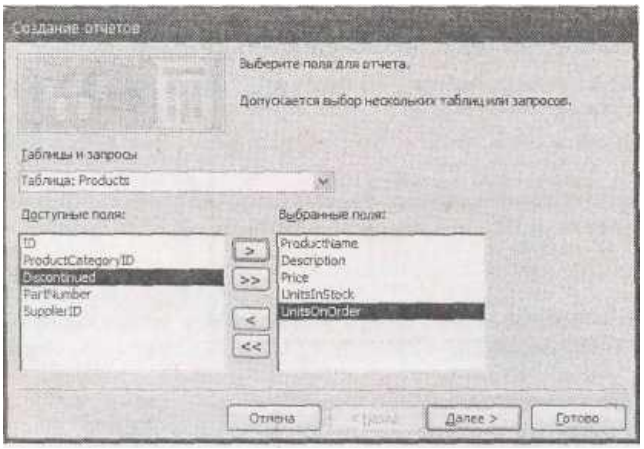


Рис. 11.10- Для добавления поля выделите его и щелкните мышью кнопку > для переноса поля из списка **Доступные поля** в список **Выбранные поля.** Щелкните мышью кнопку » для переноса всех

полей одним щелчком

4. На следующем этапе можно при необходимости задать группировку. Сейчас щелкните мышью кнопку **Далее** для создания отчета без группировки.

Вы узнаете, как применять группировку в отчетах, в *разд. "Группировка "далее в этой главе.*

5. Выберите поле (или поля), предназначенное для сортировки результатов, представленных в отчете, и щелкните мышью кнопку **Далее.**

Для сортировки данных отчета можно применять до четырех полей, но обычно для размещения данных в нужном порядке достаточно и одного.

6. Выберите вариант макета для вашего отчета (рис. 11.11).

Можно выбрать один из следующих вариантов макета.

- Макет **в столбец** (Columnar) помещает все поля в отдельные строки, следующие одна за другой. Название слегка вводит в заблуждение - по сути, в отчете формируются два столбца. Первый столбец содержит имя поля, а второй - данные поля.
- Макет табличный (Tabular) применяет невидимые макетные таблицы, которые вы изучали в *главе 10.* Программа Access преобразует каждое поле в отдельный столбец.
- Макет **выровненный** (Justified) размещает информацию на минимальном пространстве. В одну строку может быть включено несколько полей. Название "выровненный" означает заполнение данными всей ширины страницы без пропусков. Там, где закапчивается одно поле, начинается другое.

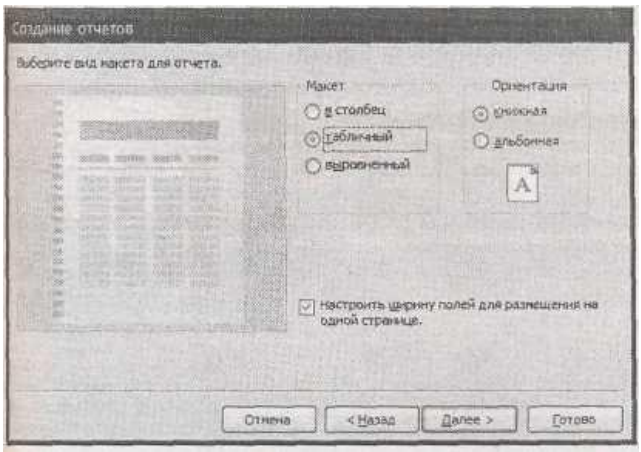


Рис. 11.11. Вариант макета сообщает программе Access, как следует скомпоновать поля в **Области данных**

7. Если вы хотите повернуть страницу на 90°, выберите альбомную ориентацию и нажмите кнопку Далее.

Ориентация **альбомная** (Landscape) позволяет разместить широкие таблицы или большие объемы информации, но содержит меньше строк на странице.

8. Выберите один из стандартных стилей и щелкните мышью кнопку Далее.

Стили определяют форматирование, применяемое программой Access в вашем отчете. К сожалению, вы не сможете увидеть окончательный результат, пока не попробуете каждый вариант стиля.

9. Введите название вашего отчета.

Когда Мастер отчетов завершит работу, он тут же сохранит ваш отчет.

10. Выберите вариант **Просмотреть отчет** (Preview the report), если хотите увидеть полученный результат в режиме предварительного просмотра, или вариант **Изменить макет отчета** (Modify the report's design), если хотите изменить отчет в Конструкторе. Затем щелкните мышью кнопку **Готово.**

Программа Access сохранит ваш отчет и затем откроет его в режиме **Предварительный просмотр** или **Конструктор** в зависимости от сделанного вами выбора.

Как видите, Мастер отчетов не слишком гибок. Он поддерживает лишь несколько типов макетов и не позволяет управлять компоновкой разных полей. Но он может стать хорошей отправной точкой (даже если это не так, он поможет исследовать макет отчета в **Конструкторе).**

11.3. *Мастер создания наклеек*

Если у вас есть таблица, содержащая адреса (домашние адреса клиентов, деловые контакты или подозрительные сайты НЛО), программа Access может предложить другой мастер. Мастер создания наклеек извлекает адресную информацию из любой таблицы, заданной вами, и использует ее для вывода на печать удобных почтовых наклеек.

Для выполнения этой работы нужно только купить несколько листов бумаги для наклеек в магазине вашего любимого офисного поставщика. Бумага для наклеек бывает разной - на одних листах информация размещается очень компактно, поэтому можно напечатать сразу десятки обратных адресов, на других применяются более крупные наклейки для размещения почтовых адресов на письмах или пакетах. Но независимо от выбранного типа бумаги у нее есть стандартный код Avery, и вы сможете создать отчет, включающий адрес в нужное место. Все что нужно сделать - это напечатать наклейку, снять ее с листа и приклеить.

Подсказка

Если у вас есть БД, хранящая информацию о людях, можно подумать о применении отчетов программы Access для создания бланков писем и других документов. Привлечение Access не всегда может облегчить задачу. Лучше воспользоваться настоящими текстовыми процессорами, такими как Word. Программа Word обладает средством слияния почтовой информации, способным извлечь данные из БД Access и затем использовать их для генерации любого нужного вам документа.

Для создания серии наклеек выполните следующие действия.

1. В области переходов выделите таблицу с адресами.

Это не обязательно должны быть адреса. Если вы хотите напечатать именные карточки сотрудников или товарные этикетки или у вас возникло непреодолимое желание поместить таинственные объекты вокруг дома, для них тоже можно использовать наклейки.

Примечание

Если нужно создать наклейки с информацией из нескольких таблиц, необходимо создать запрос с операцией объединения (join query) *(см. разд. "Запросы и связанные таблицы" главы 6),* а затем выделить его, прежде чем запускать мастер создания наклеек.

2. Выберите на ленте **Создание → Отчеты → Наклейки** (Create → Reports → Labels).
3. Запустится мастер создания наклеек. На первом этапе вам придется указать тип бумага для наклеек, которую вы используете (рис. 11.12).
4. Если у вас рулонная бумага для наклеек (а не отдельные листы), выберите тип наклейки **рулонные** (вместо **на листах).**

Если у вас принтер, относящийся к доисторическим временам компьютерного печатания, вы вряд ли сможете воспользоваться этим вариантом.

Найдите в списке наклейку, у которой тот же код товара, что и у нашей бумаги для наклеек. Еще раз проверьте размеры, предлагаемые программой Access. Обычно код товара - это код Avery, применяемый большинством пользователей. (Вы без труда найдете код Avery на лицевой стороне упаковки бумаги для наклеек.) Но если для вашей бумаги используется другая система обозначения, выберите из списка **Фильтр по изготовителю** компанию - производителя этой бумаги.

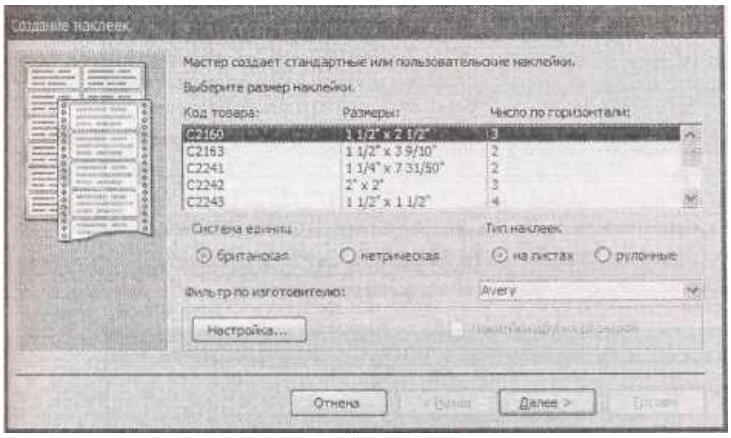


Рис. 11.12. В данном примере используются наклейки стандартного размера C2160, которые размещаются в трех столбцах на странице

Примечание

Если вы создаете необычные нестандартные наклейки собственного изготовления, щелкните мышью кнопку **Настройка** (Customize) для вывода диалогового окна **Размеры наклеек** (New Label Size) и затем щелкните мышью **Создать** (New) для отображения диалогового окна **Создание наклейки** (New Label). После этого вы сможете задать точные размеры для каждой части вашей наклейки.

5. Щелкните мышью кнопку **Далее.**

На следующем этапе мастер предлагает выбрать форматирование для текста на вашей наклейке (рис. 11.13).

6. Выберите нужный шрифт, размер и цвет шрифта и щелкните мышью кнопку **Далее.** Конечно, эти параметры можно изменить в **Конструкторе** и позже, но лучше задать их правильно с самого начала. Обычно рекомендуется сохранять размер, предлагаемый программой Access. - он рассчитан на вывод па наклейке от четырех до шести полных строк текста (в зависимости от типа наклейки).

На следующем шаге вы сможете выбрать поля, которые Access поместит на вашу наклейку.

7. Для вставки первой строки в вашу таблицу найдите нужные поля в списке **Доступные поля** и дважды щелкните их кнопкой мыши.

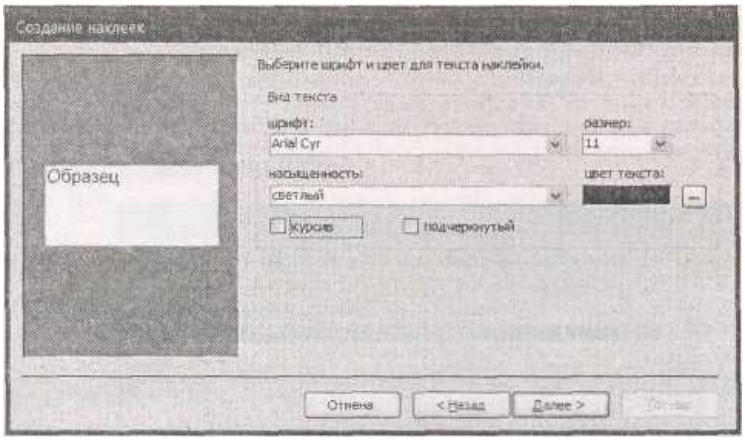


Рис. 11.13. Программа Access отображает окно с образцом, содержащим текст, чтобы вы случайно не сделали наклейку, не вмещающую заданный текст

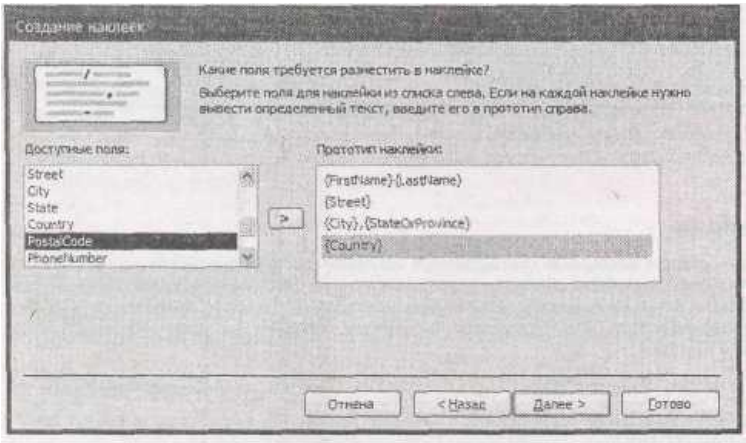


Рис. 11.14. Для вставки пробелов в данные на наклейке щелкните кнопкой мыши в области **Прототип наклейки** между двумя полями, которые вы хотите разделить. Затем нажмите клавишу <Пробел>. Можно также вставить любой текст в нужное место (например, слово "Кому: " или запятую)

Добавляйте поля в том порядке, в каком они должны выводиться на наклейке **(FirstName, LastName, Street, City** и т. д.). При выборе поля программа Access вставляет специальный заполнитель в область **Прототип наклейки.** Она вставляет код { FirstName}, чтобы показать, куда будет помещать значение из поля **FirstName.**

Вы отвечаете за вставку пробелов между этими полями. (Как правило, возникает желание вставить пробелы и запятые.) На рис. 11.14 показано, как это делать.

8. В области **Прототип наклейки** щелкните кнопкой мыши во второй строке. Теперь повторите пункт 7 для вставки полей в эту строку.

Повторяйте это действие до тех пор, пока не вставите все нужные ноля, каждое в соответствующую строку.

9. При необходимости выберите поле для сортировки наклеек и щелкните мышью кнопку Далее.

Вам может быть важен или не важен порядок сортировки. (Он может помочь установить соответствие между наклейками и письмами, если письма аналогично отсортированы. Если же вы готовите массовую рассылку, одинаковую для всех, порядок сортировки не имеет значения.)

Часто пользователи применяют не сортировку, а фильтрацию *(см. разд. "Построение условий отбора "главы 6)* для получения только некоторых наклеек (например, для клиентов, живущих в определенном городе).

Если применяется сортировка, программа Access упорядочивает наклейки на странице слева направо, а затем сверху вниз.

10. Введите имя отчета.

Мастер создания наклеек сохраняет ваш отчет сразу после завершения работы.

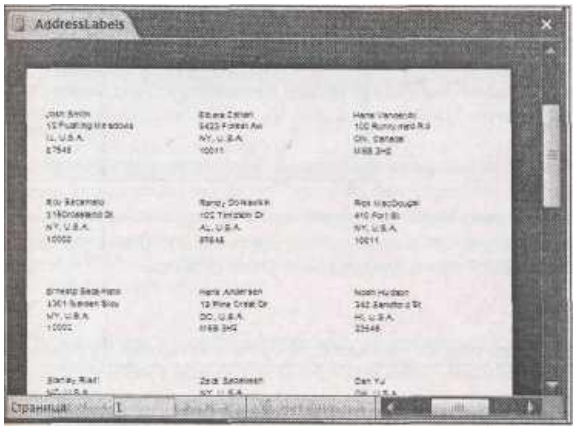


Рис. 11.15. Окончательный вид отчета с наклейками

11. Выберите **Просмотреть наклейки в том виде, как они будут напечатаны** (See the labels as they will look printed), если хотите увидеть окончательный вид отчета в режиме

Предварительный просмотр, или **Изменить макет наклеек** (Modify the label design), если собираетесь сначала изменить отчет в **Конструкторе.** Затем щелкните мышью кнопку **Готово.**

Программа Access сохранит ваш отчет и затем откроет его в режиме **Предварительный** просмотр (рис. 11.15) или Конструктор, в зависимости от сделанного вами выбора. Если отчет открывается в Конструкторе, можно добавить дополнительные штрихи (можно поместить эмблему компании в правый верхний угол адреса и т. д.).

Отчет с наклейками - это обычный отчет Access, как те, с которыми вы познакомились в этой главе. Раздел **Область данных** содержит шаблон, определяющий способ размещения программой Access полей для одной наклейки, и затем этот шаблон копируется для заполнения всей страницы.

Единственное отличие отчета с наклейками от обычных отчетов - использование нескольких колонок. В этом случае **Область данных** (представляющая одну наклейку) может копироваться слева направо вдоль ширины страницы и затем сверху вниз. Такой способ обеспечивает компактную сетку с ячейками, содержащими наклейки. (Обычно **Область данных** копируется на странице только в одном направлении: сверху вниз.)

Вы можете создать собственный многоколоночный отчет, в котором записи помещаются в несколько колонок. Нужно просто открыть отчет в **Конструкторе,** убедиться в том, что **Область данных** достаточно узкая, затем выбрать на ленте **Инструменты конструктора отчетов | Параметры страницы → Разметка страницы → Столбцы** (Report Design Tools | Page Setup → Page Layout → Columns). Этот выбор открывает диалоговое окно, в котором можно задать число колонок и расстояние между ними. Вы также можете выбрать расположение записей сначала сверху вниз, а затем слева направо или сначала слева направо, а потом сверху вниз. В любом случае проверьте внешний вид отчета в режиме предварительного просмотра, чтобы убедиться в том, что на странице все расположено как следует.

11.4. *Тонкая настройка отчетов с помощью свойств*

Как вы уже узнали, самый легкий способ тонкой настройки элементов управления в вашем отчете - кнопки панели инструментов. Но даже несмотря на то, что на панели множество полезных инструментов, на ней есть не все. За кадром у каждого элемента управления есть набор низкоуровневых параметров, называемых *свойствами.* Многие из них неизвестны, и ими редко пользуются. Некоторые известны небольшому числу необщительных фанатов Access. Но есть несколько удивительно полезных свойств, поскольку они предоставляют функциональные возможности, которые нельзя получить другим способом в программе Access. Вы можете найти и изменить эти параметры только в **Окне свойств.**

Подсказка

Окно свойств иногда полезно при разработке отчетов, но оно станет гораздо более важным, когда вы вплотную займетесь формами **в** *части IV* или добавлением кода в *части V*.

Для отображения **Окна свойств** выберите на ленте **Инструменты конструктора отчетов | Конструктор → Сервис → Страница свойств** (Report Design Tools | Design → Tools → Property Sheet). **Окно свойств** появится в правой части окна программы (рис. 11.16).

В **Окне свойств** можно производить тонкую настройку элементов управления поочередно. Выбирается объект выделением элемента управления в рабочей области **Конструктора** или в раскрывающемся списке в верхней части **Окна свойств.** Если нужно настроить конкретный элемент управления, обычно легче выделить его мышью в рабочей области отчета. В раскрывающемся списке элементы перечислены по именам, и программа Access не всегда применяет интуитивно самые понятные имена. Иногда они совпадают с лежащим в основе нолем (например, **ProductCategoryID),** а иногда нет (например, **Text3).**

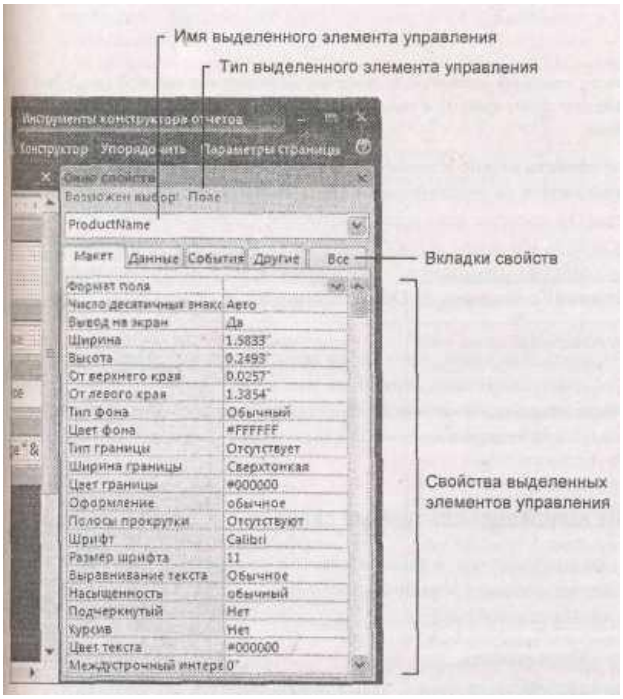


Рис. 11.16. В **Окне свойств** отображается список параметров (называемых свойствами) одного элемента управления. Полный список свойств можно увидеть на вкладке **Все,** а частичный список на одной из остальных вкладок. В данном примере выбран элемент управления **Поле**

У большинства элементов управления одинаковый набор свойств. Для того чтобы легче было просматривать этот утомляюще длинный список, **Окно свойств** разделено на следующие вкладки:

- **Макет** (Format) содержит наиболее часто изменяемые параметры, включая шрифт, его размер, цвет, границы и отступы;
- **Данные** (Data) указывает, откуда элемент управления получает информацию. Для элементов из **Области данных** на этой вкладке указывается имя связанного с ним поля;
- **События** (Event) позволяет вставить код на языке Visual Basic, который начинает действовать, если происходит что-то определенное. Вы узнаете больше о программном коде в *части V;*
- **Другие** (Other) включает свойство **Имя** (Name), определяющее имя элемента управления и несколько разнородных свойств, в большей степени относящихся к формам;
- **Все** (All) отображает полный набор свойств.

Подсказка

Для получения краткого описания непонятного свойства щелкните его кнопкой мыши для выделения в **Окне свойств** и затем найдите в нижней части окна программы, в строке состояния, его текстовое описание.

Интересно, что в **Окне свойств** можно изменять не только элементы управления. Можно корректировать параметры отчета (в верхней части **Окна свойств** выберите в раскрывающемся списке **Отчет** (Report)), которые определяют источник данных и способ представления или редактирования отчета. Вы также можете настроить параметры, относящиеся к определенным разделам отчета (например, **ЗаголовокОтчета, ПримечаниеОтчета, ВерхнийКолонтитул, НижнийКолонтитул** и **Область Данных)** и содержащие параметры разрывов страниц и дополнительного форматирования.

Конечно, одно дело - знать, что существует набор параметров, которые можно изменять, и совсем другое - знать наверняка, какие из них стоит корректировать. Окно **свойств** набито множеством параметров, которые не стоят вашего времени. В следующем разделе вы увидите табл. 11.1, в которой перечислены свойства, наиболее полезные для настройки отчетов.

11.4.1. *Корректировка самых широкоиспользуемых свойств*

Если **Окно свойств** все еще приводит вас в замешательство, выполните следующие действия. Они проведут вас через весь процесс внесения изменений.

1. Выделите в рабочей области отчета элемент управления. Его свойства появятся в **Окне свойств.**

1. Щелкните кнопкой мыши вкладку **Макет** и затем прокрутите список вниз до тех пор, пока не найдете параметр **Цвет фона** (Back Color).

Параметр **Цвет фона** определяет цвет отображения фона, расположенного за текстом в элементе управления.

2. Щелкните кнопкой мыши поле со значением параметра **Цвет фона.** В поле появится кнопка (...) с многоточием. Щелкните ее мышью. На экран будет выведено окно выбора цвета.

В поле **Цвет фона** отображается числовой код, обозначающий цвет. До тех пор пока вы не запомните сотни тысяч замысловатых кодов цветов, вам покажется более удобным выбор цвета в диалоговом окне указателя цвета.

3. Выберите цвет.

Новый цвет появится немедленно вместе с числовым кодом, обозначающим его.

Этот метод можно применить для задания фона раздела в отчете. Из раскрывающегося списка в **Окне свойств** просто выберите раздел, например **ЗаголовокОтчета** (ReportHeader) или **ОбластьДанных** (Detail), и затем выполните описанные действия.

Подсказка

Если вы изменяете фоновый цвет раздела отчета, не забудьте изменить цвет фона у всех элементов управления в этой части отчета или же около них будут отображаться белые поля. Можно сразу выделить все элементы управления, нуждающиеся в корректировке, заключив их с помощью мыши в рамку выделения или с нажатой клавишей <Shift> поочередно щелкнув кнопкой мыши каждый из них. Затем следует перейти в **Окно свойств** для внесения ваших изменений.

В данном примере проще было бы изменить цвет фона с помощью ленты. Но в **Окне свойств** можно изменить множество параметров, не имеющих эквивалентов на ленте. В табл. 11.1 приведено несколько полезных примеров свойств, все они отображаются на вкладке **Макет.**

Таблица 11.1. Полезные свойства элементов управления (на вкладке Макет)

Выбранный элемент	*Свойство*	*Описание*
Любой элемент управления, отображающий текст	**Выравнивание текста** (Text Align)	Обычно выравнивание зависит от типа выводимых данных. Например, программа Access выравнивает по правому краю числа и даты. Если вы хотите сравнивать длинный список чисел, такой порядок имеет смысл. Если же вы хотите соединить ваши числа с фрагментами текстовых данных, можно выбрать другой вариант выравнивания, например по левому краю или по центру
Поле (Text box)	**Формат поля** (Format)	Как правило, Access применяет числовой формат, который определяется типом данных, и нет нужды беспокоиться о выборе этого параметра. Но если применяется вычисляемое выражение *(см. разд. "Вычисляемые поля" главы 7),* Access выводит результат в *Основном* числовом формате, даже если вам нужны два знака в дробной части и символ валюты. Для исправления формата выберите тот, который вам нужен (например, *Денежный).* В *разд. "Числовой формат" главы 2* приведены другие возможные варианты
Поле (Text box)	**Расширение** (Can Grow)	Если в поле **Расширение** установить значение *Да,* Access расширит его до размера, вмещающего все содержимое. При создании простого табличного отчета свойство **Расширение** включено для всех полей, но в этом нет

		необходимости в других типах отчетов, которые формируются с помощью Мастера отчетов. Когда свойство не включено, Access обрезает длинное содержимое до размера доступного пространства
Отчет (Report)	**Режим по умолчанию** (Default View)	Определяет, в каком режиме открывается отчет, если дважды щелкнуть его кнопкой мыши в области переходов. Обычно он открывается в **Режиме отчета**
Отчет (Report)	**Верхний колонтитул** (Page Header) и **Нижний колонтитул** (Page Footer)	Если выбран стандартный вариант **Все страницы** (All Pages), верхний и нижний колонтитулы выводятся на каждой странице. Вы можете убрать верхний или нижний колонтитулы с тех страниц, которые содержат заголовок отчета и примечание отчета
Верхний Колонтитул (PageHeaderSection), **Нижний Колонтитул** (PageFooterSection), **ЗаголовокОтчета** (ReportHeader), **ПримечаниеОтчета** (ReportFooter)	**Режим вывода** (Display When)	Обычно эти разделы отображаются на экране и в финальной распечатке. Вы можете включить их либо в экранное представление, либо в распечатку, но не в оба варианта одновременно
Область Данных (Detail)	**Конец страницы** (Force New Page)	Стандартное значение этого свойства - *Отсутствует* (None), и программа Access размещает на каждой странице максимально возможное количество информации, прежде чем перейти на следующую страницу. Вы можете выбрать значение *До раздела* (Before Section) для того, чтобы каждую запись начинать с новой страницы. Другие значения свойства **Конец страницы** предназначены для совместного использования с группировкой
ОбластьДанных (Detail)	**Не разрывать** (Keep Together)	Если задано значение *Да*, Access никогда в распечатке не разделяет **Область данных** разрывом страницы. Если на странице недостаточно места для вывода полной записи, Access сразу переходит на следующую страницу и на ней возобновляет печать

11.5. *Выражения*

Ранее в этой главе вы узнали, как вставлять **Подпись** и задавать текст в ней. Но если вы посмотрите на элементы управления типичного отчета, то быстро заметите, что не все из них используют обычный текст. Обратите внимание на дату или номера страниц (которые появляются в правом верхнем углу простого отчета). Оба эти типа данных отображаются в обычных элементах управления, **Полях,** но текст выглядит иначе. Он начинается со знака равенства (=), что свидетельствует о наличии выражения.

Выражения позволяют отображать динамические значения. Никто не захочет вводить постоянную дату в отчет, поскольку будет вынужден корректировать ее каждый раз, когда нужно получить распечатку. Вместо этого применяют выражение, такое как `=Date()`, заставляющее программу Access считать с часов компьютера текущую дату и вывести ее на экран.

Выражения - не новость. Вы узнали о них, когда знакомились с запросами в *главе 7.* Но, возможно, до настоящего момента вы не догадывались, что они также уместны и в отчетах.

Можно вставить в отчет собственные выражения для отображения динамических данных или выполнения вычислений, базирующихся на других полях. Предположим, что вы хотите улучшить свадебный список, скомбинировав имена и фамилии гостей в компактный однострочный вывод. Как рассказывалось в *разд. "Выражения с текстовыми значениями" главы 7,* символ & - то, что требуется для соединения фрагментов текста. Далее приведено нужное вам выражение:

```
=FirstName & " " & LastName
```

Подсказка

Вернитесь к *главе* 7, чтобы просмотреть информацию о выражениях, разных типах вычислений, которые можно выполнять, и различных функциях, которые можно в них применять.

Ввести выражение в **Подпись** нельзя, поскольку этот элемент управления может содержать только текстовую константу. Вместо него вам нужен элемент управления **Поле. (**Access также применяет элемент **Поле** для вывода большинства полей БД.)

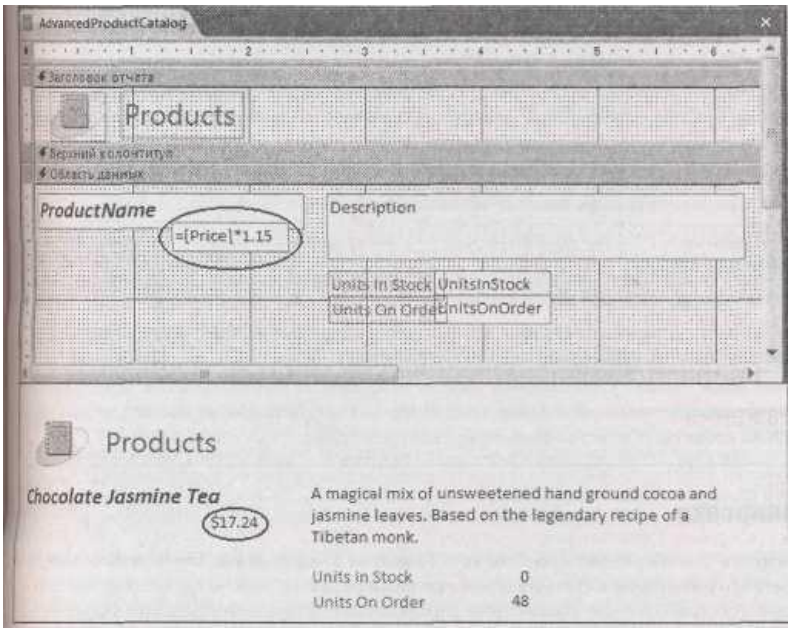


Рис. 11.17. *Вверху:* в **Конструкторе** выражение выглядит как обычное текстовое значение. Внизу: в режиме **Предварительный просмотр** программа Access выполняет вычисление и отображает результат

После того как вы добавили в рабочую область отчета **Поле,** для выделения щелкните его кнопкой мыши. Указатель мыши превратится в текстовый курсор. Щелкните элемент кнопкой мыши еще раз для редактирования текста и затем введите ваше выражение, не забудьте начать со знака равенства (рис. 11.17).

Часто задаваемый вопрос.

Ошибки выражений

Почему выражение отображает #Error в режиме предварительного просмотра?

Как ни странно код #Error означает, что в вашем выражении не все правильно. Программа Access пытается вычислить его, попадает в аварийную ситуацию и выводит вместо результата сообщение об ошибке.

Часто легко увидеть наличие проблемы, если вернуться в **Конструктор** и посмотреть ошибочный элемент управления Поле. Обычно в верхнем левом углу элемента виден зеленый треугольник, сигнализирующий о наличии проблемы. Выделите его, и слева немедленно появится пиктограмма ошибки. Можно поводить указателем мыши поверх пиктограммы, чтобы увидеть описание проблемы, и щелкнуть ее кнопкой мыши для отображения короткого меню возможных способов ее решения, которые можно применить, и вариантов контроля наличия ошибок,

способных заставить программу Access игнорировать эту проблему в будущем.

Сообщения об ошибках программы Access известны своей туманностью, поэтому даже когда вы найдете пиктограмму ошибки и получите подробности, реальная проблема может остаться во мраке неизвестности. Для того чтобы вернуть вас на правильную дорогу, предлагаю краткий перечень наиболее распространенных проблем:

- вы забыли начать выражение со знака равенства;
- вы ошиблись,' набирая имя поля, или сослались на поле, которого нет в базовой таблице или запросе;
- вы вставили непарный набор скобок;
- у элемента управления **Поле** то же имя, что и у одного из полей, используемых вами, Если у вас есть выражение =UnitsInStock+UnitsOnOrder и ваш элемент управления **Поле** назван **UnitsInStock,** Access приходит в замешательство. Для решения проблемы переименуйте **Поле** (например в **UnitsInStockCalculation)** с помощью **Окна свойств.** (Свойство **Имя** отображается в верхней части вкладки **Все.)**

11.6. *Группировка*

Группировка - неоценимое средство для придания смысла большим объемам данных счет упорядочивания их в группах меньшего размера. Затем можно выполнять вычисления в каждой отдельной группе. Рассмотрим список заказов в компании Boutique Fudge. В зависимости от того, как вы сгруппируете данные, можно посмотреть, лучше ли продается шоколадное молоко, чем шоколадное пиво, или сильнее ли клиенты из Нью-Йорка жаждут какао, чем клиенты из Алабамы, и т. д.

Существуют три способа применения группировки для анализа информации в отчете.

■ *Применение группировки в запросе.* В этом случае в ваш отчет не включаются подробности. Он только отображает вычисленные суммы, средние значения, максимумы или минимумы. Вам не понадобятся изощренные колдовские приемы создания отчета для решения этой задачи - создайте сводный отчет с группировкой (как описано в *разд. "Итоговые данные " главы* 7) и затем используйте его для формирования отчета.

■*Применение группировки в отчете.* В этом случае можно разделить информацию большого объема на подгруппы. При этом бы можете видеть все данные и применять промежуточные итоги и другие вычисления. Можно также добавить несколько уровней группировки для выявления глубинных тенденций.

■*Применение подчиненных отчетов.* Этот метод создает тот же эффект, что и группировка в отчете. Единственное отличие - формирование отчета из двух отдельных частей.

Примечание

Подчиненные отчеты - во многом наследие более ранних версий программы Access. В Access 2007 функциональные возможности группировки настолько улучшены, что в подчиненных отчетах больше нет нужды. В данной книге уделяется внимание наиболее эффективному применению группировки, а подчиненные отчеты пропущены.

11.6.1. *Группировка в отчетах*

Для создания групп выполните следующие действия.

1.Перейдите в **Режим макета** или **Конструктор.**

2.Выберите поле, которое хотите использовать для сортировки. Обычно следует сортировать таблицу по тому полю, которое планируется применять для группировки.

Если вы хотите группировать по полю **ProductCategoryID** (идентификатор категории товара), которое формирует отдельную группу товаров для каждой категории, то начать следует с сортировки результатов по полю **ProductCategorylD.** В этом случае все товары одной группы перечисляются один за другим. (Можно сортировать и но однозначно определяемому полю из связанной таблицы **ProductCategories,** например, **CategoryName.** Поскольку у каждой категории товара - свое название, это поле отсортирует товары по группам столь же эффективно.)

Примечание

Если вы создали подстановку, у программы Access хватит сообразительности для использования

при сортировке более информативного поля, а не связанного с ним поля. Поле ProductCategoryID использует подстановку, которая выводит на экран название соответствующей категории, а не связанный с ним идентификатор категории, до которого на самом деле никому нет дела. Когда вы щелкните правой кнопкой мыши поле ProductCategoryID и выберите команду сортировки, программа Access применит поле CategoryName для сортировки.

3. Для сортировки данных щелкните правой кнопкой мыши поле, по которому собираетесь сортировать, и выберите команду сортировки (например, **Сортировка от А до Я** (Sort A to Z) или **Сортировка от минимального к максимальному** (Sort Smallest to Largest)).

Точное название в меню команды сортировки зависит от типа данных, хранящихся в поле.

4 Щелкните правой кнопкой мыши поле, которое хотите использовать для группировки, и выберите команду **Группировка** (Group On). Программа Access отсортирует ваши результаты по этому полю и затем сгруппирует их.

На рис. 11.18 и 11.19 показаны два отчета, в которых товары сгруппированы по категориям.

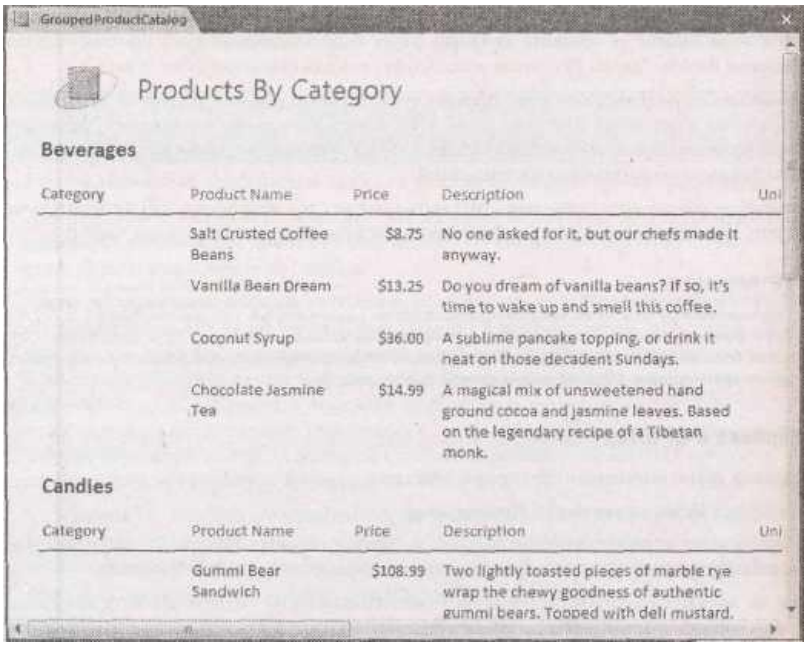


Рис. 11.18. Этот простой табличный отчет отсортирован и сгруппирован по категориям за несколько быстрых щелчков мыши

Подсказка

Когда используется группировка, возможно, нет смысла оставлять заголовки столбцов в разделе макета страницы, т. к. каждый заголовок группы разрывает таблицу. Часто гораздо лучше располагать заголовки столбцов под названием группы так, чтобы они выводились в начале каждой группы (а не в верхней части каждой страницы). На рис. 11.18 применен этот подход (на рис. 11.19 он не нужен, поскольку в отчете вообще не используются заголовки.) К сожалению, для того чтобы применить эту более привлекательную организацию данных, вам придется удалить поля из автоматически создаваемого макета *(см. разд. "Удаление полей из макета' ранее в этой главе).*

Группировка включает дополнительные разделы в отчет. Если вы группируете с помощью ноля **ProductCategoryID,** ваш отчет получает новый раздел **Заголовок группы 'ProductCategoryID'** (ProductCategoryID Header), который программа Access располагает непосредственно над **Областью данных** (рис. 11.20). Это название группы содержит сведения о группировке - в данном случае, категории товара. В **Области данных** находятся данные всех записей, включенных в группу.

Примечание

Как вы увидите позже, можно добавлять несколько уровней группировки. В этом случае пса-грамма Access вставляет один заголовок группы для каждого уровня.

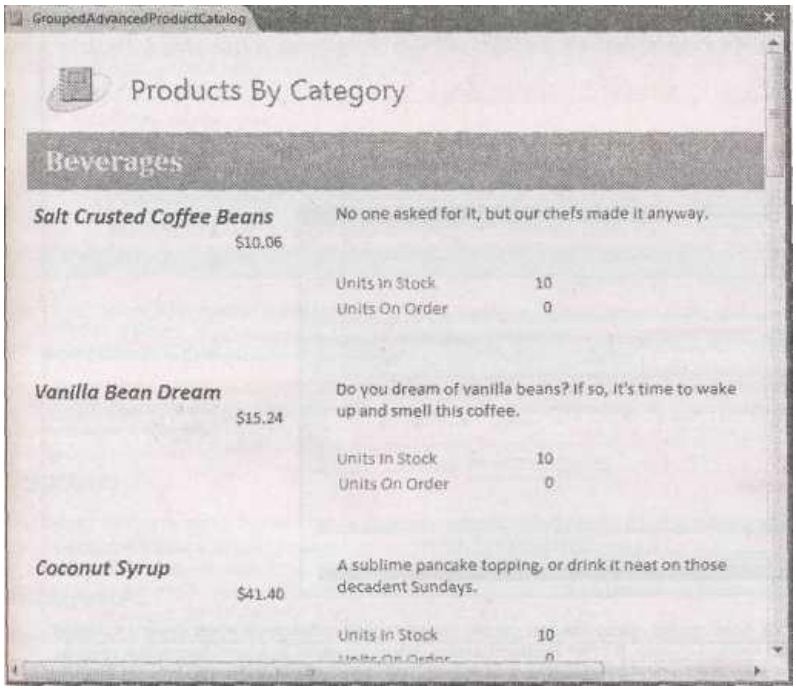


Рис. 11.19. Группировка так же хорошо функционирует и в отчетах со сложными нетабличными макетами. Но, возможно, придется потратить немного больше времени на определение начала и конца группы, поэтому подумайте о задании другого цвета фона для разделения категорий (с помощью свойства **Цвет фона)** и их выделения, как в данном примере. Или же можно использовать элемент управления **Линия** для создания разделительной черты в начале каждой категории. На рис, 11.20 показан этот отчет в **Конструкторе**

Тонкая настройка с помощью панели *Группировка, сортировка и итоги*

11.6.2. *Когда группировка задана, у вас появляются дополнительные возможности:*

- можно вставить дополнительную сортировку в пределах каждой подгруппы;
- можно выполнить сводные вычисления для каждой группы;
- можно расставить разрывы страниц в начале каждой новой группы.

Любой из этих вариантов легче всего добавить с помощью панели **Группировка, сортировка и итоги** (Group, Sort, and Total). Для ее отображения в **Конструкторе** выберите па ленте **Инструменты конструктора отчетов | Конструктор → Группировка и итоги → Группировка и сортировка** (Report Design Tools | Design → Grouping & Totals → Group & Sort) или в **Режиме макета - Работа с макетами отчетов | Формат → Группировка и итоги → Группировка и сортировка** (Report Layout Tools | Formatting → Grouping & Totals → Group & Sort).

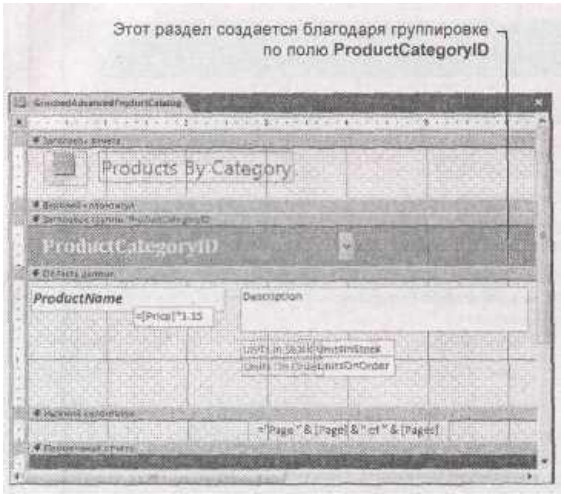


Рис. 11.20. Конструктор предоставляет самый легкий способ добавления содержимого в раздел заголовка каждой группы. В отчет ProductCatalog можно вставить в раздел **Заголовок группы 'ProductCategoryID'** дополнительные поля из таблицы ProductCategories (например, поле Description)

Панель **Группировка, сортировка и итоги** выводится в нижней части окна программы. На рис. 11.21 показан примерный вид экрана при отображении отчета (см. рис. 11.19) с товарами, сгруппированными по категориям.

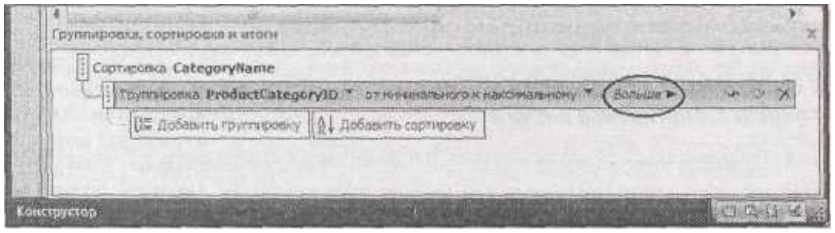


Рис. 11.21. У данного отчета один уровень сортировки (по полю **CategoryName** в алфавитном порядке) и один уровень группировки (по полю **ProductCategorylD).** Для отображения дополнительных параметров для каждого уровня выберите его и щелкните мышью кнопку **Больше** (обведена). На рис. 11.22 показаны параметры группировки, которые можно изменить

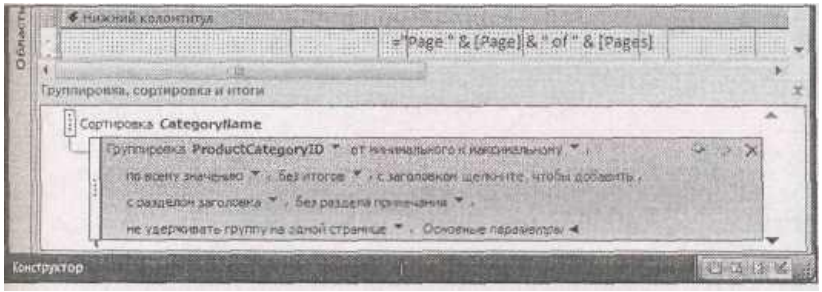


Рис. 11.22. Панель **Группировка, сортировка и итоги** позволяет быстро задать подсчет промежуточных итогов, включение разделов заголовка и примечания и установку разрывов страниц для каждого имеющегося уровня группировки

В следующих разделах описываются функциональные возможности панели Группировка, сортировка и итоги.

Сортировка...

Позволяет выбрать поле, применяемое для сортировки. На рис. 11.21 поля сортируются по полю **CategoryName** и затем группируются по полю **ProductCategorvID.**

Группировка ...

Предоставляет возможность задать поле, применяемое для группировки. Этот параметр позволяет моментально включить группировку.

Сортировка от А до Я/от минимального к максимальному

Изменяет порядок сортировки. Точное название параметра зависит от типа данных, можно сортировать текст в алфавитном порядке, числа по возрастанию или убыванию или даты в хронологическом порядке.

По всему значению

Заставляет программу Access создавать индивидуальную группу для каждого отдельного значения в поле группировки. Если вы группируете по полю **ProductCategoryID**, этот параметр обеспечивает размещение каждой категории товаров в отдельной группе. Иногда подобный подход приводит к созданию слишком большого количества групп, что затрудняет последующий анализ (и расходует груды бумаги). В подобных ситуациях необходимо формировать более крупные группы, включающие больше записей. Если группируются товары по ценам или заказы

по датам, возможно, лучше группировать по диапазону значений, как показано на рис. 11.23.

С итогами...

Промежуточные итоги - самое популярное свойство группировки. Они позволяют сравнивать количественно одну группу с другой. Панель Группировка, сортировка и итоги позволяет выполнять вычисления промежуточных итогов любых числовых полей (рис. 11.24).

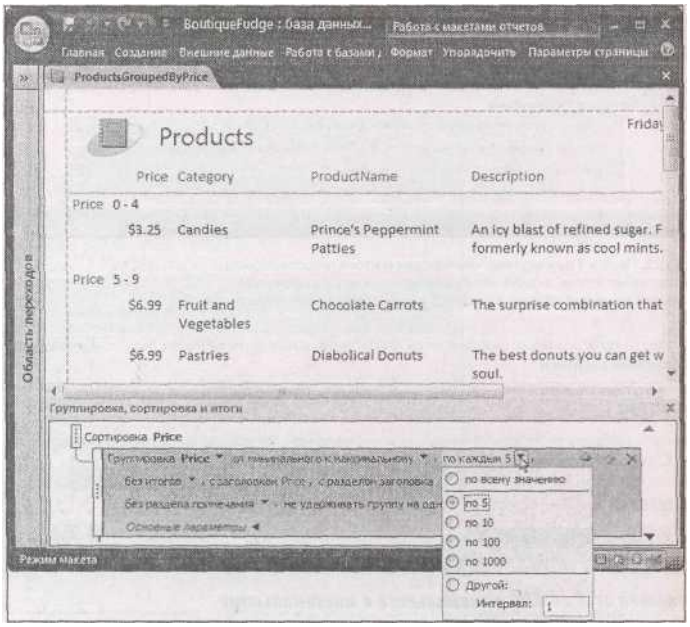


Рис. 11.23. В случае дат или числовых полей можно выбрать создание группы, включающей целый диапазон значений. В данном примере группы формируются по цене с интервалом 5 долларов. В первую группу включены товары с ценой менее 5 долларов, во вторую - с ценами от 5 до 9.99 долларов и т. д.

В зависимости от того, что вы хотите получить, можно сосчитать количество значений, суммировать их, вычислить средние или определить максимальные и минимальные значения в группе. Эту информацию можно поместить в заголовок, который выводится в начале каждой группы, или в примечание, следующее в конце. В завершение вы можете дополнить отчет финальным общим итогом, который суммирует все промежуточные.

С заголовком...

Щелкните кнопкой мыши эту область для вставки текстового заголовка, который выводится в заголовке категории, в начале каждого раздела категории. Конечно, можно вставить заголовок и без помощи панели посредством добавления элемента управления **Подпись** в **Конструкторе,** но этот параметр предоставляет быстрый и удобный способ.

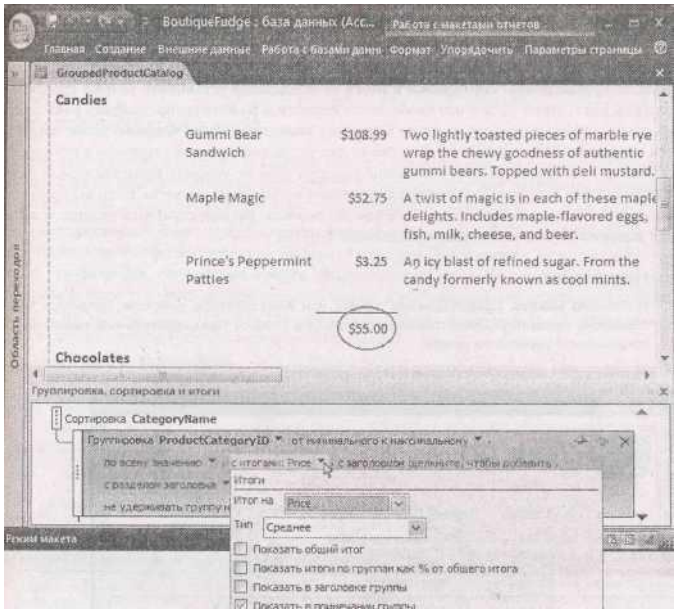


Рис. 11.24. В данном примере список товаров группируется по категории, передняя для каждой категории цена выводится в примечании (обведено)

С разделом заголовка/С разделом примечания

Можно вставить раздел заголовка в начале каждой группы и раздел примечания в конце. После добавления этих разделов в них можно вставить любое содержимое в **Конструкторе.** Чаще всего их используют для вывода информации, относящейся к группе в целом, отображают промежуточные итоги или рисуют разделительные линии с помощью элемента управления **Линия.**

Удерживать группу на одной странице

Этот режим позволяет избежать висячих заголовков категорий. В примере с каталогом товаров он гарантирует, что в конце страницы не появится название группы, например Beverages (напитки), а товары группы при этом не перейдут па следующую страницу.

Обычно программа Access не предотвращает неуместные разрывы страницы. Access просто старается заполнить до конца каждую страницу. Если вас это не устраивает, есть два варианта. Можно выбрать режим обязательного размещения всей группы на одной странице (конечно, при условии, что группа занимает меньше одной страницы) или сохранения на одной странице заголовка группы и, по крайней мере, одной записи, принадлежащей группе,

Панель **Группировка, сортировка и итоги** не предлагает установить разрыв страницы в начале каждой группы. Для этого необходимо перейти в **Конструктор,** выбрать раздел заголовка группы и затем выбрать на ленте **Страницу свойств** в группе **Формат,** найти параметр **Конец страницы.** Задать значение *До раздела* для установки разрыва страницы в начале каждого раздела или *После раздела* - в конце раздела. (Вы не увидите разницы между этими вариантами, пока в отчете используются заголовок **и** примечание отчета. Если вы применяете заголовок отчета и используете значение *До раздела,* разрыв страницы устанавливается между заголовком отчета и первым разделом.)

Примечание

В **Режиме макета, Представлении отчета** или **Конструкторе** действие параметра **Конец страницы** незаметно. Оно отображается только в режиме **Предварительный просмотр** или при реальной распечатке отчета.

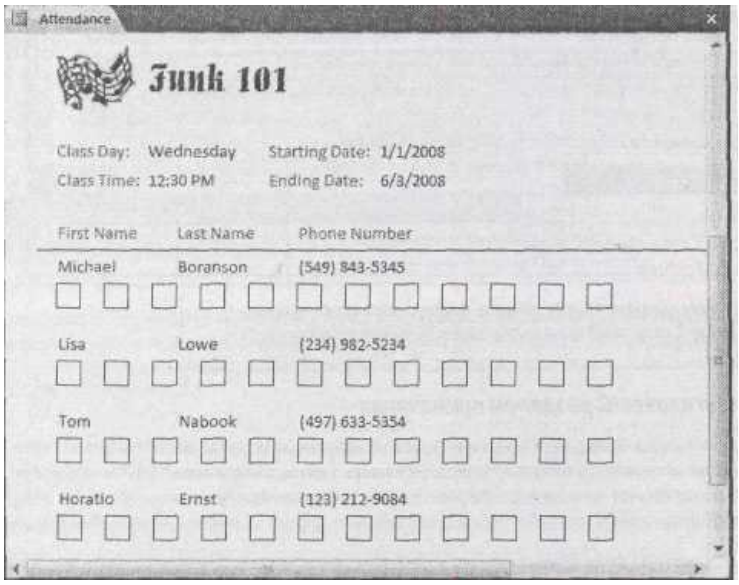


Рис. 11.25. В этом списке посещаемости для создания распечатки, совсем не похожей на типичный отчет, применяется несколько приемов, о которых вы узнали в данной главе. Отчет выводит список студентов, сгруппированных по классам. В отчете нет заголовка, а в заголовок группы для каждого класса вставлено несколько ключевых данных из таблицы **Classes.** Благодаря установке разрывов страниц список каждого класса начинается с новой страницы и за данными о

каждом студенте следует строка нарисованных от руки элементов управления **Прямоугольник,** в которых можно отмечать посещаемость

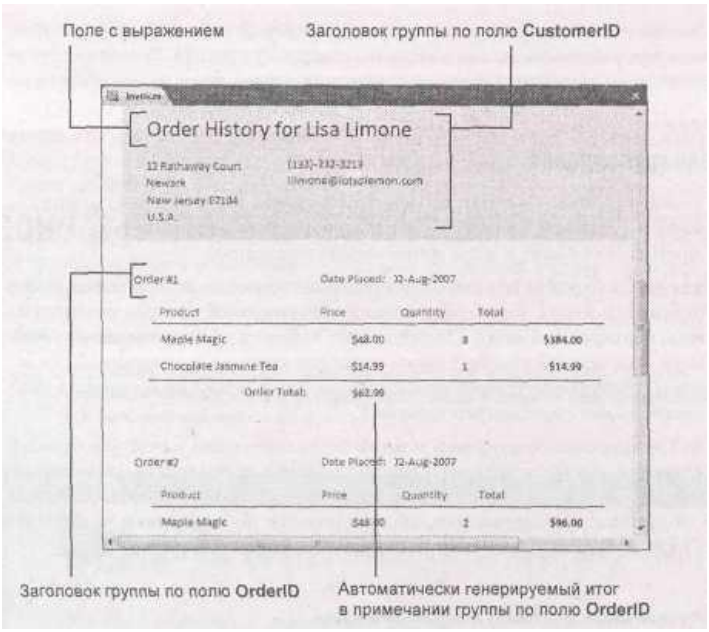


Рис. 11.27. В данном отчете информация заказа преобразуется в подготовленный для печати счет с помощью группировки и еще нескольких приемов, которые вам уже знакомы. Исходная информация извлекается из таблицы **OrderDetails** (дополненная данными из таблиц **Orders, Product** и **Customer).** Элементы управления **Линия** отделяют разные разделы счета, автоматически генерируемые итоги сообщают вам общую стоимость заказа, в выражениях комбинируется несколько полей и дополнительные текстовые фрагменты (например, "Order History for" (сведения о сделанных заказах) и "Order #" (номер заказа)). Данный пример включен в БД Boutique Fudge с загружаемым из Интернета содержимым, иллюстрирующим материал данной главы

Если у вас несколько уровней группировки, некоторую информацию можно скрыть, так чтобы отображались только сводные данные. В **Режиме макета** просто выберите **Работа** с **макетами отчетов | Формат → Группировка и итоги → Без подробностей** (Report Layout Tools | Formatting → Grouping & Totals → Hide Details). Если применить этот прием к примеру, показанному на рис. 11.27, программа Access скроет построчное содержание заказа, и вы увидите лишь общую стоимость каждого заказа.

Подсказка

В примере со счетами создается отчет, печатающий счета для всех заказов вашей БД. Но вы можете воспользоваться фильтрацией *(см. разд. "Фильтрация" главы 3)* для отбора результатов для конкретного заказа или определенного клиента.

Часть IV

Разработка пользовательского интерфейса с помощью форм

12. Глава 12. Создание простых форм

Итак, вы научились создавать таблицы, хранящие наши данные, запросы, извлекающие их, и отчеты, подготавливающие данные к печати. Вы также создавали запросы на изменение, автоматизирующие обновления большого объема информации. Но реальные пользователи вашей БД (вы или кто-нибудь другой) будут тратить большую часть времени на совершенно другую работу: ежедневное обслуживание БД.

Эта работа включает просмотр, редактирование и вставку информации. В реальных БД этот процесс идет непрерывно. Ежедневно сотрудники компании Cacophone Studios добавляют новых студентов, отдел обслуживания клиентов компании Boutique Fudge помещает новые заказы, а устроители средневековой свадьбы компании Gothic Wedding уточняют план рассаживания гостей за праздничным столом. Куклы-болванчики покупаются, адреса меняются, покупки регистрируются, тестовые результаты фиксируются, и ваши данные растут и эволюционируют.

Ежедневное техническое обслуживание БД можно проводить на листе данных *(см. главу 3),* но это не самый легкий способ. Несмотря на то, что лист данных содержит множество информации на ограниченном пространстве, его обычно трудно использовать, особенно новичкам, недавно работающим в программе Access. Лучшее решение - *формы:* специализированные объекты БД, облегчающие просмотр и редактирование информации в таблице.

Примечание

Помните о том, что при использовании программы Access в деловой среде разные люди пользуются вашей БД. Вы можете создать ее, но другим пользователям нужно иметь возможность применять ее для выполнения разнообразных задач - как правило, ввода данных и поиска сведений. Эти пользователи могут не так хорошо, как вы, знать программу Access.

12.1. *Основные сведения о формах*

Формы получили свое название от бумажных форм, которые заполняются людьми, когда под рукой нет компьютера. В зависимости от ситуации вы можете создавать в программе Access форму, которая напоминает форму, применяемую вашей компанией или организацией. Если вы работаете в банке, вы можете создать в Access форму, которая воспроизводит информацию, организованную так же, как в бумажной форме, применяемой клиентами. Такая организация данных облегчает копирование информации с бумаги в вашу БД. Но в большинстве случаев проектируемая вами форма не имеет аналога в реальной жизни. Такие формы создаются с нуля и используются для облегчения ввода данных.

Для того чтобы понять, почему формы - неотъемлемая часть почти всех БД, полезно рассмотреть слабые места листа данных. Далее перечислены некоторые характеристики, делающие формы предпочтительнее листов данных.

■ *Улучшенная компоновки.* На листе данных каждое поле занимает единственный столбец. Такая организация хороша для таблиц с небольшим числом столбцов. Если же таблица широка, она приводит к бесконечным прокруткам, необходимым для переходов к разным частям таблицы. В форме есть гарантия того, что нужные данные всегда находятся в поле зрения. Можно также использовать цвет, линии и изображения для разделения разных фрагментов содержимого.

■ *Дополнительная информация.* Вы можете включить в форму любой нужный вам текст, т. е. можно вставить пояснения, подсказывающие новичкам, какие данные следует предоставить. Можно также добавить вычисляемые данные - например, вычислить и отобразить стоимость всех покупок, сделанных клиентом, без необходимости создания кем-то отдельного запроса.

■ *Связи таблиц.* Во многих задачах требуется добавление записей в несколько связанных таблиц. Если новый клиент размещает заказ в БД Boutique Fudge, необходимо создать новую запись в таблицах **Customers** и **Orders,** а также одну или несколько записей в таблице **OrderDetails.** Форма позволяет выполнить всю эту работу в одном месте (при этом не нужно открывать два или три листа данных).

■ *Кнопки и другие графические элементы.* Формы поддерживают элементы управления: кнопки, ссылки, списки и другие полезные элементы пользовательского интерфейса, которые

можно вставлять в форму. Пользователь, работающий с вашей БД, сможет впоследствии щелкнуть мышью кнопку и запустить связанную с ней задачу (например, открыть другую форму или напечатать отчет).

Хорошо спроектированные формы - то, что компьютерные асы называют внешним представлением (front end) БД. В БД, применяющей формы, можно редактировать данные, выполнять поиск и обеспечивать выполнение повседневных задач, не касаясь листа данных.

12.1.1. *Создание простой формы*

Как и в случае отчетов, программа Access предоставляет легкий и довольно развитый способ создания формы. Он позволяет создать действующую форму, основанную на таблице или запросе. Внимательный читатель может заметить, что этот процесс в той или иной степени подобен процессу генерации простого отчета. Далее описано его применение.

1. В области переходов выберите таблицу или запрос, которые хотите использовать для генерации формы.

Попробуйте применить таблицу **Products** из БД Boutique Fudge.

Примечание

Если форма создается для родительской таблицы, связанной с другими таблицами, в результате вы будете иметь дело с несколько иным типом формы. Если создается форма для таблицы Categories (родительской для таблицы Products), вы получите в итоге двухчастную форму, позволяющую просматривать и изменять записи категорий и связанные записи товаров в каждой категории. Подробнее применение форм со связанными таблицами рассматривается в *главе 13*.

2. Выберите на ленте **Создание → Формы → Форма** (Create → Forms → Form).

На экране появляется новая вкладка с формой в **Режиме макета.** В простой форме одновременно отображается одна запись с выводом каждого поля в отдельной строке (рис. 12.1). Если в вашей таблице много полей, программа формирует несколько столбцов (рис. 12.2).

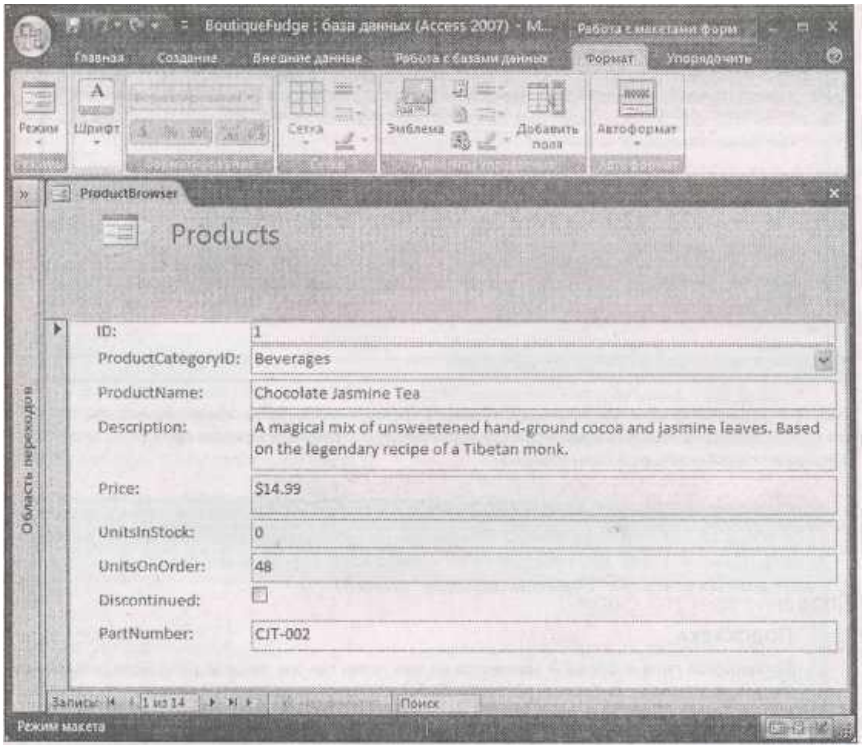


Рис. 12.1. В этой простой форме для таблицы **Products** уже присутствует разумная основа. Программа Access применяет поля ввода для всех текстовых полей, раскрывающийся список для полей с подстановкой (в данном примере поле **ProductCategoryID)** и флажок для всех полей с данными логического типа (например, поле **Discontinued). В** ней некоторые поля (например, **Description)** больше других, поскольку программа замечает, что у лежащего в основе поля таблицы допустимая длина больше максимально допустимой

Подсказка

Хорошие навыки проектирования очень пригодятся, когда вы начнете создавать формы. Если ваши тестовые поля созданы в расчете на хранение большего числа символов, чем им требуется (в соответствии со значением свойства **Размер поля,** описанного в *разд. "Длина текста" главы 2),* форма будет содержать огромные поля ввода, съедающие значительное пространство. Придется изменять их размер вручную.

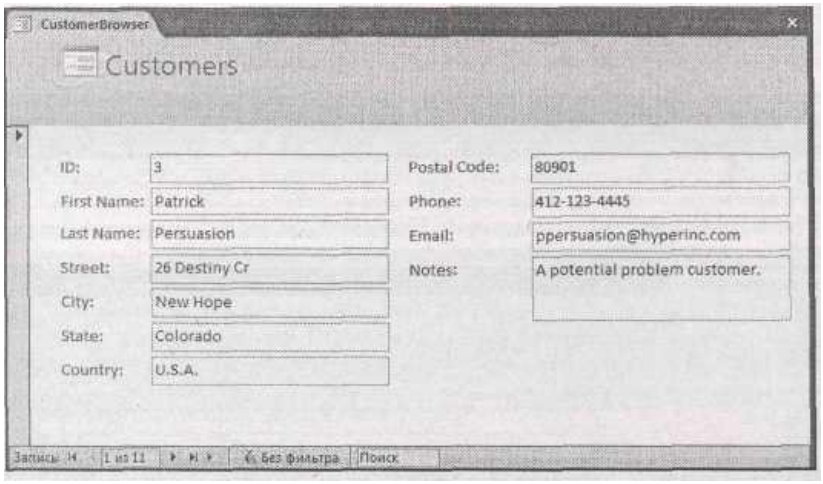


Рис. 12.2. В данной форме для таблицы **Customers** программа Access не может разместить все поля с помощью обычной компоновки, отводящей одну строку для каждого поля. В место этого добавлен еще один столбец

Когда форма создается впервые, программа Access располагает поля сверху вниз в том порядке, в каком они определены в таблице. Реорганизация полей на листе данных не играет никакой роли. Но программа не отображает в форме все столбцы, скрытые на листе данных *(см. разд. "Скрытие столбцов " главы 3).*

Подсказка

Вставляются поля в форму и удаляются из нее точно так же, как в отчете. Если окно **Список полей** не открыто, выберите на ленте **Работа с макетами форм | Формат → Элементы управления → Добавить поля** (Form Layout Tools | Formatting → Controls → Add Existing Fields). Затем перетащите в форму с помощью мыши нужное поле из окна **Список полей.** Для удаления поля щелкните его кнопкой мыши и нажмите клавишу <Delete>. Но помните о том, что часто формы применяются для добавления записей, и если вы хотите сохранить такую возможность, нужно, чтобы в вашей форме присутствовали все поля, необходимые таблице.

3. Расположите поля на форме, как хотите, перетащив их мышью в нужное место.

Несмотря на то, что простые формы выглядят иначе, нежели простые отчеты, с которыми вы познакомились в *главе 10,* работать с ними можно почти так же, как с отчетами. Самый легкий способ компоновки формы - перемещение полей из одного места в другое с помощью мыши (рис. 12.3).

4. Измените ширину столбцов.

Когда новая форма создается в Режиме макета, программа Access делает все поля достаточно широкими.

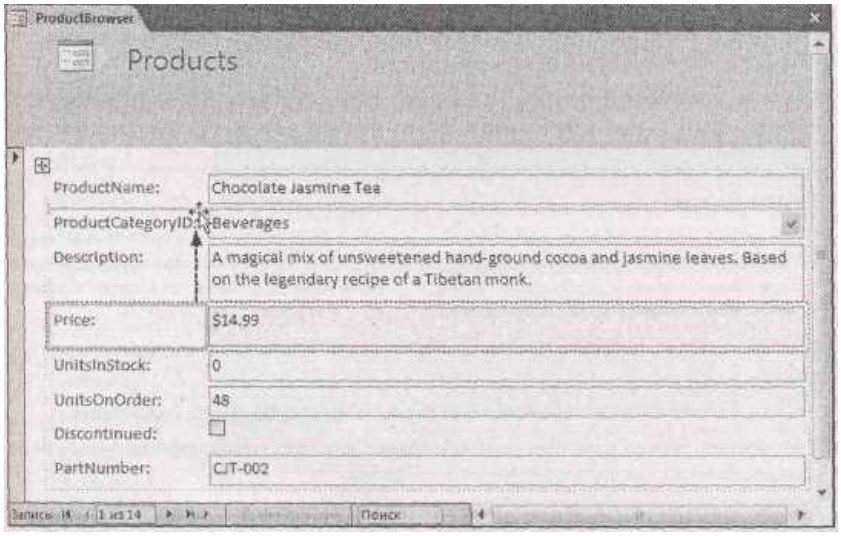


Рис. 12.3. Для перемещения поля перетащите его с помощью мыши в другое место. Программа Access переместит остальные поля соответственно. В данном примере поле Price переносится в верхнюю часть формы непосредственно под поле ProductName. Access отодвигает все остальные поля вниз для того, чтобы освободить место

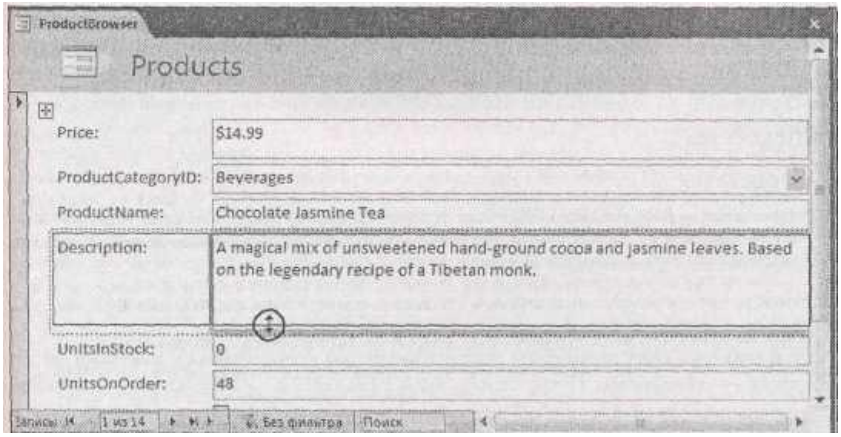


Рис. 12.4. Увеличивается высота поля **Description** для размещения в нем большего числа строк. Можно также сделать поле уже или шире, но есть опасность - подобные действия влияют на весь столбец. В данной форме для таблицы **Products** у всех полей одинаковая ширина

Как правило, нужно сжать их для создания более компактной формы. Кроме того, длинные строки текста трудно читать, поэтому лучше отображать информацию большого объема в более узком и высоком текстовом поле.

Для этого выделите соответствующее поле щелчком кнопки мыши, вокруг него появится прямоугольный контур желтого цвета. Затем перетащите мышью одну из его сторон. На рис. 12.4 показан этот процесс в действии.

Примечание

Возможно, вам потребуются изменения, которые нельзя выполнить с помощью перетаскивания мышью, например, добавление нового столбца или задание разной ширины полей. Для внесения таких изменений необходимо понять, что такое макеты, описанные в *разд. "Создание улучшенных макетов" далее в этой главе.*

5. Если необходимо, можно щелкнуть кнопкой мыши имя поля и отредактировать его текст.

Эта возможность позволит изменить имя **ProductCategoryID** просто на **Category.**

6. При желании можно изменить форматирование для того, чтобы сделать форму более привлекательной, сменив гарнитуру и цвет шрифта.

Быстрее всего изменить формат вашей формы, если выбрать соответствующую часть (щелчком кнопки мыши) и использовать кнопки в группе ленты **Работа с макетами форм | Формат → Шрифт** (FontForm Layout Tools | Formatting → Font). Эту группу ленты можно также применять для настройки отображения программой Access числовых значений. Вы познакомились со всеми возможными вариантами форматирования в *разд. "Форматирование столбцов и заголовков столбцов" главы 10,* когда создавали базовые отчеты.

Часто возникает необходимость иначе отформатировать определенные поля для того, чтобы выделить важную информацию. Можно также задать формат заголовка формы, раздела заголовка и фон формы. На рис. 12.5 показан пример разумного форматирования полей.

Подсказка

Для одновременного выбора нескольких фрагментов формы держите нажатой клавишу <Ctrl>, когда щелкаете их кнопкой мыши. Этот прием позволяет задать одинаковое форматирование одновременно для нескольких частей.

Если торопитесь (или сомневаетесь в стилистическом оформлении), можно применить отличное средство Access, названное **Автоформат,** для внесения целого набора связанных изменений. Просто выберите вариант из группы ленты **Работа с макетами форм | Формат → Автоформат** (Form Layout Tools | Formatting → AutoFormat) (в которую включены те же варианты, которыми вы пользовались при оформлении отчетов).

7. Сохраните форму.

Форму можно сохранить в любой момент, если выбрать кнопку **Office → Сохранить** (Office → Save). Если вы закроете форму без сохранения, программа Access в этот момент предложит вам сохранить ее.

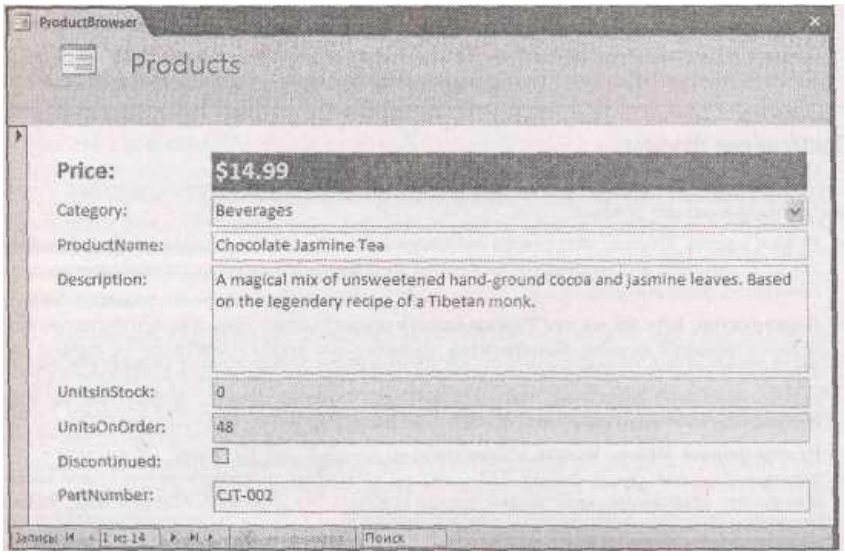


Рис. 12.5. Имя поля (например, **Price)** и поле со значением можно выделить по отдельности, что означает возможность различного форматирования этих компонентов. В этой форме задан затененный фон для полей **Price, UnitsinStock и UnitsOnOrder.** Кроме того, у поля **Price** и его имени увеличен размер шрифта, что позволяет выделить содержащуюся в них информацию

На профессиональном уровне.

Поля типа Счетчик в формах

Лучше всего однозначно идентифицировать каждую запись с помощью поля типа Счетчик *(см. разд. "Счетчик" главы 2).* Когда вставляется запись, программа Access вносит значение в поле **Счетчик.** Во все таблицы, которые вы видели в этой книге, включено поле, названное Код (ID) и использующее данные типа Счетчик.

Значения типа Счетчик может устанавливать только программа Access. По этой причине вы, возможно, не захотите отображать такое поле в своих формах. (Если вы решили не выводить его, просто выделите это поле в Режиме макета и нажмите клавишу <Delete>.) Но есть несколько

причин, которые могут заставить вас все же сохранить это поле.

• *Вы используете поле типа* ***Счетчик*** *для какой-либо канцелярской работы.* Компания Cacophone Studios сохраняет ID-номер каждого студента в его регистрационных документах. Когда впоследствии потребуется найти запись о студенте, легче использовать ID-номер, чем искать студента по имени.

• *Вы применяете поле типа* ***Счетчик*** *как отслеживающее значение или номер подтверждения (confirmation number).* После того как введена запись нового заказа в БД Boutique Fudge, вы можете зафиксировать ID-номер записи заказа. В следующий раз, когда возникнет вопрос о заказе (доставлен ли он?), можно использовать ID-номер заказа для его поиска.

В некоторых случаях ID-номер можно поместить в нижнюю часть формы вместо его обычного расположения в верхней части. Такой подход устраняет путаницу. (Меньше вероятность того, что люди будут пытаться ввести ID-номера при создании новых записей.)

12.1.2. *Применение формы*

Теперь, когда первая форма создана, самое время подвергнуть ее тестовой проверке. У всех форм есть три разных режима.

■ **Режим макета.** Именно этот режим использовался до настоящего момента. Он позволяет увидеть внешний вид формы (с реальными данными), изменить расположение полей и применить форматирование.

■ **Конструктор.** В то время как **Режим макета** предоставляет самый простой способ усовершенствования формы, **Конструктор** наделяет вас всеми средствами ее тонкой настройки. В этом режиме вы не увидите реальные данные. На экране отображается проект, сообщающий программе Access о том, как сконструировать форму. Вы начнете использовать **Конструктор** чуть позже в этой главе.

■ **Режим формы. Режим макета** и **Конструктор** введены для того, чтобы помочь вам создавать и улучшать вашу форму. Но когда вы ее довели до совершенства, самое время прекратить конструирование вашей формы и начать использовать ее для обзора таблицы, просмотра содержащейся в ней информации, внесения изменений и добавления новых записей.

Примечание

Когда форма в области переходов открывается двойным щелчком кнопки мыши, она отображается в **Режиме формы.** Если вам не нужен этот режим, щелкните вашу форму в области переходов правой кнопкой мыши и затем выберите **Режим макета** или **Конструктор** для первоначального отображения формы в другом режиме.

Для опробования созданной формы переведите ее в **Режим формы,** если она открыта в другом режиме. Просто щелкните правой кнопкой мыши заголовок вкладки и выберите **Режим формы.**

В Режиме формы можно выполнять те же задачи, что и на листе данных при работе с таблицей. В случае простой формы единственное отличие - отображение только одной записи в каждый момент времени.

Большинство пользователей считают формы интуитивно понятнее листа данных. В следующих разделах дается краткий обзор способов применения **Режима формы** для решения некоторых общих задач.

На профессиональном уровне.

Разные люди - разные формы

Часто для одной таблицы требуется создать несколько форм. В этом случае можно создавать формы с помощью специальных задач.

В головных управлениях компании Boutique Fudge один сотрудник занимается установкой цен. Этот человек (называемый корректировщиком) каждый день просматривает перечень товаров и изменяет цены на основе имеющегося запаса.

Для этого корректировщику цены нужны только три вида данных о каждом товаре: значения полей **ProductName, Price** и **UnitsInStock.** Для упрощения этого процесса можно создать форму, включающую лишь необходимые данные.

Для того чтобы сделать эту форму удобной в применении, можно добавить некоторые пока

еще незнакомые вам средства, которые описаны далее в этой главе. Вы можете запретить корректировку всех полей, кроме поля **Price,** для того чтобы избежать случайных изменений, включить несколько записей в форму для быстрой, с первого взгляда установки цены и отфильтровать перечень товаров, исключив снятые с производства. Если вы хотите произвести сильное впечатление на приверженцев программы Access, можно включить макрокод, описанный с *части V,* для создания кнопок, автоматически выполняющих задачу (например, повышение цены на 10%).

От вас зависит количество создаваемых форм. Некоторые пользователи пытаются создать как можно меньше форм и сделать их достаточно гибкими, способными решать разнообразные задачи. Другие создают десятки специализированных форм, экономящих время. В большой компании, такой как Boutique Fudge, каждый отдел (продаж, доставки, обслуживания клиентов и т. д.), вероятно, будет использовать форму, сделанную своими руками. Любая форма заставляет сотрудников делать именно то, что нужно (и не позволяет делать то, чего не следует).

12.1.2.1. *Поиск и редактирование записи*

Редко можно найти запись, которая никогда не меняется. В зависимости от типа хранимых данных большая часть работы в **Режиме формы** может состоять из поиска конкретной записи и внесения изменений. Возможно, вам нужно изменять цену товара в соответствии с заданным инкрементом, откорректировать адресные данные странствующего клиента или переопределить расписание для класса.

Прежде чем вы сможете выполнить любое из этих изменений, нужно найти соответствующую запись. В **Режиме формы** есть четыре способа получения нужной записи. В первых трех применяются кнопки перехода между записями, отображаемые в нижней части окна формы.

- *Применение кнопок перехода.* Если у вас относительно небольшая таблица, самый быстрый способ перехода - щелчок кнопкой мыши по кнопке нужного направления для перехода от одной записи к другой. *В разд. "Перемещение в таблице" главы 3* дано подробное описание использования кнопок.

■*Задание местоположения.* Если точно известно, где расположена запись, можно ввести ее номер, определяющий местоположение записи (например, 100 для сотой записи), и затем нажать клавишу <Enter>. Если вы не попадете точно на нужную запись, можно воспользоваться кнопками перехода для перемещения на запись, расположенную поблизости.

■*Поиск.* Средство быстрого поиска находит запись с конкретным фрагментом текста (или числовым значением) в одном из ее полей. Для применения быстрого поиска введите искомый текст в область поиска, как показано на рис. 12.6. Если вы хотите проверять конкретное поле или воспользоваться дополнительными параметрами, используйте команду на ленте **Главная → Найти → Найти** (Home → Find → Find).

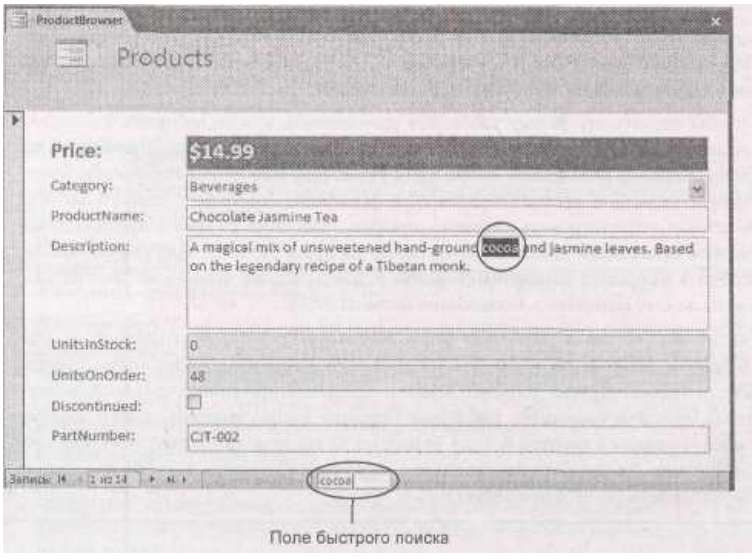


Рис. 12.6. Когда используется поле быстрого поиска, не нужно нажимать клавишу <Enter>. Программа Access ищет следующее совпадение по мере ввода

■ *Фильтрация.* С помощью фильтрации или отбора вы можете свести множество отображаемых записей к небольшому набору. Главный секрет отбора состоит в возможности применения средства, названного Фильтр по форме (filter by form), для быстрого обнаружения единственной записи. Вы узнаете, как оно действует, в *разд. "Применение фильтра по форме" далее в этой главе.*

После того как запись, которую вы хотите изменить, найдена, ее можно редактировать точно так же, как на листе данных. Если внесено изменение, нарушающее условие на значение (например, ввод текста "Exasperated Bananas" в поле, хранящее даты), вы получите аналогичные сообщения об ошибках.

Программа Access вносит любое сделанное вами изменение, как только вы переходите к другой записи или в другое поле. Для отказа от изменения нажмите клавишу <Esc> до перехода. После этого исходное значение снова выводится в ячейке, и Access отбрасывает ваши изменения. Если же вы случайно подтвердили внесение изменения, можно воспользоваться кнопкой **Отменить** на **Панели быстрого доступа** (над лентой) или нажать комбинацию клавиш <Ctrl>+<Z>, чтобы аннулировать изменение.

12.1.2.2. *Добавление записи*

Как вы уже знаете, можно добавить новую запись на лист данных, если перейти в самый конец таблицы и вводить данные в строке, следующей за последней записью. В Режиме формы принцип тот же - перейдите в самый конец таблицы, за последнюю запись.

Вы поймете, что достигли желанной точки, готовой к вставке новой записи, когда все поля вашей формы окажутся пустыми (рис. 12.7). Для того чтобы обойтись без прокрутки, воспользуйтесь кнопкой **Новая (пустая) запись** (New Record) в нижней части формы (помеченной на рис. 12.7).

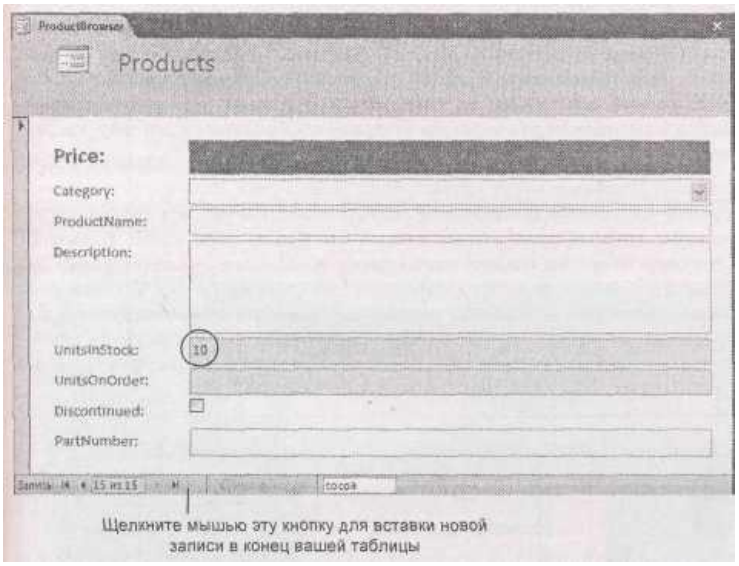


Рис. 12.7. Когда создается новая запись, вы начинаете с чистого листа, на котором отображается форматирование, но не значения. Если в таблице заданы какие-либо значения по умолчанию, вы увидите вместо пустых полей эти значения. В таблице Products у поля Units In Stock значение по умолчанию 10

Если, в конце концов, вы решили не добавлять новую запись, дважды нажмите клавишу <Esc>. При первом нажатии клавиши <Esc> программа Access стирает значение в текущем поле. При втором нажатии Access удаляет все остальные введенные значения. Теперь, когда форма очищена до исходного состояния, можно переходить к другой записи.

Если вы перешли к другой записи из новой, пока в ней оставались некоторые данные, программа Access создает новую запись и добавляет ее в таблицу. Это действие невозможно отменить. Если нужно избавиться от вновь созданной записи, следует удалить ее, как описано в следующем разделе.

Малоизвестная или, недооцененная возможность.

Вывод на экран изображений из БД

Как вы узнали в *главе 2,* можно хранить файл изображения как часть записи с помощью типа данных **Вложение.**

В формах вложения элегантно обрабатываются с помощью элемента управления **Вложение.** У элемента управления **Вложение** есть одно ценное качество - он отображает графическое содержимое непосредственно на форме.

Рассмотрим, как функционирует этот элемент управления. Если поле типа Вложение хранит изображение, это изображение появляется в поле элемента управления **Вложение,** так что можно любоваться им непосредственно на форме. Подобное поведение - существенное улучшение по сравнению с листом данных, который вынуждает для проверки открывать файл изображения в другой программе. Если поле **Вложение** хранит несколько изображений, еще важнее то, что можно использовать стрелки на удобной раскрывающейся мини-панели для перехода от одного изображения к другому, как показано на рис. 12.8.

Как вы знаете, поля **Вложение** могут хранить файл любого типа. Если хранится не изображение, элемент управления **Вложение** не столь полезен. Вы увидите только пиктограмму программы, которой принадлежит данный тип файла. Если в поле **Вложение** содержится документ Word, вы увидите пиктограмму Word. Если в поле текстовый документ, отображается пиктограмма программы Блокнот (Notepad) и т. д. Если в ваших полях **Вложение** содержатся не изображения, можно также изменить размер поля для элемента управления **Вложение,** чтобы оно было достаточным для отображения пиктограммы типа файла. Нет смысла делать его больше, поскольку остальное пространство просто теряется.

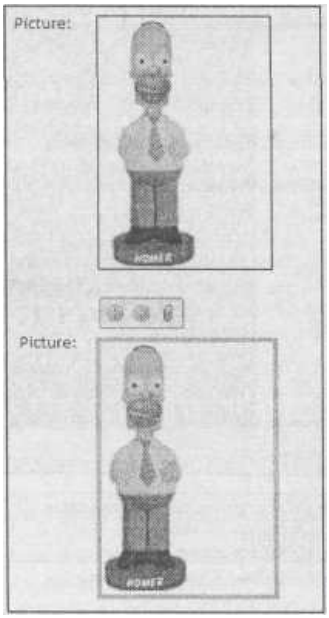


Рис. 12.8. *Вверху:* поле **Рисунок** отображает куклу-болванчика. Программа Access подбирает размер рисунка, соответствующий полю элемента **Вложение** (без неестественного растяжения или перекашивания рисунка). *Внизу:* когда выбирается поле **Рисунок,** на экране над рисунком появляется мини-панель с дополнительными элементами. Кнопки-стрелки позволяют просмотреть все вложенные файлы, относящиеся к данной записи. Пиктограмма скрепки открывает окно **Вложения,** в котором можно добавить или удалить вложения или открыть их в другой программе

12.1.2.3. *Удаление записи*

Если вы обнаруживаете запись, которой не должно быть, ее можно стереть за секунду. Самый легкий способ удаления текущей - выбрать на ленте **Главная → Записи → Удалить** (Home → Records → Delete). Но есть и другая возможность. Можно выбрать всю запись, щелкнув кнопкой мыши левое боковое поле в окне формы. После этого можно удалить ее, нажав клавишу <Delete>.

Независимо от того, какой способ вы выберите, программа Access попросит подтверждения, прежде чем удалит запись. Удаленные записи вернуть нельзя, поэтому нажимайте клавишу осторожно.

12.1.2.4. *Печать записей*

Существует малоизвестный секрет, связанный с формами. Его можно применять для создания быстрой распечатки.

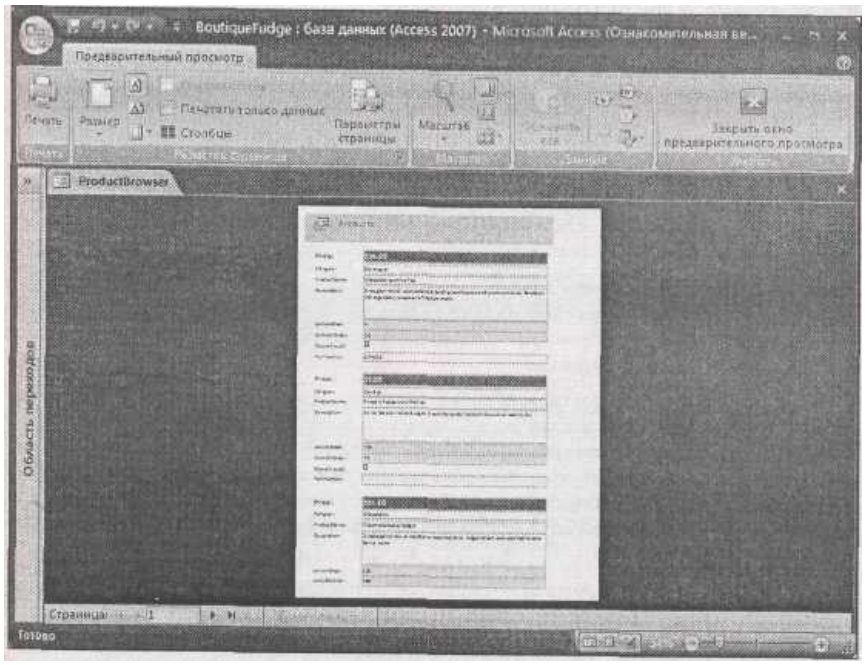


Рис. 12.9. В этом окне предварительного просмотра показано, что вы получите, если напечатаете форму со списком клиентов CustomerList. Распечатка соответствует форме, у нее то же форматирование и макет. Когда Access впервые создает форму, ей задается ширина обычного листа бумаги. Когда вы печатаете форму, программа размещает на каждой странице столько записей - в данном случае три - сколько она может вместить

Для этого откройте форму и затем выберите: кнопка **Office → Печать.** На экране появляется знакомое диалоговое окно **Печать,** в котором можно выбрать принтер и нужное число копий.

Когда вы печатаете форму, программа Access печатает все записи, одну за другой. Если вы хотите напечатать только текущую запись, в диалоговом окне **Печать** выберите переключатель **выделенные записи** (Selected Records), прежде чем щелкнуть мышью кнопку **ОК.**

Для того чтобы просмотреть результат перед выводом на печать, можно выбрать кнопку **Office → Печать → Предварительный просмотр** (рис. 12.9). Для того чтобы вернуться в окно формы, щелкните кнопкой мыши **Предварительный просмотр → Закрыть → Закрыть окно предварительного просмотра.**

Несмотря на то, что может возникнуть искушение использовать формы как удобный способ создания привлекательных распечаток, применение отчетов предоставляет больше возможностей и лучший контроль.

12.2. *Сортировка и фильтрация в формах*

Сортировка и фильтрация - две незаменимые функциональные возможности, предоставляемые программой Access в **Режиме формы.** Освоить их на самом деле легко - вы уже узнали, все что нужно, когда подробно знакомились с листом данных в *главе 3.* Создатели Access хорошенько позаботились о том, чтобы фильтрация и сортировка действовали в формах так же, как на листе данных. Вы применяете одни и те же команды в одной и той же части ленты для запуска этих средств обработки.

12.2.1. *Сортировка в форме*

Как вы уже, возможно, поняли, данные в формах отображаются в первоначальном неотсортированном порядке. Иными словами, записи выводятся в том порядке, в каком вы их создавали. (Единственное исключение - создание формы, получающей данные из запроса, в котором применялась сортировка.)

К счастью, выполнить сортировку легко. На самом деле отсортировать записи, отображаемые в

форме, можно точно так же, как записи на листе данных. Выберите поле, которое хотите использовать для сортировки, щелкните его правой кнопкой мыши и выберите один из вариантов сортировки. В текстовом поле вы увидите варианты **Сортировка от А до Я** (в алфавитном порядке) и **Сортировка от Я до А** (в порядке, обратном алфавитному). Можно также использовать кнопки **по возрастанию** и **по убыванию** в группе ленты **Главная → Сортировка и фильтр.**

Дополнительную информацию о вариантах сортировки (включая сортировку по нескольким полям) *см. в разд. "Сортировка" главы 3.*

12.2.2. *Фильтрация в форме*

Фильтрация - средство, позволяющее ограничить общее число отображаемых записей только теми, которые вас интересуют. Средствами фильтрации можно отобрать активных клиентов, имеющиеся на складе товары, дорогостоящие заказы и другие группы записей в соответствии с заданными условиями отбора.

Формы предоставляют следующие варианты фильтрации или отбора.

■*Обычный фильтр* отображает список всех значений для конкретного поля и позволяет выбрать те, которые вы хотите скрыть. Его легко применять, но он может потребовать значительного времени. Если вы хотите скрыть числовые значения, попадающие в определенный диапазон, эту работу можно сделать гораздо быстрее с помощью фильтра по условию (filter by condition) (который будет описан чуть позже). Для отображения списка значений обычного фильтра перейдите к полю, которое хотите фильтровать, и щелкните кнопкой мыши **Главная → Сортировка и фильтр → Фильтр** (Home → Sort & Filter→ Filter). В *разд. "Обычный фильтр" главы 3* приведено подробное описание обычных фильтров.

■*Фильтр по выделенному* применяет фильтрацию, основанную на имеющемся значении. Сначала найдите значение в одной из записей, щелкните его правой кнопкой мыши и выберите вариант отбора. Можно щелкнуть правой кнопкой мыши значение цены $25 и выбрать вариант **Больше или равно $25** (Greater Than or Equal to 25) для того, чтобы скрыть дешевые товары. Более подробную информацию см. в *разд. "Фильтр по выделенному " главы 3.*

■*Фильтр по условию* позволяет задать точное условие для отбора записей. Оно необязательно должно базироваться на имеющемся значении. Для вставки этого варианта фильтрации щелкните правой кнопкой мыши поле и затем найдите подменю с вариантами отбора. Его название зависит от данных, так для текстовых полей в меню включен вариант **Текстовые фильтры,** для числовых полей - **Числовые фильтры** и т. д. Вы можете больше узнать об этом типе фильтра в *разд. "Расширенный фильтр" главы 3.*

- *Расширенные фильтры* - это фильтры, в которых условия отбора формируются с помощью окна, напоминающего окно **Конструктора** запросов. Преимущество расширенных фильтров состоит в возможности применения фильтрации сразу по нескольким полям. Для создания набора расширенных фильтров выберите **Главная → Сортировка и фильтр → Дополнительно → Расширенный фильтр** (Home → Sort & Filter → Advanced Filter Options → Advanced Filter/Sort).

Примечание

Если вы вставляете запись, не отвечающую установленным в данный момент условиям отбора, новая запись будет скрыта, как только вы ее добавите. Для того чтобы вернуть ее на экран, удалите условия фильтрации с помощью ленты: выберите вкладку Главная, щелкните мышью кнопку **Дополнительно** в группе **Сортировка и фильтр** и затем выберите команду **Очистить все фильтры** (Clear All Filters). Или воспользуйтесь кнопкой **Применить фильтр** (Toggle Filter) для временной приостановки действия фильтра (позже, для того чтобы возобновить действие фильтра, снова щелкните мышью кнопку **Применить фильтр).**

12.2.3. *Применение фильтра по форме*

Существует еще одно средство фильтрации, действующее в формах: фильтр по форме. По сути "фильтр по форме" преобразует форму в полнофункциональную форму поиска (search form). С помощью этой формы поиска вы задаете одно или несколько условий отбора. Затем вы применяете фильтр для отображения записи (или записей), соответствующей условиям отбора.

Несмотря на то, что фильтр по форме можно применять на листе данных, он проявляет себя во всей красе в формах. Фильтр по форме особенно удобен для поиска единственной трудно

обнаруживаемой записи. (Если вы хотите применить фильтр для извлечения группы записей, обычно легче воспользоваться любым другим вариантом фильтрации.) Далее описан способ применения фильтра но форме.

1. Выберите на ленте **Главная → Сортировка и фильтр → Дополнительно → Изменить фильтр.**

Программа Access переведет форму в режим поиска. В этом режиме форма остается прежней, но все ее поля становятся пустыми.

Если средство "фильтр по форме" уже использовалось, и вы вернулись к нему для того, чтобы изменить параметры фильтрации, следует начать с удаления набора установленных прежде фильтров. Для этого щелкните правой кнопкой мыши пустое место в области формы и выберите команду **Очистить бланк** (Clear Grid).

2. Перейдите в поле, которое хотите использовать для фильтрации.

В нем появится стрелка, признак раскрывающегося списка.

3. Щелкните кнопкой мыши стрелку раскрывающегося списка и выберите значение, которое хотите включить в результаты.

В раскрывающемся списке отображаются все значения из разных записей таблицы (рис. 12.10). Когда вы выберите одно значение, оно появится в поле в кавычках.

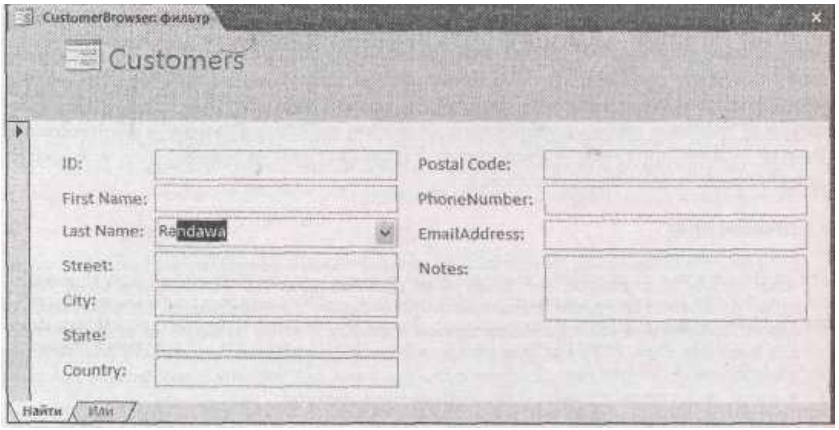


Рис. 12.10. На рисунке представлена форма **Customers** в режиме фильтра по форме. С помощью раскрывающегося списка вы можете быстро найти клиента по фамилии или найти имя, набрав несколько первых букв, вместо прокрутки списка, как здесь показано. В данном примере набор букв "Ra" приводит к выводу первого буквенного совпадения: фамилии Randawa

4. Если хотите применить фильтр к нескольким полям, вернитесь к пункту 2.

Задайте несколько условий отбора, если одно условие включает в результат больше совпадений, чем вам хотелось бы. Если вы не помните фамилию клиента, можно применить фильтр к полю **FirstName.** Но если у клиента распространенное имя, может быть, придется применить фильтр к еще одному полю, например City.

Если не нужны точные совпадения, можно создать более сложные фильтры с помощью выражения. Задайте <10 для поиска числовых значений, меньших 10, и Like Jon* для поиска текстовых значений, таких как "Jones", "Jonathon" и "Jonson". Фильтрация особенно эффективна в случае полей с датами. В *разд. "Построение условий отбора " главы 6* приводится много примеров условий отбора.

5. Если вы хотите выполнить несколько операций фильтрации и объединить результаты, щелкните кнопкой мыши вкладку Или и задайте дополнительные условия отбора (рис. 12.11).

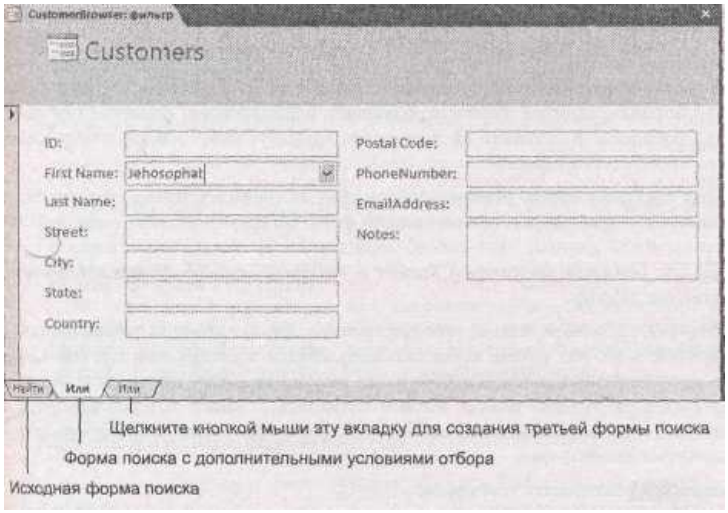


Рис. 12.11. Вкладка **Или** отображается в нижней части формы. Когда вы щелкаете вкладку **Или,** на экране появляется вторая копия формы поиска, в которую можно вставить дополнительные условия отбора. При каждом щелчке кнопкой мыши вкладки **Или** на экране появляется еще одна вкладка **Или.** Этот процесс можно повторить и заполнить десяток форм поиска за один раз, но для таких излишеств редко есть разумные основания

Если заполнить первую форму **Найти** так, что в ней ищутся записи с фамилией "Gorfinkel", и вторую форму поиска с именем "Jehosophat", в ваши результаты будут включены все записи с фамилией Gorfinkel и все записи с именем Jehosophat. Но если оба эти условия отбора поместить на одну форму поиска, будут отобраны только записи о людях с именем и фамилией Jehosophat Gorfinkel.

6. Щелкните правой кнопкой мыши пустое место на поверхности формы и выберите команду **Применить фильтр** (Apply Filter/Sort).

Программа Access переключается в нормальное представление формы и применяет условия фильтрации. В нижней части формы между кнопками переходов и полем поиска появится словосочетание **С фильтром** (Filtered), сообщающее о том, что отображаются не все записи.

Если вы решили не применять фильтр, закройте форму **Найти.** Программа Access перейдет к нормальному отображению формы и не будет применять никаких условий отбора.

Подсказка

Для удаления условий отбора, но сохранения их под рукой и применения в дальнейшем, выберите на ленте **Главная → Сортировка и фильтр → Применить фильтр**. Для повторного применения фильтра позже, щелкните мышью кнопку **Применить фильтр** еще раз. Access хранит самые последние заданные условия отбора вместе с формой, поэтому они всегда доступны.

12.2.4. *Сохранение фильтров для дальнейшего использования*

Одно из ограничений применения фильтров в формах заключается в том, что программа Access помнит только последний набор условий отбора. Если вы подготовили сложное выражение для фильтрации, которое хотите использовать в дальнейшем, подобное ограничение становится проблемой. Как только вы примените другое условие отбора, плоды вашей напряженной работы будут потеряны.

К счастью, эту проблему можно решить несколькими способами. С одной стороны, можно создать полностью новый запрос, выполняющий фильтрацию, и применить его при создании совершенно новой формы. Этот способ хорош, если вы хотите использовать ваши условия отбора для решения конкретной задачи и настроить способ функционирования формы и вывода в нее данных.

С другой стороны, если вы не планируете применять заданные условия отбора слишком часто, но хотите иметь их под рукой, когда они понадобятся в следующий раз (или если нужно хранить десятки разных условий отбора и при этом вы не хотите иметь десятки почти одинаковых форм), есть лучший выбор. Можно сохранить в вашей БД условия отбора фильтра как запрос. В

дальнейшем, когда вы захотите вернуться к ним, вы сможете загрузить их и применить в вашей форме.

Далее описано, как реализовать этот прием.

1. Примените ваши фильтры.

Используйте любой из методов, описанных в *разд. "Фильтрация в форме"ранее в этой главе.*

2. Выберите на ленте **Главная → Сортировка и фильтр** → Дополнительно → Расширенный **фильтр** (Home → Sort & Filter → Advanced → Advanced Filter/Sort).

Это действие открывает на экране окно запроса. Данный запрос использует тот же источник данных (таблицу или запрос), что и ваша форма, и применяет ваши условия отбора с помощью поля Условие отбора, расположенного под соответствующим полем. Вам не нужно вносить никакие изменения в окне запроса, поскольку программа Access автоматически заполняет поле (или поля) **Условие отбора** на основании текущих условий фильтрации.

3. Выберите на ленте **Главная → Сортировка и фильтр → Дополнительно → Сохранить как запрос** (Home → Sort & Filter → Advanced → Save as Query). Задайте имя запроса и нажмите кнопку **ОК.**

Несмотря на то, что этот запрос можно использовать как обычный, вряд ли вы захотите это делать. *Для* того чтобы не возникало путаницы, примените другой тип имени, например, **CustomerBrowser_FiIter** (фильтр обозревателя клиентов), ясно указывающий на то, что данный запрос применяется для фильтрации формы.

В следующий раз, когда захотите снова воспользоваться условиями отбора и применить их еще раз, откройте форму и выполните следующие действия.

1. Выберите на ленте **Главная → Сортировка и фильтр → Дополнительно → Расширенный фильтр.**

Этот шаг открывает окно запроса.

2. Выберите на ленте **Главная → Сортировка и фильтр → Дополнительно → Загрузить из запроса** (Home → Sort & Filter → Advanced → Load From Query).

Программа Access отобразит все запросы, использующие ту же таблицу и не содержащие операций объединения.

3. Выберите запрос с условиями фильтрации, созданный ранее, и щелкните мышью кнопку ОК.

В окне запроса появятся условия отбора из этого запроса.

4. Щелкните правой кнопкой мыши на свободном месте в окне запроса и выберите команду **Применить фильтр** (Apply Filter/Sort) для применения заданных условий отбора.

Подсказка

Этот прием можно использовать для применения одних и тех же условий отбора в разных формах, при наличии в них полей, по которым *вы* хотите фильтровать записи. (Можно воспользоваться условиями отбора, созданными для формы CustomerBrowser в другой форме, отображающей клиентов, но не в форме, которая выводит товары.)

12.3. *Создание улучшенных макетов*

Все формы, созданные вами до настоящего момента, были похожи друг на друга. Все поля находятся в одной или нескольких колонках с компактно расположенными данными. Во многих случаях это верстка очень хороша. Но иногда хочется разрешить своему творческому его вырваться наружу и начать действовать.

Вы сталкивались с подобной ситуацией в отчетах, описанных в *главе 11.* Как только вы освобождаете отчет от внутреннего табличного макета, можно создавать распечатки, более похожие на каталоги товаров розничной продажи, чем на однообразную таблицу данных. Этот же принцип действует и в формах - как только вы решите оставить незатейливый мир простых форм, вы сможете создавать гораздо более оригинальные формы. Можно применять пустое пространство для разделения плотных групп данных; создавать формы с графическим оформлением в виде изображений, линий и прямоугольников; размещать информацию более плотно или более свободно; проектировать формы, напоминающие бумажные документы и т. д.

12.3.1. *Высвобождение элементов управления из макета*

Как и в отчетах, в формах применяется полезное средство, именуемое *макетом.* Это форматируемый контейнер, который программа Access использует за кадром для компоновки группы элементов управления. Если в макете увеличить ширину одного поля, ширина остальных полей в нем также увеличится. Если передвинуть макет, все элементы управления переместятся вместе с ним. И если вы перекомпонуете макет, все элементы управления займут соответствующее пространство.

Примечание

Элементы управления - это компоненты, которые можно вставлять в отчеты и формы. К ним относятся, например, **Подписи, Рисунки** и **Поля.** Некоторые элементы управления применяются для отображения неизменного содержимого (например, заголовок формы), другие содержат меняющиеся данные (например, значения полей текущей записи).

Если вы хотите иметь возможность располагать элементы управления в точно заданной позиции, прежде всего их надо извлечь из. макета. (И, как и в отчетах, потребуется значительное время для размещения элементов управления вручную и проверки полученных результатов.)

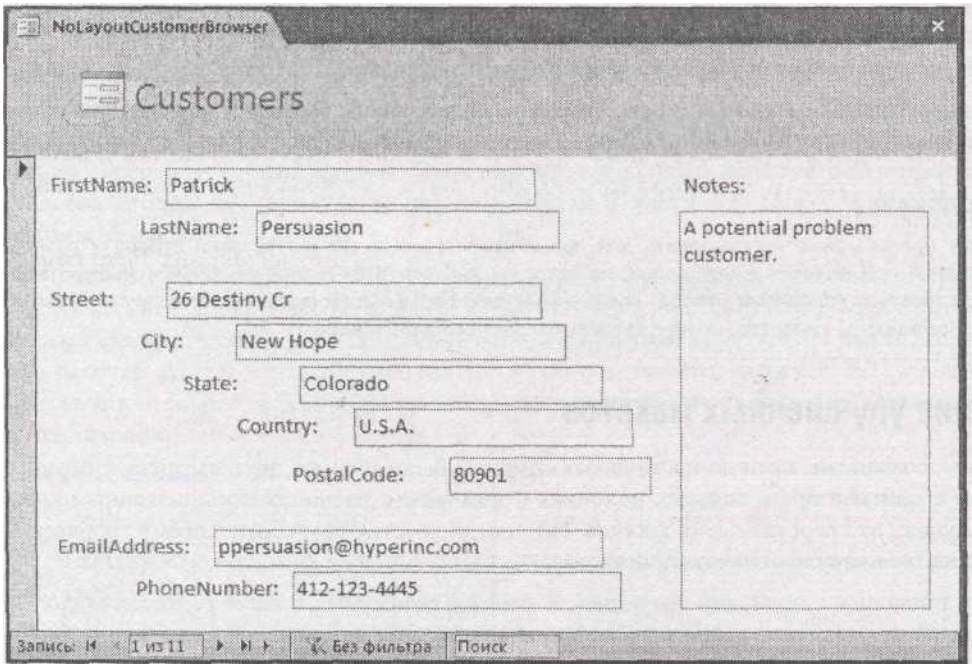


Рис. 12.12. Этого причудливого ступенчатого расположения нельзя было бы добиться в табличной структуре макета. Но за такой дизайн приходится платить (и это не только зрительное перенапряжение). Если вы когда-нибудь измените таблицу и потребуется корректировка формы, вы затратите много усилий на перекомпоновку полей, т. к. за кадром нет никакого макета, удерживающего их вместе

Прежде чем извлекать элемент управления из макета, убедитесь в том, что вы находитесь в **Режиме макета** или в **Конструкторе** (щелкните правой кнопкой мыши заголовок вкладки и выберите **Режим макета** или **Конструктор). На Режим макета** приятнее смотреть, но в **Конструкторе** легче перемещать поля. Когда вы с помощью мыши перетаскиваете в Конструкторе название поля (например, **Подпись,** содержащую слово "ProductName"), связанный с ней элемент управления, отображающий значение поля, также перемещается. В **Режиме макета** придется двигать каждый компонент отдельно, что удваивает объем работы. (Между прочим, в **Конструкторе** тоже можно перемещать **Подписи** и **Поля** отдельно, если знать, где щелкнуть кнопкой мыши. Объяснения *см. в разд. "Создание отчета без помощи мастера" главы 11.)*

Для реального перемещения поля щелкните правой кнопкой мыши поле, положение ко-

торого хотите изменить, и выберите **Макет →Удалить** (Layout → Remove). Затем перетащите поле в новое место. На рис. 12.12 показан пример формы, не применяющей макеты ни для одного из своих элементов управления.

12.3.2. *Применение нескольких макетов*

Как вы уже узнали в этой главе, программа Access располагает поля в нескольких колонках, если не может разместить их в одной (см. рис. 12.2). Когда формируется несколько колонок, каждая из них обладает собственным макетом.

Приятная новость - вы тоже можете создать несколько макетов. Такая необходимость может возникнуть в следующих ситуациях:

■ *размещение полей в смежных колонках* - это делается с помощью расположения одного макета следом за другим;

■ *размещение полей в нескольких разных группах* - эти поля могут выводиться в разных местах формы. Одна группа может располагаться вверху, а другая внизу, а между ними - другое содержимое;

■ *задание разной ширины полей* - у всех полей в макете одинаковая ширина. Если поместить поля в разные макеты, для них можно задать разную ширину.

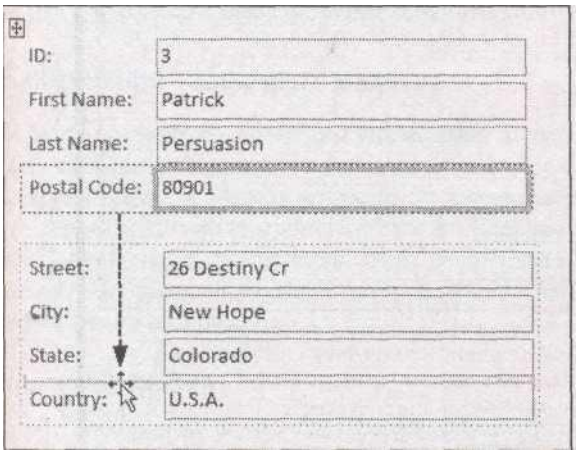


Рис. 12.13. Когда указатель мыши перемещается в области нового, только что созданного вами макета, на экране появляется желтая линия, обозначающая позицию поля после того, как вы отпустите кнопку мыши. Вы можете поместить поле в верхнюю часть макета, в нижнюю его часть или в любое место внутри макета. В данном примере поле **PostalCode** вставляется между полями **State** и **Country**

Для помещения поля в новый макет выполните следующие действия.

1. Убедитесь, что вы находитесь в **Режиме макета.**

Если нет, щелкните правой кнопкой мыши заголовок вкладки с формой и выберите **Режим макета.**

2. Щелкните правой кнопкой мыши первое поле, которое хотите удалить из макета, и затем выберите **Макет → Удалить.**

Если поле находится в середине таблицы макета, программа Access отодвигает его в сторону. Если поле расположено в конце таблицы макета, его положение не меняется.

3. Перетащите мышью поле в другую часть формы.

Если вы хотите расположить второй макет под первым, перетащите мышью поле в нижнюю часть формы на его новое место.

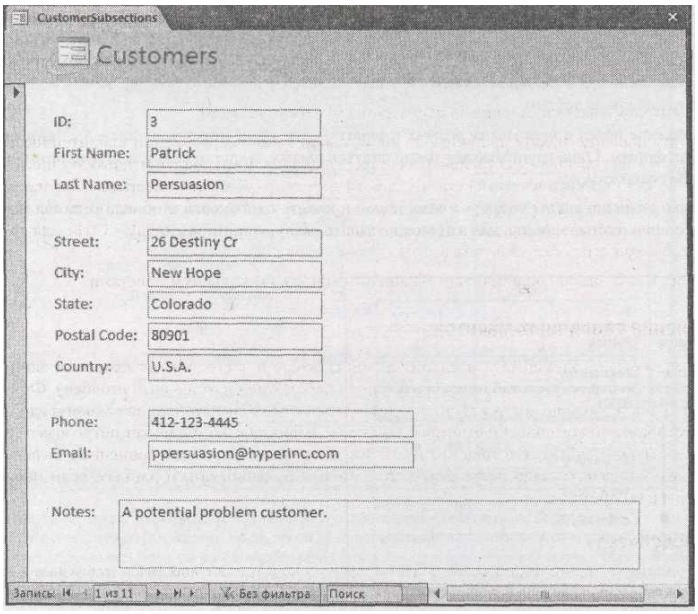


Рис. 12.14. В данной форме поля сгруппированы в четыре четких подраздела. Таким образом, создается более ясная и наглядная компоновка по сравнению с единым табличным макетом, но сложность создания формы без макетов возрастает, в особенности если позже придется добавлять дополнительные поля или менять порядок расположения полей на форме

4. Щелкните правой кнопкой мыши удаленное из макета поле и выберите команду **Макет → В столбик** (Layout → Stacked). Этим действием создается новый макет для поля, которое вы освободили в пункте 2.

Все макеты, которые вы видели до сих пор, - макеты в столбик, располагающие поля одно над другим. Пример табличного макета (поля располагаются бок о бок) вы увидите в следующем разделе.

5. Найдите поле, которое хотите перенести в новый макет, и перетащите его туда мышью (рис. 12.13).

Перетаскивание поля из одного макета в другой - рациональный способ. Он действует быстрее, чем удаление всех нужных полей из первого макета, а затем добавление их во второй макет.

6. Повторите пункт 5 для каждого поля, которое хотите перенести в новый макет.

Перенос поля из одного макета в другой может быть непростым занятием. Если все идет не так, как следует, и поле располагается в неверном месте, воспользуйтесь командой **Отменить** для исправления ситуации. (Команда **Отменить** запускается нажатием комбинации клавиш <CtrI>+<Z> или щелчком мышью кнопки **Отменить** на **Панели быстрого доступа,** находящейся над лентой.)

7. Когда второй макет скомпонован, перенесите его в нужное место.

Для перемещения макета щелкните кнопкой мыши одно из его полей и найдите пиктограмму из перекрещенных стрелок в левом верхнем углу. Перетащите мышью эту пиктограмму для переноса всего макета. Будьте внимательны - если поместить один макет слишком близко к другому, программа Access решит, что вы хотите объединить оба макета в один. (Можно во время перетаскивания держать нажатой клавишу <Ctrl> для того, чтобы запретить Access делать это.)

На рис. 12.14 показан окончательный вариант формы с несколькими макетами.

12.3.3. *Применение табличных макетов*

Макеты организуют элементы управления двумя способами: в столбик (каждое поле занимает отдельную строку) и в таблицу (каждое поле помещается в отдельный столбец). Формам, как правило, больше подходит организация полей в столбик, а табличные макеты удобнее для компактной компоновки отчетов. Тем не менее иногда в форме может потребоваться табличный

макет. Чаще всего такая необходимость возникает при одновременном выводе нескольких записей; гораздо легче выводить на экран больший объем данных, если поля заключены в столбцы.

Подсказка

Несмотря на то, что форма может содержать несколько макетов обоих типов (табличные и в столбик), они редко хорошо сочетаются друг с другом. Лучше выбрать один тип и придерживаться его.

Для замены обычной формы (с макетом в столбик) на форму с табличным макетом выполните следующие действия.

1. Убедитесь в том, что вы находитесь в **Режиме макета.**

Если нет, щелкните правой кнопкой мыши заголовок вкладки и выберите **Режим макета.**

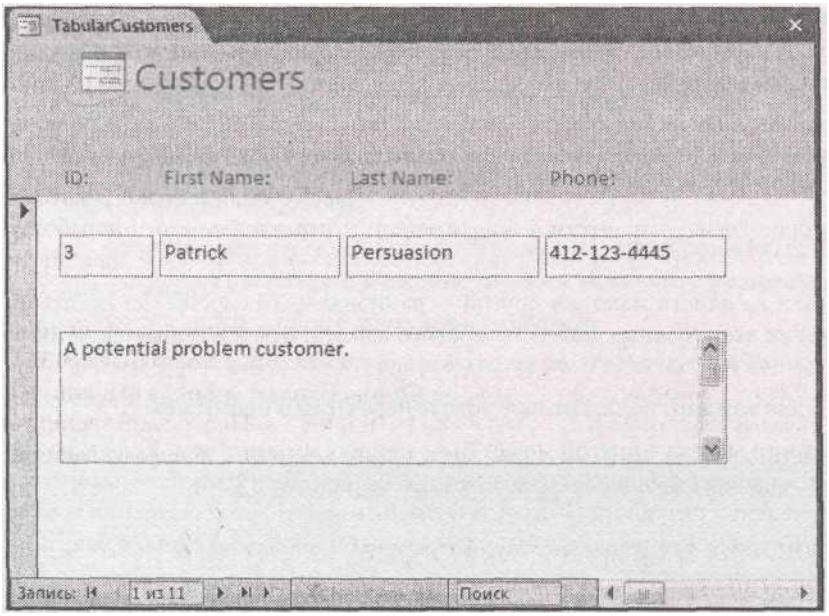


Рис. 12.15. На этом табличном макете отображены четыре столбца данных. У формы один элемент управления вне макета: поле примечаний **Notes,** которое разработчик формы переместил вниз, чтобы обеспечить ему достаточное пространство

Рис. 12.16. Обычно табличные макеты связаны с формами, отображающими много записей одновременно, как форма, показанная на рисунке

2. Выделите все поля формы, удерживая нажатой клавишу <Shift> и щелкая один раз кнопкой мыши каждое поле одно за другим.

Для экономии времени найдите на экране пиктограмму из перекрещенных стрелок в правом верхнем углу макета, которая появляется, как только вы выделите какой-либо компонент макета. Вы можете одним щелчком по этой пиктограмме выделить целиком весь макет.

3. Щелкните правой кнопкой мыши ваше выделение и выберите **Макет → Табличный.**

При создании табличного макета Access помещает имя каждого поля в область заголовков, а значение поля - ниже, как показано на рис. 12.15. Придется немного поработать мышью для того, чтобы добиться нужного порядка расположения полей и задания им подходящих размеров.

Этот процесс довольно утомителен. К счастью, существует ускоряющий прием. Если вы знаете, что будете использовать табличный макет, можно создать форму с самого начала. Вместо выбора последовательности **Создание → Формы → Форма** (Create → Forms → Form) для создания вашей формы выберите на ленте **Создание → Формы → Несколько элементов** (Create → Forms → Multiple Items). Этот шаг приведет к созданию формы, использующей табличный макет и отображающей несколько записей одновременно (рис. 12.16).

12.3.4. *Отображение нескольких записей в любой форме*

Вы можете выводить несколько записей в форме даже без применения табличного макета. На самом деле, пока ваша форма достаточно компактна, это сделать очень легко. Далее описаны необходимые действия.

1. Организуйте вашу форму так, чтобы она была максимально компактна.

При отображении набора записей они располагаются друг над другом, как показано на рис. 12.17. Таким образом, чем меньше ваша форма, тем большее число записей вы сможете увидеть одновременно. С другой стороны, не важно, насколько широка или узка ваша форма (до тех пор, пока все необходимое помещается на экране).

2. Перейдите в **Конструктор,** если еще не находитесь в этом режиме.

Как всегда, переключиться в режим **Конструктора** можно щелчком правой кнопки мыши заголовка вкладки и выбором команды **Конструктор.**

3. Измените размер формы, устранив пустое пространство, как показано на рис. 12.18.

Изменяя расположение элементов управления, вы освобождаете место в нижней части формы. Исключить освободившееся пространство за счет сжатия всей формы - это целиком ваша обязанность. Если при попытке сжать форму, она упрямо сохраняет прежний размер, вероятно, есть элемент управления, занимающий часть этого пространства. Необходимо сначала уменьшить его, а затем всю форму.

Примечание

Для того чтобы одновременно отобразить несколько записей, окно формы должно быть больше самой формы.

4. Если на экране нет **Окна свойств,** выберите на ленте **Инструменты конструктора форм | Конструктор → Сервис → Страница свойств** (Form Design Tools | Design → Tools → Property Sheet).

Как вы узнали в *главе 11,* **Окно свойств** позволяет настроить параметры элементов управления и других объектов. В данном случае параметра, который нужно изменить, нет на ленте. Он спрятан в **Окне свойств.**

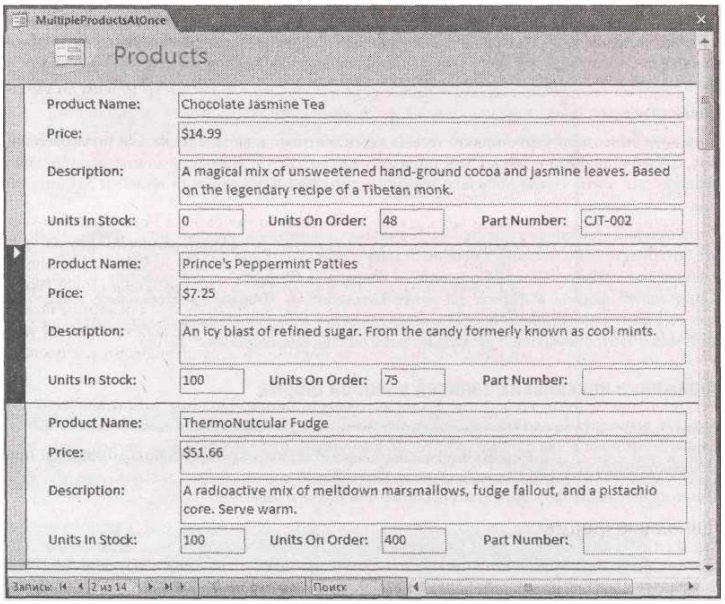


Рис. 12.17. Вы можете видеть три товара на одном экране. (Обратите внимание на то, что нижние три поля удалены из использующегося для трех верхних полей макета в столбик для того, чтобы расположить их более плотно относительно друг друга.) Стрелка в боковом поле указывает на то, что вторая запись - текущая. Для просмотра дополнительных записей можно воспользоваться кнопками перехода в нижней части формы или полосой прокрутки, расположенной справа

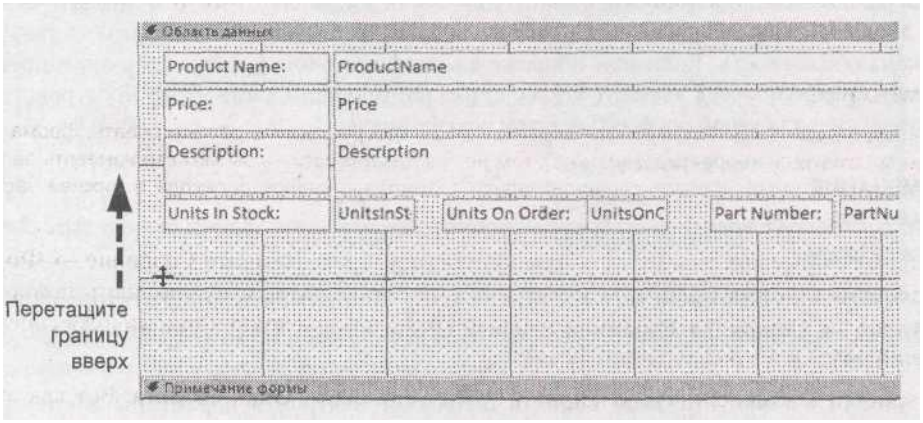


Рис. 12.18. Для сжатия формы перетащите вверх с помощью мыши нижнюю границу так, чтобы она располагалась на уровне нижней границы самого нижнего элемента управления

5. В списке **Окна свойств** выберите **Форма** (Form).

Это действие приводит к выводу на экран параметров, применяемых к форме в целом, а не к отдельному элементу управления.

6. Щелкните кнопкой мыши вкладку **Макет** (Format) и найдите параметр **Режим** по **умолчанию** (Default View).

Параметр **Режим по умолчанию** отображается в верхней части списка. Он позволяет задать режим первоначального представления при открытии формы.

7. Выберите *Ленточные формы* (Continuous Form).

Наиболее часто используемые режимы - *Одиночная форма* (Single Form) (отображает одну запись данных) и *Ленточные формы* (отображает несколько записей, одну за другой). Можно также выбрать иное представление, например *Режим таблицы* (Datasheet) (однообразная таблица, как те, что вы изучали в *главе 3), Сводная таблица* (PivotTable) или *Сводная диаграмма* (PivotChart). Наконец, можно воспользоваться режимом *Разделенная форма* (Split Form) для применения представления, которое сочетает режим таблицы с настроенной вами формой. Вы узнаете больше об этом варианте в следующем разделе.

8. При желании задайте в свойстве **Разделительные линии** (Dividing Lines) значение *Да для* отображения тонкой горизонтальной линии между записями.

Теперь при переходе в **Режим формы** вы увидите одновременно несколько записей при условии, что они могут поместиться в окне программы.

12.3.5. *Разделенные формы*

У режимов представления одиночной записи и множественных записей есть свои достоинства. В режиме отображения одиночной записи у вас много места для просмотра записи и вас не отвлекает обилие информации на экране. В режиме одновременного представления нескольких записей вы можете сравнить текущую запись с соседними записями.

В программе Access есть тип формы, который позволяет воспользоваться преимуществами обоих режимов представления сразу, - *разделенные формы.* Этот тип в одной форме сочетает оба представления данных. Идея заключается в том, что вы можете использовать таблицу для просмотра всех записей и форму для просмотра или редактирования одной записи. На рис. 12.19 показан пример.

Примечание

Обычно таблица применяется для перехода к записи, которую хотите редактировать, форма - для ее просмотра и корректировки, но в этом нет необходимости - вы можете изменять записи в таблице и переходить от записи к записи с помощью кнопок перехода в нижней части формы.

Разделенную форму создать легко - нужно просто выбрать на ленте **Создание → Формы → Разделенная форма** (Create → Forms → Split Form). Но вам нужно знать немного больше, если вы хотите превратить имеющуюся форму в разделенную или изменить способ представления разделов в разделенной форме.

Секрет кроется в изменении параметров формы с помощью **Окна свойств.** Вот как это делается.

1. Переключите форму в **Конструктор.**
2. Если на экране нет **Окна свойств,** отобразите его, выбрав **Инструменты конструктора форм | Конструктор → Сервис → Страница свойств** (Form Design Tools | Design → Tools → Property Sheet).
3. В раскрывающемся списке **Окна свойств** выберите **Форма** (Form).
4. Выберите вкладку **Макет** (Format), включающую все параметры, относящиеся к разделенным формам.
5. Найдите параметр **Режим по умолчанию** (Default Value) и задайте ему значение *Разделенная форма* (Split Form). Вы получаете окно, разделенное на две части, показанное на рис. 12.19.

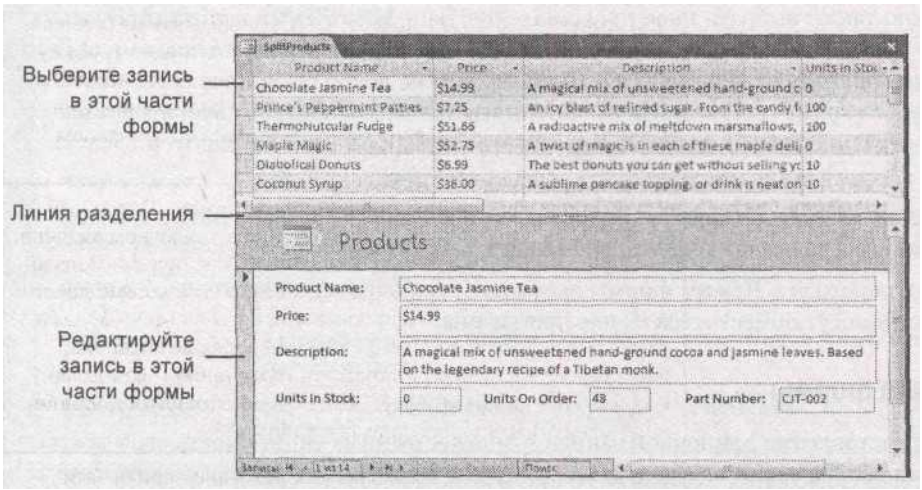


Рис. 12.19. В наиболее часто применяемом типе разделенной формы табличный раздел помещается сверху. Выбранная в данный момент запись отображается под таблицей. Размер каждого раздела можно изменить, перетащив мышью линию разделения, находящуюся между ними

Несколько дополнительных параметров позволяют управлять работой разделенных форм. В табл. 12.1 приведены подробности.

Таблица 12.1. Свойства разделенных форм

Свойство	*Описание*
Ориентация разделенной формы (Split Form Orientation)	С помощью этого параметра можно поместить раздел с таблицей в верхнюю часть окна (стандартный вариант), в нижнюю, слева или справа
Размер разделенной формы (Split Form Size)	Задает размер раздела таблицы в окне формы. Придется по-экспериментировать с разными числами, чтобы подобрать подходящий. Большинство пользователей предпочитают задавать размер раздела разделенной формы вручную в **Режиме формы**
Линия разделения в **разделенной форме** (Split Form Splitter Bar)	Если задать этому параметру значение *Нет,* исчезнет разделительная полоска между разделами окна. Вы (или пользователь, работающий с формой) не сможете изменять разделы окна, перемещая линию разделения мышью. Вместо этого вы будете жестко привязаны к размеру, заданному в параметре Размер разделенной формы
Сохранение положения линии разделения (Save Splitter Bar Position)	Если этому параметру задать значение Да, каждый раз, когда вы перемещаете линию разделения, текущий размер раздела с таблицей будет сохраняться в параметре Размер разделенной формы. При следующем открытии формы линия разделения окажется в позиции, соответствующей последнему изменению. Если задать этому параметру значение *Нет,* Access не будет сохранять изменение позиции линии разделения. Она вернется в исходную позицию в соответствии со значением параметра Размер разделенной формы
Таблица разделенной формы (Split Form Datasheet)	Измените значение параметра на *Только чтение* (Read Only), если хотите запретить изменение данных в табличном разделе окна формы. (Таблицу при этом все же можно будет использовать для перехода от одной записи к другой.) Подобное действие позволяет избежать ошибок, вызванных случайным нажатием клавиш. Если вы хотите запретить и редактирование, используйте параметры Разрешить изменение (Allow Edits), Разрешить удаление (Allow Deletions) и Разрешить добавление (Allow Additions), описанные в табл. 12.2
Печать разделенной формы (Split Form Printing)	Сообщите программе Access об использовании при печати только табличного представления (Только *таблица)* или только представления формы *(Только форма).* Стандартное значение *Только форма* означает организацию вашей информации в распечатке в соответствии с макетом вашей формы

12.3.6. *Еще более полезные свойства формы*

К настоящему моменту вы пользовались **Окном свойств** для изменения режима формы, позволяя обычной форме отображать несколько записей или используя разделенное представление. Но в **Окне свойств** хранится еще множество параметров. Одни полезны; другими вы почти никогда не будете пользоваться. В табл. 12.2 перечислено еще несколько параметров, которые могут оказаться кстати.

Таблица 12.2. Полезные свойства формы

Свойство	*Вкладка*	*Описание*
Источник записей (Record Source)	Данные (Data)	Откуда поступили данные. В этом свойстве обычно указано имя таблицы или запроса в БД. Но если вы любите технические приемы, в этом поле можно непосредственно набрать новую команду SQL (см. разд. "Режим SQL" главы 6)
Фильтр	Данные (Data)	Условие отбора, ограничивающее результирующий набор записей.

(Filter)		Это поле можно задать вручную или построить выражение с помощью ленты, как описано в разд. "Фильтрация в форме" ранее в этой главе
Фильтр при загрузке (Filter On Load)	Данные (Data)	Если задано значение Да, условие отбора применяется во время открытия формы. Если значение Нет, условие отбора сохраняется, но не применяется до тех пор, пока вы не выберите на ленте Главная → Сортировка и фильтр → Применить фильтр (Home → Sort & Filter → Apply Filter)
Порядок сортировки (Order By)	Данные (Data)	Порядок сортировки, используемый для упорядочивания результатов. Это поле можно задать вручную или установить порядок сортировки с помощью ленты, как описано в разд. "Сортировка в форме" ранее в этой главе
Сортировка при загрузке (Order By On Load)	Данные (Data)	Если задано значение Да, Access применяет заданную сортировку при открытии формы. Если - Нет, порядок сортировки запоминается, но не применяется. Этот вариант неособенно полезен - до тех пор, пока вы не откроете Окно свойств снова и не зададите значение Да в данном свойстве, сохраняемый порядок сортировки никогда не будет реализован
Применение фильтров (Allow Filters)	Данные (Data)	Если задано Нет, вы не сможете применить ни одну команду фильтрации из описанных в этой главе. Вы всегда будете видеть все записи
Подпись (Caption)	Макет (Format)	Текст, появляющийся в заголовке вкладки (или заголовок окна, если вы используете перекрывающиеся окна вместо документов со вкладками). Если это поле оставить пустым, программа Access использует имя формы как заголовок
Разрешить режим... (Allow ... View)	Макет (Format)	Эти параметры позволяют отключить определенный режим представления. Например, если задать Нет в параметре Разрешить режим макета (Allow Layout View), в меню исчезнет вариант для переключения формы в Режим макета
Разрешить изменение (Allow Edits)	Данные (Data)	Если задать Нет, вы не сможете корректировать данные в форме. Но сможете добавить новую запись с полностью новыми данными. Стандартное значение -Да
Разрешить удаление (Allow Deletions)	Данные (Data)	Если задать Нет, вы не сможете в этой форме удалять никакие записи. Стандартное значение -Да
Разрешить добавление (Allow Additions)	Данные (Data)	Если задать Нет, вы не сможете вставить новую запись в эту форму. Стандартное значение -Да
Ввод данных (Data Entry)	Данные (Data)	Если задать Да, эту форму можно будет использовать для вставки новой записи. Когда вы перейдете в Режим формы, то не увидите существующие записи. Вместо этого на экране будет чистый бланк формы, в который можно ввести новую запись. Когда вы добавляете записи, они остаются видимыми, по крайней мере, до тех пор, пока вы не закрыли форму и не открыли ее снова
Область выделения (Record Selectors)	**Макет** (Format)	Если задать *Нет, в* форму не включается левое поле. Это поле играет две роли. Во-первых, оно отображает стрелку рядом с текущей записью (что полезно в формах, отображающих несколько записей одновременно). Во-вторых, если щелкнуть поле кнопкой мыши, можно выделить целиком всю запись (после чего ее можно быстро удалить нажатием клавиши <Delete>)
Кнопки перехода (Navigation	**Макет** (Format)	Если задать *Нет, в* форму не включаются удобные элементы управления для переходов, расположенные в нижней части формы и позволяющие переходить от одной записи к другой. Вероятнее

Buttons)		всего, вы воспользуетесь этим вариантом при создании формы с абсолютно иным внешним видом, не терпящим никаких фирменных элементов Access, или при создании собственных кнопок перехода, использующих код VBA

Примечание

Многие свойства формы применяются только в редких случаях, когда используются свободно плавающие окна. Вы можете выбрать задание автоматической центровки окна **(Выравнивание по центру** (Auto Center)), формирование границы окна **(Тип границы** (Border Style)), наличие или отсутствие кнопок открытия во весь экран и сворачивания на панель задач **(Кнопки размеров окна** (Min Max Buttons)) и т. д. Эти свойства не окажут заметного влияния, если ваша БД использует более стандартные окна с вкладками.

На профессиональном уровне.

Семейство форм Access

Формы Access стараются удовлетворить любые потребности. Если вы спешите, можно создать готовую форму с базовым макетом и добавить лишь несколько штрихов форматирования. Если же вы чувствуете приближение творческого порыва, можно извлечь все поля из стандартных макетов и поместить их где угодно. Иначе говоря, формы - это гибкие объекты, предоставляющие страдающим от нехватки времени деловым людям удобства, в которых нуждаются, а серьезным художникам - творческий контроль, который им необходим.

Рассмотрим все варианты форм.

- *Простая форма* отображает единственную запись в базовом макете в столбик. Для создания простой формы выберите **Создание → Формы → Форма** (Create → Forms → Form).
- *Форма без макета* позволяет помещать элементы управления в любое место формы. Вы решаете отображать одновременно одну запись или несколько. При создании такой формы приходится делать всю работу самому. Начать можно с выбора **Создание → Формы → Пустая форма** (Create → Forms → Form Design), которая открывается в **Режиме макета.**
- *Табличная форма* отображает записи в табличном макете. Обычно в таких формах одновременно отображается несколько записей (что увеличивает сходство с таблицей). Для быстрого создания подобного детища выберите **Создание → Формы → Несколько элементов** (Create → Forms → Multiple Items).
- *Форма со сводной диаграммой* или *сводной таблицей* - это форма, единственная задача которой - вывод на экран сводной диаграммы или сводной таблицы *(см. главу 9).* Создавать такие формы можно с помощью последовательности **Создание → Формы → Сводная диаграмма** (Create → Forms → PivotChart) и **Создание → Формы → Другие формы → Сводная таблица** (Create → Forms → More Forms → PivotTable). В *разд. "Сводные таблицы" главы 9* приведена дополнительная информация.
- *Форма в режиме таблицы* (datasheet form) выглядит точно так же, как лист данных с таблицей. У этой формы не так много функциональных возможностей, как у других типов форм, но она бывает полезна, если вы хотите изменить настройку стандартного листа данных для отображения вашей информации. Можно создать табличную форму, отображающую меньше столбцов, применяющую фильтр для скрытия определенных записей, запрещающую вставку записей, использующую другое форматирование и т. д. Для создания формы в режиме таблицы выберите **Создание → Формы → Другие формы → Режим таблицы** (Create → Forms → More Forms → Datasheet).
- В *разделенной форме* в одном окне объединены два типа форм. Одна часть окна отображает текущую запись в простой форме. В другой части окна выводится таблица с несколькими записями. Для создания разделенной формы выберите **Создание → Формы → Разделенная форма** (Create → Forms → Split Form).
- *Модальное диалоговое окно* - специальный тип формы. Вместо отображения данных из таблицы модальное диалоговое окно задает вопрос. Идея заключается в том, что такое окно можно открыть в некоторый важный момент как часть автоматически выполняемой задачи. Для использования модальных форм вам придется иметь дело с кодом VBA. Вы увидите пример с использованием модальной формы (также называемой *диалоговой формой)* в *разд.*

"Добавление нового товара во время заполнения заказа" главы 17.

12.4. *Мастер создания форм*

Вы уже научились создавать разные широко используемые формы. Программа Access предлагает другой способ построения формы: с помощью Мастера создания форм. У этого мастера удивительное сходство с Мастером создания отчетов, который применялся в *главе 10.* Он задает ряд вопросов и затем создает соответствующую форму. Но вопросы крайне элементарны, и созданная форма не многим лучше добротной отправной точки для последующей настройки.

Далее описаны действия, необходимые для выполнения Мастера создания форм.

1. Выберите **Создание → Формы → Другие формы → Мастер форм** (Create → Forms → More Forms → Form Wizard). На экране появляется первое окно мастера **Создание форм.**

2. Из раскрывающегося списка выберите таблицу, которую хотите использовать. В списке **Доступные поля** отображаются все поля из вашей таблицы.

3. Добавьте поля, которые хотите включить, как показано на рис. 12.20. Когда закончите, нажмите кнопку **Далее.**

Можно выбрать поля из нескольких таблиц, при условии, что эти таблицы связаны.

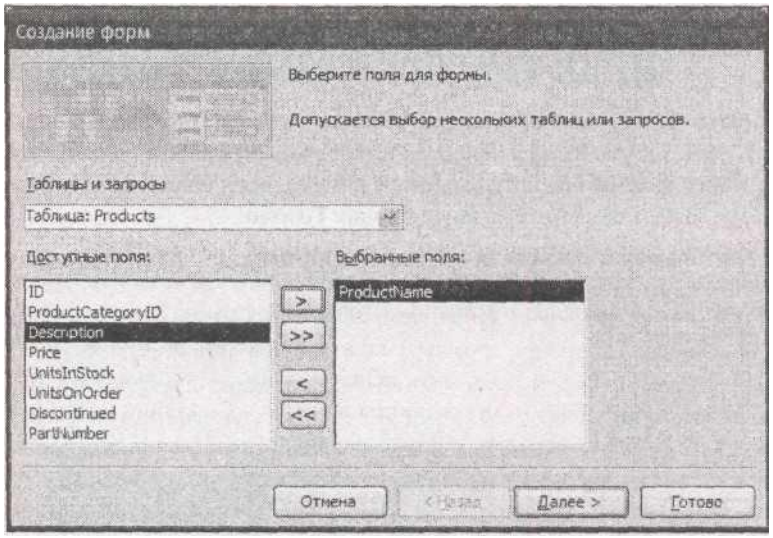


Рис. 12.20. Для вставки поля выделите его и затем щелкните мышью кнопку > для переноса поля из списка Доступные поля в список Выбранные поля. Для вставки всех полей нажмите кнопку »

4. Выберите вариант макета для вашей формы. К ним относятся следующие:

□ **в один столбец** создает форму с макетом в столбик. Этот выбор аналогичен выбору на ленте **Создание → Формы → Форма;**

□ **ленточный** создает форму с табличным макетом. Выбор этого макета аналогичен выбору на ленте **Создание → Формы → Несколько элементов;**

□ **табличный** создает форму, подобную листу данных. Этот макет аналогичен выбору на ленте **Создание → Формы → Другие формы → Режим таблицы;**

□ **выровненный** создает форму, не использующую макеты. Вместо этого элементы управления располагаются как можно ближе друг к другу, объединяя несколько полей в одной строке, если они достаточно малы. Выровненная форма - единственный вид формы, который невозможно создать только средствами ленты. Она аналогична формам без макетов, которые вы разрабатывали в *разд. "Высвобождение элементов управления из макета "ранее в этой главе.*

Примечание

Выровненные формы трудно корректировать в дальнейшем. Например, если потребуется добавить поле в середину макета формы, вы столкнетесь с трудоемкой задачей по перемещению на новые места множества полей, попадающихся на пути. Часто бывает легче создать снова форму с чистого листа с помощью мастера.

5. Выберите один из подготовленных заранее стилей и щелкните мышью кнопку **Далее**.

Стили определяют форматирование, которое программа Access применяет к вашей форме. К сожалению, трудно представить себе, как будет выглядеть конечный результат, пока не попробуете на деле каждый вариант стиля.

6. Введите имя формы.

Когда Мастер создания форм закончит работу, он тут же запишет форму с заданным именем.

7. Выберите вариант **Открыть форму для просмотра и ввода данных** (Open the form to view or edit information), если хотите начать использовать форму для работы с данными, или вариант **Изменить макет формы** (Modify the form's design), если сначала хотите настроить ее в **Конструкторе.** Затем щелкните мышью кнопку **Готово.**

Программа Access сохранит форму и откроет ее в **Режиме формы** или в **Конструкторе** в зависимости от выбранного вами варианта.

13. Глава 13. Проектирование сложных форм

Формы полезны программисту как небольшие защитные приспособления. Они упрощают выполнение повседневных задач и придают вашей БД строгий и оригинальный внешний вид. Для того чтобы стать профессионалом в разработке БД, необходимо научиться создавать первоклассные формы.

В предыдущей главе вы узнали, как формировать несколько распространенных типов форм. В этой главе вы поднимете навыки разработки форм на более высокий уровень с помощью целого арсенала новых методов и средств. Сначала вы научитесь создавать форму в свободном от каких-либо ограничений **Конструкторе,** позволяющем настроить и отшлифовать каждый квадратный сантиметр вашей формы. Затем вы познакомитесь с разными элементами управления Access и украсите вашу форму ссылками, панелями с вкладками и кнопками. Вы также научитесь работать со связанными таблицами с помощью создания форм специальных типов, называемых подчиненными формами и действующих в согласии с другими формами.

13.1. *Настройка форм в Конструкторе*

В предыдущей главе вы научились быстро создавать разные формы с помощью кнопок ленты и Мастера создания форм. Но серьезные специалисты по разработке форм применяют другой подход - они создают форму своими руками. Для решения этой задачи есть два пути.

- Создать форму в **Режиме макета.** Выберите на ленте **Создание → Формы → Пустая форма** (Create → Forms → Blank Form). Затем перетащите мышью на вашу форму нужные поля с панели **Список полей** (рис. 13.1). В *главе 12* вы узнали все необходимое для выполнения этой задачи. Можно быстро создать стандартную форму с макетом в один столбец или табличным макетом, но вы не получите дополнительных средств внешнего оформления.
- Создать форму в **Конструкторе.** Выберите на ленте **Создание → Формы → Конструктор форм** (Create → Forms → Form Design). Теперь вы начнете с пустой формы в окне **Конструктора.** Можно перетащить поля на форму с панели **Список полей** (так **же,** как в **Режиме макета)** и добавить с ленты множество разнообразных более специализированных элементов управления.

Примечание

Если на экране нет панели **Список полей,** выберите на ленте **Работа с макетами форм | Формат → Элементы управления → Добавить поля** (Form Layout Tools | Formatting → Tools → Add Existing Fields) (в **Режиме макета)** или **Инструменты конструктора форм | Конструктор**

→ Элементы управления → Добавить поля (Form Design Tools | Design → Controls → Add Existing Fields) (в **Конструкторе**).

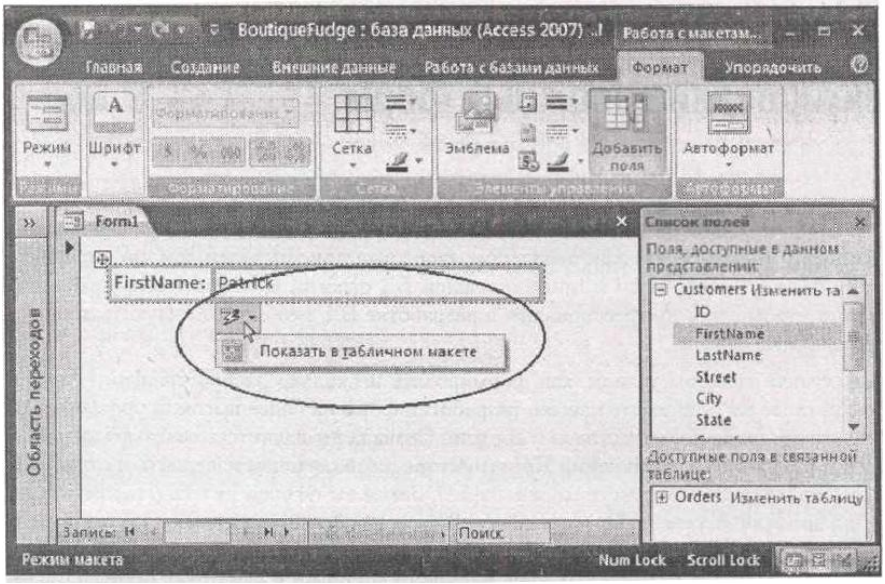


Рис. 13.1. Когда первое поле помещается на форму, открытую в **Режиме макета,** Access включает это поле в макет в один столбец и выводит на экран пиктограмму смарт-тега (обведена). Если вам нужен табличный макет, щелкните кнопкой мыши эту пиктограмму и выберите команду **Показать в табличном макете**

Конечно, в процессе работы над формой можно легко переключаться между этими двумя режимами. (Просто щелкните правой кнопкой мыши заголовок вкладки и выберите, какой хотите режим, или щелкните мышью кнопки режимов в нижнем правом углу окна программы.) Добавлять на форму поля можно в любом режиме. Но когда поля вставляются в **Режиме макета,** программа Access автоматически помещает их в макет. Когда поля добавляются в **Конструкторе,** они не привязаны к макету. Access полагает, что пользователи, применяющие **Конструктор,** хотят получить дополнительные средства управления размещением полей.

Подсказка

Вы можете свободно перемещаемое поле перенести в макет позже, перетащив его мышью в нужное место (см. рис. 12.13).

Существует и более серьезное отличие между **Режимом макета** и **Конструктором.**

В Конструкторе можно добавить элемент управления, выбрав его из десятка оригинальных элементов, таких как **Кнопки, Поля** и **Подписи.** Эти элементы определяют различие стандартных форм, сделанных автоматически программой Access, и форм, демонстрирующих персональный стиль.

13.1.1. *Разделы формы: разные части вашей формы*

В *главе 11* вы узнали, что отчет разделен на отдельные разделы (такие как **Заголовок отчета, Область данных, Примечание отчета** и т. д.), отображаемые в определенном месте. То же справедливо и для форм. Но впервые создаваемые формы начинают существование только с одним разделом: Областью данных, определяющей содержимое каждой записи.

Если хотите добавить заголовок или эмблему в верхнюю часть формы или некоторую сводную информацию или сообщение в нижнюю часть, следует вставить разделы верхнего и нижнего колонтитулов. Для добавления этих элементов в форму щелкните правой кнопкой мыши где-нибудь в области формы и выберите команду **Колонтитулы** (Page Header/Footer).

При работе с разделами формы помните о том, что размеры их должны быть малы (как показано на рис. 13.2). Размер всех разделов должен быть не более чем достаточен для отображения его содержимого. Если вы создадите слишком большую форму с множеством пустот, результаты будут выглядеть непрофессионально. На форме появятся ненужные полосы прокрутки, которые заставят вас прокручивать пустую область.

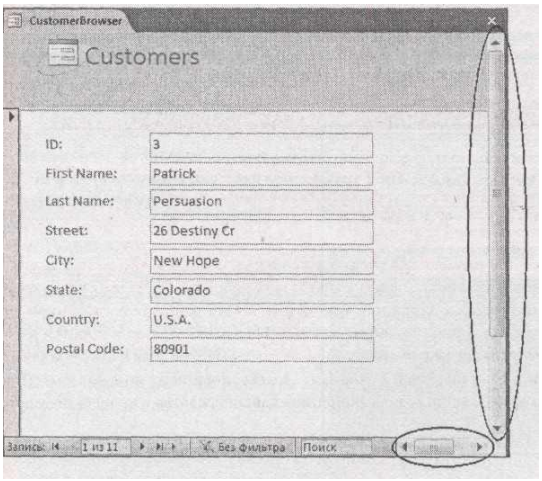


Рис. 13.2. Несмотря на то, что эта форма легко вмещает все поля в отображаемую область, у нее все же есть полосы прокрутки. Если перейти в **Конструктор,** то можно увидеть причину - форма шире и длиннее, чем нужно

Подсказка

Невозможно сделать форму меньше, чем содержащиеся в ней элементы управления. Этот факт часто становится камнем преткновения для разработчиков формы. Если программа

Access не позволяет изменить размер формы, значит, где-то что-то слишком велико. (Если все попытки тщетны, проверьте, не слишком ли большое поле в **Заголовке формы** или **Примечании формы.)**

Примечание

Если в вашей БД вместо вкладок задан режим перекрывающихся окон *(см. разд. "Открытие БД, созданной в более старой версии Access" главы 1),* вы столкнетесь с немного иной проблемой - окна ваших форм излишне велики. Иногда они могут даже не помещаться в главном окне программы Access, в этом случае Access обрезает их края.

13.1.2. *Вставка элементов управления в форму*

Впервые вы познакомились с элементами управления - графическими объектами, такими как **Подписи**[1] и **Поля** - когда создавали сложные отчеты в *главе 11*. Программа Access предоставляет такую же возможность и в формах. Для добавления их применяется та же группа на ленте. Но многие элементы управления, имеющие мало смысла в отчетах, на формах раскрываются во всей красе.

Примечание

За кадром на форме все компоненты - на самом деле элементы управления. При каждом добавлении поля вы имеете дело в итоге с двумя связанными элементами управления: Подписью, отображающей имя поля, и Полем, содержащим значение поля.

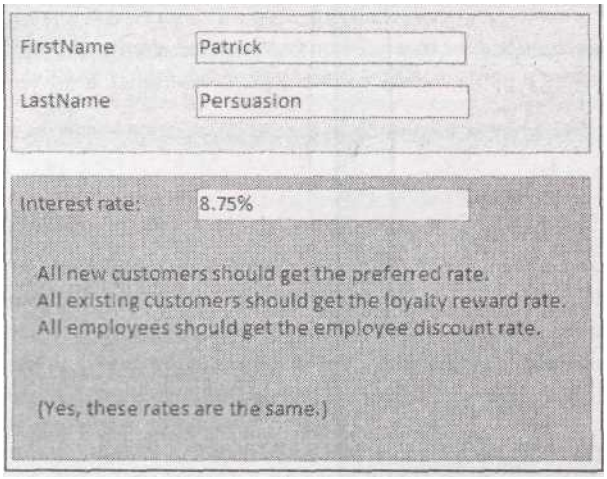


Рис.13.3. Применяйте **Подписи** для добавления пояснительных инструкций (или дерзких комментариев) на ваши формы. Элементы управления **Линии** и **Прямоугольники** добавят лоска

Один из самых простых **и** наиболее полезных элементов управления - скромная **Подпись. С** помощью **Подписи** можно **вставить отформатированный** текст в любое место

[1] В локализованной версии используются два названия этого элемента управления: **Надпись** и **Подпись.** - ***Пер.***

формы. Можно применять **Подписи** для выделения цветом дополнительных инструкций, как показано на рис. 13.3.

Для вставки элемента управления выполните следующие действия.

1. Перейдите в группу ленты **Инструменты конструктора форм | Конструктор →** Э**лементы управления** (Form Design Tools | Design → Controls).

В группе Э**лементы управления** собраны все элементы, которые можно использовать.

2. При желании нажмите кнопку **Использовать мастера** (Use Control Wizards). Эта кнопка

запускает мастера для элементов управления.

Элементы управления некоторых типов, такие как **Кнопки** и **Списки,** снабжены полезными мастерами. Как только такой элемент помещается на форму, запускается мастер, чтобы помочь вам настроить элемент. Обычно мастера включены. Но профессионалы, точно знающие чего хотят, могут решить, что мастера лишь мешают работать.

Когда кнопка **Использовать мастера** (находящаяся в крайнем правом ряду группы ленты **Инструменты конструктора форм | Конструктор → Элементы управления)** не подсвечена, мастера элементов управления оставляют вас без поддержки.

3. Щелкните кнопкой мыши пиктограмму нужного элемента управления.

На большинстве экранов мониторов программа Access не может уместить на ленте название кнопки элемента управления. (Владельцы 33-дюймовых мониторов могут поздравить себя и пропустить этот абзац.) Проведите указателем мыши по кнопке, и программа отобразит название элемента управления в поле всплывающей подсказки.

После щелчка пиктограммы кнопкой мыши она остается выделенной. Указатель мыши изменяется на крестик с маленьким присоединенным рисунком элемента управления. Это изменение - сигнал того, что элемент управления готов к использованию и ждет установки на форму.

Подсказка

Если вы впервые экспериментируете с элементами управления, почему не испробовать **Подпись** - с ней легко справиться и она действительно полезна.

4. Для того чтобы поместить элемент управления на форму, нарисуйте его контур на форме мышью с нажатой кнопкой.

Если с первого раза вы неудачно его расположили, всегда можно перетащить элемент управления на новое место или переместить его края для изменения размера элемента.

Если вы передумали вставлять выбранный элемент управления, просто щелкните мышью кнопку **Выбрать** (Select) (в крайнем правом ряду группы **Инструменты конструктора форм | Конструктор → Элементы управления).** Она выглядит как указатель мыши. Когда вы щелкните кнопку **Выбрать,** указатель мыши примет обычный вид. Теперь можно щелкнуть кнопкой мыши на форме для выделения имеющегося элемента управления. Новый элемент управления при этом не создается.

Подсказка

Еще более быстрый вариант - просто нажать клавишу <Esc> для отказа от создания элемента управления, после того как вы выбрали его на ленте.

5. Если у выбранного элемента управления есть мастер (Control wizard) и выбран режим использования мастеров элементов (см. пункт 2), на экране появится окно соответствующего мастера.

Ответьте на все вопросы для настройки элемента управления или нажмите клавишу <Esc> для пропуска мастера и выполнения самостоятельной настройки.

6. Если вы добавляете **Подпись,** введите текст, содержащийся в элементе управления.

После того как вы поместили **Подпись** на форму, программа Access ждет ввода текста **Подписи** (который присваивается свойству **Подпись** (Caption)). Если вы ничего не введете, Access решит, что вам на самом деле не нужна **Подпись** и избавится от нее.

7. Если на экране еще нет **Окна свойств** (в правой части окна программы), щелкните мышью **Инструменты конструктора форм | Конструктор → Сервис → Страница свойств** (Form Design Tools | Design → Tools → Property Sheet) для его отображения.

Для настройки многих параметров элементов управления или свойств вам придется пользоваться **Окном свойств.**

8. Измените соответствующие параметры в **Окне свойств.**

Если вы добавляете присоединенный элемент управления (bound control) *(см. примечание "На профессиональном уровне. Присоединенные элементы управления" далее в этом разделе),* выберите вкладку **Данные** (Data) и задайте в поле **Данные** (Control Source) имя поля, которое хотите отобразить.

Подсказка

Если у вас есть элемент **Подпись,** не вмещающий весь введенный текст, можно увеличить размер элемента за один шаг. Щелкните правой кнопкой мыши **Подпись** и выберите команду **Размер → по размеру данных** (Size → To Fit). Программа Access изменит размер надписи, увеличив ее настолько, чтобы поместилось все ее содержимое. Не пытайтесь проделать это с другими элементами управления, например, с **Полем** этот прием не работает.

На профессиональном уровне.

Присоединенные элементы управления

Присоединенный элемент управления - это элемент, отображающий значение поля БД. (Он называется присоединенным, поскольку тесно связан с соответствующим полем вашей таблицы.) Наиболее общий пример - **Поле,** к присоединенным элементам управления относятся также **Флажок, Список** и т. д.

Когда добавляется присоединенный элемент управления, вы должны задать связанное с ним поле, чтобы программа Access знала, что отображать. Легче всего добавить присоединенный элемент управления, перетащив на форму поле с панели **Список полей** и позволив программе Access создать элемент управления. Но ничто не мешает вам создать присоединенный элемент управления самостоятельно.

Начтите с помещения на форму элемента управления нужного типа (например, **Поля).** Затем выберите вкладку **Данные** и найдите параметр **Данные.** Именно в нем вы указываете соответствующее поле. Например, в элементе **Поле** с параметром **Данные,** содержащим **ProductName,** на вашей форме будет отображаться содержимое поля **ProductName.**

Конечно, этот принцип действует, только если источник данных вашей формы - таблица или запрос, на базе которых строится форма, - содержит поле, которое вы хотите использовать. Для изменения источника данных формы выберите объект **Форма** (Form) в Окне свойств, щелкните кнопкой мыши вкладку **Данные** (Data) и найдите свойство **Источник записей** (Record Source). В этом свойстве указано имя связанной таблицы или запроса или SQL-команда SELECT *(см. разд. "Режим SQL" главы 6),* которая получает нужные вам записи. Для выбора другой таблицы или запроса введите ее или его имя. Или же щелкните мышью кнопку с многоточием в поле свойства **Источник записей** для того, чтобы открыть окно запроса, позволяющее точно выбрать поля, которые хотите использовать, из множества связанных таблиц при необходимости с точными вариантами фильтрации и сортировки, которые нужны.

9. Если хотите, задайте элементу управления более подходящее имя в свойстве **Имя** (Name) (на вкладке **Другие** (Other)).

Если вы создали новую **Подпись,** программа Access присваивает ей имя, например, **Label46.** Если вы хотите наградить ваш элемент управления чем-то более подходящим, просто измените текст в свойстве **Имя.** В следующий раз, когда вы захотите изменить его, вам будет легче найти ваш элемент управления в раскрывающемся списке **Окна свойств.**

10. Отформатируйте элемент управления.

Несмотря на то, что с помощью **Окна свойств** вы сможете откорректировать множество параметров форматирования, гораздо легче применять ленту. Для форматирования основного шрифта и цвета используйте группу **Инструменты конструктора форм | Конструктор → Шрифт,** для задания стиля границ вокруг элемента управления - блок **Инструменты конструктора форм | Конструктор → Элементы управления.**

Подсказка

Хотите привлечь внимание к элементам управления с помощью тени и рельефного края? Выделите элемент, выберите визуальный эффект из списка **Инструменты конструктора форм | Конструктор → Элементы управления → Оформление** (Form Design Tools I Design → Controls → Special Effect). Это отличный способ заставить обычный прямоугольный элемент управления выглядеть несколько иначе.

Малоизвестная или недооцененная возможность.

Повторное применение ваших любимых настроек стиля границ

В группе **Элементы управления** есть часто не замечаемая кнопка **Задать стандартные свойства** (Set Control Defaults). Эта кнопка позволяет многократно использовать параметры границ. В этом случае, если вы подобрали отличные границы для одного элемента управления, можно быстро применить их для оформления других элементов.

Вот как действует этот прием. Допустим, вы создаете **Подпись** и пользуетесь кнопками группы **Элементы управления** для проведения вокруг элемента тщательно отформатированных границ с подходящими толщиной (волосяная линия), цветом (фуксия) и стилем линии (точечная). Вы можете повторно применить эти установки, выбрав вновь созданный элемент управления **Подпись** и щелкнув мышью кнопку **Задать стандартные свойства.** Теперь у следующей вставленной в эту форму подписи будут такие же параметры границ.

Команда **Задать стандартные свойства** действует на все элементы одного типа, поэтому можно хранить разные параметры границ для **Подписей, Рисунков, Полей** и т. д. Несмотря на то, что это интересный прием визуального оформления, многие профессионалы предпочитают задавать одинаковый формат нескольким элементам управления с помощью их одновременного выделения и последующего выбора параметров границ.

13.1.3. *Галерея элементов управления: краткий обзор*

Позже в этой главе мы рассмотрим создание часто применяемых форм с помощью элементов управления. Но сначала стоит познакомиться со всеми элементами управления, представленными на ленте, чтобы вы видели, что есть в наличии (и чего нет). В табл. 13.1 представлен каждый член семейства элементов управления.

Таблица 13.1. Элементы управления форм

Элемент управлений	*Описание*
Подпись или Надпись (Label)	Отображает постоянный текст. Подходит для заголовков, заметок и полезных инструкций
Поле (Text Box)	Отображает значение поля записи. Этот элемент можно использовать для вывода результата выражения, как описано в *разд. "Выражения" главы 11*
Флажок (Check Box)	Отображает значение логического поля. Если установлено значение *Да,* флажок помечен
Выключатель (Toggle Button)	Выводит кнопку, которая может находиться в двух состояниях: обычное и нажатое. Кнопка переводится из одного состояния в другое щелчком мыши. Выключатель - редко используемая вещь, но можно применять его для замены флажка и отображения значения логического поля. В этом случае кнопка нажата, если у поля значение *Да*
Поле со списком (Combo Box)	Отображает список, который раскрывается при щелчке кнопкой мыши по направленной вниз стрелке. Этот список может быть списком предлагаемых значений или может быть извлечен из другой таблицы. Access автоматически применяет Поле со списком для полей с подстановкой или связанных таблиц
Список (List Box)	Отображает большое поле со списком значений. Этот список может быть списком предлагаемых значений или может быть извлечен из другой таблицы. Поля со списком и Списки взаимозаменяемы - ключевое отличие состоит в том, что Списки занимают больше места, а Поля со списками позволяют ввести собственные значения, которых нет в списке
Вложение (Attachment)	Отображает первый файл, хранящийся в поле типа Вложение. Если это изображение, оно отображается непосредственно на форме. В противном случае вы увидите пиктограмму, обозначающую тип файла. Если в поле типа Вложение содержится несколько файлов, переходить от файла к файлу можно с помощью стрелок на мини-панели (которая появляется при щелчке кнопкой мыши этого поля), как показано на рис. 12.8
Группа переключателей (Option Group) и Переключатель (Option Button)	Группа переключателей - это прямоугольный контейнер, содержащий один или несколько Переключателей
Гиперссылка (Hyperlink)	Отображает фиксированную ссылку - синий подчеркнутый текст, который при щелчке кнопкой мыши переносит пользователя на конкретную Web-страницу. В *разд. "Переходы по ссылкам" далее в этой главе* показано, как этот элемент действует
Линия (Line) и Прямоугольник (Rectangle)	Это декоративные элементы управления. Умелые дизайнеры применяют их для отделения разделов и высвечивания важной информации
Рисунок (Image)	Отображает предоставленный рисунок. Отлично подходит для эмблем и изобразительных средств, делающих вашу форму внешне привлекательной и выгодно отличающейся от остальных. Задайте свойство Установка размеров (Size Mode) для

	определения, будет ли изображение обрезаться снизу для того, чтобы уместиться в отведенном ему поле (значение *Фрагмент),* растягиваться (значение *Вписать в рамку)* или масштабироваться без изменения рамки (значение *По размеру рамки,* стандартный режим отображения). Можно использовать даже свойство Мозаичное заполнение (Picture Tilling) для повторения изображения на площади большего размера. Для того чтобы ваши рисунки (и файлы БД) имели небольшой размер, пользуйтесь компактными графическими файлами JPG, а не раздутыми файлами BMP
Вкладка (Tab Control)	Отображает несколько вкладок с данными. На экран выводится одна вкладка - выбирается нужная вкладка щелчком кнопки мыши. Это фирменное средство ОС Windows позволяет разместить больше информации на меньшем пространстве. В *разд. "Компоновка с применением элемента управления Вкладка" далее в этой главе* приведен пример
Подчиненная форма (Subform)	Отображает еще одну форму внутри формы. Обычно в подчиненной форме выводятся связанные записи из подчиненной таблицы. Вы увидите действие такой подчиненной формы в *разд. "Элемент управления Подчиненная форма" далее в этой главе*
Диаграмма (Chart)	Создает базовую диаграмму с помощью Мастера диаграмм, включенного в пакет Office. Увы, диаграммы не слишком хорошо интегрированы в программу Access. Если вы хотите создать графическое представление данных, лучше использовать сводную диаграмму *(см. главу 9)* или экспортировать ваши исходные данные в программу Excel, у которой больше возможностей
Свободная рамка объекта (Unbound Object Frame)	Отображает содержимое, называемое объектом и полученное из другой программы с помощью механизма, напоминающего старый добрый метод OLE. Этот элемент управления можно применять для встраивания в форму электронной таблицы, звукового файла или документа Word. Многие отказываются от такого применения: результаты могут быть странными и приводящими в замешательство
Присоединенная рамка объекта (Bound Object Frame)	Аналогична Свободной рамке объекта, но этот элемент управления извлекает объект, который нужно отобразить, из поля текущей записи. Это средство кажется отличным, но замысловатый устаревший стандарт OLE вызывает больше проблем, чем он того заслуживает. Если вам нужно подобное средство, гораздо удобнее использовать поле типа Вложение с элементом управления Вложение, который разработан для решения таких проблем
Разрыв страницы (Page Break)	Указывает место разрыва страницы. Этот элемент управления действует только при распечатке формы. Обычно следует избегать применения этого элемента управления в формах и использовать его исключительно в отчетах, которые специально создаются для печатания
Элемент управления ActiveX	ActiveX - это стандарт разработки элементов управления, поддерживаемый различными программными платформами. Если существует специализированный графический объект, который необходимо применить в программе Access, можно купить элемент управления ActiveX у компании-разработчика компонента и затем поместить его в ваши формы. Тем не менее, будьте осторожны - применение элементов управления ActiveX часто требует кода большого объема, а это не входит в задачу данной книги

Часто задаваемые вопросы.

Осовременивание элементов управления Windows

Почему элементы управления выглядят такими старомодными?

Большинству пользователей Windows XP известна как операционная система, положившая начало эре нового декоративного оформления кнопок. Корпорация Microsoft в своем неослабевающем стремлении вносить незначительные изменения использовала Windows XP для изменения дизайна широко распространенных элементов управления, таких как кнопки и флажки.

Для нетренированного глаза отличия между средствами визуального оформления Windows XP и ее предшественников незначительны. Например, Microsoft заменила прямоугольные серые кнопки аккуратно скругленными кнопками, отбрасывающими желтую тень, когда указатель мыши перемещается поверх них. Большинство программ Windows приобрели более привлекательный внешний вид, а некоторые все еще задержались в прошлом. В предыдущих версиях Access было неважно, какую версию ОС Windows вы используете. В любом случае у ваших элементов управления было устаревшее внешнее оформление.

В Access 2007 формы автоматически получают новое внешнее оформление Windows XP (при условии, что у вас операционная система Windows XP или Windows Vista). Но возможно исключение. Если открывается БД, созданная в более ранней версии Access, формы сохраняют первоначальный устаревший внешний вид. Программа Access не хочет касаться никаких

аспектов внешнего вида ваших форм, в лучшем случае она может отказаться целиком от дизайна формы.

К счастью, последнее слово за вами. Если открывается БД старого стиля, выполните следующие действия для получения современного внешнего вида.

1. Выберите кнопку **Office → Параметры Access** (Office → Access Options).
2. На экране появится диалоговое окно **Параметры Access.**
3. В списке слева выберите категорию **Текущая база данных** (Current Database).
4. Программа Access отобразит параметры, относящиеся к файлу открытой в данный момент БД.
5. В разделе **Параметры приложений** (Applications Options) найдите параметр **Использование тем оформления Windows для элементов управления на формах** (Use Windows-themed Controls on Forms). Если вам нужны стили Windows XP, установите флажок, если нет - сбросьте его.
6. Щелкните мышью кнопку ОК.

13.1.4. *Расположение элементов управления на форме*

К настоящему моменту вы, возможно, освоили работу с элементами управления в режиме **Конструктор.**

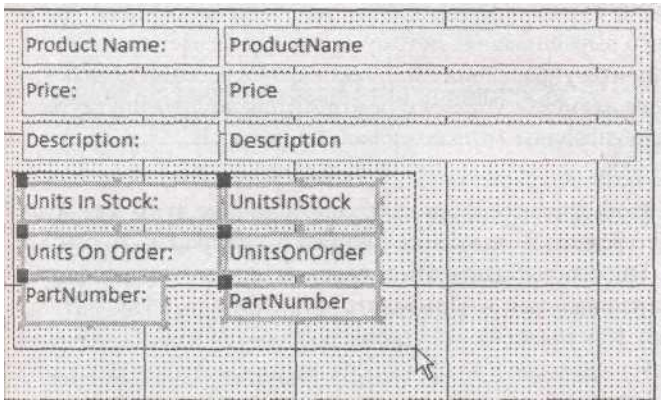


Рис. 13.4. Для одновременного перемещения нескольких элементов управления сначала щелкните кнопкой мыши в любом месте формы. Растяните рамку выделения вокруг всех элементов управления, которые хотите переместить, как показано на рисунке. После того как все элементы выделены, перетащите мышью один из них. Все элементы управления перемещаются как единое целое. (Вы могли бы держать нажатой клавишу <Shift> и щелкать кнопкой мыши каждый элемент управления по очереди.)

Далее приведена краткая сводка приемов, если вашей памяти нужен небольшой первоначальный толчок.

■ *Создание элемента управления.* Используйте ленту для выбора нужного элемента управления и затем нарисуйте его в нужном месте формы.

■ *Перемещение элемента управления.* Просто перетащите его. Можно передвигать несколько элементов управления одновременно, как показано на рис. 13.4.

■ *Изменение размеров элемента г/правления.* Перетащите мышью края прямоугольника, охватывающего элемент. Если у вас связанная комбинация **"Подпись - Поле"** (которую программа Access создает при добавлении поля БД), выбирайте нужную часть элемента для щелчка мышью. На рис. 11.9 показано, где щелкнуть кнопкой мыши для переноса только имени, только значения поля или обоих компонентов.

■ *Изменение элемента управления.* Выделите его и затем в **Окне свойств** найдите параметр, который нужно изменить.

■ *Удаление элемента управления.* Выделите его и затем нажмите клавишу <Delete> для полного стирания элемента.

Если элементы управления не включены в макет, может оказаться трудно аккуратно скомпоновать их. Для того чтобы помочь вам, программа Access предлагает ускоряющие

средства, способные выровнять шаловливые элементы управления и сгладить незначительные расхождения. В следующих разделах приведено несколько полезных советов по применению этих средств.

Практические занятия для опытных пользователей.
Как освободиться от привязки к сетке

Когда вы в **Конструкторе** помещаете элемент управления на форму или передвигаете его, программа Access всегда выравнивает его относительно ближайшего узла сетки. (Линии сетки в **Конструкторе** представлены точками, которые выводятся под элементами управления.) Access выполняет подобное выравнивание, потому что это облегчает создание согласованной формы. Если бы элементы управления были абсолютно свободно плавающими, было бы трудно выровнять два элемента управления относительно друг друга. Даже если у вас не трясутся руки, тяжело управлять мышью с такой точностью!

Однако в некоторых ситуациях возникает желание протолкнуть элемент управления между точками сетки. Обычно в этом случае на форме есть изображение, и вы пытаетесь создать интересный визуальный эффект. В подобных ситуациях программа Access позволяет освободиться от сетки. Просто выберите **Инструменты конструктора форм | Упорядочить → Макет элемента управления → Привязать** (Form Design Tools | Arrange → Control Layout → Snap to Grid). Как правило, эта кнопка подсвечена для обозначения постоянной привязки элементов управления к сетке. Для ее отключения щелкните кнопку мышью. Когда закончите, привязку можно вернуть, щелкнув эту кнопку еще раз.

При желании, если вы решите, что точки сетки отвлекают, их можно скрыть, используя кнопку **Инструменты конструктора форм | Упорядочить → Отображение → Сетка** (Form Design Tools | Arrange → Show/Hide → Show Grid). А когда вы сочтете, что нужно вернуть выравнивание элементов управления по линиям сетки, просто выделите все .. элементы, щелкните выделение правой кнопкой мыши и выберите команду **Выровнять → по узлам сетки.** Access подтолкнет каждый элемент управления к ближайшей линии сетки. Используйте команду **Размер → по узлам сетки** для того, чтобы ширина и высота элементов управления также соответствовала сетке.

13.1.4.1. *Выравнивание элементов управления*

Если есть группа элементов управления, которые следует подровнять, выделите их все (нарисовав рамку выделения, как показано на рис. 13.4), щелкните выделение правой кнопкой мыши и выберите один из вариантов в подменю **Выровнять.** Используйте во многих случаях популярный вариант **Слева** для выравнивания левых краев элементов управления. Можно также выровнять правые (рис. 13.5), верхние или нижние края элементов.

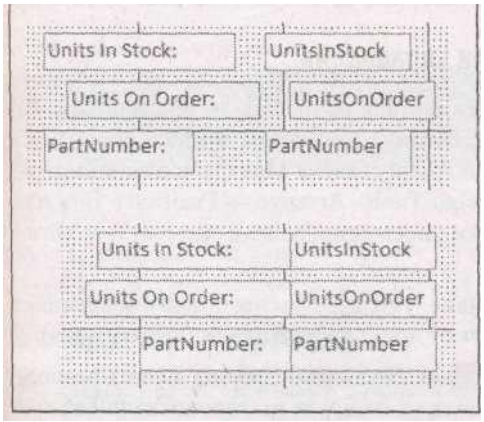


Рис. 13.5. *Вверху:* эти элементы управления выглядят неупорядоченными. *Внизу:* даже если элементы управления не включены в макет, их можно выровнять должным образом с помощью вариантов команды **Выровнять.** В данном случае применена команда **Выровнять → Справа** для выравнивания их правых краев

13.1.4.2. *Изменение размеров элементов управления*

Если на форме есть элементы управления разных размеров, можно заставить программу Access установить для них один и тот же размер.

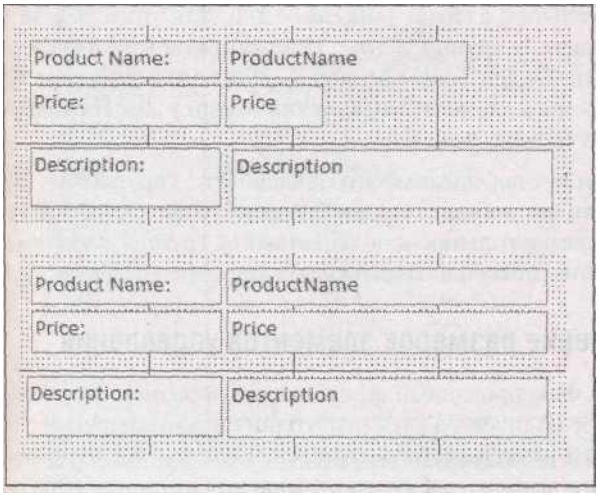


Рис. 13.6. Команда **по самому широкому** делает все эти **Поля** *(сверху)* одинаковой ширины *(внизу),* что создает более четкую и визуально более привлекательную форму

Выберите все элементы, щелкните правой кнопкой мыши выделение и выберите вариант из меню **Размер.** Используйте вариант **по самому широкому,** чтобы у всех элементов управления была ширина такая же, как у самого широкого из них (рис. 13.6). В противном случае можно сжать элементы, выбрав **по самому узкому** или изменить их высоты с помощью вариантов **по самому высокому** (To Tallest) и **по самому низкому** (To Shortest).

13.1.4.3. *Регулирование расстояния между элементами управления*

Если элементы управления случайным образом разбросаны на форме, их местоположение можно изменить так, чтобы между ними было согласованная величина пустого пространства. Для этого выделите все элементы и перейдите в группу ленты **Инструменты конструктора форм | Упорядочить → Положение** (Form Design Tools | Arrange → Position). В группе **Положение** есть несколько кнопок для регулировки расстояния между элементами управления:

- **Сделать интервалы по вертикали равными** (Make Vertical Spacing Equal) располагает элементы управления на равном расстоянии друг от друга по вертикали (сверху вниз);
- **Сделать интервалы по горизонтали равными** (Make Horizontal Spacing Equal) располагает элементы управления на равном расстоянии друг от друга по горизонтали (от края до края);
- **Увеличить интервал по вертикали** (Increase Vertical Spacing) и **Увеличить интервал по горизонтали** (Increase Horizontal Spacing) увеличивает расстояние между всеми выбранными элементами управления;
- **Уменьшить интервал по вертикали** (Decrease Vertical Spacing) и **Уменьшить интервал по горизонтали** (Decrease Horizontal Spacing) уменьшает расстояние между всеми выбранными элементами.

13.1.4.4. *Перекрывающиеся элементы управления*

Если у вас есть перекрывающиеся элементы управления, возможно, вам захочется определить, какой из них поместить поверх остальных, а какой - на самое дно. Для этого выделите один из элементов управления, перейдите в группу ленты **Инструменты конструктора форм | Упорядочить → Положение** (Form Design Tools | Arrange → Position) и выберите **На передний план** (Bring to Front) (для переноса элемента управления наверх) или **На задний план** (Send to Back) (для изгнания его на уровень фона).

Несомненно, что у большинства форм нет перекрывающихся элементов управления. Исключение составляют формы, в которых вы добиваетесь необычного графического представления или пытаетесь применить прямоугольник для обрамления группы элементов управления (в таком случае прямоугольник должен находиться под другими элементами).

13.1.5. *Привязка: автоматическое изменение размеров элементов управления*

Первоначально у элементов управления фиксированный, неменяющийся размер. Это свойство позволяет точно разместить большое количество элементов один следом за другим. Но у элементов управления фиксированного размера есть недостаток. Если вы увеличиваете окно программы Access до очень большого размера, элементы управления не смогут использовать дополнительное пространство. Наоборот, если вы делаете окно Access очень маленьким, вы неизбежно обрезаете часть формы. Другими словами, элементы управления фиксированного размера созданы для облегчения проектирования, но они лишены гибкости.

Большинство пользователей не беспокоят подобные ограничения. Они проектируют формы, которые хорошо помещаются на экране средних размеров *(см. примечание "На профессиональном уровне. Насколько велик ваш экран?" далее в этом разделе).* Но если у вас есть поле или поля, которые отображают данные большого объема - например, поле типа MEMO, до отказа заполненное текстом, - возможно, вас заинтересует эта проблема.

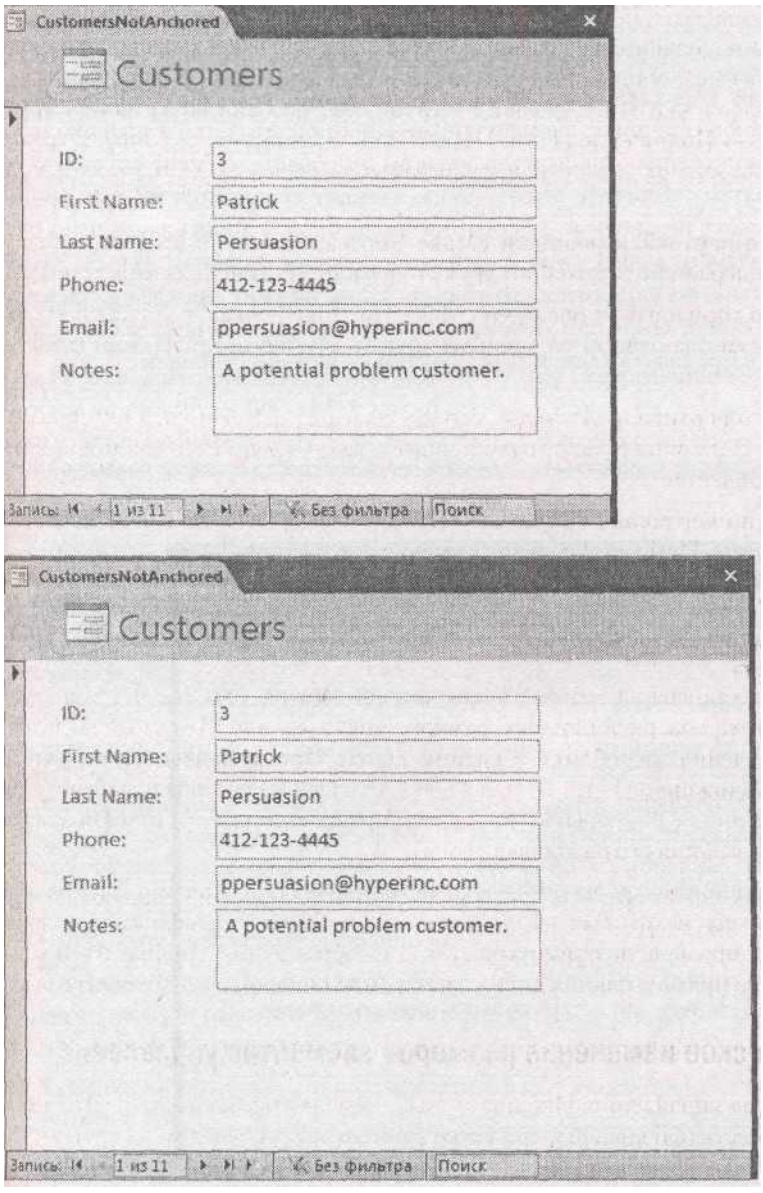


Рис. 13.7. Обычно все элементы управления привязаны к левому верхнему углу формы *(вверху).* Если размер окна формы меняется, с элементами управления ничего не происходит, поскольку левый верхний угол никогда не смещается *(внизу)*

В программу Access 2007 введено свойство, названное *привязкой,* позволяющее создавать элементы управления, которые могут увеличиваться для заполнения дополнительного пространства, когда изменяются размеры окна Access. Привязку выполнить корректно трудновато, но если у вас есть поля с тестом большого объема, овчинка стоит выделки.

По существу привязка позволяет присоединить элемент управления к сторонам формы. В

результате, когда изменяется размер формы, элемент управления переносится в новое местоположение или изменяет размер. На рис. 13.7 показана обычная форма, использующая стандартные параметры привязки. Когда изменяется размер формы, ничего не происходит.

Если привязать элемент управления к правой стороне формы, это будет совсем другая история. Когда форма становится шире, элемент управления прижимается к правой стороне, перемещаясь в новое положение. Аналогично, если вы привяжете элемент управления к нижней стороне окна и затем увеличите высоту окна, элемент сохранится на том же расстоянии от нижнего края окна, независимо от нового размера окна. Удивительная вещь произойдет, если привязать элемент управления к противоположным сторонам. В этом случае его положение не изменится, но изменится его размер. Если элемент привязан к левой и правой сторонам окна, он станет шире при увеличении ширины окна. На рис. 13.8 показано, как изменится форма, если привязать элементы управления к разным сторонам.

Подсказка

Расстояние между элементом управления и стороной привязки всегда остается неизменным.

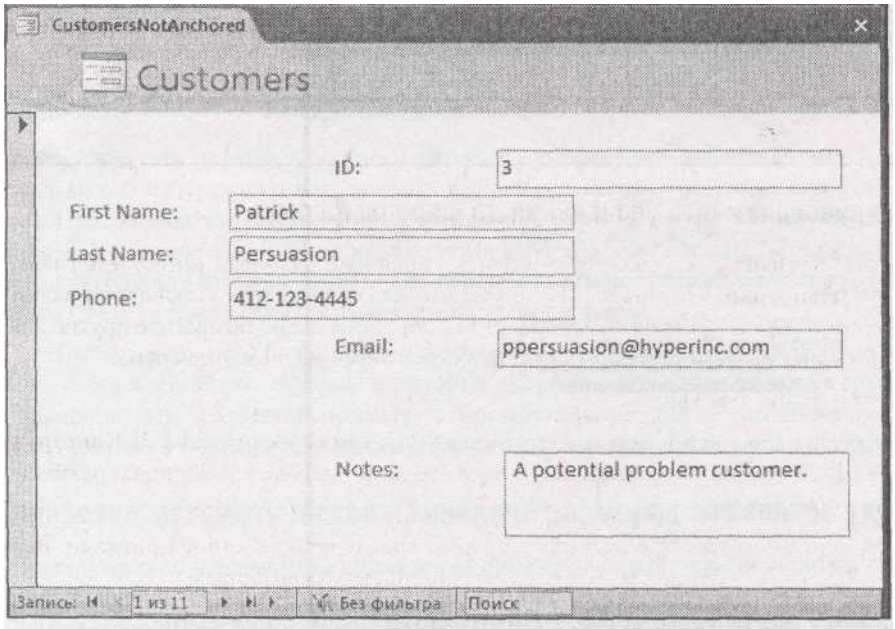


Рис. 13.8. В эту форму включены элементы управления с привязкой к разным сторонам формы. Поля **ID** и **Email** привязаны к правой и верхней сторонам, а поле **Notes** - к правой и нижней. Когда форма увеличивается, элементы управления перемещаются

Теоретически привязку можно применять для создания любого рода причудливых визуальных эффектов. Можно привязать элементы управления ко всем сторонам формы, так что они будут перемещаться и перекрывать друг друга при изменении размера формы, создавая на форме полнейшую неразбериху. В реальной жизни пользователи применяют привязку для достижения двух целей, описанных в следующих двух разделах.

На профессиональном уровне.

Насколько велик ваш экран?

Когда дело касается мониторов, размер не важен (на самом деле). Вот что действительно важно, так это разрешение монитора. Разрешение определяет количество пикселов (крошечных точек), отображаемое вашим монитором. Если разрешение вашего монитора высокое, на экране поместится больше содержимого. Единственный недостаток - все выведенное на экран будет мельче. Если раскрыть на весь экран окно программы Access и лист данных в нем, на экране с высоким разрешением вы увидите больше строк и столбцов одновременно по сравнению с экраном с низким разрешением.

Как правило, пользователи стараются применять разрешение 800x600 или 1024x768, хотя более высокие разрешения тоже не редки. (Разрешение 800x600 означает 800 пикселов по ширине и 600 пикселов по высоте.) Важно не только заданное вами разрешение. Если вы

планируете использовать БД Access совместно с другими людьми, возможно, вам захочется проверить, хорошо ли выглядят ваши формы при других часто используемых разрешениях.

Для определения текущего разрешения сверните все программы. Затем щелкните правой кнопкой мыши рабочий стол Windows и выберите команду **Свойства.** На экране появится окно **Свойства: Экран.** Выберите вкладку **Настройка** для выяснения текущего разрешения монитора и для испытания воздействия других разрешений на внешний вид ваших форм.

Увеличение ширины элемента управления до ширины формы

Первоначально вы устанавливаете размер элемента управления **Поле** и он остается неизменным. Но с помощью привязки можно растянуть или сжать элементы управления в соответствии с размером окна программы Access. И до тех пор, пока вы не поместите другие элементы управления у них на пути, у вас не будет проблем с перекрытием элементов.

Выполните следующие действия.

1. Сначала убедитесь в том, что у формы нет лишнего пустого пространства. В **Конструкторе** сожмите раздел **Область данных,** так чтобы его ширины хватало лишь на размещение элементов управления.

Если оставить лишнее пустое пространство, трудно проследить действие привязки. Вернитесь к рис. 13.2, чтобы вспомнить, как следует изменять размер формы.

2. Выделите элементы управления, которые хотите растянуть до ширины окна.

Если у вас форма, показанная на рис. 13.7, можно выбрать все элементы управления **Поле.** Держите нажатой клавишу <Shift>, пока щелкаете их кнопкой мыши.

На рис. 13.9 показан результат, который вы получите.

Примечание

Если элементы управления включены в макет *(см. разд. "Создание улучшенных макетов" главы 12),* следует перед применением привязки удалить их из макета. Несмотря на то, что можно применять параметры привязки ко всему макетному контейнеру, они не будут действовать так, как вы хотите, поскольку будут влиять как на размер элементов управления со значениями полей, так и на размер элементов управления с именами полей.

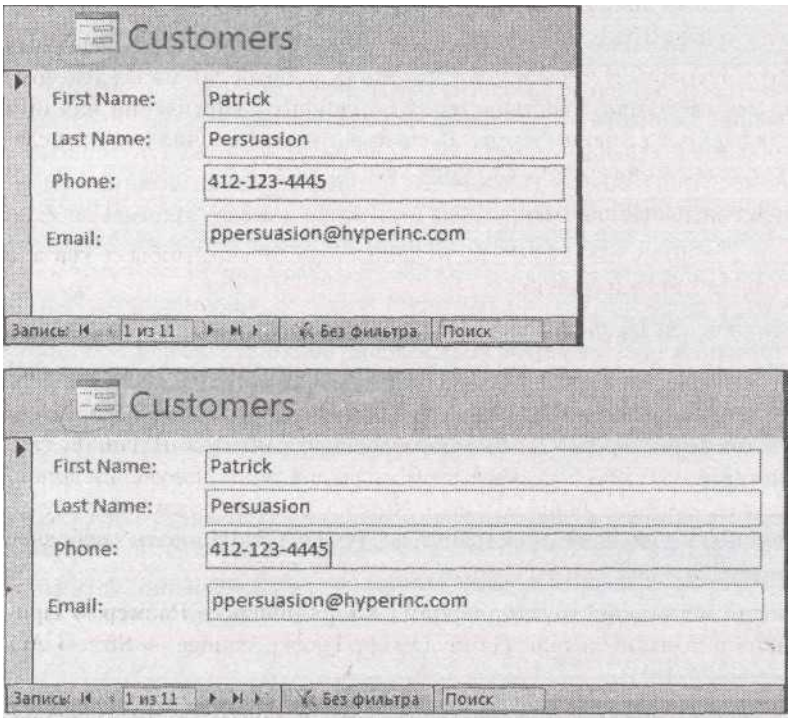


Рис. 13.9. *Вверху:* эти поля можно растянуть, чтобы они вмещали длинные имена и адреса электронной почты. Но не следует делать их такими широкими, чтобы пользователям приходилось прокручивать их от начала до конца, если их мониторы работают с более низким разрешением, чем ваш. *Внизу:* хорошим решением может стать привязка, в этом случае поля всегда используют все доступное пространство и ни на йоту больше

3. Выберите на ленте **Инструменты конструктора форм | Упорядочить → Размер → Привязка → Растянуть вдоль верхнего края** (Form Design Tools | Arrange → Size → Anchoring → Stretch Across Top). (Если вы предпочитаете настраивать параметры привязки в **Режиме макета,** выберите **Работа с макетами форм | Упорядочить → Положение → Привязка → Растянуть вдоль верхнего края** (Form Layout Tools | Arrange → Position → Anchoring → Stretch Across Top).)

Это действие привязывает элемент управления к трем сторонам формы: верхней, левой и правой. Привязка к верхней стороне гарантирует неизменность положения по вертикали при изменении высоты окна формы. Привязка к левой и правой сторонам обеспечивает расширение элементов управления при увеличении ширины формы и сжатие при ее уменьшении.

Увеличение размера элемента управления до максимально возможного

В предыдущем примере вы увидели, как применять привязку для увеличения горизонтального размера элемента управления. Привязку можно использовать для увеличения вертикального размера элемента, но здесь есть тонкость. В большинстве форм несколько элементов управления располагаются один над другим. Если не быть аккуратным, при увеличении высоты элемента управления он начинает закрывать расположенный под ним элемент.

Решением может стать увеличение вертикального размера только одного элемента управления на форме. Этот элемент (возможно, большое поле, набитое до отказа текстом) раздвигается, захватывая все свободное пространство. Все элементы управления над ним должны быть привязаны к верхней стороне формы. Все элементы управления под ним следует привязать к нижней стороне, чтобы они не стояли у него на пути.

Вот как реализовать эту модель на практике.

1. В **Конструкторе** сожмите ширину раздела **Область данных** настолько, чтобы ее хватало лишь для размещения элементов управления.

При любых типах привязки ваш главный враг - свободное пространство.

2. Выделите элемент управления, вертикальный размер которого хотите увеличить за счет имеющегося свободного пространства.

Внимательно рассмотрите форму на рис. 13.9, отображающую клиентов. В данном случае поле **Notes** с самым длинным текстом выиграет больше других от получения дополнительного пространства. Даже если привязать поле **Notes** к двум сторонам, вы получите незначительную порцию свободного пространства. Лучше использовать свободное пространство, имеющееся в нижней части формы.

3. Выберите на ленте **Инструменты конструктора форм | Упорядочить → Размер → Привязка → Растянуть вниз и по горизонтали** (Form Design Tools | Arrange → Size → Anchoring → Stretch Down and Across).

Этот шаг привяжет элемент управления ко всем четырем сторонам формы: верхней, нижней, левой и правой. В результате элемент увеличится, если форму расширить или растянуть по вертикали. Если вы хотите, чтобы элемент управления увеличивал только вертикальный размер, а горизонтальный оставался прежним, выберите **Привязка → Растянуть вниз** (Anchoring → Stretch Down).

4. Выделите первый элемент управления под элементом, увеличивающим вертикальный размер. Выберите **Привязка → Снизу слева** (Anchoring → Bottom Left).

Это действие привязывает элемент управления к левой и нижней сторонам. Таким образом, когда форма растягивается по вертикали, элемент управления смещается вниз, освобождая место для элемента, расположенного над ним.

Можно также использовать вариант **Растянуть вдоль нижнего края** (Stretch Across Bottom). В этом случае элемент управления также привязан к нижней стороне, но он увеличивается горизонтально в соответствии с шириной формы.

Примечание

В предыдущем примере подписи перед каждым полем не нуждались в привязке, поскольку они оставались на прежнем месте. Но в данном примере следует применить вариант привязки

Снизу слева ко всем подписям, расположенным под элементом управления,

растягивающемся вертикально. В противном случае подпись не будет выровнена относительно соответствующего поля со значением. Никогда не применяйте варианты растягивающей привязки к подписи, поскольку нет необходимости изменять ее размер.

5. Повторите пункт 4 для каждого элемента управления, расположенного ниже.

Если вы пропустили элемент управления, то увидите предупреждающий сигнал. Когда вы уменьшите окно формы, одни элементы управления наложатся на другие из-за несогласованности различных параметров привязки.

При условии, что привязка выполнена корректно, вы получите результат, показанный на рис. 13.10.

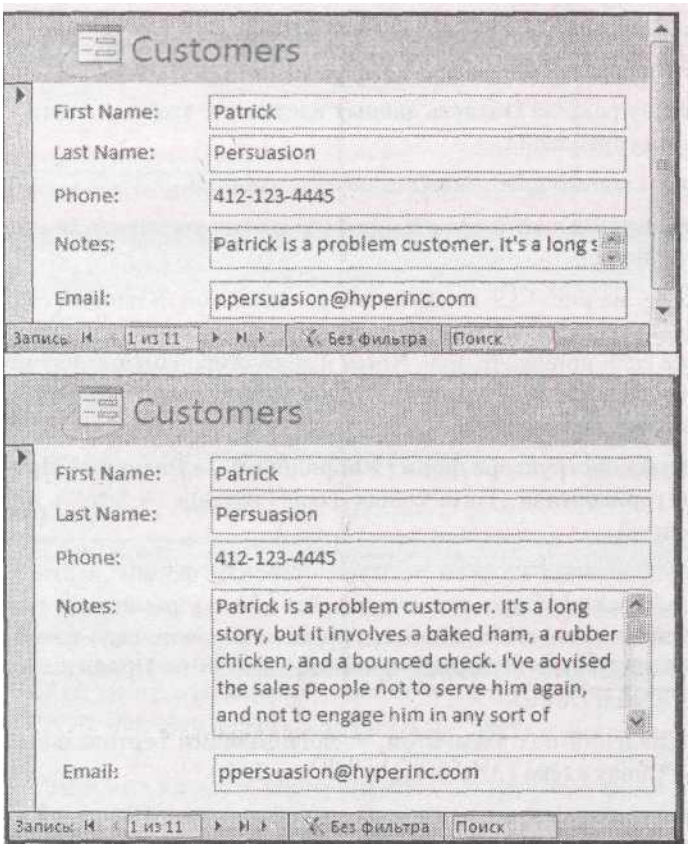


Рис. 13.10. Теперь, когда окно формы станет больше по вертикали, поле **Notes** получит дополнительное пространство

13.1.6. *Последовательность перехода: облегчение переходов с помощью клавиш*

Когда форма применяется для редактирования записи, необходимо переходить от одного поля к другому. Перейти в любое поле на форме можно с помощью мыши, но фанаты клавиатурных команд не хотят терять время на отрывание пальцев от клавиш. Тут на помощь приходит клавиша <Tab>.

Возможно, вам известно, что клавиша <Tab> позволяет переходить от одного элемента управления к другому в любом приложении Windows. Она также действует на листе данных, обеспечивая переход из одного столбца в следующий. Поэтому вас не должно удивить то, что клавиша <Tab> работает и в формах.

Первое нажатие клавиши <Tab> на форме может привести в изумление. Если вы потратили много времени на настройку элементов управления и их переупорядочивание, клавиша <Tab> необязательно приведет вас к тому элементу управления, на который вы рассчитываете. Рис. 13.11 иллюстрирует эту проблему.

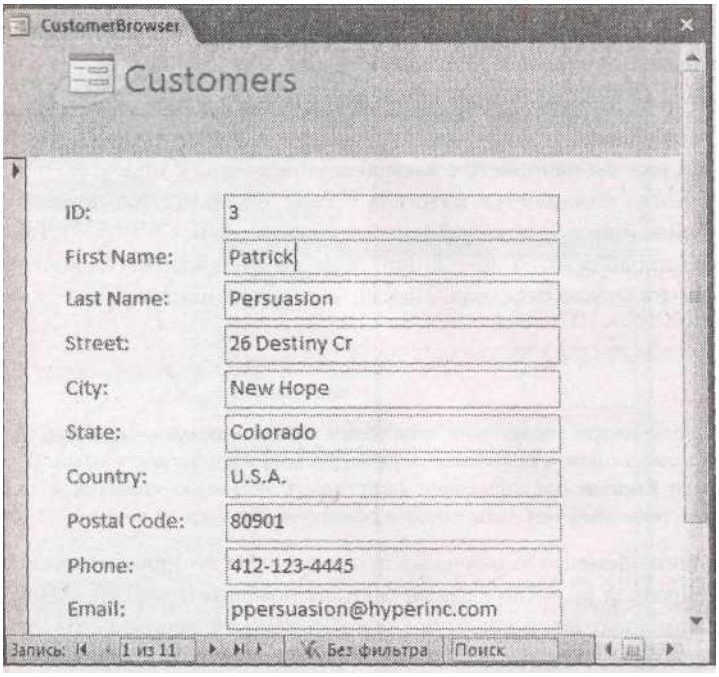


Рис. 13.11. Вы рассчитываете, что, нажав клавишу <Tab>, перейдете из поля **FirstName** в поле **LastName.** Но попробуйте сделать это и окажетесь в поле **Country,** расположенном в середине формы

Примечание

Клавиша <Tab> всегда действует корректно, если применяется табличный макет или макет в столбец, поскольку программа Access помнит ее последний переход, пока вы перемещаетесь между элементами управления. Только если вы извлекли элементы из макета, вы столкнетесь с описываемой проблемой.

Настройка формы таким образом, чтобы клавиша <Tab> обеспечивала обычный порядок перехода от одного элемента управления к следующему, называется установкой *последовательности перехода.* По сути, у каждого элемента управления, поддерживающего переходы с помощью клавиши <Tab>, есть три важных свойства (которые можно найти на вкладке **Другие** в **Окне свойств).** Это следующие свойства.

- **Переход по Tab** (Tab Stop) определяет, поддерживает ли элемент управления переходы с помощью клавиши <Tab>. Если установить значение *Да,* можно с помощью клавиши <Tab> перейти в этот элемент. Если изменить значение свойства на *Нет,* неважно, сколько раз вы нажмете клавишу <Tab> - вы никогда не попадете на этот элемент управления. Когда впервые вставляется элемент, у этого свойства всегда установлено значение *Да.*
- **Автопереход по Tab** (Auto Tab) оказывает влияние, только если элемент управления использует маску ввода *(см. разд. "Маски ввода " главы 4).* Если задать этому свойству значение *Да,* то как только вы введете последний символ в маску, вы автоматически перейдете к следующему элементу управления. Это свойство очень удобно для быстрого ввода данных, но может мешать, когда допущена ошибка, поскольку вы переходите к следующему элементу прежде, чем успеете ее исправить.
- **Индекс перехода по Tab** (Tab Index) управляет последовательностью перехода - иначе говоря, определяет, куда вы переходите при каждом нажатии клавиши <Tab>. Когда форма открывается в первый раз, вы начинаете с элемента управления с индексом перехода по <Tab>, равным 0. Когда нажимается клавиша <Tab>, выполняется переход к элементу управления с ближайшим превышающим индексом перехода по <Tab> (например, 1). Этот процесс продолжается до тех пор, пока вы не достигнете элемента управления с максимальным индексом перехода. Нажмите снова клавишу <Tab>, и вывернетесь к началу.

Примечание

Эти свойства есть только у элементов управления, способных принять фокус - другими

словами, элементов, которые можно щелкнуть кнопкой мыши для взаимодействия с ними. Очевидно, что **Поля, Флажки** и **Кнопки** поддерживают переходы с помощью клавиши <Tab>. А **Подписи** и **Рисунки -** нет, поскольку нет возможности обмена информацией с ними.

При каждом добавлении нового элемента управления программа Access присваивает ему новый более высокий индекс перехода по <Tab>. Даже если вы помещаете новый элемент на самый верх формы, Access помещает его в конец последовательности перехода. Для исправления этой проблемы можно выделить каждый элемент управления в **Конструкторе** и изменить значение свойства **Индекс перехода по Tab** вручную. Но альтернатива, требующая гораздо меньших затрат времени, позволяет установить последовательность перехода для всей формы сразу. Вот как она действует.

1. Щелкните правой кнопкой мыши свободное место на форме и выберите команду **Переходы.**

На экране появится диалоговое окно **Последовательность перехода.** В нем перечислены все элементы управления на вашей форме, поддерживающие переходы по клавише <Tab>, начиная с самого низкого индекса перехода и заканчивая самым высоким.

2. В списке **Раздел** (Selection) выберите раздел формы, с которым хотите работать. Почти всегда это **Область данных.**

В диалоговом окне **Последовательность перехода** (Tab Order) можно задать нужную последовательность отдельно для каждого раздела формы. Если в форму включены **Заголовок формы** и **Примечание формы,** можно выбрать работу с одним из этих разделов или с **Областью данных.** Но очень редко попадается форма с элементами управления, поддерживающими переходы по клавише <Tab>, расположенными за пределами **Области данных.**

3. Если хотите позволить программе Access попытаться задать правильную последовательность переходов, щелкните мышью кнопку **Авто** (Auto Order).

При щелчке мышью кнопки **Авто** Access устанавливает последовательность перехода на основе местоположения элементов управления. Порядок следования слева направо и затем сверху вниз. В большинстве случаев режим **Авто** задает верную последовательность перехода (или, по крайней мере, приближает вас к ней).

4. Для переноса одного элемента управления в новое место в последовательности перехода перетащите его мышью.

Этот шаг требует небольшой ловкости. На рис. 13.12 показано, как он выполняется.

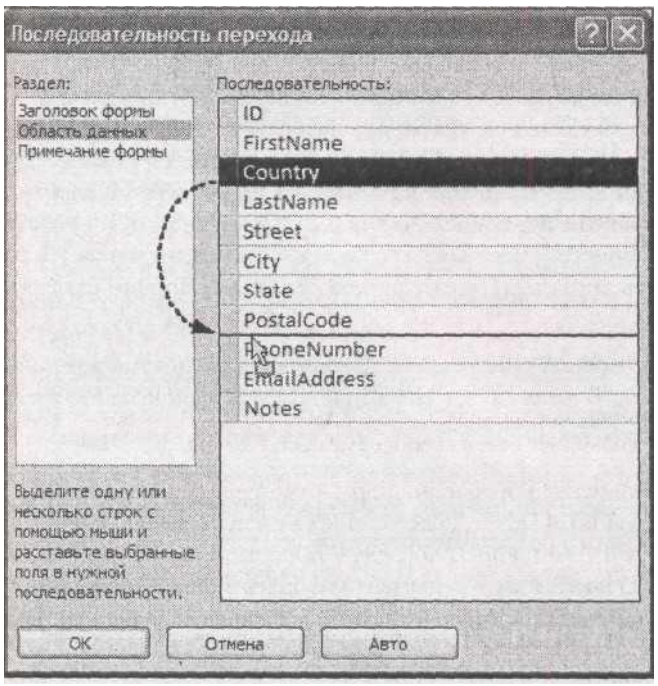


Рис. 13.12. Для изменения местоположения элемента управления в последовательности

перехода начните со щелчка кнопкой мыши серого поля, находящегося слева от элемента управления. Выделится вся строка списка. Далее перетащите элемент в новую строку в списке. В данном примере поле **Country** перемещается вниз в последовательности перехода

5. Когда исправите последовательность перехода, щелкните мышью кнопку **ОК.**

Подсказка

Последовательность перехода действует в двух направлениях. Можно перейти к очередному элементу в последовательности, нажав клавишу <Tab>, и к предыдущему элементу, нажав комбинацию клавиш <Shift>+<Tab».

13.2. *Контроль с помощью элементов управления*

Вы уже знаете, как создавать форму с чистого листа и добавлять все необходимые элементы управления. Но вы еще не воспользовались новой функциональной возможностью, позволяющей делать нечто особенное. Вы научились вставлять подписи, линии и прямоугольники. Но все эти оформительские украшения меркнут по сравнению с действительно полезными средствами, которые программа Access позволяет добавлять в ваши формы. Хотите помешать вводу ошибочных данных? Пожалуйста. Хотите добавить гиперссылки в стиле Web? Нет проблем. Список того, что вы можете сделать для расширения функциональных возможностей ваших форм, почти бесконечен. В следующих разделах описаны самые популярные способы контроля данных на форме с помощью элементов управления.

13.2.1. *Блокировка полей*

В БД почти все порции информации - объекты для изменения. Но это не означает, что следует всем предоставить свободный доступ ко всем полям.

Предположим, что компания Boutique Fudge создает форму **CurrentOrders** (текущие заказы), которая позволит персоналу склада просматривать ожидающие обработки заказы клиентов, отсортированные по дате. Персонал склада должен просмотреть каждый заказ, упаковать и затем отправить его. Единственное изменение, которое она должны вносить, - обновление состояния заказа (обозначить дату его отправки) или вставить запись в журнал регистрации доставки. Другие подробности, такие как дата заказа, содержимое заказа и клиент, получающий заказ, должны быть закрыты. У сотрудников склада нет причин изменять эти данные.

В подобных сценариях формы - мощный инструмент, поскольку они позволяют помешать изменению определенных полей. В этом случае случайно нажатая неподходящая клавиша не сможет стереть порцию корректной информации.

У любого присоединенного элемента управления (элемента, отображающего поле вашей таблицы) есть два свойства, которые можно применять для контроля редактирования. Эти свойства можно изменять в **Окне свойств** в режиме **Конструктор.**

- **Блокировка** (Locked) определяет, можно ли изменять поле. Если у свойства значение *Да,* нельзя редактировать значение поля. Но при этом можно выделить содержимое поля и скопировать его.
- **Доступ** (Enabled) позволяет полностью отключить элемент управления. Если значение свойства равно *Нет,* элемент выводится с недоступным текстом серого цвета. Несмотря на то, что вы видите значение поля в отключенном элементе управления, обмениваться с ним информацией у вас нет никакой возможности. Если это элемент управления Поле, вы даже не можете выделить и скопировать содержащийся в нем текст.

Подсказка

Если вы хотите запретить любые варианты редактирования, рассмотрите возможность применения свойств формы **Разрешить изменение** (Allow Edits), **Разрешить удаление** (Allow Deletions) и **Разрешить добавление** (Allow Additions), описанных в табл. 12.2.

13.2.2. *Предупреждение ошибок с помощью условий на значения*

В *главе 4* вы узнали, как предупреждать проникновение ошибок в ваши таблицы с помощью

правил верификации или условий на значение, значений по умолчанию и масок ввода. Такая многоуровневая проверка - существенная часть проектирования БД.

Но в некоторых ситуациях условия на значения не помогут, поскольку они применяются не всегда, а время от времени. Вы вряд ли захотите, чтобы продавцы в Boutique Fudge вводили новый заказ со старой датой. Ясно, что это ошибка - у нового заказа должна быть текущая дата. Для выявления и устранения этой проблемы умный разработчик, вроде вас, может использовать следующее условие на значение в поле **OrderDate:**

<=Date ( )

Однако через несколько недель вы обнаруживаете, что отдел поставки продуктов не счел нужным ввести информацию о своих заказах вовремя. Для сохранения записей задним числом следует снабдить эти заказы датами их первоначального появления. Вам придется удалить хорошо продуманное условие на значение, прежде чем вы сможете ввести эти записи.

Как выясняется, ситуации, подобные этой, в реальной жизни возникают часто. К счастью, существует способ обработки такого сценария без отказа от условия на значение. Прием заключается в помещении условия на значение в форму. В этом случае разные формы могут использовать различные условия на значение. Если вам нужны ничем неограниченные изменения, данные можно редактировать непосредственно в таблице на листе данных.

Если вы планируете убрать условия на значение из ваших таблиц и внести их в формы, вам будет интересно узнать о следующих свойствах элементов управления, которые можно настраивать в **Окне свойств.**

■ **Условие на значение** (Validation Rule) задает выражение, которому должно удовлетворять значение для того, чтобы считаться допустимым. Например, выражение <=Date ()
сравнивает текущее значение поля с датой, возвращаемой функцией Date () (которая представляет текущую дату). Ввод разрешается, только если вводимая дата сегодняшняя или предшествующая. Гораздо больше примеров выражений для проверки значений можно найти в *разд. "Запись условия на значение поля" главы 4.*

■ **Сообщение об ошибке** (Validation Text) задает текст сообщения об ошибке, которое выводится, если вы пытаетесь ввести значение, нарушающее условие на значение. Этот пользовательский текст заменяет общее сообщение об ошибке программы Access - "Для введенного значения не выполняется условие на значение данного поля или элемента управления" - в котором для реальных пользователей мало смысла.

■ **Маска ввода** (Input Mask) задает образец, который направляет и ограничивает пользовательский ввод. Маски ввода - хороший способ обработки текстовых значений фиксированной длины, например, телефонных номеров, почтовых кодов и номеров социального обеспечения. В *разд. "Маски ввода" главы 4* приведена подробная информация о том, как действуют и создаются маски ввода.

■ **Значение по умолчанию** (Default Value) задает значение, которое выводится в поле, когда создается новая запись. (Вы, конечно, можете изменить значение по умолчанию, если оно вас не устраивает.) Для форм значения по умолчанию очень удобны, поскольку такие значения чаще применяются в конкретных задачах, а не во всей таблице.

Примечание

Можно задать значение по умолчанию для одного и того же поля на уровне таблицы и на уровне формы. Если вы сделали это, значение по умолчанию формы перекрывает значение, заданное в таблице.

13.2.3. *Выполнение вычислений в выражениях*

Выражение - это формула, обрабатывающая некоторую информацию, например числа, даты или текст, и отображающая конечный результат (рис. 13.13). Часто выражения в вычислениях используют значения полей. Раньше вы применяли выражения для обработки чисел в запросах *(см. главу 7)* и отчетах *(см. главу 10),* а теперь вы заставите их работать в формах.

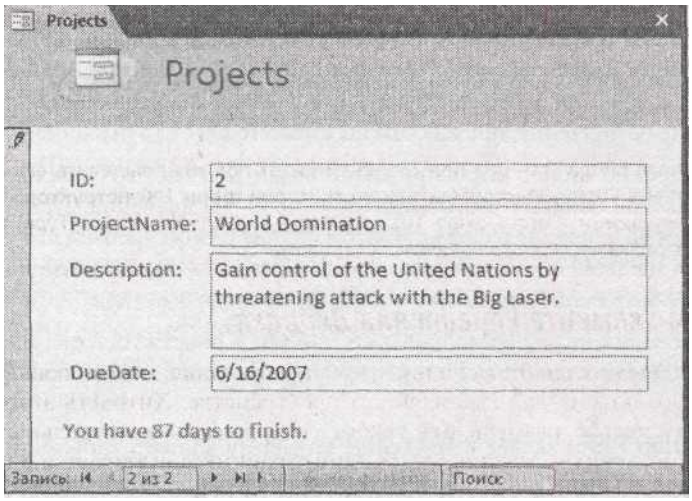


Рис. 13.13. В данной форме выражение = "You have " & [DueDate]-Date () & " days to finish" вычисляет число дней между текущей датой и датой окончания проекта и помещает его в законченное предложение. Вы увидите появление этой информации, как только введете дату окончания и перейдете в другое поле. (Это требование можно обойти и заставить поля обновляться во время ввода с помощью небольшого фрагмента VBA-кода, запускающего повторное вычисление.)

Для создания выражения выполните следующие действия.

1. Добавьте на форму элемент управления **Поле** (из группы ленты **Инструменты конструктора форм | Конструктор → Элементы управления).**

Следует использовать именно **Поле,** т. к. оно может отображать переменные значения, такие как выражения. Подпись может показывать только текстовые константы, поэтому она бесполезна.

2. В **Окне свойств** выберите вкладку **Данные** (Data). Поместите выражение в параметр **Данные** (Control Source).

Помните о том, что выражения начинаются со знака равенства. Выражение =Price*l. 15 вычисляет цену товара с учетом налога, умножая значение в поле **Price** на 1.15.

3. Можно задать в свойстве **Доступ** (Enabled) значение *Нет* для усиления запрета на изменение этого значения.

Когда создается элемент управления с применением выражения, программа Access не разрешает редактировать вычисляемое значение. Это все равно, что свойству **Блокировка** (Locked) присвоить значение *Да.* Но некоторые пользователи все равно могут делать попытки изменить эту величину. Если по-вашему такое развитие сценария может создавать проблемы, задайте в свойстве **Доступ** значение *Нет,* чтобы элемент управления отображался как недоступный и никто не мог перейти в него с помощью клавиши <Tab>. Эта установка также означает невозможность копирования значения в поле ввода.

4. При желании примените форматирование.

Вы можете настроить шрифт и цвет с помощью группы ленты **Инструменты конструктора форм | Конструктор → Шрифт** (Form Design Tools | Design → Font). Для настройки отображения программой Access числовых значений перейдите в **Режим макета** и используйте группу ленты **Работа с макетами форм | Формат → Форматирование** (Form Layout Tools | Formatting → Formatting).

Примечание

Для удаления рамки вокруг поля ввода (так оно больше напоминает подпись) выделите его в **Конструкторе,** выберите на ленте кнопку **Инструменты конструктора форм | Конструктор → Элементы управления → Тип линии** (Form Design Tools | Design → Controls → Line Type) и укажите первый элемент списка (он пустой, что означает

"отсутствует").

13.2.4. *Компоновка с применением элемента управления Вкладка*

Один из невоспетых героев мира элементов управления - элемент **Вкладка,** позволяющий представить содержимое большого объема на ограниченном пространстве. Хитрость этого элемента управления кроется в способе размещения такого содержимого на отдельных страницах. В определенный момент времени можно видеть одну страницу, а выбирается она щелчком кнопкой мыши соответствующего заголовка вкладки (рис. 13.14).

Элемент управления **Вкладка** не идеален. Его главный недостаток заключается в необходимости дополнительных щелчков кнопкой мыши для перехода с одной вкладки на другую. По этой причине не очень удобно пользоваться вкладками на формах, предназначенных для создания новых записей. В этом случае лучше упростить процесс создания записи и расположить все элементы управления на одной странице, так чтобы вы могли быстро заполнить их. Вкладки полезнее всего на формах, разработанных преимущественно для редактирования и просмотра данных. Если такие данные можно разделить на логические группы и если редактирование часто связано с корректировкой одной из групп, элемент управления **Вкладка** - удачный выбор.

Для применения элемента управления **Вкладка** выполните следующие действия.

1. В группе ленты **Инструменты конструктора форм | Конструктор →• Элементы управления** щелкните кнопкой мыши пиктограмму **Вкладка.**

Когда **Вкладка** появится на экране, вам, вероятно, захочется переместить ее или изменить ее размеры для наилучшего соответствия форме.

2. Добавьте все нужные страницы **Вкладки.**

Любой новый элемент управления **Вкладка** сначала отображается с двумя страницами. Перейти со страницы на страницу можно щелчком кнопки мыши по соответствующему заголовку.Для создания новой страницы щелкните правой кнопкой мыши любую страницу (но не заголовок вкладки) и выберите **Вставить вкладку** (Add Page). Для удаления существующей страницы щелкните ее правой кнопкой мыши и выберите команду **Удалить вкладку** (Delete Page).

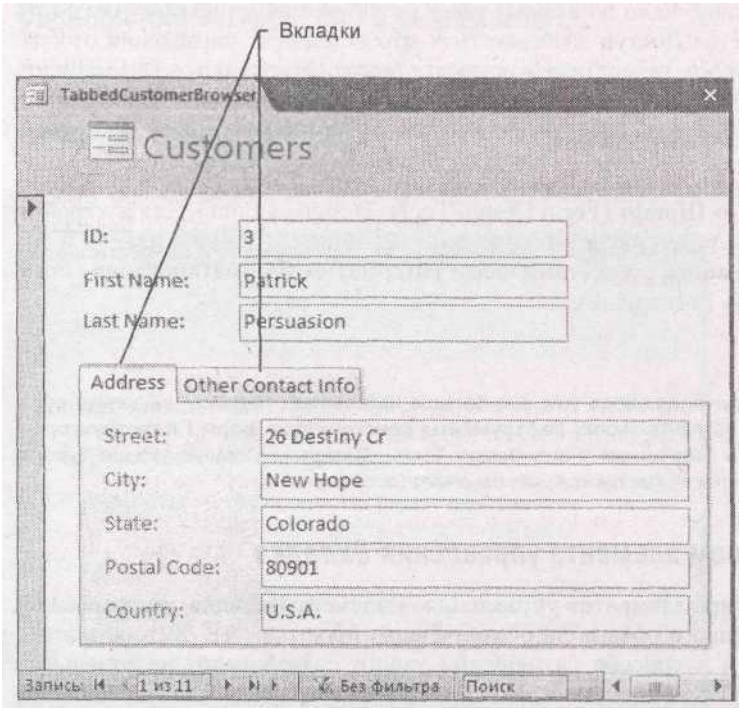


Рис. 13.14. Имеет смысл поместить адресную информацию клиента на отдельную вкладку, поскольку все эти поля образуют группу логически связанных данных

3. Задайте подходящие имена вкладок.

Имена, которые первоначально присваивает вкладкам программа Access,

малоинформативны, например, **Вкладка19** и **Вкладка20.** Для изменения заголовка или имени вкладки выделите страницу и измените параметр **Подпись** (Caption) в **Окне свойств.** У страницы, отображающей адресные поля клиента, мог бы быть заголовок "Address Information" (адресные данные).

Для изменения порядка следования страниц щелкните правой кнопкой мыши элемент управления **Вкладка** и выберите команду **Последовательность вкладок** (Page Order). Программа Access откроет диалоговое окно **Порядок страниц** со списком вкладок. Для изменения местоположения вкладки выделите ее и щелкните мышью кнопку **Вверх** или **Вниз.**

Примечание

Если вы создаете больше страниц, чем может уместиться в вашем элементе управления **Вкладка,** программа Access добавляет странную полосу прокрутки в правый верхний угол элемента, которая позволяет переходить к нужным вкладкам. Для того чтобы избежать этого неудобства, измените размер элемента управления **Вкладка,** сделав его достаточно широким для отображения всех страниц или не применяйте длинные заголовки вкладок.

4. Поместите элементы управления на разные страницы.

Вы можете перетащить мышью элементы управления с остальной части формы на вкладку или добавить новые с помощью ленты. В любом случае не забудьте выделить вкладку, которая нужна в первую очередь, и затем добавьте необходимые элементы управления. Даже в **Конструкторе** одновременно отображается только одна страница элемента управления **Вкладка.**

Подсказка

Если элементы управления включены в макет, их невозможно переместить на вкладку. Вместо этого выделите их, щелкните правой кнопкой мыши выделение и выберите команду **Вырезать.** Затем щелкните правой кнопкой мыши на странице вкладки, там где хотите разместить элементы, и выберите команду **Вставить.**

13.2.5. *Переходы по ссылкам*

Ссылки - менее многофункциональные, чем кнопки. В то время как кнопки могут выполнить почти любое действие, ссылки ограничиваются выполнением только двух задач:

- запуском вашего стандартного Web-обозревателя и переходом на конкретный сайт;
- открытием файла (например, документа Word) в программе, которой он создан.

Для создания ссылки выполните следующие действия.

1. В группе ленты **Инструменты конструктора форм | Конструктор -» Элементы управления** щелкните кнопкой мыши пиктограмму **Вставить гиперссылку.**

На экране появится диалоговое окно **Вставка гиперссылки** (Insert Hyperlink) (рис. 13.15) при условии, что у вас включены мастера создания элементов управления *(см. разд. "Вставка элементов управления в форму" ранее в этой главе).* С помощью этого окна вы сможете задать текст для ссылки и пункт назначения, куда ссылка перенесет пользователей, когда они щелкнут ее кнопкой мыши.

2. В левой части окна щелкните кнопкой мыши вариант **файлом, веб-страницей** (Existing File or Web Page).

Можно также использовать вариант **объектом в базе данных** (Object in This Database) для создания ссылки, открывающей другой объект БД, например, форму. Но для этой задачи больше подходят кнопки.

Вариант **электронной почтой** (E-mail Address) выбирается для создания ссылки на адрес электронной почты. При щелчке кнопкой мыши такой ссылки запускается ваша текущая программа работы с электронной почтой и создается новое сообщение с предоставленным вами начальным текстом.

3. В поле ввода **Текст** (Text to display) введите любой текст, который будет отображать ссылка.

Обычно такой текст включает адрес реального Web-сайта (например, **http://www.mycompany.com)** или информационное сообщение (например, "Щелкните здесь

для перехода на Web-сайт моей компании").

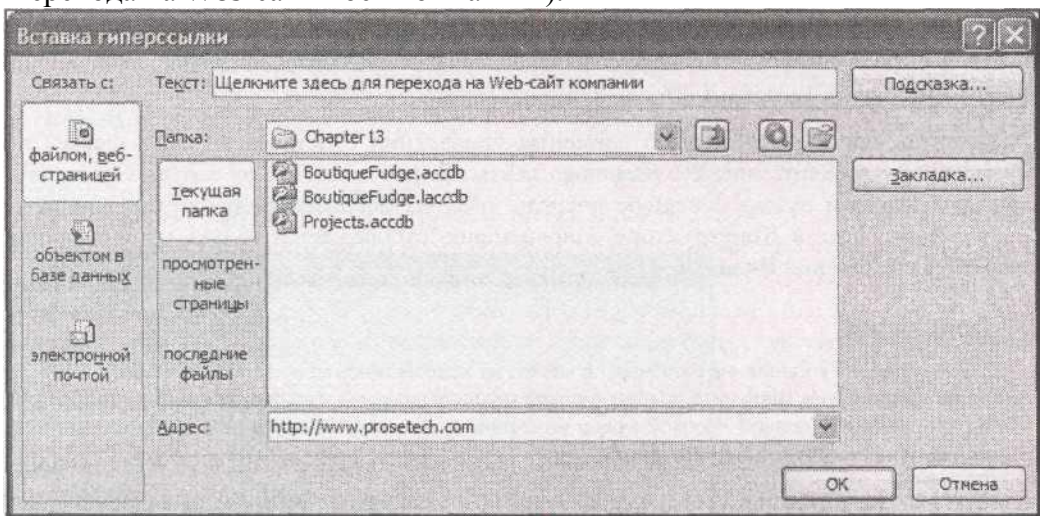


Рис. 13.15. Кто-то собрался создать новую гиперссылку. Она будет выводиться с текстом "Щелкните здесь для перехода на Web-сайт компании" (который вы, конечно, можете редактировать, чтобы сообщить то, что хотите)

4. Если хотите задать пользовательскую всплывающую подсказку для данной гиперссылки, щелкните мышью кнопку **Подсказка** (ScreenTip). Введите свое сообщение и щелкните мышью кнопку ОК.

Как вы уже наверняка знаете, *подсказка* - маленькое окно желтого цвета, содержащее сообщение и открывающееся над гиперссылкой, когда указатель вашей мыши перемещается поверх ссылки. Если не задавать собственную подсказку, программа Access выведет полный путь или URL (Web-адрес).

5. Если вы хотите добавить ссылку на документ, найдите соответствующий файл с помощью просмотра и выделите его. Если вы хотите вставить ссылку на Web-страницу,введите URL в поле **Адрес** (Address).

Если добавляется ссылка на документ, Access задает в поле **Адрес** полный путь к файлу, например, C:\MyDocuments\Resume.doc. Можно ввести этот путь самостоятельно, а если ваша сеть поддерживает, то и путь UNC (Universal Naming Convention, соглашение об универсальных именах), который ссылается на файл, хранящийся на другом компьютере с помощью указания имени компьютера, например, \\SalesComputer\Documents\CompanyPolicy.doc.

Примечание

Вы можете ссылаться на файлы на вашем компьютере или на сетевых дисках. Только помните о том, что при щелчке ссылки кнопкой мыши Access ищет именно в том месте, которое вы задали. Если файл перенесен в другое место или вы открыли БД на другом компьютере, программа Access не сможет найти файл, указанный в ссылке.

6. Щелкните мышью кнопку ОК для вставки гиперссылки.

На форме появится новая гиперссылка. Затем можно переместить ее в любое место формы.

Для применения гиперссылки просто щелкните ее кнопкой мыши. Вы увидите, что как только вы проведете указателем поверх гиперссылки, он изменит свой вид на руку с поднятым указательным пальцем.

13.2.6. *Переходы с помощью списков*

В формах программы Access существуют два вида элементов управления со списками: Список (List box) и **Поле со списком** (Combo box). Разница в том, что **Список** отображает несколько элементов одновременно (в зависимости от величины списка, заданной вами), **Поле** со **списком** выводит одно значение - для того чтобы увидеть список, необходимо щелкнуть мышью направленную вниз стрелку.

Программа Access предоставляет два варианта применения элементов управления со списками:

■ *их можно использовать для редактирования значения поля.* Access автоматически создает

Поле со списком, если для поля определена подстановка (как описано в *разд. "Поиск в связанных таблицах" главы* 5). Это **Поле со списком** действует так же, как список подстановки на листе данных;

- *их можно применять для перехода к нужной вам записи.* В этом случае список отображает значение поля для каждой записи из таблицы. Когда выбирается одно из значений, программа Access переходит к соответствующей записи.

Применение списков для перехода - по-настоящему мощное средство программы Access. Если вы часто ищите запись с помощью одного и того же условия отбора (например, если вы ищите товары по названию или сотрудников по номеру социального обеспечения), этот метод гораздо эффективнее кнопок перехода или фильтрации записей.

Далее описаны действия, необходимые для создания списка, обеспечивающего переходы между записями.

1. Убедитесь в том, что средство **Использовать мастера** для элементов управления (Control Wizard) включено.

Если вы не уверены в этом, проверьте, подсвечена ли кнопка ленты **Инструменты конструктора форм) Конструктор → Элементы управления → Использовать мастера** (Form Design Tools | Design → Controls → Use Control Wizards).

2. В группе ленты **Инструменты конструктора форм | Конструктор → Элементы управления** щелкните кнопкой мыши пиктограмму **Список** или **Поле со списком.**

Оба эти элемента управления действуют одинаково, когда применяются для перехода между записями. Единственное отличие - **Список** требует больше места. Если вы решили использовать этот элемент, поместите его в боковую часть формы. Пользователи чаще применяют элемент управления **Поле со списком** (рис. 13.16).

3. Нарисуйте элемент управления на форме.

Как только вы закончите, на экране появится мастер, чтобы помочь вам настроить список (рис. 13.17). Этот процесс аналогичен мастеру подстановки, который применялся для установки связи таблиц.

4. Выберите переключатель **Поиск записи в форме** и щелкните мышью кнопку **Далее.** Вы узнаете о других вариантах в следующем разделе.

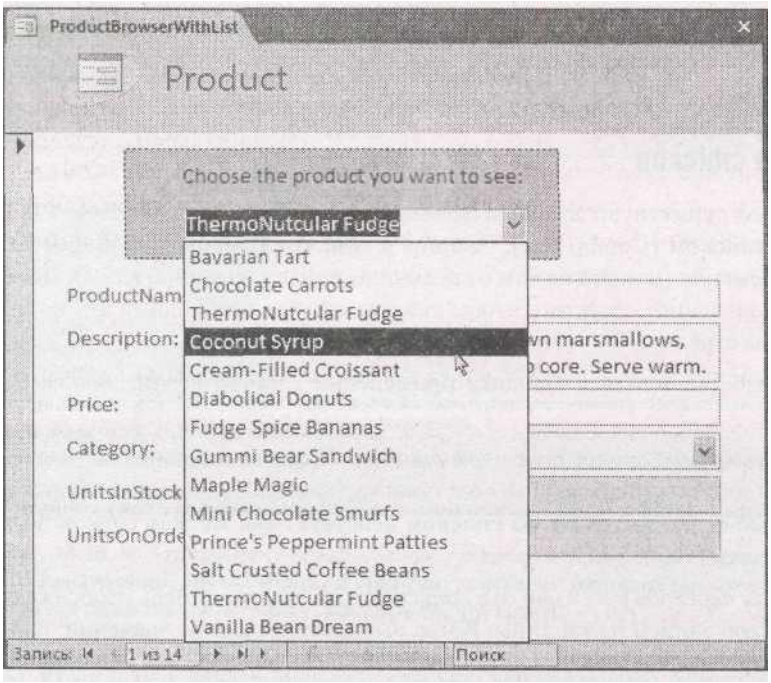


Рис. 13.16. На этой форме список позволяет перейти к нужному товару одним щелчком мыши. Обратите внимание на то, что этот список не заменяет собой элемент управления **Поле** для **ProductName.** Список может применяться для поиска нужной записи, а поле - для изменения названия товара. Конечно, если в данной форме названия товаров никогда не меняются, нет необходимости включать в нее поле **ProductName**

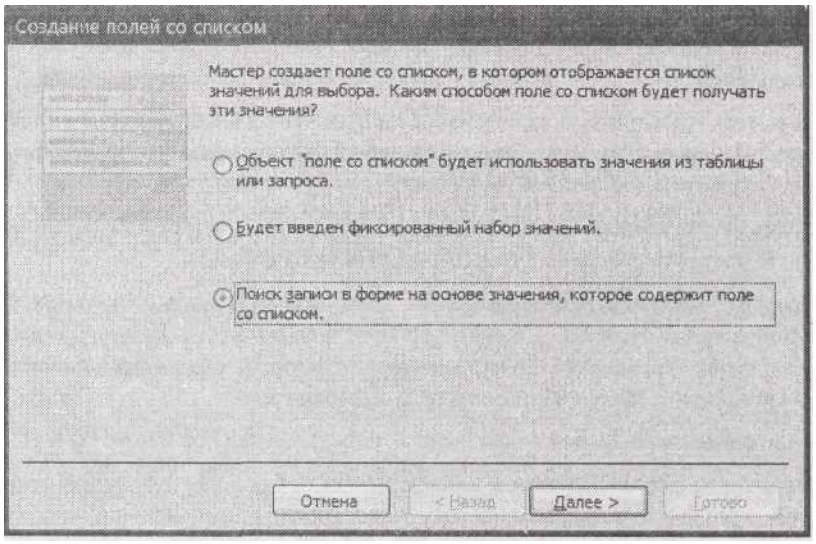


Рис. 13.17. Когда создается поле со списком (или список), мастер создания списка позволяет вам выбрать, применять его для редактирования или для переходов

5. Выберите доле, которое хотите использовать для просмотра, и щелкните мышью кнопку **Далее.**

В примере рис. 13.16 используется поле **ProductName.** Технически список всегда функционирует одинаково - он находит элементы на основе значения уникального первичного ключа *(см. разд. "Первичный ключ "главы 2).* У создаваемого вами списка на самом деле два столбца. В первом хранятся значения первичного ключа, а во втором отображаются значения выбранного вами поля. Но на форме вы не увидите первичного ключа, потому что он скрыт.

Примечание

Этот метод не будет действовать, как следует, если в выбранном поле разрешены дублирующиеся значения. Если создается список, использующий поле **LastName,** вы можете обнаружить несколько значений MacDonald. В этом случае рассмотрите возможность добавления нескольких полей в список просмотра (например, поле **LastName** и поле **FirstName).**

6. Оставьте установленным флажок **Скрыть ключевой столбец (рекомендуется)** и щелкните мышью кнопку **Далее** для продолжения.

Если вы не планируете отображать столбец с первичным ключом, - обычно это так - просто щелкните мышью кнопку **Далее,** чтобы проскочить это окно.

7. Введите название списка.

Оно появится в подписи, расположенной рядом с элементом управления **Список.** Можно использовать что-то вроде "Щелкните кнопкой мыши товар, который хотите просмотреть". Позже подпись можно будет переместить или удалить.

8. Щелкните мышью кнопку **Готово** для создания списка.

Теперь испытайте список. Щелкните правой кнопкой мыши заголовок вкладки и выберите **Режим формы** для перехода к форме. Затем выберите элемент из списка для перехода непосредственно к соответствующей записи.

Примечание

У переходов на базе списка есть одна особенность. Если изменить значение, которое выводится в списке, программа Access не обновит список до тех пор, пока вы не перейдете к другой записи. В предыдущем примере эта особенность приведет к тому, что если вы изменяете название товара, его старое название останется в списке до тех пор, пока вы не пойдете дальше.

13.2.7. *Выполнение действий с помощью кнопок*

Последний элемент управления, который будет рассматриваться, обладает максимальными функциональными возможностями. Кнопки позволяют выполнить почти любое действие, например, открыть новую форму, напечатать отчет или разделаться с прошлогодними налогами.

(Конечно, одни задачи труднее других, но если проявить волю и разобраться с кодом на Visual Basic, возможно почти все.)

Когда на форму вставляется элемент управления **Кнопка,** Access запускает полезный мастер создания кнопки, в котором можно выбрать нужное действие из списка заранее подготовленных вариантов. Затем мастер поможет создать макрос *(см. главу 15),* который сделает все, что потребуется.

Варианты в мастере Создание кнопок предоставляют подробное меню возможностей. Некоторые ярые поклонники программы Access считают, что могут сделать все, что захотят, используя только кнопки и мастер. Другим иногда требуется сделать нечто более оригинальное, в этом случае они вынуждены создавать собственные макросы или писать пользовательский код (задачи, которыми вы будете заниматься в *части V).*

Следующие действия проведут вас через процесс создания кнопки с помощью мастера.

1. В группе ленты **Инструменты конструктора форм | Конструктор → Элементы управления** щелкните кнопкой мыши пиктограмму **Кнопка.**
2. Нарисуйте элемент управления **Кнопка** на форме.

Когда вы закончите, запустится мастер Создание кнопки и сразу включится в работу. Прежде всего, он попросит выбрать действие, которое будет выполняться (рис. 13.18).

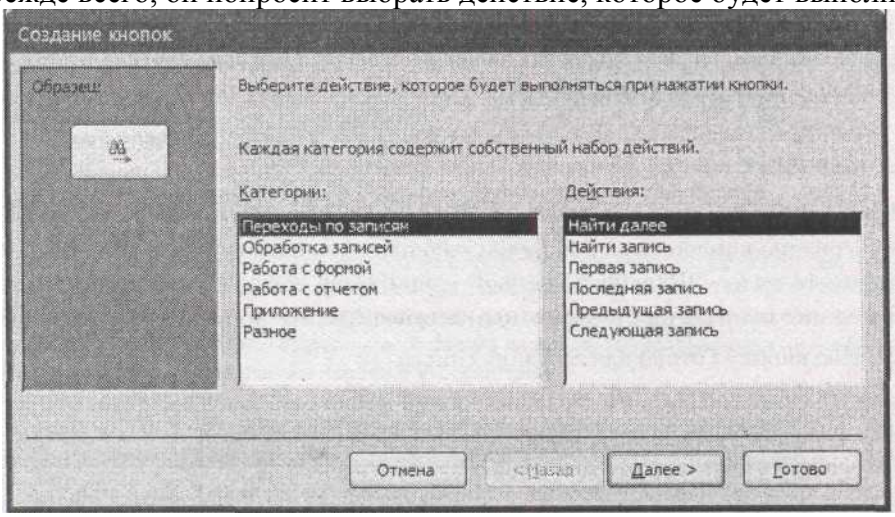


Рис. 13.18. Вы можете заставить форму выполнять шесть категорий действий. После выбора категории (в списке слева) вы увидите список действий в выбранной категории (в списке справа)

3. Выберите действие, которое хотите выполнить.

Большинство действий интуитивно понятно. Далее приведены некоторые самые интересные.

- В категории **Переходы по записям** можно применять команды, такие как **Первая запись, Последняя запись, Следующая запись, Предыдущая запись,** для создания собственных кнопок перехода. Если вы делаете именно это, задайте в свойстве формы **Кнопки перехода** значение *Нет* для скрытия стандартных кнопок.
- В категории **Обработка записей** можно создать новую пустую запись **(Добавить запись)** или обработать текущую запись (например, **Удалить запись, Дублировать запись** и **Печать записи).** Можно даже выбрать немедленное принятие изменений перед переходом к другой записи **(Сохранить запись)** или отмену последнего изменения **(Восстановить запись).**
- В категории **Работа с формой** можно закрыть текущую форму **(Закрыть форму)** или напечатать ее **(Печать текущей формы).** Можно также открыть другую форму **(Открыть форму),** наиболее часто применяемое действие кнопки, поскольку помогает переходить от задачи к задаче.

Примечание

Когда используется действие **Открыть форму,** у вас есть возможность применить фильтр на базе текущей записи. К несчастью, это средство неустойчиво работает. В *главе 14* будет рассмотрен более подробный пример, использующий фильтр для отображения связанных записей.

- В категории **Работа с отчетом** можно работать с другими отчетами, используя такие команды, как **Открыть отчет, Просмотр отчета** и **Печать отчета.** Эти действия помогают перейти от просмотра данных (в форме) к их печати (в отчете).
- В категории **Приложение** вы ограничены одним действием - очевидным **Выйти из приложения.**
- В категории **Разное** вы найдете варианты для запуска отдельного запроса **Выполнение запроса)** или макроса **(Запуск макроса). О** создании макросов рассказывается в *главе 15.*

4. Щелкните мышью кнопку **Далее.**

Следующий шаг зависит от выбранного действия. Некоторым действиям требуется дополнительная информация. Если вы выбрали отображение формы или печать отчета, программа Access попросит выбрать форму или отчет, которые хотите использовать.

После того как дополнительная информация введена, Access попросит ввести текст кнопки и выбрать изображение (рис. 13.19).

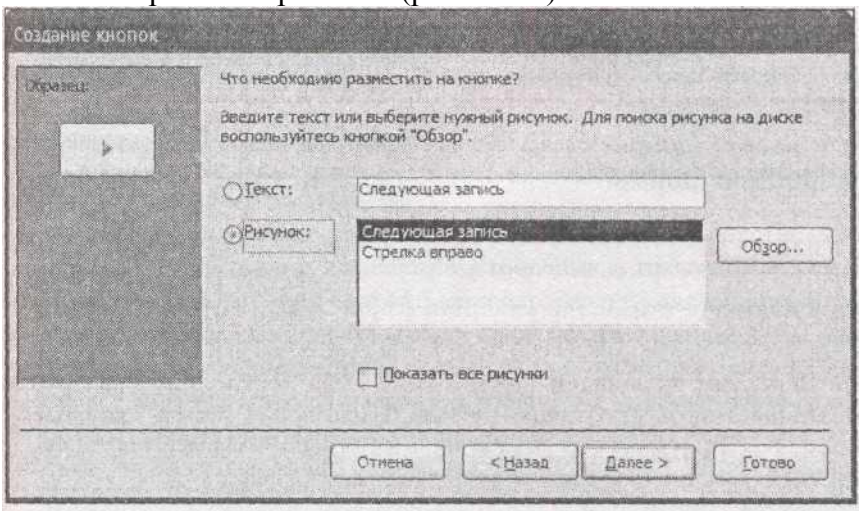


Рис. 13.19. Использовать рисунки соблазнительно, но те, что предлагает Access, определенно старомодные. Большинство фанатов программы Access выбирают кнопки без рисунков. Если вы решили включить рисунок, установите флажок **Показать все рисунки** для того, чтобы увидеть все, что может предложить Access (даже рисунки, которые не подходят для текущего действия), или воспользуйтесь кнопкой **Обзор** для вставки собственного рисунка

Примечание

Любой точечный рисунок (BMP-файл) подходит в качестве рисунка для кнопки, если он достаточно мал, чтобы уместиться на ней. Годятся также пиктограммы, файлы JPEG и GIF.

5. Введите какой-либо текст и выберите рисунок. Затем щелкните мышью кнопку Далее.

Эти данные потом можно изменить, корректируя свойства **Подпись** (Caption) и **Рисунок** (Picture) (которые выводятся на вкладке Макет).

6. Задайте имя кнопки.

Имя выводится в списке **Окна свойств.** Чем лучше выбрано имя, тем легче найти кнопку. А если вы пишете код, работающий с кнопками *(см. главу 16),* чем лучше имя, тем легче другим читать и понимать ваш код.

7. Щелкните мышью кнопку Готово.

Для испытания кнопки перейдите в **Режим формы** и щелкните кнопку мышью.

13.3. *Формы и связанные таблицы*

Как вы узнали в *главе* 5, немногие таблицы по-настоящему независимы. Большинство из них связано с другими таблицами паутиной отношений. Формы могут воспользоваться этими отношениями для отображения связанной информации. Можно применить одну форму для отображения (и редактирования) данных о клиентах и их заказах. Или можно просматривать

товары и их категории. Такой свободы нет на листе данных.

Примечание

Профессиональные разработчики Access применяют запросы с объединением *(см. разд. "Запросы и связанные таблицы" главы 6)* для отображения информации из нескольких таблиц. Но в запросе с операцией объединения нельзя редактировать связанные данные. В хорошо спроектированной форме такого ограничения нет - можно корректировать данные как в родительской, так и в дочерней записях.

13.3.1. *Связи таблиц и простые формы*

Программа Access достаточно сообразительна, чтобы заметить связи во время создания новой формы для родительской таблицы. Для того чтобы понять, что это значит на практике, выделите таблицу-родитель для другой таблицы. Можно воспользоваться таблицей ProductCategories в БД Boutique Fudge, поскольку каждая категория служит родительской записью, связанной с одной или несколькими записями-потомками в таблице Products. (Можно также использовать таблицу Customers, поскольку клиенты связаны с заказами, или таблицу **Orders,** т. к. заказы связаны с конкретными компонентами заказа. Для практической проверки примените БД Boutique Fudge для данной главы, содержащую загружаемые из Интернета данные.)

На рис. 13.20 показано, что произойдет, если выделить таблицу ProductCategories, а затем выбрать на ленте **Создание → Формы → Форма.** Программа Access создает форму, которая выводит ожидаемые записи (категории) и связанные записи из таблицы-потомка (в данном случае товары).

Примечание

Не выбирайте для создания **Разделенную форму** или форму типа Несколько элементов, Access игнорирует связи, когда создаются формы этих типов.

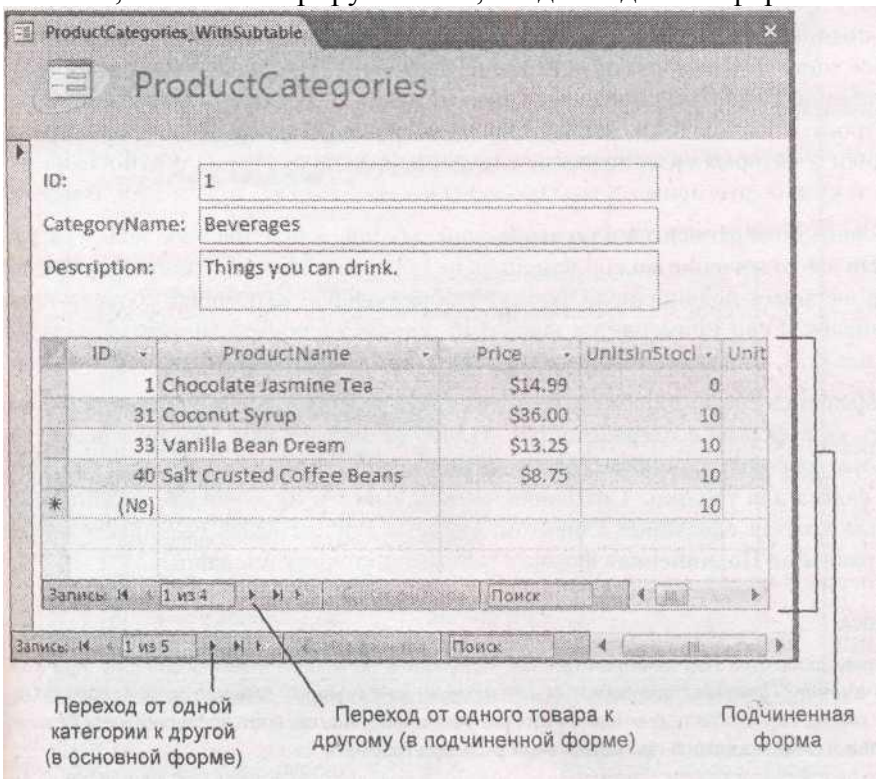


Рис. 13.20. Эта форма позволяет просматривать разные категории товаров. При каждом переходе к новой категории мини-лист данных на форме отображает связанные записи о товарах. С помощью такой формы можно редактировать данные о товарах и о категории товаров

Если таблица - родитель для нескольких потомков, программа Access отображает записи только из одной таблицы. Она выбирает первую найденную связь. Если это не та связь, которая

вам нужна, не беспокойтесь - это легко исправить, если знать, как работает элемент управления **Подчиненная форма.**

13.3.2. *Элемент управления Подчиненная форма*

Программа выводит связанные записи с помощью элемента управления **Подчиненная форма.** Этот элемент можно добавить на любую форму для отображения связанных записей. Он доступен в группе ленты **Инструменты конструктора форм | Конструктор → Элементы управления** наряду с другими элементами управления. Если добавить его вручную, Access попросит выбрать таблицу, которую следует отображать.

То, что отображает подчиненная форма, определяется тремя свойствами. Первое свойство **Объект-источник** (Source Object) задает объект БД, имеющий связанные записи. Можно выбрать существующие таблицу, запрос или форму.

Следующие два свойства - **Основные поля** (Link Master Fields) и Подчиненные поля (Link Child Fields) - позволяют определить способ связи двух таблиц. Основное поле - это поле в форме, а подчиненное поле - это поле в объекте-источнике. В примере с категориями товаров основное поле - ID (в таблице **ProductCategories)** и подчиненное поле - поле **ProductID** (в таблице **Products).** После того как эта связь определена, программа Access знает, как фильтровать подчиненную форму. Она просматривает основное поле и отображает только те записи, у которых то же значение в подчиненном поле. На рис. 13.20 Access отображает товары текущей категории.

Обычно основное поле относится к родительской таблице, а подчиненное поле - к таблице-потомку. Но это отношение можно изменить на обратное. Можно создать форму с товарами, которая включает подчиненную форму, отображающую категорию, соответствующую каждому товару. Если применяется подобный подход, подчиненная форма включает только одну запись (т. к. только один родитель связан с каждой записью).

Теперь разобравшись с принципами работы элемента управления **Подчиненная форма,** можно вставлять ее в формы с одержимостью. Ничто не мешает вам добавить несколько подчиненных форм для одновременного отображения целой коллекции связанных данных. Если создается форма для таблицы **Customers,** можно вывести на экран две подчиненные формы - одну для заказов, сделанных клиентом, а другую для платежей. Вам просто нужны два элемента управления **Подчиненная форма** с разными источниками данных.

Подсказка

Если в форму включена подчиненная форма, подумайте об использовании средств привязки, описанных в *разд. "Привязка: автоматическое изменение размеров элементов управления" ранее в этой главе,* так чтобы подчиненная форма увеличивалась за счет доступного свободного пространства, появляющегося при изменении размера формы.

13.3.3. *Создание настроенных подчиненных форм*

Если в свойстве **Объект-источник** задана таблица или запрос, программа Access всегда отображает связанные записи на мини-листе данных. Если вы намерены настраивать каждый новый фрагмент формы, возможно, вас не устроит такое отображение. Интересно, что программа Access позволяет управлять способом отображения связанных записей, если потрудиться немного больше.

Хитрость заключается в задании в свойстве **Объект-источник** формы, которую хотите отобразить в элементе управления **Подчиненная форма.** Форма появится в режиме вывода по умолчанию в соответствии со значением в свойстве **Режим по умолчанию.** Можно выводить связанные записи в табличном макете или макете в столбец. На рис. 13.21 показан пример.

Возможно, для получения эффекта, которого вы стараетесь добиться, уже есть подходящая форма, лежащая под рукой и готовая к применению. Если проектируется форма для таблицы **ProductCategories,** можно использовать форму, созданную для таблицы **Products** в элементе управления **Подчиненная форма.** Но часто хочется иметь отдельную форму, чтобы настраивать ее так, как заблагорассудится. В примере с таблицей товаров может появиться желание в подчиненной форме отображать товары совершенно иначе, чем в их собственной

специализированной форме. Помимо всего прочего, когда применяется элемент управления **Подчиненная форма,** у вас гораздо меньше места на ней, поэтому можно выбрать более компактный формат и вовсе пропустить заголовок формы.

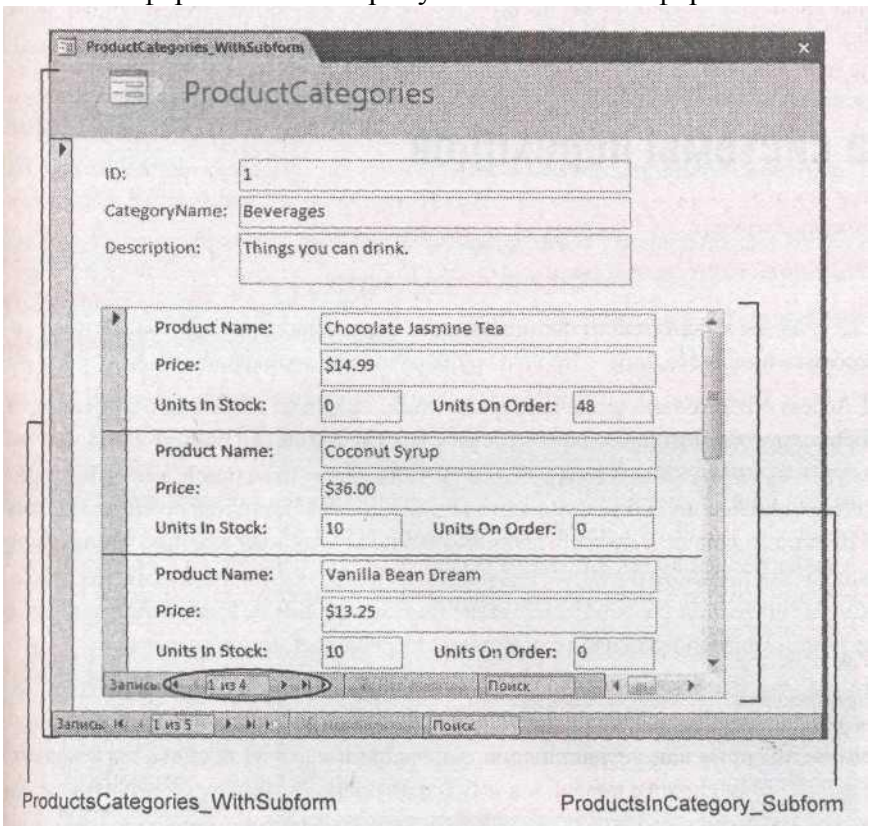


Рис.13.21. Благодаря магии подчиненных форм в этом окне на самом деле одновременно отображаются две формы: **ProductCategories_WithSubform** и **ProductslnCategory_Subform.**

В данном примере подчиненная форма использует макет **Ленточная форма** и, таким образом, отображает список всех отобранных товаров. Для отображения остальных товаров необходимо использовать второй набор кнопок перехода (обведены)

Подсказка

Если вы решили создать специализированную форму, использующую элемент управления **Подчиненная форма,** отразите ее роль в названии. Название **ProductslnCategory Subform** (Подчиненная форма для товаров из данной категории) подразумевает форму, спроектированную для применения в качестве подчиненной формы.

Предпринимайте разные попытки, поскольку не существует способа разместить все на небольшом пространстве подчиненной формы, включенной в форму. В этом случае у вас есть две возможности: попытаться создать более компактную подчиненную форму или использовать две отдельные формы. В *главе 14* показано, как применять переходы и фильтр для отображения связанных записей в отдельной форме.

14. Глава 14. Создание системы переходов

На протяжении 13 глав вы накапливали составляющие первоклассной БД. Но если нет хорошего способа собрать их вместе, они - просто груда разрозненных фрагментов.

В лучшие БД Access обязательно включено какое-либо средство перехода пользователей от одной части БД к другой. Главная цель - сделать удобнее и легче работу с БД. Вместо того чтобы заставлять вас выискивать нужный объект в области переходов, такие БД начинают с формы, содержащей меню того или иного вида и позволяющей сформировать свой способ перехода от одной задачи к другой щелчком мышью удобных кнопок. Такого рода проект БД - находка для пользователей, не знакомых с хитростями и уловками Access. Если система переходов тщательно разработана, таким пользователям не нужно знать ничего о программе Access - они могут начать вводить данные, не осваивая ничего нового.

Вы уже знаете все, что необходимо для создания первоклассной системы переходов. Теперь вам нужен новый взгляд на БД, такой, чтобы БД могли (и стали) вести себя как обычные Windows-программы, а не как устрашающие форты данных. В этой главе вы узнаете о разных способах добавления средств переходов D БД, настроенных дружески по отношению к пользователям. Вы научитесь создавать кнопочные формы (формы, перенаправляющие пользователей к другим формам), стартовые формы, появляющиеся при первом открытии БД, и отображать связанную информацию в отдельных формах. Но, прежде всего, начнем с более близкого знакомства с областью переходов, чтобы научиться управлять переходами, не создавая ничего нового.

14.1. *Освоение области переходов*

Область переходов была описана в *главе 1,* и с тех пор вы пользовались ею для перемещения по объектам БД. Но по мере роста БД область переходов становится все более загруженной, При наличии примерно 20 объектов БД, точное число зависит от размера вашего монитора, все они не помещаются на экране одновременно. В результате приходится пользоваться прокруткой сверху вниз для поиска нужного объекта, что может стать причиной мучительной боли в лучевом суставе.

Один из методов борьбы с этим затруднением - проектирование собственных кнопочных форм, позволяющих перемещаться в БД. Но прежде чем перейти к этому решению, стоит рассмотреть некоторые средства, встроенные непосредственно в область переходов. Они могут решить проблему с меньшими затратами.

14.1.1. *Настройка списка области переходов*

Для начала рассмотрим применение фильтрации для сокращения объема информации, отображаемой в области переходов. Возможно, у вас есть БД с тремя десятками объектов, из которых вы регулярно пользуетесь только 10. В этом случае незачем выводить на экран объекты, которые не пользуются.

По существу программа Access предоставляет два решения, касающиеся области переходов:

■ можно выбрать способ упорядочивания объектов в области переходов. Этот процесс называют *систематизацией* объектов БД;

■ можно не отображать некоторые объекты. Этот процесс называют *фильтрацией* объектов БД.

Смутить может то, что выбор обоих вариантов делается в одном и том же меню. Для того чтобы открыть это меню, щелкните направленную вниз стрелку в зоне заголовка области переходов. На рис. 14.1 показано это меню.

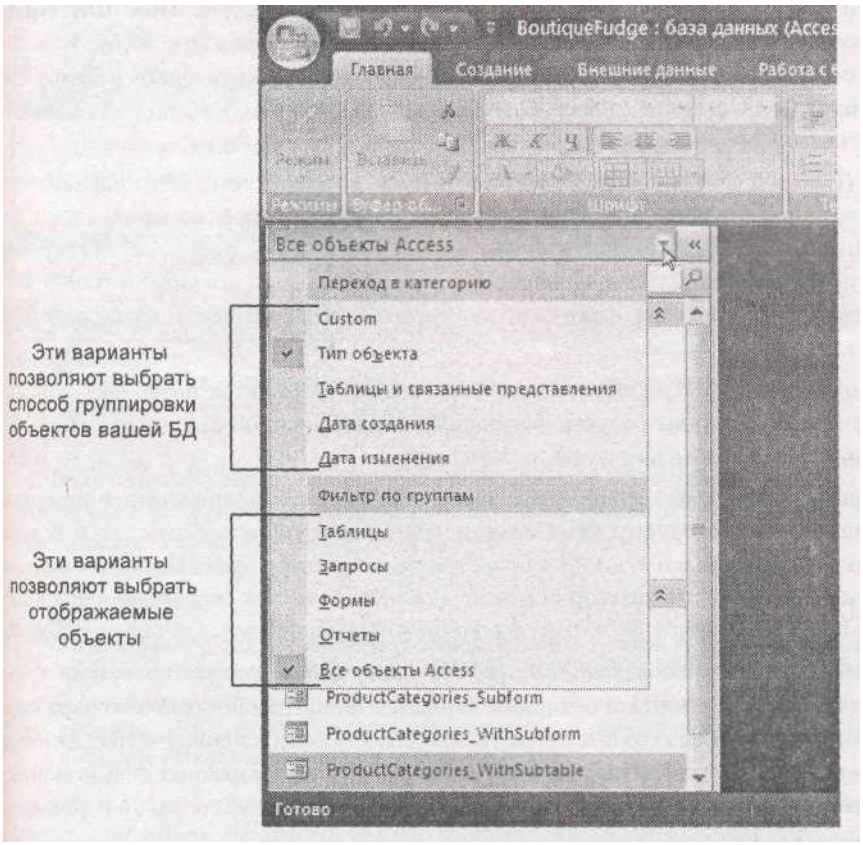


Рис. 14.1. Когда вы готовы сообщить программе Access о способе организации объектов в области переходов, выделите верхнюю порцию меню (Переход в категорию). Текущий вариант - Тип объекта - группирует таблицы, запросы, формы и отчеты в отдельные разделы. Для выбора отображаемых объектов выберите вариант в нижней части меню (Фильтр по группам)

Систематизировать область переходов можно пятью способами.

■ **Таблицы и связанные представления** (Tables and Related Views) группируют объекты БД на основе используемой ими таблицы. Если созданы две формы, три запроса и отчет для таблицы **Students,** вы увидите все эти объекты собранными вместе, в одной группе (под заголовком "Students"). Связанная с этим вариантом проблема заключается в трудности различения объектов разных типов, особенно если у них схожие имена. Придется внимательно рассматривать пиктограмму, чтобы определить, является ли данный объект формой, отчетом или чем-то еще. **Таблицы и связанные представления -** вариант систематизации, с которого начинает программа Access.

Подсказка

Многие объекты БД используют несколько таблиц. Если создается запрос, применяющий операцию объединения для отображения товаров с информацией об их категории, в этом запросе используются таблицы **Products** и **ProductCategories.** В режиме **Таблицы и связанные представления** вы увидите этот запрос в двух местах - под заголовком **Products** и под заголовком **ProductCategories.**

Когда выбран режим **Таблицы и связанные представления,** раздел меню **Фильтр** по **группам** включает все таблицы БД. Если выбрать конкретную таблицу, то будут выводиться только объекты, связанные с этой таблицей. Можно также выбрать вариант **Несвязанные объекты** (Unrelated Objects), чтобы увидеть любые объекты определенных категорий, не связанные ни с одной таблицей, например, файлы с кодом.

■ **Тип объекта** (Object Type) группирует объекты БД на основе их типа. Этот вариант четко отделяет таблицы от форм, отчетов и объектов других типов. Многие специалисты Access предпочитают этот режим при просмотре завершенной БД, поскольку он укрощает самую неуправляемую БД. Данный вариант очень хорош, если вы забыли точное имя нужного объекта. Если известно, что нужно напечатать отчет со списком классов, можно сразу перейти в группу **Отчеты.**

Когда применяется режим **Тип объекта,** список вариантов фильтра позволяет увидеть объекты одного типа. Если вы создали формы для каждой нужной задачи, выберите вариант **Формы,** чтобы увидеть на экране только формы.

■ **Дата создания** (Created Date) группирует объекты БД на основе времени их создания. Программа Access создает группу для **Сегодня,** группирует по остальным дням недели **(Понедельник, Воскресенье** и т. д.) и по более длительным периодам **(Прошлая** неделя, **Две недели назад** и т. д.). Эта категория может не использоваться регулярно, поскольку по истечении времени объекты переходят из одной группы в другую. Но она очень удобна для отображения последних результатов работы.

Когда применяется категория **Дата создания,** варианты фильтрации позволяют отобрать те объекты, которые созданы сегодня, вчера, на прошлой неделе, в прошлом месяце и т. д. (как показано на рис. 14.2). Если вы помните, когда были созданы важные форма или отчет, но забыли их имена, такая возможность сэкономит время.

■ **Дата изменения** (Modified Date) действует так же, как вариант **Дата создания** за исключением того, что этот вариант позволяет отобрать объекты БД, измененные в последнее время. Он удобен, если вы хотите игнорировать таблицы и другие объекты, которые используются редко.

Когда применяется группировка по дате изменения, доступны те же варианты фильтрации, что и для варианта **Дата создания.**

Рис. **14.2.** При выборе группировки **Дата создания** отображаются группы, в которых объекты упорядочены на основе даты создания

■ Вариант **Custom** позволяет выбрать точно, какие объекты отображать, а какие скрыть. Он хорош, если у вас есть определенные, часто используемые объекты и объекты, которые следует скрыть.

Подсказка

Можно быстро задать условие отбора. Щелкните правой кнопкой мыши заголовок группы и выберите **Показать только *[Имя_группы].*** Для отображения только таблиц при группировке **Тип объекта** щелкните правой кнопкой мыши группу **Таблицы** и выберите **Показать только таблицы.** Для удаления фильтра снова щелкните правой кнопкой мыши область переходов и выберите **Показать все группы** (Show All Groups).

Когда в области переходов применяется фильтр, программа Access полностью скрывает то,

что вы не хотите видеть. Но как вы, вероятно, уже знаете, Access предоставляет еще один вариант. Можно щелкнуть кнопкой мыши *стрелки сворачивания* (collapse arrows), расположенные рядом с конкретным разделом для сворачивания его до отображаемого заголовка (рис. 14.3). Когда раздел вам понадобится, вы сможете снова развернуть его.

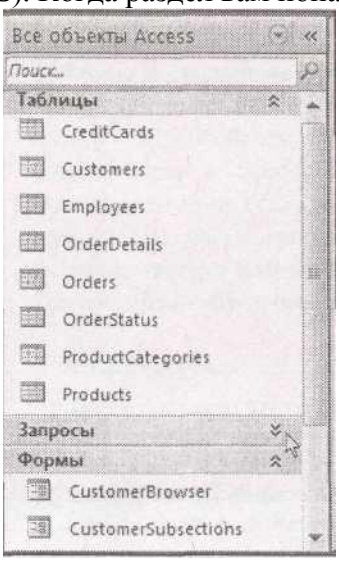


Рис. 14.3. Щелкните кнопкой мыши стрелки сворачивания для быстрого скрытия объектов в конкретной группе. В данном примере аккуратно скрыта группа запросов

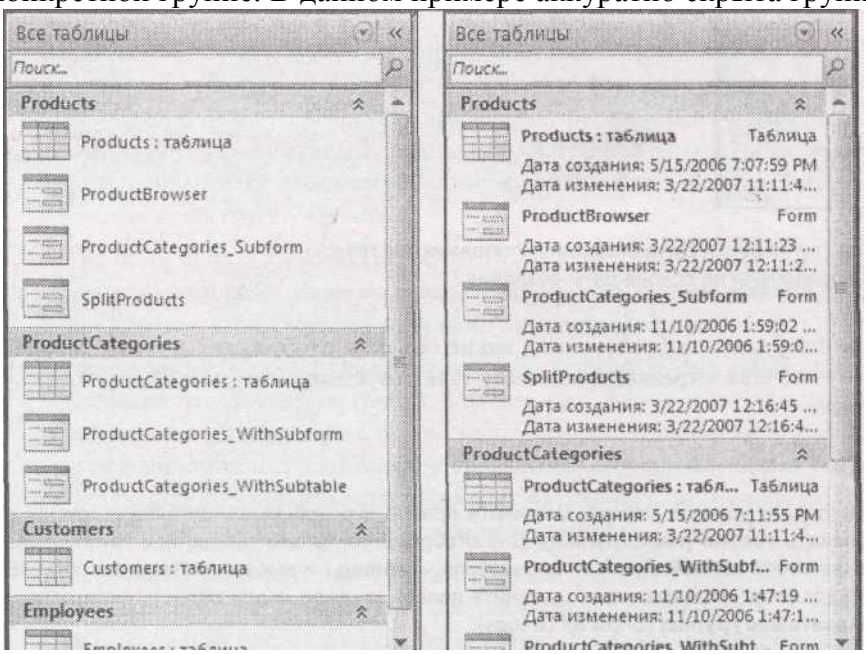


Рис. 14.4. До сих пор вы пользовались режимом списка в области переходов, обеспечивающим наиболее компактное представление. Но программа Access позволяет применять режим немного более крупных значков *(слева)* или режим отображения с подробностями, включающий сведения о датах создания и изменения объекта *(справа)*

Малоизвестная или недооцененная возможность.
Варианты сортировки и просмотра в области переходов

У области переходов есть много тщательно спрятанных параметров, которые можно настраивать. Например, если вам не нравится порядок отображения элементов в каждой группе, у вас есть несколько вариантов их сортировки. Для того чтобы увидеть все возможные варианты, щелкните правой кнопкой мыши полосу заголовка области переходов и выберите подменю **Сортировка** (Sort By).

Как вы увидите, можно применить сортировку по возрастанию или убыванию к любому из следующих критериев:

- **Имя** (Name) сортирует в соответствии с именем объекта БД;
- **Тип** (Type) сортирует в соответствии с типом объекта (форма, отчет, таблица и т. д.). Этот вариант не оказывает влияния, если данные уже сгруппированы по типу объекта;

• **Дата создания** и **Дата изменения** сортируют так, что более поздние или более ранние объекты выводятся первыми.

Можно также изменить внешний вид области переходов, щелкнув правой кнопкой мыши заголовок области переходов и выбрав вариант из меню **Просмотр** (View By). На рис. 14.4 сравниваются разные параметры отображения.

Улучшенная фильтрация

У системы фильтрации есть одно ограничение - она позволяет в каждый момент времени выбрать только одну категорию. Если выбрать **Таблицы и связанные представления,** можно отфильтровать список объектов, связанных с одной таблицей. Но нельзя включить в условие отбора две или несколько групп таблиц. Аналогично, если выбирается **Тип объекта,** можно отобразить все формы или все отчеты в вашей БД, но нельзя показать формы и отчеты, не включив всего остального (хотя прием сворачивания, показанный на рис. 14.3, помогает высвободить большую часть пространства).

Это ограничение можно легко обойти. Для более полного управления условиями отбора выполните следующие действия.

1. Щелкните правой кнопкой мыши полосу заголовка области переходов и затем выберите команду **Параметры переходов** (Navigation Options).

На экране появится одноименное диалоговое окно (рис. 14.5).

2. Выберите категорию, которую хотите настраивать - либо **Таблицы и связанные представления,** либо **Тип объекта.**

Список справа отобразит все группы в этой категории.

3. Если вы не хотите отображать группу в списке области переходов, сбросьте флажок рядом с ней.

Если вы хотите показывать в области переходов только отчеты и формы, выберите категорию **Тип объекта** и сбросьте флажки рядом с группами **Таблицы, Запросы, Макросы** и **Модули.**

4. Если вы настраиваете категорию **Таблицы и связанные представления,** можно изменить порядок групп, как показано на рис. 14.6.

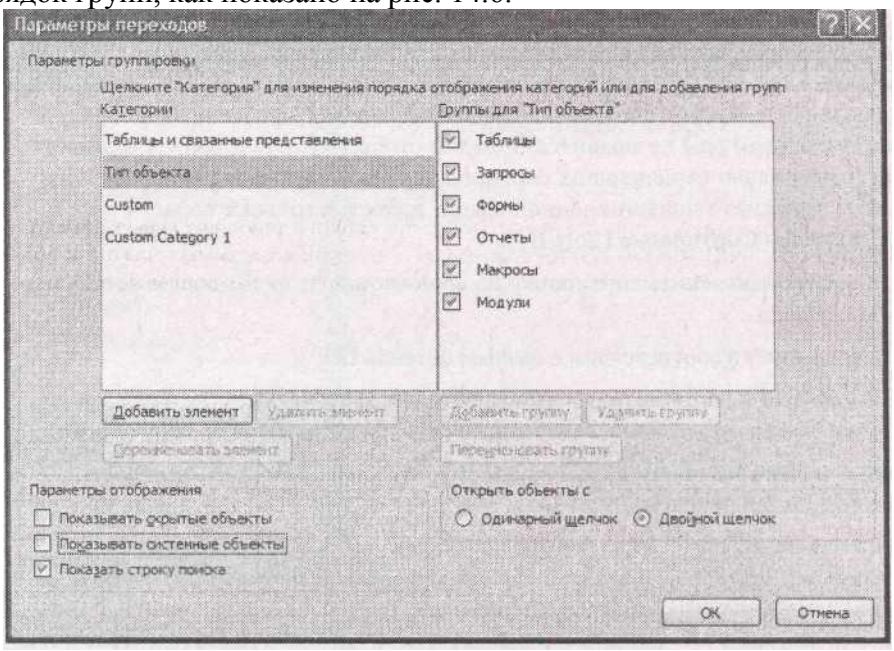


Рис. 14.5. Список слева отображает разные варианты систематизации области переходов.

Вы не увидите варианты **Дата создания** и **Дата изменения,** поскольку их невозможно настроить.

Список справа отображает группы в выбранной в данный момент категории

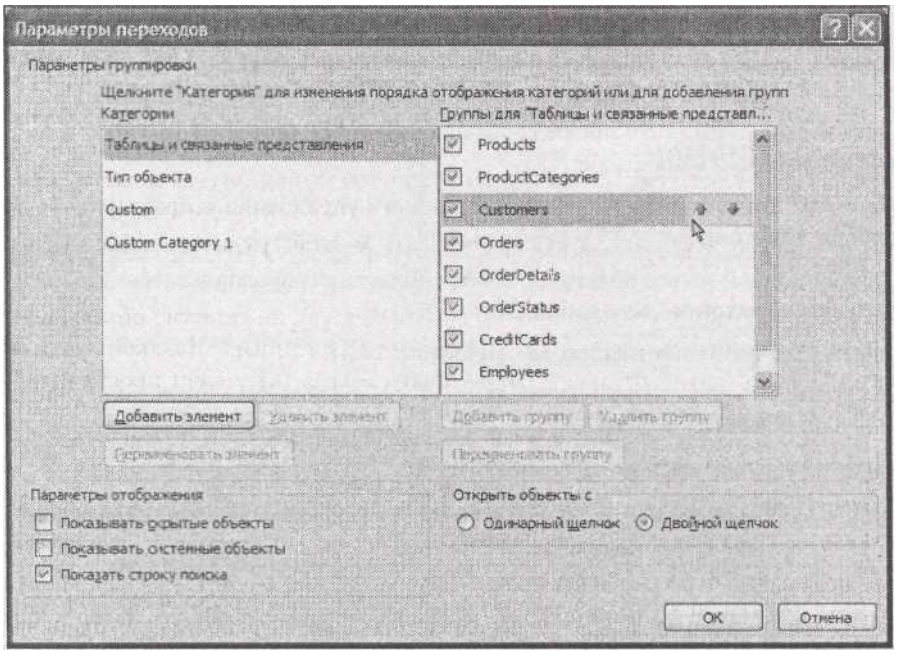


Рис. 14.6. Для перемещения группы просто выделите ее. На экране рядом с группой появляются стрелки, направленные вверх и вниз. Можно щелкнуть эти кнопки мышью для переноса группы вверх или вниз

Единственный элемент, который нельзя переместить, - **Несвязанные объекты,** всегда отображаемый в конце списка. И вообще нельзя изменить порядок групп в категории **Тип объекта.**

6. Щелкните мышью кнопку ОК для закрытия окна.

Подсказка

Многие БД в целом становятся гораздо понятнее, когда скрыты дополнительные объекты. Если вы снабдили вашу БД полным комплектом форм и отчетов, возможно, только эти объекты и стоит отображать. Почему не пойти дальше и не скрыть низкоуровневые таблицы, запросы и код?

14.1.2. *Скрытие объектов*

Скрытие групп, которые вы не хотите видеть, - это замечательно, но что если есть только один объект, который нужно скрыть от глаз? Возможно, необходимы гарантии того, что пользователи, применяющие вашу БД, не будут сбиты с толку несколькими потенциально опасными запросами на обновление *(см. главу 8),* которые на самом деле они не должны использовать. Нет проблем. Просто щелкните правой кнопкой мыши запрос в области переходов и выберите команду **Скрыть в этой группе** (Hide in this Group).

Примечание

Когда скрывается объект, он исчезает в текущем режиме отображения, в текущей группе. (Напоминаю, что в режиме **Таблицы и связанные представления** некоторые объекты могут появляться в нескольких группах.) Если нужно скрыть объект повсеместно, необходимо отследить его в каждой группе и скрыть.

Для отображения скрытого объекта сначала необходимо настроить область переходов так, чтобы она выводила на экран скрытые объекты. Для этого щелкните правой кнопкой мыши полосу заголовка, выберите команду **Параметры переходов** и установите флажок **Показывать скрытые объекты,** а затем нажмите мышью кнопку ОК. Теперь скрытые объекты отображаются в области переходов, но они слегка обесцвечены, поэтому вы можете отличить их от других, нескрытых объектов. Для того чтобы сделать объект видимым, щелкните его правой кнопкой мыши и выберите команду **Показать в этой группе** (Unhide in this Group).

Все эти методы - фильтрация, создание пользовательских групп, скрытие объектов - разработаны для облегчения использования БД. Но они не обеспечивают никакой защиты (пользователь, который действительно хочет использовать объект БД, может просто измерить параметры области переходов и получить доступ к нему.)

Примечание

В *разд. "Подготовка вашей базы данных" главы 18* вы узнаете, как разделить БД на отдельные файлы - наилучший способ сохранения некоторых объектов в стороне от преступных рук. Но независимо от того, что вы делаете, программу Access нельзя назвать пуленепробиваемой. Access разработана в расчете на интуицию, податливость и легкость ее применении. В отличие от серверных БД, таких как SQL Server *(см. разд. "Нужно ли переходить на SQL Server?" главы 20),* она не предназначена для блокирования злоумышленников, захвативших файлы вашей БД.

14.1.3. *Использование групп Custom*

Обычные пользователи не оперируют таблицами и объектами БД. Они думают о задачах, которые следует выполнить. Ни один из готовых вариантов группировки не соответствует такому подходу. К счастью, можно сформировать собственные группы, соответствующие ему. Далее описано, как это сделать.

1. Щелкните кнопкой мыши направленную вниз стрелку в зоне заголовка области переходов и выберите команду **Custom** (пользовательская группировка).

В новой БД вы начинаете с двух групп в режиме **Custom.** Первая, **Custom Group 1,** пуста. Вторая, **Неназначенные объекты** (Unassigned Objects), содержит все объекты вашей БД.

2. Можно создать новую группу и перенести в нее объект за один шаг. Для этого щелкните правой кнопкой мыши объект, который хотите переместить (в разделе **Неназначенные объекты)** и выберите **Добавить в группу → Создать группу** (Add To Group → New Group). Введите имя группы и нажмите клавишу <Enter>. На рис. 14.7 показан результат.

Повторите этот пункт для создания всех нужных групп. Если нужно перенести объект в существующую группу, щелкните его правой кнопкой мыши, выберите команду **Добавить в группу** (Add To Group) и затем укажите имя соответствующей группы.

Подсказка

Для ускорения работы просто перетащите мышью объекты в соответствующие группы.

Рис. 14.7. Зачастую удобно создавать группы с именами, отражающими типы задач, как в данной БД

Ваши группы можно переименовывать, удалять и реорганизовывать. Легчайший способ для этого - применение диалогового окна **Параметры переходов.** Щелкните правой кнопкой мыши заголовок области переходов и выберите **Параметры переходов.**

Диалоговое окно **Параметры переходов** позволяет выполнять несколько полезных задач, относящихся к группам:

- задать другое имя, выделив группу и щелкнув мышью кнопку **Переименовать группу;**
- удалить группу, просто выделив ее и щелкнув мышью кнопку **Удалить группу;**
- добавить первоначально пустую группу, щелкнув мышью кнопку **Добавить группу;**
- переупорядочить группы, щелкнув кнопкой мыши одну из них и используя появившиеся пиктограммы стрелок для перемещения группы вверх или вниз;
- переместить вашу пользовательскую категорию в другое место в списке, что повлияет на

способ отображения меню, появляющегося на экране при щелчке мышью направленной вниз стрелки в области переходов;

- скрыть группу (временно или на длительный срок), сбросив флажок, расположенный рядом с именем группы.

Единственное, чего нельзя сделать в диалоговом окне **Параметры переходов,** - изменить состав группы. (Для этого нужно мышью перетащить объекты в области переходов, как описано в пункте 2.)

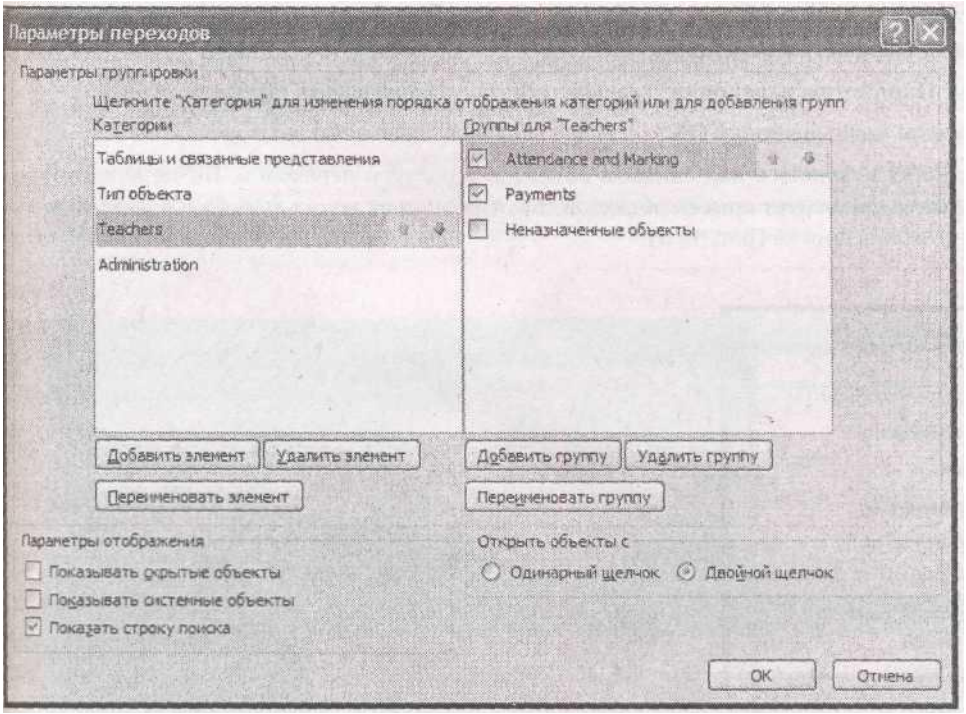


Рис. 14.8. Причина, по которой может возникнуть необходимость разных представлений, - использование вашей БД разными людьми. В примере с БД Cacophone Studios администрация просматривает формы для создания классов и регистрации студентов (с помощью представления Administration), в то время как преподаватели должны печатать ведомости посещаемости и создавать домашние задания студентам (с помощью представления **Teachers,** выделенного в данном примере). Как видно, это представление содержит группы **Attendance and Marking** (посещаемость и успеваемость) и **Payments** (платежи). У каждой из них свой набор форм и отчетов

Можно также изменить имя представления, содержащего все ваши группы. Первоначально эта категория называется **Custom,** но можно задать более информативное имя, выбрав категорию в диалоговом окне **Параметры переходов** и щелкнув мышью кнопку **Переименовать элемент.** Если хочется большего, можно создать несколько режимов отображения пользовательских категорий верхнего уровня. Щелкните мышью кнопку **Добавить элемент,** чтобы вставить новую категорию, и кнопку **Удалить элемент** для удаления категории. На рис. 14.8 приведен пример с несколькими пользовательскими категориями.

3. Когда завершите внесение изменений, щелкните мышью кнопку **ОК.**

14.1.4. *Поиск в списке области переходов*

Если вам не нравится, когда что-то исчезает из вида, возможно, в этом случае придется выводить в области переходов громоздкий список объектов. Но у программы Access есть удобный инструмент, позволяющий экономить время на прокрутке, - поле поиска, которое помогает перейти к объекту почти мгновенно при условии, что вы знаете его имя.

Для отображения поля поиска выполните следующие действия.

1. Щелкните правой кнопкой мыши полосу заголовка области переходов и выберите команду **Параметры переходов.**
2. В окне **Параметры переходов** установите флажок **Показывать строку поиска.**
3. Щелкните мышью кнопку ОК.

Поле поиска появится над списком объектов в области переходов. По мере ввода программа Access фильтрует список объектов так, чтобы он включал только объекты, соответствующие строке поиска (рис. 14.9).

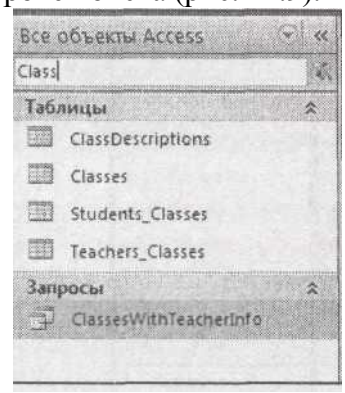


Рис. 14.9. Объекты, соответствующие строке поиска, содержат введенный текст. Если ввести "Class", вы увидите такие объекты, как **Classes** и **Students_Classes**

14.2. *Построение форм со средствами автоматического перехода*

Область переходов - неоценимое средство перемещения в вашей БД, но оно устраивает не всех. Пользователи, никогда раньше не работавшие в программе Access, могут счесть его несколько замысловатым и лишенным защиты от произвольного изменения параметров переходов (и открытия объектов, которые открывать не следует).

Для получения дополнительных средств управления и создания дружественной внешней оболочки многие специалисты Access встраивают средства переходов в свои формы (и иногда в отчеты). В конце концов, форма предоставляет практически неограниченные возможности настройки. Можно вставить абзац текста, добавить яркий выигрышный фон и эмблему компании и ограничить приводящие в замешательство варианты несколькими простыми дружественными кнопками.

Если вы твердо решили использовать формы для переходов, прежде всего, нужно выбрать тип формы для проектирования. Программа Access предлагает несколько вариантов, включая встроенную поддержку объекта, именуемого кнопочной формой.

14.2.1. *Создание кнопочной формы*

Кнопочная форма - это форма, единственная цель которой перенаправить вас к другим формам (обычно при щелчке мышью кнопки формы). Она - своего рода главное меню вашей БД. Такая форма - одновременно и отправная точка, и центр действий. Типичная кнопочная форма содержит набор кнопок, направляющих в **разные** места.

На рис. **14.10** показана разновидность кнопочной формы, создаваемая программой Access.

Примечание

Созданная кнопочная форма подобна форме любого другого типа, поэтому после ее создания можно использовать навыки, приобретенные в последних двух главах, для придания вашей кнопочной форме внешней привлекательности.

Рис. 14.10. Эта кнопочная форма программы Access предоставляет доступ к пяти разным формам одним щелчком кнопки мыши. Достоинство кнопочных форм Access заключается в возможности построения собственного варианта за считанные секунды. К недостаткам можно отнести несколько старомодный внешний вид, заставляющий впечатлительных пользователей проектировать собственные кнопочные формы с нуля

Для автоматического создания кнопочной формы следует применять Мастер форм (Form wizard). Далее описывается, как он работает.

1. Выберите на ленте **Работа с базами данных → Работа с базами данных → Диспетчер кнопочных форм** (Database Tools → Database Tools → Switchboard Manager).

Когда вы первый раз в БД щелкните мышью эту кнопку, программа Access сообщит о том, что не может найти кнопочную форму и предложит ее создать. Щелкните кнопку Да для продолжения работы Диспетчера кнопочных форм (рис. 14.11).

Если кнопочная форма уже существует, переходите к пункту 2, в котором можно редактировать текущую кнопочную форму.

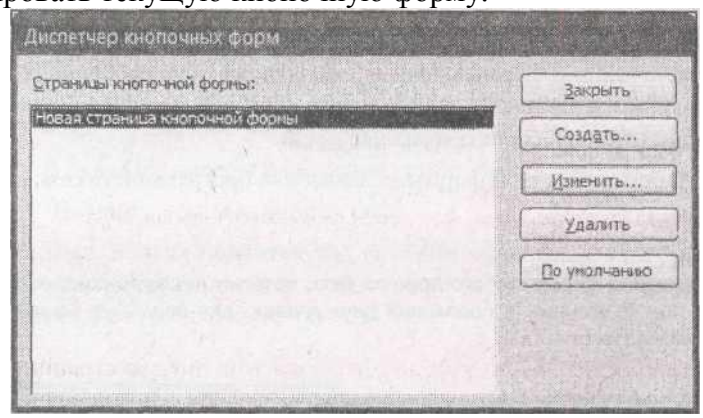


Рис. 14.11. Диспетчер кнопочных форм выводит на экран список страниц. Каждая страница - отдельная часть меню кнопочной формы. У самых простых кнопочных форм только одна страница, что означает поддержку формой одного уровня глубины и выполнение каждой кнопкой полезного действия (например, открытие формы или отчета)

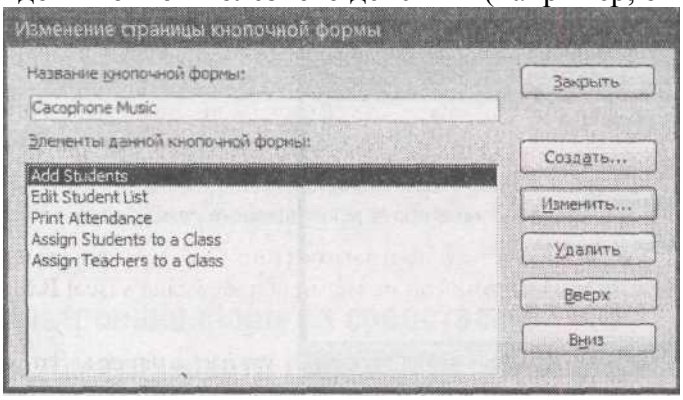


Рис. 14.12. Окно **Изменение страницы кнопочной формы** позволяет создавать команды меню, удалять те, которые больше не нужны, изменять порядок их следования (этот порядок определяет порядок команд на кнопочной форме)

2. Щелкните мышью кнопку **Изменить** для редактирования страницы кнопочной формы.

На экране появляется окно **Изменение страницы кнопочной формы** (рис. 14.12). Именно здесь определяются реальные команды меню.

3. Для создания новой команды меню щелкните мышью кнопку **Создать.**

На экран выводится окно **Изменение элемента кнопочной формы** (рис. 14.13). Для создания команды меню необходимо предоставить две порции информации: текст, появляющийся на форме, и команду, которую должна выполнить программа Access, когда вы щелкните мышью кнопку.

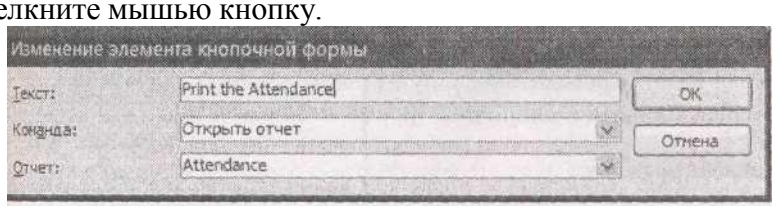


Рис. 14.13. Эта команда запускает отчет **Attendance** (посещаемость).

Из раскрывающегося списка **Команда** выберите действие, которое должна выполнять команда

4. Введите текст меню и затем выберите действие, которое должна выполнять кнопка. Возможны следующие варианты:

- **Перейти к кнопочной форме** (Go to Switchboard) - переходит на другую страницу кнопочной формы. Страницы кнопочной формы можно использовать для разбиения действительно длинных меню на несколько более коротких:
- **Открыть форму для добавления** (Open Form in Add Mode) - открывает форму в режиме ввода данных, поэтому можно вставлять новые записи;
- **Открыть форму для изменения** (Open Form in Edit Mode) - открывает форму в обычном режиме для просмотра и редактирования записей. Этот режим не разрешает редактировать форму, вопреки вводящему в заблуждение названию;
- **Открыть отчет** (Open Report) - открывает отчет в режиме предварительного просмотра;
- **Конструктор приложения** (Design Application) - открывает окно Диспетчера кнопочных форм, таким образом, позволяя редактировать меню кнопочной формы. Этот вариант редко требуется включать в меню;
- **Выйти из приложения** (Exit Application) - завершает программу Access:
- **Выполнить макрос** (Run Macro) и **Выполнить программу** (Run Code) - запускают созданный вами макрос *(см. главу 15)* или написанный вами код на языке Visual Basic *(см. главу 16).*

5. Повторяйте пункты 3 и 4 до тех пор, пока не создадите все нужные команды. Затем щелкните мышью кнопку **Закрыть** для возвращения в главное окно Диспетчера кнопочных форм.

У кнопочных форм есть неприятный секрет. На каждой странице можно поместить только восемь команд меню. Если вам нужно больше (а кому же не нужно?), следует добавить дополнительные страницы в ваше меню.

На профессиональном уровне.

Кнопочные формы с несколькими страницами

В каждый момент времени кнопочная форма отображает одну страницу с командами меню. Когда такая форма создается впервые, у нее всего одна страница. Но вы можете легко добавить дополнительные в окне Диспетчера кнопочных форм (см. рис. 14.11) с помощью кнопки **Создать.**

У кнопочных форм нет встроенных средств перехода с одной страницы на другую. Вместо этого вы должны самостоятельно вставить команду **Перейти к кнопочной форме** (Go to Switchboard). Допустим, у вас есть три страницы - главная и еще две с дополнительными командами. На главной странице вам понадобятся две команды **Перейти к кнопочной форме,** каждая будет переходить на одну из новых страниц. На новых страницах также нужна команда

Перейти к кнопочной форме, которая позволит вернуться на главную страницу.

Применение нескольких страниц в кнопочных формах нельзя назвать идеальным. Они очень напоминают досаждающие кнопочные меню тонального набора в автоматизированных системах голосовой почты. Прежде чем вы освоите их, вы вынуждены переходить на страницы, у которых есть дополнительные страницы и так пробираться через все команды, пока не найдете нужную. Поэтому, если возможно, ограничьте вашу кнопочную форму восемью задачами. Если вы Достаточно искусны, то можете собрать вместе несколько форм, используя либо подчиненные формы, либо кнопки, позволяющие переходить к связанным записям. Если же вам действительно и непременно нужна более сложная кнопочная форма, подумайте о разработке собственной. В этом случае вы сможете разделить все кнопки переходов на отдельные секции, нарисовать границы вокруг логически связанных кнопок, добавить надписи с пояснительным текстом и т. д. В результате она, возможно, будет выглядеть гораздо лучше.

1. Если вы решили использовать несколько страниц в кнопочной форме, щелкните мышью кнопку **Создать** для добавления страницы, введите имя страницы и щелкните мышью кнопку **ОК.** Далее щелкните мышью кнопку **Изменить** для вставки команд и в завершение кнопку **Закрыть.**

Следуйте инструкциям, приведенным в пунктах 3-5 для вставки команд на данную страницу.

2. Когда создание кнопочной формы завершено, щелкните мышью кнопку **Закрыть** в окне Диспетчера кнопочных форм.

Для того чтобы испытать вашу кнопочную форму, найдите новую, созданную программой Access кнопочную форму и дважды щелкните ее кнопкой мыши.

Кнопочные формы не всегда корректно отображаются в режиме окон со вкладками, используемом программой Access. Главным образом у них появляется дополнительное пустое пространство внизу и справа. Для решения этой проблемы можно отобразить кнопочную форму как всплывающее окно, которое выводится поверх всех остальных окон. В этом случае размер окна отображается корректно. Для внесения этого изменения откройте форму в **Конструкторе.** Настройте форму с помощью **Окна свойств,** выбрав на вкладке **Другие** свойство **Всплывающее окно** и задав в нем значение *Да.*

За кадром.

Меню кнопочных форм сохраняются в БД

Если вы относитесь к типу людей, которым нравится беспокоиться о потенциальных проблемах, которые могут возникнуть на горизонте, то, возможно, уже заметили в мире кнопочных форм кое-что подозрительное. Легко представить себе сценарий, в котором вы создали замечательную кнопочную форму, а затем решили перепроектировать свою БД с помощью нескольких новых форм. В этой ситуации не хотелось бы иметь новые формы и старую кнопочную форму, в которой нет кнопок, предназначенных для отображения новых форм.

К счастью, разработчики корпорации Microsoft, создавшие кнопочные формы, подумали о проблеме именно такого рода и решили заставить программу Access сохранять меню кнопочной формы в БД.

Вот как действует это средство. Когда создается кнопочная форма, Access добавляет в БД таблицу, названную **Switchboard Items** (Элементы кнопочной формы). Когда вы вводите пункты меню в кнопочную форму, программа Access вводит их в упомянутую таблицу. Для того чтобы такая система функционировала, программа должна выполнить несколько дополнительных действий. А именно, когда открывается кнопочная форма, Access выполняет макрос *(см. главу 15),* который извлекает список элементов кнопочной формы из таблицы и использует его для формирования набора кнопок, отображаемых на кнопочной форме.

Воспользоваться этим можно следующим образом. Если после создания кнопочной формы вы решили, что хотите ее изменить, это нетрудно. Просто откройте Диспетчер кнопочных форм (выберите на ленте **Работа с базами данных → Работа с базами данных → Диспетчер кнопочных форм)** и затем отредактируйте элементы формы. Программа Access обновит записи в таблице **Switchboard Items.** Саму кнопочную форму изменять не нужно. Таким образом, если

вы настроили кнопочную форму (добавив свое содержимое или собственные кнопки), она нисколько не пострадает.

Обычно нет нужды открывать непосредственно таблицу **Switchboard Items,** следовательно, почему бы не скрыть ее в области переходов и избежать осложняющих ситуаций? В *разд. "Область переходов "главы 1* написано, как это сделать.

14.2.2. *Назначение стартовой формы*

Взгляд на кнопочную форму как на интерфейс для вашей БД - хорошая отправная точка для пользователей, собирающихся работать с этой БД. Вы можете заставить программу Access открывать любую форму (например, кнопочную) автоматически, когда кто-нибудь в первый раз открывает БД. Вот как это делается.

1. Выберите последовательность **Office → Параметры Access** (Office → Access Options). На экране откроется окно **Параметры Access.**

2. В списке слева щелкните кнопкой мыши категорию **Текущая база данных** (Current Database). На экране появятся параметры текущей БД.

3. Под заголовком **Параметры приложений** (Application Options) найдите поле **Форма просмотра** (Display Form). Выберите в списке кнопочную форму.

4. Если ваша кнопочная форма полностью исключает необходимость применения области переходов, найдите под заголовком **Переходы** (Navigation) и сбросьте флажок рядом с параметром **Область переходов** (Display Navigation Pane).

Если вы боитесь, что чрезмерно усердные пользователи могут открыть то, что не следует, скройте область переходов и научите их пользоваться кнопочной формой для любых нужд. Это напоминает перемещение по БД с помощью автомобильного тренажера.

Подсказка

Каждый раз, когда завершается задача обработки БД, вы возвращаетесь на кнопочную форму и выбираете другую задачу (или завершаете работу в программе Access). Для облегчения этого процесса вы можете добавить на каждую создаваемую форму кнопку, которая ее закрывает, позволяя кнопочной форме снова выйти на передний план. Сделать это можно с помощью Мастера кнопок *(см. разд. "Выполнение действий с помощью кнопок" главы 13).*

14.2.3. *Альтернативы кнопочной формы*

Кнопочные формы - замечательная вещь, но они не лишены очевидных недостатков. Восьмиэлементное ограничение, слегка старомодный внешний вид и дополнительное техническое сопровождение - серьезные основания для того, чтобы дважды подумать о применении кнопочной формы, если только она не упрощает по-настоящему переходы в БД, работать с которой без такой формы существенно сложнее.

Если вы не убеждены твердо в том, что кнопочная форма Access - именно то, что нужно, можно попробовать применить другие средства, описанные в следующих разделах.

14.2.3.1. *Пользовательские кнопочные формы*

Простейший и самый неотразимый вариант - построение собственной кнопочной формы вручную и затем превращение ее в стартовую форму для вашей БД.

Посмотрите на форму, показанную на рис. 14.14 и отображающую те же кнопки переходов, что и кнопочная форма на рис. 14.10, но с изрядной порцией современного стилевого оформления. На этой кнопочной форме отображено чистое, ничем не занятое пространство, наряду с привлекательной графикой. Она также содержит несколько обычных элементов управления, **Кнопок,** созданных Мастером кнопок *(см. разд. "Выполнение действий с помощью кнопок" главы 13).* В свойстве всех кнопок **Тип фона** (Back Style) задано значение *Прозрачный* (Transparent), чтобы придать им более современный плоский внешний вид, В свойстве **Указатель при наведении** (Cursor On Hover) задано значение *Указатель на гиперссылку* (Hyperlink hand), так что пиктограмма указателя мыши меняется на руку с поднятым указательным пальцем, когда мышь перемещается по кнопке, и тем самым дает знать, что в этом

месте можно щелкнуть кнопкой мыши.

Примечание

Другой вариант - использование изображения как фона для всей формы и расположение других элементов управления поверх него. Для этого нужно задать следующие свойства формы: **Рисунок** (имя файла с изображением, которое хотите выводить), **Мозаичное заполнение** (повторять вывод изображения для заполнения всего доступного пространства или нет), **Выравнивание рисунка** (используйте значение *Сверху слева,* так чтобы рисунок начинался в левом верхнем углу формы) и **Масштабы рисунка** (используйте значение *Обрезать,* тогда рисунок не растягивается, не масштабируется и не искажается никаким другим образом). Все элементы управления с фоновым рисунком, помещаемые на верхний слой формы, должны иметь значение *Прозрачный в* свойстве **Тип** фона, так чтобы сквозь них было видно изображение.

Проверьте страницу "Missing CD" на Web-сайте www.missingmanuals.com, чтобы посмотреть экранный образец (интерактивное анимационное руководство), демонстрирующий создание пользовательской кнопочной формы, показанной на рис. 14.14.

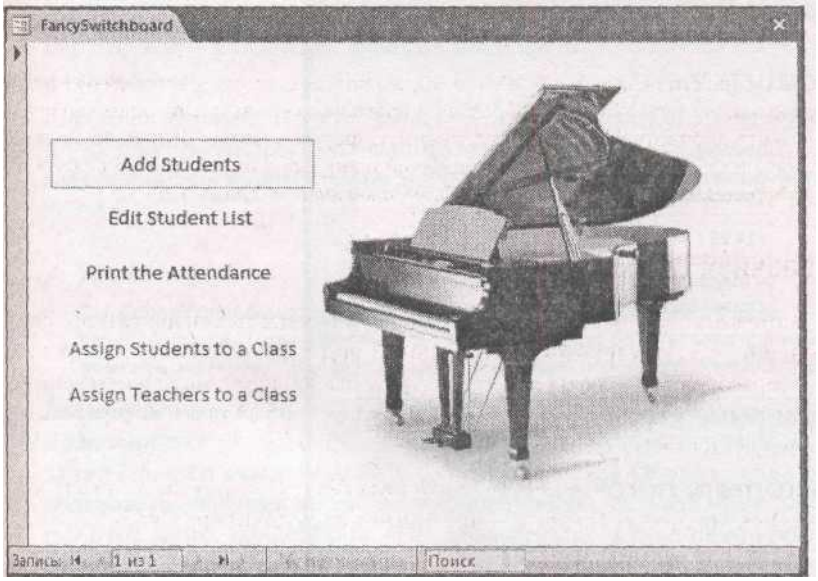


Рис. 14.14. Эта пользовательская кнопочная форма - обычная форма с множеством кнопок переходов. Преимущество разработки собственной кнопочной формы заключается в том, что вы можете делать все так, как вам нравится. Недостаток в том, что потребуется больше усилий для ее обновления, если изменяется БД. При каждом добавлении новой формы придется изменять проект кнопочной формы, чтобы можно было использовать новую форму

14.2.3.2. *Составные формы*

Другой подход позволяет забыть о способе проектирования переходов от формы к форме и вместо этого создать форму, содержащую все, что вам нужно. Это средство, называемое *составной формой,* использует элемент управления Подчиненная форма, с которым вы познакомились в *главе 13.*

В *главе 13* рассказывалось о том, как элемент управления Подчиненная форма позволяет отображать связанные данные (такие как перечень товаров, входящих в текущую категорию товаров). Но применять подчиненную форму имеет смысл и для отображения нескольких несвязанных таблиц в одном месте. Просто оставьте пустыми свойства подчиненной формы **Основные поля** и **Подчиненные поля** - в этом случае подчиненная форма отображает все записи без фильтрации. На рис. 14.15 показан пример.

Подсказка

Существует метод, ускоряющий создание составной формы. Сначала выберите на ленте **Создание → Формы → Конструктор форм** для создания пустой новой формы. Найдите форму,

которую хотите использовать в подчиненной форме, и перетащите ее мышью из области

переходов на рабочую поверхность вашей новой формы. Программа Access создаст элемент управления **Подчиненная форма,** в котором отображается эта форма. Можно также перетащить таблицу на вашу форму, в этом случае Access создаст подчиненную форму для таблицы (и попросит выбрать для нее имя).

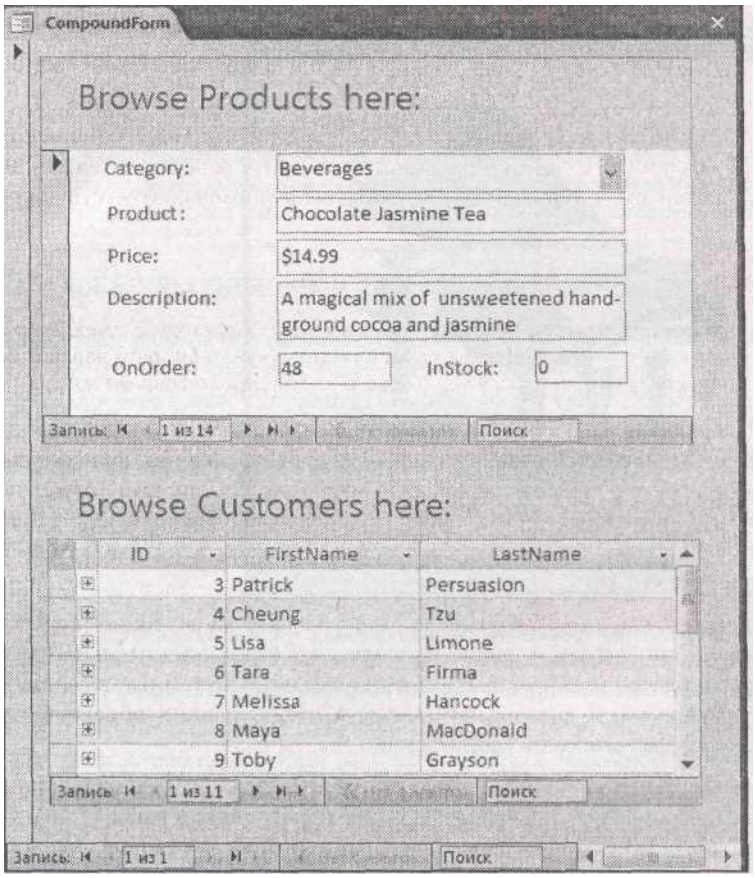


Рис. 14.15. Эта составная форма - пульт управления "все в одном" для добавления и просмотра товаров и просмотра списка клиентов. Подготовленные и включенные в состав программы Access шаблоны часто используют составные формы для размещения нескольких связанных задач редактирования в одном месте

Если в области переходов применяется режим **Таблицы и связанные представления,** составная форма обычно отображается в группе **Несвязанные объекты,** поскольку кнопочная форма сама по себе не использует никакие таблицы. Вместо этого она содержит подчиненные формы, и эти формы используют различные таблицы, которые вы выводите на экран.

14.2.4. *Отображение всех форм в списке*

Это последнее средство может оказаться полезным при разработке пульта управления переходами. Вместо создания кнопок для всех форм, которые хотите использовать, можно создать элемент управления **Список,** включающий их все. Когда пользователь, работающий с БД, выбирает форму из списка, программа Access переходит к этой форме. Такой подход очень удобен при большом количестве форм, способных сделать кнопочный метод непоправимо запутанным.

Примечание
Этот метод работает так же хорошо с отчетами, как и с формами.

Первый шаг - включение в список имен форм. Программа Access позволяет сделать это тремя способами.

■ *Ввод имен вручную.* Просто поместите на форму элемент управления **Поле со списком.** Когда запустится мастер создания элемента управления, выберите переключатель **Будет введен фиксированный набор значений** и затем введите имена форм в соответствующем порядке.

Примечание

См. более подробную информацию о мастере создания списка в *разд. "Переходы с помощью списков" главы 13.* Только помните о том, что в конце работы мастера нужно выбрать вариант **Запомнить значение.** Ваш список применяется для переходов, а не для редактирования записей.

■ *Извлечение имен из пользовательской таблицы, созданной вами.* Создайте новую таблицу и заполните ее именами форм, которые хотите включить в список. Затем в процессе создания **Поля со списком** выберите переключатель **Объект "поле со списком" будет использовать значения из таблицы или запроса** и задайте вашу пользовательскую таблицу. Этот метод концептуально подобен методу функционирования кнопочной формы, создаваемой программой Access.

■ *Извлечение имен из системной таблицы.* В качестве действительно эффективного средства можно получить полный список форм прямо из вашей БД без дополнительных усилий. Суть в использовании одной из скрытых системных таблиц. Системные таблицы - это таблицы, которые программа Access применяет для отслеживания объектов БД. У каждой БД, созданной Access, есть такие таблицы, скрытые от глаз.

Первые два варианта просты. Третий - более впечатляющий, но требует немного больше работы. Обычно системные таблицы скрыты от глаз. Отобразить их можно (рис. 14.16), установив флажок **Показывать системные объекты** в окне **Параметры переходов.** Надолго оставлять их видимыми не стоит, т. к. любое изменение в них может повредить вашу БД и озадачить программу Access.

Системные таблицы можно использовать и не выводя на экран. Самая интересная системная таблица - **MsysObjects,** в которой перечислены все объекты БД. Можно получить список всех форм вашей БД, создав запрос к этой таблице с помощью SQL-команды *(см. в разд. "Режим SQL" главы 6* информацию об использовании в запросах языка SQL). Поле Name содержит имена объектов БД, поле **Type -** числовой код, обозначающий тип объекта. В табл. 14.1 приведены типы, которые могут вас заинтересовать.

Таблица 14.1. Коды полезных типов

Объект	Тип	Объект	Тип
Таблица (Table)	1	Форма (Form)	-32 768
Запрос (Query)	5	Отчет (Report)	-32 764

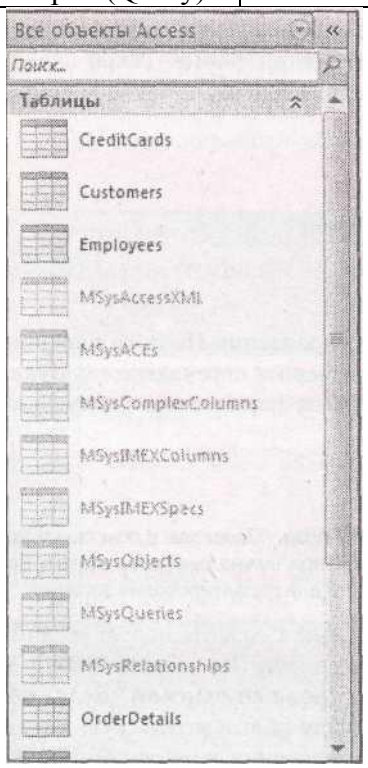


Рис. 14.16. В области переходов отображается группа системных таблиц, которые обычно

скрыты. Их можно открыть и просмотреть, но трудно понять смысл содержащихся в них данных (большей частью числовых)

На основе этих данных можно получить список форм - извлечь поле **Name,** а затем отобрать записи со значением типа -32 768.

Легче всего реализовать описанную логику в элементе управления **Список,** добавив список на форму и пропустив мастер создания списка (нажмите клавишу <Esc>, когда мастер запустится). Затем элемент управления можно настроить с помощью **Окна свойств.** На вкладке Данные найдите свойство **Источник строк** и введите следующую SQL-команду, которая выполнит нужный запрос:

SELECT Name FROM MSysObjects WHERE MSysObjects.Type=-327 68

Теперь у вас есть список, отображающий все формы вашей БД. Можно заменить число -32 764 числом -32 768 и получить все отчеты. На рис. 14.17 показан результат.

Итак, вы увидели только половину нужного решения. Известно как включить список в подходящий элемент управления, но при использовании этого элемента пока ничего не произойдет. Вам нужен реальный способ перехода к выбранной в списке форме или отчету.

Оказывается, это решение немного сложнее примеров, виденных вами до сих пор. Для того чтобы заставить его работать, придется изменить макрос. (Макрос - это перечень нескольких инструкций, хранящийся как объект БД, который можно использовать в любое время.)

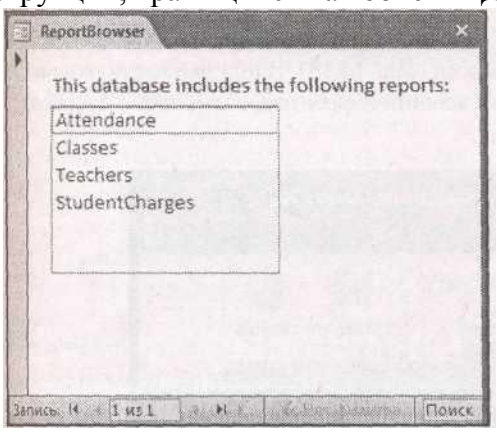


Рис. 14.17. В данной форме отображен список всех доступных отчетов

Как вы узнали из *главы 13,* в процессе создания кнопки Мастер кнопок задает несколько вопросов и затем формирует нужный макрос. Но мастер создания кнопок катастрофически ограничен функционально. К примеру, он может создать макрос, выполняющий переход к конкретной форме, и не способен создать макрос перехода к любой форме. Но вы можете создать простой макрос с помощью мастера, а затем, приложив немного дополнительных усилий, исправить его в соответствии с собственными нуждами. Вот как это делается.

1. Поместите кнопку на вашу форму.

Расположите ее рядом с элементом управления **Поле со списком.** Запустится Мастер кнопок.

2. Выберите категорию **Работа с отчетами** и действие **Открыть отчет,** а затем щелкните мышью кнопку **Далее.**

Если вы отображаете список форм, выберите категорию **Работа с формами** и действие **Открыть форму.**

3. Выберите любой отчет или форму и щелкните мышью кнопку **Далее.**

Неважно, какой объект вы выберите, поскольку эту часть вы измените позже.

4. Завершите мастер.

Убедитесь, что у кнопки подходящий заголовок, например "Выполнить", "Открыть форму" или "Вывести отчет".

Когда мастер завершил работу, самое время внимательнее взглянуть на кнопку в **Окне свойств.**

5. В **Окне свойств** выделите вновь созданную кнопку и перейдите на вкладку **События** (Event).

События - это действия или явления, которые запускают ваш макрос. Например, у всех

кнопок есть событие **Нажатие кнопки** (OnClick), которое возникает, когда вы щелкаете кнопку мышью.

6. Найдите событие **Нажатие кнопки** (OnClick) и щелкните мышью в поле свойства, в котором выводится строка **[Внедренный макрос]** ([Embedded Macro]).

В углу поля появляется кнопка с многоточием (...).

7. Щелкните мышью многоточие для редактирования макроса.

На экран выводится окно редактирования макроса (рис. 14.18). В нижней части окна находится раздел **Аргументы макрокоманды,** позволяющий корректировать работу макроса.

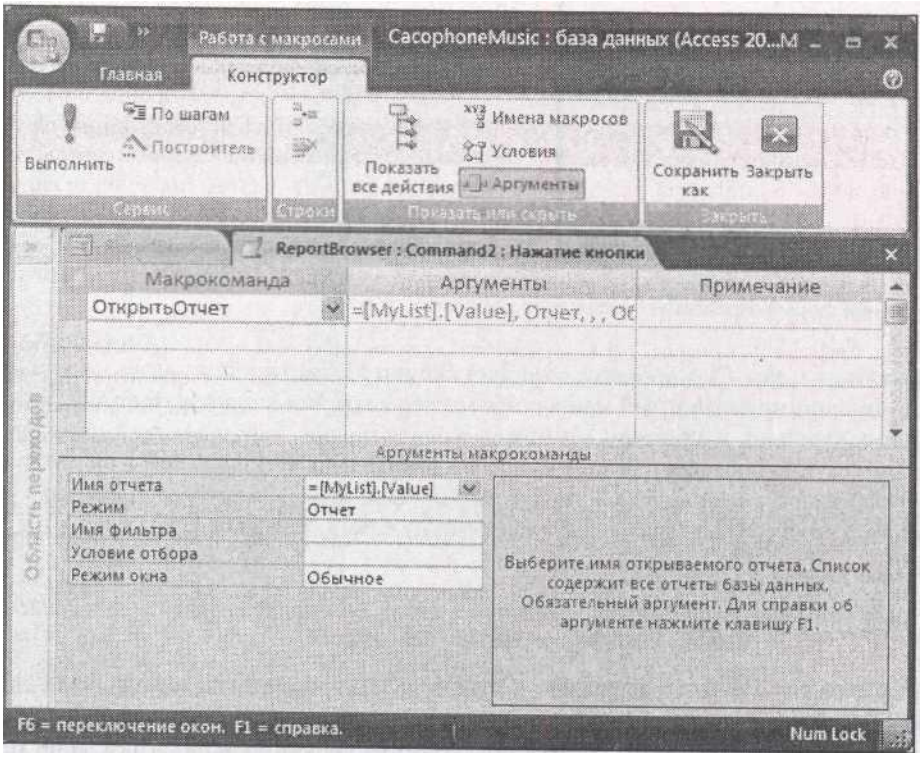


Рис. 14.18. Вы узнаете об этом окне гораздо больше в *главе 15.* Сейчас вам нужно знать лишь то, что у данного макроса единственная макрокоманда (представленная одной строкой в табличной сетке). Эта макрокоманда открывает отчет (на что указывает значение **ОткрытьОтчет** в столбце **Макрокоманда)**

8. В разделе **Аргументы макрокоманды** найдите свойство **Имя отчета** (или **Имя формы).**

Замените его значение выражением =MyList.Value.

Это выражение находит поле со списком и извлекает выделенное в данный момент значение. Предполагается, что поле со списком названо **My List.** Если нет, измените выражение соответствующим образом. (Если вы не помните имени вашего поля со списком, выделите элемент щелчком кнопки мыши и посмотрите, какое имя выводится в раскрывающемся списке в верхней части **Окна свойств.)**

9. Закройте окно макроса и ответьте Да на предложение сохранить изменения.

Вы вернетесь в окно **Конструктора** формы.

10. Перейдите в **Режим формы** и проверьте магические свойства вашего нового списка.

У вас появится возможность выделить форму в списке и затем щелкнуть мышью кнопку для открытия выбранной формы.

14.3. *Ссылки на связанные данные*

Кнопочная форма - функциональная возможность взглянуть на БД с высоты птичьего полета. Но ваша работа на этом не закончена. Хорошо спроектированная система переходов позволяет легко переходить от одной формы к другой так, что вы можете эффективно перемещаться по всей БД.

Секрет системы переходов от формы к форме заключается в продумывании рабочего потока ваших данных (порядка, в котором вы переходите от задачи к задаче в процессе работы с БД). Предположим, что у вас мебельная компания, продающая кофейные столики с ручной росписью. Что происходит, когда вы получаете новый заказ? Возможно, вы начинаете с

создания или выделения информации о клиенте (в одной форме), а затем добавляете сведения о заказе этого клиента (в другой форме). Кнопочная форма не должна переходить сразу к информации о заказе. Начинать следует с формы, содержащей данные о клиенте. У этой формы должна быть кнопка (или другой элемент управления), которая позволит перейти к форме с заказами.

Вы должны совершить аналогичный мыслительный процесс, создавая формы, скажем, для отдела по работе с клиентами. В этом случае им нужен способ выбора клиента и быстрого просмотра подробных сведений о выписанных счетах и выплатах клиента, информации о заказе и записей о доставках. Для этого сценария лучшим решением могло бы стать создание составной формы, собирающей все эти сведения вместе.

Переход от одной формы к другой не составляет труда. Все, что нужно, - это подходящая кнопка. В следующих двух разделах подробно описаны два часто встречающихся примера.

14.3.1. *Отображение связанных записей в отдельной форме*

В *главе 13* вы узнали, как элемент управления **Подчиненная форма** может отображать связанные записи в одном месте. Но подчиненные формы не всегда предоставляют достаточно места для работы. В зависимости от способа обработки и величины объема реальной информации, с которой вы сталкиваетесь, возможно, вы предпочтете отображать подчиненные записи в другом месте. Можно добавить на форму кнопку, которая раскрывает другую форму со связанными записями. Для реализации этого метода во второй форме применяется фильтрация для отображения только подчиненных записей. На рис. 14.19 и 14.20 показан пример из БД Cacophone Studios.

Формы, представленные на рис. 14.19 и 14.20, можно создать без особых усилий. Немного сложнее обстоит дело с кнопкой **See Students in this Class.**

Далее перечислено все, что нужно сделать, для реализации кнопки, открывающей вторую форму для отображения связанных записей.

1. Откройте родительскую форму.

В данном случае начните с формы Classes.

2. На вкладке ленты **Конструктор** щелкните кнопкой мыши пиктограмму **Кнопка.** Нарисуйте кнопку на вашей форме.

Запустится Мастер кнопок.

3. Выберите категорию **Работа с формой** и действие **Открыть форму** и щелкните мышью кнопку Далее.

На следующем этапе мастер отобразит все формы в вашей БД.

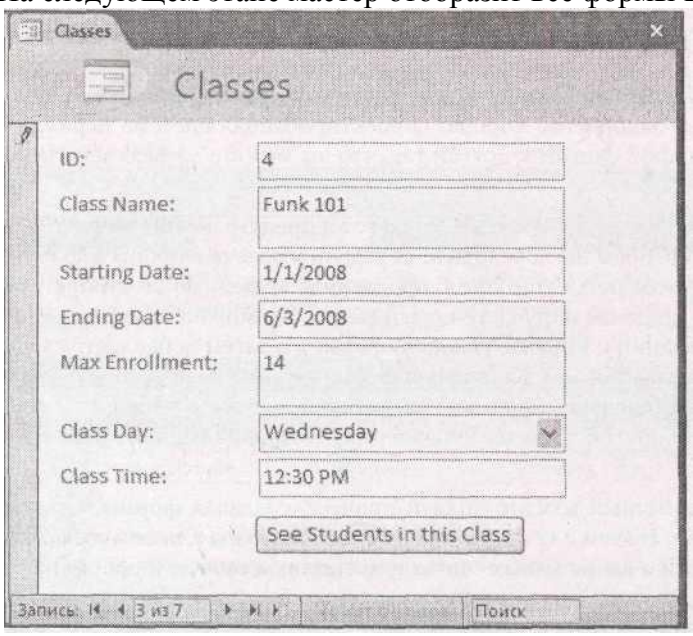


Рис. 14.19. Форма **Classes** выводит на экран список классов. Щелкните мышью кнопку **See Students in this Class** (просмотреть список студентов этого класса) для открытия второй формы (рис. 14.20)

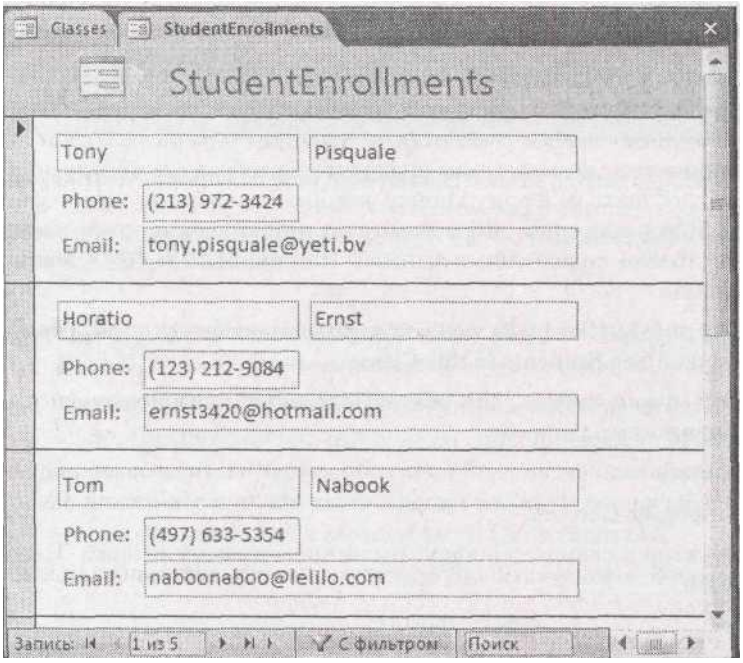


Рис. 14.20. Форма **StudentEnrollments** (список студентов) содержит студентов только одного класса

4. Выберите подчиненную форму, в которой есть связанные записи, и затем щелкните мышью кнопку **Далее.**

В данном случае выберите форму **StudentEnroIlments.**

5. Выберите переключатель **Открыть форму и показать все записи** и затем щелкните мышью кнопку **Далее.**

Этот пункт кажется немного странным - разве вы не собирались отображать только связанные записи из таблицы **StudentEnrollments?** Конечно, да. Но, к сожалению, Мастер кнопок не может помочь - в этой области у него существенная ошибка, мешающая создать правильное условие отбора. Поэтому вам придется проделать немного больше работы, определив самостоятельно условие отбора записей.

6. Введите какой-нибудь текст и выберите рисунок.

Начиная с этой точки, Мастер кнопок отображает стандартные этапы, которые вам уже известны *(см.разд. "Выполнение действий с помощью кнопок" главы 13).*

7. Задайте имя кнопки и щелкните мышью кнопку **Готово.**

Теперь у вас есть кнопка, открывающая нужную форму, но не задана фильтрация. Для этого необходимо изменить макрос, который использует данная кнопка.

Примечание

Макрос - это список действий, которые вы хотите заставить выполнить программу Access. В следующей главе будут подробно рассмотрены макросы. А сейчас у вас достаточно знаний для создания нужной вам кнопки.

8. Если на экране нет **Окна свойств,** выберите на ленте **Инструменты конструктора форм | Конструктор → Сервис → Страница свойств** (Form Design Tools | Design → Tools → Property Sheet).

9. Выделите кнопку, щелкнув ее мышью на рабочей поверхности формы.

Ее также можно выбрать из списка в верхней части **Окна свойств.**

10. В **Окне свойств** выберите вкладку **События** и щелкните кнопкой мыши поле **Нажатие кнопки** (OnClick).

Вы увидите в нем текст **[Внедренный макрос],** свидетельствующий о том, что к данному событию присоединен макрос.

11. Щелкните мышью кнопку с многоточием для открытия окна редактирования макроса.

Появится новая вкладка, на которой перечислены по порядку все макрокоманды, выполняемые макросом. Вы познакомитесь с этим окном в *главе 15.* Сейчас нужно внести только два простых изменения.

12. В начале списка вы увидите макрокоманду **ОткрытьФорму.** (Она открывает

подчиненную форму при щелчке мышью кнопки.) Выделите ее, щелкнув кнопкой мыши (рис. 14.21).

Когда в макросе выбрана макрокоманда, в разделе **Аргументы макрокоманды,** расположенном в нижней части окна, появляется набор сведений о ней.

13. Щелкните кнопкой мыши поле **Условие отбора** (в нижней части окна, в разделе **Аргументы макрокоманды)** и затем введите ваше выражение для фильтрации записей.

Это условие отбора должно выбрать связанные записи. В данном примере это означает, что вас интересуют записи, у которых текущий код класса.

Далее приведено нужное вам условие отбора:
[ClassID]=[Forms]![Classes]![ID]

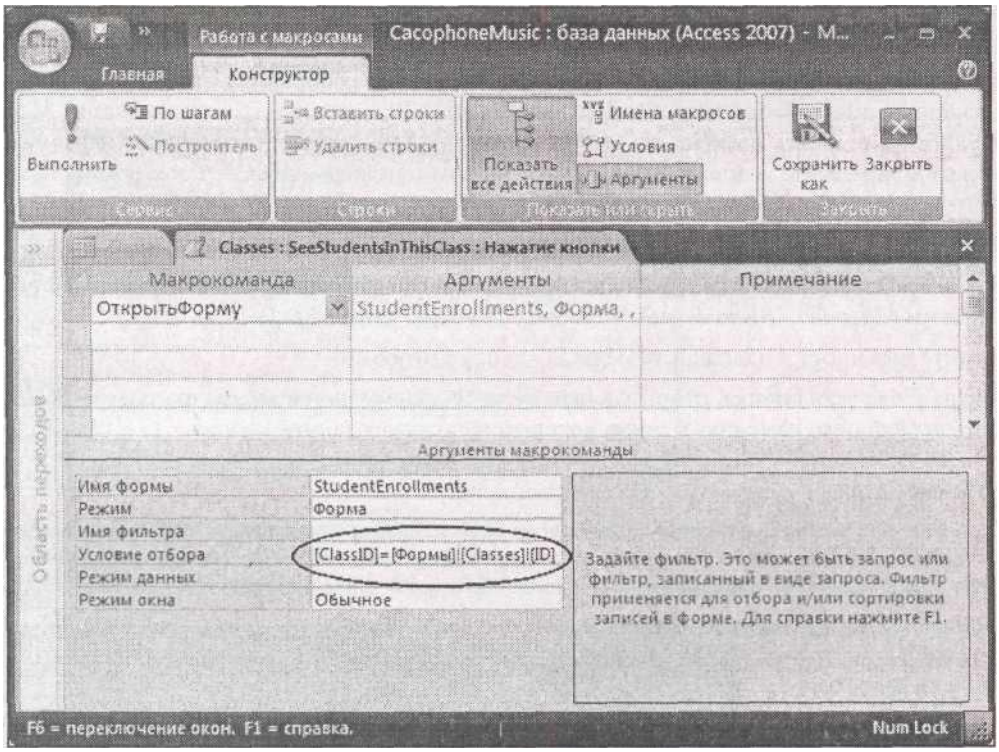


Рис. 14.21. Выражение для фильтрации следует поместить в поле **Условие отбора**

Это выражение сообщает программе Access о том, что необходимо отображать запись, только если значение поля **ClassID** в форме **StudentEnrollments** совпадает со значением поля ID в форме **Classes.** Другими словами, вы получите список студентов, зачисленных в текущий класс.

Примечание

Странные восклицательные знаки в выражении для фильтрации позволяют связать две формы. Условие отбора задается в форме, которую вы открываете (форма **StudentEnrollments)** и у которой есть поле **ClassID.** Но вы должны сократить число отображаемых ею записей на основе поля **ID,** хранящегося в другой форме **(Classes).** Синтаксическая запись [Forms] ! [Classes] ! [ID] - просто замысловатый способ сказать программе Access о том, что искать нужное ей значение **ID** следует в открытой в данный момент форме с именем **Classes.**

14. Изменение, сделанное в предыдущем пункте, почти завершает корректировку макроса, Но хорошо бы добавить еще одну макрокоманду. Щелкните кнопкой мыши поле, расположенное под макрокомандой **ОткрытьФорму,** и введите **Обновление** (Requery) (рис. 14.22).

Данная команда заставляет программу Access обновить текущую форму (форму **StudentEnrollments,** которую вы только что открыли). Этот шаг необходим, поскольку форма **StudentEnrollments** могла быть уже открыта, когда вы щелкнули мышью кнопку **See Students**

in this Class. Если так и было, в вашем макросе изменился фильтр, но он не выполнил фильтрацию. Для обновления отображаемых записей необходимо выполнить команду Обновление и заставить форму обновить саму себя.

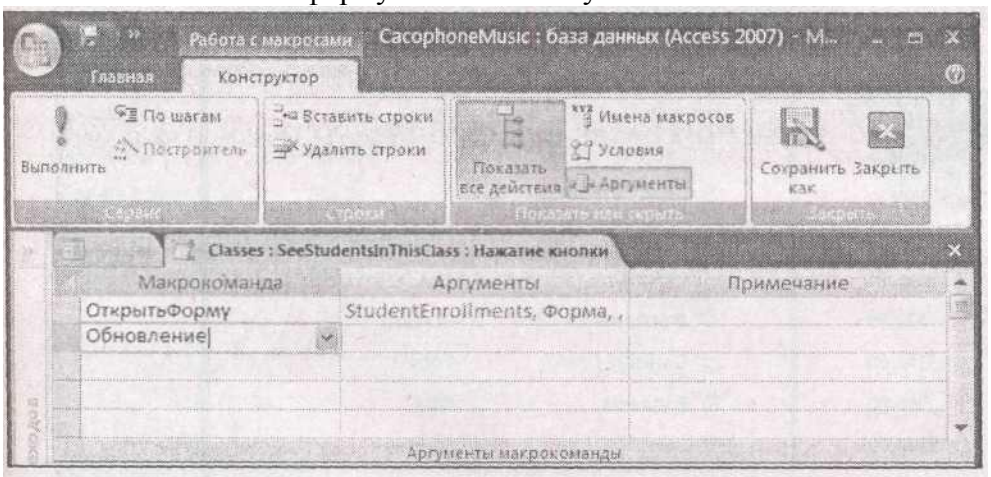


Рис. 14.22. Макрокоманда **Обновление** обновляет отображение текущей формы. Ей не нужны никакие дополнительные данные

15. Теперь работа завершена. Закройте вкладку с макросом и щелкните мышью кнопку Да в ответ на предложение программы Access сохранить макрос.

У вас появилась замечательная кнопка перехода, которая отображает связанную форму и ограничивает ее лишь теми записями, которые вас интересуют. В следующей главе вы узнаете гораздо больше о тонкой настройке макросов.

Для испытания вашей кнопки перейдите в **Режим формы** и щелкните кнопку мышью. Когда вы щелкните мышью кнопку **See Students in this Class** и откроется форма **StudentEnrollments,** сработает заданное условие отбора записей.

Подсказка

Любой пользователь может удалить ваше условие отбора с помощью группы ленты **Главная → Сортировка и фильтр** (или щелкнув мышью поле **С фильтром** (Filtered), которая выводится в нижней части формы, рядом с кнопками перехода). Если вам не нужна такая гибкость настройки, можно настроить форму **StudentEnroIIments** так, что она никому не позволит изменять свои параметры фильтрации. Для этого откройте форму в режиме **Конструктора,** выделите элемент **Форма** в списке **Окна свойств** и измените значение свойства **Применение фильтров** с *Да* на *Нет.*

14.3.2. *Отображение более подробных отчетов с помощью связей*

Для обеспечения переходов между отчетами можно использовать аналогичный метод. При желании можно сформировать возможность перехода из одного отчета в другой, подчиненный Макрос, который нужно создать, почти идентичен тому, который мы рассматривали в предыдущем примере.

Как правило, специалисты Access применяют этот метод для запуска основного отчета и предоставляют возможность пользователям щелчками кнопки мыши проложить путь к более подробному отчету, который акцентирует внимание на части данных (рис. 14.23 и 14.24).

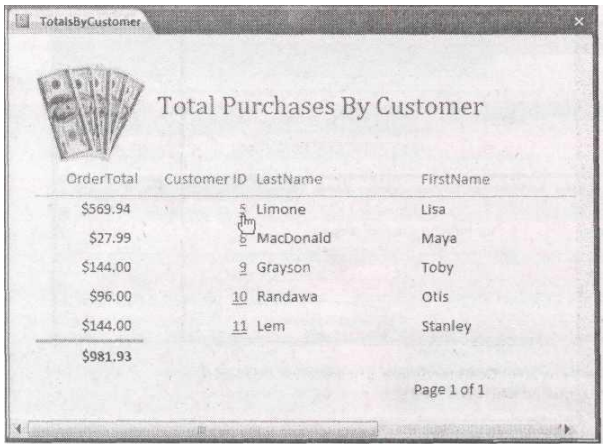


Рис. 14.23. Первый отчет **(TotalsByCustomer)** (общая стоимость заказов клиента) отображает всех клиентов и общую стоимость заказов для каждого из них. Щелкните кнопкой мыши одного из клиентов, и программа Access запустит вывод более подробного отчета, показанного на рис. 14.24

Примечание

Отчеты проектируются для вывода на печатающие устройства. Поэтому большие серые кнопки выглядят несколько неуместно. Другое решение - ссылки - как показано в данном примере, гораздо более распространено, поскольку отображает данные, которые нужно распечатать, и в то же время добавляет интерактивность.

Для создания такого перехода необходимо начать с создания поля ввода, которое выглядит как гиперссылка. (Настоящий элемент управления **Гиперссылка** использовать нельзя, поскольку он отображает только фиксированный, неменяющийся текст. Вместо этого нужен способ отображения содержимого поля как ссылки - в данном примере код клиента.) Затем можно создать макрос, который запускается при щелчке кнопкой мыши поля для перехода к новому отчету. Задача этого макроса - открыть подробный отчет, который вам нужен, и затем применить фильтр для отображения только связанных записей.

Отформатировать поле ввода легко. Можно выделить любой элемент управления и затем изменить его цвет, шрифт и т. д. с помощью команд на ленте. Но вам даже не нужно выполнять эту работу. У каждого поля есть странное свойство, названное **Гиперссылка** (Is

Hyperlink), - задайте для него значение *Да,* и элемент управления **Поле** превратится в подчеркнутый текст синего цвета, что вам и нужно

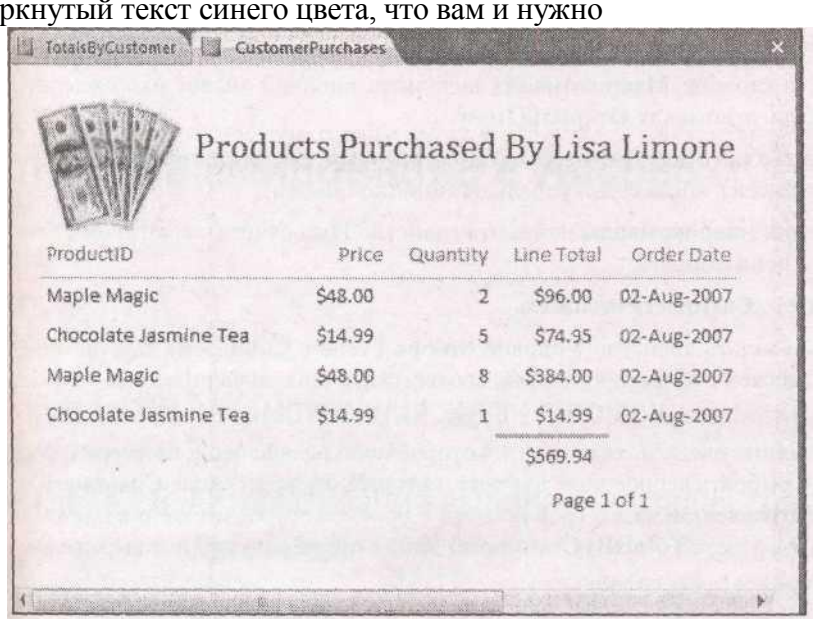


Рис. 14.24. Представленный отчет **CustomerPurchases** отражает привычки выбранного клиента тратить деньги. Выражение построения текстовой строки (="Products Purchased By " & [FirstName] & " " & [LastName]) помещает имя и фамилию текущего клиента в заголовок формы

После того как вы справились с описанной странностью, самое время добавить необходимый макрос. Можно предпринять следующие действия с БД Boutique Fudge (включена в загружаемое из Интернета содержимое примеров для данной главы) для вставки ссылки, открывающей отчет **CustomerPurchases** (покупки клиента) в отчете **TotalsByCustomer** (общая стоимость заказов клиента).

1. Откройте отчет, который хотите использовать, в режиме **Конструктора.**
В данном примере все начинается с формы **Totals By Customer.**

2. Если на экране не видно **Окна свойств,** выберите на ленте **Инструменты конструктора отчетов | Конструктор → Сервис → Страница свойств** (Report Design Tools | Design → Tools → Property Sheet).

3. Решите, какое поле хотите использовать для создания ссылки, и выделите его.

Обычно применяется уникальное значение кода (ID), связывающее две таблицы друг с другом. В данном примере используется поле ID, хранящее идентификатор клиента. Если вы его еще не отформатировали, сделайте это сейчас с помощью свойства **Гиперссылка** так, чтобы поле выглядело как ссылка.

Теперь следует создать и присоединить макрос.

4. В **Окне свойств** перейдите на вкладку **События** и щелкните кнопкой мыши поле **Нажатие кнопки** (OnClick). Щелкните мышью кнопку с многоточием (...)

На экране появляется диалоговое окно **Построитель** (Choose Builder) и запрашивает способ создания кода, который будет выполняться, когда ссылку щелкают кнопкой мыши.

5. Выберите **Макросы** (Macro Builder) и затем щелкните мышью кнопку ОК.
На экране появится окно редактирования макроса.

6. В первой строке, в столбце **Макрокоманда** щелкните кнопкой мыши направленную вниз стрелку. Выберите команду **ОткрытьОтчет.**

Вы также можете выбрать макрокоманду **ОткрытьФорму** для открытия формы (для редактирования записей), когда ссылку щелкают кнопкой мыши.

7. В разделе **Аргументы макрокоманды** измените свойство **Имя** отчета, задайте имя отчета, который хотите использовать.

В данном примере это **CustomerPurchases.**

8. Теперь необходимо задать свойство **Условие отбора** (Where Condition) для применения фильтра. Он должен отбирать клиентов, соответствующих значению ID в текущей записи.

Нужное вам выражение очень похоже на то, которое использовалось в примере с формами. Необходимо выбрать верное поле в отчете, который открыли (поле **CustomerlD** в отчете **CustomerPurchases),** и затем сравнить его с полем, которое вы щелкнули кнопкой мыши (поле ID в отчете **TotalsByCustomers).** Далее приведено нужное выражение:

[CustomerlD]=[Reports]![TotalsByCustomer]![ID]

Как и в примере с формами, это выражение использует замысловатый синтаксис с восклицательными знаками, чтобы сообщить программе Access о том, как найти отчет **TotalsByCustomers.**

9. Под макрокомандой **ОткрытьОтчет** введите **Обновление.**

Как и в примере с формами, следует обновить отчет так, чтобы отбор сработал, даже если отчет уже открыт.

10. Закройте окно макроса и ответьте Да на предложение сохранить изменения.
Вы вернетесь в окно **Конструктора** отчетов.

11. Перейдите в режим **Представление отчета** и опробуйте ссылку.

Теперь можно щелкнуть ссылку кнопкой мыши и углубиться в более подробный отчет.

Вы можете проверить этот пример самостоятельно, воспользовавшись тренировочными БД для данной главы.

Часть V

Программирование в Access

15. Глава 15. Автоматизация задач с помощью макросов

Секрет долгих и счастливых отношений с программой Access кроется в умении заставить ее работать так, как вы хотите.

Как вы уже видели, истинные фанаты Access не пользуются листом данных для ввода информации. Вместо этого они создают собственные настраиваемые формы ввода данных. Приверженцы программы Access также не печатают данные с помощью вызывающих зевоту таблиц. И конечно же, профессионалы Access не повторяют одни и те же утомительные действия при выполнении часто выполняемой задачи - они создают макросы, заставляющие программу Access делать работу за них.

Макрос - это небольшая программа, которую вы создаете и храните в БД. Макрос может быть крайне простым (например, команда отображения формы) и головоломно сложным (например, макрос с условием, проверяющий количество сырого мяса на складе и автоматически печатающий заказ в трех экземплярах, если ваш холодильник пуст).

В данной главе вы узнаете, как создавать базовые макросы. Затем вы научитесь делать их более сообразительными. В конце главы вы сможете собрать вместе макросы, способные запуститься при необходимости, выполнить целую последовательность шагов и даже принимать решения. Эта глава также подготовит переход к следующей главе, в которой вы подойдете к полноценному программированию на языке Visual Basic.

На профессиональном уровне.

Макросы по сравнению с программным кодом

В прошлом у макросов была немного сомнительная репутация. Некоторые специалисты Access избегали их, предпочитая применять более мощный язык программирования Visual Basic (VB) (с которым вы познакомитесь в следующей главе). Корпорация Microsoft внесла в эту ситуацию свою лепту, полагая, что макросы - это устаревшие средства и не лучший выбор для передовых разработчиков.

В Access 2007 Microsoft наконец уделила макросам больше внимания и улучшила их репутацию. Несмотря на то, что у макросов и близко нет таких функциональных возможностей, как у чистого VB-кода, они просты, ясны и быстры настолько, что никакой VB-код не сможет с ними сравниться. Но самое главное достоинство макросов - безопасность. Поскольку программа Access знает, что делает каждый макрос, она может утверждать, что большинство из них безопасны. Другими словами, Access знает, что макрокоманда **ОткрытьФорму** (OpenForm) может применяться только для открытия формы, поэтому не стоит беспокоиться о том, что она может удалить ваши файлы, отправить снам вашим друзьям или отформатировать ваш жесткий диск. В случае VB-кода у программы Access нет такой уверенности.

В результате она склонна блокировать ваши снабженные кодом средства, даже если они не более опасны, чем два волоска, свернувшиеся на подушке. (Вы более подробно познакомитесь с проблемой безопасности в *разд. "Макросы и безопасность" далее в этой главе.)*

Даже если вы решили не тратить времени на макросы и стать программистом на Visual Basic, обладающим черным поясом, все равно следует начать с данной главы. Здесь вы узнаете важную информацию о том, как макросы встраиваются в формы *(см. разд. "Присоединение макросов к формам" далее в этой главе)*. Как вы увидите в следующей главе, программные процедуры взаимодействуют с формами точно так же.

15.1. *Базовые сведения о макросах*

Несмотря на то, что вы могли этого не осознать, вы уже пользовались макросами. В *главе 14* вы создавали кнопки, способные выполнять полезные задачи, например, открывать другие формы или переходить к конкретной записи. Для создания этих макрокоманд вы применяли мастер создания кнопок, который задает несколько простых вопросов и затем генерирует сделанный по мерке макрос.

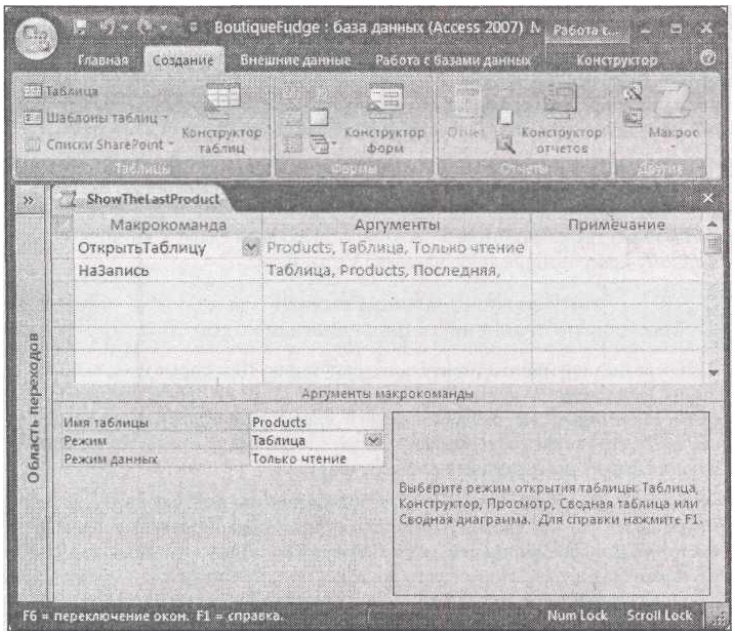


Рис. 15.1. Этот макрос состоит из двух макрокоманд. Первая открывает таблицу, а вторая переходит к заданной записи. Когда вы запускаете макрос, программа Access начинает с вершины списка и двигается вниз, последовательно выполняя макрокоманды

Хотя мастер создания кнопок прост в применении, его не обвинишь в излишней гибкости. Теперь вы достаточно подготовлены для того, чтобы стать еще сильнее и создать собственный макрос.

15.1.1. *Создание макроса*

В следующем примере вы начнете, не спеша, с создания простого макроса, который открывает таблицу и переходит сразу к последней строке. Далее перечислены действия, необходимые для создания макроса.

1. Выберите на ленте **Создание → Другие → Макрос.**

На экране появится новое окно для создания нового макроса. Неофициально оно называется конструктором макроса.

Любой макрос состоит либо из последовательности одного или нескольких шагов, либо из действий (макрокоманд). Для создания макроса вы формируете список макрокоманд, помещая каждую из них в отдельную строку (рис. 15.1). Первоначально этот список пуст и ваш макрос ничего не выполняет.

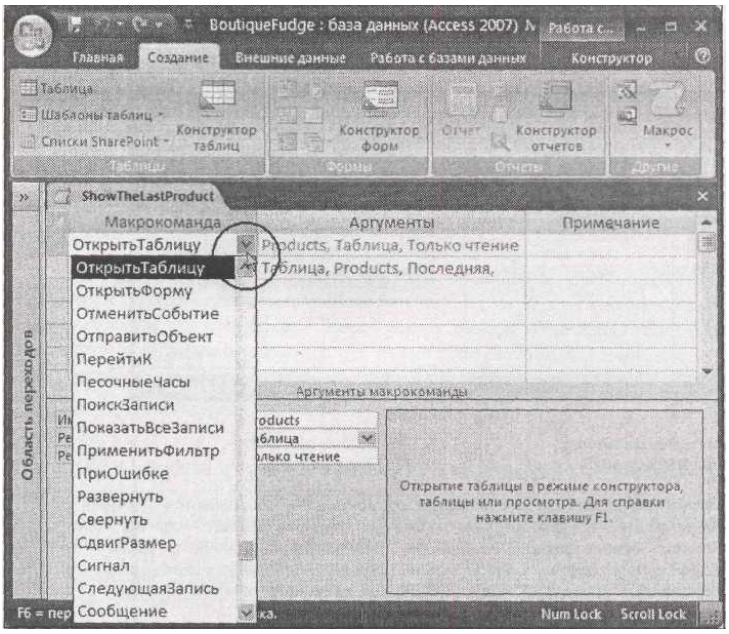


Рис.15.2. Щелкните кнопкой мыши направленную вниз стрелку (обведена), чтобы просмотреть представленные в алфавитном порядке макрокоманды, которые можно использовать. После того как макрокоманда выбрана, в нижнем правом углу окна появляется краткое, но полезное описание

2. Выберите первую макрокоманду.

У программы Access есть предварительно подготовленный список макрокоманд, которые можно использовать для приготовления макросов. Когда добавляется команда, ее просто выбирают из этого списка, как показано на рис. 15.2. В данном примере начните с выбора макрокоманды **ОткрытьТаблицу** (OpenTable).

Примечание

В данный момент вы работаете только с макрокомандами, которые программа Access считает безопасными для всех БД. Немного позже *(см. разд. "Опасные макрокоманды" далее в этой главе)* вы рассмотрите возможность применения нескольких команд, использование которых программа Access считает рискованным делом.

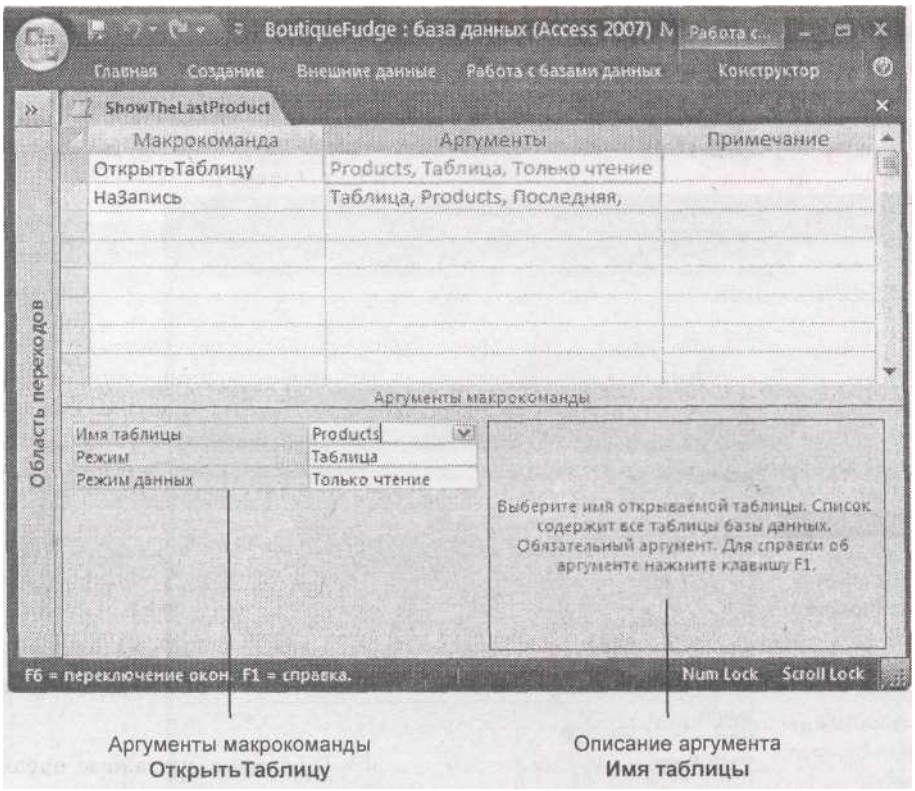


Рис. 15.3. Когда в списке выбирается макрокоманда, все аргументы отображаются в нижней части окна. Как видно из данного примера, у макрокоманды **ОткрытьТаблицу** - три аргумента **(Имя таблицы, Режим** и **Режим данных).** Каждый раз, когда вы щелкаете кнопкой мыши в одном из этих полей, в поле, расположенном справа, появляется краткое описание аргумента. Значения ваших аргументов также отображаются в виде разделенного запятыми списка в столбце **Аргументы,** расположенном рядом с макрокомандой

3. В нижней части вкладки макроса в разделе **Аргументы макрокоманды** (рис. 15.3) выберите параметры для вашего макроса.

Большинству макрокоманд для выполнения их работы нужна от вас некоторая информация. В макрокоманде **ОткрытьТаблицу** (OpenTable) мало смысла до тех пор, пока вы не сообщите программе Access, какую именно таблицу хотите открыть. Эти дополнительные сведения и называют *аргументами.*

У макрокоманды **ОткрытьТаблицу** (OpenTable) три аргумента.

1) **Имя таблицы** ссылается на таблицу, которую вы хотите открыть. Его можно выбрать из раскрывающегося списка таблиц. В данном примере можно использовать любую непустую таблицу.
2) **Режим** позволяет выбрать применяемый режим представления. Можно выбрать обычный режим **Таблица** для ввода информации, **Конструктор** для изменения структуры таблицы, **Просмотр** для подготовки данных к печати или **Сводную таблицу** и **Сводную диаграмму** для работы с итогами сводной таблицы. В данном примере выберите режим **Таблица.** (Конечно, когда таблица открыта, можно переключиться в другой режим представления,

щелкнув правой кнопкой мыши заголовок или кнопку режима на ленте.)

3) **Режим данных** определяет, какой тип изменений разрешен. Можно использовать стандартный вариант **Изменение** (Edit) для разрешения любых изменений, **Только чтение** (Read Only) для запрета каких бы то ни было изменений или **Добавление** (Add) для разрешения только добавления записей. В данном примере выберите **Только чтение.**

Примечание

Вы уже видели, что у пользовательского макроса гораздо больше функциональных возможностей, чем у мастера построения макрокоманды (Command Builder wizard). Когда применяется мастер, можно открывать формы и отчеты, но не обычные таблицы, и нет средств управления режимом представления или разрешенными видами изменений. Макрос не сталкивается с подобными ограничениями.

4. При желании введите в столбец **Примечание** дополнительную информацию, которая напомнит назначение макрокоманды.

Не используйте столбец **Примечание** для объяснения очевидного (как, например, "Открывает таблицу Products"). В этом столбце следует объяснять важность шагов в наиболее сложных операциях. Сейчас вам не нужны комментарии, но позже, когда вы создадите группы макросов и будете применять условия, вы поймете полезность комментариев.

5. Перейдите на следующую строку и повторите шаги 2-4 для вставки другой макрокоманды.

Вы можете вставить в макрос практически неограниченное число макрокоманд. (При наличии маленьких буферов в программе Access макрос может содержать до 999 макрокоманд.) Каждая команда занимает отдельную строку, и программа Access выполняет их по порядку, перемещаясь сверху вниз.

Для завершения данного примера добавьте макрокоманду **НаЗапись** (GoToRecord). Это действие обеспечивает получение нужной записи в таблице, которую вы только что открыли.

При использовании аргументов следует указывать корректный объект (задайте Таблица в качестве **Типа объекта** и имя таблицы, которую вы выбрали в пункте 3, в качестве **Имени объекта).** Затем можно использовать аргументы **Запись** (Record) и **Смещение** (Offset) для точного задания той позиции, в которую вы хотите перейти. С помощью аргумента **Запись** можно выбрать переход к предыдущей строке (Предыдущая (Previous)), к следующей строке (Следующая (Next)), к заполнителю новой строки в конце таблицы **(Новая** (New)), к заданной строке **(Конкретная** (Go To)), к первой строке **(Первая** (First)) или, как в данном примере, к последней строке (Последняя (Last)). Если выбрать **Конкретная,** можно использовать аргумент **Смещение** для указания точной позиции, например, если задать **Смещение** 5, обеспечивается переход к 5-й записи.

Примечание

Некоторые макрокоманды зависят от действия предшествующих макрокоманд. Отличный пример - команда **НаЗапись** (GoToRecord). В ней предполагается, что вы открыли таблицу, форму или запрос, содержащие запись, которую хотите отобразить. Если применить команду **НаЗапись** без предварительного открытия соответствующего объекта, вы получите сообщение об ошибке во время выполнения макроса.

И просто для смеха, почему бы не вставить еще одну макрокоманду? Испытайте команду **Сообщение** (MsgBox), отображающую выбранное вами сообщение в небольшом информационном окне. Задается сообщение в аргументе **Сообщение** (Message). Задайте, например, такой текст: "Ваш первый макрос только что сделал свое дело". Можно также добавить необязательный заголовок (с помощью аргумента **Заголовок** (Title)), предупреждающий звуковой сигнал (задав в аргументе **Сигнал** (Beep) значение Да) и встроенную пиктограмму (с помощью аргумента **Тип** (Type)).

Подсказка

В любой момент можно изменить порядок выполнения макрокоманд. Просто щелкните кнопкой мыши поле слева от элемента, который хотите переместить; этот щелчок выделяет

макрокоманду. Затем перетащите команду с помощью мыши в новое место. Программа Access автоматически раздвинет другие макрокоманды.

6. Нажмите комбинацию клавиш <Ctrl>+<S> для сохранения макроса и задайте ему имя.

В этом примере можно назвать макрос **ShowTheLastProduct** (отображение последнего товара). Если вы не сохраните макрос, программа Access вежливо предложит вам сделать это во время закрытия окна макроса или первого запуска вашего макроса.

Макрос отображается в области переходов. Если ваши объекты сгруппированы по типу объекта, то вы заметите, что у макроса свой тип. Если применяется вариант группировки **Таблицы и связанные представления** (Tables and Related Views), программа Access вставит макрос в дополнительную группу в конце списка, названную **Несвязанные объекты** (Unrelated Objects).

Примечание

Когда используется мастер построителя макрокоманды, также создается макрос. Но этот макрос не выводится в области переходов, поскольку он спрятан в конкретной форме. Такой тип макроса называется *внедренным,* поскольку он встроен внутрь формы.

15.1.2. *Запуск макроса*

Теперь, когда макрос завершен, можно его испытать. Программа Access предоставляет четыре способа запуска макроса.

■ *Можно запустить его явно.* Найдите нужный макрос в области переходов и дважды щелкните его кнопкой мыши. Этот вариант действует, если макрос еще не открыт. Если же макрос открыт, выберите на ленте **Работа с макросами | Конструктор → Сервис → Выполнить** (Macro Tools | Design → Tools → Run).

Подсказка

Если вы применили в области переходов фильтрацию, такую что макрос не отображается на экране, вы все же можете его запустить. Выберите на ленте **Работа с базами данных → Макрос → Выполнить макрос** (Database Tools → Macro → Run Macro). У вас появится возможность выбрать ваш макрос из списка.

Можно запустить макрос с помощью клавиш, например, задать запуск макроса, открывающего ежемесячный финансовый отчет при нажатии комбинации клавиш <Ctrl>+<F>. Вы узнаете, как это делается, в *разд. "Назначение макросу комбинации клавиш " далее в этой главе.*

Можно запускать макрос автоматически при первом открытии БД. Вы можете создать макрос, который позволяет всегда начинать работу с БД запуском вашего любимого запроса и отображением его результатов. В *разд. "Настройка макроса запуска" далее в этой главе* вы попробуете сделать это.

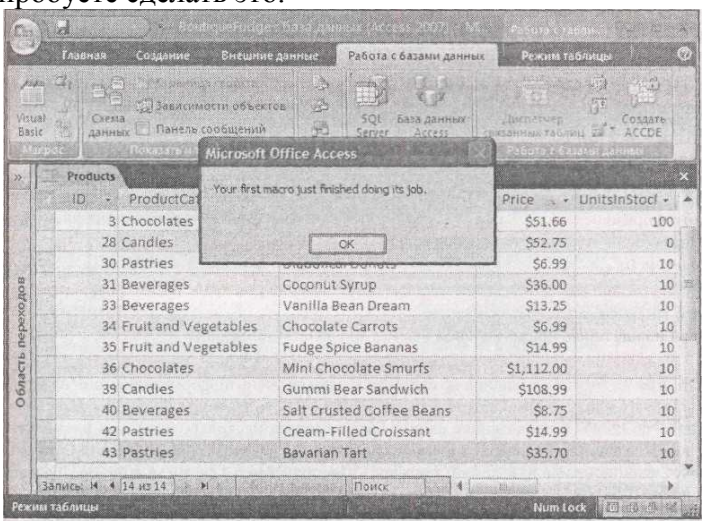


Рис. 15.4. Результат работы макроса **ShowTheLastProduct.** Программа Access открывает

таблицу **Products (в** режиме **Только чтение,** поэтому запрещены любые изменения) и переходит к самой последней, самой свежей записи, а затем выводит на экран сообщение, информируя вас о завершении работы макроса

■ *Можно присоединить макрос к форме.* Вы можете задать автоматический запуск макроса при щелчке мышью кнопки или при вводе новых данных. Это наиболее распространенный способ выполнения макроса и именно так действует мастер построения макрокоманды. В *разд. "Присоединение макросов к формам" далее в этой главе* вы сможете опробовать этот метод.

В данной главе вы получите возможность проверить все описанные способы. Но прямо сейчас выберите простейший вариант и запустите созданный в предыдущем разделе макрос с помощью команды **Работа с макросами | Конструктор → Сервис → Выполнить,** На рис. 15.4 показан результат.

Подсказка

Если вы хотите откорректировать уже созданный макрос, в области переходов щелкните его правой кнопкой мыши и выберите режим **Конструктор.** Вы попадете снова в окно макроса, которое использовали для его создания.

15.1.3. *Отладка макроса*

Не все макросы, действуют без сучка, без задоринки. Если вы допустили ошибку - может быть, ваш макрос пытается открыть несуществующий объект или использует бессмысленный аргумент - то получите исчерпывающее сообщение об ошибке, как показано на рис. 15.5.

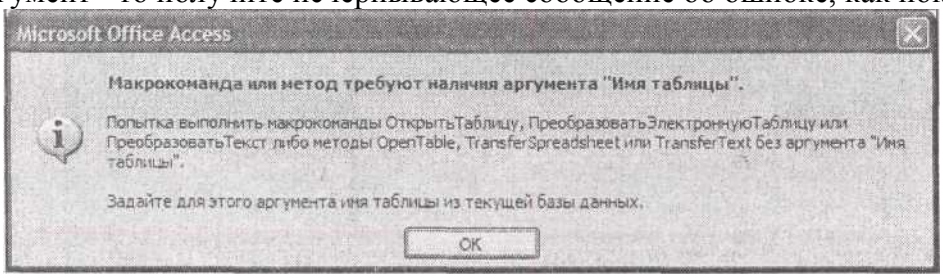


Рис. 15.5. Это сообщение об ошибке выводится, если применяется макрокоманда **ОткрытьТаблицу** без заполненного обязательного аргумента **Имя таблицы**

Несмотря на то, что сообщения об ошибках макросов очень информативны, они не всегда предоставляют всю информацию, необходимую для точного указания проблемы. Например, сообщение об ошибке, приведенное на рис. 15.5, содержит несколько возможных причин -она могла (т. е. ошибка) появиться из-за аварийного завершения одной из команд **ОткрытьТаблицу** (что и произошло в данном примере), **ПреобразоватьТекст** или **ПреобразоватьЭлектроинуюТаблицу.** И даже если вы знаете, что виновата макрокоманда ОткрытьТаблицу, это знание не поможет вам, если в одном макросе эта команда вызывается несколько раз.

Для диагностики проблем можно использовать отладку - программное средство, позволяющее поместить ваш макрос под микроскоп и рассмотреть, что происходит на самом деле, Тип отладки, предоставляемый программой Access для макросов, называется *пошаговой отладкой,* поскольку позволяет поочередно тестировать команды макроса. Таким образом, вы узнаете, где именно возникает ошибка.

Для применения пошаговой отладки выполните следующие действия. 1. Откройте макрос в **Конструкторе.**

Все новые макросы начинают с режима **Конструктор.** Если вы хотите тестировать макрос, созданный ранее, найдите его в области переходов, щелкните правой кнопкой мыши и выберите команду **Конструктор.**

2. Выберите на ленте **Работа с макросами | Конструктор → Сервис → По шагам** (Macro Tools | Design → Tools → Single Step).

По шагам (Single Step) - это кнопка-выключатель, т. е. она при выделении подсвечивается. После того как вы щелкнули эту кнопку, она должна быть подсвечена. (Если этого не произошло, значит, пошаговая отладка была уже включена, и вы ее только что выключили. Щелкните мышью кнопку **По шагам** еще раз, чтобы снова включить этот режим.)

3. Выберите на ленте **Работа с макросами | Конструктор → Сервис → Выполнить.**

Ваш макрос начнет выполняться. Но появилось отличие. Перед каждым действием программа Access выводит на экран важную информацию в окне **Пошаговое исполнение макроса.** На рис. 15.6 показано, как оно функционирует.

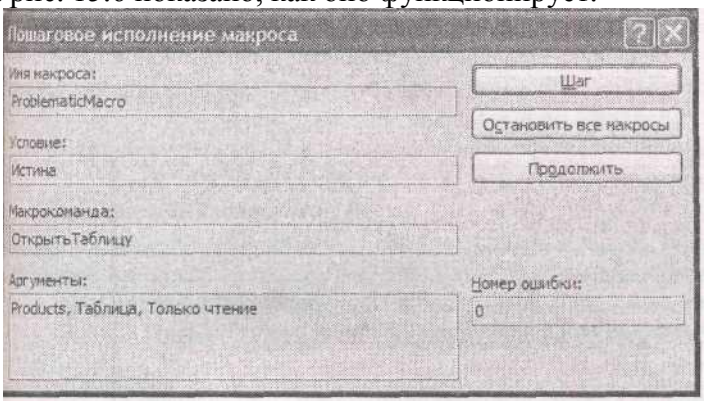


Рис. 15.6. Это окно сообщает о том, что вы выполняете макрос, названный **ProblematicMacro.** Следующий шаг - выполнение макрокоманды **ОткрытьТаблицу** с аргументами, отображенными в области **Аргументы.** (Не обращайте внимания на поле **Условие,** поскольку вы пока не научились выполнять макросы с условием)

4. В зависимости от того, что вы хотите делать дальше, щелкните мышью одну из кнопок: **Шаг, Продолжить** или **Остановить все макросы.**

□ Кнопка **Шаг** выполняет команду. Если действие завершается успешно, программа Access снова приостанавливает макрос и выводит окно **Пошаговое исполнение макроса** с информацией о следующей команде. Именно поэтому описываемый процесс называется пошаговым и позволяет выполнять в каждый момент времени одно действие. Если вы щелкните мышью кнопку **Шаг** и команда завершится аварийно, то увидите информацию об ошибке, как показано на рис. 15.7.

□ Кнопка **Продолжить** выключает режим пошагового исполнения и выполняет оставшуюся часть макроса без прерываний. Если возникает ошибка, она приводит к завершению исполнения и выводу сообщения об ошибке, как в предыдущем варианте.

Рис. 15.7. Когда возникает ошибка, вы не можете двигаться дальше. В окне **Ошибка выполнения макрокоманды** программа Access выводит номер ошибки для вашей проблемы (который полезен, если нужно искать помощь в интерактивной базе знаний корпорации Microsoft), и не разрешает вам идти дальше. Вы вынуждены щелкнуть мышью кнопку **Остановить все макросы,** устранить проблему и затем пробовать снова

□ Кнопка **Остановить все макросы** завершает исполнение макроса досрочно. Слово "все" в названии кнопки означает, что если одновременно выполняется несколько макросов, Access завершит их все. Вы можете создать макрос, который запускает другой макрос. Если вы прекращаете процесс с описанной последовательностью действий, оба макроса будут завершены досрочно.

Примечание

Режим пошагового исполнения влияет на все макросы, включая те, которые созданы с помощью мастера построителя макрокоманды. Не забудьте отключить его, когда закончите тестирование своего макроса. В противном случае появится окно **Пошаговое исполнение макроса,** когда вы применяете макрос, работающий совершенно правильно.

Практические занятия для опытных пользователей.
Обработка ошибок макроса

В макросах вы сталкиваетесь с двумя типами ошибок. Во-первых, существуют ошибки, которые сделаны в процессе создания макроса. С помощью пошаговой отладки вы можете локализовать их и исправить. Во-вторых, есть ошибки, которые возникают, когда макрос применяется в неверном контексте. Возможно, нужных вам данных нет в текущей записи или форма, которую вы пытаетесь использовать, не открыта. Устранить такого сорта ошибки не удастся с помощью изменения макроса, но можно сообщить программе Access, как поступать с ними.

Обычно Access прерывает выполнение макроса при возникновении ошибки. Если вы хотите применить другой подход, начните ваш макрос с макрокоманды **ПриОшибке** (OnError). Команда **ПриОшибке** выбирает в зависимости от значения аргумента **Перейти** (GoTo) один из трех вариантов обработки ошибок. Задайте его значение равным *Сбой* (Fail) для реализации стандартного поведения. Задайте в этом аргументе значение *Далее* (Next), и программа Access пропустит вызывающую сбой макрокоманду и выполнит следующую команду из списка.

Если же задать в аргументе значение *Имя макроса* (Macro Name), Access будет перемещаться к концу списка, пока не найдет заданный макрос. (С помощью значения *Имя макроса* вы сообщаете программе, к какому макросу перейти.) Как действует задание имен макросов в аргументе, вы узнаете в *разд. "Группы макросов" далее в этой главе.*

ПриОшибке (OnError) - необычная макрокоманда, поскольку она воздействует на весь остальной макрос (или, по крайней мере, до тех пор, пока не встретится еще одна макрокоманда **ПриОшибке).** В длинном сложном макросе макрокоманда **ПриОшибке** может вызываться несколько раз. Но применяйте варианты обработки ошибок с осторожностью, чтобы не вызвать дополнительные проблемы. Во многих макросах одна макрокоманда зависит от другой, поэтому лучше всего прервать его выполнение при первом же сигнале тревоги.

15.2. *Макросы и безопасность*

В последние годы сотрудники корпорации Microsoft помешаны на безопасности. Они стали более требовательны к программам пакета Office, таким как Access, стремясь нейтрализовать разработчиков злонамеренных вирусов. И хотя эти изменения сделали программу Access безопаснее, они же стали помехой при использовании определенных типов макросов.

15.2.1. *Опасные макрокоманды*

Программа Access различает два типа макросов: те, которые всегда безвредны, независимо от метода их использования, и те, в которых есть возможность злоупотреблений. Макрокоманда **ОткрытьТаблицу** безопасна. Она может открыть таблицу, которую вы совсем не хотите видеть, но не может нанести реальный вред. С другой стороны, макрос **Печать** (PrintOut) не столь невинен. Злоумышленник может отправить на принтер 400 копий ваших данных с кеглем шрифта 80 пунктов. Аналогичным образом **УдалитьОбъект** (DeleteObject) может привести к реальному хаосу в вашей БД и команда **ЗапускПриложе-ния** (RunApp), безусловно, опасна - она может запустить новейшее шпионское программное обеспечение или установить компьютерный вирус.

В окне создания макроса в раскрывающемся списке приведены только стопроцентно безвредные макрокоманды. Их называют *безопасными макрокомандами.* Конечно, существуют серьезные причины применения потенциально опасных макросов. Возможно, вам действительно нужно напечатать отчет, удалить объект или выполнить другую программу. В этом случае вы вынуждены применять потенциально опасные макрокоманды - команды, которым программа Access не доверяет безоговорочно.

Примечание

До тех пор пока вы один управляете вашей БД, вы знаете, что она не содержит стороннего кода и злоупотреблений. В этом случае нет серьезных причин отказываться от применения потенциально опасных макрокоманд. Но если кто-то послал вам БД по электронной почте или вы загрузили БД из Web, возможно, уверенности у вас поубавится. По этой причине программа Access автоматически блокирует небезопасные макросы в БД до тех пор, пока вы не зададите ей что-то другое. Вы узнаете больше об этом механизме в следующем разделе.

Для того чтобы увидеть полный список макрокоманд, включая те, которые программа Access считает опасными, создайте новый макрос (или откройте уже имеющийся) и выберите на ленте **Работа с макросами | Конструктор → Показать или скрыть → Показать все действия** (Macro Tools | Design → Show/Hide → Show All Actions). Теперь раскрывающийся список макрокоманд содержит несколько больше возможных вариантов. Когда во время создания макроса выбирается опасная макрокоманда, Access дает вам знать об этом с помощью предупреждающей пиктограммы (рис. 15.8).

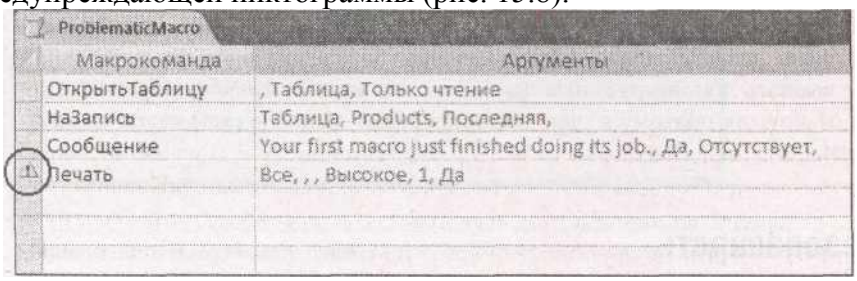


Рис. 15.8. Восклицательный знак в треугольнике выделяет макрокоманды, которые программа Access может отказаться выполнять. В данном примере макрокоманда **Печать** может вызвать проблемы

Примечание

У программы Access нет представления о том, какие макрокоманды более, а какие менее опасны. Она просто делит их на безопасные и опасные.

На профессиональном уровне.

Макрокоманды, которым Access не доверяет

Ниже приведен перечень наиболее распространенных **потенциально опасных** макрокоманд.

- *Удаление объекта.* Очевидно опасное действие.
- *Печать объекта.* Неизвестно, сколько бумаги вам понадобится для печати.
- *Копирование объекта.* Злоумышленник может использовать это действие для создания макроса, который заполнит вашу БД.
- *Сохранение объекта.* Это действие может показаться совершенно невинным, но его легко объединить с другими действиями для создания макроса, изменяющего объект БД и впоследствии сохраняющего поддельную версию объекта.
- *Копирование файла вашей БД.* В конечном счете, это действие может перезаписать копию, которую вы уже сделали, или заменить какой-либо важный файл. Экспорт данных считается столь же рискованным.
- *Раскрытие окна на весь экран, сворачивание окна на панель задач и перемещение окна.* Возможно, корпорация Microsoft перестраховывается, решив не доверять подобным макрокомандам, которые позволяют менять местоположение форм и других окон. В любом случае эти макрокоманды используются нечасто в Access 2007, поскольку они не применимы в случае окон со вкладками, а могут использоваться только в случае редко применяемых свободно плавающих окон.
- *Выполнение команд SQL.* Как вы узнали из *разд. "Режим SQL" главы 6,* SQL - это язык, лежащий в основе запросов программы Access. Вы можете применять непосредственно команды SQL для выполнения практически любой задачи в вашей БД, начиная от удаления десятков записей и заканчивая созданием новой таблицы.
- *Выполнение VB-кода.* Несмотря на то, что это действие не помечается пиктограммой с восклицательным знаком, Access считает весь VB-код опасным. Вы узнаете больше об этом в

главе 16.

• *Пересылка случайных нажатий клавиш.* Макрокоманда **КомандыКлавиатуры** (SendKeys) позволяет пересылать поток клавиатурных кодов в активное в данный момент окно. Вы можете сделать все что угодно, и в этом проблема. Респектабельные пользователи Access вообще избегают применения команды КомандыКлавиатуры, поскольку она содержит ошибки. (Фатальные проблемы возникают, если щелкнуть кнопкой мыши во время выполнения макроса, и приводят к направлению клавиатурных кодов совсем не в то окно, для которого они предназначались.)

Некоторые макрокоманды могут рассматриваться как опасные с учетом применяемых в них аргументов.

• *Завершение программы Access.* Access позволяет выполнить, обычную макрокоманду **Выход** (Quit), в которой пользователю, работающему с БД, предлагается сохранить изменения, отменить их или отказаться от требования выхода из программы. Но макрокоманду **Выход** можно настроить так, что она будет закрывать программу немедленно без предложения сохранить что бы то ни было (или закрывать немедленно и сохранять все неподтвержденные изменения). Если использовать один из этих вариантов, Access трактует команду как опасную.

• *Отправка электронной почты.* Эта макрокоманда считается опасной, если вы не разрешаете пользователю макроса перед отправкой подтвердить корректность электронного сообщения или отказаться от него.

15.2.2. *Как Access обрабатывает опасные макросы*

Вы уже знаете разницу между опасными и безопасными макрокомандами, но вы еще не рассматривали действия программы Access при столкновении лицом к лицу с рискованными действиями. В предыдущих версиях Access на экран выводился поток предупреждающих сообщений. Программа Access 2007 решает проблему самостоятельно, незаметно отключая опасные макросы при любом открытии файла БД.

Как вы уже, безусловно, заметили, когда открывается БД, программа Access отображает сообщение системы безопасности, показанное на рис. 15.9. Оно предупреждает о том, что Access отключает любые потенциально опасные части вашей БД.

Примечание

Панель с сообщением можно скрыть. Если полагаете, что программа Access отключила некоторые макросы, но не хотите видеть панель сообщений, выберите на ленте **Работа** с **базами данных → Показать или скрыть → Панель сообщений** (Database Tools → Show/Hide → Message Bar).

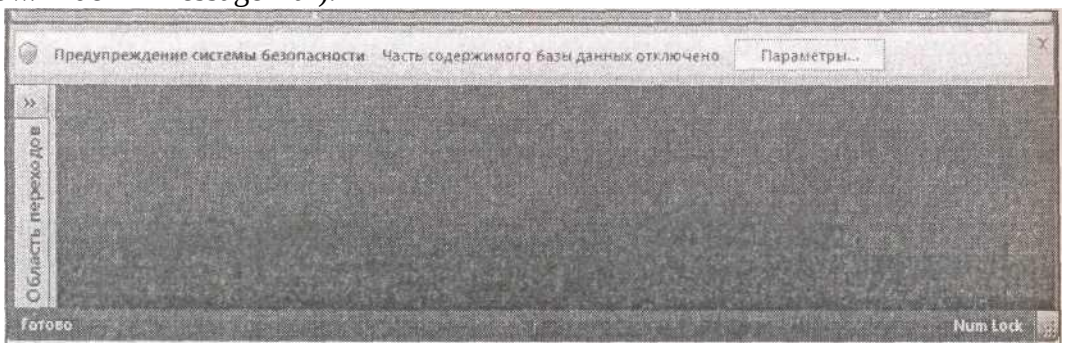


Рис. 15.9. На панели сообщений выводится угрожающее предупреждение. Для повторного включения опасных макросов в вашей БД щелкните мышью кнопку **Параметры** и затем в появившемся окне (см. рис. 15.10) выберите переключатель **Включить это содержимое.** Разрешение, которое вы установите, продлится до тех пор, пока БД будет открыта, поэтому выбирать переключатель **Включить это содержимое** придется при каждом открытии БД. Можно также щелкнуть кнопкой мыши ссылку **Открыть центр управления безопасностью** (показанную на рис. 15.10) для того, чтобы настроить параметры безопасности на более длительный период

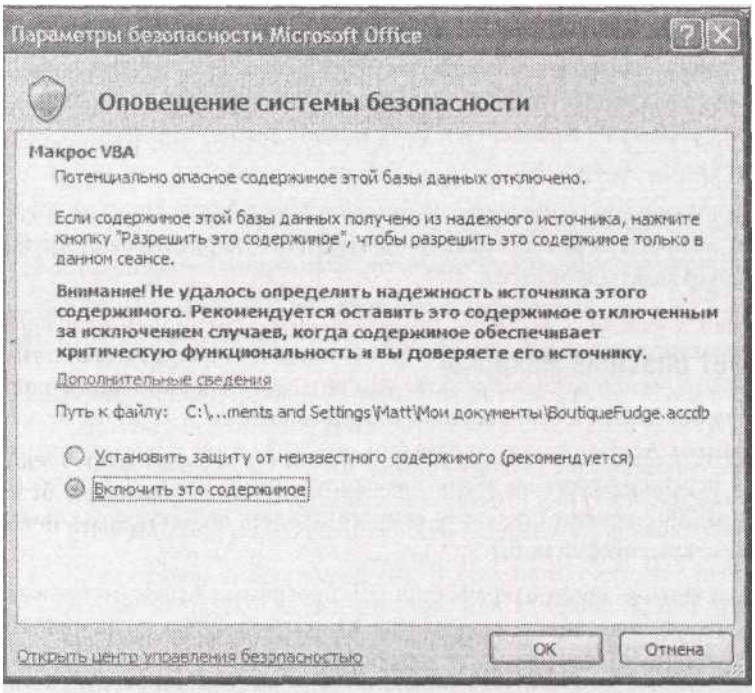


Рис. 15.10. При щелчке мышью кнопки **Параметры** (см. рис. 15.9) программа Access растолковывает проблему с помощью этого слегка замысловатого поля **Сообщение.** Выберите переключатель **Включить это содержимое,** щелкните мышью кнопку ОК и спокойно действуйте дальше

Вся эта суета вокруг опасных и безопасных макросов кажется лишней при условии, что можно включить все ваши макросы и возобновить нормальную работу быстрым щелчком кнопкой мыши переключателя **Включить это содержимое.** Но жизнь не простая штука, и вот почему.

■ Несмотря на то, что вас могут не беспокоить несколько появившихся ненужных предупреждающих сообщений, другие пользователи могут оказаться более доверчивыми. Они увидят сообщение системы безопасности и дважды подумают, что означает для них невозможность использования всех функциональных средств вашей БД. Они могут и просто не понять вопрос или не осознать необходимость щелкнуть кнопкой мыши переключатель **Включить это содержимое.**

■ В корпоративной среде системный администратор может настроить программу Access так, что она вообще не будет отображать предупреждение системы безопасности. Ваши макросы будут тихо отключены, и пользователь, работающий с вашей БД, даже не поймет, почему некоторые средства не работают.

■ Щелчок в тысячный раз переключателя **Включить это содержимое** может вывести из равновесия. Уж поверьте.

15.2.3. *Центр управления безопасностью*

Так вы не хотите видеть панель сообщений при каждом открытии БД? Программа Access предоставляет три способа, облегчающих работу с БД, содержащими опасные макросы.

■ Можно понизить уровень безопасности программы Access настолько, что опасные макросы будут разрешены. Такой подход не рекомендуется, т. к. он разрешает выполнение любого кода, включенного в вашу БД. Если случайно вы откроете БД, содержащую код, провоцирующий аварийные ситуации, никакой защиты у вас не будет.

■ Вы можете попросить программу Access доверять файлам БД, хранящимся в определенных папках на вашем компьютере (или на других компьютерах). Это самый удобный метод действия.

■ Вы можете попросить программу Access доверять базам данных, созданным надежным издателем. Этот вариант наиболее безопасен, но для его установки необходимо заплатить другой компании для получения сертификата безопасности. По этой причине только большие компании, у которых полно денег, могут его себе позволить.

Все описанные действия выполняются в одном и том же окне - **Центре управления**

безопасностью (рис. 15.11). Для вывода его на экран в диалоговом окне **Параметры безопасности Microsoft Office** (см. рис. 15.10) щелкните кнопкой мыши ссылку **Открыть центр управления безопасностью.** Или примените следующий, более обходной путь.

1. Выберите кнопку **Office → Параметры Access.**
2. В окне **Параметры Access** выберите **Центр управления безопасностью** (Trust Center).
3. Щелкните мышью кнопку **Параметры центра управления безопасностью** (Trust Center Settings).

Центр управления безопасностью включает шесть разделов.

■ Раздел **Надежные издатели** (Trusted Publishers) позволяет сообщить Access, что следует доверять БД с цифровой подписью определенных людей. Для того чтобы воспользоваться этим средством, вашей компании нужно купить цифровой сертификат у такой компании, как VeriSign **(www.verisign.com).** Затем, когда вы откроете подписанную БД, программа Access свяжется с компанией, выдавшей сертификат, и проверит его правомерность. Если он действителен, все законно, БД надежна и все опасные макросы разрешены. Цифровые сертификаты не рассматриваются в этой книге.

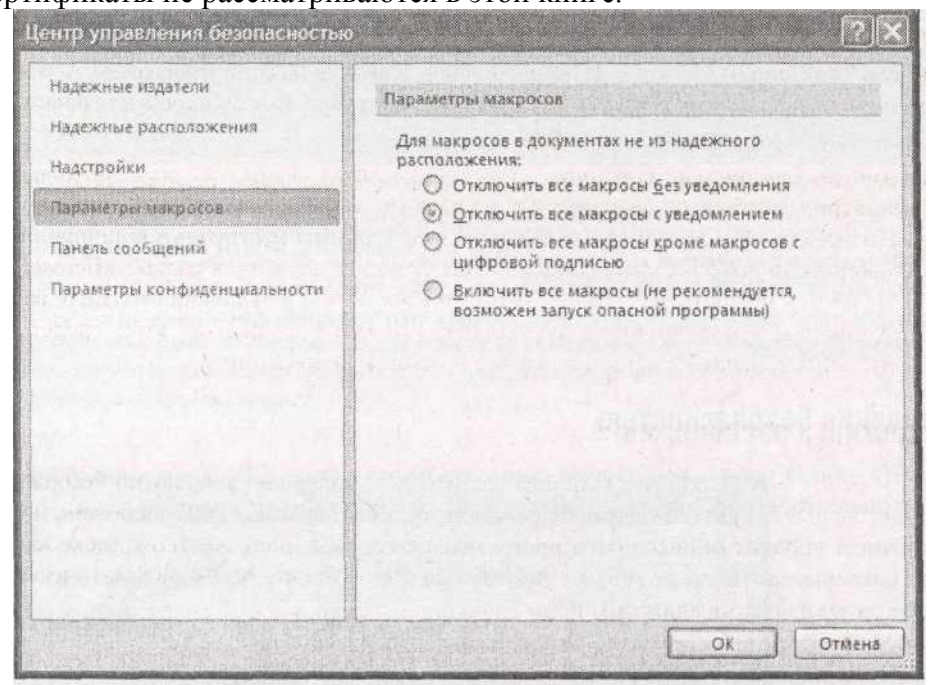


Рис. 15.11. Раздел **Параметры макросов** позволяет задать реакцию Access на опасные макросы. Вы можете выбрать включение или отключение опасных макросов и попросить Access уведомлять вас или не уведомлять об отключении чего бы то ни было

Примечание

Если заниматься раскопками длительное время, можно обнаружить, что у корпорации Microsoft есть средство (называемое makecert.exe) для генерации собственных цифровых сертификатов. Но это только тестовое средство, поскольку созданные с его помощью сертификаты не действуют на других компьютерах. Остерегайтесь - некоторые книги и Web-сайты по Access могут сбить с пути.

■ Раздел **Надежные расположения** (Trusted Locations) позволяет выбрать места на вашем жестком диске для хранения ваших БД. В этом случае Access будет доверять только вашим файлам БД и никому другому. Вы узнаете, как задавать надежное расположение, в следующем разделе.

■ Раздел **Надстройки** (Add-ins) разрешает определить, следует ли поддерживать дополнительные модули (add-ins) (мини-программы, расширяющие функциональные возможности Access), даже если они созданы не подтвержденным издателем. Обычно все дополнительные модули разрешены. (В конце концов, если вы не доверяете какому-либо дополнительному модулю, не устанавливайте его!) Эти установки применяются только в корпоративной среде, где необходима серьезная защита Access для предотвращения даже намека на потенциальную проблему.

- Раздел **Параметры макросов** (Macro Settings) позволяет настроить обработку макросов программой Access. Вы можете сделать ее более строгой (запретив все макросы, если они не принадлежат надежному издателю) или менее (разрешив все макросы, независимо от того, что они могут делать). Лучше всего оставить стандартный выбор **Отключать все макросы с уведомлением.**
- Раздел **Панель сообщений** (Message Bar) позволяет задать, нужно ли программе Access отображать панель сообщений, когда программа блокирует опасные макросы в ненадежной БД.
- Раздел **Параметры конфиденциальности** (Privacy Options) позволяет настроить несколько параметров, вообще не связанных с макросами. Можно задать необходимость проверки в Интернете обновлений файлов справки программы и отправки корпорации Microsoft информации о сбоях при обнаружении проблем (таким образом, Microsoft сможет найти ошибки и узнать, как улучшать программу Access в дальнейшем). Если вас беспокоят интернет-злоумышленники, можно отключить некоторые из этих средств. Чаще всего эти параметры необходимы лишь теоретикам конспирации.

15.2.4. *Задание надежного расположения*

Хорошо было бы иметь возможность различать ваши БД, содержащие абсолютно безопасный код, и все остальные? Для облегчения решения этой задачи в Access 2007 включено новое средство. Оно позволяет определить конкретную папку на вашем жестком диске как надежное расположение. Если открыть БД, хранящуюся в этой папке, Access автоматически будет доверять ей и разрешит опасные макросы.

Примечание

Конечно, вы должны гарантировать невозможность проникновения потенциально опасных БД в надежное расположение. Если вы пропустите их, то при открытии такой БД вы лишитесь всякой защиты. Но это требование вполне приемлемо, т. к. у приверженцев Access и так есть привычка помещать свои БД в отдельную папку.

Далее описаны действия, необходимые для создания нового надежного расположения.

1. Откройте окно **Центр управления безопасностью** (Trust Center).

Если вы его еще не открыли, следуйте указаниям, приведенным в предыдущем разделе.

2. Откройте раздел **Надежные расположения** (Trusted Locations).

Вы увидите окно, в котором перечислены все надежные расположения (рис. 15.12). Сначала в список включается одно расположение: папка ACCWIZ, которую программа Access использует для хранения своего мастера.

3. Убедитесь в том, что сброшен флажок **Отключить все надежные расположения…**

Если флажок установлен, сбросьте его, прежде чем добавлять новое надежное расположение.

4. Если вы хотите доверять папке в сети вашей компании или домашней сети, установите флажок **Разрешить надежные расположения в моей сети.**

Эта установка немного рискованна, поскольку сетевое расположение находится вне вашего контроля. Хакер может вставить зараженную вирусом БД в это расположение незаметно для вас. Но если вы абсолютно уверены в безопасности сети (и другие пользователи, применяющие эту папку, не будут загружать БД из Интернета и помещать их в это расположение), можно не беспокоиться.

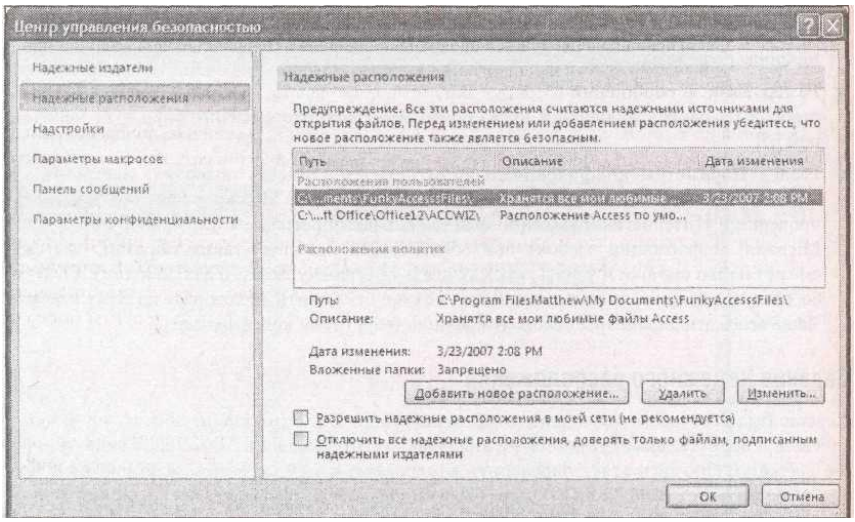


Рис. 15.12. В данном примере добавлено новое надежное расположение для папки FunkyAccessFiles, находящейся на жестком диске в папке My Documents

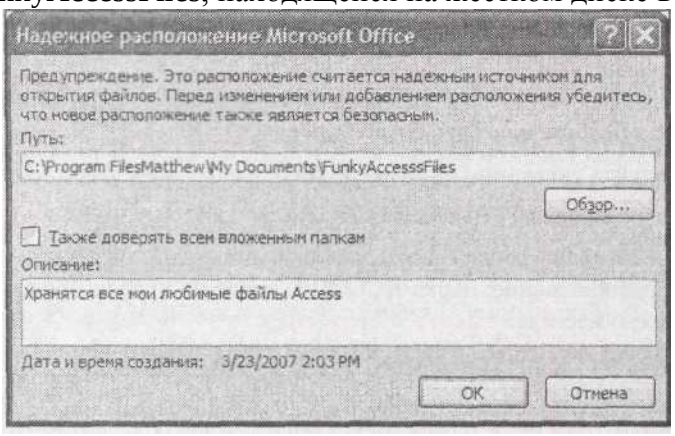


Рис. 15.13. Для настройки надежного расположения необходимо указать путь (щелкните мышью кнопку **Обзор,** чтобы найти нужную папку). Вы также можете решить, доверять ли вложенным папкам и вставить необязательное описание, которое выводится в списке надежных расположений

5. Щелкните мышью кнопку **Добавить новое расположение** (Add new location). Программа Access запросит некоторую дополнительную информацию (рис. 15.13).

6. Для включения нового расположения в список щелкните мышью кнопку **ОК.**

Расположение можно настраивать и удалять в любое время, выбрав его из списка и используя не требующие дополнительных пояснений кнопки **Удалить** и **Изменить.**

15.3. *Три примера макросов*

Вы уже создали базовый макрос, опробовали его и всерьез задумались о безопасности макросов. Пришло время получить вознаграждение за труды и рассмотреть несколько практических способов применения макросов.

Полный перечень макрокоманд содержит множество действий, которые не так уж интересны или относятся только к проектам определенных типов (например, проекты Access, выходящие на БД SQL Server, которые будут рассматриваться в *главе 18).* В следующих разделах представлено несколько наиболее полезных макрокоманд. Вы можете проверить их на загружаемых из Интернета примерах к данной главе или испытать рецепты быстрого приготовления в своей собственной БД.

15.3.1. *Поиск записи*

Макрокоманда **НайтиЗапись** (FindRecord) работает аналогично средству **Поиск** на листе данных, которое обсуждалось в *разд. "Поиск" главы 3.* Все необходимую для поиска информацию вы задаете в аргументах.

Скажем, вы хотите искать слово "hay" (сено) в поле **Diet** (пищевой рацион) таблицы **AnimalTypes.** Далее перечислены макрокоманды, которые вы можете использовать.

■ **ОткрытьФорму** (OpenForm) для открытия формы, отображающей найденную запись (в данном случае **AnimalTypes).** Эту команду можно заменить макрокомандой **ОткрытьТаблицу** (OpenTable) и выполнять поиск на листе данных.

■ **К ЭлементуУправления** (GoToControl) для перехода к полю, в котором будет выполняться поиск (в данном случае **Description).** Если вы хотите применить поиск во всех полях, этот пункт можно пропустить.

■ **НайтиЗапись** (FindRecord) для поиска текста. Вы решаете, начать поиск с первой или с текущей записи, как в данном примере. Можно также выбрать поиск текста в любом месте поля или потребовать, чтобы значение поля целиком совпадало со строкой поиска.

Когда вы соберете все команды вместе, получится нечто похожее на приведенные в табл. 15.1 строки.

Таблица 15.1. Макрос поиска записи

Макрокоманда	*Важные аргументы*[1]	*Описание*
ОткрытьФорму	Имя формы: AnimalTypes	Открывает форму. Если она открыта, переключает в существующее окно
КЭлементу Управления	Имя элемента: *Diet*	Переходит к полю Diet
НайтиЗапись	Образец поиска: ="hay" Совпадение: *С любой частью поля* Только в текущем поле: *Да* Первое вхождение: *Нет*	Находит заданный текст в любой части поля Diet, начиная с текущей записи

[1] Вы можете применять значения по умолчанию и во всех остальных аргументах.

Примечание

Вы могли заметить, что аргумент **Образец поиска** макрокоманды **НайтиЗапись** начинается со знака равенства. Он принимает выражение. В данном примере выражение - не что иное, как текстовая константа, заключенная в кавычки. Но ее можно заменить более сложным выражением, включающим операции, функции и другие сложные средства.

Изюминка этого макроса состоит в том, что его можно использовать несколько раз в строке для поиска нескольких вхождений текста. Если форма **AnimalTypes** уже открыта, этот макрос просто переходит к следующему найденному вхождению текста.

Подсказка

Для большей гибкости можно создать макрос, который использует только команду **Найти Запись.** В этом случае вы сможете искать заданный текст в любом поле и в любой форме или таблице. Конечно, если попытаться выполнять этот макрос без открытых форм или таблиц, команда **НайтиЗапись** не сможет ничего сделать, и вы получите сообщение об ошибке.

15.3.2. *Печать отчета*

Вам нужен полезный макрос, автоматически выдающий часто используемый отчет? Программа Access предоставляет несколько возможностей. Далее приведены две из них.

■ Если вы хотите использовать стандартные параметры печати, можно напечатать отчет с помощью макрокоманды **ОткрытьОтчет** с аргументом **Режим,** имеющим значение *Печать.*

■ Если вы хотите отрегулировать качество печати, задать число копий и номера начальной и конечной страниц, вам нужно применить трехшаговый подход. Начать с команды **ОткрытьОтчет,** применить команду **Печать** (PrintOut) для отправки отчета и закончить командой **Закрыть** (Close) для корректного выхода.

Подсказка

Не пытайтесь применять любой из этих методов с ненадежными БД - Access не разрешит вам их применить.

[2] *Вы можете применять значения по умолчанию и во всех остальных аргументах.*

Приведенная в табл. 15.2 последовательность макрокоманд демонстрирует второй метод. Этот макрос, используя отчет **CheapskateCustomers** (прижимистые клиенты), печатает две копии списка так называемых клиентов, которые на самом деле еще не заказали ни одной вещи.

Таблица 15.2. Макрос печати отчета

Макрокоманда	*Важные аргументы* [3]	*Описание*
ОткрытьОтчет	Имя отчета: CheapskateCustomers	Открывает отчет (но как вы увидите, он появится через пару секунд)
Печать	Число копий: 2	Можно использовать другие аргументы для печати только диапазона страниц или изменения качества. Но нельзя выбрать принтер
Закрыть	Тип объекта: Отчет Имя объекта: CheapskateCustomers	Нет смысла оставлять отчет открытым после того, как он отправлен на печать

После того как программа Access выполнит макрокоманду **Печать,** страницы будут направлены на принтер, заданный по умолчанию на вашем компьютере. У вас нет возможности подтвердить или отменить операцию. Забавы ради можно включить в данный макрос дополнительные шаги для того, чтобы напечатать несколько отчетов одновременно.

Подсказка

Есть еще одна возможность. Можно открыть таблицу или отчет с аргументом **Режим,** имеющим значение *Просмотр* (Print Preview). В этом случае данные на самом деле не посылаются на принтер, но приближают на шаг к этой операции. Это лучший выбор, если вам нужно выбрать принтер, убедиться в корректности данных и проверить объем предназначенной для печати информации. Он также работает с ненадежными БД.

15.3.3. *Отправка данных по электронной почте*

Одна из скрытых возможностей макроязыка программы Access - команда **ОтправитьОбъект** (SendObject) - универсальная команда для отправки сообщений электронной почты.

На профессиональном уровне.

ОтправитьОбъект работает с вашей программой элекронной почты

Макрокоманда **ОтправитьОбъект использует** стандарт, именуемый MAPI (Messaging Application Programming Interface, интерфейс прикладного программирования для электронной почты), т. е. позволяет вам применять любую Windows-программу электронной почты. Неважно, предпочитаете вы Outlook, Eudora, Pegasus или что-то более экзотическое - **ОтправитьОбъект** способна запустить вашу программу электронной почты и использовать ее для отправки сообщения. Если вы не знаете, какая программа используется по умолчанию на вашем компьютере для отправки электронной почты, это легко установить.

[3] *Вы можете использовать значения по умолчанию для всех остальных аргументов.*

Откройте **Панель управления,** выберите пиктограмму **Свойства обозревателя** и щелкните кнопкой мыши вкладку **Программы.** Вы найдете на ней ваш стандартный Web-обозреватель, приложение электронной почты и несколько менее широко используемых приложений, относящихся к Интернету (например, просмотр групп новостей).

Команда **ОтправитьОбъект** на удивление универсальна. Ее можно применять в следующих случаях.

■ *Для отправки по электронной почте объекта БД другому пользователю.* Объект БД преобразуется в другой выбранный вами формат, например электронную таблицу Excel, Web-страницу на языке HTML или даже подготовленный для вывода на печать PDF-файл (если вы установили свободно распространяемый дополнительный модуль "Save As PDF" (сохранить как PDF-файл), описанный в *разд. "Получение дополнительного модуля "ave As PDF'" главы 10).* Объект, который вы хотите послать, задается с помощью аргументов **Тип объекта** и **Имя объекта.**

■ *Для отправки по электронной почте текущего объекта БД.* В этом случае вы получаете неограниченно гибкий макрос, способный отправить любые данные, которые вы просматриваете в настоящий момент. Единственное ограничение - знание типа объекта, который планируется отправить, перед вами полная таблица, запрос, выделяющий важную информацию, или отчет с группировкой и промежуточными итогами. Просто задайте соответствующий тип в аргументе **Тип объекта,** а аргумент **Имя объекта** оставьте пустым.

■ *Для отправки обычного электронного сообщения.* Для этого оставьте пустыми оба аргумента: **Тип объекта** и **Имя объекта.** Вы можете написать сообщение в свойстве **Сообщение** (Message Text). Этот метод удобен для оповещения кого бы то ни было о том, что вы вставили новые данные или внесли значительные корректировки.

Примечание

Команда **ОтправитьОбъект** может отправлять только по одному объекту. Если нужно отправить несколько объектов БД, придется применить ее несколько раз. Для отправки трех отчетов вам понадобятся три электронных сообщения с тремя вложенными файлами. В некоторых случаях вы сможете обойти это ограничение, создав изобретательный запрос, который собирает вместе всю нужную вам информацию и позволяет отправить один комплект результатов.

Самое приятное то, что команду **ОтправитьОбъект** можно использовать в ненадежных БД при соблюдении следующего правила: у аргумента **Изменение сообщения** (Edit Message) должно быть значение *Да.* В этом случае, когда выполняется макрос, у вас есть последняя возможность просмотреть сообщение, изменить любой текст и отменить его отправку, если чем-то не довольны. Если же у аргумента **Изменение сообщения** значение *Нет,* макрокоманда **ОтправитьОбъект** отправляет сообщение, не предоставляя возможности его просмотра и корректировки. Такое поведение считается рискованным, поэтому программа Access не допускает его в ненадежных БД.

В приведенном в табл. 15.3 макросе два запроса с данными о продажах преобразуются в электронные таблицы Excel. Затем они посылаются ведущим руководителям.

Таблица 15.3. Макрос отправки данных по электронной почте

Макрокоманда	*Важные аргументы*[4]	*Описание*
ОтправитьОбъект	Тип объекта: Запрос Имя объекта: MonthlySalesTotals Формат вывода: Excel Workbook (.xlsx) Кому: headhoncho@acme.com Тема: Monthly Update Сообщение: Здесь представлены самые свежие объемы продаж, непосредственно из применяющей макросы БД Access. Вы получите итоги по клиентам в отдельном электронном письме Изменение сообщения: Да	Отправляет сообщение электронной почты руководителю headhon-cho@acme.com с данными из запроса MonthlySalesTotals (месячные итоги продаж), преобразованного в рабочую книгу Excel. Тема сообщения и его текст заданы в аргументах Тема и Сообщение. У вас есть возможность подправить их перед отправкой сообщения. На рис. 15.14 показано это действие
ОтправитьОбъект	Тип объекта: Запрос Имя объекта: CustomerSalesTotals Формат вывода: Excel Workbook (.xlsx) Кому: headhoncho@acme.com Тема: Monthly Update Сообщение: Здесь представлены итоги по клиентам Изменение сообщения: Да	Отправляется второе электронное сообщение руководителю headhon-cho@acme.com с данными из запроса CustomerSalesTotals

Если вы технически грамотны, то можете отправить сообщения по электронной почте одновременно огромному количеству людей. Самый простой вариант - вставить полный список адресов в аргументы **Кому, Копия** или **СК,** разделив адреса точкой с запятой (;). Но лучше применить список рассылки. Этот метод может меняться в зависимости от исполь-зуемой почтовой программы, но в программах Outlook и Outlook Express его использовать легко - просто вставьте имя списка рассылки в поле **Кому.** Если вы создали список, названный FairweatherFriends (друзья хорошей погоды), введите слово FairweatherFriends в аргумент **Кому.**

Подсказка

Не хватает места для редактирования сообщения? Нажмите комбинацию клавиш <Shift>+<F2> во время редактирования свойства **Сообщение** для отображения **Окна ввода** большего размера, в котором видны одновременно несколько строк.

[4] *Вы можете использовать значения по умолчанию для всех остальных аргументов.*

Рис. 15.14. Когда у свойства **Изменение сообщения** значение Да, у вас есть последняя возможность просмотреть (или изменить) сообщение перед отправкой

15.4. *Управление макросами*

По мере создания все большего числа привлекательных макросов возникает необходимость в их организации, гарантирующей, что нужные макросы окажутся под рукой, как только они понадобятся. У программы Access есть несколько инструментов, помогающих справиться с этой задачей, включая группы макросов, объединяющие связанные макросы в один объект для облегчения хранения, и комбинации клавиш для макросов, позволяющие запускать подходящий макрос именно тогда, когда он нужен.

15.4.1. *Группы макросов*

В среднем макрос содержит только от трех до пяти команд. Но средняя БД, применяющая макросы, быстро накапливает десятки макросов. Управлять этими крошечными программами порой трудно, особенно если нужно помнить, что делает каждый из них.

Можно применить группу макросов. Внешне *группа макросов* выглядит как один макрос, поскольку хранится в едином объекте БД. Но группа макросов может содержать неограниченное количество отдельных макросов, у каждого из которых есть свое имя. После того как вы поместили связанные макросы в одну группу, вам будет легче найти нужный макрос, когда придет время его редактировать.

Подсказка

Профессионалы Access применяют группы макросов для группировки в одном месте макросов, используемых в одной форме, работающих с одной таблицей или выполняющих однотипные задачи (например, печать или редактирование записей).

Для создания группы макросов просто создается макрос, использующий столбец **Имя макроса.** Обычно столбец **Имя макроса** скрыт, поскольку все макрокоманды - это компоненты одного и того же макроса. Но при создании группы макросов следует выбрать на ленте **Работа с макросами | Конструктор → Показать или скрыть → Имена макросов** (Macro Tools | Design → Show/Hide → Macro Names) для отображения этого столбца.

Тут есть хитрость. Каждый раз, когда вы начинаете создавать новый макрос, вы вводите его имя в столбец **Имя макроса.** Таким образом, можно накладывать один макрос на другой до тех пор, пока вы помните о необходимости смены имени. Можно даже использовать пустые строки для отделения макросов в группе и облегчения чтения и понимания макросов. Лучший способ понять - посмотреть пример группы макросов на рис. 15.15.

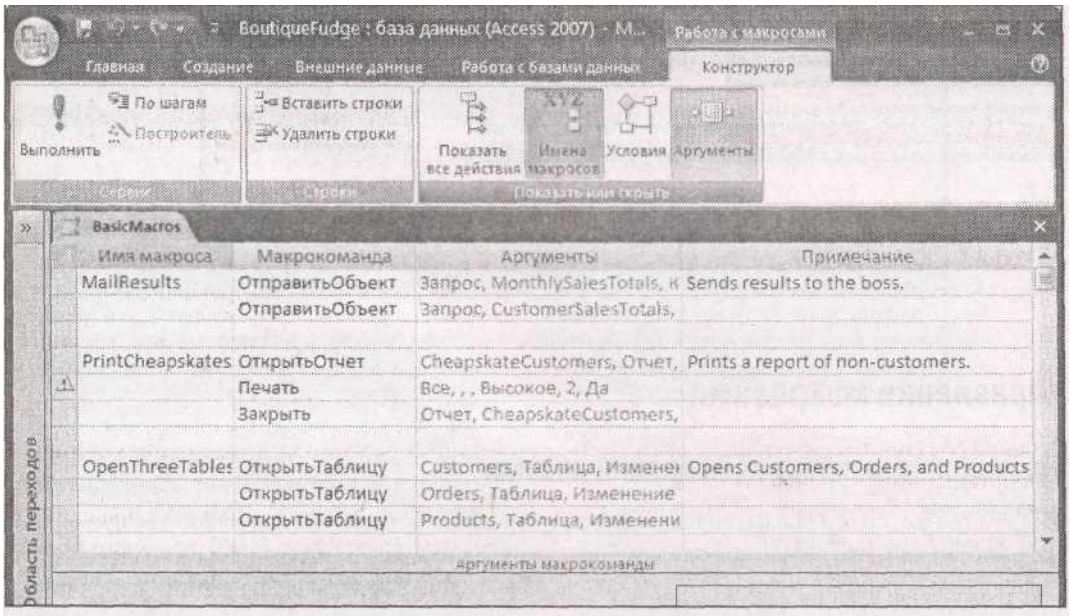


Рис. 15.15. Эта группа макросов объединяет три макроса, перечисляя все их макрокоманды одну за другой. В начале каждого нового макроса в столбце **Имя макроса** появляется его имя. Обратите внимание на пустые строки и многочисленные комментарии. Программа Access игнорирует эти детали

Подсказка

Легче всего применять группы к коротким макросам (макросам, у которых не слишком много макрокоманд). Когда необходимо редактировать макрос, можно использовать удобную команду ленты **Вставить строки** (Insert Rows) для увеличения доступного свободного пространства.

У каждого макроса, включенного в группу, двухчастное имя. Первая часть - имя группы макросов, а вторая часть - любой текст, который вы ввели в столбец **Имя макроса.** Полное имя макроса **PrintCheapskates** (печать прижимистых клиентов) в группе макросов **Basic-Macros** (основные макросы), показанного на рис.. 15.15, - **BasicMacros.PrintCheapskates.** При запуске макроса следует использовать его полное имя.

Одно из ограничений групп макросов заключается в невозможности использования их из области переходов. Если щелкнуть правой кнопкой мыши группу макросов в области переходов и выбрать команду **Выполнить** (Run), программа Access выполнит только первый макрос в группе. Для запуска остальных макросов следует выбрать на ленте **Работа с базами данных → Макрос → Выполнить макрос** (Database Tools → Macro → Run Macro). Затем можно ввести правильное двухчастное имя или выбрать его из раскрывающегося списка (как показано на рис. 15.16).

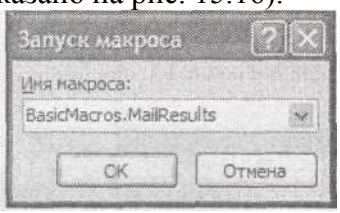


Рис. 15.16. В данном примере программа Access готова к выполнению макроса **MailResults** из группы **BasicMacros**

Примечание

Если вам кажется, что этот способ запуска потребует больших усилий, не волнуйтесь. Большая часть макросов не запускается из области переходов, а связывается с формой, а в этом случае полное имя не приведет к дополнительной работе. Но если у вас есть макрос, который вы определенно хотите запускать из области переходов, группировка макросов в этой ситуации - не ваш путь.

15.4.2. *Назначение макросу комбинации клавиш*

Иногда создается макрос настолько удобный, что хочется иметь его под рукой все время. Сделать это можно, назначив макросу комбинацию клавиш. Затем вместо направления в область переходов вы нажимаете, например <Ctrl>+<M>, и ваш макрос немедленно стартует.

Подсказка

Клавиши очень ценны. Назначайте макросу комбинации клавиш, только если он наверняка будет часто использоваться и с множеством разных форм и таблиц.

Довольно странный способ присваивания макросу комбинации клавиш, принятый в программе Access, состоит в создании еще одного макроса. Этот макрос должен быть назван **AutoKeys** и его единственная задача - присвоить сочетания клавиш другим макросам.

Как же действует макрос **AutoKeys?** Все дело в имени. Когда добавляется макрос в группу **AutoKeys,** вы даете ему специальным образом закодированное имя, реально представляющее собой сочетание клавиш. Когда вы называете макрос ^**M**, программа Access знает, что его надо запускать, когда нажимается комбинация клавиш <Ctrl>+<M>. На рис. 15.17 показано несколько примеров макросов.

Примечание

На рис. 15.17 видно, что каждый макрос в группе выполняет отдельно хранящийся объект-макрос с помощью макрокоманды **ЗапускМакроса.** Такой проект необязателен (можно кодировать каждый макрос непосредственно внутри группы **AutoKeys** с помощью включения в нее всех необходимых макрокоманд), но он улучшает упорядочивание. Он обеспечивает больше гибкости, поскольку можно использовать наборы разных макросов с одними и теми же сочетаниями клавиш и при этом не удалять макросы из группы **AutoKeys.**

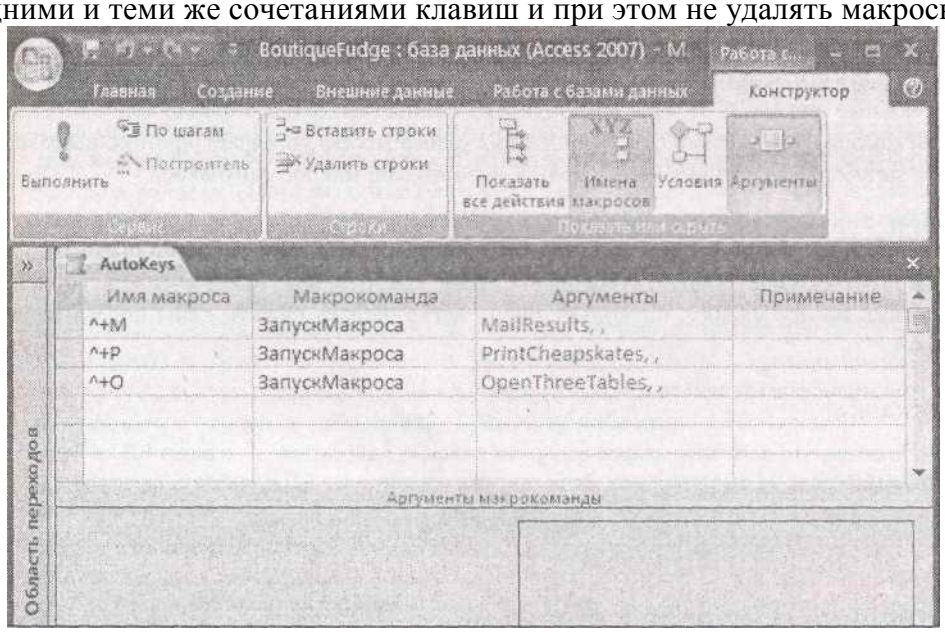


Рис. 15.17. В данном примере группа макросов **AutoKeys** содержит три макроса, которым назначены комбинации клавиш <Ctrl>+<Shift>+<M>, <Ctrl>+<Shift>+<P> и <Ctrl>+<Shift>+<T> соответственно

Единственная особенность применения группы макросов **AutoKeys -** знание правил именования макросов, позволяющих программе Access применять те сочетания клавиш, которые вы назначили. Access разрешает использовать буквы и цифры в комбинации с клавишами <Ctrl> и <Shift>. (Клавиша <Alt> запрещена, поскольку используется при выборе команд на ленте.) Кроме того, можно использовать функциональные клавиши (<F1>- <F12>) и клавиши <Insert> и <Delete>, также в сочетании с клавишами <Ctrl> и <Shift>.

Вот как именуется макрос:

- ^ обозначает клавишу <Ctrl>. Таким образом, ^**M** означает <Ctrl>+<M>;
- **+** обозначает клавишу <Shift>. Таким образом, ^+**M** означает <Ctrl>+<Shift>+<M>;
- **{F1}** обозначает клавишу <F1>. Таким образом, **+{Fi}** означает <Shift>+<Fl>. Все

остальные функциональные клавиши применяются аналогично;

■ **{INS}** обозначает клавишу <Insert> и **{DEL}** обозначает клавишу <Delete>. Таким образом, ^**{INS}** - это <Ctrl>+<Insert>.

Подсказка

Прежде чем вы назначите макросу комбинацию клавиш, следует проверить, не выполняет ли это сочетание клавиш что-нибудь полезное. Ваш макрос переопределяет встроенные команды

Access. Примером может служить сочетание <Ctrl>+<S>, сохраняющее текущий объект. Для снижения вероятности возникновения конфликтов при использовании клавиатурных сочетаний используйте комбинации, включающие клавишу <Shift>, которая применяется реже.

15.4.3. *Настройка макроса запуска*

Неизбежно, время от времени вы будете создавать макрос, который настолько важен, что вы захотите запускать его, как только будет открываться БД. Возможно, этот макрос открывает некоторые важные формы и отчеты, импортирует данные из другого файла или запускает очищающий запрос. Независимо от причины программа Access упрощает процесс выполнения макроса запуска. Все, что нужно сделать, - назвать ваш макрос **AutoExec.**

Access также предоставляет способ обхода макроса **AutoExec.** Если держать нажатой клавишу <Shift> во время первой загрузки БД, программа Access не выполняет макрос **AutoExec** (и не отображает любую стартовую форму, которую вы, может быть, настроили). Но не надейтесь на эту уловку, поскольку очень легко забыть нажать клавишу <Shift> в нужное время.

Подсказка

Помните о том, что если ваш макрос содержит опасные макрокоманды и ваша БД не надежна, программа Access не выполнит его. Если открыть ненадежную БД и выбрать включение опасного содержимого БД с помощью панели сообщений, Access повторно загрузит БД и в этот момент запустит макрос **AutoExec.**

15.5. *Присоединение макросов к формам*

Самые изощренные макросы работают вместе с формами вашей БД. С помощью такого объединения можно создать макрос, который выполняется автоматически, когда что-то происходит (например, когда кнопка щелкается мышью или изменяется запись). Можно также создать гораздо более гибкие макросы, у которых нет фиксированных значений аргументов - вместо этого нужные данные они могут взять из текстовых полей формы.

В следующих разделах вы узнаете, как помочь макросам и формам объединиться.

15.5.1. *Что такое событие*

До настоящего момента вы выполняли макросы тривиальным способом: выискиванием в области переходов нужных вам элементов и затем запуском их вручную. Но в хорошо спроектированной БД макросы редко играют на публике. Чаще они остаются за кадром, пока не засверкают в действии. Можно создать макрос, который запускается при щелчке кнопки мышью, открытии формы или внесении изменения в текстовое поле. Эти побудительные причины называют *событиями.*

У формы есть три типа событий.

■ *События элементов управления.* Эти события особенно полезны. Они происходят, когда вы что-то делаете с элементом управления. Например, когда вы щелкаете кнопку мышью, возникает событие **Нажатие кнопки** (On Click). (Подходящий момент для выполнения любой макрокоманды.) Когда изменяется значение в элементе **Поле,** возникает событие **Изменение** (On Change). (Самое время проверить, имеет ли смысл введенное значение с помощью удачно подобранного условия на значение.) Как вы увидите, большинство названий событий на английском языке начинаются со слова "On".

Примечание

У многих элементов управления одинаковые события. Если у вас на форме два поля и одна кнопка, у всех у них есть событие **Нажатие кнопки** (On Click). Но путаницы не возникает, поскольку программа Access отслеживает, какое событие возникает в каком элементе управления.

События разделов. Как вы узнали раньше, формы имеют разделы, поэтому можно отделить содержимое заголовка и примечания от остальных данных записи. У каждого раздела есть несколько собственных событий, которые происходят, когда вы перемещаете мышь в разделе **(ПеремещениеУказателя** (On Mouse Move)) или щелкаете кнопкой мыши на пустом месте **(НажатиеКнопки** (On Click)). Эти события, как правило, менее полезны для программистов макроса.

События формы. Длинный список более общих, событий относится к форме. Этот список включает события, которые возникают, когда форма открывается впервые **(Открытие** (On Open)) и когда она закрывается **(Закрытие** (On Close)), когда вы переходите от одной записи к следующей **(Текущая Запись** (On Current)) и когда завершается операция над данными, например, обновление **(После обновления** (After Update)).

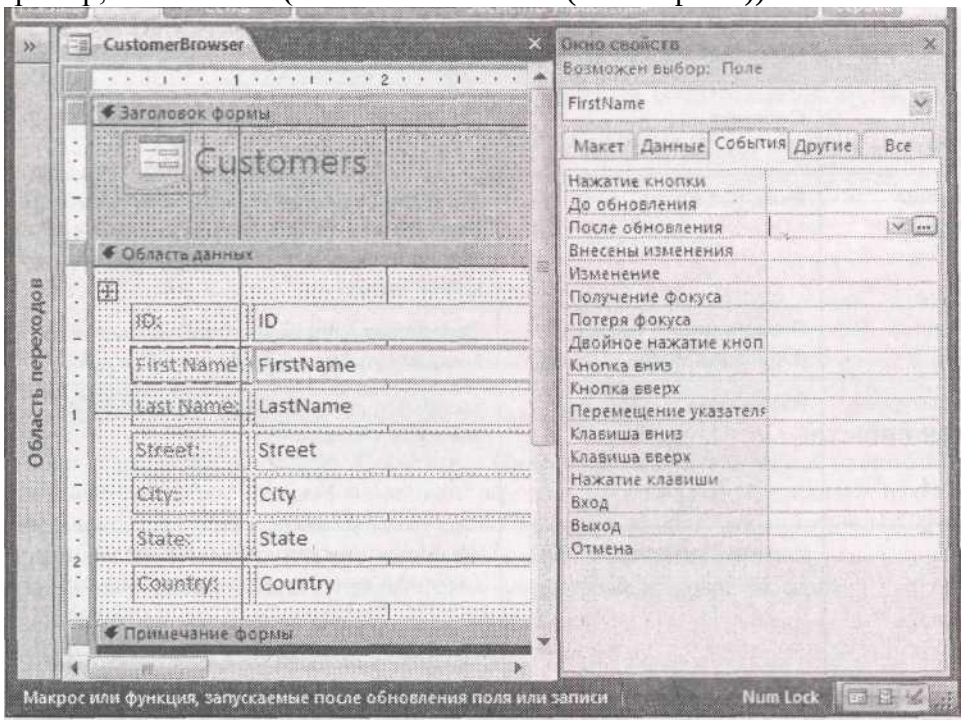


Рис. 15.18. Здесь показаны события для типичного поля ввода. Если щелкнуть кнопкой мыши одно из полей вкладки **События,** в строке состояния, в левой нижней части окна, появляется однострочное описание события. Как можно видеть, сейчас все поля событий пусты, что означает отсутствие присоединенного макроса

Для просмотра событий, относящихся к разным частям формы, выполните следующие действия.

1. Откройте форму в режиме **Конструктора.**

Если на экране нет **Окна свойств,** отобразите его, выбрав на ленте **Инструменты конструктора форм | Конструктор → Сервис → Страница свойств** (Form Tools | Design → Tools → Property Sheet).

2. Выберите элемент, имеющий события, которые вы хотите исследовать.

Можно выбрать отдельный элемент управления, раздел или форму. Если не удается выделить мышью нужный элемент в рабочей области формы, просто выберите его по имени из раскрывающегося списка в верхней части **Окна свойств.**

3. В Окне свойств перейдите на вкладку **События.**

Теперь вы увидите список событий, предоставляемых элементом управления, как показано на рис. 15.18.

Самая большая проблема применения событий - выбор тех, которые нужно использовать.

Если потратить несколько секунд на изучение событий вашей формы, обнаружатся десятки событий, многие из которых редко используются или являются узкоспециализированными. Тут может оказаться очень кстати табл. 15.4 - в ней обращается внимание на события, наиболее полезные для программирования макроса.

Таблица 15.4. События, полезные для программирования макроса

Элемент управления	*Событие*	*Описание*
Все элементы управления	Вход (On Enter)	Возникает, когда вы переходите к элементу управления в первый раз (либо нажатием клавиши, такой как <Tab>, либо щелчком кнопки мыши)
	Перемещение указателя (On Mouse Move)	Происходит, когда мышь перемещается по элементу управления
Любой редактируемый элемент управления	Изменение (On Change)	Возникает, когда изменяется значение в элементе управления
Кнопка	Нажатие кнопки (On Click)	Происходит, когда кнопку щелкают мышью. У других элементов управления тоже есть событие, связанное со щелчком кнопкой мыши, но большинство пользователей привыкло щелкать мышью кнопки, чтобы выполнить какое-либо действие
Поле со списком	Отсутствие в списке (On Not In List)	Возникает, когда вводится значение, которого нет в списке
Форма	Загрузка (On Load)	Происходит, когда форма открывается впервые (и вы можете инициализировать ее)
	Закрытие (On Close)	Возникает, когда форма закрывается. Можно отменить это событие, если хотите оставить форму открытой
	Текущая запись (On Current)	Происходит, когда вы переходите к записи (включая открытие формы и переход к первой записи)
	Внесены изменения (On Dirty)	Возникает, когда вы вносите первое изменение в запись. Сейчас она в режиме редактирования
	Отмена (On Undo)	Происходит, когда вы выходите из режима редактирования и отменяете внесенные изменения (обычно нажатием клавиши <Esc>)
	До вставки (Before Insert), До обновления (Before Update), До подтверждения (Before Del Confirm)	Возникает в процессе вставки, обновления или удаления. Можно отменить это событие, если не нравится то, что видите (например, обнаружили некорректные данные)
	После вставки (After Insert), После обновления (After Update), После подтверждения (After Del Confirm)	Происходит после завершения операции. Вы уже не можете отменить его, но может возникнуть желание отреагировать на изменения выполнением другой задачи или обновлением отображенных данных

Примечание

События обновления, вставки и подтверждения (две последние строки таблицы) также применимы к любому редактируемому элементу управления. Поле использует события **До обновления** и **После обновления** для обозначения момента изменения его значения. В *главе 17* вы найдете пример, применяющий это событие для немедленной реакции на изменение конкретного поля (не дожидаясь, пока будет обновлена вся запись).

Если просмотреть вкладку **События** в **Окне свойств,** то можно найти множество событий, включая такие, которые позволяют реагировать на нажатие клавиши, щелчок кнопки мыши в любом месте или переход от одного элемента управления к другому. Не старайтесь запомнить

все эти варианты прямо сейчас. После того как вы научитесь отвечать на событие, используя макрос, вы сможете обрабатывать практически любое событие.

15.5.2. *Присоединение макроса к событию*

Теперь, когда вы увидели события, которые предлагают формы и элементы управления, пора попробовать присоединить макрос. Основная последовательность действий проста.

1. Создайте и сохраните макрос, как описано в *разд. "Создание макроса "ранее в этой главе.*
2. Откройте вашу форму в **Конструкторе** и убедитесь в том, что на экране отображается **Окно свойств.**
3. Выберите элемент управления, раздел или всю форму.
4. В **Окне свойств** выберите вкладку **События** и найдите событие, которое хотите использовать.
5. В поле **Событие** щелкните кнопкой мыши направленную вниз стрелку и выберите макрос, который хотите использовать.

На рис. 15.19 показан пример.

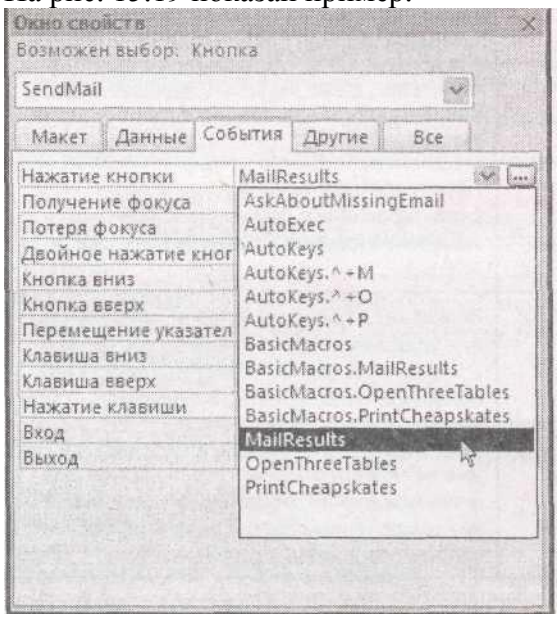


Рис. 15.19. В этом примере к событию **Нажатие кнопки** присоединяется макрос **MailResults.** Теперь при каждом щелчке " кнопки мышью будет выполняться данный макрос

Часто задаваемый вопрос.

Внедренный макрос

Я создал макрос с помощью Мастера кнопок. Как его можно отредактировать?

Когда вы помещаете кнопку на форму, программа Access запускает Мастер кнопок, который создает для вас макрос *(см. разд. "Выполнение действий с помощью кнопок" главы 13).* Мастер кнопок создает внедренные макросы, т. е. макросы, которые хранятся внутри объекта-формы. У этой системы есть несколько достоинств (например, вы можете перемещать вашу форму из одной БД в другую без потери связанных с ней макросов). Но у нее есть и недостатки, а именно: вы не можете редактировать или выполнять макрос независимо.

К счастью, у вас все же есть возможность редактирования внедренных макросов (или просмотра их просто из любопытства). Для этого нужно использовать **Окно свойств.**

Далее описано, как это делается.

1. Выберите элемент управления, который использует макрос (в данном случае, кнопку).
2. Найдите событие с внедренным макросом. В случае кнопки это событие **Нажатие кнопки** (On Click). В поле **Событие** вы увидите текст **[Внедренный макрос] ([Embedded Macro]),** а не имя макроса.
3. Щелкните один раз кнопкой мыши внутри поля **Событие.** Рядом с ним появится кнопка с многоточием (...).
4. Щелкните кнопку с многоточием, чтобы отредактировать макрос в знакомом конструкторе или построителе макроса (macro builder).

15.5.3. *Считывание аргументов из формы*

Раньше, в этой главе, вы видели макросы, которые могут искать записи, печатать отчеты и отправлять данные по электронной почте. Во всех этих случаях аргументы макросов были фиксированными значениями - другими словами, вы их вводили явно и они никогда не менялись. В *разд. "Поиск записи "ранее в этой главе* вы познакомились с макросом, который находил текст "hay". Несмотря на удобство этого макроса, его нельзя применить для поиска другого текста. Если нужно найти другой текст, придется создать целиком новый макрос.

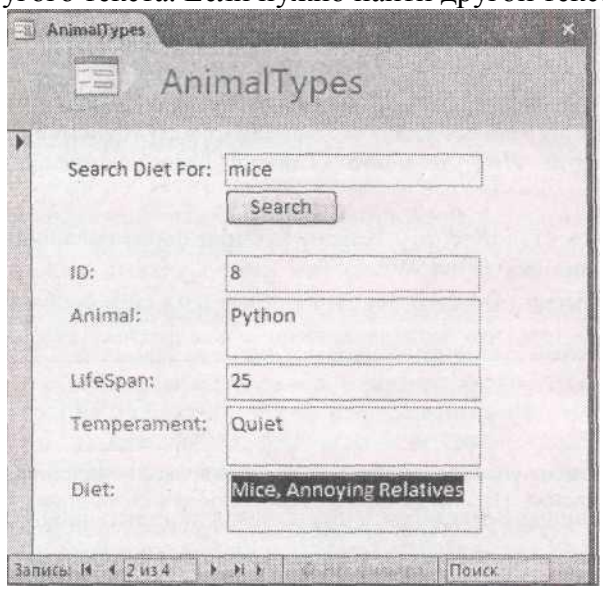


Рис. 15.20. Вместо поиска слова "hay" в этом примере ищется любой нужный вам текст. В чем хитрость? Вы задаете текст для поиска в поле ввода в верхней части формы

Для создания более гибких макросов вместо фиксированного значения можно использовать выражение. Раньше вы неоднократно применяли выражения (см. примеры с запросами в *разд. "Вычисляемые поля" главы 7,* с отчетами в *разд. "Выражения" главы 11* и с формами в *разд. "Выполнение вычислений в выражениях"главы 13),* поэтому у вас не будет проблем при формировании базовых выражений, объединяющих текст, добавляющих числа и использующих функции Access. Но при создании макроса полезнее всего выражения, извлекающие значение из формы. Вам нужно знать только имя элемента управления.

Для знакомства с этим процессом в действии можно просмотреть пример с фильтрацией, приведенный ранее, и создать форму поиска, подобную показанной на рис. 15.20.

Для создания данного примера следует начать с добавления элемента управления **Поле,** необходимого для поиска. Вот как это делается.

1. Откройте форму в режиме **Конструктора.**
2. Выберите на ленте **Инструменты конструктора форм | Конструктор → Элементы управления → Поле** (Form Tools | Design → Controls → Text Box) и нарисуйте элемент **Поле** на форме.
3. После этого выделите его и затем в **Окне свойств** выберите вкладку **Другие.**
4. В верхней части вкладки **Другие** измените свойство **Имя** на что-то более интуитивно понятное, например, **SearchText** (ИскомыйТекст).

Примечание

Не всегда нужно создавать новый элемент управления. Макросы могут считывать значения из любого элемента управления, находящегося на форме, включая те, которые присоединены к записи БД. Но в данном примере нужен способ задания текста, не являющегося частью записи, поэтому для этой цели имеет смысл вставить еще один элемент управления **Поле.**

Теперь можно создавать макрос. Вам больше не нужна макрокоманда **ОткрытьФорму** (OpenForm), которую вы применяли в исходном макросе, поскольку, как вы могли догадаться, программа Access будет запускать данный макрос из уже открытой формы **AnimalTypes** (виды животных). Поэтому вам нужна, прежде всего, макрокоманда **КЭлементуУправления**

(GoToControl) с аргументом **Имя элемента** (Control Name), имеющим значение *Description* (описание).

Вторая макрокоманда - **НайтиЗапись** (FindRecord). Вместо задания фиксированного текста (="hay") в аргументе **Образец поиска** (Find What) вам нужно указать элемент управления **SearchText** с помощью его имени (=SearchText). Если в имени есть пробелы или специальные символы, убедитесь в том, что имя заключено в квадратные скобки (=[SearchText]).

Примечание

Если вы ссылаетесь на поле или элемент управления в текущей форме, вам понадобится только имя поля или элемента управления. Но иногда бывает нужно сослаться на элемент управления в другой форме. В этом случае придется применить причудливый синтаксис, обозначающий имя формы и имя элемента управления. Если вы хотите сослаться на элемент с именем **SearchText** на форме, названной **SearchForm,** следует написать =Forms!SearchForm!SearchText.

После того как макрос отшлифован, последний шаг - вставка кнопки в форму **AnimalTypes** для запуска макроса. Далее приведены необходимые действия.

1. Выберите на ленте **Инструменты конструктора форм | Конструктор → Элементы управления → Кнопка** (Form Tools | Design → Controls → Button) и нарисуйте кнопку на форме.
2. Нажмите клавишу <Esc> для отказа от запуска Мастера кнопок.
3. В **Окне свойств** выберите вкладку **События.**
4. Щелкните кнопкой мыши направленную вниз стрелку в поле события **Нажатие кнопки** (On Click) и затем из списка выберите только что созданный макрос.
5. Теперь выберите вкладку **Макет** и в поле **Подпись** введите слово *Search* (Поиск). Этот пояснительный текст появится на кнопке.

Данный шаг завершает пример. Для его проверки перейдите в **Режим формы,** введите что-нибудь в поле **SearchText** и щелкните мышью кнопку **Search.** Вы перейдете сразу к записи, в которой есть искомый текст.

15.5.4. *Изменение свойств формы*

Значения свойств формы можно не только читать, но и изменять. Хитрость заключается в применении макрокоманды **ЗадатьЗначение** (SetValue). Это очень мощная команда, поскольку она способна изменить любое свойство элемента управления. Ее можно применять для изменения текста в элементе управления, скрытия элемента, изменения его форматирования и т. д. (Дополнительную информацию о свойствах разных элементов управления, которые вы могли бы использовать, *см. в главе 13.)* Единственная загвоздка - программа Access считает команду **ЗадатьЗначение** опасной, поэтому не разрешит выполнить ее в ненадежной БД *(см. разд. "Как Access обрабатывает опасные макросы "ранее в этой главе).*

У макрокоманды **ЗадатьЗначение** только два аргумента. Первый аргумент, **Элемент** (Item) обозначает то, что вы хотите изменить. Можно изменять форму, раздел, поле или элемент управления. Второй аргумент **Выражение** (Expression) задает новое значение. Можно использовать константу или считать нужное значение из другого элемента управления с помощью выражения.

Если вы хотите создать макрос, стирающий текст поиска из поля **SearchText,** нужно вставить макрокоманду **ЗадатьЗначение** и задать в свойстве **Элемент** значение *SearchText,* а в свойстве **Выражение -** "" (обозначает пустую текстовую строку).

Примечание

В данном примере считается, что вы применяете макрокоманду **ЗадатьЗначение** к текущей форме (например, нажатием кнопки **Clear** (Очистить)). Если вы запускаете макрос из области переходов, необходимо **SearchText** заменить полным именем Forms ! AnimalTypes ! SearchText, которое сообщает программе Access, какую именно форму вы используете.

Если вам понравилась команда **ЗадатьЗначение,** возможно, вас заинтересует и связанная с

ней макрокоманда **ЗадатьСвойство.** Эта команда изменяет одно из свойств элемента управления. (Вы выбираете, какое свойство изменить, с помощью аргумента **Свойство.)** Команду **ЗадатьСвойство** можно использовать для изменения цвета элемента управления, его положения или подписи. Но чаще всего макрокоманду **ЗадатьСвойство** применяют для изменения свойств **Включено** (Enabled) (для блокировки элементов управления, которые нельзя редактировать) или **Видно** (Visible) (для скрытия второстепенных элементов управления). Обоим свойствам можно задать значения *Истина или Ложь.*

Главное достоинство команды **ЗадатьСвойство** заключается в том, что программа Access всегда считает ее безопасной. Единственный недостаток состоит в том, что Access не разрешает задавать свойство **Текст** (Text) элемента управления, поскольку его можно использовать для модификации таблицы.

15.6. *Макросы с условиями*

Все макросы, которые вы видели до настоящего момента, были линейные. Они выполняли свои макрокоманды по порядку от начала к концу. Если это кажется немного однообразным, что же, так оно и есть. Но ваш макрос вовсе не должен оставаться таким. Вы можете разрешить ему принимать решения и выполнять условные макрокоманды. И в качестве поощрения их очень легко задать.

Для создания макроса с условием необходимо применить столбец **Условие** (Condition). Обычно он не отображается. Для того чтобы сделать его видимым в конструкторе макроса, выберите на ленте **Работа с макросами | Конструктор → Показать или скрыть → Условия** (Macro Tools | Design → Show/Hide → Conditions).

Неудивительно, что именно в столбце **Условие** задаются условия. Условие похоже на выражение, но оно в результате всегда выдает одно из двух значений: *Истина* или *Ложь.* Программа Access проверяет условие и решает с его помощью выполнить или нет соответствующую макрокоманду. (На языке программистов это называется проверкой условия.)

Вот как действует этот механизм.

- если оставить столбец **Условие** пустым (как в случае обычных макросов), программа Access всегда выполняет данную макрокоманду, если она не сбрасывается из-за ошибки;
- если условие задано и оказывается *Истиной,* Access выполняет соответствующую макрокоманду;
- если условие задано и оказывается *Ложью,* Access пропускает макрокоманду и переходит к выполнению следующей команды в списке.

Вкратце - у вас есть способ выполнять макрокоманду только иногда, по мере необходимости.

15.6.1. *Построение условия*

В ходе этого обсуждения возникает интересный вопрос: как сформировать условие? В простейших типах условий сравниваются два разных значения. Вот пример:

```
[ProductName] = "Baloney"
```

Это условие сравнивает текущее значение элемента управления **ProductName** со словом "Baloney" (болонская копченая колбаса). Если в данный момент **ProductName** содержит этот текст (и только его), условие равно *Истине.* Если же **ProductName** содержит что-то еще, результат условия - *Ложь.*

Подсказка

Иногда вы получаете условие прямо противоположное тому, которое хотели создать. В трудный момент всегда можно изменить условие на противоположное, поместив в начало слово Not. Условие Not [ProductName] ="Baloney" равно *Истине,* только если текущий элемент не содержит любимый всеми мясной продукт.

Знак равенства (=) - один из основных компонентов условной логики, но не единственный вариант. Можно использовать знаки операций "больше чем" (>) и "меньше чем" (<) и оператор "не равно" (о). Далее приведено выражение, проверяющее, больше ли числовое поле определенного значения:

```
[Price]  > 49.99
```

Для пущей важности можно добавить в смесь ваши любимые функции Access. (В *главах 4 и 7* описано множество полезных функций.) Далее приведено условие, проверяющее длину поля и возвращающее значение *Истина,* если поле меньше трех символов:

```
Len ( [FirstName] )  <  3
```

Вместо операторов, которые вы видели до сих пор, для создания собственных условий можно применять функцию, дающую результат *Истина* или *Ложь.* Знатоки программирования называют результат, который может быть *Истиной* или *Ложью* и ничем иным, булевым значением в честь британского суперматематика Джоржа Буля.

У программы Access есть лишь несколько функций, возвращающих булевы значения, но самая замечательная из этой звездной плеяды называется IsNull ( ). Как вы узнали раньше, незаполненные поля - это поля, не содержащие никаких данных. Функция IsNull ( ) проверяет, не пусто ли данное поле или элемент управления. Вот как можно ее применять для выявления пропущенной фамилии:

```
IsNull ( [LastName] )
```

Это условие дает в результате значение *Истина,* если в текущем поле **LastName** нет никаких данных.

Описанный прием - основной блок формирования логики проверки (как вы увидите в следующем разделе). Вы используете функцию IsNull () для обнаружения пропущенных данных и предупреждения пользователя, применяющего ваш макрос, о том, что он упустил нечто важное.

В заключение, последний прием, который может применяться в условиях, - сочетание нескольких условий для создания более мощных суперусловий. Для этого есть два ключевых слова, способных помочь вам объединять условия: And (И) и Or (Или).

And требует одновременного выполнения обоих условий, таким образом, делая ваше условие более строгим. Следующее условие возвращает результат *Истина,* только если оба поля и **FirstName,** и **LastName** длиннее трех символов каждое:

```
Len([FirstName])  <  3 And Len([LastName])  <  3
```

Or предоставляет две альтернативы для удовлетворения условия. Следующее условие возвращает результат *Истина,* если поле **FirstName** или **LastName** пусто. Оно возвращает *Ложь,* только если в обоих полях есть текст.

```
IsNull([FirstName])  Or  IsNull([LastName])
```

С помощью этих строительных блоков - условных операторов, функций и ключевых слов Not, And и Or - можно создать множество условий. В следующем разделе вы увидите пример, заставляющий условия работать.

15.6.2. *Проверка данных с помощью условий*

Многие гуру программы Access применяют макросы для предотвращения некорректного редактирования и других подозрительных операций над данными (например, вставок и удалений). Теперь, когда вы понимаете, как написать условие, вы сможете легко разработать проверочную логику такого сорта.

Примечание

Как известно из *главы 4,* у программы Access есть несколько средств, способных помочь защитить данные в вашей таблице, включая маски ввода, условия на значения и подстановки. Следует всегда пытаться применить эти средства прежде, чем обращаться к макрокодированию. Однако существует много типов ошибок, требующих применения макросов. Один общий пример - корректность одного поля, зависящая от значения другого поля.

Рис. 15.21. Когда устанавливается флажок Please notify me about special offers (Пожалуйста, сообщайте мне о специальных предложениях), в поле WantsEmail устанавливается значение Да. Но у этой записи есть проблема - в поле Email нет значения

Первый шаг - реагировать на подходящие события - главным образом события формы **До вставки, До обновления** и **До подтверждения.** Когда они возникают, можно выполнить условия и найти среди них ошибочные. Если вы видите что-то, что вам не нравится, используйте команду **ОтменитьСобытие** (CancelEvent) для полного прерывания процесса и, таким образом, отмены операции вставки, обновления или удаления.

Примечание

У команды **ОтменитьСобытие** нет аргументов - она просто прерывает выполняющийся в данный момент процесс. **ОтменитьСобытие** работает с любым событием, начинающимся со слова "До" ("Before"), что означает готовность реальной операции к выполнению, но еще не само ее выполнение.

Допустим, вы хотите создать простое условие, которое останавливает обновление определенной записи. Рассмотрите форму, показанную на рис. 15.21.

В данном примере пропущенный электронный адрес вызывает серьезную головную боль. Эту проблему можно решить, сделав поле **EmailAddress** обязательным, но вы хотите на самом деле нечто более замысловатое. Когда **WantsEmail** равно *Да,* поле **EmailAddress** не должно быть пустым. С помощью макроса с условием можно реализовать именно эту логику.

Вам понадобится следующее логическое условие:

WantsEmail = Yes And IsNull ([EmailAddress])

Это условие возвращает значение *Истина,* если **WantsEmail** равно *Да* и **EmailAddress** пусто. Когда это выполняется, самое время отменить обновление с помощью макрокоманды **ОтменитьСобытие.**

Примечание

Другого варианта нет. Можно использовать макрокоманду **ЗадатьЗначение** для замены некорректных значений. Но, как правило, лучше дать возможность пользователю, выполняющему обновление, попытаться самостоятельно устранить проблему.

Когда событие отменяется, оно не выполняет откат всех изменений. Отмена события лишь мешает завершению операции. Если изменяется запись и делается попытка перейти к другой записи, возникает событие **До обновления** (Before Update). Если макрос отменяет это событие, программа Access запрещает переходить к другой записи и оставляет вас на прежнем месте. Но текущая запись остается в режиме редактирования со всеми отредактированными значениями. Ваше дело устранить проблему или нажать клавишу <Esc> для полного отказа от попытки

обновления.

Условная макрокоманда **ОтменитьСобытие** - сердцевина многих проверочных макросов. Но вам нужен еще один завершающий штрих: сообщение об ошибке. Иначе пользователь, выполняющий обновление, не догадается, в чем дело. Он будет склонен считать, что Access совсем слетела с катушек.

Для отображения сообщения об ошибке можно использовать макрокоманду **Сообщение** (MsgBox). Понятно, что вы хотите выводить сообщение об ошибке, только если ошибка действительно возникает, поэтому команды **Отменитьсобытие** и **Сообщение** должны включать условия.

У программы Access есть фантастическое средство ускорения, выручающее при необходимости применять одну и ту же макрокоманду дважды. Вместо повторения одинакового условия рядом с каждой макрокомандой (что немного досадно) просто вставьте три точки (...) в столбец **Условие** для каждой следующей макрокоманды. Эти три точки - сокращенный вариант указания программе Access применять условие из предыдущей макрокоманды.

На рис. 15.22 показан законченный макрос, а на рис. 15.23 - выполняющийся макрос.

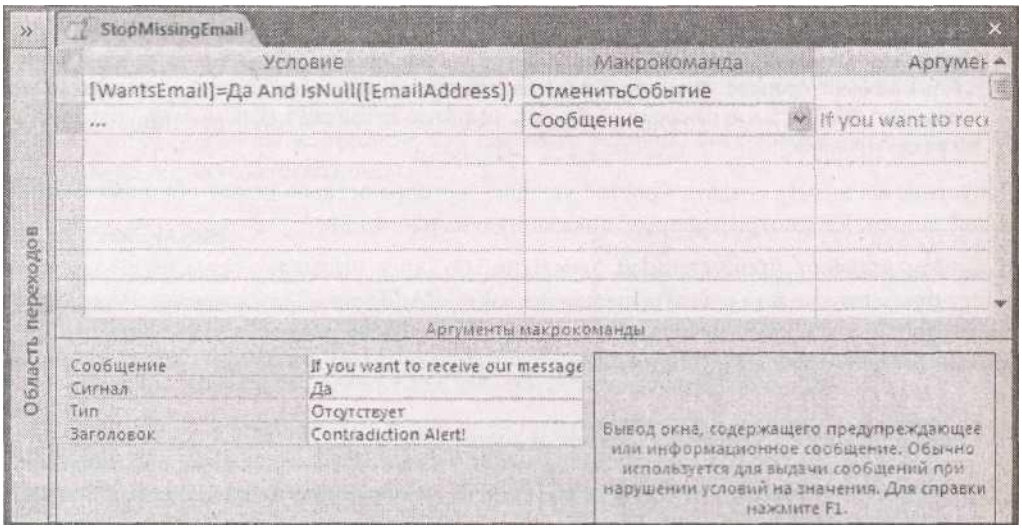


Рис. 15.22. Этот макрос состоит из двух макрокоманд с условием. Для блокировки некорректных данных присоедините его к событиям **До обновления** и **До вставки**

Рис. 15.23. Макрос находит пропущенный электронный адрес и поясняет возникшую проблему

15.6.3. *Макросы с более сложными условиями*

Поскольку макросы с условиями получаются длиннее и сложнее, иногда ими труднее управлять. У вас может быть несколько одновременно действующих условий, каждое из которых требует выполнения собственной макрокоманды. Если некоторые из условий истинны, возможно, вы захотите пропустить проверку всех остальных условий или полностью остановить выполнение макроса.

Для того чтобы увидеть пример типичных проблем, с которыми вы можете столкнуться, стоит вернуться к макросу StopMissingEmail, который рассматривался в предыдущем разделе. Но теперь добавим новый метод. Вместо отмены обновления или вставки ваш макрос будет запрашивать подтверждения заданных вами действий, как показано на рис. 15.24.

Создать сообщение, запрашивающее подтверждение, достаточно легко. Все можно сделать с помощью следующего причудливого условия:

MsgBox("Is this really what you want to do?", 4) = 7

В первой части условия применяется функция MsgBox () для отображения окна сообщения. Число 4 сообщает программе Access о том, что окно сообщения должно включать кнопки **Yes** (Да) и No (Нет). Функция MsgBox () возвращает результат 6, если щелкнуть мышью кнопку Yes, и 7, если щелкнуть кнопку No, таким образом, это условие истинно, только если вы щелкнули мышью кнопку, отменяющую изменения.

Рис. 15.24. Теперь вам решать, применять ли это явно противоречивое обновление: подписаться на получение обновлений по электронной почте и не предоставить адрес. (Возможно, вам действительно нужны регулярные электронные сообщения, но тогда придется вернуться позже и добавить корректный адрес электронной почты)

Примечание

Функцию MsgBox () легко перепутать с макрокомандой **Сообщение** (MsgBox). Они очень тесно связаны. Но этот пример заставляет работать именно функция MsgBox (), поскольку ее можно запустить из условия. Дополнительную информацию о функции MsgBox (например, какие у нее есть еще варианты для отображения разных наборов кнопок) можно получить, щелкнув мышью кнопку **Справка** программы Access и найдя в справке функцию MsgBox.

Итак, вам нужен макрос, который ищет некорректные данные и если находит, отображает на экране окно сообщения. Концептуально этот макрос не слишком сложен. Но если вы начнете создавать нужный макрос, то поймете, что нет удобного способа объединения условий. Найти недопустимые вводимые значения довольно легко, но как гарантировать отображение второго условия (окна сообщения) только при столкновении с первым условием?

Лучший способ решения подобных проблем - применение макрокоманд **Остановить-Макрос** и **ЗапускМакроса. ОстановитьМакрос** завершает выполнение текущего макроса, что очень удобно для выхода из макроса, если известно, что следующие макрокоманды не применяются. **ЗапускМакроса** запускает другой макрос, что облегчает выполнение отдельной задачи при выполнении конкретного условия.

С помощью команды **ОстановитьМакрос** можно завершить выполнение макроса **AskAboutMissingEmail** (вопрос по поводу пропущенного электронного адреса). Вот, что следует сделать для этого.

1. Применить условие для проверки корректности данных. Если данные корректны, не нужно предпринимать дополнительных шагов и можно выполнить макрокоманду **ОстановитьМакрос.**

2. Если макрос все еще выполняется, значит, пропущен электронный адрес. Следующая макрокоманда использует сообщение подтверждения, как условие. Если нажата кнопка No (Нет), выполните команду **ОтменитьСобытие** для прекращения редактирования.

На рис. 15.25 показан окончательный макрос.

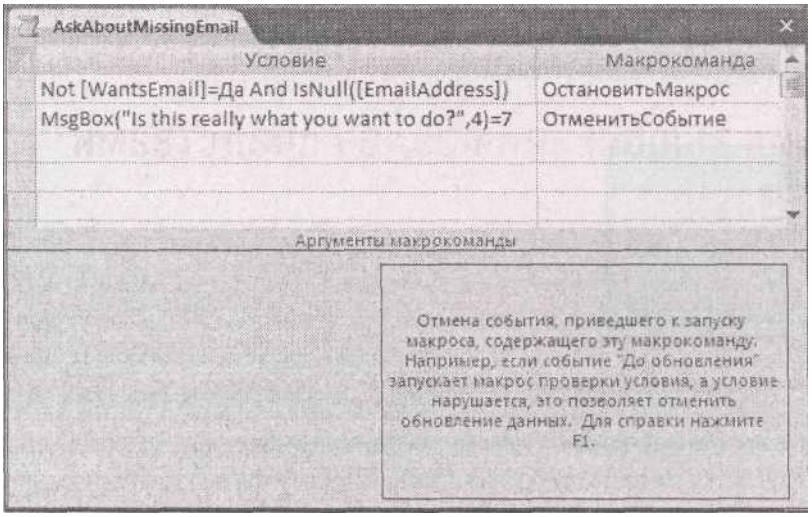


Рис. 15.25. Исправленный макрос CatchMissingEmail (выявление пропущенного адреса) нуждается в двух макрокомандах - одна для прекращения процесса, если все хорошо, и вторая для отмены обновления, если нажата кнопка **No** в окне **Подтверждающее сообщение**

16. Глава 16. Автоматизация выполнения задач средствами языка Visual Basic

Макросы - это конечно замечательная вещь, но они могут только то, что могут. Если нет заранее подготовленной макрокоманды, которая делает то, что вам нужно, применить макрос не удастся. Таких ограничений лишен мир программ на языке Visual Basic (VB), в котором можно выполнять практически все, что угодно (если провести достаточное количество ночных часов за клавиатурой компьютера).

Далее приведены примеры задач, которые можно выполнять с помощью программ, но не с помощью макросов:

- изменить группу записей одновременно;
- выполнить интеллектуальную обработку ошибок так, чтобы программа Access не выводила на экран малопонятные сообщения;
- выполнить сложные вычисления, например, можно вычислить код подтверждения заказа с помощью секретного алгоритма или преобразовать строку текста в Pig Latin ("поросячью латынь");
- написать более сложные процедуры верификации, блокирующие некорректные данные.

В этой (и следующей) главе не ставится задача сделать из вас профессиональных программистов. Если вы мечтаете об этом, придется продолжить обучение и прочесть специальные книги о программировании в Access. Данная глава стремится научить вас основам программирования в Access. Другими словами, вы узнаете достаточно особенностей программирования на VB, чтобы использовать популярные практические приемы программирования, обсуждаемые в следующей главе. Вы также заложите мощный фундамент для дальнейшего изучения.

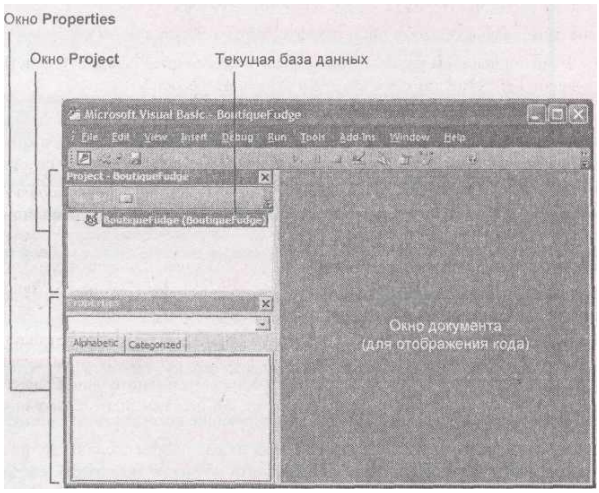
Примечание

Версия языка Visual Basic, применяемая в программе Access и других приложениях пакета Office, называется VBA (Visual Basic for Applications, Visual Basic для приложений).

16.1. *Редактор Visual Basic*

Несмотря на то, что программный код на Visual Basic хранится в вашей БД, вы должны применять специальное средство для его -просмотра и редактирования. Оно называется редактором Visual Basic.

Редактор Visual Basic работает во взаимодействии с программой Access, но открывается в отдельном окне. Для перехода в редактор Visual Basic выберите на ленте **Работа с базами данных → Макрос → Visual Basic** (Database Tools → Macros → Visual Basic). Программа Access запустит отображенное на рис. 16.1 самостоятельное окно, снабженное старомодными меню и панелью инструментов.



Примечание

Закрыть редактор Visual Basic можно в любой момент. Если этого не сделать, программа Access сама закроет его во время завершения работы программы.

Рис. 16.1. В первый момент работы в редакторе Visual Basic все его разделы пусты. В окне **Project** нет программных модулей и не отображается никакой программный код (пока)

Редактор Visual Basic разделен на три основные области. Вверху слева окно **Project** (проект) отображает все *модули* вашей БД. (Каждый модуль - это контейнер для одной или нескольких программных процедур.) Сначала окно **Project** (проект) почти пусто, поскольку еще не создано никакого кода. На рис. 16.1 в окне **Project** (проект) всего один элемент (названный BoutiqueFudge в соответствии с именем текущей БД). Но в этом проекте нет программных модулей.

Примечание

Если в программе Access недавно решались определенные задачи, в окне **Project** (проект) может выводиться проект со странным именем acwzrtool. Это надстройка программы Access, которая предназначается для мастеров, применяемых в программе. Не пытайтесь просмотреть код в этом проекте, Access не разрешит вам сделать это.

Сразу под окном **Project** находится окно **Properties** (свойства), отображающее параметры выделенного в данный момент в окне **Project** элемента, которые можно изменять. В нижней части окна находится окно **Immediate** (отладка) - средство быстрого тестирования, позволяющее выполнить код до помещения его в вашу БД. Все остальное свободное пространство используется для отображения файлов с программным кодом, после того, как вы создали их. Эта область первоначально пуста.

В следующих разделах вы узнаете самый простой способ создания фрагмента программного кода.

1. Сначала вы создадите новый модуль, являющийся контейнером, в который помещается код.
2. Затем в этом модуле вы напишите процедуру с простейшим кодом.
3. Наконец, вы выполните ваш код, чтобы посмотреть на него в действии.

Когда с этим будет покончено, вы узнаете, как код можно включить в формы и отчеты, которые уже есть в вашей БД. (Именно здесь начнется самое интересное.)

16.1.1. *Добавление нового модуля*

Обычно вы будете создавать программные процедуры, которые присоединяются к формам и запускаются, когда возникают определенные события *(см. разд. "Присоединение макросов к формам" главы 15).* Но в этой главе не будем спешить и создадим самостоятельную процедуру, которая запускается по вашему требованию.

Прежде всего, нужно добавить новый модуль для вашего кода. В меню редактора Visual Basic выберите команды **Insert → Module** (Вставка → Модуль). На рис. 16.2 показан результат.

Когда добавляется новый модуль, редактор Visual Basic автоматически открывает окно с кодом, отображающее содержимое данного модуля. (Если в вашей БД несколько модулей, можно открыть нужный вам модуль двойным щелчком мышью его имени в окне **Project.)** Первоначально у новоиспеченного модуля одна строка кода, которая выглядит следующим образом: Option Compare Database

Это инструкция, определяющая, как с помощью языка Visual Basic обработать операции сравнения текстовых фрагментов. Исходно у языка Visual Basic есть собственные правила обработки текста, но приведенный оператор требует вместо них использовать установочные параметры программы Access.

Параметры Access зависят от локализации текущей БД (например, использует ли ваша версия ОС Windows язык Английский (США) (U.S. English) или Японский иероглифический шрифт (Japanese kanji script)). Окончательный результат приведенной строки кода - возможность для англоязычных пользователей применять сравнения без различения строчных и прописных букв. Это означает, что слово "помадка" считается равной слову "пОмАдкА", что аналогично способу обработки текста программой Access при написании вами запросов.

Прежде чем вы начнете писать код, который действительно выполняет какие-то действия, следует в начало вашего кода добавить еще одну инструкцию. Сразу после (или до) строки Option Compare Database вставьте следующую строку:

```
Option Explicit
```

Эта инструкция заставляет Visual Basic применять более строгую проверку ошибок, способную выявлять часто встречающиеся ошибки при использовании переменных *(см. разд. "Хранение информации в переменных" главы 17).*

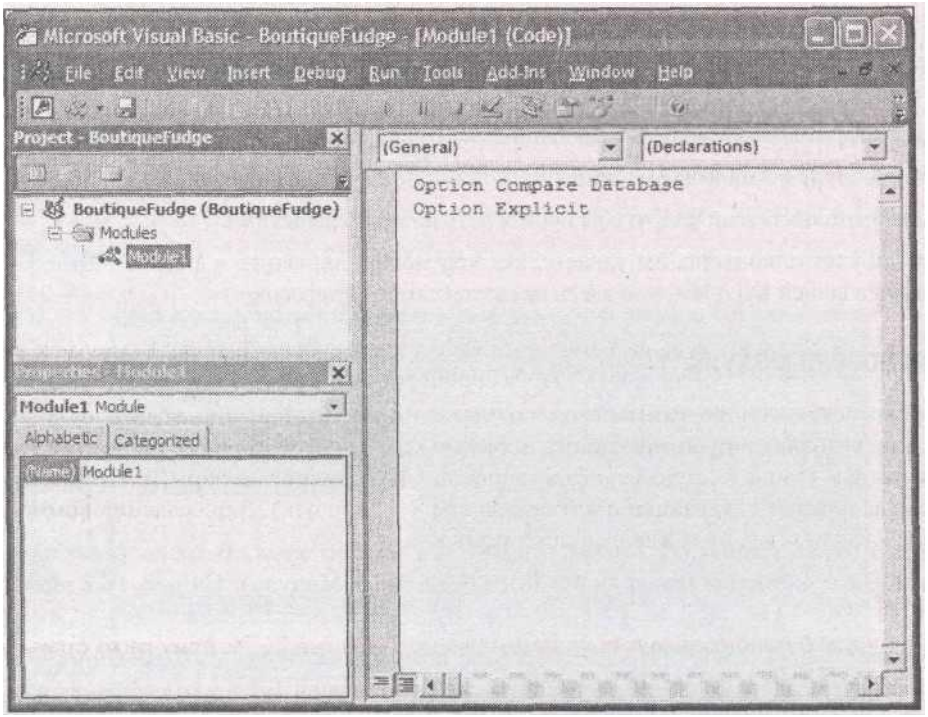


Рис. 16.2. Когда ваш проект включает хотя бы один модуль, окно **Project** отображает группу **Modules.** Программа Access присваивает новым модулям невыразительные имена, например, **Module1**, **Module2** и т. д. Для задания чего-то более осмысленного, выделите модуль в окне **Project** и в окне **Properties,** расположенном сразу под ним, измените свойство **Name. DataCleanupCode** (код очистки данных) - подходящее имя для модуля

Подсказка

Вы можете потребовать от Visual Basic автоматически добавлять строку Option Explicit во все новые файлы с кодом. Для этого выберите **Tools → Options** (Сервис → Параметры), установите флажок **Require Variable Declarations** (требовать объявления переменных) и затем щелкните мышью кнопку ОК. Специалисты Access всегда применяют эту установку.

Как и в случае других объектов БД Access, при закрытии редактора Visual Basic программа Access напоминает о необходимости сохранить вновь созданные модули. Если вы не хотите ждать так долго, выберите последовательность команд File → Save *[DatabaseName]* (Файл → Сохранить [ИмяБазыДанных], где *[DatabaseName]* - имя файла вашей БД).

Примечание

После того как модуль сохранен, его можно увидеть в области переходов окна программы Access. Если используется режим отображения **Таблицы и связанные представления,** ваш модуль выводится в категории **Несвязанные объекты,** если применяется режим вывода **Тип объекта,** модуль появляется в категории **Модули.** Если модуль в области переходов щелкнуть дважды кнопкой мыши, программа Access откроет его в редакторе Visual Basic.

16.1.2. *Написание процедуры с простейшим программным кодом*

Внутри всех модулей (за исключением пустых) находится одна или несколько процедур на языке Visual Basic. *Процедура* - это именованный блок кода, выполняющий определенную

задачу. В языке VB процедуры начинаются словом Sub, за которым следует имя процедуры. Завершаются процедуры оператором End Sub. Далее приведен пример процедуры, которая названа довольно неоригинально MyCodeRoutine:

```
Sub MyCodeRoutine ()
' Здесь располагается ваш код End Sub
```

Этот маленький фрагмент кода на VB иллюстрирует два важных правила. Во-первых, он показывает, как начинать и заканчивать любую процедуру (с помощью операторов Sub и End Sub). В этом коде также показано, как создавать комментарии. *Комментарии* - это специальные операторы, которые программа Access полностью игнорирует. Они не что иное, как заметки для вас самих (например, объяснение на обычном английском (или русском), что на самом деле делает следующая или предыдущая строка кода). Для создания комментария в начало строки вставляется знак апострофа (').

Подсказка

Редактор Visual Basic отображает все комментарии зеленым цветом, поэтому легко отличить код от комментариев.

Сейчас процедура MyCodeRoutine ничего не выполняет. Для того чтобы сделать ее пособразительнее, следует добавить операторы кода между Sub и End Sub. Следующая чрезвычайно простая процедура отображает поле сообщения:

```
Sub MyCodeRoutine ( )
' Следующий оператор отображает поле сообщения
MsgBox "Свидетельство силы моего кода." End Sub
```

Этот код действует, потому что язык Visual Basic включает команду, названную MsgBox. *(См. примечание "Малоизвестная или недооцененная возможность. Справка по Visual Basic" далее в этом разделе* для получения рекомендаций по применению команд, которые есть в вашем распоряжении.) Эта команда применяется для вывода на экран стандартного окна с выбранным вами сообщением. Само сообщение - это текстовый фрагмент (или строка на языке программистов), который как все текстовые значения в языке VB должен быть заключен в кавычки. Таким образом, программа Access знает, где он начинается и где заканчивается. (Access заставляет вас применять те же правила, что и при использовании текста в выражении.)

Как только вы ввели эту команду (начинайте - действуйте!), вы готовы к выполнению вашей программной процедуры. Для этого поместите курсор где-нибудь внутри тела процедуры, это дает знать редактору Visual Basic, какой код вас интересует. Затем на панели инструментов Visual Basic щелкните мышью кнопку **Run** (Выполнить) (которая выглядит как кнопка воспроизведения на панели управления видеомагнитофона), или в меню выберите **Run → Run Sub/UserForm** (Выполнить → Выполнить процедуру/пользовательскую форму). На рис. 16.3 показан результат.

Рис. 16.3. Вашу первую программную процедуру нельзя назвать ужасно полезной, но она доказывает, что вы знаете достаточно для того, чтобы написать строку кода и выполнить ее

Программа Access выполняет код процедуры построчно, от начала до конца. Когда вы выводите MsgBox, выполнение кода приостанавливается до тех пор, пока вы не щелкните мышью кнопку ОК в окне сообщения, в этот момент выполнение кода продолжится до завершения процедуры.

Примечание

Помните о том, что программа Access считает код VB потенциально опасным, поэтому не выполняет его в ненадежной БД. Иначе говоря, если вы увидите предупреждение системы

безопасности *(см. разд. "Как Access обрабатывает опасные макросы" главы 15),* следует щелкнуть мышью кнопку **Параметры** для отображения диалогового окна **Параметры безопасности Microsoft Office,** выбрать переключатель **Включить это содержимое** и затем щелкнуть мышью кнопку ОК. Или можете создать надежное расположение для файлов вашей БД. В *разд. "Задание надежного расположения" главы 15* приводится подробное описание.

Малоизвестная или недооцененная возможность.
Справка по Visual Basic

Язык Visual Basic набит таинственными командами вроде MsgBox. Вы познакомитесь со многими из них в этой главе и следующей, но для углубленного изучения всех команд нужно запустить справку Access. Далее описано, как это сделать.

1. В меню редактора Visual Basic выберите последовательность команд **Help → Справка: Microsoft Visual Basic** (Справка → Справка языка Microsoft Visual Basic)[1].

[1] *Справка приводится на английском языке. - Пер.*

2. Вы увидите список ссылок, обещающих рассказать вам больше о языке VB.
3. Щелкните кнопкой мыши **Visual Basic for Applications Language Reference** (Руководство по языку Visual Basic для приложений).
4. На экране появятся дополнительные темы, касающиеся многочисленных подробностей VB.
5. Щелкните кнопкой мыши **Visual Basic Language Reference** (Руководство по языку Visual Basic), чтобы проникнуть глубже.
6. Теперь вы увидите темы, касающиеся непосредственно языка VB. Эти основы относятся к версии Visual Basic в любой программе пакета Office.
7. Щелкните мышью строку **Functions** (Функции), чтобы увидеть команды Visual Basic, включая MsgBox. (Щелкните кнопкой мыши любую функцию для отображения страницы с подробной справкой об этой функции.)

Справка программы Access - замечательный способ узнать больше о Visual Basic, но после того, как вы познакомились с основными элементами языка. Если вы углубитесь в ее изучение слишком быстро, то можете обнаружить, что объяснения так же прозрачны, как суп из протертого гороха. Но когда вы закончите работу с примерами кода, приведенными в данной книге, вы будете достаточно подготовлены к ее использованию и пополнению ваших знаний.

16.2. *Помещение кода в форму*

Непосредственное выполнение процедуры слегка неудобно. Явно запустить макрос, по крайней мере, можно из области переходов или с помощью удобной кнопки на ленте *(см. разд. "Запуск макроса" главы 15).* Ни один из этих вариантов не предусмотрен для кода VB. Вместо этого необходимо открыть редактор Visual Basic, выбрать подходящий модуль, перейти к нужной процедуре и затем щелкнуть кнопкой мыши команду **Run** (Выполнить). В реальной жизни никто так не поступает, поскольку это слишком трудоемко.

К счастью, есть лучший способ. Можно поместить код в форму и затем выполнять его автоматически, когда происходит что-то важное. В следующих разделах объясняется, как это делается.

16.2.1. *Реакция на событие формы*

Вместо запуска программных процедур прямо из редактора, поклонники Access связывают их с событиями формы так же, как вы делали это с макросами. Далее описано, как воспользоваться этой возможностью.

1. Откройте форму в **Конструкторе.**

Самый быстрый способ - щелкнуть форму в области переходов правой кнопкой мыши и выбрать **Конструктор** (Design View).

2. Добавьте новую кнопку.

Для этого выберите на ленте **Инструменты конструктора форм | Конструктор → Элементы управления → Кнопка** (Forms Tools | Design → Controls → Button) и затем нарисуйте кнопку в рабочей области формы.

3. Когда запустится Мастер кнопок, нажмите клавишу <Esc>, чтобы отказаться от его выполнения.

Вам не нужно создавать макрос для кнопки. Эта кнопка будет снабжена чистым кодом VB.

4. Если **Окна свойств** нет на экране, выберите **Инструменты конструктора форм | Конструктор → Сервис → Страница свойств** (Forms Tools | Design → Tools → Property Sheet).

5. В Окне свойств выберите вкладку **Другие** (Other), а на ней свойство **Имя** (Name) для того, чтобы задать кнопке подходящее имя.

Программа Access использует имя кнопки для именования процедуры для вашей кнопки. Гораздо легче запомнить, что делает кнопка **CommitOrder_Click** (щелчок для подтверждения заказа), чем кнопка **Command42_Click** (щелчок для выполнения коман-ды42). Сейчас также подходящее время задать текст, отображаемый на кнопке (свойство **Подпись** (Caption)), если это еще не сделано.

6. В **Окне свойств** выберите вкладку **События** (Event) и выделите событие **Нажатие кнопки** (On Click).

Когда вы щелкните кнопкой мыши внутри поля **Событие** (Event), в нем появится направленная вниз стрелка.

7. Щелкните кнопкой мыши направленную вниз стрелку, расположенную рядом с событием **Нажатие кнопки,** а затем выберите строку **[Процедура обработки событий]** ([EventProcedure]).

Этот шаг сообщает программе Access о том, что вы вместо макроса предоставите код VB для обработки события.

8. В поле события **Нажатие кнопки** щелкните мышью кнопку с многоточием (...)

Этот шаг открывает редактор Visual Basic и создает программную процедуру для вашей кнопки. Если процедура уже создана, программа Access переходит к имеющемуся коду, и, таким образом, его можно редактировать.

Когда код вставляется в форму впервые, программа Access создает для этой формы новый модуль. Этот модуль именуется с использованием имени формы и помещается в окне **Project** в специальную группу, называемую **Microsoft Office Access Class Objects** (объекты классов Microsoft Office Access) (рис. 16.4). Если в ту же форму вставляется дополнительный код независимо от того, относится он к тому же элементу управления или к другому, Access включает новую процедуру в существующий для этой формы модуль.

Подсказка

Модули форм не выводятся в области переходов. Если они нуждаются в редактировании, следует открыть редактор Visual Basic и затем в окне Project дважды щелкнуть кнопкой мыши нужный модуль. Или же можно открыть соответствующую форму, выбрать нужное событие и использовать кнопку с многоточием для перехода прямо в интересующую вас программную процедуру.

Когда выполняются описанные действия, программа Access не создает новый модуль, а вставляет новую пустую процедуру для вашего события. Если ваша кнопка названа **ButtonOfPower** (кнопка запуска), вы увидите код, похожий на приведенный далее:

```
Private Sub ButtonOfPower_Click() End Sub
```

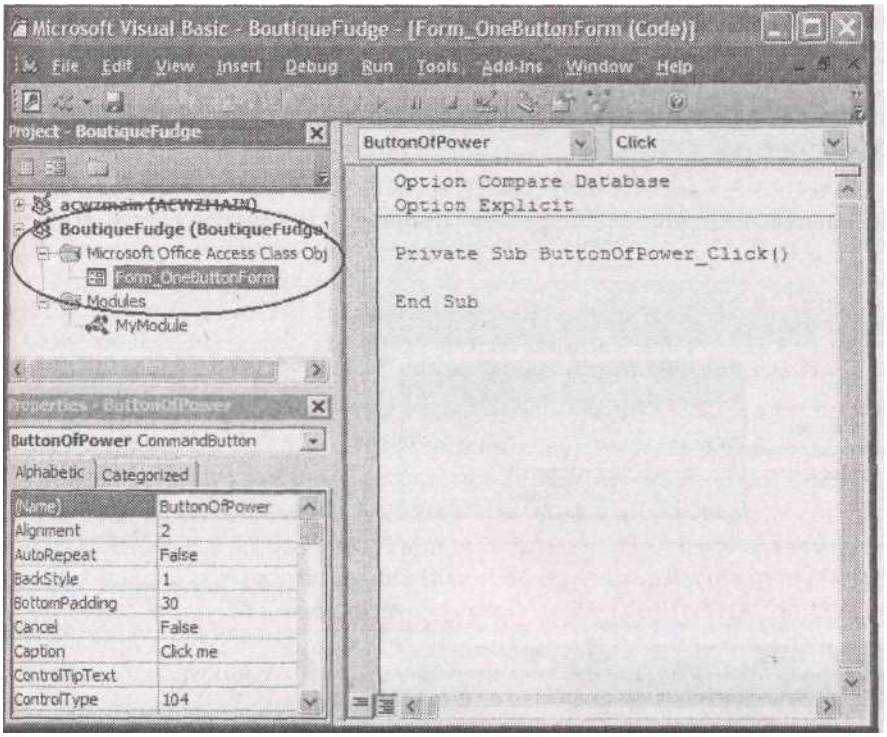
Эта процедура похожа на процедуру, созданную раньше, но с двумя отличиями.

■ Она начинается со слова Private. Это слово гарантирует невозможность использования данной процедуры другими модулями. Данная процедура доступна только в форме, содержащей кнопку. (Если не указать слово Private, будет применяться стандартный режим доступа, называемый Public, и ваша программная процедура станет общедоступной.)

В большинстве случаев выбор режима доступа не важен, но Private считается более предпочтительным.

Ее имя соответствует формату *[ИмяЭлементаУправления]_[ИмяСобытия]*. Например, приведенная ранее процедура относится к событию **Нажатие кнопки** (On Click) кнопки с именем **ButtonOfPower.**

Рис. 16.4. Модуль для формы всегда именуется ***Form_[FormName].*** Вы видите модуль для формы, названной **OneButtonForm** (однокнопочная форма)



Примечание

Постойте, разве событие не называется On Click? Сохраняя все принятые соглашения в силе, язык Visual Basic, в отличие от разработчика форм в программе Access, применяет немного иное правило именования событий. Он отбрасывает предлог "On" и все имеющиеся пробелы, таким образом, событие On Click (нажатие кнопки) становится событием Click. Лучше всего не волноваться по поводу расхождения имен и предоставить возможность программе Access сформировать правильные имена для ваших процедур.

Для того чтобы проверить вашу процедуру, нужно добавить в нее код. Пока вы познакомились лишь с одной командой, поэтому попробуйте применить ее для вывода сообщения:

```
Private  Sub ButtonOfPower_Click(     )
MsgBox  "Вы щелкнули мышью кнопку ButtonOfPower." End Sub
```

Теперь вернитесь к форме и щелкните вашу кнопку мышью. Вы должны увидеть сообщение, показанное на рис. 16.5, оно означает, что ваш код уловил событие и успешно на него отреагировал.

Подсказка

Во время редактирования модуля нет необходимости сохранять его. Можно спокойно переходить между окном кода и формой, отображенной в окне программы Access, туда и обратно для проверки каждого вносимого в код изменения.

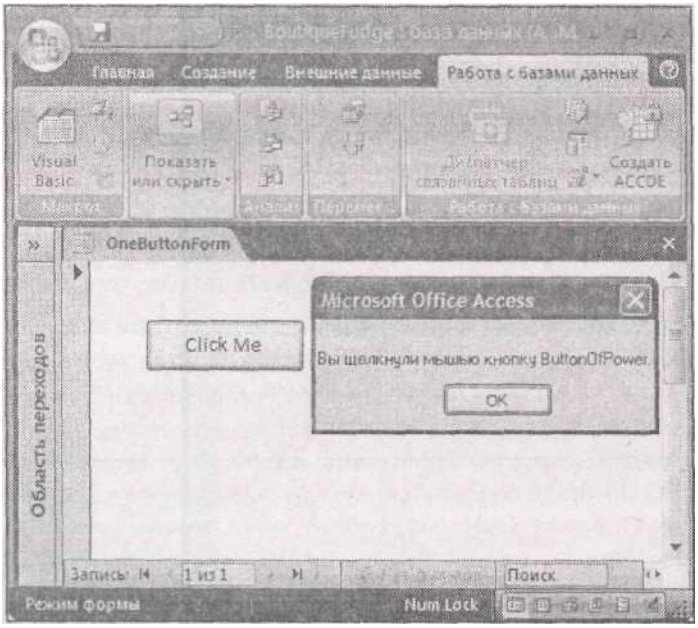


Рис. 16.5. События обеспечивают автоматическое выполнение программного кода. В данном случае, если щелкнуть мышью кнопку **ButtonOfPower,** Access тут же отобразит сообщение

Если позже вы уберете с формы кнопку **ButtonOfPower,** программа Access не удалит программный код. Он останется на прежнем месте, но будет неактивен. Это удобно, если в программном коде есть что-то полезное, что впоследствии, возможно, вы захотите применить где-нибудь еще. (В данном случае вас выручат операции вырезания и вставки.) Но если это просто фрагмент старого кода, при первой же возможности воспользуйтесь редактором Visual Basic для удаления процедуры.

Для тех кто понимает.

Как код связывается с событиями

Все дело в имени - в имени процедуры. **Когда вы открываете** форму, в которой есть соответствующий программный код' программа Access ищет процедуры с определенными именами.

Если она находит процедуру, названную MyButton Click, то, прежде всего, проверяет элемент управления с именем **MyButton** и наличие у него события с именем Click (нажатие кнопки). Если обе детали обнаружены, эта программная процедура становится обработчиком данного события, что на своеобразном жаргоне программистов означает связывание вашего кода с событием. Когда возникает событие (например, когда кнопку щелкнули мышью), Access выполняет код из вашей процедуры.

Если программа Access находит процедуру, названную MyButton_Click, а на форме нет элемента управления с именем **MyButton,** не стоит паниковать. Access просто предполагает, что вы создали процедуру для собственных внутренних нужд. Поскольку эта процедура не является обработчиком события, программа Access не запускает ее автоматически в ответ на возникновение события. Но она все же при необходимости может обращаться к вашему коду, как описано на этой странице.

У описанной системы два возможных камня преткновения. Во-первых, не меняйте имя обработчика события по своему усмотрению - если вы сделаете это, разорвется связь между формой и программным кодом, и обработчик события не будет выполняться, когда событие произойдет. Во-вторых, не меняйте имя элемента управления с помощью **Окна свойств,** поскольку это действие тоже разорвет связь. Если же на самом деле нужно исправить неудачное имя, убедитесь в том, что вы изменили имя процедуры так, что оно соответствует новому имени элемента управления.

В обоих случаях (при переименовании процедуры или элемента управления) Access не предупреждает о возможных последствиях. Поэтому помните эти рекомендации, чтобы избежать нежелательных неожиданностей.

16.2.2. *Вызов кода в модуле*

Полученные знания могут вызвать вопрос: в каких ситуациях вообще может возникнуть необходимость в создании стандартного модуля вручную? Помимо всего прочего нет удобного способа выполнения программного кода и возможности связать его с событием элемента управления на форме.

Стандартные модули полезны, если создан удивительно удачный фрагмент программного кода, который хочется применять в разных местах. Скажем, вы разработали отличную процедуру поиска и хотели бы использовать ее в двух, трех или четырех десятках различных форм. Этот код можно вырезать и вставить во все процедуры, которые в нем нуждаются. Помимо того, что дублирование кода - всегда плохой выбор (как и дублирование данных). Почему? Подумайте, что произойдет, если вам придется исправлять ошибку или вносить корректировку. Из-за дублирующегося кода придется найти все его копии и повторить в них одно и то же изменение. Это надежный способ потратить свои выходные дни.

Правильное решение - взять вашу полезную, многоразовую процедуру и поместить ее в модуль. Затем вы можете вызвать ее, когда понадобится, и редактировать ее только один раз, когда необходимо обновить процедуру или устранить ошибку. Для вызова программной процедуры в другом модуле используется имя модуля с последующей точкой (.), за которой указывается имя процедуры. Вот пример:

```
Private  Sub ButtonOfPower_Click(    )
MyModule.MyCodeRoutine End Sub
```

Далее приведен оперативный отчет о происходящем в процессе применения данного кода.

1. Нажимается кнопка **ButtonOfPower.**
2. Программа Access находит процедуру ButtonOf Power_Click и выполняет ее.
3. Код процедуры запускает другую процедуру, MyCodeRoutine в модуле **MyModule.** Этот код отображает на экране сообщение, которое вы видели раньше (см. рис. 16.3).
4. После того как процедура MyCodeRoutine завершила работу, Access выполняет оставшийся код в процедуре ButtonOf Power_Click. В данном примере в процедуре нет никаких операторов, поэтому процесс завершается в этот момент.

Вы можете вставить одну задачу в любое число процедур. Можно также вызывать процедуры, которые в свою очередь вызывают другие процедуры, также вызывающие дополнительные процедуры и т. д.

Примечание

Подобный прием можно использовать только в случае общедоступных процедур. Частные (Private) не доступны для программного кода, находящегося за пределами модуля, в котором эти процедуры хранятся. Автоматически всем процедурам назначается режим доступа Public до тех пор, пока вы не вставили слово Private перед словом Sub. Таким образом, процедура MyCodeRoutine в *разд. "Написание процедуры с простейшим программным кодом" ранее в этой главе* общедоступна.

Если обе процедуры, ButtonOf Power_Click и MyCodeRoutine, хранятся в одном моду-, ле, можно применить некоторую рационализацию. Не нужно включать имя модуля в имя процедуры MyCodeRoutine. Вы можете использовать приведенный далее код:

```
Private Sub ButtonOfPower_Click( )
MyCodeRoutine End Sub
```

Теперь программа Access полагает, что процедура MyCodeRoutine должна быть в том же модуле и ищет ее в нем. В данном случае неважно, процедура MyCodeRoutine общедоступная или частная - ваш код все равно может вызывать ее.

Подсказка

Если вы хотите повторно использовать код в нескольких местах одной и той же формы (например, в ответ на нажатие разных кнопок), подумайте о создании собственной процедуры и размещении кода в ней. Если же предполагается использование одного фрагмента кода в нескольких формах, лучше поместить его в процедуру, хранящуюся в отдельном модуле.

16.2.3. *Чтение и запись полей на форме*

Как вы узнали в *главе 15,* самые замечательные макросы - те, которые контролируют ваши формы и элементы управления. Код VB справляется с этой задачей потрясающе легко. Вам просто нужно знать имена всех элементов управления, с которыми вы хотите работать.

Предположим, что вы разрабатываете (нечто опасное) программную процедуру, очищающую текстовое поле. Вы планируете применять ее для сбрасывания текста в поле **Description** (описание). Далее приведена строка кода, выполняющая эту работу:

```
Description = ""
```

В этой строке приведен основной оператор присваивания языка Visual Basic (строка кода, изменяющая порцию данных), и вся строка сосредоточена вокруг знака равенства (=). Когда Access выполняет эту строку кода, программа берет содержимое справа от знака равенства (в данном случае пустую строку со знаками кавычек, представляющими текст) и вставляет его во вместилище, расположенное слева от знака равенства (в данном случае поле **Description).** В результате содержимое поля **Description** стерто.

Примечание

Можно также использовать знакомые квадратные скобки, т. е. [Description] вместо Description. Скобки не обязательны, если вы не настолько самонадеянны, чтобы пренебрегать правилами именования, с которыми познакомились в *разд. "Правило 1. Выбирайте подходящие имена полей" главы 2.* Если у вас есть имя текстового поля с пробелом, такое имя нужно обязательно заключать в квадратные скобки.

Конечно же, можно использовать, конкретный текстовый фрагмент:

```
Description = "Введите сюда что-нибудь, пожалуйста"
```

Результат выполнения этого кода аналогичен вашему собственноручному вводу текста (за исключением того, что происходит все гораздо быстрее). Как известно, при корректировке любого поля текущая запись переводится в режим редактирования. Как только вы переходите к другой записи или закрываете форму, программа Access фиксирует изменения и сохраняет новые значения в БД.

С помощью фиксированных значений можно решить только ограниченный круг задач. В результате, когда используется фиксированное текстовое значение, вначале нужно решить, что именно вы хотите использовать. Когда вы щелкните кнопку мышью и запустите код, возможно, ваши желания изменятся. По этой причине программисты редко используют фиксированные значения в подобном случае. Вместо этого они предпочитают более сложные выражения, очень похожие на выражения Access, которые применялись в вычислениях запросов *(см. разд. "Вычисляемые поля" главы 7)* и условий на значения *(см. разд. "Правила верификации или условия на значения" главы 4).*

С текстом можно использовать оператор слияния & для создания длинного фрагмента текста из нескольких более коротких фрагментов. Далее приведен пример, в котором берется текущее описание, и к нему в конец добавляется предложение, идентифицирующее товар по названию.

```
Description = Description & " This is a description for " & ProductName & "."
```

Если начальное значение **Description** равно "Enjoy delectable waves of fudge." ("Насладитесь упоительными волнами сладости."), в результате оно может стать таким "Enjoy delectable waves of fudge. This is a description for Fudge Tsunami." ("Насладитесь упоительными волнами сладости. Это описание Fudge Tsunami.").

Малоизвестная или недооцененная возможность

Разбиение длинных строк кода

Если вы имеете дело с чрезмерно длинными строками кода, самое время применить символ продолжения строки языка Visual Basic, таково необычное название символа подчеркивания (_).

Заканчивайте любую строку пробелом и знаком подчеркивания и можете продолжить программный код сразу в следующей строке:

```
Description = Description & _
```

```
" This is a description for " & _
ProductName & "."
```

Если вы собираетесь применить этот прием, стоит задать отступ во всех строках, кроме первой, так вы сможете сразу увидеть, что это часть одного оператора кода.

Гораздо чаще в выражениях используются числовые значения или даты. Далее приведен код для кнопки IncreasePrice (повышение цены), которая повышает цену на 10% при каждом нажатии кнопки (и самое замечательное в том, что вы сможете щелкать кнопку мышью столько раз, сколько захотите):

```
Private  Sub  IncreasePrice_Click
Price = Price  *  1.10
End Sub
```

Обзор разных операций, которые можно применять в выражениях для выполнения вычислений различных типов (например, сложения, умножения, деления и т. д.), приведен в табл. 7.1.

Примечание

Язык Visual Basic трактует поля *Да/Нет* как поля *True/False* (Истина/Ложь). Конечный результат такой же, а синтаксис немного отличается. Для задания значения поля *Да/Нет* используется одно из двух встроенных ключевых слов Visual Basic: True или False.

16.3. *Что такое объекты*

На самом деле с элементами управления можно делать гораздо больше. Вместо простого изменения их содержимого у вас есть возможность изменять их цвет, шрифт, местоположение, видимость и множество других характеристик. Для того чтобы стать волшебником, следует усвоить, что все элементы управления - программируемые объекты.

В мире программирования объект - это не что иное, как удобный способ собрать вместе некоторые связанные параметры. Поле Description - не просто одно значение, это целый объект "поле ввода", а это значит, что у него есть все виды встроенных параметров. Если понять, как действует поле ввода, можно добраться до всех остальных параметров элемента.

Примечание

Программа Access создает некоторую путаницу, потому что применяет слово объект в двух разных смыслах. На протяжении всей книги вы называли объектами БД все компоненты вашей БД (такие как таблицы, запросы и формы). Программисты пользуются словом "объект" в более строгом смысле для обозначения программной структуры, собирающей вместе связанные параметры и функциональные возможности (и в данной главе этот термин применяется именно в этом смысле).

Взаимодействовать с объектами можно тремя способами.

■ *С помощью свойств.* Свойства - это порции данных, относящихся к объекту. У объекта **Поле ввода** есть свойство Fontsize (Размер шрифта), которое управляет размером его текста.

• *С помощью методов.* Методы - это действия, которые можно выполнять с помощью объекта. Например, у всех форм есть метод Requery (обновление), позволяющий повторно выполнить запрос, с помощью которого получены данные формы.

■ *С помощью событий.* События - это сообщения, которые посылает объект, и на которые вы можете откликнуться с помощью вашего программного кода. Вы можете отреагировать на нажатие кнопки мышью, используя событие кнопки On Click (нажатие кнопки).

В следующих разделах все три эти характеристики объекта рассматриваются более подробно.

16.3.1. *Свойства*

Свойства не должны быть для вас новостью. В конце концов, вы потратили достаточно времени на их настройку в **Окне свойств,** добиваясь подходящего форматирования и поведения. Но свойства предстают совсем в ином свете, когда вы получаете возможность изменять их средствами программного кода. С помощью кода свойства можно динамически изменять в

ответ на различные действия (например, нажатие кнопки или редактирование текста в поле ввода). Этот подход открывает целый мир новых возможностей.

Секретный ключ к встроенным свойствам объекта - скромная точка (которую фанаты программирования называют *операцией "точка").* Предположим, что вы хотите изменить цвет фона поля **Description.** Это можно сделать с помощью задания значения в свойстве BackColor (цвет фона) соответствующего объекта "поле ввода". Вот как это делается:

```
Description.BackColor = vbYellow
```

В этой строке программного кода берется объект **Description** и затем применяется операция "точка" для выбора его свойства BackColor. Задается значение свойства BackColor с помощью специально созданного ключевого слова vbYellow. Имя свойства в программном коде, как вы видели в случае событий, не всегда совпадает с его именем в **Окне свойств.**

В программном коде имена свойств никогда не содержат пробелы.

Эту строку кода можно использовать в любой процедуре модуля формы до тех пор, пока у этой формы на самом деле есть элемент управления **Поле,** названный **Description.**

Примечание

Цвета в Access задаются числовыми кодами. VB упрощает жизнь для большинства широко распространенных цветов, предлагая использовать заранее определенные имена, начинающиеся с букв vb. Эти имена - рациональный способ ссылки на соответствующий числовой код цвета. За кадром vbYellow - это 65 535. (Если поискать в справке Access слово "vbYellow", можно найти полный список восьми основных цветовых констант. В *примечании "Практические занятия для опытных пользователей. Получение нужного цвета" разд. "Обозначение измененной записи" далее в этой главе* вы узнаете, как выбирать дополнительные цвета.)

Если не применять операцию "точка", вы будете использовать свойство но умолчанию. Для поля ввода свойство по умолчанию - Value (значение), предоставляющее содержимое поля. Вот почему можно написать не содержащую операции "точка" строку, подобную приведенной далее:

```
Description = "Действительно чудесный товар."
```

Теперь, когда вы узнали, что все элементы управления на ваших формах - это объекты с целым набором полезных параметров, которые можно изменять, возникает важный вопрос: как узнать, какие свойства есть у объекта и как найти нужные вам? Несколько рекомендаций окажут необходимую помощь.

- *У идентичных элементов управления одинаковые свойства.* Несмотря на то, что каждое поле ввода на вашей форме представлено отдельным объектом "поле ввода", у всех объектов "поле ввода" одни и те же свойства. Очевидно, что значения свойств могут отличаться, но вы можете быть уверены в том, что если вы нашли свойство BackColor в одном поле ввода, это же свойство вы найдете и у всех остальных полей ввода.
- У *похожих элементов управления похожие свойства.* У них у всех есть свойство BackColor, будь то поле ввода, кнопка или скромная подпись. Даже объекты, представляющие разные части формы (например, Detail (область данных), FormHeader (заголовок формы), FormFooter (примечание формы)) содержат свойство BackColor для задания цвета их фона. Этот вид стандартизации позволяет применить знания об одном элементе управления к другому элементу.
- *Свойство можно найти в **Окне свойств.*** Имена свойств, применяемые в программном коде, обычно соответствуют именам свойств, которые вы видите в **Окне свойств** (в англоязычной версии программы). Существует одно отличие - у имен свойств, используемых в коде, нет пробелов, поэтому свойство **Back Color** (Цвет фона) в **Окне свойств** в программе превращается в свойство BackColor.

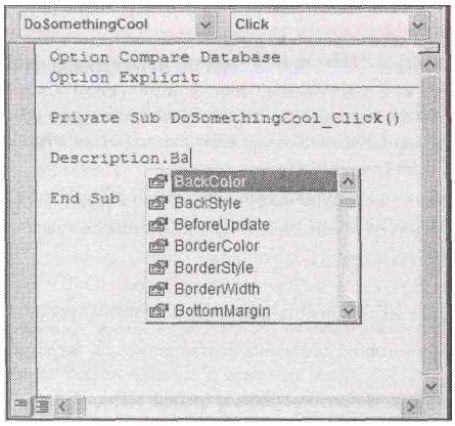


Рис. 16.6. Когда вводится имя объекта и затем точка, Visual Basic выводит список вариантов. Если ввести несколько букв, Visual Basic перейдет к соответствующему участку списка. Если вы увидели нужное свойство, его можно вставить щелчком кнопки мыши или нажатием клавиши <Пробел>

■ *Свойство можно найти с помощью средства Visual Basic IntelliSense.* Редактор Visual Basic предлагает замечательный инструмент, способный помочь найти нужное свойство. Как только вы ввели точку после имени объекта, редактор выводит на экран список всех свойств и методов, которые можно применять к данному объекту (рис. 16.6).

Примечание

Список IntelliSense содержит два вида элементов: свойства (отдельные характеристики, касающиеся объекта) и методы (действия, которые можно выполнять с объектом). Свойств больше и они помечены пиктограммой руки, держащей почтовую открытку. Методы снабжены пиктограммой парящего зеленого ластика. Вы узнаете, как пользоваться методами в следующем разделе.

На профессиональном уровне.

Взаимодействие с другими формами

Как вы узнали в *главе 15,* можно извлекать и задавать значения свойств полей и элементов управления на текущей форме или других открытых в данный момент формах. Хитрость заключается в том, что нужно явно указать программе Access, какую форму вы пытаетесь использовать.

Предположим, что вы хотите изменить цвет элемента управления **Price** (цена) на форме **Product** (товар) при щелчке мышью кнопки, находящейся на форме **PriceChanger** (преобразователь цен). Приведенный далее код не работает, поскольку Access ищет несуществующий элемент управления **Price** на форме **PriceChanger:**

```
Price.BackColor = vbRed
```

Следующий код отлично использует описанную хитрость и направляет программу Access к правильной форме:

```
Forms("Product").Price.BackColor = vbRed
```

Технически приведенный код заставляет программу Access заглянуть в коллекцию Forms (формы), отслеживающую все открытые в данный момент формы. (Если в настоящий момент форма **Product** не открыта, этот оператор даст сбой.) Он извлекает из коллекции форму **Product,** переходит к форме для доступа к элементу управления **Price** и затем углубляется в него для поиска свойства BackColor (Цвет фона).

Логически равноценную строку кода можно записать двумя способами. Программисты Access старого образца применяют причудливый синтаксис с восклицательными знаками, который выглядит следующим образом:

```
Forms!Product!Price!BackColor = vbRed
```

Программа Access интерпретирует обе строки одинаково. Это просто дело вкуса. Но вы должны знать оба варианта, на случай, если столкнетесь со странным кодом с восклицательными знаками.

Если вас огорчает, что этот подход приводит к ошибке в случае закрытой нужной формы,

оба описанных метода не смогут выручить вас. В *главе 17* вы узнаете, как открывать форму, когда захочется.

В табл. 16.1 перечислены некоторые свойства элементов управления, которые могут вам понадобиться в программном коде на языке Visual Basic.

Таблица 16.1. Полезные свойства элементов управления

Свойство	*Тип данных*	*Описание*
Value (значение)	Зависит от конкретного элемента	Хранит значение элемента управления. Обычно все элементы связаны с полями, поэтому свойство Value позволяет считывать или изменять значение из текущей записи. В зависимости от типа данных поля свойство может быть текстовым, числовым, логическим и т. д.
Enabled (включено или доступно)	True (истина) или False (ложь)	Определяет возможность корректировки значения элемента управления. Если задать в свойстве значение False, элемент управления блокируется и пользователь, работающий с формой, не сможет редактировать поле (несмотря на то, что ваш код может все еще пытаться изменять значение Value). Отключенные элементы управления выглядят не так, как включенные элементы управления - обычно у них подернутый серой пеленой "серый" внешний вид
Visible (видимый)	True (истина) или False (ложь)	Определяет, может ли пользователь, работающий с формой, видеть элемент управления. Если у свойства значение False (ложь), элемент исчезает с формы. Это свойство - удобный способ скрыть неиспользуемые поля. Если клиент живет в Замбии, можно скрыть поле State (штат)
ForeColor (цвет текста) и BackColor (цвет фона)	Число	Определяет цвет для вывода текста (цвет текста) и цвет, отображаемый за текстом (цвет фона)
Left (слева) и Top (сверху)	Число	Определяет местоположение элемента управления на форме. Свойство Left (слева) определяет расстояние от левого края формы до левого края элемента. Свойство Top (сверху) задает расстояние от верхнего края формы до верхнего края элемента управления. Оба значения задаются в пикселах
Width (ширина) и Height (высота)	Число	Определяет размер элемента управления в пикселах
FontName (шрифт) и FontSize (размер шрифта)	Текстовая строка и число (соответственно)	Определяет шрифт, который применяется для отображения текста в элементе управления. FontName - это название гарнитуры шрифта (например, "Arial"), а FontSize - его размер в пунктах (например, 10)
FontBold (жирное начертание) и FontI-talic (курсив)	True (истина) или False (ложь)	Определяет способ начертания текста (жирный или наклонный)
Picture* (рисунок)	Текстовая строка	Позволяет отображать фоновое изображение на части формы, на вкладке или кнопке. Вы задаете путь к файлу с рисунком
Text* (текст)	Текстовая строка	Задает текущий текст в текстовом поле. В большинстве случаев это свойство содержит те же данные, что и свойство Value. Но если кто-то редактирует текст и еще не перешел к другому элементу управления, содержимое этих двух свойств разное. Свойство Value содержит текст, хранящийся в таблице, а свойство Text - отредактированные данные, которые еще не применены
Caption* (подпись)	Текстовая строка	Задает текст подписи или кнопки или заголовок формы. Это свойство важно, когда создаются подписи, не связанные с полями таблицы. Элемент управления Подпись можно применять для отображения сообщения о состоянии или статусе
IternsSelected* (выбранные элементы)	Объект-коллекция	Представляет собой коллекцию, объект специального типа, содержащий ноль или больше подобъектов. Данная коллекция содержит значения всех выбранных в данный момент в списке элементов. Свойство ItemsSelected полезно, только если создан список, поддерживающий многочисленные выделения, В противном случае используйте свойство Value

** Это более специальные свойства и в большинстве элементов управления они не используются.*

16.3.2. *Методы*

Методы позволяют выполнять действия с объектом. Во многих случаях вызов метода дает больше, чем просто задание свойства. В самом деле, один метод может выполнить комплексную операцию, которая повлияет на многие свойства. Метод Requery (обновление) заставляет вашу форму получить самые свежие данные из БД и затем обновить содержимое всех ее элементов управления.

Подсказка

Если применяются элементы управления, большая часть времени тратится на работу со свойствами. Действительно, у элементов управления огромное множество свойств и лишь несколько добавочных методов.

Для применения метода вводится имя объекта с последующей точкой, за которой набирается имя метода. Но вам не нужен знак равенства, т. к. вы не задаете значение метода. Вы просто вызываете его для выполнения или активизируете.

Далее приведен пример обновления текущей записи формы с помощью метода Refresh (обновить объект):

```
Form.Refresh
```

Иногда методу требуется дополнительная информация. Если у вас как раз тот случай, вы узнаете об этом, т. к. средство Visual Basic IntelliSense даст знать в процессе написания программного кода (рис. 16.7).

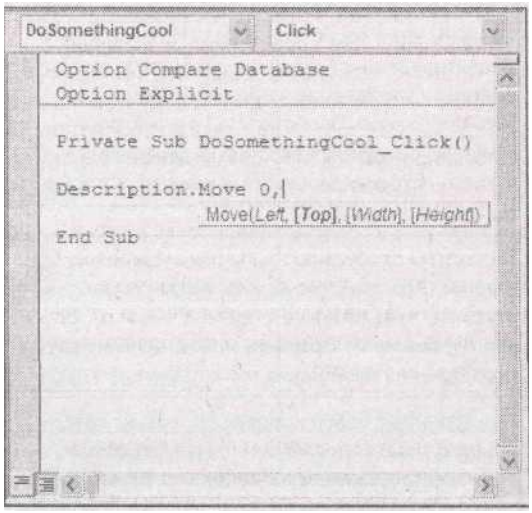


Рис. 16.7. Редко применяемый метод Move позволяет одним махом изменить местоположение и размер элемента управления. Как только вы введете имя метода, редактор Visual Basic отобразит четыре значения, которые нужно задать. В данном примере только первое значение (Left) обязательно - остальные заключены в квадратные скобки, что свидетельствует о возможности их пропуска

Если в методе нужно задать дополнительные данные, следует добавить пробел после имени метода и дальше указать соответствующее значение. Если требуется ввести несколько значений, каждое из них следует отделять запятой. Далее приведен пример, перемещающий элемент управления в левый верхний угол формы:

```
Description.Move 0, 0
```

В табл. 16.2 перечислены наиболее важные методы элементов управления.

Таблица 16.2. Полезные методы элементов управления

Метод	*Описание*
SetFocus (установить фокус)	Переносит курсор внутрь элемента управления, делая его, таким образом, действующим элементом. Этот метод полезен при выполнении проверки условий на значения. Если в поле найдена ошибка, можно вернуть пользователя к элементу управления с

	ошибкой
Undo (отмена)	Аннулирует самые последние (незафиксированные) изменения в элементе управления. Этот метод можно вызвать в форме для отмены всех изменений и возврата к исходным значениям. Если в данный момент форма не в режиме редактирования, метод не делает ничего
Recalc (повторное вычисление)	Пересчитывает любые выражения в элементах управления формы
Refresh* (обновить объект)	Получает из таблицы самые свежие значения для данной записи и, соответственно, обновляет форму. Этот метод полезен, если вы только что выполнили другую задачу, которая могла изменить запись, или работаете с многопользовательской БД (см. главу 18), в которой несколько человек одновременно могут изменять запись
Requery* (обновление)	Повторно выполняет запрос, применяемый для получения данных формы, и затем отображает полученные данные, начиная с первой записи. Этот метод подобен методу Refresh, но воздействует не на текущую запись, а на все записи. Этот метод можно применять к списку подстановки для обновления его содержимого

** Эти методы применяются только к объектам формы, а не отдельным элементам управления.*

16.3.3. *События*

Как вы знаете, события - это сообщения, которые объекты используют для передачи в ваш программный код сведений о том, что только что произошло нечто важное. Вы уже управляли событиями и применяли их в данной главе для реагирования на щелчки мышью кнопок. Список самых распространенных событий элементов управления приведен в табл. 15.4.

До сих пор не рассматривался один аспект: как события могут предоставлять дополнительные биты данных. Как вы, наверное, уже заметили, у каждой процедуры есть пара скобок. Посмотрим на них еще раз:

```
Private Sub ButtonOfPower_Click ()
```

В предыдущих примерах в этих скобках не было ничего. Но они введены не просто так. Некоторые события передают в ваш код дополнительную информацию о событии, и она вставляется в этот сэндвич из двух скобок.

Рассмотрим событие **Нажатие клавиши** (On Key Press) поля ввода, которое возникает каждый раз, когда кто-то нажимает символ на клавиатуре. Оно предоставляет специальный числовой код, обозначающий нажатую клавишу (программисты называют его кодом ASCII.)

Если добавить процедуру, реагирующую на событие **Нажатие клавиши** (On Key Press), программа Access сгенерирует код, подобный приведенному далее:

```
Private Sub MyTextBox_KeyPress(KeyAscii As Integer) End Sub
```

Этот код означает, что событие **Нажатие клавиши** (On Key Press) снабжает ваш код новой порцией информации. Это целое число, названное KeyAscii, его можно использовать в вашем коде.

Далее приведен пример, который просто отображает код нажатой клавиши в окне сообщения:

```
Private Sub MyTextBox_KeyPress(KeyAscii As Integer)
MsgBox "Вы нажали клавишу с кодом: " & KeyAscii
End Sub
```

Некоторые события предоставляют несколько порций данных. В этих случаях вы увидите в скобках целый список. Каждая порция данных отделяется запятой и называется *параметром.*

Примечание

Формально параметры - разновидность переменных. *Переменные* - это удобные контейнеры, хранящие некоторые данные. (Эти данные могут меняться, поэтому их и назвали переменными.) Вы узнаете больше об использовании переменных а *разд. "Хранение информации в переменных" главы 17.*

Далее приведен пример для события **Перемещение указателя** (On Mouse Move), которое возникает при перемещении указателя мыши поверх элемента управления. Открывающееся объявление процедуры такой длины, что приходится разделить его на две строки с помощью знака подчеркивания:

```
Private  Sub SomeControl_MouseMove(Button As   Integer,  _
Shift As  Integer,   X As  Single,   Y As  Single)
End Sub
```

В данном случае вы получаете четыре порции данных. Параметр Button обозначает, какие кнопки мыши нажаты в данный момент. Параметр Shift показывает, удерживаются ли нажатыми во время перемещения мыши клавиши <Shift>, <Ctrl> и <Alt>. И, наконец, параметры X и Y определяют местоположение указателя мыши (его координаты).

16.4. *Применение объектов*

Теперь, когда вы познакомились с основами языка Visual Basic, вам, наверное, не терпится начать писать реальный программный код. В следующих разделах представлены два примера, заставляющие элементы управления работать.

Подсказка

Если хотите знать больше, можно найти подробное руководство по применению объектов в справочной, системе Access. Для получения справки выберите в редакторе VESA[2] **Help → Справка: Microsoft Visual Basic** (Help → Microsoft Visual Basic Help). Дальше последовательно выберите следующие темы: **Visual Basic for Applications Language Reference → Microsoft Forms Visual Basic Reference → Reference.** Затем вы увидите список всех объектов, предлагаемых программой Access (щелкните мышью строку **Objects)** или сформируйте комбинированный список событий, методов и свойств, предлагаемых объектами Access (щелкните кнопкой мыши строку **Events, Methods** или **Properties).**

[2] *Справка редактора VBA приводится на английском языке. -. Пер.*

16.4.1. *Обозначение измененной записи*

Редактирование записи - двухэтапный процесс. Сначала вы изменяете одно или несколько значений полей, что переводит запись в режим редактирования. Затем вы закрываете форму или переходите к другой записи, что фиксирует внесенные вами изменения. Или вы нажимаете клавишу <Esc> для отмены изменений и затем возвращаетесь к первоначальным значениям.

Если вы пользуетесь панелью **Запись** (Record Selection) (при условии, что в свойстве **Кнопки перехода** (Record Selectors) установлено значение *Да,* стандартный вариант), программа Access показывает переход в режим редактирования, заменяя крошечную стрелку в верхнем левом углу формы маленькой пиктограммой карандаша. Эта пиктограмма - полезный индикатор того, что в вашей форме что-то изменено, и нужно решать, принимать ли эти изменения. Но новички программы Access, как и профессионалы, легко могут не заметить крошечную пиктограмму карандаша. Именно поэтому большинство пользователей предпочитают более явное свидетельство внесенных изменений, такое как отображение сообщения на форме или изменение цвета фона.

В следующем примере демонстрируется этот подход. Результат показан на рис. 16.8.

Для создания этого примера необходимо начать с построения подходящей формы. Возьмите обычную форму и добавьте элемент управления **Подпись** в раздел **Примечание формы.** Задайте для подписи приемлемое имя, например, InfoMessage (информационное сообщение), изменив значение свойства **Имя** в **Окне свойств.** Теперь можно начать писать программный код.

Примечание

Имена элементов управления важны. Они используются для ссылок на объекты в программном коде. Когда вы читаете фрагмент кода, никто, даже вы, не догадается, что

обозначает имя **Label44.**

Прежде чем добавить ваш код к форме, необходимо определить следующее.

■ Когда запустится ваш код? Иными словами, нужно указать событие, которое должно его запускать. После того как вы дали ответ на этот вопрос, можно создавать корректную процедуру.

■ Что должен делать ваш код? Иначе говоря, следует решить, какие объекты использовать и какие свойства изменять. Когда ответ найден, можно писать код в вашей процедуре.

В данном примере необходимо реагировать на событие формы Изменение данных (On Dirty). Оно возникает, когда запись изменяется любым способом и форма переключается в режим редактирования. Если изменяются несколько значений, событие **Изменение данных** (On Dirty) возникает только для первого изменения. После этого форма уже находится в режиме редактирования.

Примечание

У каждого отдельного элемента управления есть собственное событие **Изменение данных** (On Dirty), которое происходит, когда кто-то в первый раз изменяет определенный элемент. Не стоит обращать внимание на эти события. Вам нужно использовать событие формы **Изменение данных** (On Dirty), таким образом вы поймаете все возможные изменения.

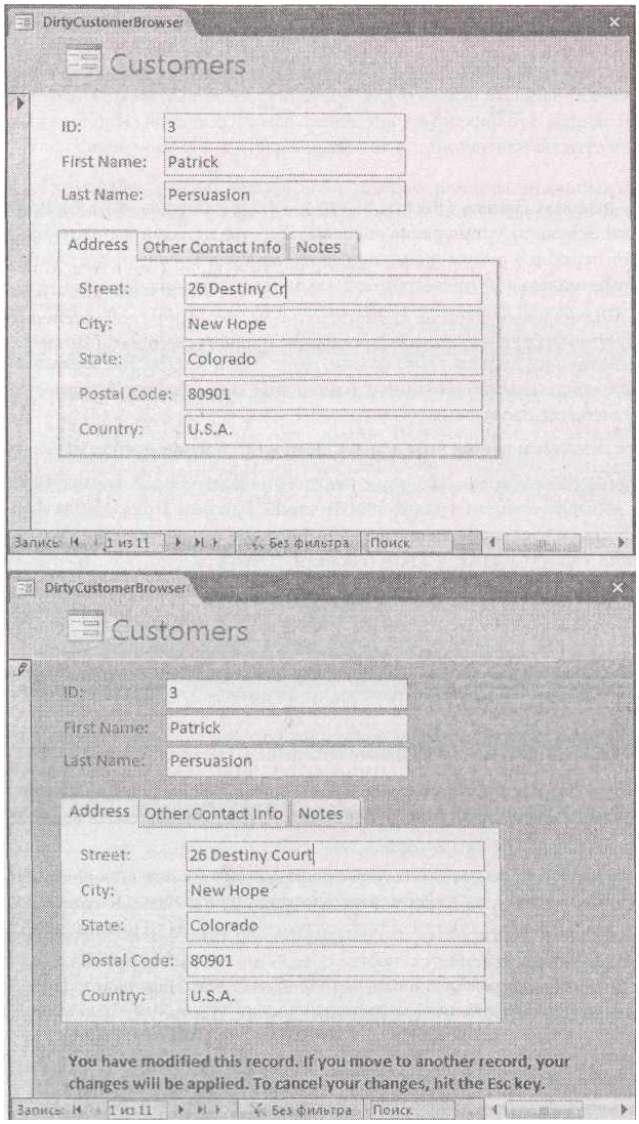


Рис. 16.8. *Вверху:* на первый взгляд форма кажется обычной. *Внизу:* если откорректировано какое-либо значение, цвет фона изменяется и в нижней части отображается текстовое сообщение

Далее приведена процедура, необходимая для реагирования на событие вашей формы **Изменение данных** (On Dirty):

```
Private Sub Forrn_Dirty(Cancel As Integer) End Sub
```

Примечание

Эта процедура выглядит несколько иначе, чем виденные вами ранее процедуры, т. к. она включает поддержку отмены действий. Сейчас не обращайте внимания на этот параметр - вы узнаете о нем все в конце *разд. "Принятие решений" главы 17.*

Данную процедуру можно вручную внести в существующий модуль (если только использовать для вашей процедуры то же самое имя) или добавить его с помощью **Окна свойств** (просто выбрать **Форма** в списке **Окна свойств,** затем указать событие **Изменение данных,** далее в поле события выбрать из списка **Процедура обработки событий** и щелкнуть мышью кнопку с многоточием).

Далее следует самая увлекательная часть - написание программного кода. Сначала вам нужен оператор, изменяющий цвет фона формы. У объекта Form нет свойства BackColor (цвет фона), зато такое свойство есть у объектов, представляющих отдельные разделы формы (Details (область данных), FormFooter (примечание формы) и FormHeader (заголовок формы)). Таким образом, можно применить следующий программный код: Detail.BackColor = vbRed

Необходимо также задать текст сообщения в элементе управления **Подпись**[3]**:**

```
InfoMessage.Caption = "Вы изменили данную запись. " & _
"Если перейдете к другой записи, ваши изменения будут внесены. " & _
"Для отмены изменений нажмите клавишу Esc."
```

Поместите два оператора в процедуру Form_Dirty и все готово.

Практические занятия для опытных пользователей.

Получение нужного цвета

Если задавать цвет только с помощью ключевых слов, таких как vbRed, vbWhite и vbYellow, вы обделите себя. Существует множество пастельных тонов и волнующих оттенков, только и ждущих применения в вашем коде на языке Visual Basic. К сожалению, для задания этих цветов нельзя использовать ключевые слова. Вместо них придется применять числовые коды цветов.

Чаще всего вы не будете знать правильного числового кода для цвета, который хотите использовать. Но эту проблему можно решить с помощью удобной функции RGB, включенной в язык Visual Basic (как и функция MsgBox, которая применялась раньше). Функция RGB принимает три отдельных числа, представляющих красную, зеленую и синюю составляющие цвета, и преобразует их в код цвета, который можно использовать для задания свойств ForeColor или BackColor.

[3] *На рис. 16.8 текст сообщения выводится на английском языке. Вы, разумеется, можете тоже выводить текст на английском. - Ред.*

Далее приведен пример, использующий описанный метод, для применения оттенка светлого оранжево-розового цвета:

```
Detail.BackColor = RGB(266, 160, 122)
```

Этот оператор выполняется в два этапа. Сначала программа Access выполняет функцию RGB для создания кода цвета. Затем она заносит код цвета в свойство BackColor. На первый взгляд преимущество применения функции RGB может показаться не столь очевидным, поскольку она требует указания трех отдельных чисел. На самом деле RGB-обозначение цвета - общепринятый стандарт, применяемый в Web-пространстве и в большинстве Windows-приложений. Найти цвет можно даже в указателе цвета программы Access, а затем определить нужные RGB-компоненты, выполнив следующие действия.

1. Откройте форму в Конструкторе.
2. Выберите элемент управления и затем в **Окне свойств** щелкните кнопкой мыши поле **Цвет текста** (ForeColor) или **Цвет фона** (BackColor).
3. Щелкните мышью кнопку с многоточием (...) в поле цвета для перехода в окно быстрого выбора цвета, в котором отображаются некоторые распространенные и недавно использовавшиеся цветовые варианты.
4. Выберите кнопку **Другие цвета** (More Colors) для отображения полного набора цветов.
5. Щелкните кнопкой мыши вкладку Спектр (Custom).

6. Выберите цвет, как показано на рис. 16.9.
7. Запишите RGB-значения. Их можно использовать в вашем программном коде.
8. Щелкните мышью кнопку **Отмена** (Cancel) для возврата в программу Access.

У формы в этот момент обнаруживается дефект. Когда выполняется первое изменение, появляется подпись и меняется цвет фона, как и должно быть. Однако, после того как вы зафиксировали изменение, перейдя к другой записи, сообщение и цвет фона сохраняются. Полученный результат, очевидно, - не то, что вам нужно.

Для устранения ошибки необходимо отреагировать на другое событие: событие формы После обновления (After Update). Оно возникает после того, как программа Access успешно зафиксирует изменение в БД. Далее приведен код, необходимый для возврата обычного внешнего вида формы:

```
Private Sub Form_AfterUpdate ()
Detail.BackColor = vbWhite
InfoMessage.Caption = "" End Sub
```

Примечание

Не следует использовать событие **До обновления** (Before Update), поскольку оно возникает как раз перед фиксацией изменений. В этот момент вы не знаете, найдет ли программа Access некорректные данные, отобразит ли сообщение об ошибке и помешает ли обновлению (а в этом случае красный цвет фона следует сохранить).

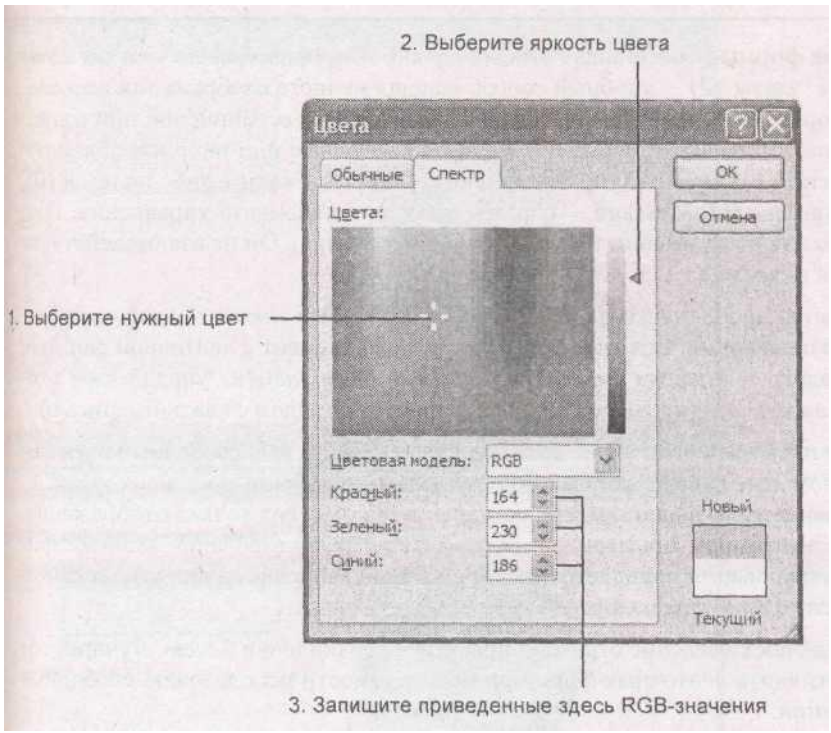


Рис. **16.9.** Для выбора цвета щелкните кнопкой мыши в цветовой сетке. (Перекрестье указывает вашу текущую позицию.) Затем используйте вертикальный ползунок для регулировки яркости цвета. Вы увидите, что по мере изменения цвета будут меняться значения **Красный** (Red), **Зеленый** (Green) и **Синий** (Blue)

Пример все еще не закончен. Помимо фиксации изменений пользователь может также нажать клавишу <Esc> для их отмены. На эту возможность тоже следует отреагировать и применить тот же код для возврата к обычному состоянию формы. В данном случае используется событие **Отмена** (On Undo):

```
Private Sub Form_Undo () Detail.BackColor = vbWhite InfoMessage.Caption = ""
End Sub
```

Этот шаг завершает пример. Для того чтобы увидеть все три процедуры вместе и испытать

их в действии, загрузите из Интернета пример БД для данной главы (в *разд. "Примеры" во введении* объясняется, как обращаться с примерами БД).

Часто задаваемый вопрос.

Ленточные формы и неприсоединенные элементы управления

Я изменил свойство ***Режим по умолчанию*** *(Default View) моей формы на значение* ***Ленточные формы*** *(Continuous Form), и мой программный код сошел с ума. Что произошло?*

Режим Ленточные формы (Continuous Form) *(см. разд. "Отображение нескольких записей в любой форме" главы 12)* - удобный способ одновременного отображения нескольких записей на форме. Но у него появляется ряд существенных ограничений при соприкосновении с кодом. Вы увидите их, когда вставите свободные или неприсоединенные элементы управления - элементы управления, не связанные с каким-либо полем в БД. В предыдущем примере **InfoMessage** - образец свободного элемента управления. Ваш код применяет его для отображения текста по вашему желанию. Он не взаимодействует со значением поля в таблице.

В этом и заключается проблема: когда используется свободный элемент, у вас для обработки есть только одна копия. Если соединить свободный элемент с ленточной формой, вы получите парадокс - а именно, существует только один элемент управления, отображающийся на экране в нескольких местах одновременно (рядом с каждой записью).

Это явление - не проблема, если вы не хотите корректировать ваш свободный элемент. Поскольку на самом деле существует только один неприсоединенный элемент управления, когда вы изменяете его в одном месте, он изменяется в других точках отображения. В только что рассмотренном примере, когда начинается редактирование записи, признак режима редактирования появляется рядом с каждой записью, даже если на самом деле редактируется только одна из них.

К сожалению, подобное поведение отражает проектное ограничение Access. Лучший обходной прием - избегать ленточных форм при необходимости использовать свободные элементы управления.

16.4.2. *Создание эффекта перемещения указателя мыши*

Эффект перемещения указателя мыши (mouseover effect) - это действие, возникающее при перемещении мыши поверх какого-либо участка формы. Web-дизайнеры часто используют этот эффект для изменения внешнего вида кнопок при перемещении по ним указателя мыши.

В программе Access легко создать эффект перемещения указателя. Необходимо только отреагировать на событие **Перемещение указателя** (On Mouse Move). Вы можете использовать событие формы **Перемещение указателя** (On Mouse Move), если хотите следить за перемещением мыши по всей форме. Но гораздо чаще событие **Перемещение** указателя (On Mouse Move) применяется к конкретным элементам управления, что позволяет определить перемещение указателя поверх этих элементов управления.

В форме, приведенной на рис. 16.10, применяется эффект перемещения указателя мыши.

Как обычно для реализации подобного поведения, следует начать с добавления дополнительных необходимых элементов управления, например кнопки **Don't Click Me** (не щелкай меня мышью) и рисунка (названного **HappyFace** (счастливое лицо)).

Когда описанные детали добавлены, необходимо создать две процедуры. Первая реагирует на событие кнопки **Перемещение указателя** (On Mouse Move). Она заменяет рисунок счастливого лица рисунком с расстроенным лицом, когда мышь перемещается поверх кнопки:

```
Private Sub DoNotClickButton_MouseMove(Button As Integer, _
    Shift As Integer, X As Single, Y As Single)
HappyFace.Picture = "C:\Images\UnHappy.jpg" End Sub
```

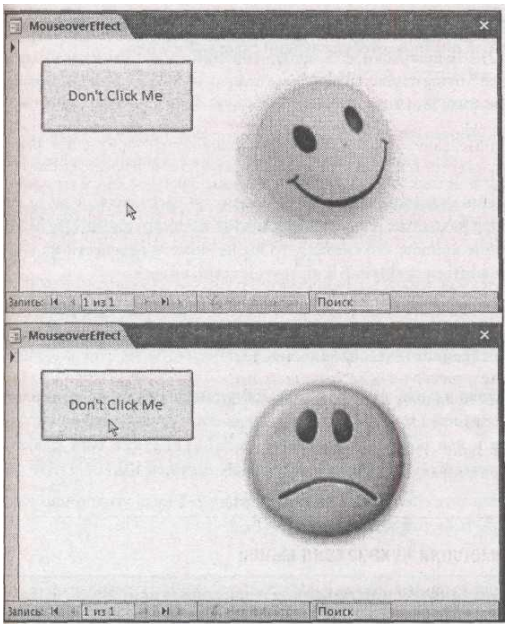


Рис. **16.10.** *Вверху:* первоначальное отображение формы. *Внизу:* когда мышь перемещается поверх кнопки **Don't Click Me** (не щелкай меня мышью), на изображении, расположенном сбоку, счастливое лицо сменяется расстроенным. Подвигайте мышь где-нибудь в другом месте, и счастливая физиономия вернется на экран

В этом коде предполагается, что файл с изображением (названный UnHappy.jpg) помещен в папку C:\Images.

Как и при создании другого программного кода, можно ввести текст процедуры вручную в уже существующий модуль формы, или, что гораздо удобнее, применить для ее создания **Окно свойств** *(см. разд. "Помещение кода в форму "ранее в этой главе).*

Подсказка

Событие **Перемещение указателя** (On Mouse Move) возникает очень часто. Когда мышь перемещается от одного края формы к другому, подобные события возникают многократно. По этой причине следует убедиться в быстроте выполнения написанного вами кода, реагирующего на данное событие, иначе он может сделать вашу форму медлительной.

Вторая процедура реагирует на событие **Перемещение указателя** (On Mouse Move) объекта формы Область данных, которая возникает, когда вы отодвигаете мышь от кнопки и перемещаете ее поверх свободного пространства, находящегося вокруг нее. Данная процедура переключает на исходный рисунок, отображающий счастливое лицо.

```
Private Sub Detail_MouseMove(Button As Integer, _
Shift As Integer, X As Single, Y As Single)
HappyFace.Picture = "C:\Images\Happy.jpg"
End Sub
```

У написанного кода есть один недостаток. В данный момент он рассчитывает на то, что файлы UnHappy.jpg и Happy.jpg находятся в конкретных местах жесткого диска. Эту деталь трудно гарантировать - в конце концов, кто сказал, что вы не можете перенести их куда-нибудь в другое место или попытаться открыть БД на другом компьютере?

Лучше поместить рисунки в ту же папку, что и файл БД. Вы можете указать программе Access на подобное местоположение с помощью следующего кода:

```
HappyFace.Picture = CurrentProject.Path & "\Happy.jpg"
```

В этом коде применяется специальный объект, всегда доступный в любом коде, который вы пишите: объект CurrentProject (текущий проект), предоставляющий информацию о текущей БД и содержащихся в ней объектах. У объекта CurrentProject есть свойство Path (путь), задающее в виде текстовой строки местоположение текущей БД.

Поместив эту строку, вы сможете спокойно копировать вашу БД куда угодно при условии,

что файлы рисунков находятся в той же папке.

Практические занятия для опытных пользователей.

Связывание записей с рисунками

В *главе 2 вы узнали, как сохранять изображения* в таблице с помощью поля типа **Вложение.** Но этот способ не всегда подходит, особенно если ваши файлы рисунков нуждаются в изменении или применении вне программы Access или из-за их крайне большого объема. В этих случаях предпочтительнее хранить имя файла вашего рисунка.

Тем не менее вы сможете отображать рисунок в форме Access. Метод прост - выполните лишь следующие действия.

1. Добавьте на форму новый элемент управления **Рисунок,** но не связывайте его ни с каким полем. Вы выведете нужное изображение с помощью программного кода.

2. Создайте обработчик события формы **Текущая запись** (On Current), который запускается при каждом переходе к записи.

3. В обработчике события задайте в свойстве Picture (Рисунок) элемента управления то изображение, которое хотите вывести на экран. Если в вашей таблице есть поле, названное **ImageFileName** (имя файла с изображением), и элемент управления с именем **Img,** можно написать следующий код:

```
Img.Picture = CurrentProject.Path & _
"\Images\" & ImageFileName
```

В данном примере предполагается, что файлы рисунков хранятся в папке Images (изображения), находящейся в папке с файлом вашей БД. Когда форма загружается впервые (и при каждом переходе к другой записи), этот код выполняется и помещает соответствующее изображение в элемент управления **Рисунок.**

При использовании данного кода следует также применить обработку ошибок *(см. разд. "Обработка ошибок" главы 17).* Обработка ошибок важна, поскольку вы не можете быть уверены в том, что изображения не были перемещены или удалены, а если это произошло, вы заинтересованы в грамотном решении проблемы.

17. Глава 17. Написание кода с более развитой логикой

В *главе 16* вы с головой погрузились в мир программирования на языке Visual Basic, создавая процедуры, способные выводить на экран сообщения, откликаться на события и изменять формы. Попутно вы узнали достаточно о языке Visual Basic и объектно-ориентированной системе, придающей волшебную силу VB.

Тем не менее, кое-что еще осталось. В данной главе вы узнаете, как применять VB-код для решения некоторых самых широко распространенных проблем, с которыми сталкиваются профессионалы Access. Вы сосредоточитесь на усовершенствовании БД Boutique Fudge, с которой вы работали на протяжении всей книги. Но решения, которые будут использоваться, настолько полезны, что вам захочется включить их в ваши собственные БД.

Прежде чем приниматься за эти более сложные примеры, углубим знание языка Visual Basic, научившись находить ошибки и повнимательней присмотревшись к объектам. Эту тему помогут дополнить представление о Visual Basic и подготовят ваш переход в разряд настоящего программиста Access.

17.1. *Изучение языка Visual Basic*

Несмотря на то, что вы уже знаете достаточно для реагирования на события и изменения свойств элементов управления, все еще остается множество вещей, касающихся самого языка Visual Basic, с которыми следует познакомиться. В следующих разделах вы узнаете, как применять переменные, условные операторы и циклы для создания более мощного программного кода. В конце будет показано, как использовать эти средства для разработки процедуры с более замысловатым программным кодом, ищущей некорректные номера кредитных карт.

17.1.1. *Хранение информации в переменных*

Все языки программирования включают идею переменных, временных контейнеров в оперативной памяти, в которых можно отслеживать важную информацию.

Допустим, вы хотите поменять местами содержимое двух полей. На первый взгляд эта операция совершенно очевидна. Нужно лишь взять текст из одного поля и поместить его в другое поле, а затем взять текст, бывший во втором поле, и поместить его в первое поле. Далее приведен первый вариант решения:

```
TextBoxOne.Value = TextBoxTwo.Value
TextBoxTwo.Value = TextBoxOne.Value
```

Для того чтобы код заработал, необходимо поместить его в подходящую процедуру. В данном примере код выполняется, когда кто-либо щелкает мышью кнопку на форме. Можно создать процедуру для события **Нажатие кнопки** (On Click) с помощью **Окна свойств.**

К сожалению, этот код обречен с самого начала. На рис. 17.1 показана проблема.

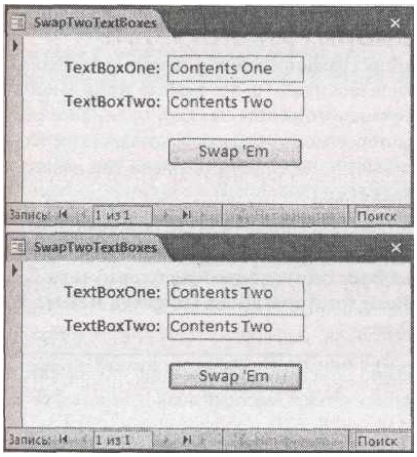


Рис. 17.1. *Вверху:* первоначально в каждом поле собственные данные. *Внизу:* после выполнения процедуры перестановки данных вы не получите ожидаемый результат. Как

только вы вставляете новое содержимое во второе поле, затирается содержимое, которое вы хотели поместить в первое поле. Конечный результат - два поля с одинаковым содержимым

Простейший способ обойти данную проблему - применить переменную для хранения нужной вам информации. Для создания переменной в языке VB используется странное ключевое слово Dim (от англ. *dimension* - величина, что на программистском жаргоне значит "создать новую переменную"). После слова Dim вводится имя переменной.

Вот как создается переменная TextContent: Dim TextContent

Практические занятия для опытных пользователей.

Применение более сложных переменных

В примере, приведенном выше, показан простейший способ создания переменной в коде на языке VB. В нем создается переменная, известная как Variant и способная хранить любой тип данных, включая текст, числа, логические значения и т. д. Грамотные VB-программисты предпочитают быть строже и явно задают тип данных для каждой создаваемой ими переменной. В этом случае никто случайно не сохранит текст в переменной, предназначенной для числового содержимого и наоборот.

Для создания переменной с фиксированным типом данных вы добавляете в объявление ключевое слово As. Вот как создается переменная TextContent для хранения только текста:

```
Dim TextContent As  String
```

Далее создается переменная, которая хранит большие целые числа:

```
Dim NumberContent As  Long
```

Подобный подход считается хорошим стилем программирования и помогает выявлять определенные типы ошибок. Но для его применения нужно знать разные типы данных языка Visual Basic. К наиболее часто используемым относятся String, Date, Boolean (значения *Истина* или *Ложь),* Long (целое, которое может быть очень маленьким или очень большим), single (число с дробной частью) и Currency (числовой тип данных, идеально подходящий для хранения финансовых сумм).

Описание всех типов данных языка VB можно найти в справочной системе программы Access. Для перехода к ней выберите **Справка → Справка: Microsoft Visual Basic** (Microsoft Visual Basic Help → Microsoft Visual Basic help) из меню редактора Visual Basic. Далее выберите последовательно **Visual Basic for Applications Language Reference → Visual Basic Language Reference → Data Types.**

После того как переменная создана, в нее можно помещать информацию и извлекать из нее данные. Для выполнения обеих операций применяется известный знак равенства, так же как и в случае свойств.

Далее приведен пример сохранения некоторого текста в переменной:

```
TextContent = "Тестовый текст"
```

В следующем примере все сведения собраны вместе. В нем используется переменная для взаимообмена информацией между двумя текстовыми полями.

```
Dim TextContent
' Копируется текст из первого поля для использования в дальнейшем
TextContent = TextBoxOne.Value

' Заменяется текст в первом поле
TextBoxOne.Value = TextBoxTwo.Value

' Заменяется текст во втором поле с помощью переменной
TextBoxTwo.Value = TextContent
```

17.1.2. *Принятие решений*

Условная логика, еще один ключевой элемент программирования, - это код, который выполняется при соблюдении определенного условия. Для применения условной логики нет никаких ограничений. Можно помешать обновлению, если вновь введенные данные не проверены. Или же возможна иная настройка элементов управления на форме в зависимости от их

данных. Все это и многое другое можно сделать с помощью условной логики.

Любой условный блок начинается с условия: простого выражения, которое может принимать значение *Истина* или *Ложъ* (программисты называют этот процесс проверкой истинности или ложности условия). Затем ваш код может принять решение о выполнении разных логических алгоритмов в зависимости от оценки условия. Для построения условия необходимо сравнить переменную или свойство с помощью логической операции, например = (равно), < (меньше чем), > (больше чем) и **о** (не равно). Например, Price=10 - это условие. Оно может быть истинно (если в поле Price содержится число 10) или ложно (если в этом поле содержится любое другое число). Условия уже применялись в правилах верификации или условиях на значения. В языке Visual Basic условия следуют набору очень похожих правил.

Само по себе условие не может ничего сделать. Но в сочетании с другим кодом оно может стать чрезвычайно мощным инструментом. После того как подходящее условие создано, его можно поместить в специальную структуру, именуемую блоком If. Блок If вычисляет условие и выполняет часть кода, если условие истинно. Если условие ложно, программа Access полностью игнорирует код.

Далее приведен пример блока If, проверяющего, не превышает ли значение в поле **Price** числа 100. Если да, то Access выводит на экран сообщение:

```
If Price > 100 Then
MsgBox  "Надеюсь,   у вас выделены деньги для этого."
End If
```

Обратите внимание на то, что блок If всегда начинается со слова If и заканчивается оператором End If. Внутри блока If можно поместить столько строк кода, сколько захотите. Это условный код, он выполняется, только если условие истинно.

В условном блоке может проверяться несколько разных условий. Далее приведен пример, в котором вычисляется цена товара с учетом всех налогов, а затем отображается в элементе управления **Подпись.** Хитрость заключается в том, что налоговая ставка зависит от другого поля **(Country)** и именно здесь в игру вступает условная логика.

```
' В этой переменной сохраняется налоговая ставка,
‘которую вы хотите применить
Dim TaxRate
If Country =  "U.S.A."  Then
'  Налоги меняются для клиентов из США   (7%)
TaxRate =1.07
Elself Country = "Canada" Then
'   Еще большие налоги взимаются с клиентов из Канады   (14%)
TaxRate =1.14  Else
'     Все остальные освобождаются от налогов
TaxRate =1 End If
 ' Окончательный итог отображается в подписи
TotalWithTax.Caption = Price * TaxRate
```

В блоке If выполняется только один сегмент кода. В данном примере программа Access по-своему обрабатывает блок, проверяя каждое условие до тех пор, пока не найдет подходящее. Как только совпадение найдено, выполняется условный блок кода и последующий переход к завершающему оператору End I f, а затем продолжается обработка остального кода, входящего в процедуру. Если нет подходящих условий, Access выполняет код в заключительном операторе Else (если вы его вставили). На рис. 17.2 показано выполнение данного кода.

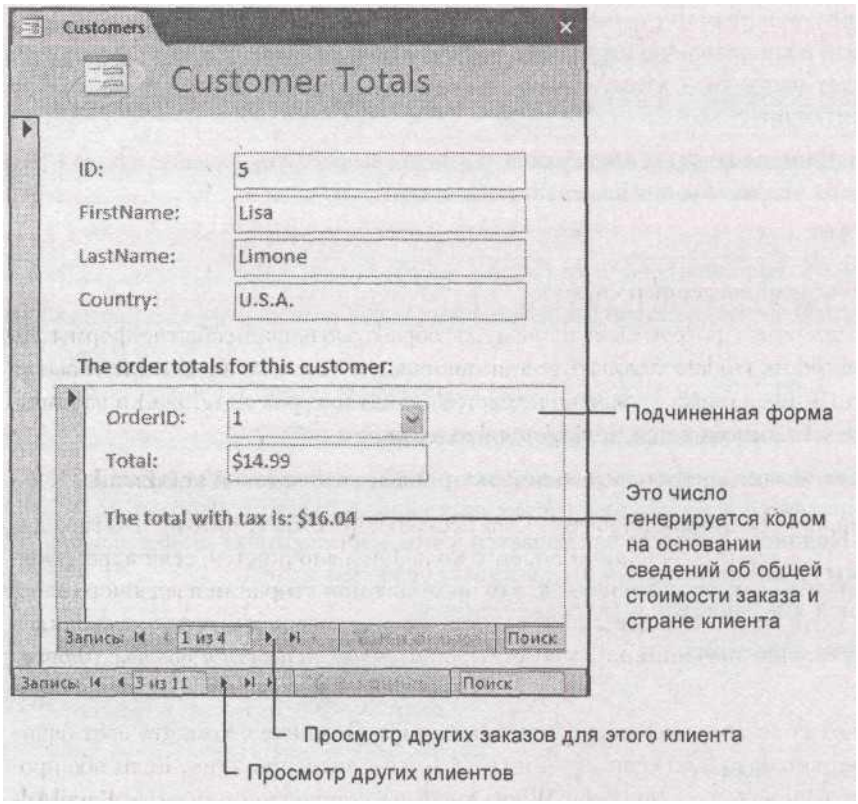


Рис. 17.2. Благодаря условной логике эта подчиненная форма всегда отображает правильно вычисленную окончательную цену, учитывая как текущую цену, так и страну клиента. Она действует в ответ на событие **Текущая запись** (On Current), возникающее каждый раз, когда запись выводится на экран

Эти примеры показывают лишь лежащие на поверхности возможности условной логики. Можно применять ключевые слова And и Or для комбинирования условий, помещать один условный блок в другой и т. д.

В *главе 15* вы видели пример, который выполнял проверку на значение конкретного типа в записях клиентов. Эта проверка использовала два поля: **WantsEmail** и **Email Address.** Если в поле **WantsEmail** было задано значение *Да* (Yes), поле **EmailAddress** не могло оставаться пустым. Но если у поля **WantsEmail** было значение *Нет* (No), пустое поле **EmailAddress** считалось вполне приемлемым. Вы можете с помощью VB-кода реализовать идентичную логику проверки, но тут есть хитрость - в проверке применяются два блока If (следом за кодом дается построчное объяснение[1]):

```
1  Private Sub Form_BeforeUpdate(Cancel As Integer)
' Проверяется, нужен ли электронный адрес
2  If WantsEmail = True Then
' Убеждаемся, что поле EmailAddress не пустое или незаполненное
3  If EmailAddress = "" Or IsNull(EmailAddress) Then
' Это некорректный вариант.
' Отмена изменения и вывод сообщения
4  MsgBox "You can't be notified without an email address."
5  Cancel = True
6  End If
7  End If
8  End Sub
```

Вот как выполняется приведенный код.

■ В *строке 1* объявляется программная процедура, обрабатывающая событие формы До обновления (On Before Update). Обратите внимание на то, что этот обработчик события

получает одну порцию данных - значение True (Истина) или False (Ложь) в параметре Cancel, который позволяет задать запрет обновления.

■ В *строке 2* начинается блок If, проверяющий установлен ли флажок **WantsEmail.**

■ В *строке 3* выполняется вторая проверка. Она несколько сложнее, потому что есть две вещи, способные вызвать выполнение условного кода. Он выполняется, если адрес электронной почты равен пустому значению (что происходит при стирании введенного электронного адреса) или если поле адреса не заполнено (т. е. электронный адрес не вводился ни разу; см. обсуждение значений null в *разд. "Пропущенные значения и пустые строки" главы 4).*

ш Строка 4 выводит пояснительное сообщение об ошибке. Помните о том, что этот фрагмент кода выполняется, только если оба блока If имеют значение True. Если обе проверки выдают значение False (флажок **WantsEmail** не установлен или поле **EmailAddress** не задано), Access немедленно возвращается назад.

Примечание

Технически можно оба эти блока If объединить в один блок с помощью более сложного условия, проверяющего все сразу. Но сделать это правильно (и впоследствии понять, что вы написали) гораздо труднее. Опытные программисты знают, что всегда лучше писать ясный код, даже если при этом он становится слегка многословнее.

[1] *В тексте строки кода пронумерованы. Обратите внимание, что в реальном коде нумеровать строки не следует. - Ред.*

■ *Строка 5* отменяет обновление с помощью параметра Cancel, предоставляемого событием **До обновления** (On Before Update). В этом случае изменения не происходят, и запись остается в режиме редактирования.

■ *Строки 6-8* - завершающие: они закрывают оба блока If и заканчивают процедуру.

У программы Access есть множество событий, которые вы можете отменить, как событие **До обновления** (On Before Update). Поищите параметр Cancel, располагающийся в скобках после имени процедуры. Если он там, можно задать ему значение True для прекращения действия, которое готово произойти.

17.1.3. *Повторение действий с помощью цикла*

Цикл - это инструмент, позволяющий повторять операцию столько раз, сколько нужно. В языке Visual Basic есть несколько типов циклов, которые можно использовать. Наиболее популярны блоки Do/Loop и For /Next, и в этом разделе вы познакомитесь с обоими.

Далее приведен пример блока Do/Loop, который наверняка выведет читателей из равновесия:

```
Do
MsgBox "Ever ever get that nagging deja vu feeling?" Loop
```

Когда программа Access обрабатывает этот код, она начинает с вывода окна сообщения и остановки выполнения остального вашего кода. После того как вы щелкнули мышью кнопку ОК, выполнение кода продолжается до финального оператора Loop в конце цикла. В этот момент программа автоматически переходит к началу цикла (оператор Do) и повторяет ваш код, выводя второе окно сообщения. Проблема в том, что этот процесс продолжается бесконечно! Если допустить ошибку и запустить этот код, ваша БД будет заблокирована на неопределенное время (пока вы не нажмете комбинацию клавиш аварийного останова <Ctrl>+<Break>).

Для того чтобы избежать подобной ситуации, следует создавать все циклы с условием выхода из цикла, сигнализирующим о моменте его завершения. Далее приведен исправленный вариант того же цикла, который прекращается после того, как сообщение отображено пять раз:

```
' Отслеживает количество повторений цикла
Dim NumberOfTimes

' Начинает отсчет с 0
NumberOfTimes = 0

Do
MsgBox "Ever ever get that nagging deja vu feeling?"
```

```
' Увеличивает счетчик на 1
NumberOfTimes = NumberOfTimes + 1
Loop Until NumberOfTimes =5
```

Важная часть - оператор в конце цикла, Until Number0fTimes=5. Он определяет условие, и как только оно становится True (значение переменной NumberOfTimes равно 5), программа Access достигает конца цикла и переходит к выполнению оставшегося кода процедуры.

Если вам нужен цикл с фиксированным количеством повторений, возможно, вас заинтересует цикл типа For/Next. Этот тип цикла в точности такой же, как цикл Do/Loop за исключением того, что у него есть встроенный счетчик, увеличивающийся автоматически при каждом проходе цикла.

Теперь можно переписать предыдущий пример в более компактной форме с применением цикла For/Next:

```
Dim NumberOfTimes
For NumberOfTimes = 1 To 5
MsgBox  "Ever ever get that nagging deja vu feeling?"
Next
```

Важная часть цикла - оператор NumberOfTimes=l To 5, сообщающий программе Access о начальном значении переменной NumberOfTimes, равном 1, о необходимости увеличения ее значения на 1 при каждом проходе цикла и о завершении цикла после пятого прохода.

Цикл Do/Loop удобен при обработке коллекции данных. Его можно применять для обработки информации до тех пор, пока она не иссякнет, даже если заранее вы не знаете, сколько данных у вас есть. Вы увидите пример использования этого метода в конце данной главы, когда будете выполнять пакетное обновление вашей БД с помощью программного кода.

С другой стороны цикл For/Next предстанет во всей красе, если вы можете точно определить заранее, сколько раз хотите повторить цикл. В справедливости этого вы убедитесь чуть позже в этой главе, когда будете проверять номера кредитных карт.

17.1.4. *Создание пользовательских функций*

Вы уже научились создавать собственные процедуры, но еще не знаете о том, как создать их старшую сестру, функцию.

Функция, как и процедура, - изолированный фрагмент кода, способный включать в себя произвольное число операторов. И, так же как и процедуры, функции можно добавлять в модули. В модуле может сосуществовать рядом любое количество процедур и функций.

```
Function DoSomething()
'Здесь  код функции
End Function
```

Главное отличие функции от процедуры заключается в том, что функция формирует конечный результат. Другими словами, функции предоставляют порцию нужных вам данных.

Вы задаете результат, написав строку кода, которая присваивает окончательное значение имени функции. (По существу, вы считаете, что имя функции - это переменная, в которой можно хранить некоторые данные.)

Далее приведен пример:

```
Function GetMyFavoriteColor()
GetMyFavoriteColor = "Magenta" End Function
```

Эта функция названа GetMyFavoriteColor (получение моего любимого цвета). Ее результат - текстовая строка "Magenta" (пурпурный).

Вызов функции несколько отличается от вызова процедуры. Для вызова процедуры вы используете имя модуля, за которым следует точка, а затем имя процедуры. Этот же способ можно применить и для вызова функции:

```
MyModule.GetMyFavoriteColor
```

Но возникает проблема. Этот шаг запускает функцию GetMyFavoriteColor, но отбрасывает результат (строку с текстом Magenta).

Если вас интересует результат, можно вызвать функцию в операторе присваивания. В приведенном далее коде создана переменная для хранения результата и последующего

отображения его в окне сообщения:

```
' Создается переменная для хранения результата
Dim Color

' Вызывается функция и результат сохраняется в переменной
Color = MyModule.GetMyFavoriteColor

' Результат отображается в окне сообщения
MsgBox "Your favorite color is " & Color
```

Если вы по-настоящему сообразительны, то можете сократить этот код до одной строки и вообще избежать использования переменной Color:

```
MsgBox "Your favorite color is " & MyModule.GetMyFavoriteColor
```

Функция GetMyFavoriteColor предельно проста, т. к. не использует аргументы. Но ничто не мешает вам проявить большую изобретательность. Рассмотрите следующую пользовательскую функцию, которая принимает два аргумента - длину и ширину - и вычисляет площадь, перемножая их:

```
Function Area(Length, Width)
Area = Length * Width
End Function
```

Эти два параметра определяются в скобках после имени функции. Количество параметров может быть любым при условии, что вы отделяете один от другого запятой.

Далее приведен пример вызова данной функции и отображения результата. В нем для определения параметров Length (длина) и Width (ширина) применяются числовые константы. Но их вполне можно заменить именем поля, переменной или свойством, которые вы хотите использовать в функции Area.

```
MsgBox "The area of a 4x4 rectangle is " & Area(4, 4)
```

На экране появится:

```
The area of a 4x4 rectangle is 16
```

Ни функция GetMyFavoriteColor (), ни функция Area () не продемонстрировали ничего впечатляющего, но в следующем разделе этой главы вы создадите гораздо более мощную функцию для проверки номеров кредитных карт.

Практические занятия для опытных пользователей.

Применение пользовательских функций в запросах

После того как вы создали функцию, ее можно использовать в вашей БД для построения запросов или условий на значения. Единственное требование - ваша функция должна находиться в пользовательском модуле, который вы добавили (не в модуле формы) и в ее объявлении не должно быть слова Private. Если ваша функция соответствует указанным требованиям, вы можете вызывать ее так же легко, как любую встроенную функцию программы Access.

Можно создать, например, такой запрос с вычисляемым полем (при условии, что в него включены два поля, названные **LengthOfRoom** (длина комнаты) и **WidthOfRoom** (ширина комнаты) соответственно):

```
RoomArea: Area (LengthOfRoom, WidthOfRoom)
```

Или можно создать следующее условие на значение для таблицы:

```
Area(LengthOfRoom * WidthOfRoom) < 10000
```

См. в *главе 7* дополнительные соображения по поводу применения функций в вычисляемых полях и в *главе 4* дополнительную информацию об условиях на значения или правилах верификации. Если же вы хотите увидеть этот конкретный пример в действии, обратитесь к БД MyHouse, которая включена в примеры к данной главе.

17.1.5. *Подытожим: функция для проверки кредитных карт*

Теперь, когда вы познакомились с языком Visual Basic, самое время подвести итог с помощью примера, демонстрирующего все, что вы узнали о VB (и даже немного больше).

В данном примере рассматривается пользовательская функция ValidateCard

(допустимая карта), которая проверяет номер кредитной карты. Функция ValidateCard возвращает одно из двух значений: True (что означает допустимый номер карты) и False (что означает неверный номер).

Важно понимать, что допустимый номер кредитной карты - это просто номер, соответствующий несекретным правилам нумерации кредитных карт (см. дополнительную информацию в следующем примечании). Этот номер может соответствовать реальной кредитной карте или нет. Функция ValidateCard достаточно сообразительна для вылавливания случайных ошибок и не слишком умелых хакеров. По-настоящему злонамеренные пользователи могут найти программы, позволяющие генерировать потенциально допустимые номера кредитных карт.

На профессиональном уровне.

Алгоритм Луна (Luhn Algorithm)

В функции ValidateCard применяется алгоритм, названный алгоритмом Луна, который разработал специалист компании IBM в 1960 гг. Алгоритм Луна действует, т. к. компании, выпускающие кредитные карты, следуют его правилам. Другими словами, они выпускают только такие номера, которые считаются допустимыми с точки зрения алгоритма Луна.

Полное объяснение алгоритма Луна можно найти на Web-странице **http://en.wikipedia.org/wiki/Luhn_algorithm.** Далее приведено описание его принципа действия в версии журнала "Reader's Digest".

1. Удвойте каждую вторую цифру номера кредитной карты, начиная с последней цифры номера. Оставьте все цифры с четным[2] номером неизменными. Например, номер 1111 превратится вномер2121.

2. Если в процессе удвоения получается число большее 9, сложите две цифры полученного результата вместе и поставьте их на место исходной цифры. Например, номер 1166 превратится в номер 2136. Вторая от конца цифра 6 была удвоена (до 12) и цифры результата (1 и 2) сложили вместе (для получения 3).

3. Сложите все цифры получившегося номера вместе. Если у вас в данный момент номер 2136, сложите вместе 2+1+3+6 (что даст 12).

4. Если результат заканчивается 0 (или иначе, если он кратен 10), номер карты допустимый. В противном случае - нет.

Алгоритм Луна проверяет, может ли предоставленный вами номер быть номером реальной кредитной карты. Но он делает только то, что делает. Этот алгоритм не может обнаружить номер кредитной карты, технически приемлемый, но на самом деле не присвоенный никакому банковскому счету (и ясно, что не может определить финансовое положение владельца счета и наличие на счету требуемого для покупки лимита).

Далее приведен полный программный код функции ValidateCard. Все строки кода пронумерованы, поэтому вы можете разбить его на элементарные порции (построчные объяснения даются после кода):

```
1Function ValidateCard(CardNumber As String)
' Это промежуточный итог (создаваемый с помощью алгоритма Луна)
2       Dim SumOfDigits
3       SumOfDigits = 0
' Эта переменная определяет, в какой вы находитесь позиции,
'  нечетной или четной.
' Вы начинаете с нечетной позиции (1)
4       Dim OddNumbered
5       OddNumbered = True
```

[2] В этом случае четные от начала номера, применяемого в качестве примера. - *Пер.*

```
 6  Dim i
7   For i = Len(CardNumber) To 1 Step -1
8   Dim CurrentNumber
9   CurrentNumber = Mid(CardNumber, i, 1)
10 If OddNumbered = False Then
' Цифра удваивается
```

```
11 CurrentNumber = CurrentNumber * 2
12 If CurrentNumber >= 10 Then '
' Если результат состоит из двух цифр, они складываются.
' Это странная часть, поскольку нужно использовать
' функции преобразования строк
13 Dim NumText As String
14 NumText = CurrentNumber
15 CurrentNumber = Val(Left(NumText, 1)) +
16          Val(Right(NumText, 1))
17 End If
18 End If
' К промежуточному итогу добавляется полученное число
19 "    SumOfDigits = SumOfDigits + CurrentNumber
 ‘Переход из нечетной позиции в четную или наоборот.
' Эта строка кода изменяет значение True на False или
' False на True
                20  OddNumbered = Not OddNumbered
                21  Next
 ' Если сумма кратна 10, номер допустимый
22 If SumOfDigits Mod 10 = 0 Then
23 ValidateCard = True
24 Else
25 ValidateCard = False
26 End If
27 End Function
```

Функция работает следующим образом.

- В *строке 1* объявляется функция. Обратите внимание на то, что у функции один параметр - текст с номером кредитной карты. Он явно определяется как строка с помощью оборота As String. Таким образом, вы избегаете ошибок, возникающих при попытке передать в функцию число.
- В *строках 2-3* создается переменная, хранящая в течение всего процесса обработки промежуточный итог.
- В *строках 4-5* создается переменная, которая следит за тем, в какой позиции номера вы находитесь, четной или нечетной, считая от конца номера.
- В *строках 6-7* начинается цикл For/Next. Этот цикл немного отличается от виденных вами ранее циклов, поскольку у него в конце есть выражение Step -1. Оно сообщает о том, что после каждого прохода цикла из счетчика вычитается 1 (в отличие от добавления 1 при стандартном поведении). Вы можете обрабатывать номер от конца к началу.

Примечание

В цикле применен еще один прием - нижнее граничное значение задается с помощью функции Len, которая возвращает длину текстового фрагмента. Другими словами, если номер кредитной карты состоит из 11 цифр, этот код выполняется 11 раз (один раз для каждой цифры).

- В *строках 8-9* извлекается цифра в текущей позиции, на которую указывает счетчик цикла. Функция Mid позволяет вырезать одну цифру.
- В *строке 10* проверяется, находится ли полученная цифра в четной или нечетной позиции, считая от конца номера.
- *Строки 11-17* выполняются, только если позиция цифры не кратна двум, считая от конца номера. В этом случае ее следует удвоить (строка 11). Если получилось двузначное число, его цифры нужно сложить (строки 13-15).
- В *строке 19* текущее число добавляется к промежуточному итогу. Если вы находитесь в четной позиции, число не менялось. Если вы в нечетной позиции, число было удвоено и скомбинировано.
- В *строке 20* гарантируется переход из четной позиции в нечетную позицию (и обратно).

■ В *строке 21* выполняется переход к строке 6 и повторение цикла для следующей цифры в номере кредитной карты.

■ В *строках 22-26* проверяется окончательный итог. Если он кратен 10, номер допустимый. Для проверки применяется операция Mod, выполняющая деление и извлекающая остаток от деления нацело (Если остатка нет при делении числа на 10, вы знаете, что оно без проблем делится нацело.)

Может потребоваться некоторое время для того, чтобы разобраться в работе функции от начала до конца и выяснить точно, что происходит, но в конце концов все это относится к основным элементам языка VB, таким как условия, циклы и переменные. Если вы действительно хотите изучить данный пример, его можно посмотреть в действии с помощью средств отладки.

После завершения создания функции, подобной ValidateCard, ее можно вызвать для проверки соответствия номера кредитной карты.

Далее приведен пример, реагирующий на ввод данных кредитной карты в текстовое поле, названное CardNumber:

```
Private Sub CardNumber_BeforeUpdate(Cancel As  Integer)
If ValidateCard(CardNuraber)  Then
MsgBox  "Your card is valid," Else.
MsgBox  "Your card is  invalid.  "  & _
"Did you forget  a number,  or are  you  trying to  cheat us?" Cancel = True
End If
End Sub
```

Для проверки выполните приведенный код и введите один из номеров ваших кредитных карт в поле CardNumber, как показано на рис. 17.3.

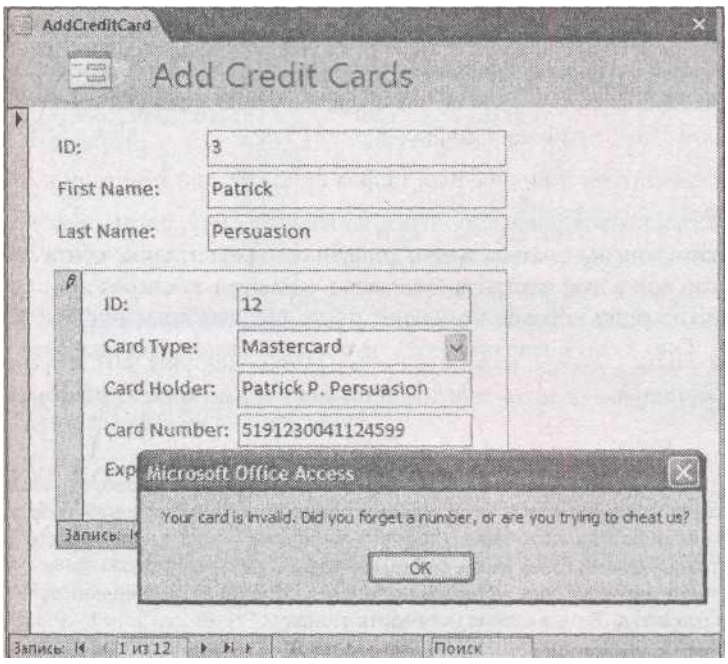


Рис. 17.3. Эта форма демонстрирует работу функции Validate Card в форме **AddCreditCard** (вставка кредитной карты) из БД Boutique Fudge . Когда бы ни менялось поле **CardNumber,** процедура проверяет его допустимость и отменяет изменения, если значение не приемлемо

17.2. *Обработка сбойных ситуаций*

Было бы хорошо сделать вид, что программа Access всегда успешно справляется с вашим программным кодом без малейшей заминки. Но следует признать, что ошибки все же возникают и случается это часто. Этот факт не должен пугать вас. В конце концов, одна из причин применения кода на языке Visual Basic вместо обычных макросов состоит в том, что вы можете с изяществом выявлять ошибки и реагировать на них.

Вы столкнетесь с двумя типами ошибок в вашем коде.

■ *Ошибки.* Это ошибки кодирования, которые вы вносите случайно. Обычно они обнаруживаются при тестировании вашей БД. (Если повезет, редактор Visual Basic заметит ошибку, как только вы введете ее, и затем выведет соответствующее предупреждающее сообщение.)

■ *Непредвиденные ограничения.* Эти ошибки возникают в определенных обстоятельствах, которые вы возможно не предусмотрели. Скажем, вы создаете две формы: **Order** и **Order_Subform.** Форма **Order_Subform** проектировалась для применения в качестве подчиненной формы в форме **Order** и содержит программный код, который обращается к элементам управления на форме **Order.** Если же кто-либо откроет непосредственно форму **Order_Subform,** форма **Order** окажется недоступной и этот программный код завершится аварийно.

Ваша задача, как добросовестного программиста, исправить все ошибки и обработать непредвиденные ограничения наилучшим образом. Для выхода из затруднений Visual Basic предлагает два средства. Можно использовать отладку для диагностики и устранения необычных проблем и код обработки ошибок для выявления неожиданных проблем и оповещения о них других пользователей.

17.2.1. *Отладка*

Отладка - это отличное средство, позволяющее пройтись по вашему коду, понаблюдать за его работой и заметить ошибки. Отладка программного кода аналогична отладке макроса в том смысле, что позволяет выполнить ваш алгоритм пошагово, оператор за оператором. Но у кода более мощное средство отладки, которое позволяет проверять сложные процедуры, циклы и условные операторы. Оно даже может показать, что в данный момент хранится в ваших переменных.

Подсказка

Настоящее преимущество отладки заключается в том, что она позволяет вам проверить ваши предположения. У каждого программиста есть собственные предположения о том, как работает фрагмент кода. Однако если код делает именно то, чего вы ждали, у вас, вероятнее всего, нет ошибок. С помощью отладки можно точно найти место, где код делает что-то неожиданное - когда вычисление дает странный результат, условный оператор отправляет неверным путем, цикл повторяется лишний раз и т. д. Затем можно исправить ошибку.

Самый легкий способ отладки - установка точек прерывания или останова, специального маркера, сообщающего программе Access о том, где вы хотите начать отладку. Когда Access достигает строки кода с точкой останова, программа приостанавливает выполнение кода. Затем она позволяет вам выполнять код с заданной вами скоростью, поочередно одну строку кода за другой.

Точки останова применяют следующим образом.

1. Найдите первую строку кода, которую хотите отладить.

Если нужно проверить подпрограмму целиком, начните с оператора Sub или Function. Если вы хотите проверить конкретную часть кода, перейдите к ней.

2. Щелкните слева кнопкой мыши для установки точки останова в этой строке (рис. 17.4). Каждая точка останова - это сигнал программе Access о месте начала отладки.

В некоторых строках кода нельзя поместить точки останова. Эти строки не содержат выполняемого кода, строки с пробелами, комментарии и объявления переменных. Все остальные строки - поле для игры по правилам.

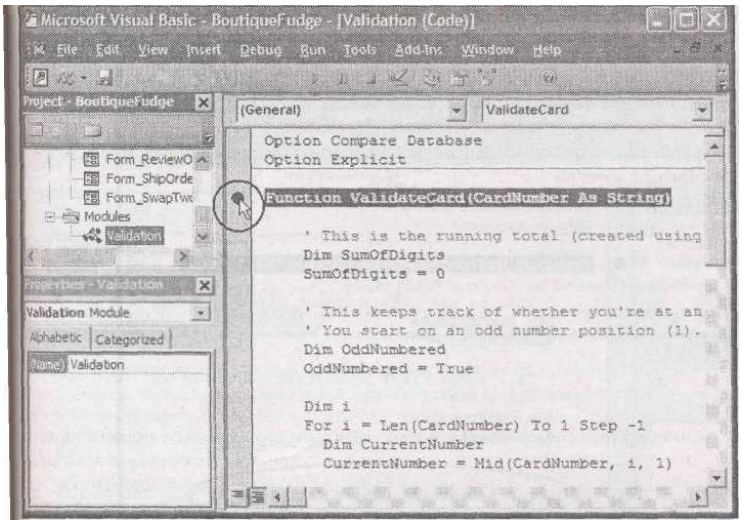


Рис. 17.4. Все точки останова выглядят как кружки красного цвета. Удалить точку останова можно, щелкнув ее кнопкой мыши. В данном примере точка останова (обведена) помещается вначале функции ValidateCard

Примечание

Когда вы закроете вашу БД и откроете ее позже, все точки останова исчезнут.

3. Запустите ваш код.

Вы можете начать выполнять код обычным образом. Если вы отлаживаете обработчик события для нажатия кнопки, откройте соответствующую форму и затем щелкните кнопку мышью.

Когда программа Access достигнет точки останова, она приостановит выполнение и переключится в режим прерывания (break mode). Все в вашем приложении замрет.

В режиме прерывания у вас есть несколько вариантов.

Можно выполнять код пошагово. Это означает, что вы выполняете по одному оператору, останавливаясь после каждого из них. Для опробования нажмите клавишу <F8>. Это действие выполняет текущий оператор (который выделен желтой стрелкой), переходит к следующему выполняемому оператору и снова останавливается (рис. 17.5). Вы можете продолжать, сколько захотите, нажимая клавишу <F8> для выполнения каждой строки кода.

Подсказка

Пошаговая отладка позволяет следить за работой кода. Если применить ее к функции ValidateCard, описанной ранее, вы увидите, как программа Access выполняет цикл несколько раз и как она переходит к разным секциям условного блока в зависимости от того, обрабатывается цифра в нечетной или четной позициях.

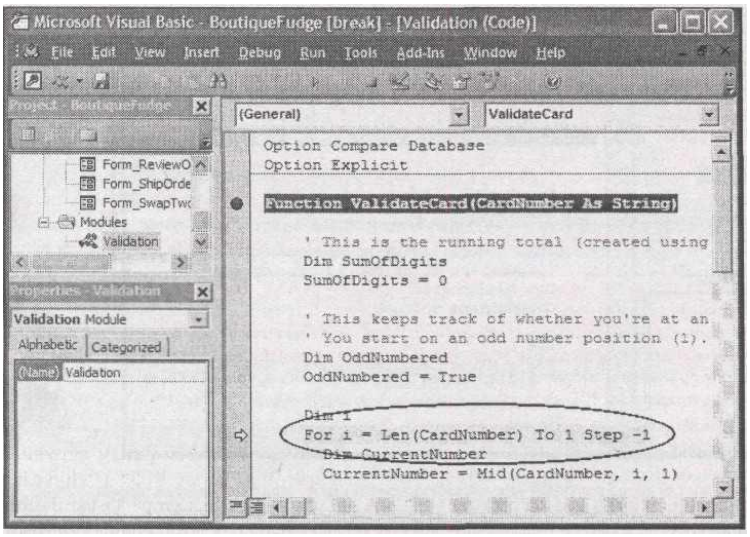


Рис. 17.5. В данном примере точка останова останавливает выполнение кода в начале функции ValidationCard. Затем пользователь, проводящий отладку, нажимает несколько раз клавишу <F8> для продвижения по коду. В данный момент код приостановлен в начале цикла For/Next (обведено)

Можно прекратить выполнение кода. Нажмите кнопку **Stop** (остановить) (она выглядит как квадрат) на панели инструментов редактора Visual Basic для завершения выполнения вашего кода.

Можно внести изменения. Если вы нашли ошибку, можно исправить ваш код и затем продолжить выполнение с внесенными изменениями. Конечно, существуют определенные типы корректировок, которые заставляют программу Access остановить отладку. Если вы внесли именно такое изменение, то увидите окно сообщения, предупреждающее о том, что "This action will reset your project" ("Это действие сбрасывает ваш проект"). Если щелкнуть мышью кнопку ОК, программа Access остановит выполнение вашего кода, как будто вы щелкнули мышью кнопку **Stop** (остановить) на панели инструментов редактора Visual Basic.

- *Можно просмотреть, что хранится в переменной.* Для этого просто проведите указателем мыши поверх имени переменной где-нибудь в вашем коде (рис. 17.6).

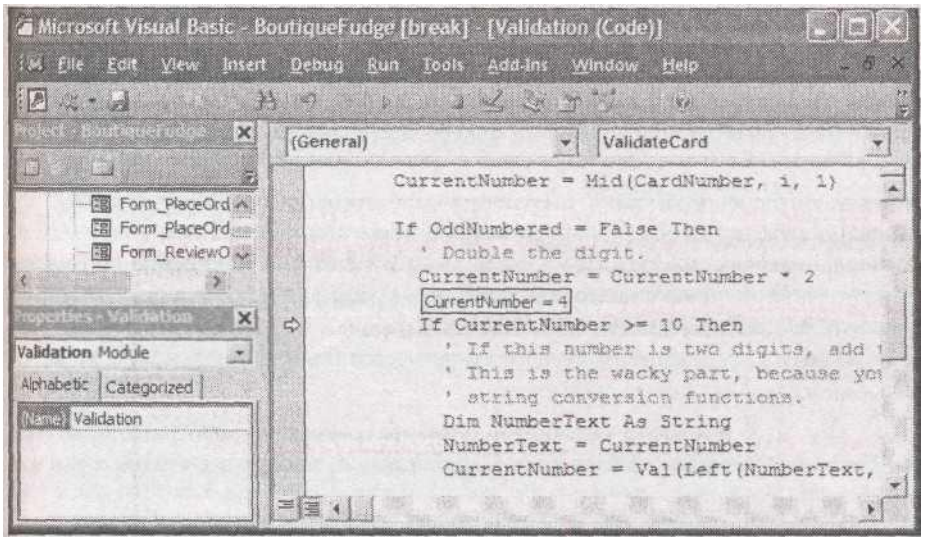


Рис. 17.6. Проведя указателем мыши поверх имени переменной CurrentNumber, можно увидеть, что в данный момент она хранит число 4. Можно провести указателем мыши поверх имен переменных в любой строке кода, кроме текущей строки. Если применяется клавиша <F8> для пошагового прохода кода, можно следить за изменением значения по мере выполнения операций

■ *Можно возобновить выполнение в нормальном режиме.* Если вы обнаружили источник проблемы и не хотите продолжать отладку, просто нажмите клавишу <F5> (или щелкните мышью кнопку Play (выполнить) на панели инструментов редактора Visual Basic). Программа Access выполнит текущую строку и затем продолжит выполнение быстрым способом (по крайней мере, до встречи с очередной точкой останова).

Подсказка

Вы можете выполнить ловкий трюк с желтой стрелкой. Ее можно использовать для выполнения строк, расположенных в другом месте кода. Просто перетащите стрелку с нажатой кнопкой мыши к строке, которую хотите выполнить следующей, и затем нажмите клавишу <F5> для выхода из режима отладки и возобновления выполнения кода в обычном режиме.

У редактора Visual Basic есть множество других средств отладки. Однако точек останова вполне достаточно для начала исследования того, что делается под капотом, когда выполняется ваш код.

17.2.2. *Обработка ошибок*

Некоторые ошибки возникают не по вашей вине. Быть может, вы пытаетесь выполнить задачу с данными, которые получаете от кого-то, и эти данные некорректны. Представьте себе, что произойдет, если кто-нибудь вызовет функцию ValidateCard и передаст в нее номер кредитной карты, содержащий буквы и знаки пунктуации!

Несмотря на то, что такие ошибки могут произойти в результате чьей-то небрежности, именно вы должны обработать их наилучшим образом. Следует объяснить возникшую проблему с помощью понятного окна сообщения и закончить текущую задачу (или перейти к следующему шагу). Вы можете позаботиться об этом, добавив код обработки ошибок.

Подсказка

Лучше всего с помощью отладки найти и исправить все проблемы во фрагменте кода. После завершения этого процесса можно добавить код обработки ошибок, рассчитанный на непредвиденные проблемы. Если код обработки ошибок вставить раньше, отладить ваше приложение, возможно, будет немного труднее.

Обычно, когда программа Access обнаруживает ошибку, она переходит к коду, вызвавшему проблему, переключается в режим прерывания и отображает сообщение об ошибке. Такое поведение полезно, если вы планируете устранять проблему, но оно лишь травмирует обычных пользователей, которые, возможно, работают с вашей БД. Мало того, что они никогда раньше не видели программного кода, они окажутся в большой опасности, попытавшись исправить его, и создадут новые проблемы.

Вместо этого вам нужен способ обработки ошибки средствами программного кода. В языке Visual Basic есть специальный оператор, сообщающий программе Access о том, как поступать с ошибками. Это оператор On Error.

Оператор On Error предоставляет несколько вариантов. Можно заставить программу Access пропустить ошибки и попытаться выполнить очередную строку кода, например, следующим образом:

```
On Error Resume Next
```

Этот вариант почти всегда - не лучший выбор. Если ошибка возникла, за ней вероятнее всего последуют другие. В худшем случае такая ситуация может вынудить вашу программу делать вовсе не то, для чего она предназначена.

Можно также заставить программу Access перейти в конкретное место кода. Далее приведен пример.

```
On Error Goto ErrorHandlingCode
```

В данном примере программа Access переходит к разделу, названному ErrorHandlingCode, как только она обнаруживает какую-либо проблему. Вы должны обозначить этот раздел, указав в отдельной строке его имя и следом за ним вставив двоеточие (:), например, так:

```
ErrorHandlingCode:
' Если возникла ошибка, Access начинает выполнять ваш код с этой точки
```

Очень легко понять, как действует система обработки ошибок, если рассмотреть ее использование на примере функции ValidateCard:

```
Function ValidateCard(CardNumber As String)
On Error Goto ErrorHandlingCode

' Здесь расположен код, реализующий алгоритм Луна
Exit Function
ErrorHandlingCode:
MsgBox "Oops. Did your credit card number have letters?"
ValidateCard = False
End Function
```

Перечислим несколько важных деталей. Во-первых, оператор On Error помещается в

самом начале программного кода процедуры, поэтому вы можете обнаружить ошибки, возникшие в любом месте последующего кода. Во-вторых, обратите внимание на то, что после ого, как закончен код проверки номера, процедуру завершает оператор Exit Function. Этот оператор не дает программе Access попасть в следующий далее код обработки ошибок, если никакой ошибки не произошло. Наконец, код обработки ошибок выводит окно сообщения, в котором сообщается о нарушении естественного хода событий и возвращается результат, ясно обозначающий проблему. Чаще всего разработчики именно так обрабатывают ошибки. Только помните об обязательном использовании оператора Exit Sub или Exit Function, чтобы избежать случайного выполнения кода обработки ошибок.

Примечание

Как было указано ранее, пользователь, применяющий форму **AddCreditCard,** может получить два типа сообщений об ошибке - одно, объясняющее проблему, связанную с включением в номер карты букв и знаков пунктуации, и второе, констатирующее очевидный факт недопустимости номера. Если это сообщение кажется ненужным наказанием, вы можете перенести код обработки ошибок из функции ValidateCard в обработчик события **При обновлении** (On Update), с которым он на самом деле связан. В этом случае в обработчике события **При обновлении** (On Update) можно выбрать точный способ решения проблемы. Для того чтобы увидеть преобразованный код, посмотрите загружаемые из Интернета примеры к данной главе.

У вас есть еще только один вариант обработки ошибок. Можно заставить программу Access немедленно остановить выполнение и перейти в режим отладки с помощью следующего оператора:

```
On Error Goto 0
```

Конечно, такое поведение уже стало стандартным при обработке ошибок. Указанный оператор следует применять только, если вы переключаетесь многократно между разными методами обработки ошибок в одной и той же процедуре.

17.3. *Углубленное рассмотрение объектов*

В жизни любого программиста Access наступает момент, когда вы осознаете, что знаете достаточно о языке VB, чтобы сводить концы с концами. С этого момента вы будете проводить большую часть времени, изучая различные объекты, а это гораздо более трудоемкая задача.

В программе Access есть несколько десятков встроенных объектов, которые собранные вместе формируют то, что программисты называют *объектной моделью.* Наряду с объектами элементов управления и форм, которые вы хорошо знаете, в программе есть объекты, представляющие запросы, проекты, отчеты, смарт-теги, принтеры и многое другое. Вы не сможете познакомиться со всеми этими объектами в одной главе. Но даже если бы могли, то обнаружили бы, что многие из них вам просто не интересны. Однако следует знать достаточно для того, чтобы найти нужные вам средства, когда принимаетесь за особенно трудную задачу на языке VB.

Вы можете изучить объектную модель программы Access несколькими способами:

- воспользоваться справочной системой программы Access (см. указания по поиску нужной информации в разд. *"Применение объектов" главы 16);*
- использовать интерактивное руководство по языку VBA (Visual Basic for Applications), предоставляемое корпорацией Microsoft (перейдите на страницу **http://msdn.microsoft.com/office/reference/vba).**

Даже если вы освоили раскидистую объектную модель программы Access, за ее пределами остается еще много дополнительных объектов. Если же вы обладатель черного пояса как VB-программист, то можете выбрать вариант создания собственных объектов. Если нет, возможно, вы решите применить какой-нибудь *компонент,* предоставляющий еще больше объектов для работы.

Примечание

На языке программистов *компонент* - это просто файл, содержащий некоторые объекты,

которые можно использовать в вашем программном-коде. В файле acedao.dll есть объекты, которые можно применять для непосредственного взаимодействия с вашей БД *(см. разд. "Обновление единиц наличного запаса" далее в этой главе).*

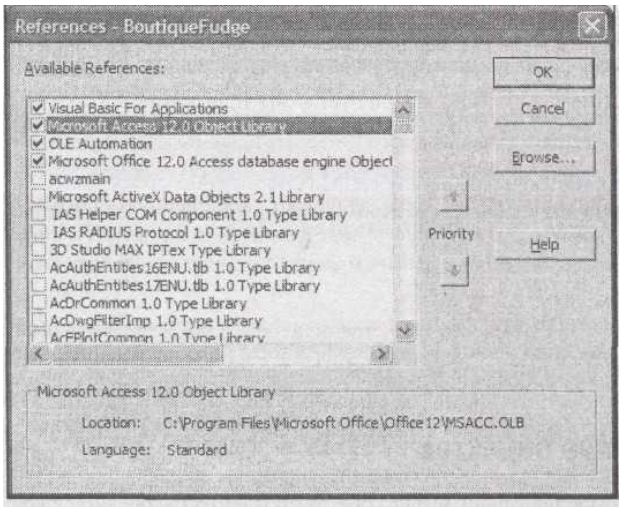


Рис. 17.7. Для добавления ссылки на компонент, который хотите использовать, найдите его в списке и затем установите флажок, расположенный рядом с ним. Компоненты, на которые есть ссылки в данный момент, приведены в верхней части списка. Здесь показаны объекты, на которые автоматически устанавливаются ссылки в каждой БД, - объекты, встроенные в язык Visual Basic и поставляемые вместе с программой Access, и объекты доступа к данным, которые можно применять для непосредственного чтения и редактирования БД

Позже в этой главе вы узнаете, как использовать DAO (Data Access Objects, объекты доступа к данным) для взаимодействия с вашей БД. Технология DAO - настолько популярная составляющая программирования в Access, что большинство считают эту библиотеку встроенной частью объектной модели Access. Однако технически DAO состоит из набора объектов, предоставляемых отдельным компонентом, поддерживаемым программой Access. Множество дополнительных компонентов ждут, чтобы вы нашли их.

Для применения нового компонента необходимо добавить ссылку на него в вашу БД. Для этого в меню редактора Visual Basic выберите последовательность команд **Tools → References** (Сервис → Ссылки). Вы увидите диалоговое окно **References,** показанное на рис. 17.7.

Проблема диалогового окна **References** заключается в том, что вам нужно точно знать, какой компонент вы хотите использовать. Список **Available References** (доступные ссылки) полон компонентов со звучными именами, которые не предназначены для использования в программе Access и не будут корректно работать с вашим программным кодом. Среди компонентов, которыми можно воспользоваться, есть компоненты Microsoft, позволяющие взаимодействовать с другими приложениями пакета Office. Но, самостоятельно экспериментируя, вы не многого добьетесь. Следует найти пример кода в Интернете или в справочной системе программы Access.

Часто задаваемый вопрос.

Запуск других Windows-программ

Как мне открыть Word (или Excel, или Блокнот (Notepad) или танцевальную видеоигру Dance Dance Revolution)?

В язык Visual Basic включена функция shell, которая позволяет запускать другую программу. Для применения функции Shell необходимо задать полный путь, указывающий на файл программы. Вот приведен пример запуска Windows-программы Калькулятор:

```
Shell "C:\Windows\calc.exe"
```

Когда вы применяете функцию shell, ОС Windows запускает запрошенную программу, а ваш код продолжает выполняться. Но у вашего программного кода нет реальной возможности взаимодействовать с программой. Вы не можете заставить ее сделать что-либо или выяснить,

что она закрыта.

Shell кажется удобной функцией, но у нее есть существенная проблема. Для того чтобы использовать функцию Shell, нужно знать точное местонахождение программы. Вы не можете просто сказать: "Запусти Microsoft Word" или "Открой этот документ". Вместо этого вы должны глубоко зарыться в файловую систему жесткого диска, чтобы найти файл нужной программы (который обычно находится где-то в зоне вашего компьютера с именем Program Files). Хуже того, после того как вы заставили функцию Shell работать на вашем компьютере, нет никакой гарантии, что она заработает на какой-либо другой машине - в конце концов, та же программа может быть установлена где-то совсем в другом месте.

И как с этим бороться? Можно воспользоваться гиперссылкой, которая запускает нужную программу автоматически, если по ссылке щелкнуть кнопкой мыши. Но некоторые программы, включая других членов семейства Microsoft Office, предлагают лучший вариант. Они предоставляют собственные объекты, которыми можно манипулировать в коде на языке Visual Basic. Благодаря этим объектам можно применять эти программы, не беспокоясь об их местонахождении. Вы конечно же можете делать с ними гораздо больше, задавая различные свойства и вызывая разнообразные методы.

Можно заставить программу Word открыть документ, добавить в него некоторый текст, отправить 10 копий на принтер и затем завершить программу.

Объекты, реализующие этот процесс, не рассматриваются в данной книге, но далее приведен очень простой пример, который запускает программу Word, выводит на экран окно программы и загружает в нее документ GothicWedding.doc:

```
Dim Word  As   Object
Set Word = CreateObject("Word.Application")
Word.Visible  =  True
Word.Documents.Open  CurrentProject.Path   &   "\GothicWedding.doc"
```

Если эта технология заинтересовала вас, обратитесь к справочной системе программы Word, из нее можно узнать гораздо больше об объектной модели Word. Другой полезный ресурс - Microsoft's Office Developer Center (Центр разработчиков Microsoft Office) на Web-сайте **http://msdn.microsoft.com/office.**

17.3.1. *Объект DoCmd*

Объект DoCmd - единственный наиболее полезный объект в мире программирования Access. Он обеспечивает "покупку всего нужного в одном месте" для самых разнообразных задач, таких как открытие форм и отчетов, запуск других программ, поиск записей и выполнение макросов.

В отличие от виденных вами ранее объектов, у объекта DoCmd нет никаких свойств. Вместо этого он состоит из методов, выполняющих разные действия. Если нужно открыть форму с именем **ProductCatalog,** можно использовать метод OpenForm следующим образом:

```
DoCmd.OpenForm "ProductCatalog"
```

Как большинство методов объекта DoCmd, OpenForm может использовать несколько необязательных параметров. Visual Basic подскажет, отобразив список возможных параметров в процессе ввода имени метода. Далее показан пример, в котором пропущены второй и третий параметры (обратите внимание на запятые без значений между ними), но задается фильтр в четвертом параметре и режим данных в пятом:

```
DoCmd.OpenForm  "ProductCatalog",   ,   ,"ID=5",   acFormReadOnly
```

Эта команда открывает форму **ProductCatalog,** применяет фильтр для вывода на экран одной записи с ID (Код), равным 5, и использует режим "только чтение" для запрета каких-либо изменений.

Примечание

В данном примере используется *константа* acFormReadOnly. Константы - это числовые значения, которым присвоены более информативные имена. Таким образом вместо запоминания числа, обозначающего режим "только чтение", можно применять более осмысленную константу acFormReadOnly. Всегда, когда встречается переменная, начинающаяся с ac или vb, и вы ее не создавали сами, знайте, что это константа. Конечно, для того чтобы пользоваться

константами, нужно все-таки знать их имена, но в этом может помочь средство IntelliSense, как показано на рис. 17.8.

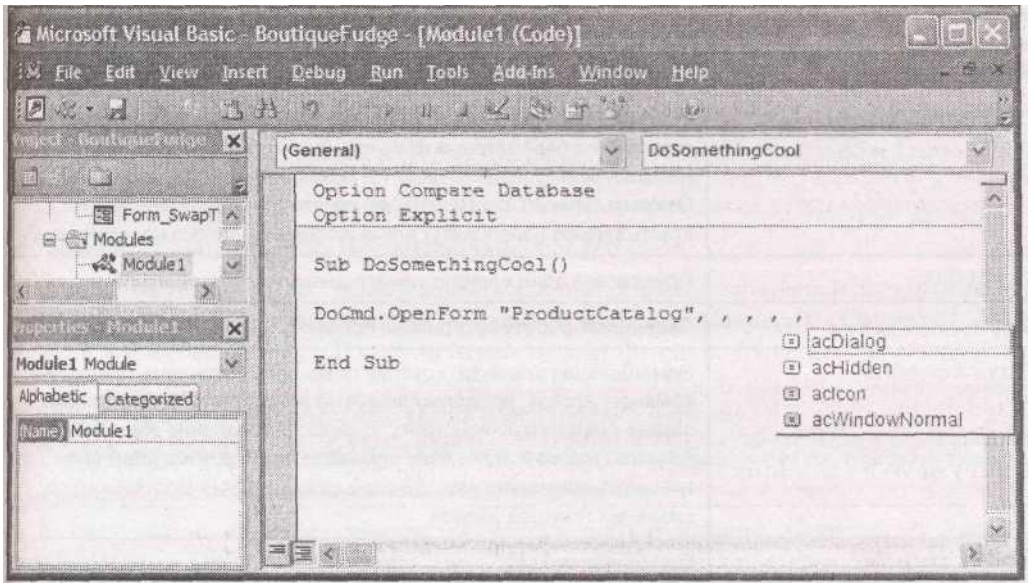


Рис. 17.8. Когда вы добираетесь до параметра режима данных, редактор Visual Basic выводит на экран список всех допустимых констант, которые можно использовать. Для того чтобы выяснить, что они означают (если это не очевидно), следует обратиться к справочной системе Access

Метод OpenForm может показаться знакомым, потому что вы уже видели такие же функции в макрокоманде **ОткрытьФорму** (OpenForm) *(см. главу 15).* В действительности все методы объекта DoCmd соотносятся с макрокомандами, которые вы изучали в *главе15.* В табл. 17.1 перечислены наиболее полезные методы.

Таблица 17.1. Полезные методы объекта DoCmd

Метод	*Описание*
ApplyFilter	Применяет фильтр к таблице, форме, запросу или отчету для того, чтобы сконцентрировать внимание на интересующих вас записях
Beep	Производит некоторый сигнал. Обычно используется для привлечения внимания к возникшей проблеме
Close	Закрывает текущий объект БД (или конкретный объект, который задан)
CopyDatabaseFile	Предоставляет быстрый способ создания резервной копии БД
FindRecord, FindNext и GoToRecord	Предоставляет разные способы поиска нужной записи
Hourglass	Включает отображение указателя мыши в виде песочных часов (или выключает). Применяется для того, чтобы дать знать пользователю о том, что выполняется требующая времени задача и следует остыть
OpenForm, OpenQuery, OpenReport и OpenTable	Открывает соответствующий объект БД в нужном вам режиме представления с параметрами фильтрации и необязательными уточнениями. Как вы узнали в главе 15, можно применять макрокоманду ОткрытьОтчет (OpenReport) для печати отчетам команду От-крытьЗапрос (OpenQuery) для выполнения запроса на изменение
Printout	Предлагает один вариант печати данных из текущего объекта БД
Quit	Завершает работу программы Access
RunCommand	Замещающая команда, которая позволяет выполнить различные команды Access, представленные на ленте. Необходимо только задать правильную константу. В разд. "Управление выполнением заказов" далее в этой главе приведен пример, в котором программист применяет метод RunCommand для немедленного сохранения текущей записи
RunMacro	Выполняет макрос

RunSQL	Выполняет групповой SQL-оператор. Эту команду нельзя использовать для извлечения данных из вашей БД. Она позволяет выполнять команды, которые изменяют записи или таблицы
ShowAllRecords	Удаляет параметры текущего фильтра, поэтому можно увидеть все записи в таблице, форме, запросе или отчете

17.3.2. *Преобразование макроса в VB-код*

Если хотите узнать больше о языке Visual Basic и объекте DoCmd, можно взять существующий макрос и преобразовать его в чистый программный код, подпрограмму. Далее приведены необходимые действия.

1. В области переходов выберите макрос, который хотите использовать.
2. Выберите на ленте **Работа с базами данных → Макрос → Преобразовать макросы** (Database Tools → Macro → Convert Macros to Visual Basic). Можно также преобразовать внедренный в форму макрос, открыв форму и выбрав на ленте **Работа с базами данных → Макрос → Преобразовать макросы формы** (Database Tools → Macro → Convert Form's Macros to Visual Basic).

На экране появится окно с двумя параметрами для выбора (рис. 17.9).

Рис. 17.9. Вы увидите это окно, если попросите программу Access преобразовать макрос **FindHayEater**

3. Если хотите добавить базовую обработку ошибок, убедитесь в том, что флажок **Добавить программу обработки ошибок** (Add error handling to generated functions) установлен.

Немного обработки ошибок никогда не помешает.

4. Если хотите включить комментарии макроса в комментарии VB, убедитесь в том, что установлен флажок **Добавить примечания макросов** (Include macro comments).

Если вы потратили время на добавление пояснительного текста, стоит сохранить его.

5. Щелкните мышью кнопку **Преобразовать** (Convert).

Программа Access создаст новый модуль для преобразованного кода и даст ему имя, подобное имени **Преобразованный макрос-*[ИмяВашегоМакроса]*** (Converted Macro-[YourMacroName]). Внутри модуля Access создаст функцию с именем как у вашего макроса. Если преобразуется группа макросов *(см. разд. "Группы макросов" главы 15),* Access вставит по одной подпрограмме для каждого макроса в группе.

После завершения процесса преобразования программа Access откроет ваш модуль в редакторе Visual Basic, чтобы вы могли просмотреть код.

В следующем примере показан результат преобразования макроса из *главы 15* (приведенного в *разд. "Поиск записи" главы 15),* который ищет определенный текст в таблице **AnimalTypes:**

```
Function FindHayEater (     )
On Error GoTo FindHayEater_Err
 DoCmd.OpenForm "AnimalTypes",   acNormal,   " ",   " ",   ,   acNormal
DoCmd.GoToControl   "Diet"
DoCmd.FindRecord  "=""hay""",   acAnywhere,   False,  , _
False,   acCurrent,   False

FindHayEater_Exit : Exit  Function
FindHayEater_Err:
MsgBox Error$
Resume  FindHayEater_Exit End Function
```

Вы заметите, что в преобразованном коде интенсивно используется объект DoCmd -

действительно почти в каждой строке кода встречается объект DoCmd. Сначала он применяется в методе OpenForm для открытия формы, затем - в методе GoToControl для перехода в поле **Diet** и, наконец, он ищет первую запись, в которой встречается текст "hay". Эта строка выглядит причудливо, поскольку в ней удваиваются знаки кавычек (""). В языке Visual Basic знаки кавычек имеют особый смысл (они показывают, где начинается и заканчивается текст). Если в текстовом фрагменте вы хотите на самом деле использовать кавычки, нужно поместить знаки кавычек дважды, одни за другими. Стрнно, но правильно.

Завершается код процедурой обработки ошибок, названной FindHayEater__Err, которая просто сообщает о проблеме в окне сообщения и затем завершает работу.

Примечание

Когда макрос преобразуется в программный код, программа Access всегда генерирует функцию, а не процедуру. Но функция не возвращает результат, т. к. этого не требуется. (По-видимому, Access действует таким образом, чтобы дать вам возможность позже воспользоваться возвращаемым результатом.)

17.4. *Улучшение работы компании средствами Visual Basic*

На протяжении последних 16 глав вы узнали и полюбили БД компании Boutique Fudge, которая представляет собой БД действительных продаж, отслеживающую данные о клиентах, товарах и заказах. Однако, несмотря на то, что в БД Boutique Fudge хранится вся необходимая информация, она все еще не полностью интегрирована в повседневную деятельность компании. И прежде чем вы попытаетесь это исправить, следует понять, почему выигрыш столь мал.

Большинство людей, работающих в компаниях, подобных Boutique Fudge, не думают о таблицах и операциях над данными (таких как добавление, обновление и удаление записей). Они мыслят задачами, например, размещение заказа, доставка заказа и обработка жалобы клиента.

Многие задачи тесно связаны с операциями над данными и в этом случае у вас нет проблем. Задача "регистрация нового клиента" включает открытие таблицы **Customers** и последующее добавление новой записи. Следить за ней можно с помощью простой формы. Задача "размещение заказа" немного сложнее. Она включает добавление записей в несколько таблиц (таблицы **Orders** и **OrderDetails)** и использование данных из связанных таблиц (таблицы **Products** и **Customers)** для заполнения заказа. Можно создать обычную форму для выполнения этой работы, но она не будет действовать так, как хотели бы продавцы (рис. 17.10).

То же справедливо и в отношении задачи "доставка заказа". Этой задаче требуется несколько шагов - изменение статуса заказа, регистрация отправки заказа, обновления количества единиц товара на складе. Вы можете интерпретировать эту задачу как несколько операций над данными, но гораздо лучше создать единую форму, которая будет заботиться о процессе в целом.

Сейчас очень пригодится VB. С помощью подходящего программного кода вы сможете спроектировать интеллектуальную форму, которая будет соответствовать методам работы сотрудников компании. Интеллектуальная форма - это не просто способ добавления, редактирования и удаления записей в таблице - это средство, помогающее вести коммерческую деятельность. В следующих разделах вы увидите, как разрабатывать улучшенные формы с некоторыми программируемыми свойствами. К этим формам относятся следующие:

- **PlaceOrder** позволяет создать новый заказ. Она действует в месте с подчиненной формой **PlaceOrder_Subform,** позволяющей включать отдельные товары в заказ;
- **AddProduct** позволяет создать новый товар. Вы можете использовать ее непосредственно из формы **PlaceOrder** для вставки товара внутрь заказа;

ShipOrders позволяет обновить заказ сведениями о доставке. Она также работает с формой ReviewOrderDetails для вывода на экран компонентов заказа.

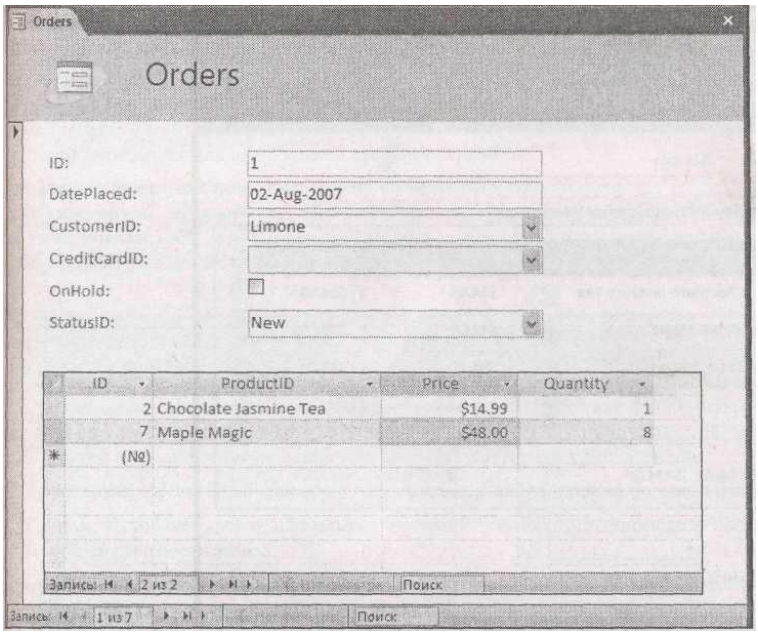


Рис. 17.10. Эта форма позволяет добавлять записи в таблицы Orders и OrderDetails. Но ей не хватает нескольких украшений, которые пользователи рассчитывают увидеть на форме для заказа - например, автоматическое заполнение поля с ценой каждого товара, заказанного вами, вычисление промежуточных итогов по мере заполнения заказа и возможность добавить товар на лету

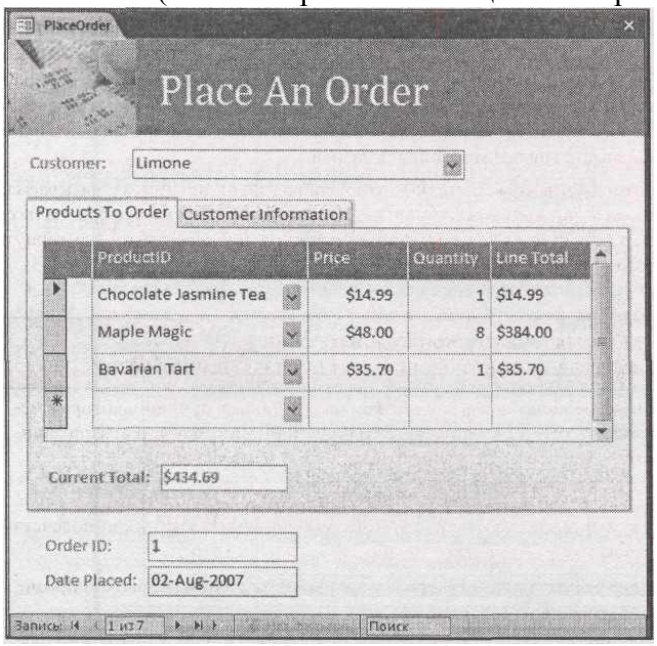
Проверить конечный результат можно с помощью загружаемых из Интернета БД, предназначенных для данной главы *(см. разд. "Примеры" во введении).*

Подсказка

Всегда хорошо называть форму в соответствии с выполняемой ею задачей (размещение заказа, доставка заказа и т. д.), а не таблицей, которую она использует. Такой подход поможет вам запомнить, кто пользуется каждой формой, поэтому вы сможете приспособить ее для соответствующей аудитории.

17.4.1. *Хранение промежуточного итога*

Лишь немногие клиенты настолько бесстрашны, чтобы поместить заказ без точных сведений о его стоимости. В типичной форме для заказа в строке отображается стоимость каждого элемента (за счет перемножения цены товара и его количества) и еще более важная ито-



говая стоимость заказа (рис. 17.11).

Рис. 17.11. Форма PlaceOrder с промежуточными итогами и общим итогом

Примечание

Форма **PlaceOrder** также включает несколько уже знакомых вам тонкостей, например, размещение адресной информации клиента на отдельной вкладке, перенос автоматически генерируемых полей (идентификационный номер заказа ID и дата заказа) в нижнюю часть окна, где они не будут никого отвлекать, и установка в их свойстве **Блокировка** (Locked) значения Да для запрета изменений. В свойстве формы **Ввод данных** (Data Entry) также установлено значение Да, что позволяет начать создавать новый заказ сразу после открытия формы.

Код не нужен только для строчного итога. Действительно, эту проблему можно решить, добавив элемент управления **Поле,** использующий следующее выражение в подчиненной форме **PlaceOrder_Subform:**

=Quantity * Price

Это выражение действует, поскольку нужная информация (поля **Price** и **Quantity)** располагается на той же форме, что и вычисляемое поле. А вот общий итог получить не так легко.

Для пущей важности можно соединить это выражение с функцией Format, чтобы быть уверенным в том, что выводится нужное число десятичных знаков и символ валюты ($):

=Format (Quantity * Price, "Currency")

Для вычисления общей суммы необходимо использовать данные полей **Quantity** и **Price** в таблице **OrderDetails.** К сожалению, у формы **PlaceOrder** нет легкого способа получить эту информацию. Дело не только в том, что она находится где-то еще (на подчиненной форме), но и в том, что она включает несколько отдельных записей. Даже если извлечь данные полей **Quantity** и **Price** из подчиненной формы, можно будет получить значения только для текущей записи, а не для всего списка заказанных товаров.

Для решения этой проблемы нужна специализированная функция Access, называемая *статистической функцией по подмножеству* или *функцией обработки набора записей* (domain function). Функция по подмножеству может обработать целую таблицу и вернуть одну порцию данных. Дополнительную информацию см. в следующем разделе.

На профессиональном уровне.

Станьте знатоком статистических функции по подмножеству

Статистические функции по подмножеству похожи на групповые функции, которые использовались при подсчете итогов в запросах. Эти функции принимают диапазон записей, затем выполняют вычисления или поиск и возвращают одно значение.

В программу Access включено восемь статистических функций по подмножеству.

- DSum вычисляет сумму нескольких значений. Ее можно использовать для подсчета общей стоимости заказа.
- DAvg рассчитывает среднее арифметическое нескольких значений. Ее можно применить для расчета средней цены товаров.
- DCount считает количество соответствующих записей. Она используется для вычисления числа элементов в заказе или количества заказов, сделанных клиентом.
- DMin и DMax находят наименьшее или наибольшее значение в подмножестве. Их можно использовать для поиска удешевленных или самых дорогих товаров.
- DFirst и DLast извлекают первое или последнее значение из подмножества. Если отсортировать список заказов по датам, можно найти самый старый и самый свежий заказы.
- DLookup находит значение, удовлетворяющее заданным критериям. Ее можно применять для просмотра таблицы и поиска названия товара с заданным ID.

Все статистические функции по подмножеству принимают три одинаковых параметра. Первый - поле (или вычисляемое выражение), которое хотите извлечь или использовать в вычислении. Второй - применяемая таблица или запрос. Третий параметр содержит любые условия отбора, используемые для сокращения числа строк. Если вы хотите найти среднюю

цену всех напитков, продаваемых компанией Boutique Fudge, нужно использовать поле **Price** (в качестве первого параметра), таблицу **Products** (как второй параметр) и отфильтровать ее, включив в подсчет товары с категорией Beverages (напитки) (третий параметр).

Для вычисления стоимости всех компонентов заказа применяется функция DSum. Нужная вам информация хранится в таблице **OrderDetails,** но вы хотите отобрать только те записи, у которых поле **OrderlD** совпадает с идентификационным номером текущего заказа. Наконец, нужно сложить вместе стоимости всех компонентов заказа. И как вы знаете из ранее изложенного, стоимость в строке заказа вычисляется перемножением полей **Price** и **Quantity.**

Держа все это в голове, можно создать следующее вычисляемое поле:

=DSum("Price*Quantity","OrderDetails","OrderID=" & [ID])

Первый аргумент - вычисляемое поле, которое берется из каждой записи. Второй аргумент - имя используемой таблицы. Третий аргумент отбирает только те записи, которые соответствуют текущему заказу. Если у текущего заказа идентификационный номер (ID) 455, последний параметр отберет все записи из таблицы **OrderDetails,** у которых OrderID=4 55. И снова вы сможете охватить все разом с помощью функции Format, если хотите, чтобы окончательный результат выглядел как денежная сумма.

У данного вычисляемого поля есть одна хитрость, но сначала придется внести еще одно усовершенствование. Обычно программа Access подсчитывает вычисляемые ноля при первом отображении записи. Однако вам нужна гарантия того, что общий итог вычисляется заново при каждом изменении в списке элементов заказа. Для этого необходимо вызывать метод Form.Recalc, когда запись из таблицы **OrderDetails** добавляется, обновляется или удаляется. Далее приведен программный код, реализующий этот прием:

```
Private  Sub  Form_Afterlnsert()
Forms("PlaceOrder").Recalc
End Sub
Private Sub Form_AfterUpdate()
Forms("PlaceOrder").Recalc
End Sub
Private Sub Form_AfterDelConfirm(Status As Integer)
Forms("PlaceOrder").Recalc
End Sub
```

Теперь можно создать и заполнить заказ, не строя догадок о его общей стоимости.

17.4.2. *Получение сведений о цене*

Как вы узнали из *главы 5,* иногда в таблице приходится хранить моментальные данные - информацию, которая копируется из одной таблицы в другую, поскольку может меняться со временем. Хороший пример - цены товаров, которые эволюционируют со временем ("эволюционирование" - это мягкий аналог "неуклонного роста"). Итак, у текущего товара необязательно та же цена, по которой вы заказывали его на прошлой неделе. Для того чтобы отслеживать величину вашего долга компании, в таблице **OrderDetails** нужно хранить продажную цену товара.

Но эта система создает проблему при заполнении заказа. Выбрать компонент заказа довольно легко - нужно выделить товар в списке подстановки. Но список подстановки устанавливает только поле **ProductID** для записи таблицы **OrderDetails.** Целиком ваша задача - выяснить правильную цену и скопировать ее из таблицы **Products** в новую запись.

К счастью, это можно сделать довольно легко. Можно отреагировать на событие On Change (Изменение) в списке **ProductID,** которое возникает при каждом выборе товара. Затем можно применить статистическую функцию по подмножеству DLookup для поиска соответствующей цены и вставить ее в поле **Price** автоматически. Далее приведен код, делающий это:

```
Private  Sub ProductID_Change (     )
Price = DLookup("Price",   "Products",   "ID="  & ProductID)
Quantity =1
End Sub
```

В данном коде также задается значение 1 для поля **Quantity,** что служит важной отправной точкой. Если необходимо, можно изменить значения полей **Price** и **Quantity** после того, как товар выбран. Или создать более строгую форму - можно задать значение *Да* в свойстве, Блокировка (Locked) элемента управления **Price,** чтобы запретить любые изменения цены 1 (как сделано в БД Boutique Fudge). В этом случае, когда создается заказ, вы вынуждены использовать действующую в данный момент цену без возможности скидок.

Примечание

Этот метод можно применять для заполнения других моментальных или зависящих от времени данных. Можно извлечь адресные данные текущего клиента и использовать их как отправную точку для адреса доставки. Можно даже применить функцию DLookup для создания более сложных процедур проверки на значения. Можно использовать этот способ в БД школы Cacophone Music для поиска предварительных условий и максимального размера класса, прежде чем зачислить студента в данный класс.

17.4.3. *Добавление нового товара во время заполнения заказа*

Boutique Fudge - управляемая клиентами компания. Если кому-то нужен самый новый товар, которого еще нет в каталоге товаров (например, картофель в шоколадной глазури), компания готова создать его по требованию.

Обычно список подстановки для поля **ProductID** не разрешает такого рода создание нового товара на лету. Если попытаться ввести название несуществующего товара, вы получите строгую отповедь от программы Access. Но добавление новых элементов в список на ходу - распространенный метод программирования в Access и специальное событие разработано, чтобы помочь вам в этом: **Отсутствие в списке** (On Not In List).

Если ввести несуществующий товар и применить событие **Отсутствие в списке** (On Not In List), программа Access начнет с кода обработки события. Можно создать элемент списка, вывести сообщение или исправить проблему до того, как Access выразит недовольство.

У события **Отсутствие в списке** (On Not In List) два параметра: NewData и Response. NewData - это данные, которые набираются в поле списка и которых еще нет в списке. Response - это значение, предоставляемое для того, чтобы сообщить программе Access о том, как решать проблему.

Далее приведен базовый скелет подпрограммы, создаваемой Access, если выбрана обработка события **Отсутствие в списке** (On Not In List) для поля с именем **ProductID:**

```
Private Sub ProductID_NotInList(NewData As String, Response As Integer) End Sub
```

Когда возникает событие **Отсутствие в списке** (On Not In List), прежде всего, нужно спросить пользователей, работающих с формой, означает ли это - желание ввести несуществующий товар. Выполнить этот шаг можно с помощью знакомой функции MsgBox, используемой необычным образом. Сначала нужно сообщить программе Access о необходимости создать окно сообщения с двумя кнопками: **Да** (Yes) и **Нет** (No). Затем нужно суметь перехватить возвращаемое функцией MsgBox значение, чтобы определить, какая кнопка была нажата:

```
Dim ButtonClicked
ButtonClicked = MsgBox("Do you want to add a new product?", vbYesNo)
```

В данном коде создается переменная ButtonClicked и затем отображается сообщение. Когда пользователь закрывает окно сообщения (щелкнув мышью кнопку Да или Нет), Visual Basic помещает в переменную ButtonClicked число, которое сообщает вам о том, что произошло. Число равно 6, если была нажата кнопка Да, и 7, если была нажата кнопка Нет. Но вместо того, чтобы обрабатывать непосредственно числа и увеличивать риск ошибки, можно воспользоваться полезными константами vbYes (которая равна 6) и vbNo (которая равна 7).

Далее приведен до некоторой степени законченный код обработчика события **Отсутствие в списке** (On Not In List). Он выводит на экран сообщение, запрашивающее о необходимости добавления нового элемента в список (рис. 17.12), и затем отменяет редактирование списка, если пользователь, работающий с формой, нажал кнопку **Нет:**

```
Private Sub ProductID_NotInList(NewData As String, Response As Integer) '
Отображает сообщение Да/Нет и получает результат
```

```
Dim ButtonClicked
ButtonClicked = MsgBox("Do you want to add a new product  for "  & _
NewData  & "?",  vbYesNo)
'  Visual Basic предоставляет удобные константы vbYes и vbNo,
' которые можно использовать для определения нажатой кнопки
If ButtonClicked = vbNo Then
'  Отмена редактирования
ProductID.Undo
'   Сообщает Access о запрете вывода сообщения об ошибке.
'  Вы уже обработали ее
Response = acDataErrContinue
 Else
'   (Поместите сюда код для добавления нового товара в список)
End If End Sub
```

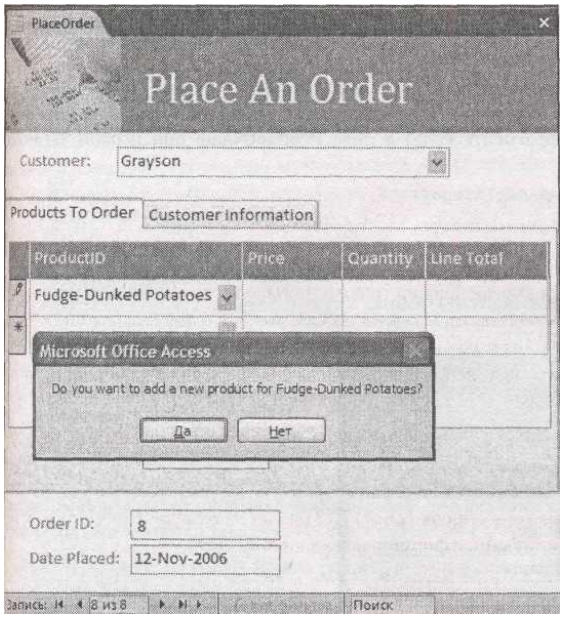


Рис. 17.12. Картофель в шоколадной глазури - в данный момент не предлагается в списке продуктов. Если его ввести и нажать клавишу <Enter>, программный код запросит подтверждение для добавления этого продукта в список

Далее предлагается код, добавляющий новый товар. В этом примере кода нет смысла самостоятельно включать товар полностью - в конце концов, для этого товара нужно предоставить дополнительную информацию (например, цену или категорию), прежде чем считать его допустимым. Вместо этого нужно отобразить другую форму для добавления товаров. Ключом решения может стать метод DoCmd. OpenForm:

```
' Попросите    Access не беспокоиться,   поскольку вы сами добавите
‘ пропущенный товар Response = acDataErrAdded
' Откройте форму AddProduct  с тремя дополнительными аргументами
DoCmd.OpenForm "AddProduct",  ,  ,  ,  ,  acDialog,  NewData
```

Два аргумента, используемые в методе OpenForm, особенно важны.

■ acDialog открывает форму в диалоговом режиме, т. е. программа Access задерживает вызов кода в ProductID_NotInList до тех пор, пока форма **AddProduct** закрыта. Этот шаг важен, поскольку после завершения процесса добавления вам понадобится выполнить дополнительный код для обновления формы **PlaceOrder.**

■ NewData принимает вновь введенные данные и присваивает их свойству AddProduct.OpenArgs. В этом случае форма **AddProduct** может извлечь их, когда запустится, и самостоятельно откорректировать.

Далее приведен программный код, который нужно вставить в форму **AddProduct** для копирования вновь введенного названия товара (значение, переданное с помощью переменной NewData в предыдущем фрагменте кода) в поле **ProductName** при первой загрузке формы

AddProduct.

```
Private Sub Form_0pen(Cancel As Integer)
ProductName = Form.OpenArgs
End Sub
```

На рис. 17.13 показано, как выглядит эта форма.

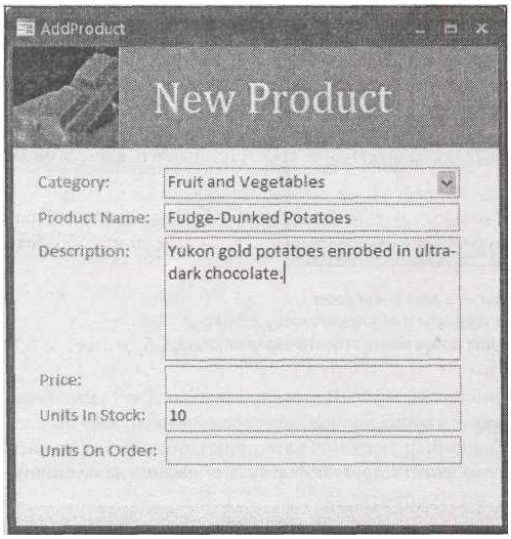


Рис. 17.13. Форма **AddProduct** позволяет ввести остальные данные для нового товара, который вы хотите создать. Обратите внимание на то, что форма открывается как всплывающая, и программа Access автоматически считает, что вы вставляете новую запись (а не просматриваете имеющиеся товары). Access действует таким образом, поскольку в свойствах формы **Всплывающее окно (Pop Up)** и **Ввод данных** (Data Entry) задано значение *Да*

После того как вся информация о товаре введена, можно закрыть форму **AddProduct.** В этот момент в процедуре ProductID_NotInList выполняется немного дополнительного кода. Он расположен сразу после оператора DoCmd. OpenForm. Его задача - обновить новый элемент заказа, чтобы использовать товар, который вы только что ввели:

```
' Отмена редактирования,  поскольку нужно  обновить  список
' прежде,  чем вы сможете выбрать новый товар ProductID.Undo
' Обновление списка  ProductlD.Requery
  ' Теперь  ищется ProductID для вновь  введенного  товара  с помощью DLookup
Product ID =  DLookup (“ID”,  "Products",  "ProductName='"  & NewData  &  "'")
```

Примечание

Этот код выполняется, даже если вы отменили вставку нового товара, нажав клавишу <Esc> в форме **AddProduct.** В этом случае функция DLookup не сможет ничего найти, поэтому вернет Null (пустое значение) в поле **ProductlD.** В результате вы получите знакомое предупреждающее сообщение программы Access, извещающее о том, что выбранного вами товара нет в списке.

Еще одно уточнение. Когда возникает событие **Отсутствие в списке** (On Not In List), событие **Изменение** (On Change) уже произошло. Таким образом, вы уже упустили возможность выполнить код, применявшийся ранее для вставки соответствующей цены в поле **Price** списка элементов заказа.

К счастью, эту проблему можно решить довольно легко. Нужно добавить еще одну строку кода, которая заставит программу Access двигаться дальше, и снова выполнить обработчик события (процедуру ProductID_Change): Product ID_Change

Для того чтобы увидеть полный программный код к этому примеру, обратитесь к БД Boutique Fudge в примерах к данной главе.

17.4.4. *Управление выполнением заказов*

Теперь, когда процесс размещения заказов отлажен, можно уделить внимание дальнейшим действиям.

В БД Boutique Fudge у каждой записи в таблице **Orders** есть поле **OrderStatus** (состояние заказа), отслеживающее его состояние или статус. У вновь созданных заказов статус New (новый). На складе сотрудники хранилища ищут заказы со статусом New (новый) и выбирают один из них для обработки. В этот момент они изменяют статус заказа на In Progress (выполняющийся в данный момент), поэтому никто больше в это же время не попытается его доставить. Наконец, когда заказ укомплектован, его статус меняется на Shipped (отправлен) и затем в поле **ShipDate** записывается точное время отправки.

Логически эта модель вполне осмыслена. Но немного трудно применять к ней обычные таблицы и формы. Для того чтобы следовать этому технологическому процессу, работники склада должны несколько раз изменять статус в записи заказа, помнить о необходимости зафиксировать дату отправки и при этом не изменять другие данные о заказе. Если они пропустят какой-нибудь этап - скажем, никогда не переведут статус заказа в In Progress (выполняющийся в данный момент) - вполне возможно, что кто-то из сотрудников попытается выполнить тот же самый заказ.

Решением может быть создание формы **ShipOrders,** которая проведет работников склада через все нужные этапы. Вначале эта форма отображает список заказов с минимальной информацией (рис. 17.14).

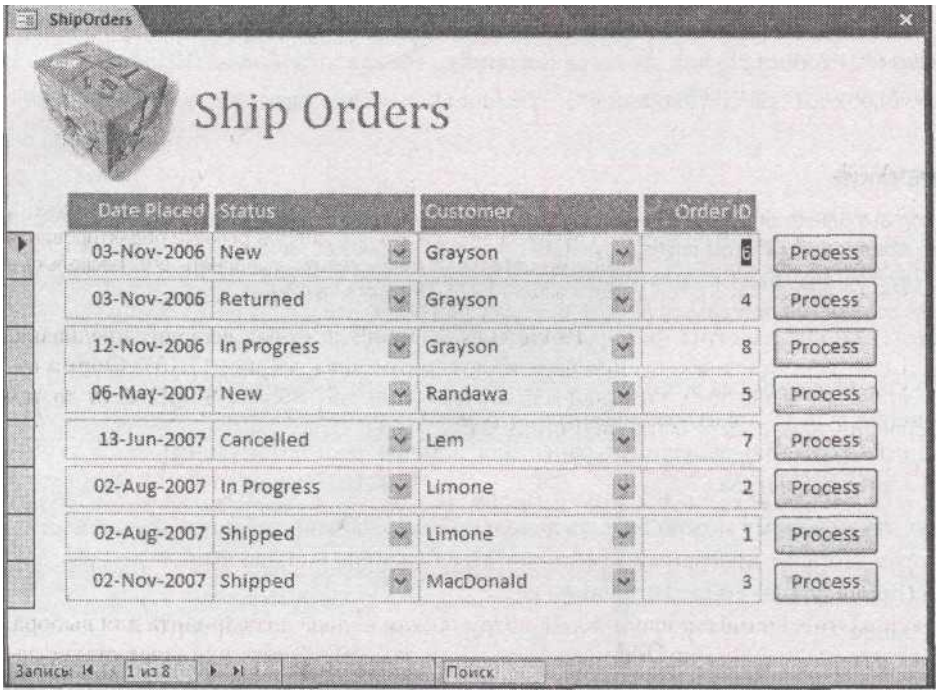


Рис. 17.14. Список заказов отсортирован, так что самые старые заказы (которые следует обработать первыми) появляются в верхней части списка. Свойство **Блокировка** (Locked) для всех полей задано со значением *Да,* поэтому никто не сможет изменить никакие данные. Рядом с каждым заказом расположена кнопка **Process** (обработка), которая начинает процесс выполнения заказа (в форму можно добавить фильтр, позволяющий отображать только заказы с определенными статусами)

Когда кто-либо щелкает мышью кнопку **Process** (обработка), должны выполняться несколько действий. Далее приводится последовательный разбор программного кода, поочередно, один фрагмент за другим.

Сначала ваш код должен обновить запись. Этот шаг помогает определить, не начал ли кто-то еще выполнять данный заказ на другом компьютере:

```
Private Sub ProcessOrder_Click()
Form.Refresh
```

Далее необходимо проверить статус записи. Если у нее статус не New, значит, она не годится

для обработки:

```
' StatusID для статуса New равен 2
If StatusID o 2 Then
MsgBox "This order is not available."
```

В противном случае нужно изменить статус на In Progress (выполняющийся в данный момент) и сразу сохранить запись, чтобы никто другой не пытался выполнить этот заказ:

```
Else
' StatusID для статуса In Progress равен 3
StatusID = 3
' Сохранение изменения
DoCmd.RunCommand acCmdSaveRecord
```

Примечание

В подобной ситуации крайне важно сохранить запись (с помощью метода DoCmd. RunCommand, как показано в примере). В противном случае запись заказа останется в режиме редактирования и новый статус не сохранится в БД. Другие работники могут начать выполнять его, поскольку у них нет возможности узнать о том, что вы изменили статус этого заказа.

Теперь самое время запустить форму **ReviewOrderDetails,** которая выводит предназначенное только для чтения представление всех компонентов заказа (рис. 17.15). Форма открывается в диалоговом режиме, который блокирует открытие формы **ShipOrders** до тех пор, пока не завершится процесс выполнения заказа:

```
DoCmd.OpenForm "ReviewOrderDetails", , , _
"OrderID =" & ID, , acDialog End If
End Function
```

Форма **ReviewOrderDetails** предоставляет сотрудникам склада два варианта для выбора. Если они щелкают мышью кнопку **Ship** (доставить), программа Access изменяет статус заказа на Shipped (отправлен) и процесс завершается.

```
Private Sub Ship_Click( ) ' Эта форма закрывается DoCmd.Close
' Обратный переход к форме ShipOrders DoCmd.OpenForm "ShipOrders"
' Обновление заказа
' StatusID для статуса Shipped равен 4 Forms ("ShipOrders").StatusID = 4
DoCmd.RunCommand acCmdSaveRecord
End Sub
```

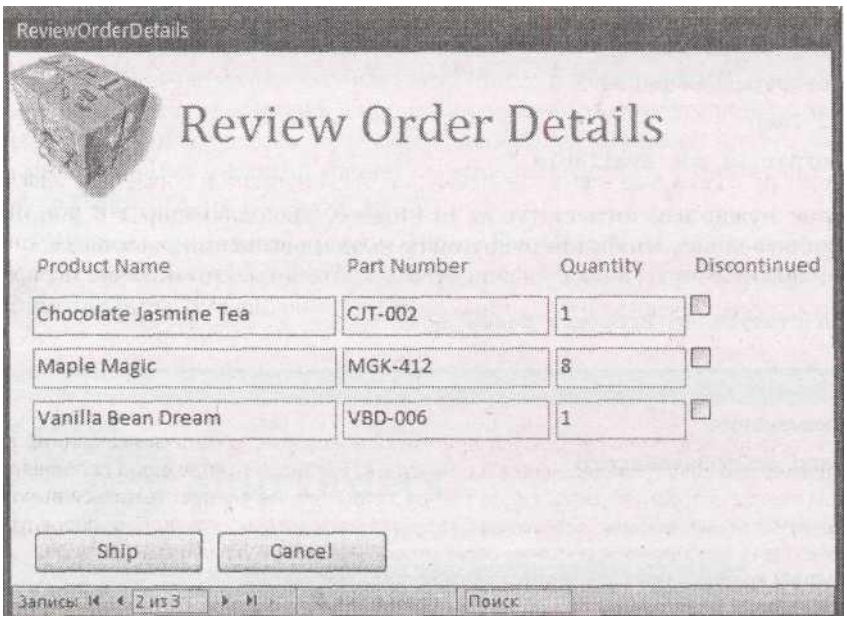


Рис. 17.15. В форму **ReviewOrderDetails** не нужно включать подробности, касающиеся цены товара. Она разработана просто для того, чтобы представить наиболее эффективным

способом работникам склада нужную им информацию. Форма **ReviewOrderDetails** применяет запрос с объединением для получения некоторых связанных данных, например, поля **PartNumber** из таблицы **Products**

В свойствах **Кнопка оконного** меню (Control Box) и **Кнопка закрытия** (Close Button) формы **ReviewOrderDetails** установлено значение *Нет.* В этом случае работники склада ни при каких обстоятельствах не могут закрыть это окно без щелчка мышью кнопки Ship (доставить) или **Cancel** (отмена). (Если не применять такой подход, придется писать дополнительный код, который переустанавливает статус заказа, когда кто-нибудь щелкнет мышью пиктограмму x в правом верхнем углу окна, чтобы закрыть форму ReviewOrder-Details.)

Подсказка

В этом месте кода удобно применить метод DoCmd. OpenReport для вывода на печать отчета, создающего транспортную наклейку со списком всех товаров, включенных в заказ.

Если же сотрудники щелкнут мышью кнопку **Cancel** (возможно, они выяснили, что на складе нет нужного количества товара), применяется аналогичный программный код для возврата заказу статуса New (новый):

```
Private Sub Cancel_Click()
' Закрытие этой формы DoCmd.Close
' Возврат к форме ShipOrder DoCmd.OpenForm "ShipOrders"
 ' Обновление заказа
Forms("ShipOrders").StatusID = 2
DoCmd.RunCoiranand acCmdSaveRecord End Sub
```

Эта часть завершает программный код, необходимый для координации обработки заказа. Как и формы, которые вы изучали в *части IV,* формы данного примера извлекают всю информацию из таблиц вашей БД. Но в отличие от примеров, приведенных в *части IV,* они используют код для автоматического выполнения некоторой работы. Это отличие превращает ваши формы из простых средств ввода данных в Супероснащенные средства автоматизации технологического процесса.

Подсказка

Можно создать специальное значение статуса для заказов, которые пытались выполнить, но не смогли (например, On Hold (незавершенный) или Waiting For Stock (ожидающий пополнения запаса)). В этом случае сотрудники склада будут знать о том, что не следует пытаться выполнять эти заказы. Если вы решили применить этот шаг, убедитесь, что код в процедуре ProcessOrder_Click исправлен, и можно обрабатывать заказы с указанным статусом.

17.4.5. *Обновление единиц наличного запаса*

Благодаря очень толковой форме **ShipOrders,** которую вы видели в предыдущем разделе, коммерческая деятельность компании Boutique Fudge протекает гладко. Но в один прекрасный день сотрудники склада приходят в главное управление с жалобой. Несмотря на то, что заказы проходят без отклонений, товарные запасы не поддерживаются в должном порядке. Никто не удосужился исправить данные в поле **UnitsInStock,** поэтому они становятся все более бесполезными.

По настоящему автоматизированное решение должно было бы автоматически обновлять данные **UnitsInStock,** когда заказ отправлен. В конце концов, не для этого ли прежде всего разработана программа Access?

Эта задача полностью отличается от тех, которые вы решали до сих пор, поскольку вынуждает вносить изменения в совершенно другой набор записей - записи, которые не отображаются ни на одной из форм.

Вы уже знаете, что можно применять статистические функции по подмножеству для из-

влечения информации из других таблиц. Но, к сожалению, у программы Access нет похожего набора функций, которые позволяют вносить изменения. Вместо этого придется обратиться к совершенно новому набору объектов, названных *объектами доступа к данным* (или для краткости просто DAO).

Технология DAO позволяет выполнять любую обработку данных, независимо от ваших форм. Но это довольно сложная структура.

* DAO следует использовать очень специфическим образом. Если методы применяются в неверной последовательности или пропущен шаг, вы столкнетесь с ошибкой. Часто легче всего начать с работающего примера (как, например программный код, включенный в примеры к данной главе), скопировать его и затем переделать, как нужно.

■ Технология DAO не применяет объекты запросов. Вместо этого следует писать SQL-операторы *(см. главу 6)*.

DAO включает два важных метода. Первый метод, CurrentDb. Execute, позволяет выполнять прямую SQL-команду, задавая ее в виде строки:

```
CurrentDb.Execute MyUpdateCommand
```

Этот метод - быстрый практический способ внесения в БД изменений, таких как операции очищающего обновления, удаления или вставки.

Второй важный метод предназначен для извлечения записей с помощью специализированного объекта Recordset. Для применения этого объекта вы должны начать с вызова метода CurrentDb. OpenRecordset и задать строку с SQL-командой выбора.

```
Dim Recordset
Set Recordset = CurrentDb.OpenRecordset(MySelectCommand)
```

Объект Recordset представляет группу записей, но доступ в каждый момент времени возможен только к одной из них. Для перехода от одной записи к другой применяется метод Recordset .MoveNext. Для проверки, достигнут ли конец набора, используется свойство Recordset .EOF, означающее конец файла (end-of-file). Если оно равно True, вы прошли последнюю запись.

Чаще всего объект Recordset применяется в цикле. Свойство Recordset .EOF можно использовать как условие цикла, так что цикл завершается, как только программа Access достигает конца набора Recordset. Внутри цикла можно извлекать значения полей текущей записи. В конце каждого прохода цикла следует вызывать метод MoveNext для перехода к следующей записи: Do While Recordset.EOF = False

```
'   Отображается значение поля  ProductName ■  MsgBox Recordset("ProductName")
'  Переход к следующей записи Recordset.MoveNext
Loop
```

Держа в голове эти непременные условия, можно создать следующий код, корректирующий величину запасов товаров, основываясь на только что отправленном заказе. (Построчный анализ приведен после кода.)

```
1 Sub UpdateStock()
' Если возникает ошибка, переход в секцию DataAccessError
2    On Error GoTo DataAccessError
' Создание команды SELECT
3       Dim Query
4       Query = _
"SELECT ProductID, Quantity FROM OrderDetails WHERE OrderID=" & ID
 ' Получение набора записей с помощью этой команды
5   Dim Recordset
6   Set Recordset = CurrentDb.OpenRecordset(Query)
' Перебор набора записей с просмотром каждой из них.
' Каждая запись - отдельный компонент заказа
7   Do Until Recordset.EOF
' Получение для каждого компонента ID товара и его количество
8   Dim ProductID, Quantity
9   ProductID = Recordset("ProductID"}
10 Quantity = Recordset("Quantity")
```

```
' Формирование команды UPDATE,
' которая изменяет уровни запасов
11 Dim UpdateCommand
12 UpdateCommand = _
"UPDATE Products SET UnitsInStock = UnitsInStock-" & _
13 Quantity & " WHERE ID=" & ProductID
' Выполнение команды
14 CurrentDb.Execute UpdateCommand
' Переход к следующему компоненту заказа (если таковой есть)
15 Recordset.MoveNext
16 Loop
' Самое время для очистки
17 Recordset.Close
18 CurrentDb.Close
19 Exit Sub
20 DataAccessError:
' Сюда вы попадаете только при возникновении ошибки.
' Отображение сообщения об ошибке
21 MsgBox Err.Description
22 End Sub
```

Вот что происходит в программном коде.

- В *строке 1* объявляется новая процедура. Поскольку этот код очень сложен, есть смысл поместить его в отдельную процедуру, которую можно вызывать, когда мышью щелкается кнопка **Ship** (отправить) и заказ отправляется клиенту.
- *Строка 2* заставляет программу Access перейти в конец процедуры, если возникает ошибка. В программном коде доступа к данным всегда возможны ошибки, поэтому лучше быть начеку.
- В *строках 3-4* создается SQL-команда, необходимая для выбора записей из таблицы **OrderDetails,** относящихся к текущему заказу. (Дополнительную информацию об SQL-командах SELECT см. в *разд. "Анализ запроса" главы 6.)*
- *Строки 5-6* выполняют эту команду и получают все соответствующие записи в объекте Recordset.
- В *строке* 7начинается цикл, обрабатывающий весь набор Recordset.
- *Строки 8-10* получают поля **ProductID** и **Quantity** для текущей записи таблицы **OrderDetails** (первой в объекте Recordset).
- *Строки 11-13* используют эту информацию для построения SQL-команды UPDATE. Команда вычитает количество заказанных товаров из общего числа хранящихся на складе товаров. Пример полной команды выглядит следующим образом: UPDATE Products SET UnitsInStock = UnitslnStock-4 WHERE ID= 14. Она вычитает 4 единицы из 14 единиц товара.
- *Строка 14* выполняет обновление.
- В *строках 15-16* выполняется переход к следующей записи и повторяется процесс обновления (до тех пор, пока в наборе Recordset не останется ни одного компонента заказа).
- В *строках 17-18* выполняется очистка.
- В *строке 19* - выход из процедуры. Если вы им воспользовались, мои поздравления - все прошло без сучка, без задоринки!
- *Строки 20-22* обрабатываются, только если где-то возникла ошибка. В этом случае описание ошибки выводится в окне сообщения.

Этот код гораздо более амбициозный, чем все, что вы видели до сих пор. Но он построен на приемах, которые вы совершенствовали в последних трех главах. Повторюсь еще раз - самый лучший способ освоения этого кода - загрузить пример БД, проверить его в действии и попытаться изменить его. Удачных экспериментов!

Уголок ностальгии.
DAO ПРОТИВ ADO

Для того чтобы усложнить жизнь, корпорация Microsoft давным-давно ввела еще одну

технологию доступа к данным, названную ADO (ActiveX Data Objects, объекты данных ActiveX). И DAO, и ADO выполняют задачи обработки данных с помощью удобных объектов.

Ключевая разница между ними заключается в том, что Microsoft проектировала ADO как универсальную технологию доступа к данным, которая может работать с другими системами управления БД, например SQL Server, а DAO предназначена только для Access.

Некоторые программисты Access считают (неправильно), что ADO - приемник DAO и лучший выбор при написании программного кода для БД Access. (В действительности корпорация Microsoft, возможно, даже заявляла об этом в определенный момент, но все свидетельства этого уничтожены.) В настоящий момент официальная версия заключается в том, что лучше применять DAO, поскольку эта технология настроена для программы Access. Это означает, что DAO легче использовать, и она обеспечивает лучшую производительность в большинстве случаев. ADO применяют лишь опытные VB-программисты, кто уже знает, как работает эта технология, и не хочет тратить время на изучение DAO, или же разработчики, которым требуется какое-нибудь экзотическое средство, имеющееся в ADO и отсутствующее в DAO.

Часть VI

Совместное использование Access

18. Глава 18. Совместное использование БД несколькими пользователями

Теперь, когда совершенная БД создана, возможно, вы захотите использовать ее совместно с друзьями и коллегами. В одних руках программа Access- первоклассное средство управле-ния данными. Но когда пристрастие разделяет группа людей, программа предоставляет от-личную возможность для совместной работы.

Совместное использование БД особенно важно, если она играет связующую роль в организации. Представьте себе, что вы создаете БД, которая отслеживает проекты компании и сроки их выполнения. (Обычно такая БД формируется как экономящее время средство в руках фаната Access, у которого есть небольшой избыток свободного времени.) Вскоре у других отделов возникает желание примкнуть к вам и сохранять свои проекты в вашей БД. И возможности этим не ограничиваются; если вы используете БД совместно с достаточно широкой аудиторией, можно собрать воедино все виды связанных задач. Сотрудники могут регистрировать время, потраченное на каждый проект. Тестировщики качества продукции могут фиксировать все нерешенные проблемы, влияющие на проект. Руководители проектов могут обозначить проекты с требующими напряжения сроками, главные начальники могут рассчитать премии, а генеральный директор оценить с высоты орлиного полета всю работу своей компании. Очень скоро будет трудно представить себе жизнь без вашей БД Access.

В этой главе вы научитесь использовать вашу БД вместе с небольшой группой, узнаете о возможных ловушках и о том, что нужно делать, чтобы все протекало гладко.

18.1. *Открытие вашей базы данных всему миру*

Когда вы решаете предоставить ваши данные для совместного использования, прежде всего, следует решить, могут ли другие пользователи изменять вашу информацию. Как вы увидите, распространять копии своей БД легко, а вот обеспечить работу многих пользователей с одним и тем же файлом БД в одно и то же время - более сложное дело.

В целом у вас есть четыре основных подхода для того, чтобы предоставить свои данные массам.

- *Экспортирование данных.* Можно взять данные из вашей БД и экспортировать их в дру-гой формат (например, в Web-страницу на языке HTML или электронную таблицу Excel). Этим путем можно пойти, если у пользователей, которым нужны ваши данные, нет программы Access. Как экспортировать данные, вы узнаете в *главе 19.*
- *Копирование вашей БД.* Можно предоставить другим пользователям копию вашей БД. Например, если у вокальной группы "Uncle Earl" есть копия Access, можно послать им по электронной почте список семейных адресов. Ограничение такого подхода состоит в отсутствии легкого способа синхронизации изменений в разных копиях. Если группа "Uncle Earl" добавит несколько новых людей в список, ваш исходный список останется прежним. Если вы измените исходный список, устареет копия группы "Uncle Earl".
- *Переход на серверное программное обеспечение.* Можно перенести ваши данные на серверный программный продукт промышленного уровня, например SQL Server или SharePoint. После этого практически неограниченное число пользователей сможет получать данные. Каждый из них использует собственную копию программы Access для соединения с сервером, у которого есть централизованное хранилище информации. Очевидный недостаток такого подхода - сложность, поскольку настройка подобных программных продуктов - непростая задача даже для технически образованных пользователей. Вы опробуете два лучших варианта подобного подхода в *главе 20* (SQL Server) и в *главе 21* (SharePoint).
- *Применение средств коллективного использования программы Access.* Можно поместить вашу БД в папку с общим доступом (например, на сетевой диск), чтобы несколько человек могли использовать ее одновременно. В этом случае все работают с одним и тем же множеством данных (и изменения, сделанные группой "Uncle Earl", не пропадут). При таком подходе программа Access должна координировать работу пользователей. Если у вас небольшая группа - скажем, не более 40 человек используют вашу БД одновременно - коллективное

использование будет функционировать. Но если группа достаточно велика, программа Access для нее не подойдет. В этом случае нужен серверный программный продукт, который рассчитан на большие объемы и многопользовательский доступ. (В следующем разделе приведен удобный перечень, который поможет решить, годится ли вам такой подход.)

Данная глава целиком посвящена последнему, приведенному в списке подходу - много-пользовательской поддержке, жестко запрограммированной в Access. Но перед тем как начать, важно выяснить ограничения, с которыми вы столкнетесь, чтобы оценить, удовлетворяют ли вашим нуждам средства, предлагаемые программой Access.

18.1.1. *Как действует многопользовательская поддержка в Access*

Разобраться в средствах коллективного использования БД программы Access легко. Сначала нужно поместить файл с вашей БД в такое место, к которому у всех есть доступ - например, в папку с общим доступом на вашем компьютере или (что еще лучше) в папку на серверном компьютере в сети вашей компании. Теперь всем, кто хочет использовать БД, достаточно открыть ее файл.

Звучит, просто, не так ли? Но не торопитесь. Прежде чем реорганизовывать работу целой компании, ориентируя ее на единственный файл БД, следует учесть несколько чисел. Далее приведены некоторые индикаторы, свидетельствующие о том, что многопользовательская поддержка Access вам подойдет.

■ *Не более 40 человек одновременно пользуются БД.* Количество пользователей, одновременно обращающихся к БД, - это ключевой момент. Вы можете использовать одну и туже БД совместно с сотнями пользователей, при условии, что все они не будут открыватьэту БД одновременно.

Примечание

Это число (40) - осмысленная рекомендация, а не железное правило. Некоторые гуру Access проектировали БД, которые могли выдерживать от 90 до 100 одновременных пользователей. Но без серьезных (и сложных) оптимизационных мер вы очень скоро упретесь в кирпичную стену.

■ *Не более 15 человек одновременно изменяют БД.* Читать БД легко, а обновление БД со-пряжено с некоторыми серьезными проблемами. Очевидная проблема возникает, когда несколько человек разными способами пытаются изменить одну и ту же запись одновременно. Из-за проектных особенностей программы Access даже те изменения, которые не приходят в столкновение друг с другом, могут снизить общую производительность. Вы попытаетесь преодолеть это затруднение позже в данной главе, когда будут обсуждаться блокировки *(см. разд. "Применение блокировок для предотвращения наложения обновлений "далее в этой главе).*

Примечание

Это число (15) - рекомендация с перестраховкой. Если разные пользователи вносят изменения в совершенно разные таблицы, возможно, вам удастся протолкнуть и больше одновременных корректировок. С другой стороны, если все захотят изменить несколько одних и тех же записей, вы можете попасть в аварийную ситуацию и раньше. Если сомневаетесь, попробуйте.

■ *Структура БД меняется нечасто.* Другими словами, вы не планируете регулярно изменять структуру таблиц, добавлять новые поля или корректировать связи между таблицами. В идеале следует окончательно откорректировать свои таблицы, а затем предлагать БД для совместного использования. Для получения наилучших результатов только одному человеку следует поручить роль *главного разработчика таблиц,* который в случае необходимости отвечает за изменение структуры БД.

■ *Разные пользователи намерены работать с разными таблицами.* Если все, кто пользуется вашей БД, выполняют одну и ту же задачу (и обращаются к одной и той же таблице), у вас будут проблемы. Но если один пользователь поддерживает каталог товаров, еще пять вводят заказы, а шестой регистрирует их отправку, вы в гораздо лучшей ситуации. Несмотря на то,

что все пользуются одной и той же БД, их работа не перекрывается.

■ *Ваша БД не является критически важной.* Данные всегда важны. Но если вы обеспечиваете работу компании, занимающейся электронной коммерцией, с помощью Web-сайта, функционирующего 24 часа в сутки, вам не избежать хотя бы кратковременных затруднений. К сожалению, программа Access не может гарантировать такого рода стабильности. Хотя подобные ситуации случаются редко, внезапные сетевые проблемы или сбой компьютера в момент, когда пользователь вносит изменения, могут с большой долей вероятности повредить вашу БД.

Примечание

Всем почитателям программы Access следует выполнять регулярные копирования своих БД в течение дня. Для автоматизации этого процесса можно применять средства планирования (например, Планировщик заданий ОС Windows).

Не будет преувеличением сказать, что многопользовательские БД Access - основное решение для многих маленьких компаний. Но если вы ознакомились с ограничениями программы и решили, что Access не сможет удовлетворить ваши потребности, самое время перейти на более мощный программный продукт, например, SQL Server. (Не паникуйте - доступна бесплатная версия, и вы можете управлять вашими данными в привычном интерфейсе Access.) Вы узнаете об SQL Server в *главе 20 и о* SharePoint Server в *главе 21.*

С другой стороны, если программа Access соответствует вашим потребностям, поздравляю - вы всего в одном шаге от преобразования вашей одинокой однопользовательской БД в ресурс, которым сможет пользоваться вся ваша компания. Просто читайте дальше.

18.2. Подготовка вашей базы данных

Если вы добрались до этих строк, значит, вы решили, что многопользовательские средства программы Access - все, что вам нужно. Но прежде чем ваша БД станет общедоступной, возможно, потребуется несколько изменений. Наиболее важное из них - разделение БД -часто пропускаемый, несмотря на важность, этап, придающий вашей многопользовательской БД дополнительную надежность.

Примечание

Когда вы предоставляете свои данные для коллективного использования, очень важно применять разделенную БД. Совместное использование обычной БД может привести к самым неожиданным поворотам и случайностям, которые сделают вашу БД ненадежной.

18.2.1. Что такое разделенная БД

Разделенная база данных - это БД, объекты которой разделены на два отдельных файла:

■ *внутренняя* или *серверная* БД (back-end database) содержит исходные данные - иначе говоря, таблицы и ничего больше;

■ *внешняя* или *клиентская* БД (front-end database) содержит все, чем вы пользуетесь при работе с таблицами, а именно все объекты БД других типов, например, запросы, отчеты, формы и макросы.

После разделения базы данных внутренняя БД помещается в общедоступное место (например, на сетевой диск). Внешняя БД функционирует несколько иначе. Ее копируют на все компьютеры, которые собираются использовать внутреннюю БД. На рис. 18.1 показан этот принцип работы.

Разделенные БД обладают рядом достоинств.

■ *Производительность.* Когда применяется разделенная БД, у каждого клиента есть копии объектов, необходимых ему для работы, - например, формы - готовые и ждущие на его компьютере. Это означает, что пользователю не нужно извлекать эту информацию из совместно используемой БД, что потребовало бы больше времени (и создало бы дополнительный трафик в вашей сети). В этом случае из совместно используемой БД вам необходимо получить только нужные для работы данные.

■ *Более легкое обновление.* Изменение данных в многопользовательской БД относительно

безопасно для людей, но программа Access плохо справляется с неразберихой, возникающей при попытках пользователей изменить структуру объектов БД. Разделенная БД I лишена этой проблемы, поскольку объекты, которые часто нуждаются в корректировках, например запросы, отчеты и формы, находятся в клиентской БД. Если эти объекты нужно модифицировать (или добавить новые), можно безопасно изменить клиентскую БД на одном компьютере, а затем распространить ее между всеми, кому она понадобится.

Такой подход не только легче, но и гораздо надежнее, т. к. не может возникнуть неразрешимая проблема, именуемая *разрушением* или *повреждением* БД *(см. разд. "Повреждение данных" далее в этой главе).*

Рис. 18.1. В системе с разделенной БД у вас одна внутренняя или серверная БД, централизованно хранящая данные, и несколько внешних или клиентских БД для каждого пользователя, подключающегося к вашей БД

- *У разных людей разные клиентские БД.* Когда применяется БД коллективного пользования, можно создать разные клиентские БД для разных типов пользователей - например, отделу маркетинга необходимы отчеты, отображающие сведения о продажах, а сотрудникам склада нужна форма, которая выводит на экран невыполненные заказы. Этим подходом можно воспользоваться для того, чтобы наверняка не показывать пользователям формы, отчеты и таблицы, не предназначенные для них, что снизит риск мелких ошибок (например, генеральный директор случайно не уничтожит весь каталог товаров). Но не впадайте в крайность - чем больше создается вариантов клиентских БД, тем больше усилий понадобится для их сопровождения.

Примечание

С технической точки зрения вы не сделаете вашу БД более защищенной, если предоставите пользователям клиентские БД с ограниченными возможностями. В конце концов, технически грамотные пользователи Access могут просто создать собственные клиентские БД и использовать их для получения неограниченного доступа к серверной БД. Но хотя клиентская БД не может остановить злоумышленника, она сможет защитить от небрежного или чересчур любопытного пользователя.

У вас есть два способа разделения БД. Можно воспользоваться мастером или сделать это вручную с помощью средств экспорта и импорта программы Access. В следующих разделах описаны оба метода.

Для тех, кто понимает.
Поиск места в сети для вашей БД

Прежде чем разделять БД, нужно найти место для файла внутренней или серверной части БД.

Один из вариантов (довольно рискованный) - предоставить доступ к ней прямо на вашем компьютере. Нужно только перетащить мышью вашу БД в папку Общие документы или создать новую папку с общим доступом. Процесс может немного отличаться в зависимости от вашей версии ОС Windows и настроек вашего ПК, но приведенные инструкции подходят для большинства компьютеров с ОС Windows XP.

1. Запустите Проводник.
2. Найдите место хранения вашей БД, щелкните эту папку правой кнопкой мыши и выберите в меню строку **Свойства** (Properties).
3. Перейдите на вкладку **Доступ** (Sharing) и затем выберите переключатель **Открыть общий доступ к этой папке** (Share this Folder).
4. При желании задайте альтернативное имя для папки и максимальное количество пользователей, которые смогут получить доступ к папке одновременно.
5. Щелкните мышью кнопку ОК.

Теперь другие пользователи сметут получить доступ к вашему компьютеру - и вашей БД - с помощью обзора содержимого **Сетевого окружения.**

Возможно, этот процесс показался вам слишком легким, для того чтобы быть правильным, но он таков и есть. Проблема заключается в том, что ваш компьютер - не идеальный сервер. Если вы выключите его, отправляясь в отпуск, все будут резко отлучены от вашей БД. Подобным образом, когда вы заняты игрой "Revenge of the Demon Spawn Legion Part IV", а другие пользователи в это время пытаются работать с вашей БД, производительность их работы (и ваша игровая практика) страдает. Но гораздо более серьезная проблема возникнет, если вы перезагрузите компьютер, т. к. все, пользующиеся в данный момент вашей БД, будут внезапно отсоединены от нее. Потеря чьей-то работы гарантирована, а возможна и порча данных БД.

Из-за названных причин настоятельно рекомендуется помещать вашу БД на серверный компьютер. Серверный компьютер не обязательно отличается от вашего - в действительности он может быть обычным ПК под управлением ОС Windows, включенным в сеть. Отличие заключается в том, что никто не использует этот компьютер непосредственно. Он остается один и может сосредоточиться на важной работе, раздавая ограниченными порциями данные тем, кому они нужны.

18.2.2. *Разделение БД с помощью мастера*

Самый легкий путь разделения БД - применение удобного мастера, который именно для этой цели включен в программу Access. Мастер создает новую серверную БД и переносит в нее все таблицы из текущей БД. В текущем файле остаются все остальные объекты, и таким образом он становится клиентской БД.

Вот как следует применять мастер.

1. Откройте любую БД, в которой есть таблицы и другие объекты (например, запросы, отчеты или формы).

Эти действия можно проделать с БД Boutique Fudge и Cacophone Studios, которые применялись в предыдущих главах.

2. Прежде чем начать, хорошо бы сделать резервную копию БД.

Никогда не знаешь, когда что-то пойдет не так. Для создания резервной копии просто скопируйте файл БД Access в другую папку (или выберите кнопку **Office → Управление → Резервная копия базы данных** (Office → Manage → Back Up Database)), чтобы заставить программу Access сделать ее для вас.

3. Выберите на ленте **Работа с базами данных → Переместить данные → База данных Access** (Database Tools → Move Data → Access Database).

На экране появится первое окно мастера разделения БД (Database Splitter) (рис. 18.2).

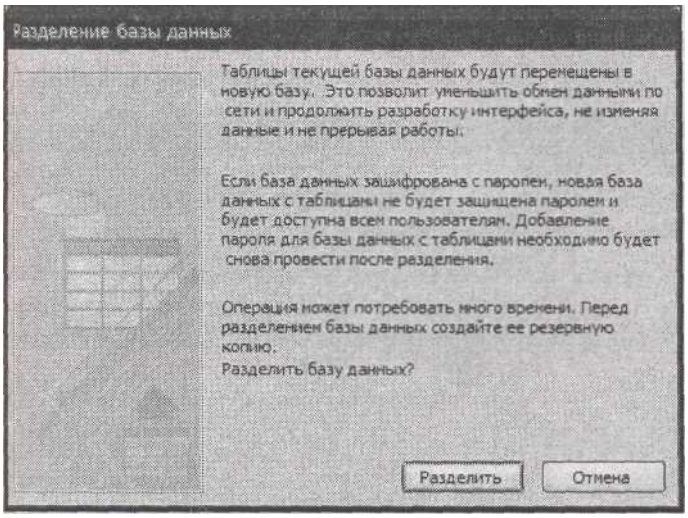


Рис. 18.2. Первый шаг мастера чрезвычайно прост. Описывается принцип работы мастера и делается напоминание о необходимости создания резервной копии перед тем, как двигаться дальше

4. Щелкните мышью кнопку **Разделить** (Split Database).

На экране появится окно, предлагающее выбрать местонахождение и имя файла для серверной БД.

Помните о том, что нужно выбрать место, доступное всем сотрудникам вашей компании или организации. (См. некоторые советы в *примечании "На профессиональном уровне. Указание местонахождения в сети" далее в этом разделе.)*

Примечание

Вы можете сохранить серверную БД на вашем компьютере в данный момент и переместить ее в общедоступное место позже (но в тот момент потребуется обновить ссылки на таблицы, как описывается в следующем разделе).

5. Выберите имя файла для серверной БД и щелкните мышью кнопку **Разделение** (Split) (рис. 18.3). Программа Access начнет экспортировать таблицы - другими словами, копировать их из текущей БД в новый файл серверной БД. Этот процесс может занять какое-то время.

Когда программа завершит экспорт, она выведет на экран сообщение: "База данных успешно разделена" ("Database successfully split"). Access успешно создала серверную БД. БД, с которой вы начинали (и которая открыта в настоящий момент), теперь клиентская. Она больше не содержит таблиц со всеми данными; в ней остался набор связанных таблиц, которые позволяют извлекать данные из серверной БД. (Вы узнаете, как работают связанные таблицы, в следующем разделе.)

6. Теперь пора распространить клиентскую БД среди всех тех, кому нужно ее использовать.

Способ распространения клиентской БД целиком зависит от вас. Ее можно отправить по электронной почте, записать на CD и затем отнести в руках или поместить ее в расположение с общим доступом. Важно, чтобы все пользователи понимали, что перед использованием клиентской БД они должны скопировать БД на свои компьютеры.

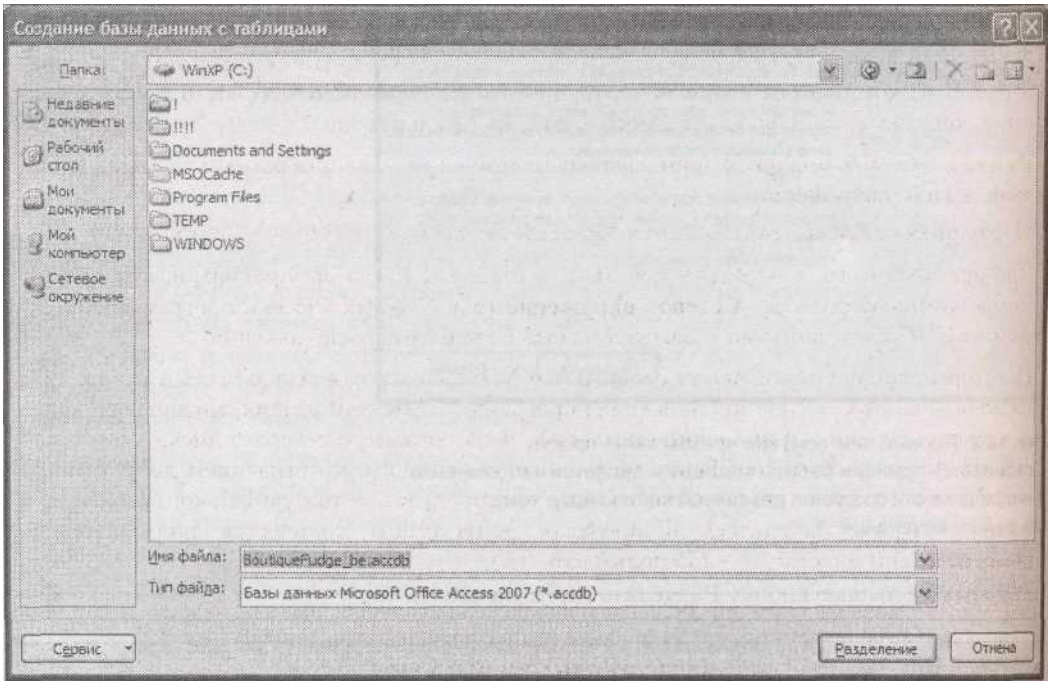


Рис. 18.3. Access рекомендует добавлять комбинацию символов "_be" в конец имени файла вашей БД для обозначения серверной части. Например, файл JoesTaxidermy.accdb превращается в файл JoesTaxidermy_be.accdb

Примечание

Если вы распространяете клиентскую БД, помещая ее в сети, вы рискуете, т. к. пользователи могут запустить ее прямо из сети без предварительного копирования на свои компьютеры. Если это не проверить, возникнут все проблемы, свойственные обычной (неразделенной) БД, такие как снижение производительности и увеличение риска ошибок.

На профессиональном уровне

Указание местонахождения в сети

Есть две возможности указать местонахождение в сети. Во-первых, использовать подключенный сетевой диск, который присваивает месту в сети букву дискового устройства вашего компьютера. Подключенные сетевые диски выглядят так же, как обычные диски - например, у вас может быть диск C:, представляющий жесткий диск на вашем компьютере, диск D:, представляющий привод CD-ROM, и диск F:, который представляет место в сети.

К сожалению, подключаемые сетевые диски могут по-разному конфигурироваться на различных компьютерах. Например, диск, обозначенный F:, на каком-то другом компьютере может стать диском H:. В результате клиентская БД, работающая на одном компьютере, при переносе на другой компьютер не сможет найти нужную ей серверную БД. К счастью, эту проблему легко решить. Нужно просто связать клиентскую БД с соответствующим местонахождением серверной БД, как описывается в следующем разделе.

Если хотите целиком избавиться от подобной путаницы, можно использовать UNC-путь (Universal Naming Convention, соглашение об универсальном назначении имен) вместо подключаемого сетевого диска. UNC - стандарт создания пути, указывающего на местонахождение в сетевом окружении. Преимущество UNC-пути состоит в том, что он не меняется от компьютера к компьютеру. Другими словами, если UNC-путь действует на одном компьютере, он будет действовать и на другом, и в сети.

Узнать UNC-путь можно по двум начальным символам - двум обратным слэшам. Далее приведена базовая форма:

\\ИмяКомпьютераВСети\ИмяПапкиСОбщимДоступом

Пример UNC-пути - \\SalesComputer\Database. Когда вы просматриваете содержимое компьютера через **Сетевое окружение** (My Network Places), программа Access создает UNC-путь, который указывает на выбранное вами расположение.

Некоторые специалисты Access сообщают о более надежных результатах в случае использования UNC-путей при поиске БД программой Access. В редких, трудно воспроизводимых ситуациях применение синтаксиса подключаемого сетевого диска может заставить Access выдать сообщение об ошибке, связанной с превышением допустимого числа пользователей, при попытке открыть многопользовательскую БД, когда точно известно, что вы находитесь в пределах допустимого количества пользователей (теоретический максимум - 255 пользователей).

Часто задаваемый вопрос.
Как поведут себя старые версии Access?
Что произойдет, если некоторые сотрудники будут использовать более старые версии программы Access?

В идеальном мире у всех есть копия самой последней и самой замечательной версии Access - Access 2007. В действительности наверняка найдутся слабо технически развитые ренегаты, до сих пор предпочитающие ОС Windows 95.

Если вы поддерживаете любителей версии Access 2003, нужно сохранить серверную БД в формате Access 2003. (Дополнительную информацию о сохранении копии вашей БД в другом формате см. *примечание "Для тех, кто понимает. Использование Access БД, созданных в более ранних версиях программы" в разд. "Создание новой базы данных" главы 1.)* Что касается клиентской БД, возможно, придется хранить две версии - одну для членов клуба поклонников Access 2007, а другую для семейства любителей Access 2003. Вы лишитесь некоторых средств в формате Access 2003, но вам не придется ослабленных технологически пользователей отлучать от вашей БД.

18.2.3. *Как действуют связанные таблицы*

Идея разделенной БД кажется достаточно понятной. Один файл (серверная БД) хранит исходные данные, в то время как другой (клиентская БД) предоставляет средства для работы с ними. Но есть одна деталь, которую вы пока не учли - а именно, способ получения доступа клиентской БД к таблицам серверной БД. Секрет кроется в *связывании таблиц* (table linking).

Связывание позволяет одной БД видеть таблицу в другом файле БД. Связывание может использоваться в любой БД - на самом деле вы можете использовать его, даже если не собираетесь предоставлять коллективный доступ к БД. Например, можно разделить ваши таблицы на два или несколько файлов БД для преодоления ограничения, касающегося размера файла (которое равно 2 Гбайт или гигабайтам на каждый файл БД). Или можно воспользоваться связыванием для упорядочивания разросшейся БД с десятками таблиц. Наконец, вы можете решить, что оно поможет отделить общедоступную информацию от сверхсекретных подробностей. Если поместить таблицы с секретными данными в отдельную БД, вы сможете спокойно предоставить для коллективного пользования копии основной БД, не беспокоясь о том, что легко уязвимые данные попадут в плохие руки.

Благодаря связыванию все таблицы серверной БД оказываются и в клиентской БД (рис. 18.4), но реальные данные находятся в отдельном файле. Когда вы открываете таблицу или вместо этого связываетесь с ней, программа Access обращается к связанной таблице для получения нужной вам информации.

Единственный недостаток связанных таблиц - вероятность того, что при поиске связанной таблицы программа Access не сможет найти другой файл БД. Такое случается, когда файл серверной БД переносится в другую папку или переименовывается.

Примечание
Если у вашей БД обнаруживается недействительная связь при попытке открыть таблицу (или другой объект, использующий таблицу, например, запрос или отчет), вы получите сообщение, информирующее о том, что программа Access не может найти нужный вам файл.

К счастью, связи легко обновляются. Выполните следующие действия.

1. Выберите на ленте **Работа с базами данных → Работа с базами данных → Диспетчер связанных таблиц** (Database Tools → Database Tools → Linked Table Manager) (или щелкните

правой кнопкой мыши любую связанную таблицу и затем выберите **Диспетчер связанных таблиц).**

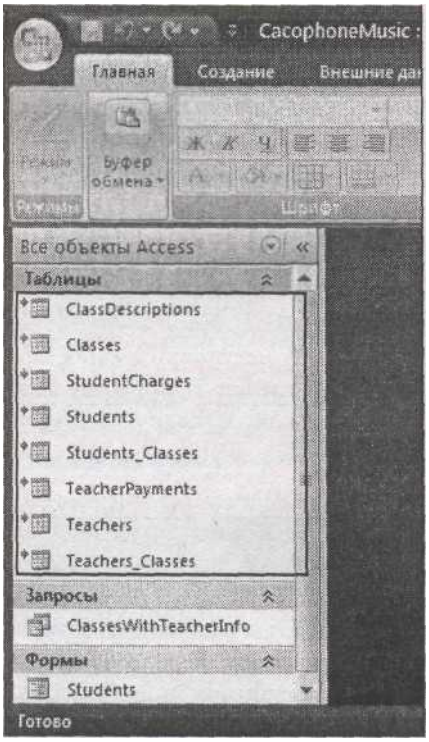
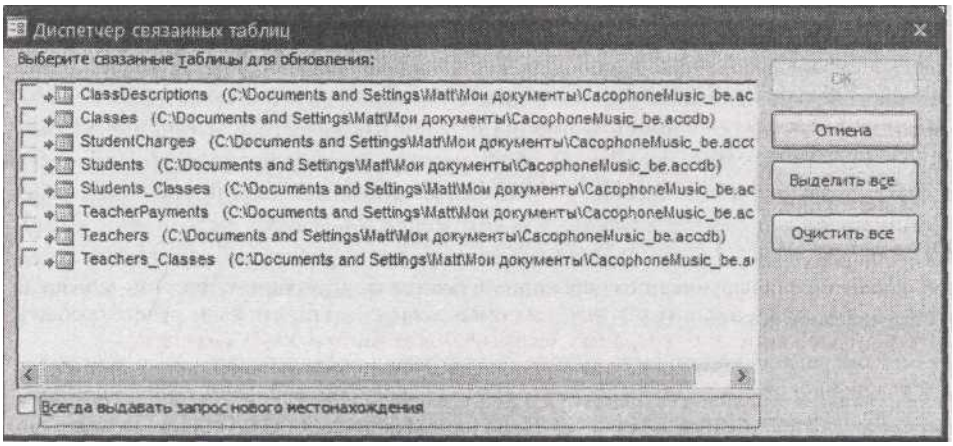


Рис. 18.4. Рядом с этими таблицами расположена пиктограмма со стрелкой, обозначающая связанные таблицы. На самом деле эти таблицы хранятся не в текущем файле БД, но программа Access знает, где найти данные, когда они потребуются

На экране появится окно Диспетчера связанных таблиц со списком всех связанных таблиц вашей БД (рис. 18.5).

Рис. 18.5. В этой БД восемь связанных таблиц. Все ссылки указывают на одну и ту же серверную БД, общую для всех, что не является обязательным условием



2. Установите флажок рядом с каждой ссылкой, которую хотите изменить.

Если нужно обновить все ссылки, щелкните мышью кнопку **Выделить все** (Select All).

Чаще всего все ваши ссылки будут указывать на один и тот же файл БД. Но если нужно связать таблицы с разными файлами, установите флажок **Всегда выдавать запрос нового местонахождения** (Always prompt for new location).

3. Щелкните мышью кнопку ОК.

Программа Access выведет на экран знакомое окно для выбора файла. Просмотрите папки для поиска файла БД, содержащего вашу таблицу, выделите его и щелкните мышью кнопку ОК.

Если вы установили флажок **Всегда выдавать запрос нового местонахождения,** программа

Access будет выводить окно выбора файла для каждой ссылки. Посмотрите на заголовок окна, чтобы выяснить, связь с какой таблицей вы обновляете. Если данный флажок не установлен, можно обновить все связи за один шаг.

Аварийная ситуация.
Мистическая ошибка “Файл уже используется”

Теперь, когда БД с многопользовательской поддержкой создана, вы рассчитываете, что она будет поддерживать толпу пользователей, поэтому вас приводит почти в шоковое состояние таинственная ошибка "Файл уже используется" ("File already in use"). Разве не для того создаются БД с многопользовательской поддержкой, чтобы оставаться доступными, даже когда кто-нибудь их использует?

Эта ошибка возникает, поскольку кто-то уже открыл БД с монопольным доступом (Exclusive mode). Монопольный доступ *(см. разд. "Открытие БД с монопольным доступом " далее в этой главе)* позволяет одному пользователю связаться с БД и заблокировать попытки всех остальных. Хитрость заключается в том, что при определенных обстоятельствах программа Access может применять монопольный доступ, даже если вы ее об этом не просили.

Наиболее распространенная проблема - отсутствие у пользователя подходящих разрешений на папку с общим доступом, в которой хранится многопользовательская БД. *(Разрешения* (Permissions) - составляющая системы безопасности ОС Windows, определяющая способ использования файлов и папок тем или иным пользователем.) В особенности эта проблема проявляется, когда вы первым открываете БД и не имеете разрешения на создание новых файлов. В этой ситуации программа Access не может создать файл с расширением laccdb. Этот файл отслеживает блокировки *(см. разд. "Применение блокировок для предотвращения наложения обновлений" далее в этой главе).* Без файла с расширением laccdb Access не может координировать работу многих пользователей. Поэтому программа тихо переходит в режим монопольного доступа, который блокирует остальные обращения к БД.

Понятно, что решением может быть точное определение всех, нуждающихся в использовании БД, и гарантированное предоставление им разрешения на создание новых файлов в папке с общим доступом. Конечно, все было бы гораздо проще, если бы программа Access могла предупредить вас о том, что не может нормально открыть БД и вынуждена применить монопольный доступ.

18.2.4. *Разделение БД вручную*

Для разделения БД не обязательно применять мастер. Вы можете самостоятельно перенести таблицы в отдельный файл БД и затем вручную установить с ними связь. Главная причина такого подхода заключается в желании разделить вашу БД на несколько фрагментов - например, вы хотите создать одну клиентскую часть и четыре серверных файла.

Для разделения серверной БД существуют некоторые веские основания. К ним относятся следующие.

- *Повышение надежности.* На самом деле, если один файл будет поврежден, остальные останутся в прежнем состоянии.
- *Повышение уровня безопасности. С* помощью средств ОС Windows вы можете управлять разрешениями на открытие конкретных файлов. Благодаря этим средствам можно использовать преимущества разделения БД для блокирования доступа пользователей к тем частям серверной БД, с которыми они не связаны.
- *Возможность дальнейшего роста файла БД.* Как упоминалось ранее, программа Access ограничивает размер БД 2 Гбайт. Если планируется хранить большое число записей с вложениями (например, изображениями), хорошо бы иметь уверенность в том, что дос таточно дискового пространства доступно в данный момент и будет доступно в обозри мом будущем.

Для разделения БД вручную необходимо использовать средства импорта и экспорта программы Access. Описанные далее действия продемонстрируют разделение БД Boutique Fudge на три отдельных файла, таким образом, вы сможете хранить данные кредитных карт отдельно от остальной информации. (Если хотите повторять эти действия на компьютере, найдите БД примеров на странице "Missing CD" Web-сайта **www.missingmanuals.com.)**

1. Создайте необходимые серверные БД.

В данном примере вам нужны две серверные БД: одна для хранения данных кредитных карт (назовем ее BoutiqueFudgeSecrets_be.accdb) и вторая для остальных подробностей (назовем ее BoutiqueFudge_be.accdb). Прежде чем двигаться дальше, создайте обе эти БД в программе Access и поместите их в общедоступную папку, но пока оставьте пустыми.

2. Откройте файл серверной БД.

Следующая задача - добавить соответствующие таблицы в каждую из серверных БД. Для этого используйте мастер импорта программы Access.

Начнем с файла BoutiqueFudgeSecretsJbe.accdb. Он легче, поскольку должен включать всего одну таблицу.

3. Выберите на ленте **Внешние данные → Импорт → Access** (External Data → Import → Access).

Начнет работу мастер импорта (рис. 18.6).

Примечание

В данном примере импортируются таблицы, необходимые для серверной БД. Можно попробовать и обратный прием - экспортировать таблицы из клиентской БД. Но у операций экспорта больше ограничений, чем у импорта, т. к. они позволяют преобразовать только одну таблицу одновременно.

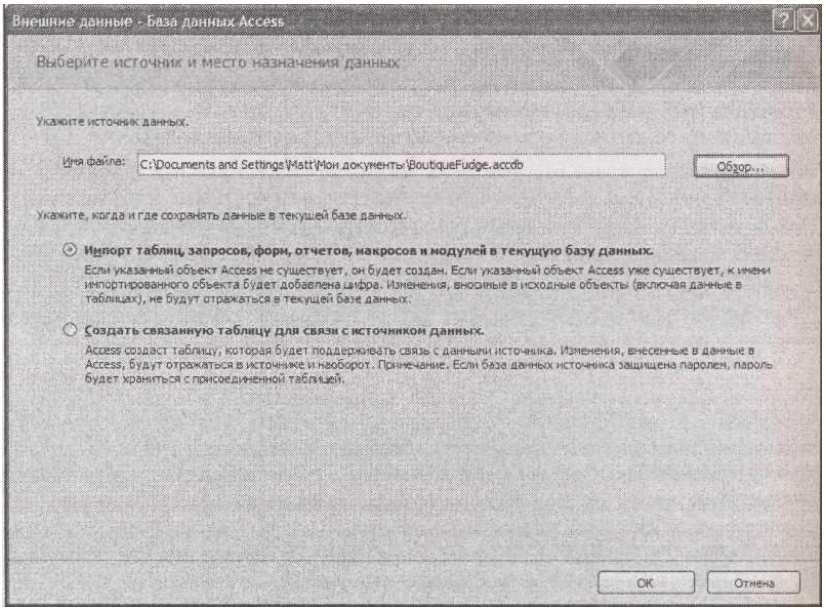


Рис. 18.6. В первом окне мастера импорта выбирается файл с таблицами, которые надо импортировать, и затем решается, копировать таблицы или просто создать ссылки на них

4. В поле **Имя файла** (File name) задайте местоположение вашей клиентской БД.

В данном примере - это файл BoutiqueFudge.accdb, в настоящий момент содержащий полный набор (таблицы, запросы, формы и отчеты).

5. Выберите первый переключатель **Импорт таблиц ... в текущую базу данных** (Import tables ... into the current database).

Второй переключатель позволяет создать связанные таблицы. Вы примените их позже в этом процессе.

6. Щелкните мышью кнопку ОК.

На экране появится окно **Импорт объектов** (Import Objects) со всем содержимым вашей БД (рис. 18.7).

6. Выделите таблицы, которые хотите импортировать, и щелкните мышью кнопку **ОК**.

БД BoutiqueFudgeSecrets_be нужна единственная таблица **CreditCards** (кредитные карты).

После нажатия кнопки **ОК** программа Access копирует таблицы в вашу БД. Файл БД можно закрыть.

8. Повторите пункты 2-7 для заполнения остальных файлов серверных БД.

В данном примере необходимо открыть файл BoutiqueFudge_be.accdb и импортирован в него все таблицы за исключением таблицы **CreditCards.**

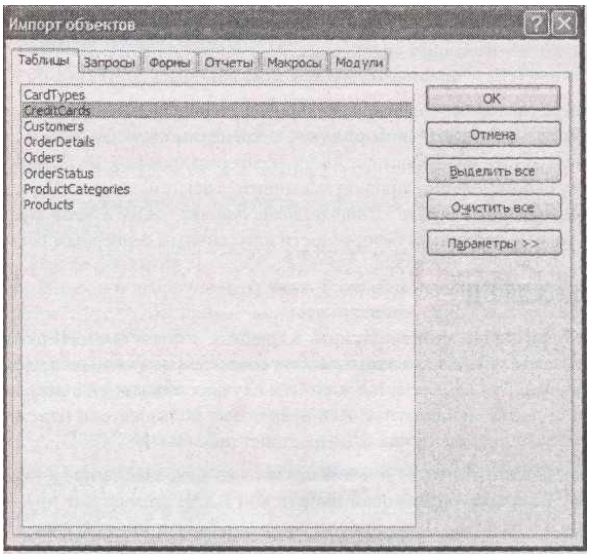


Рис. 18.7. На вкладке **Таблицы** перечислены все таблицы вашей БД. Выделите одинарным щелчком кнопки мыши те, которые хотите импортировать

После завершения передачи данных в серверные БД, самое время обновить клиентскую БД.

9. Откройте клиентскую БД.

В нашем случае файл BoutiqueFudge.accdb.

10. Удалите все таблицы.

Не бойтесь - в конце концов, вы только что скопировали их в файлы серверных БД. После завершения удаления следует выполнить последний шаг - создать нужные вам связи в клиентской БД. Если ваши таблицы связаны друг с другом, начните с подчиненных или дочерних таблиц.

11. Выберите на ленте **Внешние данные → Импорт →» Access** (External Data → Import → Access).

Снова запустится мастер импорта.

12. Укажите мастеру импорта на первый серверный файл, выберите переключатель **Создать связанную таблицу для связи с источником данных** (Link to the data source) и затем нажмите кнопку ОК.

Начните с файла BoutiqueFudgeSecrets_be.accdb.

13. Выберите все таблицы и щелкните мышью кнопку ОК.

Программа Access создаст соответствующие связанные таблицы в вашей БД. Рядом с каждой таблицей в области переходов отображается сигнальная пиктограмма стрелки, чтобы дать вам знать о применении связи.

14. Повторите пункты 11 - 13 для каждой серверной БД.

Если вы начали с файла BoutiqueFudgeSecrets_be, пора перейти к файлу BoutiqueFudge_be, содержащему все остальные нужные вам таблицы.

Если вы выполнили все перечисленные действия, то получите три файла БД, работающие вместе: BoutiqueFudgeSecrets__be.accdb с информацией о кредитных картах, Boutique-Fudge__be.accdb с остальными таблицами и BoutiqueFudge.accdb с запросами, формами и отчетами. Если хотите увидеть окончательный продукт, загляните в загружаемые из Интернета примеры к данной главе. Перейдите в *разд. "Защита базы данных" далее в этой главе,* чтобы узнать, как применять разные параметры безопасности к различным серверным БД.

18.2.5. *Блокировка вашей клиентской БД*

Прежде чем выпустить вашу БД в жизнь, стоит подумать о вредных последствиях. В руках не слишком смышленых пользователей Access ваши любовно создаваемые формы и отчеты могут быть безнадежно испорчены. Эта самая частая жалоба в случаях совместного использования БД Access: раньше или позже любопытные или небрежные пользователи изменят что-

нибудь, что не стоило трогать, и их клиентская БД перестанет работать.

Несмотря на то, что вы не сможете стоять за плечом каждого из них, можно предотвратить проделки пользователей с помощью блокировки клиентской БД. В этом случае другие люди не смогут изменять формы и отчеты. (При этом они все равно смогут просматривать и редактировать данные.)

Секрет установки блокировки вашей клиентской БД заключается в замене расширения файла accdb на расширение accde. Несмотря на разницу всего в одной букве формат с расширением accde ограничивает пользователей несколькими способами:

- они не могут изменять формы и отчеты, на самом деле они даже не могут открывать эти объекты в режиме **Конструктора;**
- они не могут создавать новые формы и отчеты;
- они не могут переименовывать существующие формы и отчеты (хотя могут их удалить);
- они не могут редактировать или даже просматривать ваш программный код и макросы. В действительности весь программный код откомпилирован, т. е. преобразован из операторов кода, о которых вы узнали в *главах 16-17,* в "стенографическую" запись, понятную только компьютеру.

Примечание

Программа Access предоставляет такие же возможности в отношении БД более старого mdb-формата. Для блокировки внесения изменений в файл с расширением mdb создайте файл с расширением mde.

Создать файл с расширением accde проще простого. Достаточно выполнить следующие действия.

1. Откройте вашу клиентскую БД.
2. Убедитесь, что она запускается как надежная БД или БД с полным доверием.

Если вы открыли ее не из надежного расположения *(см. разд. "Задание надежного расположения" главы 15),* необходимо щелкнуть мышью кнопку **Параметры** (Options) на панели сообщений, выбрать команду **Включить содержимое** (Enable Content) и затем щелкнуть мышью кнопку ОК.

3. Выберите на ленте **Работа с базами данных → Работа с базами данных → Создать ACCDE** (Database Tools → Database Tools → Make ACCDE).

На экране появится диалоговое окно **Сохранить как** (Save As).

4. Введите имя для вашего файла с расширением accde.

Программа Access не изменит исходную БД - вместо этого она создаст копию нового формата.

После создания файла с расширением accde убедитесь в том, что исходный файл с расширением accdb у вас в руках. Рано или поздно вам придется вносить изменения. Программа Access не предоставляет никакого способа обратного преобразования файла с расширением accde в исходный формат, поэтому единственная возможность вернуться к первоначальному файлу - внести изменения и затем экспортировать его в новый файл с расширением accde.

Примечание

Если исходный файл с расширением accdb потерян, нет возможности изменить ваши формы и отчеты. Вы остаетесь с неизменной во времени БД. В качестве последнего средства можно обратиться к Web-пространству, в котором другие компании предлагают утилиты, умеющие (обычно) преобразовывать файл с расширением accde в файл с расширением accdb.

Часто задаваемый вопрос.

Когда не следует пользоваться форматом ACCDE

Формат ACCDE подходит только для клиентских БД?

Преобразовать в файл с расширением accde можно любую БД. Но следует дважды подумать, прежде чем применять преобразование к неклиентской БД, поскольку трудно обновлять файл с расширением accde, содержащий данные.

Для того чтобы понять проблему, представьте себе, что создается файл с расширением accde для БД, содержащей всю информацию о продажах по сниженным ценам средств по уходу за

волосами. Эта БД включает все исходные данные - списки клиентов, доступные сервисы и счета - и содержит формы и отчеты, облегчающие жизнь. Разделения на серверную и клиентскую часть нет.

Через несколько недель вы решаете добавить новый отчет, отображающий клиентов, разделенных на подгруппы в зависимости от цвета их краски для волос.

Конечно, вы не можете редактировать непосредственно файл с расширением accde, поэтому вы доводите до совершенства отчет в исходном файле с расширением accdb и создаете новый файл с расширением accde. И тут возникает проблема - в вашем исходном файле с расширением accdb старые данные. Теперь у вас два неполных файла: accde-файл с новыми данными, но старыми формами и отчетами и новый accde-файл с правильными формами и отчетами, но некорректными данными. Для разрешения этой ситуации нужно выполнить требующую много времени операцию импорта, как описывалось ранее.

Во избежание подобных проблем синхронизации данных применяйте формат accde для того, для чего он предназначен - блокировки клиентских БД, не содержащих никаких таблиц.

18.2.6. *Использование БД совместно с пользователями, у которых нет Access*

Было бы неплохо, если бы пользователи могли работать с вашей БД и пользоваться вашими формами и отчетами без установки полной версии программы Access на своих компьютерах? Это может показаться фантастикой, но такой способ есть.

Корпорация Microsoft предоставляет усеченную версию программы Access, которая называется *исполняемой средой Access* (Access runtime engine). Вместо покупки отдельной копии Access для каждого пользователя, которому приходится использовать вашу БД, можно им всем дать копию такой исполняемой среды. Затем они могут использовать ее для загрузки вашей БД и применения ее форм и отчетов для просмотра и редактирования данных.

Исполняемая среда Access не обладает всеми возможностями программы Access. Сразу заметно, что у нее нет ленты и области переходов. На самом деле она не предоставляет пользователям возможностей для изменения конфигурации или проекта БД. (Это ваша работа как разработчика БД.) Единственно, что вы можете делать с помощью исполняемой среды Access - применять формы и отчеты, включенные в клиентскую БД.

Примечание

Если в исполняемой среде Access использовать хорошо спроектированную клиентскую БД, пользователи могут даже не знать, что они работают в Access.

Итак, как же добраться до исполняемой среды Access? Во время написания книги она еще не была выпущена. Однако корпорация Microsoft обещала сделать ее доступной в начале 2007 г. (и в отличие от Access 2003 Microsoft обещает, что исполняемая среда Access 2007 будет доступна на ее Web-сайте и тесно связана со средствами программирования Visual Studio). Для того чтобы узнать последние новости о состоянии исполняемой среды Access, щелкните кнопкой мыши ссылку на страницу "Missing CD" на Web-сайте www.missingmanuals.com.

Между тем уже сейчас можно посмотреть, как будет выглядеть БД в исполняемой среде. Вот как это сделать.

1. Откройте вашу БД и убедитесь в том, что у нее есть стартовая форма или форма просмотра.

У исполняемой среды Access нет области переходов, поэтому у вас должна быть форма просмотра, обеспечивающая работу пользователя. Она может быть кнопочной формой с кнопками, направляющими на другие формы.

Для установки стартовой формы выберите кнопку **Office → Параметры Access** (Office → Access Options). В списке слева выберите категорию Текущая база данных (Current Database). И, наконец, задайте в параметре **Форма просмотра** (Display Form) форму, которую хотите отображать автоматически при открытии БД.

2. Замените расширение файла БД accdb расширением accdr. (Возможно, буква "r" обозначает runtime (исполняемая).)

3. Дважды щелкните кнопкой мыши вашу БД для запуска в режиме исполнения. Вы увидите форму просмотра, но без ленты и области переходов (рис. 18.8).

Не беспокойтесь, Access может восстановить расширение accdb в вашем файле, чтобы вернуть его обычный формат.

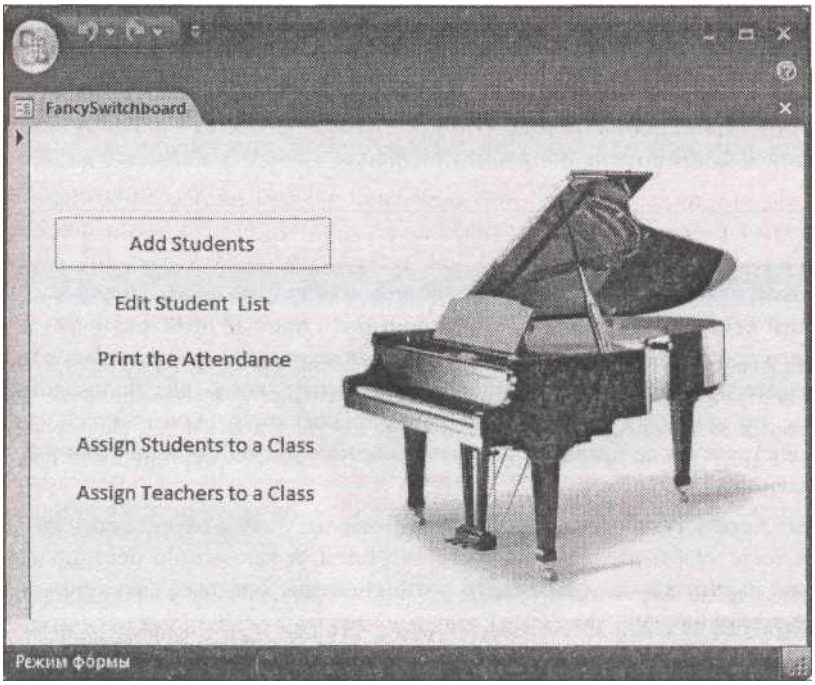


Рис. 18.8. Этот файл с расширением accdr использует искусную кнопочную форму

Подсказка

Исполняемая среда Access - действительно полезный способ совместного использования БД без покупки несметного количества лицензий Access и забивания голов пользователей многочисленными свойствами интерфейса полнофункциональной версии программы Access. Если вы координируете деятельность маленькой коммерческой компании, она вполне подходит.

Уголок ностальгии.

Отмирание страниц доступа к данным

У программы Access 2003 было средство, названное страницами доступа к данным и предназначенное для создания клиентской части БД в виде Web-страницы. Идея была замечательная - с помощью таких Web-страниц любой пользователь мог просматривать информацию в вашей БД или вносить в нее изменения, даже не имея программы Access.

К сожалению, у Access и Web натянутые отношения. Как вы уже знаете, программа Access может обрабатывать только ограниченное число одновременных обращений.

У Web-технологий нет таких ограничений. В действительности размещение БД в Интернете равносильно приглашению полчищ пользователей для попыток одновременного использования БД. По этой причине, как и по некоторым другим (например, трудность настройки страниц доступа к данным и отсутствие совместимости с другими обозревателями), корпорация Microsoft отказалась от этого средства в Access 2007.

Если вы все же хотите придать Web-обличье вашей БД, программа Access мало чем может помочь вам. Лучшим выбором будет перенос БД на серверный программный продукт, например SQL Server, и применение другого средства разработки для построения Web-страницы. Серьезные программисты любят ASP.NET (см. www.asp.net), единый комплект корпорации Microsoft для создания Web-сайтов, от простейших до самых сложных.

18.3. *Многопользовательский доступ*

Многопользовательский доступ - это вечное жонглирование. Если все пользователи хотят читать информацию, все просто. Но в тот момент, когда они захотят внести изменения, могут возникнуть серьезные проблемы. Например, что произойдет, когда два пользователя попытаются одновременно изменить одну и ту же запись? Или, когда вы пытаетесь изменить

запись, кто-то еще старается ее удалить? Или же вы хотите прочесть свежие данные во время процесса обновления?

Ясно, что программе Access нужен способ управления хаосом. В этом разделе вы узнаете, что делает Access для того, чтобы все держать под контролем, и как можно регулировать этот процесс. Вы также научитесь предупреждать возникновение опасных ситуаций, способных привести к повреждению данных.

18.3.1. *Как вносятся изменения*

Представим себе следующий сценарий. Вы говорите по телефону с расточительным клиентом компании Boutique Fudge. Используя вашу надежную БД Access, вы просматриваете имеющиеся в наличии товары и сообщаете вашему клиенту цену каждого из них. А в это же время главный повар без вашего ведома просматривает ту же таблицу и повышает цены на самые популярные блюда. Возникает вопрос: когда вы заметите, что цены выросли?

Программа Access обрабатывает ситуации, подобные этой, с помощью автоматических обновлений. Каждые 60 секунд Access проверяет клиентскую БД, чтобы выяснить, что изменилось. Затем программа обновляет соответствующую информацию на вашем экране, если вы просматриваете форму, запрос или непосредственно таблицу. В примере с Boutique Fudge новые цены появятся после очередного обновления, выполняемого программой Access, но не позже 60 секунд, после внесения изменений.

Из правила обновления есть несколько исключений.

■ Когда начинается редактирование записи (щелчок кнопкой мыши в одном из полей), Access немедленно обновляет именно эту запись. Это действие гарантирует, что перед тем как вносить изменения, вы получите самую свежую копию записи.

■ Если вы не можете ждать 60 секунд и вас беспокоят возможные изменения с момента последнего обновления, можно запустить немедленное обновление с помощью команды **Главная → Записи → Обновить все** (Home → Records → Refresh All). Если щелкнуть мышью часть кнопки команды с нарисованной стрелкой, направленной вниз, можно выбрать обновление только текущей записи, в которой находится курсор (выберите команду **Обновить** (Refresh) вместо команды **Обновить все** (Refresh All)).

■ В отчетах не применяется автоматическое обновление. Если вы формируете отчет, подождите, а затем решите, нужно ли обновлять результаты. У вас есть два варианта. Можно закрыть отчет и снова открыть его или использовать кнопку на ленте **Обновить.**

Если вам не нравится правило 60 секунд, можно настроить частоту повторения автоматических обновлений программой Access. Для этого выберите кнопку **Office → Параметры** Access (Office → Access Options). В окне **Параметры Access** выберите категорию параметров **Дополнительно** (Advanced) и прокручивайте перечень, пока не увидите параметр **Период обновления (с)** (Refresh interval (sec)) (рис. 18.9).

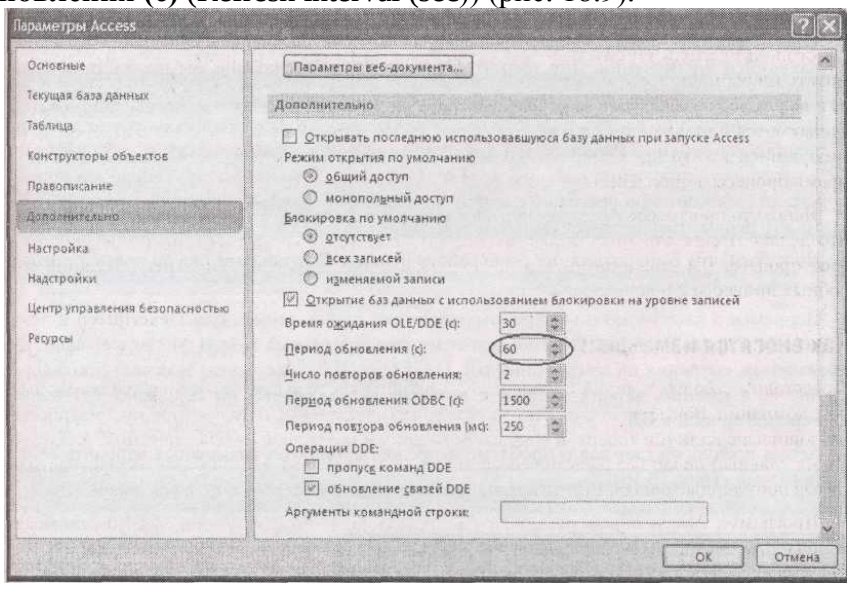


Рис. 18.9. Период обновления определяет, как часто программа Access проверяет в многопользовательской БД внесение изменений. Можно выбрать значение (в секундах) от 1 до 32 766

Примечание

Период обновления (с) - параметр программы Access, влияющий на все многопользовательские БД, открываемые на данном компьютере. Если вы хотите, чтобы все использовали одинаковый период обновления, необходимо, чтобы все пользователи обновили свои параметры Access.

Чем короче период обновления, тем быстрее вы увидите изменения, внесенные другими пользователями. Но укороченные интервалы обновления создают дополнительный трафик в вашей сети. Многие любители программы Access знают, что можно спокойно применять более короткий период обновления, если используется медленная сеть.

18.3.2. *Обработка конфликтов редактирования*

Многопользовательские БД - пример неограниченной свободы. Обычно программа Access не накладывает никаких ограничений на многопользовательские корректировки. Если повезет, пользователи будут вносить изменения в определенном порядке, один за другим. Но рано или поздно, изменения наложатся друг на друга и, возможно, с серьезными последствиями.

Далее приведен пример, показывающий, к чему приводит наложение изменений.

1. Вы открываете запрос, отображающий все товары в БД Boutique Fudge.
2. Ищите запись - самый хорошо продаваемый чизкейк с названием Chocolate Abyss, нуждающийся в изменении. Для того чтобы начать редактирование, вы щелкаете кнопкой мыши в поле **Description.**
3. В это же время Билл Эванс (Bill Evans) в отделе продаж запускает форму, которая тоже использует таблицу **Products.** Он добирается до той же самой записи и, догадываясь о возможности получений большей прибыли, начинает изменять цену. Теперь два пользователя работают одновременно с одной записью. Что произойдет дальше, зависит от того, кто первым зафиксирует свои изменения.
4. Допустим, что Билл выполнил свою работу первым. В мгновение ока он повысил цену и перешел к другой записи.
5. Вернемся к вашему компьютеру, вы закончили исправление поля **Description** и переходите к другой записи. Обычно в этот момент программа Access фиксирует ваши изменения, сохраняя их в серверной БД. Но в данном случае Access замечает противоречие - а именно, версия записи, с которой вы работаете, не совпадает с текущей версией записи в БД.
6. Access предупреждает вас о проблеме и предоставляет три возможных варианта ее ис правления (рис. 18.10).

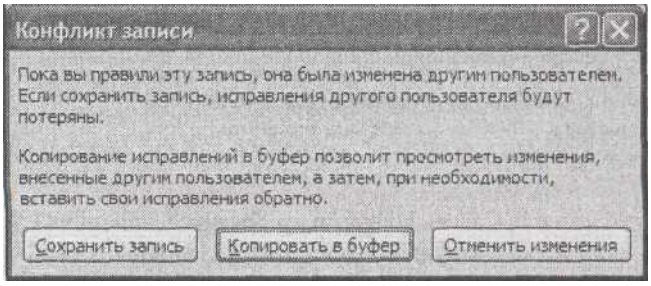


Рис. 18.10. Между временем начала вашего редактирования и моментом внесения его в БД кто-то еще внес изменения. Программа Access разрешает выбрать вариант обработки возникшей конфликтной ситуации

Разрешить конфликтную ситуацию можно тремя способами.

Вариант **Сохранить запись** (Save Record) - самый легкий и самый опрометчивый. Если выбрать его, программа Access запишет вашу версию записи поверх имеющейся в БД. Проблема состоит в том, что этот вариант аннулирует изменения, сделанные другим пользователем. В предыдущем примере в БД будет сохранено новое описание товара, но потеряется изменение цены, поскольку Access запишет ее прежнее устаревшее значение.

Вариант **Отменить изменения** (Drop Changes) отменяет ваше редактирование. Access обновит запись и отобразит самую свежую информацию, а затем вы можете попытаться внести ваше изменение снова. Этот вариант заслуживает внимания, если вы легко сможете повторить

редактирование, но его нельзя считать хорошим выбором, если вы только что закончили подробное обновление большого текстового поля.

Вариант **Копировать в буфер** (Copy to Clipboard), как и предыдущий вариант, отменяет ваше редактирование. Однако значения, которые вы изменили, копируются в буфер, что облегчит повторное их применение, как показано на рис. 18.11.

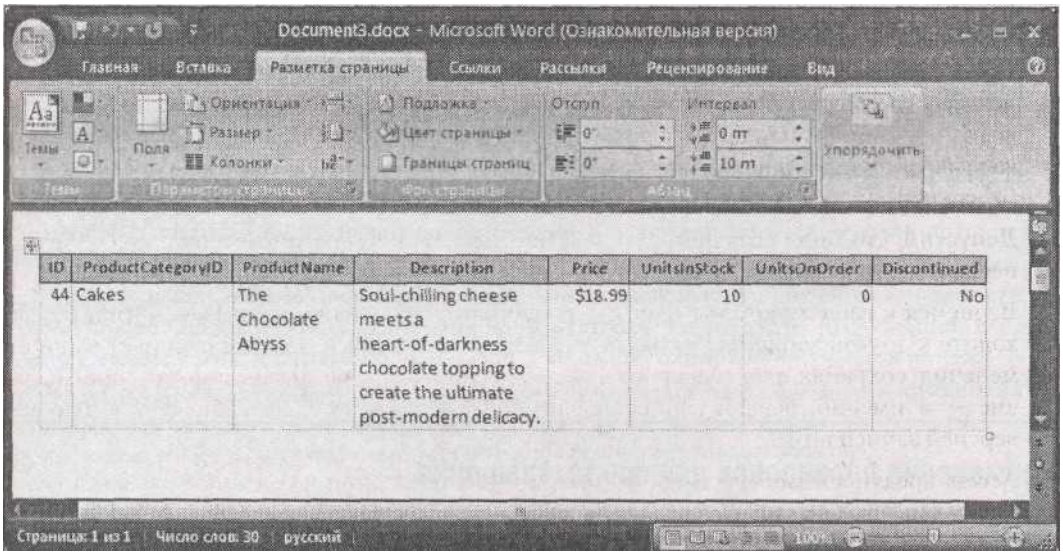


Рис. 18.11. Если результаты последнего редактирования скопированы в буфер, понадобятся два шага для повторного их внесения. Сначала вставьте их в другую программу (например, показанную здесь программу Word). Затем выделите данные, которые хотите использовать, и снова скопируйте их в буфер с помощью комбинации клавиш <Ctrl>+<C>. И, наконец, перейдите в окно программы Access к полю, которое хотите изменить, и вставьте новое значение, нажав комбинацию клавиш <Ctrl>+<V>

Обычно лучше всего копировать в буфер и пытаться повторить редактирование. К сожалению, людей невозможно заставить поступать правильно. Ленивые работники могут выбрать более быстрый вариант **Сохранить запись,** который незаметно уничтожит чью-то работу. Хуже всего, что у пользователя, выполнившего изменение первым, нет возможности узнать, что его работа пошла насмарку. Если у вас в организации много накладывающихся корректировок, придется потратить много времени на обучение персонала правилам поведения при обработке подобных ситуаций.

Примечание

Любители Access часто говорят о том, что им хотелось бы иметь возможность объединять изменения, т. е. обновлять только те поля, которые менялись. В предыдущем примере такая возможность позволила бы применить новое описание поля без повреждения изменения цены, сделанного предыдущим пользователем, поскольку оба обновления воздействуют на разные поля. Но программа Access не предоставляет такого варианта. Одна из причин - отсутствие возможности проверить непротиворечивость обоих изменений. Нет ничего хуже записи, противоречащей самой себе.

Практические занятия для опытных пользователей.

Разделение таблиц для более безопасных корректировок

Один из способов снижения числа накладывающихся корректировок - разделение таблицы на более мелкие фрагменты. Основная идея - взять единую таблицу с большим числом полей и разделить ее на две меньшие таблицы, каждая из которых включает только некоторое число этих полей. Например, можно взять таблицу Customers (клиенты) и разделить на таблицу **CustomerAddress** (адрес клиента) и таблицу **CustomerFinancial** (финансовая информация клиента). Каждая запись таблицы **CustomerAddress** связана с единственной записью таблицы **CustomerFinancial** с помощью отношения "один-к-одному". Для получения всей информации о клиенте вам понадобятся обе записи.

Лучше всего делить таблицу, когда вы выясните, что типичное редактирование включает

поля только из одной таблицы. Возможно, вы знаете, что отдел по обслуживанию клиентов часто обновляет адресные данные, а финансовый отдел работает с финансовыми сведениями, в этом случае разделение таблицы - прекрасная идея. Отдел по обслуживанию клиентов будет использовать таблицу **CustomerAddress** почти монопольно, а финансовый отдел будет работать с таблицей **CustomerFinancial.** Вероятность наложения корректировок резко снижается, поскольку обработка делится между двумя таблицами.

18.3.3. *Применение блокировок для предотвращения наложения обновлений*

Если накладывающиеся изменения вызывают слишком много проблем, у вас есть еще одна возможность. Можно применить программный трюк, именуемый *блокировкой,* препятствующей наложению корректировок.

По существу блокировка использует ту же идею, которая мешает двум людям встретиться в одной туалетной кабинке. Когда один человек заходит, он или она включает блокировку и все остальные вынуждены ждать, пока дело не будет сделано. Аналогичным образом, когда пользователь пытается изменить запись, программа Access начинает с установки блокировки этой записи. Любой другой пользователь, желающий внести изменения, вынужден ждать, пока не завершится первая операция.

Самый легкий способ применения блокировок - включение их с помощью параметров Access. Для этого выберите кнопку **Office → Параметры Access** (Office → Access Options), Затем перейдите к категории **Дополнительно** (Advanced) и найдите параметр **Блокировка по умолчанию** (Default record locking).

У вас есть три следующих варианта для выбора.

- Вариант **Отсутствует** (No locks) - это стандартная установка в программе Access. Когда применяется этот вариант, Access вообще не будет использовать блокировки и наложение корректировок становится возможным.
- Вариант **Всех записей** (All records) заставляет программу Access блокировать всю таблицу целиком, если кто-либо начнет редактировать запись. Это значение применяется крайне редко, поскольку оно блокирует все записи и препятствует работе с таблицей кого бы то ни было, если вносится хотя бы одно изменение. Такое ограничение может ввести любую организацию в тяжелый ступор.
- Вариант **Изменяемой записи** (Edited record) блокирует отдельные записи, когда они редактируются, что препятствует наложению корректировок.

Последний вариант - наиболее распространенный вид блокировок. Когда применяется блокировка отдельных записей, программа Access не разрешит начать редактирование записи, если в данный момент кто-то еще ее изменяет. Когда вы сделаете попытку, Access выведет на экран пиктограмму, обозначающую блокировку записи, показанную на рис. 18.12.

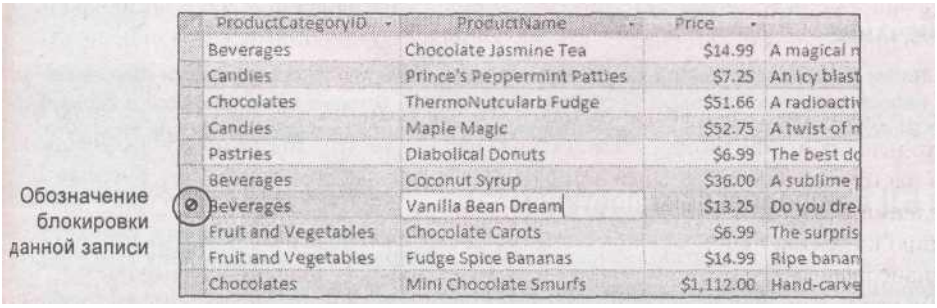


Рис. 18.12. Символ "Вход воспрещен" предупреждает о необходимости ожидания вместо редактирования записи, которая уже используется. Если вы все же попытаетесь ввести данные в поле, программа Access решительно проигнорирует вашу попытку

Блокировки мешают превращению вашей БД в груду беспорядочных сведений, но они порождают другие проблемы. Программа Access вынуждена выполнять дополнительную работу по отслеживанию каждой блокировки - она должна действовать как перегруженный работой "умывальников начальник", выдающий ключи от кабинок. Access отслеживает блокировки, создавая файл с расширением laccdb. Например, когда кто-то в первый раз открывает

БД с многопользовательской поддержкой BoutiqueFudge_be.accdb, программа Access создает файл, названный BoutiqueFudge_be.laccdb. (Символ "l" обозначает "locking" (блокировка).) Когда последний пользователь закрывает БД, Access удаляет файл с блокировками.

Подсказка

Если вы загляните в папку с общим доступом и не увидите там файла с расширением laccdb, знайте, что в этот момент никто не использует БД или же кто-то открыл ее с монопольным доступом (Exclusive mode).

Кроме того, блокировки тормозят работу остальных пользователей, заставляя их ждать нужные им данные. Небрежный пользователь может прекратить работу с записью на неопределенное время, оставив ее в режиме редактирования.

Примечание

Отправляясь на перерыв, вы можете свернуть работу всей компании, даже не догадываясь об этом. Еще хуже то, что хотя потенциальные редакторы будут видеть, что запись заблокирована, они никак не смогут установить кто - причина их бед. Им остается только ждать и ждать.

Если вы все же решили применять блокировки, гораздо лучше использовать их в отдельных формах, а не включать во всех БД с помощью параметров программы Access. Можно использовать для всей БД стандартное значение для блокировки *Отсутствует* (No locks), но задать применение блокировок во всех формах, использующих особенно важную таблицу, скажем **Invoices** (счета).

Для изменения способа обработки блокировок в формах откройте **Окно свойств** (Property Sheet) и найдите свойство **Блокировка записей** (Record Locks). Оно поддерживает те же три значения: *Отсутствует* (No Locks) (значение по умолчанию), *Всех записей* (All Records) и *Изменяемой записи* (Edited Record).

Примечание

Этот прием оставляет открытой дверь черного хода. Если кто-то решит внести изменение, открыв непосредственно таблицу, он обойдет блокировки, которые вы установили в своих формах. Как всегда, людей легко направить верным путем, но трудно удержать их на этом пути.

18.3.4. *Открытие БД с монопольным доступом*

Одно из ограничений БД с многопользовательской поддержкой - невозможность изменять структуру таблиц во время работы пользователей с БД. Прежде чем вы сможете внести радикальные изменения, нужно открыть БД с монопольным доступом (Exclusive mode).

Монопольный доступ временно предоставляет БД в единоличное пользование одному человеку. Когда БД открыта с монопольным доступом, никто другой не может к ней обращаться, независимо от того, какую клиентскую БД они используют. У вас есть драгоценное время для выполнения радикальных изменений, которые обычно вы не можете внести.

Для того чтобы открыть БД с монопольным доступом, выполните следующие действия.

1. Попросите всех пользователей закрыть БД.

Вы не сможете открыть БД с монопольным доступом, если она кем-то используется в настоящий момент. В большой компании это самый трудный этап. Иногда системные администраторы прибегают к помощи электронной почты, чтобы дать всем знать, что пришло время завершать работу. Другой вариант - научить пользователей, работающих с вашей БД, закрывать ее всегда на ночь, прежде чем уйти домой, что позволит вам без потерь дремать во время обновления, проводимого в ночные часы.

2. Выберите кнопку **Office → Открыть** (Office → Open).

На экране появится окно **Открытие файла базы данных.**

3. Выберите файл БД, который хотите открыть, и щелкните кнопкой мыши направленную вниз стрелку на кнопке **Открыть** (Open).

На экране появится список специализированных параметров открытия вашего файла (рис.

18.13).

4. Выберите **Монопольно** (Open Exclusive).

Программа Access откроет БД. Теперь можно внести изменения без каких-либо ограничений. Но действуйте быстро - чем дольше вы держите БД открытой с монопольным доступом, тем дольше другие пользователи вынуждены ждать момента, когда они смогут приступить к своей работе.

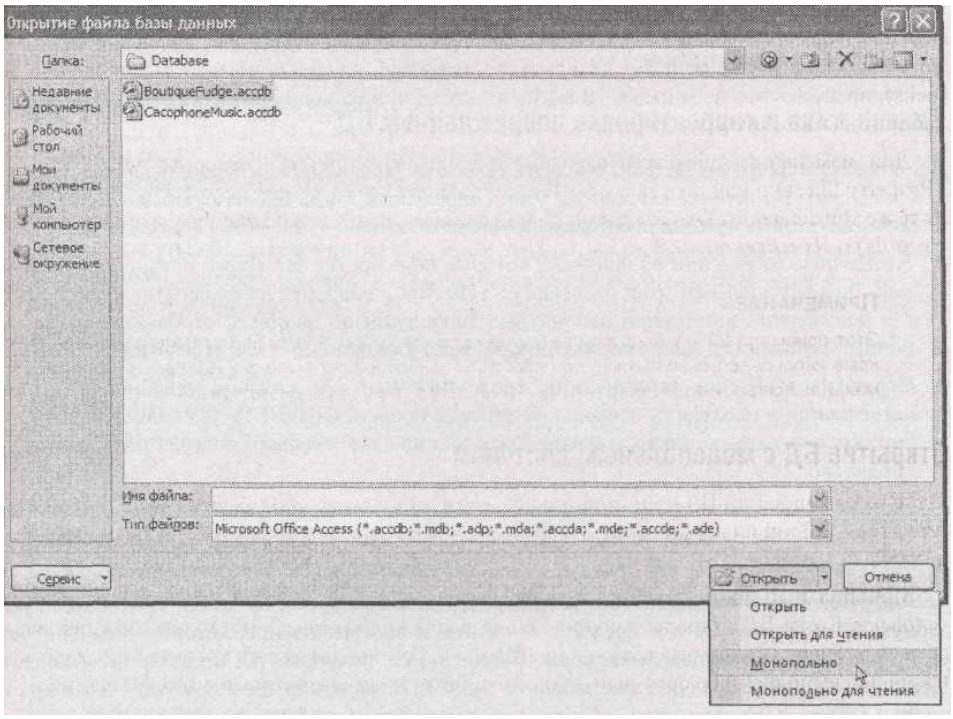


Рис. 18.13. Если открыть БД **Монопольно,** никто яругой не сможет ее открыть, пока вы ее не закроете. Если открыть файл в режиме **Открыть для чтения,** нельзя будет внести никакие изменения

Примечание

Можно настроить программу Access так, что она всегда будет пытаться открыть любую БД монопольно. Но для такого подхода редко появляются основания, поскольку он сводит на нет многопользовательскую поддержку.

18.4. *Повреждение данных*

Повреждение или *нарушение целостности данных* - термин, вселяющий страх в сердца стойких гуру Access. Будем надеяться, что пользователи вашей БД, будут вести себя достойно, сеть в, которой она размещена, будет надежной, а вашей БД никогда не будет угрожать опасность. Но если судьба не будет к вам столь благосклонна, важно быть готовым к ее ударам.

Повреждение данных - многосмысловой термин, описывающий, что происходит, когда повреждается часть файла БД. Представим себе Джессику Бакстер (Jessica Baxter), которая успела внести половину большого обновления, когда произошло отключение питания (или какой-то шутник в офисе выдернул ее сетевой кабель). Серверная БД останется с недостоверными данными, поскольку была получена только часть информации Джессики. В результате запись, с которой она работала, может быть искажена до неузнаваемости. Если же вы особенно невезучи, проблема может затронуть несколько записей или заставить всю БД вести себя странным образом.

18.4.1. *Диагностика и корректировка поврежденных БД*

Все специалисты Access должны обладать базовыми навыками выживания в случае повреждения данных. Во-первых, вы должны уметь определять, когда БД испортилась. Далее пере-

числены некоторые признаки, сигнализирующие об этом.

■ Загадочные сообщения об ошибках, которые появляются без видимой причины, например, "нехватка памяти" (out of memory). (Имейте в виду, что их не следует путать с всегда популярной категорией непонятных сообщений об ошибках, отображаемых на законных основаниях, например таких, как "файл уже используется" (file already in use).)

■ Строки, содержащие тарабарщину, вроде ### или ???. Обычно подобные значения встречаются в последних нескольких строках поврежденной БД, что свидетельствует о том, что пострадало только несколько новых вставок, а остальное содержимое, возможно, корректно.

■ Полная невозможность использования БД. Если вы получаете ужасную ошибку "Unrecognizable database format" ("Нераспознаваемый формат базы данных"), знайте - беда пришла.

Как только вы установили, что БД повреждена, самое время начинать ее лечить, чтобы вернуть ей здоровье. Первое средство - сжатие и восстановление, которые ликвидируют узел проблем и возвращают непомерно разросшимся, раздутым БД приемлемые размеры. Для опробования этого средства откройте вашу БД с монопольным доступом и выберите кнопку **Office → Управление → Сжать и восстановить базу данных** (Office → Manage → Compact and Repair Database). Процесс может потребовать некоторого времени, особенно в случае больших БД.

Средство сжатия и восстановления исправляет только таблицы, но не формы и отчеты. Но если вы были предусмотрительны и создали разделенную БД, в серверной части нет объектов указанных типов.

Примечание

Прежде чем исправлять поврежденную БД, немедленно сделайте резервную копию. В этом случае вы сможете испробовать несколько стратегий восстановления.

Иногда сжатие и восстановление не решают проблему или могут лишь частично исправить файл вашей БД. В этом случае пора воспользоваться другим методом восстановления, Если оставшиеся проблемы не значительны (например, несколько строк с подозрительными данными), можно просто удалить бракованные данные и ввести их снова. Но иногда

программа Access отказывается отображать испорченные записи без многочисленных сообщений об ошибках. Если вы столкнулись с такой ситуацией, выберите все корректные записи и скопируйте их в другую таблицу. Затем удалите таблицу с испорченными данными и присвойте ее имя сделанной вами копии.

Последним средством спасения может стать создание новой пустой БД и попытка импортировать в нее таблицы серверной БД с помощью операции импорта. Это действие заставит программу Access создать заново каждый объект и перестроить каждый индекс. Если и это сработает не до конца, возможно, вам удастся импортировать большую часть таблиц.

И самое последнее средство - возврат к последней резервной копии БД. Вы ведь храните резервные копии, не правда ли?

18.4.2. *Предупреждение повреждений*

Приведенные далее рекомендации могут помочь сделать редким такое жуткое событие, как повреждение данных.

■ Придерживайтесь благоразумных установок по умолчанию, описанных в *разд. "Как действует многопользовательская поддержка в Access" ранее в этой главе.* Если десятки пользователей попытаются внести изменения одновременно, шансы появления проблем многократно увеличатся.

■ Всегда разделяйте БД, чтобы облегчить загрузку серверной части и держать формы и отчеты в безопасном месте.

■ Используйте надежную сеть. Если ваше сетевое соединение не заслуживает доверия, обновление может быть прервано, а это одна из главных причин нарушения целостности данных.

■ Приучите пользователей закрывать БД, когда они закончили ее использование или даже просто прервались на завтрак.

■ Регулярно применяйте средство сжатия и восстановления к вашей серверной БД (выберите из меню **Office** последовательность **Управление → Сжать и восстановить базу данных** (Manage → Compact and Repair Database)). Чем больше людей вносят изменения, тем сильнее разрастаются файлы БД и становятся более неорганизованными. Команда сжатия и восстановления реорганизует БД, делая ее более эффективной, более компактной и менее подверженной сбоям.

■ Делайте резервные копии как можно чаще. В зависимости от частоты внесения изменений может оказаться достаточным ежедневное резервное копирование. Но ничто не мешает вам создавать резервные копии каждый час или чаще, если это необходимо.

Подсказка

Обязательно храните коллекцию самых свежих резервных копий. Если сохранять один файл с резервной копией, велик риск того, что эта копия была сделана с уже поврежденной БД и у вас нет более старой копии для успешного отступления.

18.5. *Защита базы данных*

В большинстве многопользовательских БД разные пользователи выполняют различные задачи. Легче всего удержать каждого на верном пути, если создать несколько отдельных клиентских БД, по одной для каждой группы пользователей. Это позволит мягко управлять пользователями в соответствии с выполняемыми ими задачами.

Однако настройка клиентской БД не ограничит возможности упрямого злоумышленника. В большой компании, рассчитывающей на многопользовательскую БД, вы думаете не о руководстве пользователями, а скорее об ограничении их возможностей.

К сожалению, у программы Access 2007 очень ограниченная модель безопасности. Вы можете защитить БД от посторонних с помощью пароля, но у вас нет более тонко настраиваемых средств, необходимых для запрета использования конкретных таблиц или выполнения определенных действий тем или иным пользователем. Как вы увидите в следующих разделах, обходные средства все-таки есть, но ни одно из них не сравнится с уровнем безопасности, обеспечиваемым системой управления серверной БД, такой как, например, SQL Server.

Подсказка

И снова программа Access предлагает достаточно для того, чтобы многопользовательские БД могли работать, но не более того. Вам решать, подойдет ли Access для вашей организации. Маленькие фирмы, вероятно, сочтут программу отличной, а большим организациям, возможно, понадобится серверное программное обеспечение.

Уголок ностальгии.

Защита с помощью рабочих групп упразднена

Опытные специалисты Access, может быть, помнят, что в предыдущие версии Access была включена гораздо более полезная форма защиты на уровне пользователя, называемая *защитой с помощью рабочих групп.* Программа Access хранила отдельный файл, в котором определялось, что разрешено делать с БД каждому пользователю и группе. Такую защиту легче реализовать, чем защиту на уровне файлов, и она гораздо гибче. На самом деле она представляется прекрасным решением.

К сожалению, защита с помощью рабочих групп никогда не обеспечивала настоящей защиты. Для ее преодоления могли использоваться широко доступные программистские приемы. Чаще всего это не представляло серьезной проблемы, потому что специалисты Access по-настоящему не стремились остановить умелых хакеров. Они больше заботились о сохранении контроля над обычными пользователями.

Однако в последние годы корпорация Microsoft зациклилась на безопасности. Когда она готовила к выпуску Access 2007, было решено больше не поддерживать средство безопасности, которое на самом деле ненадежно, особенно когда в других программных продуктах, например, свободно распространяемой версии SQL Server, доступны лучшие механизмы. По этой причине корпорация Microsoft отказалась от поддержки безопасности с помощью рабочих групп в файлах с расширением accdb. Это средство все еще можно применять в более старых файлах с

расширением mdb, но это стоит делать, только если вам нужно сохранить поддержку БД, которую вы проектируете, старой версией программы Access. Новые файлы всегда должны создаваться в формате accdb - помимо всего прочего, Access представляет его как формат будущего для БД.

18.5.1. *Защита паролем*

Защиту БД паролем программа Access предлагает как простое, без излишеств средство защиты. Вы выбираете единый пароль для вашей БД и с этого момента ее нельзя открыть без данного пароля. Более того, данные в вашей БД шифруются с помощью ключа, который генерируется из вашего пароля. Это гарантирует невозможность извлечения каких-либо данных, даже если технически грамотные хакеры сумеют пробраться непосредственно в файл БД с помощью специализированных средств.

Примечание

В Access 2007 защита паролем стала довольно серьезной. Программа применяет искусное шифрование, не позволяющее умелым хакерам взломать ваши файлы с помощью специальных средств - по крайней мере, не посвятив решению этой задачи довольно много времени.

Применить пароль до смешного просто. Вот как это делается.

1. Выберите кнопку **Office → Открыть** (Office → Open).

Для применения пароля следует открыть вашу БД монопольно. Этот шаг необходим, поскольку программа Access не может шифровать БД, пока она используется кем-то.

2. Выделите файл, который хотите открыть, щелкните мышью направленную вниз стрелку на кнопке **Открыть** (Open) и выберите строку **Монопольно** (Open Exclusive).

Программа откроет БД с монопольным доступом.

3. Выберите на ленте **Работа с базами данных → Работа с базами данных → Зашифровать паролем** (Database Tools → Database Tools → Encrypt with Password).

Access запросит пароль (рис. 18.14).

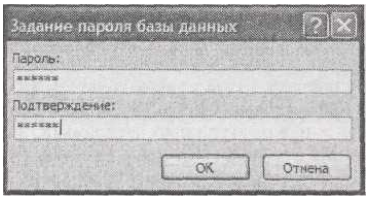


Рис. 18.14. Для большей уверенности программа Access попросит ввести пароль дважды

4. Введите пароль и щелкните мышью кнопку **ОК**.

Для того чтобы защитить вашу БД как следует, необходимо выбрать усиленный пароль. Хороши длинные пароли (10 символов или больше), такие, которые нельзя найти в словарях (хакеры используют словари для запуска автоматических атак) и содержащие специальные символы (например, цифры, знаки пунктуации и другие символы). Пароль hellodata - плохой выбор, пароль w0nDER_wh@t_32 гораздо надежнее.

Программа Access применяет пароль для шифрования вашей БД и затем автоматически сохраняет ее. Теперь при следующем открытии БД прежде всего программа запросит у вас пароль.

Если позже вы решите, что вам для защиты не нужен пароль, щелкните кнопкой мыши команду на ленте **Расшифровать базу данных** (Remove Database Password and Encryption).

18.5.2. *Пароли и разделенные БД*

Как пароли действуют в обычных БД, абсолютно понятно, но при работе с разделенными БД есть несколько интересных особенностей. Прежде всего, пароль всегда применяется для защиты серверной БД - в конце концов, защищать нужно именно данные, а не ваши формы и отчеты. Но тут есть интересная деталь: когда создается клиентская БД, которая связана с защищенной паролем серверной БД, программа Access спокойно сохраняет пароль в клиентской части. Это означает, что пока вы используете подходящую клиентскую БД, вам вообще

не нужно вводить пароль.

Примечание

Для того чтобы с успехом пользоваться паролем для серверной БД, вы должны применить пароль до того, как разделите БД. В противном случае программа Access не сохранит пароль в клиентской БД, и связанные таблицы не будут функционировать.

С технической точки зрения эта модель не обеспечивает безопасности промышленного уровня, поскольку ловкий хакер сможет украсть пароль, пробравшись в файл клиентской БД. Но до тех пор, пока вы уверены в том, что ваши клиентские БД не попадут в чужие руки, можно воспользоваться несколькими интересными возможностями.

- *Можно сохранять пароль в тайне, что помешает пользователям обращаться к серверной БД напрямую.* Вместо этого они должны полагаться на предоставленную вами клиентскую БД со встроенным паролем.
- *Клиентскую БД можно защитить другим паролем.* В этом случае коварный хакер, крадущий пароль вашей клиентской БД, будет все равно отрезан от БД.
- *Можно разделить серверную БД на отдельные файлы.* В этом случае каждому файлу можно присвоить свой пароль и помешать доступу пользователей к неподходящим таблицам. Если у их клиентской БД нет связи с нужной серверной БД, они не смогут использовать эти таблицы.

18.5.3. *Применение защиты файлов ОС Windows*

Защита паролем - не единственный доступный вам вариант защиты. Можно также использовать защиту ОС Windows для задания конкретных пользователей и групп, которые могут получить доступ к файлу.

Для выполнения этой работы необходимо разделить серверную БД на несколько файлов. Затем, после того как файлы размещены в папке с общим доступом, можно точно указать, кому разрешено обращаться к каждому из них. Надеюсь, что у вас есть администратор, готовый помочь вам. Базовый процесс выглядит следующим образом.

1. Используйте Проводник Windows, щелкните правой кнопкой мыши файл БД, который хотите защитить, и выберите строку **Свойства** (Properties).

На экране появится окно **Свойства** (Properties) с несколькими вкладками, содержащими информацию о файле.

2. Выберите вкладку **Безопасность** (Security) (рис. 18.15).

ОС Windows отслеживает пользователей двумя способами - она определяет каждого с помощью уникального имени пользователя и формирует группы пользователей с помощью имен групп. Например, вы можете зарегистрироваться как MarkHamlon и быть членом нескольких групп, включая группы **Пользователи, Администраторы, ОтделПродаж** и т. д. Это дает возможность администратору изменять параметры безопасности для отдельного пользователя или для группы людей с помощью единственного правила.

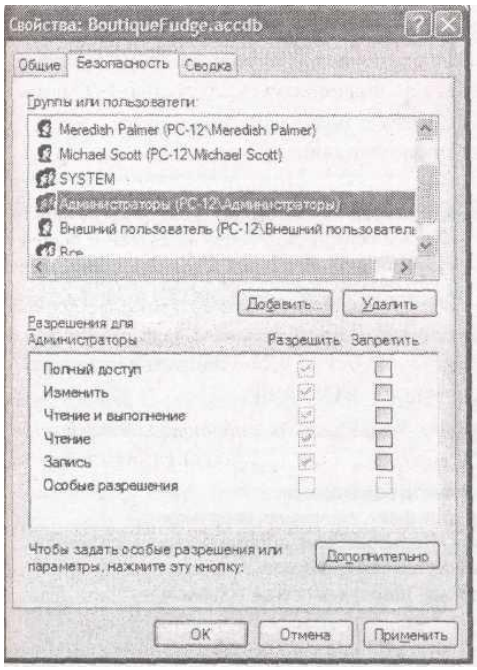


Рис. 18.15. На вкладке Безопасность перечислены все пользователи (и группы), которым разрешено использовать данный файл, и указано, что именно им разрешено делать. В данном примере все имена пользователей и групп начинаются с PC-12, поскольку имя компьютера, на котором определены учетные записи пользователей, - PC-12

3. Для того чтобы изменить для группы или пользователя набор действий над файлом, выберите их в списке и затем измените параметры **Разрешить** (Allow) или **Запретить** (Deny) (рис. 18.16).

Скажем, вы не хотите, чтобы люди из группы **Пользователи** (Users) имели доступ к этому файлу; выделите группу **Пользователи** в списке и установите флажки в столбце **Запретить** (Deny) для каждого разрешения.

Примечание

Параметры в столбце **Запретить** (Deny) всегда обладают более высоким приоритетом. Например, если пользователь - член двух групп и одной группе разрешено использовать файл, а другой нет, параметр **Запретить** (Deny) переопределяет что бы то ни было.

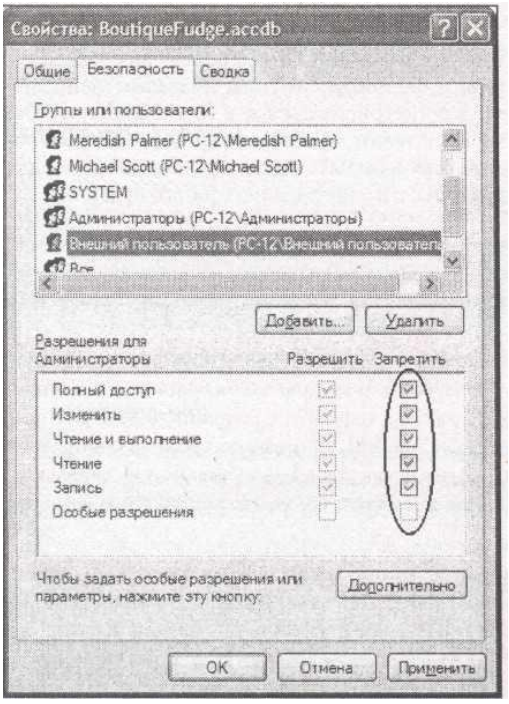


Рис. 18.16. Если флажок тускло-серый - значит, параметр наследуется, т. е. он основан на значении для папки, содержащей этот файл. Например, невозможно изменить параметры в

столбце **Разрешить** для группы **Пользователи,** поскольку они наследуются. Но можно добавить установки в столбец **Запретить** (как показано для пользователя с именем **Внешний пользователь).** Установки в столбце **Запретить** всегда побеждают установки в столбце **Разрешить**

4. Если вы хотите добавить в список нового пользователя или группу, щелкните мышью кнопку **Добавить** (Add), задайте имя пользователя или группы и нажмите кнопку ОК.

Быть может, вы решите заблокировать не всю группу, а выделить конкретного пользователя.

Защита файлов ОС Windows предоставляет стандартный уровень защиты. Она разработана не для работы с БД Access. Для ее более полного использования придется делить вашу БД на все более и более мелкие части, которыми, может быть, трудно управлять. Вы также не можете управлять набором действий, которые разрешены пользователю, - защита файлов либо блокирует пользователей целиком, либо предоставляет им полный контроль над добавлением, удалением, обновлением и реорганизацией информации в вашей БД.

Если нужна действительная защита на уровне пользователя, вам больше подойдет серверный программный продукт, например, SQL Server. Если же просто необходимо помешать доступу к некоторой конфиденциальной информации, средства защиты файлов могут вас выручить.

19. Глава 19. Импорт и экспорт данных

БД Access напоминает тщательно построенный форт. Он берет строго организованную и проверенную на наличие ошибок информацию и умело блокирует ее. Очень немногие программы охраняют свои данные такой броней, как программа управления БД. Текстовые процессоры и программы обработки электронных таблиц принимают почти любое содержимое и позволяют на лету формировать структуру документа. БД далеко не так раскованы.

Большую часть времени БД живут в независимом мире.

Но время от времени всем приходится одним из указанных далее способов преодолевать разрыв:

■ вы хотите взять данные из другой программы и импортировать их - в основном заполнить ими вашу БД;

■ вы хотите взять какую-то информацию из БД Access и экспортировать ее, чтобы можно было работать с этими данными в другой программе.

Для преобразования данных у программы Access есть несколько разных средств. Можно использовать непритязательный буфер, сложные средства импорта и экспорта или неизменно популярный XML-стандарт. В этой главе вы узнаете обо всех имеющихся в вашем распоряжении вариантах, включая один новый и очень искусный метод, позволяющий пользователям послать по электронной почте их обновления для вашей БД. Это уже не та программа Access, с которой работали ваши отцы.

19.1. *Аргументы в пользу экспорта и импорта*

Если вы не слишком задумывались об импорте и экспорте, значит, пока они вам не очень были нужны. Многие БД вполне счастливы, живя тихой уединенной жизнью. Но импорт и экспорт могут пригодиться в некоторых ситуациях. Рано или поздно вы можете оказаться в одной из них.

19.1.1. *Что такое экспорт*

Экспорт - более легкая составляющая уравнения. Операция экспорта проще операции импорта, поскольку она включает перенос информации из строго организованного расположения (БД) в менее строгое (документ другого типа).

Примечание

Экспорт - метод передачи ваших данных в другое место. Исходная копия всегда остается в Access. Нет никакого резона изменять экспортированную копию. Если нужны изменения, внесите их в БД, а затем выполните операцию экспорта еще раз.

Далее перечислены самые частые причины, заставляющие пользователей экспортировать информацию.

■ Вы хотите отправить какие-либо данные вашему другу по электронной почте. Копию БД Access вы отправлять не желаете, потому что у друга нет копии программы Access или вы хотите показать ему только часть данных, а не все.

■ Вы создаете презентацию в программе PowerPoint. Самый легкий способ ослепить и убедить ваших пользователей - продемонстрировать им некоторую впечатляющую информацию из вашей БД.

Подсказка

Программа Access хранит огромные объемы информации, и часто другие программы просто не могут их обработать. Вы никогда не сможете скопировать таблицу в презентацию PowerPoint - в лучшем случае слайд сможет вместить лишь горстку записей. Но можно показать результаты сводного запроса *(см. разд. "Итоговые данные" главы 7),* в котором применяется группировка для получения результатов в виде нескольких промежуточных итогов.

■ Вы хотите анализировать данные в программе Excel. Программа Access отлично подходит

для хранения ваших данных и управления ими, но она не предоставляет средств, помогающих понять, что все это значит. Если вы хотите спрессовать данные с помощью тяжеловесных рабочих формул и воспользоваться средствами построения диаграмм, есть смысл перенести данные в программу Excel.

Некоторые программы обладают интеллектом, достаточным для того, чтобы извлечь данные из БД Access самостоятельно. Например, программа Word, у которой есть средство слияния сообщений электронной почты, позволяющее взять список имен и адресов из БД и затем использовать их для создания почтовых наклеек, персонифицированных форм и разного рода групповых документов. Для применения этого средства не нужен экспорт - достаточно указать в программе Word на файл вашей БД Access.

19.1.2. *Что такое импорт*

Импорт всегда нужен, если есть данные за пределами БД, которые являются ее составляющей. Допустим, вы создаете самую современную БД электронной коммерции для вашей фермы по разведению бизонов. Но некоторые ваши торговые партнеры все еще заполняют формы с помощью старинной электронной таблицы Excel. Вам понадобятся средства для извлечения данных из электронной таблицы Excel и переноса их в вашу БД.

Подсказка

Ваш торговый персонал вас подвел. Им не следовало вводить данные в документ, предназначенный для другой программы. Они должны были бы использовать форму, которая разработана для регистрации продаж, как описано в *главе 12.*

Импорт информации связан с двумя ключевыми проблемами. Первая - обеспечение соответствия данных строгим требованиям БД. Как вы узнали в *главе 1,* БД помешаны на правилах и грубо отбрасывают неподходящие данные (например, текст в поле для даты). Вторая трудность - обработка информации, которая не полностью соответствует, - другими словами, ее представление в БД не соответствует ее представлению во внешнем документе. Эта проблема гораздо распространеннее, чем можно было бы предположить.

В вашей БД могут применяться коды статуса (например, 4302), а в электронной таблице, которую вы хотите импортировать, используются именованные константы (например, High Priority). Или же нужно разделить импортируемую информацию на несколько связанных таблиц, несмотря на то, что она хранится в едином документе. Электронная таблица с заказами клиентов для вашей фермы по разведению бизонов могла включать сведения о клиентах (которые относятся к таблице **Customers)** и данные о заказах (для таблицы **Orders).** К несчастью, нет легких способов решения подобных проблем. Если данные точно не соответствуют представлению в БД, придется исправлять их вручную до или после операции импорта.

Специалисты иногда пытаются решить проблемы, подобные описанным, с помощью программ на Visual Basic, которые читают данные и создают соответствующие записи. (Для этого вам придется использовать объекты DAO, описанные в *разд. "Обновление единиц наличного запаса" главы 17.)* Несмотря на то, что программный подход чрезвычайно гибок, написание кода и его сопровождение быстро превращается в кошмар, поэтому старайтесь избегать его применения изо всех сил.

На профессиональном уровне.

SQL Server и SharePoint: два частных случая

В этой главе не будут рассматриваться две программы.

SQL Server - программное обеспечение для создания мощной серверной БД, которое описывается в *главе 20.* Если ваша БД Access разрастается экспоненциально, вы можете попробовать перенести ваши данные на SQL Server. Но для этого не применяется стандартное средство экспорта. У программы Access есть специальное средство преобразования, которое поможет вам в этом случае. Вы узнаете о нем в *главе 20.*

SharePoint - другой программный серверный продукт промышленного уровня, который хранит большие объемы данных. Но в отличие от SQL Server, SharePoint разработан для того,

чтобы помочь группам пользователей совместно использовать информацию и взаимодействовать друг с другом с помощью внутренних сетей или Web-пространства. Если вы захотите перенести данные в список SharePoint (или из него), придется заглянуть в *главу 21*.

19.2. *Применение буфера обмена*

Любой, кто проводит много времени за компьютером, знаком с буфером обмена - скрытым контейнером, который временно хранит данные, давая возможность переносить их из одной программы в другую. С помощью буфера обмена можно скопировать фрагмент текста в документе Word и затем вставить его в поле таблицы Access или наоборот. Это довольно легко, но, возможно, вы не задумывались о том, что можно скопировать целую таблицу с данными.

Подсказка

Почти все Windows-программы применяют одни и те же комбинации обмена клавиш для работы с буфером. Используйте комбинацию клавиш <Ctrl>+<C> для копирования данных, <Ctrl>+ +<X> для вырезания (т. е. копирования и удаления) и <Ctrl>+<V> для вставки информации.

Прежде чем проверить комбинации клавиш на практике, следует понять два ключевых факта, касающихся буфера обмена.

- Буфер обмена может хранить данные разных типов. Большую часть времени вы применяете его для копирования обычного текста. Но в зависимости от используемой вами программы можно копировать в буфер обмена контуры, рисунки, таблицы и т. д.
- Данные некоторых типов способны самостоятельно преобразовываться в информацию других типов. Если копируется группа ячеек в программе Excel, ее можно вставить как форматированную таблицу в программу текстового процессора, например, Word или WordPerfect. Если копируется диаграмма в программе Visio, ее можно вставить как рисунок в программу Paint. В обоих примерах вы копируете объект специального типа (ячейки Excel или диаграмму Visio) в буфер обмена ОС Windows. Этот объект способен упростить себя, если в этом есть необходимость. В исходную программу вы сможете вставить полнофункциональную копию объекта без потерь, а в менее мощную программу его можно вставить, преобразовав в нечто более простое.

Эта гибкость - главный секрет передачи данных из программы Access и в нее. В следующих разделах объясняется, как это делается.

Примечание

Применение буфера обмена - более простой метод, чем операции импорта и экспорта программы Access. В результате это более быстрый вариант (состоящий из нескольких шагов). Конечно, он предоставляет меньше возможностей, да и работает не со всеми программами.

19.2.1. *Копирование таблицы из программы Access*

Программа Access позволяет копировать набор строк или целую таблицу в другую программу без применения мастера экспорта. Access копирует выбранные строки в буфер обмена как объект с развитой логикой, способный преобразовать себя в разные программные форматы. Его можно вставить как ячейки Excel, HTML-текст (язык форматирования, применяемый в Web-пространстве) или RTF-документ (стандарт форматирования, разработанный корпорацией Microsoft и поддерживаемый основными текстовыми процессорами). I Поскольку форматы HTML и RTF поддерживаются многими программами, у вас практически никогда не будет проблем при копировании данных в другую программу с их помощью.

Вот как это делается.

1. Если нужно скопировать целую таблицу, выделите ее в области переходов. Если хотите скопировать только несколько строк, выделите их в Режиме таблицы, как показано на рис. 19.1.

Вы не ограничены копированием только таблиц. Можно скопировать результаты запроса. Просто выделите запрос в области переходов. Но формы или отчеты скопировать не удастся.

При копировании строк или всей таблицы программа Access учитывает параметры скрытия

столбцов *(см. разд. "Скрытие столбцов" главы 3)*. Если вы скрыли столбец, чтобы он не отображался на листе данных (с помощью его выделения и последующего выбора команд Главная → Записи → Дополнительно → Скрыть столбцы (Home → Records → More → Hide Columns)), Access не будет копировать его в буфер обмена. Этот прием помогает отбросить данные, которые вы не хотите копировать.

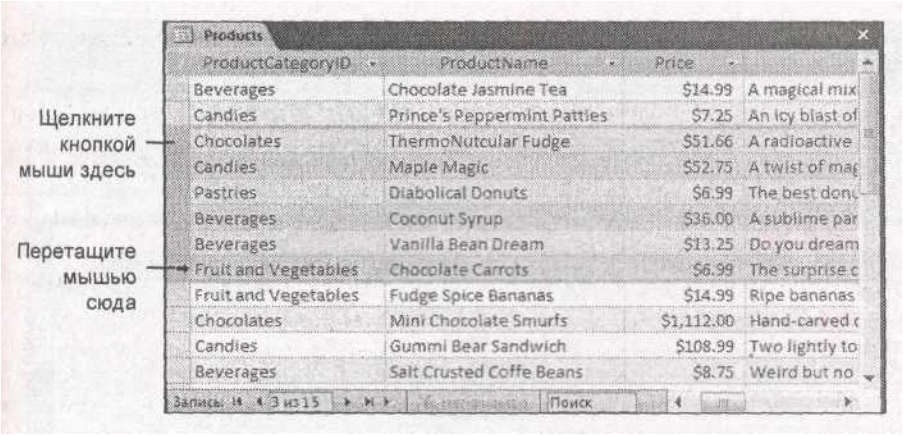


Рис. 19.1. При выделении строк на листе данных щелкните мышью серый отступ слева у первой из строк, которые хотите выделить. Далее с нажатой кнопкой мыши сместите ее указатель вниз для выделения нужного числа строк. Если не хотите отрывать руку от мыши, можно скопировать их, если нажать и удерживать клавишу <Ctrl> и щелкнуть правой кнопкой мыши одну из выделенных строк. Далее из раскрывающегося меню можно выбрать команду **Копировать**

Примечание

Можно скопировать лишь непрерывную область выделения, что означает возможность копирования только смежных строк, расположенных рядом друг с другом. Если в вашей таблице 10 строк, можно скопировать строки с третьей по шестую, но нельзя скопировать первую и последнюю строки. (Конечно, можно копировать более мелкими порциями, чтобы скопировать оторванные от общего выделения строки.)

2. Нажмите комбинацию клавиш <Ctrl>+<C> для копирования выделенных строк.

Это действие помещает строки в буфер обмена ОС Windows. Теперь их можно вставить в программу Access или другую программу.

3. Перейдите в программу, в которую хотите вставить информацию. Если вы прямо сейчас проверяете это средство в первый раз, воспользуйтесь программой Excel или Word (рис. 19.2).

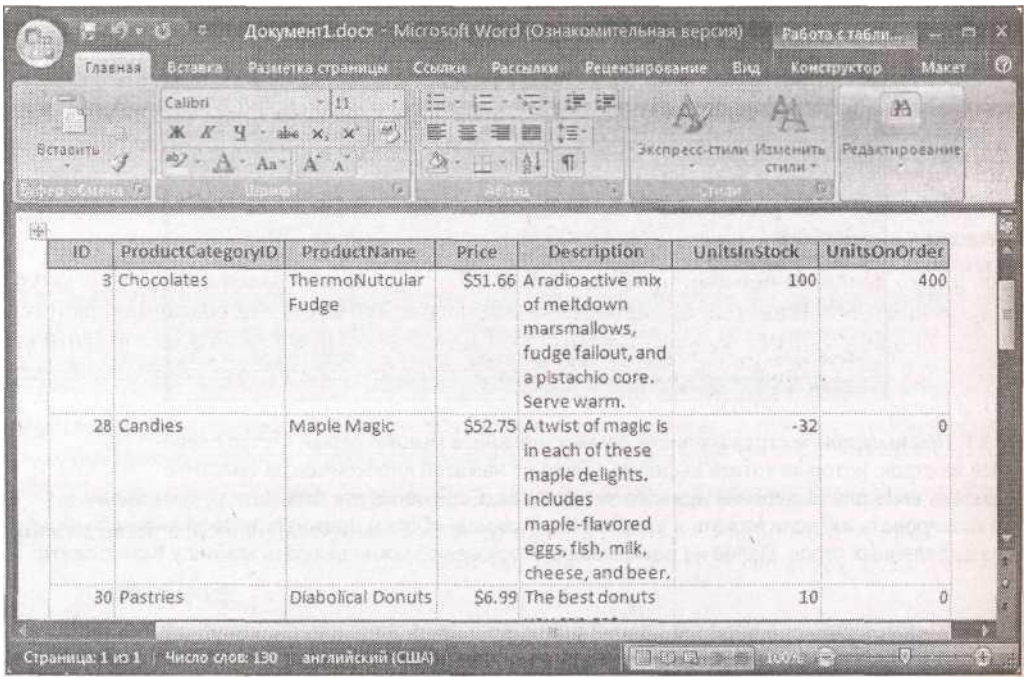
4. Нажмите комбинацию клавиш <Ctrl>+<V> для вставки выделенных строк (см. рис. 19.2).

Программа Access вставляет выделенные вами строки, снабжая их заголовками. Если на листе данных применялось форматирование *(см. разд. "Форматирование листа данных" главы 3),* большая его часть переносится.

В некоторых программах, в которые вставляются записи, можно увидеть пиктограмму смарт-тега, появляющуюся в правом углу только что вставленного содержимого. В приложениях пакета Office можно применять этот смарт-тег для изменения параметров вставки данных (например, с форматированием или без).

Примечание

Копировать текст, числа и даты легко. Но данные некоторых типов плохо поддаются переносу. Если скопировать поле типа **Вложение,** во вставленном контенте отобразится число вложений, но сами файлы будут отсутствовать.



ID	ProductCategoryID	ProductName	Price	Description	UnitsInStock	UnitsOnOrder
3	Chocolates	ThermoNutcular Fudge	$51.66	A radioactive mix of meltdown marsmallows, fudge fallout, and a pistachio core. Serve warm.	100	400
28	Candies	Maple Magic	$52.75	A twist of magic is in each of these maple delights. Includes maple-flavored eggs, fish, milk, cheese, and beer.	-32	0
30	Pastries	Diabolical Donuts	$6.99	The best donuts	10	0

Рис. 19.2. С помощью копирования или вырезания можно преобразовать таблицу БД в таблицу документа Word, показанную здесь. После вставки содержимого, возможно, придется подкорректировать ширину столбцов, для того чтобы все выглядело как следует

Сберегающая время подсказка.

Копирование из одной БД в другую

Способ копирования, описанный только что, можно применить для копирования данных из одной БД Access в другую БД Access, которая открыта в отдельном окне. Но этот прием сработает, только если копируется вся таблица (или другой объект), а не набор строк.

Для опробования данного способа в области переходов щелкните правой кнопкой мыши нужный вам объект и затем выберите команду **Вставить** (Paste). Access запросит имя вставляемой таблицы и предложит три варианта вставки.

- Вариант **только структура** (Structure) создает табличную структуру, но оставляет ее пустой.
- Вариант **структура и данные** (Structure and Data) создает точный дубликат таблицы со всеми данными.
- Вариант **добавление данных в таблицу** (Append Data to Existing Table) не создает новую таблицу - он добавляет данные в заданную вами таблицу. Для функционирования этого варианта у таблицы должна быть структура, в точности совпадающая со структурой скопированной вами таблицы.

Описанный прием позволяет создать дубль таблицы (или другого объекта) в той же самой базе.

19.2.2. *Копирование ячеек из Excel в Access*

Скопировать данные из программы Access в другую программу достаточно легко, но вы, вероятно, не рассчитываете на то, что можно выполнить и обратный процесс. Помимо всего прочего БД - это жесткая, строго структурированная коллекция данных. Если попытаться скопировать таблицу из программы текстового процессора, вы столкнетесь с отсутствием жизненно важной информации, например типов данных для каждого столбца. По этой причине Access не разрешит вставку.

Но для столь любимой всеми программы Excel Access делает исключение. Можно скопировать набор ячеек в Excel и затем вставить их в программу Access для создания новой таблицы. Эта процедура действует, потому что Excel различает данные разных типов (хотя она далеко не так придирчива, как Access). Например, Excel по-разному трактует числа, даты, текст и логические значения.

Вот как действует этот метод.

1. В программе Excel выделите ячейки, которые хотите скопировать.

Если у электронной таблицы есть заголовки столбцов, включите их в область выделения.

Программа Access сможет использовать их как имена полей.

Примечание

Неважно, какая у вас версия Excel - этот метод действует во всех версиях программы.

2. Нажмите комбинацию клавиш <Ctrl>+<C>, чтобы скопировать выделенную область.
3. Перейдите в программу Access.
4. Щелкните кнопкой мыши где-нибудь в области переходов и нажмите комбинацию клавиш <Ctrl>+<V>.

Программа Access заметит, что вы пытаетесь вставить группу ячеек Excel, и попытается преобразовать их в таблицу. Сначала она поинтересуется, содержатся ли в первой строке области выделения заголовки столбцов.

5. Если в пункте 1 вы выделили заголовки, щелкните кнопку Да, в противном случае **Нет.**

Если выбран вариант Да, программе Access не нужно создавать случайные имена полей - она может использовать ваши заголовки.

Access создает новую таблицу для работы с новыми данными. Эта таблица названа так же, как таблица Excel. Если у таблицы имя листа **Лист1** (Sheetl) (как у большинства таблиц Excel), теперь у вас есть таблица **Лист1.**

Когда Access закончит вставку, программа выведет на экран подтверждающее сообщение, дающее знать, что все завершилось успешно.

6. Щелкните мышью кнопку ОК.

Теперь можно проверить таблицу, чтобы убедиться в том, что типы данных и имена полей такие, как вы хотели.

19.3. *Операции импорта и экспорта*

Несмотря на то, что метод применения буфера с командами вырезания и вставки очень удобен, он не всегда решает задачу. Если нужно экспортировать данные в файл, а на вашем компьютере не установлена соответствующая программа (или вы просто не хотите суетиться и запускать ее), необходим другой способ передачи ваших данных. Подобным образом, если вы загружаете данные из Всемирной паутины или извлекаете информацию из программы, не поддерживающей метод ОС Windows вырезания и вставки, вам потребуется полнофункциональное средство импорта программы Access.

Когда корпорация Microsoft разрабатывала программу Access 2007, было потрачено много времени на то, чтобы сделать средства импорта и экспорта яснее и понятнее. Теперь вы можете выполнять все операции импорта и экспорта с помощью одной вкладки ленты, названной **Внешние данные** (External Data) (рис. 19.3).

Рис. 19.3. Группа **Импорт** вкладки ленты **Внешние данные** позволяет передать данные в программу Access с помощью разнообразных форматов. Группа **Экспорт** выполняет обратные действия и экспортирует данные в группу различных форматов

Примечание

В группах **Импорт** (Import) и **Экспорт** (Export) есть легко доступные кнопки для большинства наиболее популярных форматов. Если вы не видите нужного формата, щелкните мышью кнопку **Дополнительно** (More), чтобы вывести на экран расширенный список форматов.

Импортируете вы данные или экспортируете их, процесс один и тот же. Вы отвечаете на

несколько вопросов об используемом файле и способе преобразования данных, и затем про грамма Access выполняет ваши распоряжения.

После того как операция импорта или экспорта закончена, Access предоставляет возможность сохранить все шаги этого процесса. Если вы сделаете это, то позже сможете их применить повторно *(см. разд. "Повторное применение параметров импорта и экспорт далее в этой главе).* Этот метод позволяет сэкономить массу времени, если вам придется повторить тот же процесс экспорта или импорта еще раз (например, если нужно импортировать некоторые данные каждый день или экспортировать итог в конце каждого месяца).

19.3.1. *Импортируемые типы файлов*

Чаще всего вы будете импортировать данные одного из следующих пяти распространенны форматов.

■ *Access.* Когда используется этот вариант, никакое преобразование не выполняется. Вы берете объект БД из другого файла БД Access и копируете его в текущую БД. Этот вариант применялся в *главе 18,* когда создавалась клиентская БД.

■ *Excel.* Извлекаются данные из электронной таблицы Excel.

■ *Список SharePoint.* Добываются данные из списка, расположенного на сервере SharePoint. Для работы с данными SharePoint их не нужно импортировать. Редактировать списки SharePoint можно непосредственно в программе Access. В *главе 21* вы найдете дополнительную информацию о совместной работе Access и SharePoint.

■ *Текстовый файл.* Извлекаются данные из обычного текстового файла. Как правило, в текстовых файлах для разделения значений полей используются символы определенного типа (например, запятая). Этот всегда понятный формат поддерживают многие программы, включая почти все варианты когда-либо написанных программ электронных таблиц. Когда применяется этот формат, Access просматривает текстовый файл и пытается выяснить его организацию. У вас есть возможность согласиться с предположениями программы на этот счет или откорректировать их, прежде чем импортировать какие-либо данные.

■ *XML-файл.* Получаются данные из структурированного XML-файла (Extensible Markup Language, расширяемая спецификация языка, предназначенного для создания Web-страниц). XML - совместимый с разными платформами формат, используемый для представления информации любого типа. Но вам не удастся успешно импортировать любые XML-файлы - для того чтобы операция импорта могла завершиться успешно, в XML-файле должна применяться структура, подобная табличной.

Если воспользоваться кнопкой **Дополнительно** (More), можно обнаружить некоторые другие, более экзотические форматы импортирования.

■ *База данных ODBC* (ODBC Database). Получается информация практически из любой БД при условии, что у нее есть драйвер ODBC (Open DataBase Connectivity, открытый интерфейс доступа к базам данных). Этот вариант особенно хорош, если необходимо получить данные из высокопроизводительных серверных БД, например, Oracle, SQL Server или MySQL.

■ *Документ HTML* (HTML Document). Извлекается информация из списка или таблицы, размещенных на Web-странице. Поскольку стандарт HTML (HyperText Markup Language, язык разметки гипертекста) печально известен как слабый (а временами откровенно сырой), лучше избегать его применения. Вполне вероятно, что при импорте вы столкнетесь с проблемами.

■ *Папка Outlook* (Outlook Folder). Добываются данные из папки программ Outlook или Outlook Express.

■ *Файл dBase* (dBase File), *Файл Paradox* (Paradox File), *Файл Lotus 1-2-3* (Lotus 1-2-3 File). Извлекается информация из файла, созданного в одной из этих программ эпохи палеолита.

19.3.2. *Импорт данных*

Независимо от того, какой тип данных вы хотите импортировать, вам придется выполнить одни и те же действия.

1. В группе ленты **Внешние данные → Импорт** (External Data → Import) щелкните мышью кнопку, соответствующую типу файла, который вы хотите импортировать.

Когда формат выбран, запускается мастер импорта **Внешние данные** (рис. 19.4).

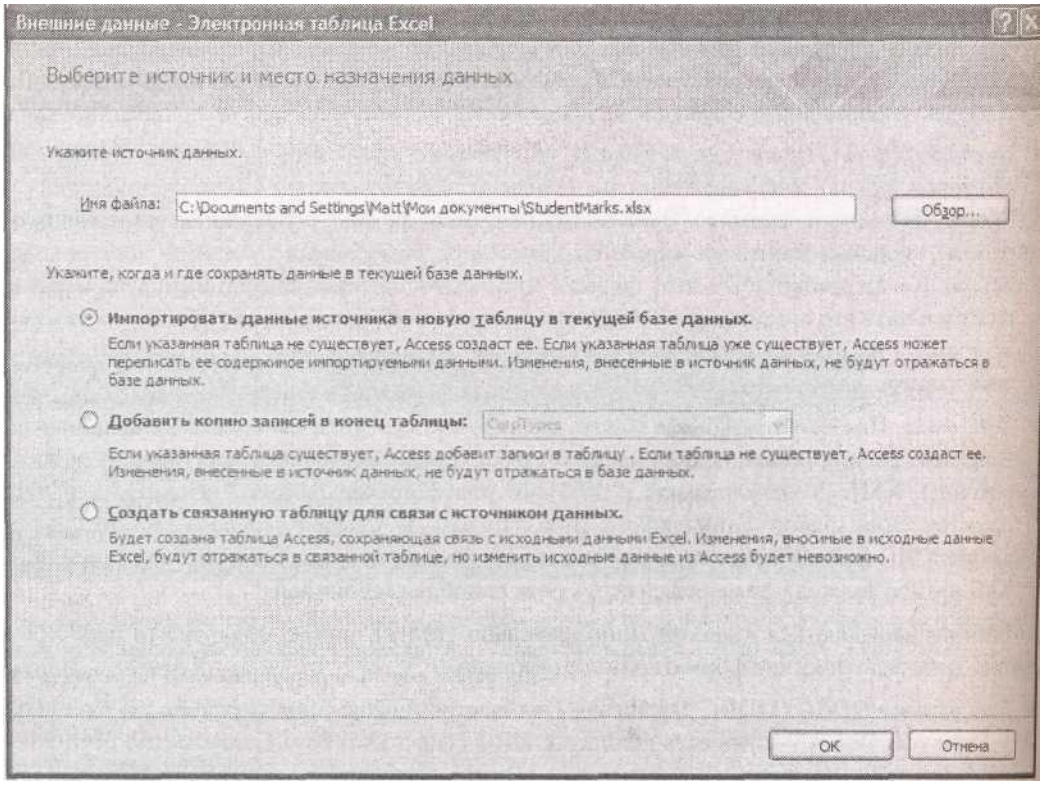


Рис. 19.4. Независимо от выбранного формата мастер импорта почти один и тот же, хотя определенные параметры могут быть ограничены. На первом этапе выбирается имя файла-источника и способ вставки информации в вашу БД программой Access

2. Введите имя файла, который хотите импортировать.

Если вы не помните путь к файлу (или не хотите набирать его вручную), щелкните мышью кнопку **Обзор...** (Browse...) и затем перейдите в нужное место в окне Открытие **файла** (File Open). После того как файл найден, дважды щелкните его кнопкой мыши.

3. Выберите, куда помещать в вашей БД импортированные данные.

У вас есть три возможных варианта размещения данных. Не для всех файловых форматов импорта все они доступны.

- **Импортировать данные источника в новую таблицу в текущей базе данных** (Import the source data into a new table in the current database). Этот вариант создает новую таблицу для импортируемых данных, что убережет вас от головной боли из-за беспокойства о конфликтующих записях. Но если имя таблицы совпадает с именем уже существующей в БД Access таблицы, этот вариант стирает последнюю.
- **Добавить копию записей в конец таблицы** (Append a copy of the records to the table).
- Данный вариант добавляет импортируемые вами строки в существующую таблицу, Для успешного действия этого варианта нужно, чтобы структура импортируемых данных совпадала со структурой таблицы, которую вы используете. Например, имена полей должны точно совпадать. В импортируемых данных могут быть пропущены необязательные поля и значения по умолчанию.
- **Создать связанную таблицу для связи с источником данных** (Link to the data source by creating a linked table). Если применяется этот вариант, программа Access на самом деле не переносит информацию **с** вашу БД. Вместо этого при каждом просмотре связанной таблицы Access проверяет исходный файл для получения самой свежей информации. Самое замечательное заключается в том, что в связанной таблице всегда отображается новейшая информация. В любом другом варианте импортированная таблица остается нетронутой, если изменяется исходный файл. Но связанные таблицы тоже опасны, поскольку у вас нет гарантий того, что файл не перекочует в другое место на вашем жестком диске (где программа Access не сможет его найти). Вы применяли связанные таблицы для создания разделенной БД в *главе 18.*

Примечание

Связанные таблицы - хороший способ преодолеть разрыв между разными БД Access или другими БД (например, SQL Server). Но они плохо работают с более ограниченными форматами, такими как текстовый файл.

4. Щелкните мышью кнопку ОК.

Запустится мастер, который соберет оставшуюся информацию, необходимую программе Access. Если импортируется файл Excel, Access запрашивает, какую электронную таблицу использовать. Если вы импортируете текстовый файл, Access запрашивает, как разделены поля в файле.

5. Ответьте на все вопросы в мастере, чтобы сообщить программе Access все необходимые ей сведения о структуре импортируемых данных.

Когда этот этап закончен, Access задает последний вопрос - нужно ли сохранить шаги импорта.

6. Если вы хотите выполнять тот же импорт снова в дальнейшем, установите флажок **Сохранить шаги импорта** (Save import steps). Затем щелкните мышью кнопку **Закрыть** (Close).

В *разд. "Повторное применение параметров импорта и экспорта" далее в этой главе* показано, как повторно использовать сохраненную операцию импорта.

Примечание

Если в процессе импорта программа Access обнаруживает какие-то ошибки, она создает еще одну таблицу с тем же именем, что и у таблицы, в которую вы импортируете данные, с присоединенной в конец имени добавкой _ОшибкиИмпорта (_ImportErrors). Access вставляет в эту таблицу по одной записи на каждую проблему. Если вы пытаетесь импортировать группу данных в таблицу с именем **SalesData,** и программа Access не может преобразовать значения в данные нужного вам типа (например, в столбце, который должен содержать только числа, присутствует текст), вы получаете таблицу, названную **SalesData_ОшибкиИмпорта.**

В следующих разделах вы познакомитесь с особенностями двух распространенных форматов данных, которым для импорта потребуются дополнительные шаги: рабочих книг Excel и текстовых файлов.

На профессиональном уровне.

Опасность дубликатов

Если в процессе импорта записи добавляются в существующую таблицу (добавляются в конец таблицы), вас подстерегает опасность, самый ужасный ночной кошмар любого импортера, - дублирование.

Все очень просто - у программы Access нет способа сообщить о том, импортировались ли эти данные ранее или нет. Если вы установили в программе Access автоматическое заполнение в каждой записи значения **Код** (ID) с типом Счетчик, она запросто может вставить одни и те же данные несколько раз, каждый раз задавая другое значение поля Код (ID). С другой стороны, если не использовать автоматически генерируемые значения в поле **Код** (ID), а импортируемые данные содержат первичный ключ, программа Access вообще не сможет импортировать новые данные. Ясно, что оба варианта далеки от идеала.

Если вы связаны с импортом надолго, единственное решение - быть внимательным. Далее приведено несколько советов.

- Если вы хотите повторно использовать файл после того, как вы уже импортировали содержащиеся в нем данные, убедитесь в том, что вы удалили из файла всю информацию, импортированную ранее.
- Если вам кажется, что вы могли импортировать одни и те же данные дважды, примените для проверки запрос. Можно создать собственный запрос или использовать запрос типа **Повторяющиеся записи** (Find Duplicates), создаваемый Мастером запросов *(см. разд. "Создание простого запроса с помощью Мастера запросов" главы 6).*

- Лучше чаще вносить мелкие изменения, чем реже выполнять более крупные обновления. В этом случае вы быстрее и гораздо легче обнаружите ошибки.
- Если вам нужно более надежное решение, необходимо создать его самостоятельно. Можно воспользоваться программным кодом на Visual Basic для того, чтобы управлять способом передачи данных в программе Access (вас ждет много дополнительной работы).

19.3.3. *Импорт из файла Excel*

Для импорта из файла Excel данные должны быть организованы в базовую таблицу. В идеале заголовки столбцов этой таблицы должны совпадать с именами полей в вашей БД. Нужно удалить любые данные, которые вы не хотите импортировать (как и ячейки под таблицей, не входящие в нее). Необходимо также удалить значения, вычисляемые с помощью формул Excel. (Как вы узнали из *разд. "Правило 5. Избегайте избыточной информации" главы* 5, в таблице не следует хранить вычисляемые значения, поскольку они повышают риск возникновения противоречивых данных.)

Примечание

Ранее в этой главе вы узнали, как обычным образом вырезать данные Excel и вставить их в таблицу Access. Но когда выполняется полнофункциональный импорт, у вас появляется возможность изменить имена полей, настроить типы данных и применить индексирование.

После того как таблица данных в файле Excel вычищена, вы готовы к запуску операции импорта.

1. Выберите на ленте **Внешние данные → Импорт → Excel** (External Data → Import → Excel) и задайте способ вставки импортируемой информации в вашу БД. Затем щелкните мышью кнопку ОК.

Вы узнали, как выбрать способ вставки данных в пунктах 1-3 в предыдущем разделе.

2. Выберите рабочую книгу Excel, в которой содержатся ваши данные (рис. 19.5).

Файлы или рабочие книги Excel вначале обычно состоят из трех листов. Большинство пользователей плюхают свои данные на первый из них, который первоначально назван **Лист1** (Sheetl). Если вы хорошо знакомы с программой Excel, возможно, вы обозначили как именованный диапазон некоторую область в более сложной таблице. Если так, ваш именованный диапазон можно выбрать из предлагаемого списка.

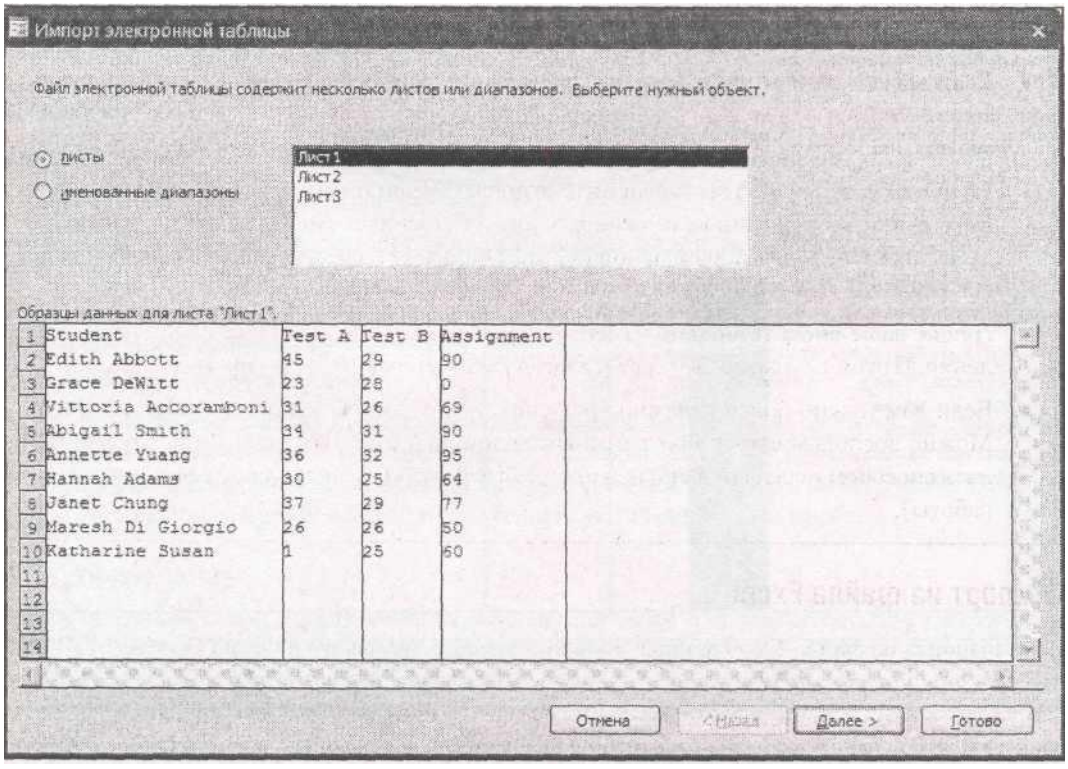


Рис. 19.5. Этот файл рабочей книги Excel содержит три стандартных листа: **Лист1, Лист2** и **Лист3.**

Когда данные выделены, их можно увидеть на экране в области предварительного

просмотра

3. Щелкните мышью кнопку **Далее.**

4. Если первая строка ваших данных Excel содержит заголовки, выберите **Первая строка содержит заголовки столбцов** (First Row Contains Column Headings).

Эти заголовки станут отправной точкой для именования ваших полей. Если не выбрать вариант **Первая строка содержит заголовки столбцов,** программа Excel интерпретирует первую строку как обычную запись.

5. Щелкните мышью кнопку **Далее.**

Если для импортируемых записей создается новая таблица, программа Access попросит более точно определить создаваемые поля. Если записи добавляются в существующую таблицу, переходите к пункту 7.

6. Для каждого поля можно выбрать имя, тип данных и наличие или отсутствие индексирования. Затем щелкните мышью кнопку **Далее.**

Программа Access строит кое-какие догадки, основываясь на полученных данных, но тонкая настройка деталей целиком лежит на вас (рис. 19.6).

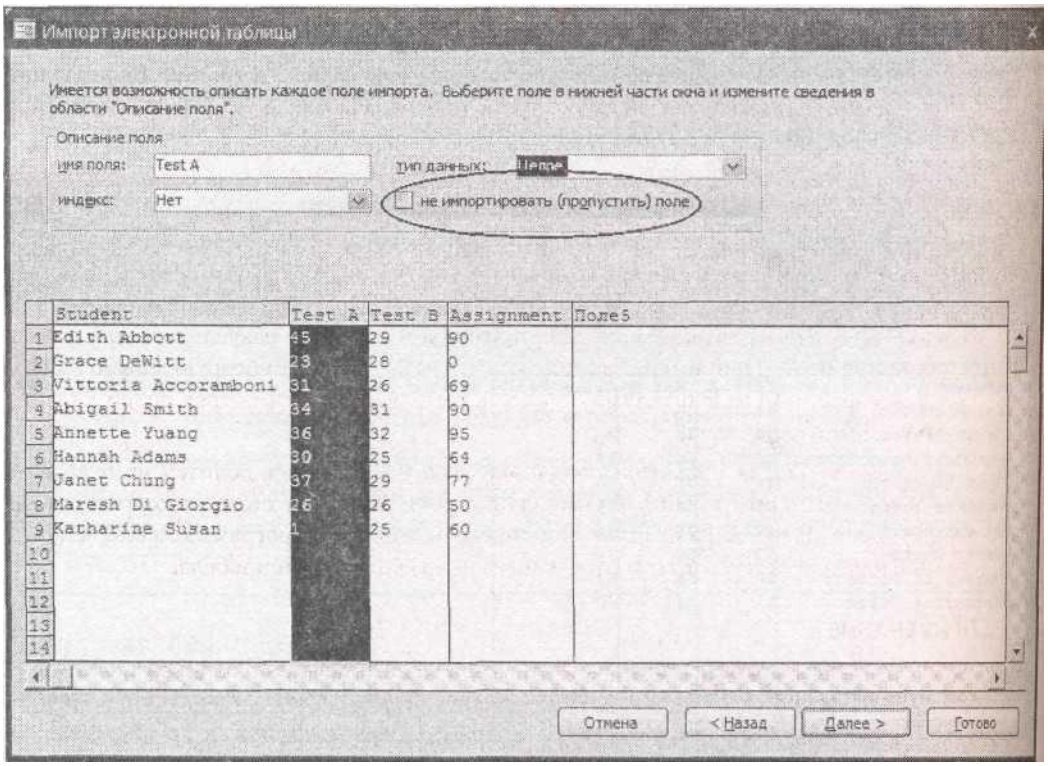


Рис. 19.6. Для определения поля выберите его в области предварительного просмотра и затем задайте параметры. Если вы решили совсем не импортировать поле, установите флажок **не импортировать (пропустить) поле** (обведен), чтобы полностью его игнорировать

7. Решите, хотите ли вы, чтобы программа Access создала первичный ключ.

Выберите вариант **автоматически создать ключ** (Let Access add primary key) для создания поля **Код** (ID) с типом **Счетчик** (что всегда хорошая мысль). Если импортируемые данные уже содержат поле, которое вы хотите использовать как ключ, выберите вариант **определить ключ** (Choose my own primary key) и затем укажите нужное поле.

8. В текстовое поле **Импорт в таблицу** (Import to Table) введите имя таблицы, которую хотите создать или в которую хотите добавить записи.

9. Для завершения выбора нажмите кнопку **Готово** (Finish).

После завершения импорта можно выбрать вариант сохранения шагов этой операции для повторного ее использования.

При импорте данных из Excel могут обнаружиться потенциально ошибочные блоки. Пустые значения и поля - самая распространенная проблема, когда мастер импорта принимает за данные часть электронной таблицы, не содержащей никакой информации. (Это может произойти, если где-то в электронной таблице есть ячейка, содержащая просто пробел или

использовавшаяся для хранения данных, которые с тех пор были уже удалены.) После выполнения импорта для устранения подобных проблем, возможно, придется почистить таблицу, удалив пустые поля и записи.

19.3.4. *Импорт из текстового файла*

Текстовые файлы - "наименьший общий знаменатель" для обмена данными. Если вы применяете программу, создающую файлы, которые программа Access не может импортировать, возможно, обычный текст - единственный выход для вас.

И снова начинайте с выбора файла, а затем укажите, как добавлять информацию в вашу БД. Далее мастер импорта выполнит несколько дополнительных действий.

1. Задайте тип текстового файла.

Программа Access может импортировать текстовые файлы двух типов.

□ В *текстовых файлах с разделителями* применяется особый разделитель для обозначения конца поля. Например, Джо, Пискепоун, 43 - строка, которую можно найти в текстовом файле с разделителями, представляет собой значения трех полей, отделенные друг от друга запятой.

□ В *текстовых файлах с фиксированной шириной полей* запись делится на отдельные поля в соответствии с позицией символа в строке. Каждому полю отводится определенное число символов, и если вы заполняете не все поля, программа Access заполняет оставшееся пустое место (до следующего поля) символами пробела.

Примечание

Текстовые файлы с разделителями используются чаще и обладают большей гибкостью по сравнению с текстовыми файлами с фиксированной шириной полей (поскольку они содержат значения данных с самой разной длиной).

2. Щелкните мышью кнопку Далее.

Если импортируется текст с разделителями, программа Access запрашивает символ-разделитель - другими словами, какой символ отделяет поля друг от друга (рис. 19.7). Самые распространенные разделители - запятая и символ табуляции.

Если вы импортируете текст фиксированной ширины, Access разрешает указать границы полей перетаскиванием разграничительных линий столбцов в нужную позицию в окне предварительного просмотра.

3. Завершите мастер.

Далее мастер выполняет те же шаги, что и в случае импорта данных из программы Excel.

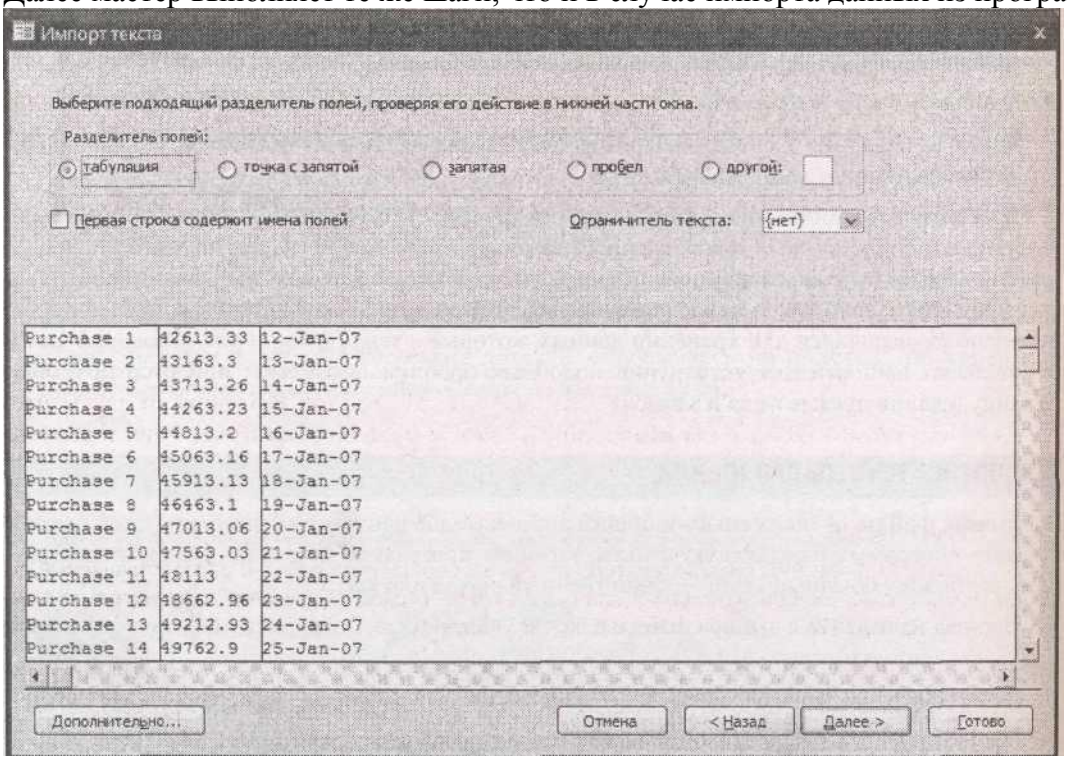


Рис. 19.7. В данном примере поля разделены знаком табуляции

Если создается новая таблица для хранения импортируемых вами данных, на следующем этапе вам придется настроить поля, которые хотите создать, указав их имена, типы данных и вариант индексирования (см. рис. 19.6). Когда этот этап пройден, можно решить, создавать или нет поле **Код** (ID) с типом **Счетчик** и затем использовать его в качестве первичного ключа.

Наконец, на последнем этапе необходимо ввести имя таблицы, которую хотите создать или в которую хотите добавить данные. Затем можно щелкнуть мышью кнопку Готово (и при желании выбрать вариант сохранения шагов импорта для последующего повторного применения).

19.3.5. *Экспортируемые типы файлов*

Также как можно импортировать информацию из других файлов и включить ее в вашу БД, можно взять имеющиеся данные и экспортировать их в другой формат. Чаще всего вам придется выполнять эту операцию, чтобы разрешить другому пользователю или программе воспользоваться вашей информацией без обращения к программе Access.

При экспорте ваших данных можно применять все те же форматы, которые использовались в операции импорта, плюс несколько дополнительных.

- *Access*. Преобразует таблицу Access (или объект другого типа) в другой файл БД Access. Это средство гораздо слабее импорта объектов Access, т. к. можно экспортировать только один объект за другим, поочередно. По этой причине данный формат применяется нечасто.
- *Excel*. Помещает данные в ячейки электронной таблицы Excel. Идеален, если нужно воспользоваться средствами программы Excel для анализа тенденций объемов продаж или построить диаграмму доходов.
- *Word*. Переносит данные в документ Word, отделяя каждый столбец знаком табуляции, а каждую строку символом перевода каретки. Этот формат оставляет желать лучшего, т. к. трудно реорганизовать данные после их переноса в программу Word. (Гораздо удобнее могло бы быть средство экспорта, помещающее данные отчета в таблицу Word, с которой было бы гораздо легче работать.)
- *PDF* или *XPS*. Формируется готовый к печати PDF-файл с жестким форматом и макетом, который можно увидеть при печати таблицы на принтере. В отличие от документов Excel или Word PDF-файл нельзя редактировать - можно просмотреть отчет и распечатать его.

Примечание

Формат PDF или XPS появляется, только если вы установили свободно распространяемый дополнительный модуль надстройки для пакета Office. В *разд. "Получение дополнительного модуля "Save As PDF"" главы 10* описано, как его получить.

- *Документ HTML*. Создает подготовленную для Всемирной паутины Web-страницу, которую можно поместить на Web-сайт или в интранет-сети компании. Формат HTML, генерируемый Access, выглядит удивительно, как реальный отпечатанный отчет.
- *Текстовый файл*. Выгружает данные в обычный текстовый файл с табуляциями и пробелами, применяемыми для организации данных. Теряются цвета, шрифты, рамки и другие детали форматирования. Этот формат не очень полезен - воспринимайте его как последнее средство для передачи данных в другую программу, если никакие другие варианты экспорта не срабатывают.
- *XML-файл*. Сохраняет данные в текстовом файле с расширением xml без форматирования. Этот вариант имеет смысл использовать, если применяется автоматически выполняемая программа, способная читать экспортированный XML-файл и обрабатывать данные.

19.3.6. *Экспорт данных*

Для выполнения операции экспорта выполните следующие действия.

1. В области переходов выделите таблицу, которую хотите экспортировать.

К сожалению, нельзя экспортировать несколько таблиц одновременно. Но можно экспортировать только часть таблицы. Один из способов экспорта фрагмента таблицы - открыть

таблицу и затем выбрать строки, которые хотите экспортировать. (Когда процесс экспорта начнется, вы увидите вариант, позволяющий экспортировать только выделенные строки.) Можно также создать запрос, который извлекает только нужные вам строки. Можно экспортировать результаты запроса, выделив в области переходов запрос, а не лежащую в его основе таблицу.

2. Щелкните мышью кнопку, соответствующую типу файла, в который вы хотите экспортировать данные.

Когда формат выбран, программа Access запускает мастер экспорта (рис. 19.8).

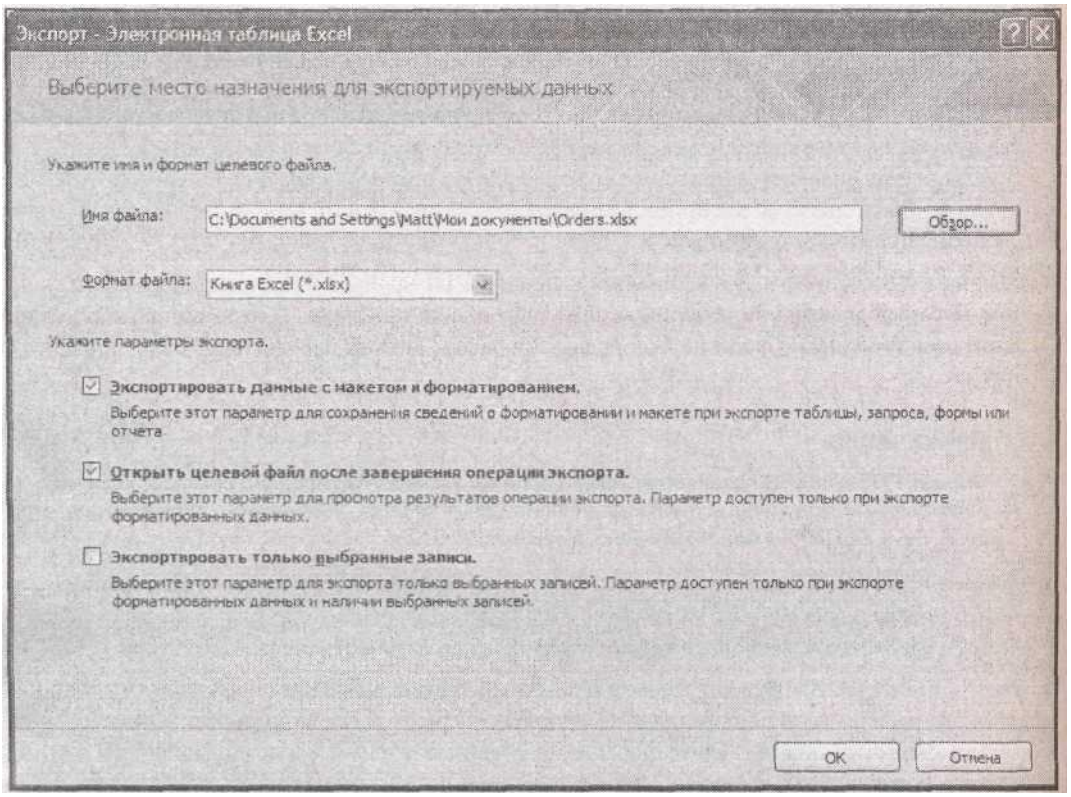


Рис. 19.8. Мастер экспорта изменяется в зависимости от используемого формата экспорта. Но первый шаг всегда заключается в выборе файла и задании параметров экспорта, показанных на рисунке

3. Введите имя файла, который хотите создать.

Программа Access создает этот файл во время выполнения операции экспорта. В некоторых случаях у вас есть возможность выбрать формат файла. Например, если данные экспортируются в программу Excel, можно использовать более новый, основанный на языке XML, формат электронной таблицы (xlsx-стандарт), или более старый xls-стандарт, поддерживаемый более ранними версиями программы, например, Excel 97.

4. Если хотите сохранить форматирование, используемое в вашей БД, установите флажок **Экспортировать данные с макетом и форматированием** (Export data with formatting and layout).

Если вы украсили таблицу привлекательными шрифтами и цветами, программа Access сохраняет эти детали в экспортированном файле. Ясно, что этот вариант работает не со всеми форматами. Например, простые текстовые файлы не могут обрабатывать никакое форматирование.

5. Если хотите дважды проверить экспортированный документ, установите флажок **Открыть целевой файл после завершения операции экспорта** (Open the destination **file** after the export operation is complete).

Никогда не вредно убедиться в том, что вы получили те данные и форматирование, на которые рассчитывали. Если применить этот вариант, программа Access запускает экспортированный файл, открывая его в программе, которой он принадлежит (Excel для электронных таблиц, Блокнот для текстовых файлов и т. д.). Конечно, этот вариант действует, если у вас на компьютере установлена соответствующая программа.

6. Если вы выделили только несколько записей в таблице, установите флажок **Экспортировать только выбранные записи** (Export only the selected records).

В этом случае программа Access экспортирует только текущее выделение, а не всю таблицу или запрос.

7. Щелкните мышью кнопку ОК для выполнения операции экспорта.

Программа Access может запросить дополнительные подробности, если ей нужно больше сведений для определения способа создания экспортируемого файла.

После завершения описанного этапа Access задает последний вопрос - сохранять или нет шаги экспорта.

8. Если в дальнейшем вы хотите выполнять эту операцию экспорта повторно, установите флажок **Сохранить шаги экспорта** (Save export steps). Затем щелкните мышью кнопку **Закрыть.**

В следующем разделе объясняется, как воспользоваться сохраненной операцией экспорта.

Малоизвестная или недооцененная возможность.

Экспорт отчетов

Экспортировать можно не только таблицы и запросы. Программа Access позволяет экспортировать и отчеты. Если выбрать вариант сохранения форматирования и макета, Access попытается сделать так, чтобы экспортируемый файл выглядел как напечатанный отчет.

Этот вариант прекрасен, если вы хотите переслать отчет кому-либо, не имеющему программы Access. Если нужно просто предоставить данные для коллективного использования, можно использовать формат Word. Если вы хотите сохранить форматирование точно таким, как при печати, которую можно выполнить позже, больше смысла имеет формат PDF. В *разд. "Экспорт отчета" главы 10* обсуждается, как экспортировать отчет со всеми подробностями.

Программа Access также позволяет экспортировать форму, но результаты могут оказаться не такими, как вам хотелось бы. Access применяет форматирование и макет из **Режима таблицы** (Datasheet view). Большинство форм использует аккуратно расположенный набор элементов управления в **Режиме формы** (Form view) и редко применяет **Режим таблицы.** Но в процессе экспорта формы Access полностью игнорирует **Режим формы.**

19.3.7. *Повторное применение параметров импорта и экспорта*

В некоторых ситуациях оказывается необходимым регулярное повторение операций импорта и экспорта. Вам может понадобиться перенос данных из электронной таблицы Excel в вашу БД один раз в неделю. Или потребуется создавать ежемесячно отчет с итогами продаж в формате PDF. В этих случаях можно сэкономить массу времени, если полностью выполнить мастер. Это особенно справедливо, когда выполняется импорт, потому что вам может понадобиться выбирать столбцы для импорта, задавать подходящие типы данных и затем настраивать другие параметры точно так же, как и в первый раз при выполнении операции импорта.

К счастью, для этих случаев у программы Access есть решение. Можно собрать все параметры, заданные в мастере импорта или мастере экспорта, и сохранить их в вашей текущей БД. Затем, когда понадобится повторить процесс, можно будет применить эти параметры парой щелчков кнопки мыши (без особого напряжения ума).

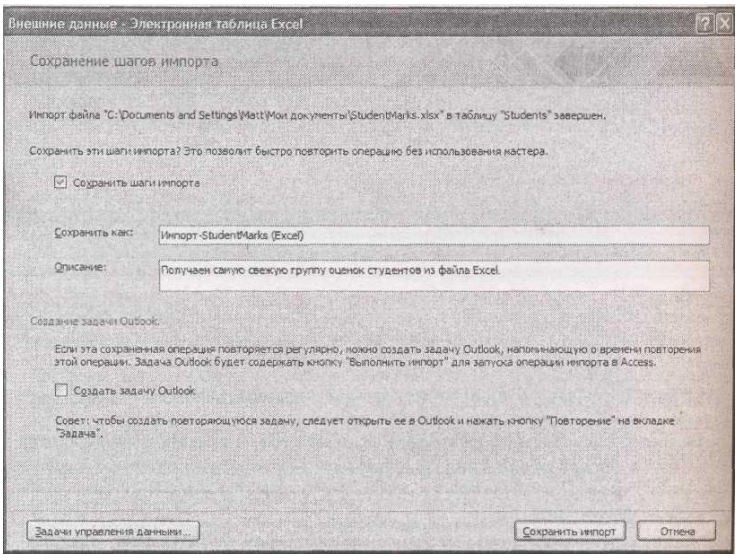


Рис. 19.9. В данном случае процесс импорта сохраняется для повторного использования в дальнейшем. Можно заполнить необязательное описание этой операции, чтобы вы легче вспомнили, что это все значит. Если вы пользуетесь популярной программой электронной почты Outlook корпорации Microsoft, можно отметить флажок **Создать задачу Outlook** и создать автоматическое напоминание, сообщающее о том, что пришло время для вашего импорта или экспорта

Для сохранения этих шагов просто установите флажок **Сохранить шаги импорта** (Save import steps) или **Сохранить шаги экспорта** (Save export steps) в конце выполнения операции, когда ваши данные импортируются или экспортируются в первый раз. Следует подобрать информативное название для ваших параметров, как показано на рис. 19.9, и затем щелкнуть мышью кнопку **Сохранить импорт** (Save Import).

ПОДСКАЗКА

Если сохраняется операция импорта, подумайте, как следует, о выборе новой таблицы или пополнении уже существующей. Если создается новая таблица, то при каждом выполнении импорта программа Access будет перезаписывать таблицу, заменяя ее новой с новыми данными. Если же выбрать вариант дозаписи в конец имеющейся таблицы, Access добавит новые данные к тем данным, которые вы уже получили. (В этом случае нужно следить за появлением дубликатов.)

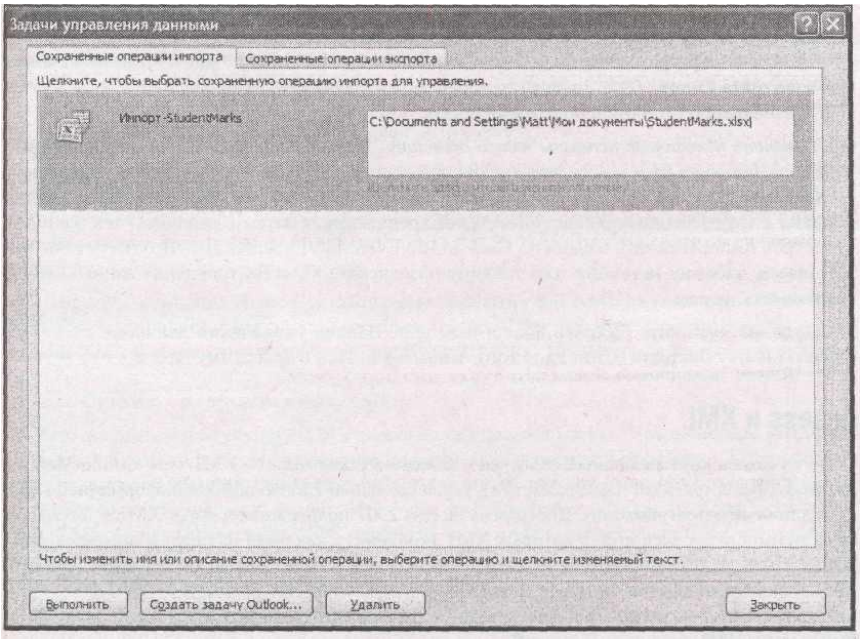


Рис. 19.10. В данном примере сохранена единственная операция импорта. Щелкнув кнопкой мыши имя файла, можно заменить его другим именем или ввести вручную новый путь к файлу

Когда-нибудь в будущем вы сможете перезапустить операцию импорта или экспорта. Если хотите повторить импорт, выберите на ленте **Внешние данные → Импорт → Сохраненные операции импорта** (External Data → Import → Saved Imports). Для повторения экспорта выберите на ленте **Внешние данные → Экспорт → Сохраненные операции экспорта** (External Data → Import → Saved Exports). В любом случае вы попадете в диалоговое окно

Задачи управления данными (Manage Data Tasks) (рис. 19.10) на вкладку **Сохраненные операции импорта** (Saved Imports) или вкладку **Сохраненные операции экспорта** (Saved Exports). На этих вкладках перечислены операции импорта и экспорта, которые вы сохранили в данной БД.

Далее перечислено все, что можно делать в диалоговом окне **Задачи управления данными.**

■ *Повторно выполнить операцию.* Выберите ее в списке и щелкните мышью кнопку **Выполнить** (Run). Программа Access предупредит вас, если ей придется перезаписать существующую таблицу (во время импорта) или файл (во время экспорта). В остальном процесс выполнится в одно мгновение.

■ *Удалить сохраненную вами операцию.* Просто выделите ее и щелкните мышью кнопку **Удалить** (Delete).

■ *Создать задачу Outlook для операции.* Это средство можно использовать для напоминания о необходимости выполнения данной операции в урочное время в будущем (или через определенные промежутки времени). Для этого щелкните мышью кнопку **Создать задачу Outlook** (Create Outlook Task) для создания задачи и затем найдите и настройте эту задачу в программе Outlook. Когда напоминание сработает, появится удобная кнопка **Выполнить импорт** (Run Import), которую можно щелкнуть мышью и немедленно запустить операцию импорта в программе Access.

■ *Изменить некоторые аспекты вашей операции.* Можно изменить имя, описание и имя файла, щелкнув мышью соответствующую деталь в диалоговом окне **Задачи управления данными** (Manage Data Tasks) (см. рис. 19.10). В этом случае можно начать импорт в файл c:\My Documents\FancyFiles\WildExpenses.xlsx, а затем с помощью тех же параметров импортировать в файл d:\HankSmith\EvenMoreExpenses.xlsx. Другие детали, например, таблицу-источник или таблицу-назначение, в Access или типы данных полей изменять нельзя.

Когда вы закончите работу в диалоговом окне **Задачи управления данными,** щелкните мышью кнопку **Закрыть** (Close) для того, чтобы вернуться в программу Access.

19.4. *Access и XML*

Одно из самых крутых модных словечек в компьютерном мире - XML (extensible Markup Language, расширяемый язык разметки), универсальный способ обмена информацией между различными программами. Программа Access 2007 поддерживает язык XML с помощью средств импорта и экспорта, в которых XML появляется как один из поддерживаемых форматов. Но если вы действительно хотите понять, как взаимодействуют средства Access и XML и вносят ли они что-то новое, нужно немного углубиться в предмет.

19.4.1. *Что такое XML на самом деле?*

Сам по себе язык XML воспринимается как нечто суперсовременное. Пользователи часто описывают его как формат хранения информации. Например, вместо сохранения данных в документах Word, электронных таблицах Excel или обычных текстовых файлах можно сохранять данные в XML-файле. Эта простота кажущаяся, и два фактора делают язык XML чем-то особенным.

■ *Язык XML обладает большой гибкостью.* Вы можете применить XML для хранения информации самых разных типов: изображений, каталогов товаров, данных счетов, рецептов, номенклатуры каталогов, технических условий для всех Dodge Minivan, когда-либо созданных, и т. д.

■ *Язык XML широко распространен.* Компьютерные приложения, написанные на разных языках программирования (например, Java, Visual Basic либо C++) или выполняющиеся под управлением разных ОС и компьютерного оборудования (например, Windows, Mac или Linux), могут применять XML практически одинаково. Это качество делает язык XML

отличным решением для обмена информацией между людьми, компаниями и даже компьютерами, которые запрограммированы на автоматическую пересылку данных от одного к другому (средства, подобные последнему, заставляют даже типов, занимающихся управлением цепочками поставок, истекать слюной, когда речь заходит об XML).

В противовес мнению многих, XML - это не формат представления данных (как язык HTML - формат, применяемый для создания Web-страниц). Если бы язык XML был просто форматом представления данных, он не стал бы столь незаменимым, поскольку независимо от качества формата он не может подходить для всех. Например, несмотря на то, что всем компаниям нужны счета, большинство компаний не удовлетворится общим форматом хранения данных счета. Одной компании может понадобиться отслеживание имен клиентов, в то время как другая может отслеживать идентификационные номера клиентов. Основной итог заключается в том, что большинству компаний необходимо хранить немного отличающиеся данные слегка разнящимися способами. Это означает, что одно удовлетворяющее всех решение почти всегда обречено на неудачу.

Итак, если язык XML - не формат представления данных, то что же он такое? С технической точки зрения XML - это метаязык, или проще говоря, XML - это язык для создания других языков. Язык XML обеспечивает подобное создание с помощью нескольких простых правил, которые позволяют формировать собственный формат представления информации, подходящий именно для ваших данных.

Например, компания Acme Company может создать свой XML-формат для счетов и назвать его Acmelnvoice. Между тем компания Budget Company может сформировать собственный XML-формат счетов и назвать его Budgetlnvoice. Несмотря на то, что оба формата разработаны для хранения данных счета, они могут содержать совершенно разные типы данных. Сила их - в гибкости языка XML.

В то же время эта гибкость XML способна создать проблемы. Предположим, что банк, названный Worldwide Green, установил систему автоматической обработки XML-счетов с определенным форматом. Система работает гладко до тех пор, пока Acme Corporation не отправит свой доморощенный счет. Несмотря на то, что счет компании Acme использует язык XML, он не соответствует XML-формату, на который рассчитывает банк, и поэтому он нарушит работу банковского приложения автоматической обработки счетов. Неожиданно язык XML перестал казаться таким уж полезным.

В итоге язык XML содержит возможность универсального коллективного использования данных, но если вы не создадите некоторые правила и не будете следовать им, вы останетесь с грудой несовместимых форматов.

Примечание

Язык XML действительно чрезвычайно прост, но существует множество других стандартов, таких как XML Schema и XSLT (Extensible Stylesheet Language Transformations, таблицы стилей языка преобразований XML-документов), работающих совместно с XML и предоставляющих решения для проверки XML, исследования XML, преобразования XML и т. д. Эти дополнительные стандарты очень сложны и не обсуждаются в данной книге.

19.4.2. *Три правила XML*

Для того чтобы лучше понять, как настроить программу Access для обработки XML, рассмотрим простой пример. Технически вам не нужно знать, как выглядит язык XML, для того чтобы применять XML-средства в программе Access, но чем больше вы поймете, тем проще будет жизнь. В этом разделе вы узнаете три самых важных правила, которые формируют все XML-документы. Если вы уже кое-что знаете о языке XML, можно пропустить этот раздел и двигаться дальше.

Между прочим, перед тем как вы начнете, хорошая новость: язык XML написан на базе текстового, удобного для восприятия формата. Вы можете использовать такую программу, как Блокнот для разбора имеющегося XML-файла и получения базового представления о его формате и структуре. Вы даже можете написать XML-файл с нуля в Блокноте. С типичной БД Access такое проделать не удастся, потому что она хранится в двоичном формате, который можно прочесть, только если просматривать данные в программе Access. (Если попытаться

открыть БД в Блокноте, вы увидите груду неподдающихся расшифровке символов.)

19.4.2.1. *Пролог*

Все заслуживающие уважения XML-документы начинаются с так называемого пролога документа (document prolog). В этом кусочке просто объявляется, что то, на что вы смотрите, и есть XML-документ. Он также указывает на кодировку документа, которая иногда задает применение в документе специального набора символов (например, неанглийский алфавит).

Далее приведен типичный пролог документа, указывающий, что в документе используется версия 1.0 XML-стандарта (наиболее распространенная версия):

```
<?xml version="1.0"   ?>
```

Если XML-документ создается вручную, следует убедиться в том, что пролог находится в самой первой строке файла.

19.4.2.2. *Элементы*

Базовый строительный блок любого XML-документа - элемент. Элементы - это информационные контейнеры. Например, если вы хотите сохранить имя человека, можно создать элемент с именем Name.

Типичный элемент состоит из открывающего и закрывающего тегов. Реальные данные располагаются между этими двумя тегами. Открывающие и закрывающие теги легко узнать, поскольку они используют угловые скобки о. Далее приведен возможный открывающий тег:

```
<Name>
```

Этот тег обозначает начало элемента Name. Закрывающий тег выглядит аналогично, за исключением того, что он начинается с символов </ вместо просто символа <. Вот как следует закрыть элемент Name:

```
 </Name>
```

Для того чтобы действительно сохранить некоторые данные в XML-документе, содержимое нужно вставить между открывающим и закрывающим тегами элемента. Далее показано, как сохранить чье-либо имя в XML-документе:

```
<Name>Patrick</Name>
```

Можно создать список имен, размещая один элемент <Name> за другим, или добавить другие элементы, хранящие информацию разных типов, например, адрес, звание, имя работодателя и т. д. Для формирования XML-документа все эти теги вместе помещаются в файл.

На профессиональном уровне.

Более внимательный взгляд на теги

Теги следуют очень строгим правилам именования. Теги могут быть любой длины, указываются с учетом регистра, могут содержать любые цифровые или буквенные символы, дефисы (-), знаки подчеркивания (_) и точки (.). Другие специальные символы, в том числе пробелы, применять нельзя, и имя тега должно начинаться с буквы или знака подчеркивания. XML-документы также поддерживают символы, не включенные в английский алфавит.

Самое важное, что вам следует понять о тегах, состоит в том, что создание тегов - ваша задача. Если нужно хранить список имен, можно создать XML-формат, который использует тег <Name>. Одновременно кто-то еще может решить отслеживать фамилию, имя и отчество с помощью другого XML-формата, использующего такие элементы, как <firstName> и <lastName>. В этих элементах может храниться та же информация, что и в вашем элементе <Name>, но они отличаются, и документ, написанный с использованием тегов <f irstName> и <lastName>, несовместим с вашими документами.

Поскольку существует так много возможных XML-форматов, множество умных людей потратили уйму времени и энергии на попытки создать варианты определения разных XML-форматов и управления ими. Компании и организации также собрались вместе для определения специальных XML-стандартов для разных отраслей. Если поискать в Интернете, можно найти заранее определенные форматы для юридических, научных, касающихся недвижимости и многих других документов.

19.4.2.3. *Вложенность*

Пока вы видели примеры XML-элементов, содержащих текст. Можно также создать элемент, содержащий один или несколько других элементов. Это основной принцип организации информации в языке XML.

Допустим, вы хотите отслеживать имена и возрасты нескольких людей. Приведенный далее фрагмент не очень понятен, поскольку трудно сказать, кто конкретно связан с тем или иным возрастом:

```
<Name>Lisa Chen</Name>
<Age>19</Age>
<Name>Bill Harrison</Name>
<Age>48</Age>
```

Лучше было бы сгруппировать элементы <Name> и <Age> для каждого человека и поместить их в другой элемент, например, так:

```
<Person>
<Name>Lisa Chen</Name>
 <Age>19</Age> </Person>
<Person>
<Name>Bill Harrison</Name>
<Age>48</Age> </Person>
```

В этом примере приведены два элемента <Person>, каждый из которых содержит сведения о конкретной персоне. Информация об отдельном человеке хранится в элементах <Name> и <Age>, вложенных в соответствующий элемент <Person>.

Не существует ограничений на количество уровней вложенности данных, что делает этот метод организации информации чрезвычайно гибким. На самом деле это одна из составляющих, которые обеспечивают языку XML возможность работы с множеством различных типов данных.

В языке XML установлено еще одно правило. Каждый документ должен начинаться единственным элементом, следующим непосредственно за прологом. Все остальное содержимое помещается в этот элемент, который называется *корневым элементом* или *элементом документа.* Виденные вами до настоящего момента примеры были лишь выдержками из XML-документа. В следующем далее листинге показан полный, хорошо оформленный XML-документ - список с данными о двух людях, - который начинается с элемента документа <PeopleList>:

```
<?xml version="1.0".?>
<PeopleList>
    <Person>
<Name>Lisa Chen</Name>
<Age>19</Age> </Person>
<Person>
<Name>Bill Harrison</Name>
<Age>48</Age> </Person> </PeopleList>
```

Этот документ можно усовершенствовать, добавив дополнительные элементы <Person> или другие элементы, позволяющие отслеживать дополнительные сведения о каждом человеке.

Возможно, вы заметили, что в приведенных примерах XML у элементов разных уровней различные отступы. Такое оформление облегчает чтение всей структуры, но не является обязательным требованием. На самом деле приложения, читающие XML-файлы (включая программу Access), игнорируют все пустое пространство между элементами, поэтому вставленные пробелы, табуляции и пустые строки не играют никакой роли. В действительности с точки зрения компьютеров приведенный только что документ абсолютно идентичен следующей менее легкой для восприятия версии:

```
<?xml  version="1.0"   ?>
<PeopleListXPerson><Name>Lisa Chen</Name><Age>19</
AgeX/Person><Person><Name>Bill Harrison</Name><Age>48 </Agex/Person></PeopleList>
```

19.4.3. *Файлы и схемы XML*

Как вы уже узнали, XML-документы можно хранить в файле. Но так же легко их можно поместить и в БД или другие места внешней памяти. В реальной жизни иногда XML-данные не сохраняются нигде - пользователи применяют их для обмена информацией между приложениями через Интернет. Но если XML-данные используются в программе Access, всегда применяются XML-файлы (если ваша компания не создала пользовательское решение с помощью мощных средств программирования в Access).

У большинства XML-файлов расширение xml. Например, самое разумное - взять документ со списком людей, показанный ранее, и поместить его в текстовый файл, названный PersonList.xml.

Еще один очень важный тип XML-документа - XML-схемы. XML-схемы разрабатываются для решения общей проблемы, а именно определения правил для конкретного, основанного на языке XML формата. Например, схема указывает имена элементов, которые можно использовать, способ организации элементов и тип информации, которую может содержать каждый элемент. Приложение с поддержкой XML может применить схему для проверки правильности структуры и допустимости содержимого XML-документа. В идеальном мире каждый раз, когда компания создает XML-формат, она разрабатывает и XML-схему, его определяющую. (Возможно, вас не удивит, то, что так бывает далеко не всегда.)

Для применения схемы нужно просто иметь ее копию в файле. (Сами по себе схемы сложны и уродливы и не относятся к первостепенным вещам, которые должны и хотят изучать сотрудники типичного офиса.) Обычно у файлов со схемами расширение xsd.

Примечание

Для более полного введения в язык и схемы XML познакомьтесь с отличным интерактивным руководством, предлагаемым компанией W3 Schools, на сайте **www.w3schools.com/xml.**

19.4.4. *Поддержка XML в программе Access*

Язык XML - это прекрасный способ обмена данными между разными компьютерными программами. Но что делать с программой Access, у которой уже есть собственный отличный способ хранения данных? Ситуация такова: сегодня все больше и больше компаний применяют язык XML для обмена данными. Когда компании обмениваются коммерческими заказами, например, или новостные организации посылают очерки, или фирмы по торговле недвижимостью перечисляют предлагаемую для продажи собственность, очень вероятно, что они используют формат на основе XML. Если вы хотите отправить ваши данные Access в такие системы, нужен способ извлечения их из специализированного ACCDB-формата БД и помещения их в простой и понятный XML-формат.

К сожалению, поддержка языка XML в программе Access до сих пор очень ограничена. Проблема заключается в том, что Access не разрешает выбрать нужный вам XML-формат. Вместо этого программа создает собственный формат, который точно соответствует вашей таблице. Посмотрите на таблицу на рис. 19.11. (При экспорте в XML-файл всегда экспортируется таблица целиком.)

ProductID	Name	ProductNumber	SafetyStockLevel	ReorderPoint
371	Thin-Jam Hex Nut 7	HJ-7161	1000	750
372	Thin-Jam Hex Nut 8	HJ-7162	1000	750
373	Thin-Jam Hex Nut 12	HJ-9080	1000	750
374	Thin-Jam Hex Nut 11	HJ-9161	1000	750
375	Hex Nut 5	HN-1024	1000	750
376	Hex Nut 6	HN-1032	1000	750
377	Hex Nut 16	HN-1213	1000	750
378	Hex Nut 17	HN-1220	1000	750
379	Hex Nut 7	HN-1224	1000	750
380	Hex Nut 8	HN-1420	1000	750
381	Hex Nut 9	HN-1428	1000	750
382	Hex Nut 22	HN-3410	1000	750

Рис. 19.11. Пример данных, готовых к новой жизни в XML-формате

Когда экспортируется приведенная таблица, программа Access создает XML-документ, который выглядит следующим образом:

```
<dataroot>
<Product>
<ProductID>371</ProductID>
<Name>Thin-Jam Hex Nut 7</Name>
<ProductNumber>HJ-7161</ProductNumber>
<SafetyStockLevel>1000</SafetyStockLevel>
<ReorderPoint>750</ReorderPoint> </Product> <Product>
<ProductID>372</ProductID>
<Name>Thin-Jam Hex Nut 8</Name>
<ProductNumber>HJ-7162</ProductNumber>
<SafetyStockLevel>1000</SafetyStockLevel>
<ReorderPoint>750</ReorderPoint>
</Product>
```

Независимо от экспортируемой таблицы программа Access всегда следует одним и тем же правилам:

- корневой элемент документа называется <dataroot>;
- программа Access создает отдельный элемент для каждой строки таблицы, используя имя таблицы. В данном примере это означает, что вы получите один элемент <Product> для каждой записи;
- внутри каждой записи Access создает отдельный элемент для каждого поля. В данном примере вы получите поля <Name>, <ProductNumber> и т. д.

В подобной структуризации XML-документа нет ничего плохого. Однако поскольку у вас нет возможности изменять структуру, вы потерпите неудачу, если захотите использовать другую программу, которая принимает XML-документы с другим форматом. Например, ваша программа рассчитывает, что корневой элемент будет назван <ProductRecords>, а не <dataroot> или она принимает несколько иную вложенность элементов. Незначительные нестыковки, подобные описанным, могут полностью нарушить работу приложения с XML-обработкой.

Печально, но эту проблему не обойти. Для того чтобы использовать XML-файлы программы Access, нужно специально разрабатывать программу, распознающую эту структуру, или применять другое средство для преобразования XML-файла в стандарт, который вам на самом деле нужен. Средство экспорта в XML-файл программы Access позволяет начать, но не предоставляет вариантов для представления данных.

Примечание

Если вам необходимо отфильтровать записи или не интересующие вас поля либо переименовать поля, задачу можно решить с помощью запроса. Просто создайте запрос, который представляет информацию нужным вам образом, и затем экспортируйте результаты запроса (вместо целой таблицы).

Те же ограничения возникают, когда импортируется XML-содержимое. Программа Access рассчитывает на получение XML-содержимого с жестким табличным форматом. Если вы попытаетесь представить ей XML-документ другого типа, то получите сообщение об ошибке.

19.4.5. *Экспорт в XML-файл*

Теперь, когда вы познакомились с языком XML и рассмотрели ограничения, установленные в программе Access, вы готовы применить его для собственных нужд. Далее перечислены необходимые для этого действия.

1. Выберите на ленте **Внешние данные → Экспорт → Дополнительно → XML-файл** (External Data → Export → More → XML File).

Начинается знакомый вам процесс экспорта.

2. Введите имя файла, который хотите создать, и нажмите кнопку ОК.

Программа Access предполагает, что будет использоваться имя таблицы. Например, если экспортируется таблица **Orders,** рекомендуемое программой имя XML-файла - Orders.xml.

3. Выберите типы файлов, которые хотите создать (рис. 19.12):

- **данные (XML)** (Data (XML)) - создается XML-файл с действительным содержимым из всех записей вашей таблицы;
- **схема данных (XSD)** (Schema (XSD)) - создается файл схемы с расширением xsd. Схема не содержит никаких данных, а хранит краткое определение, описывающее вашу таблицу и содержащиеся в ней поля. У схемы два назначения - ее можно передать профессиональным программистам, чтобы они знали, какого типа XML-документ ждать от программы Access, или использовать ее для создания новой пустойтаблицы в другой БД Access;
- **презентация данных (XSL)** (Presentation (XSL)) - создает файл преобразования с расширением xsl. В этом файле определяется, как обозреватель может преобразовать исходные данные из XML-файла в Web-страницу на языке HTML, подходящую для отображения в Web-обозревателе. Когда этот флажок установлен, Access также создает htm-файл, использующий xsl-файл. Например, если экспортируется таблица **Products,** вы получите в результате файл Web-страницы Products.htm. Откройте ее в вашем Web-обозревателе, и он воспользуется файлом Products.xsl для отображения данных из файла Products.xml.

Рис. 19.12. Обычно создается XML-файл, хранящий реальные данные из вашей таблицы. Кроме того, можно создать два дополнительных файла поддержки

4. Если вы хотите экспортировать связанные таблицы в одном и том же XML-документе, щелкните мышью кнопку **Дополнительно...** (More Options...).

На экране появится окно **Экспорт XML** (Export XML) с дополнительными параметрами. Большинство из этих параметров лучше всего оставить гуру XML. Но наиболее интересна вкладка **Данные** (Data) - она позволяет экспортировать связанные таблицы (рис. 19.13).

Например, при экспорте таблицы **Orders** у вас есть две возможности.

□ *Экспорт других подчиненных таблиц.* Можно было бы также экспортировать записи таблицы **OrderDetails** для каждого заказа. Программа Access вкладывает в XML-файл элементы **OrderDetails** внутрь соответствующего элемента **Orders.**

□ *Экспорт связанных записей из главной таблицы.* Можно было бы, например, экспортировать записи из таблиц **OrderStatus** и **CreditCards.** Эти записи появляются под заголовком **[Данные подстановки]** ([Lookup Data]), поскольку они предоставляют дополнительные данные, связанные с заказом (в данном случае текущий статус заказа и кредитную карту, применяемую для оплаты заказа).

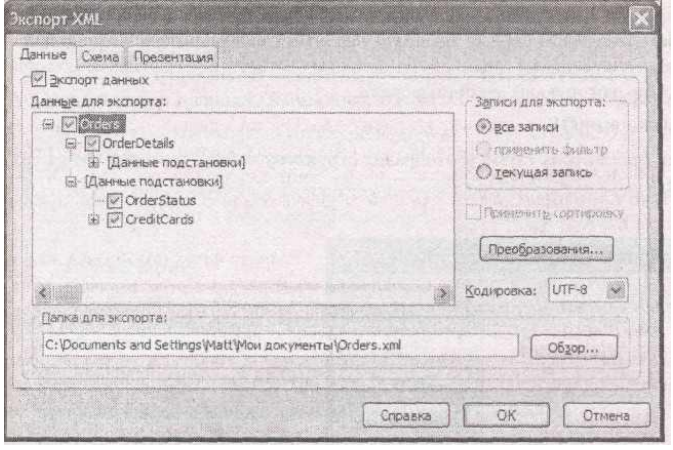


Рис. 19.13. На вкладке **Данные** отображается дерево с экспортируемой таблицей в корне, а ветви дерева представляют связанные таблицы. Если вы хотите включить данные из этих связанных таблиц, просто установите флажок, расположенный рядом с каждой из них

Примечание

Когда экспортируются главные или родительские таблицы, данные не вкладываются в XML-документе, поскольку это могло бы привести к дублированию (например, если у нескольких заказов один и тот же статус или в них применяется одна и та же кредитная карта). Вместо этого они добавляются после основной экспортируемой таблицы.

5. Щелкните мышью кнопку ОК.

Программа Access создаст файлы, выбранные в пункте 3.

6. Если вы хотите повторить процесс экспорта несколько раз, установите флажок **Сохранить шаги экспорта** (Save export steps).

Щелкните мышью кнопку **Закрыть** (Close) для возврата в окно программы Access.

19.4.6. *Импорт из XML-файла*

Программа Access делает очень легким процесс импорта XML-данных, при условии что у них структура, на которую рассчитывает программа. Для проверки на практике выберите только что экспортированную таблицу и заново импортируйте ее в новую БД. Вот как это делается.

1. Выберите на ленте **Внешние данные → Импорт → Импорт XML-файла** (External Data → Import → XML File).

Начнется знакомый процесс импорта.

2. Если создается новая таблица и для ваших данных есть схема, задайте имя файла, содержащего схему. Если у вас уже есть таблицы, которые хотите использовать, или под рукой нет схемы, перейдите сразу к пункту 6.

Можно импортировать непосредственно из XML-файла, но всегда лучше применить схему, если таблица создается впервые, поскольку в схеме хранится информация о типах данных всех полей. Эти сведения гарантируют большее соответствие создаваемой таблицы исходной таблице, которую вы экспортировали.

3. Щелкните мышью кнопку ОК.

Программа Access просмотрит схему и отобразит структуру таблиц, которые будут созданы (рис. 19.14).

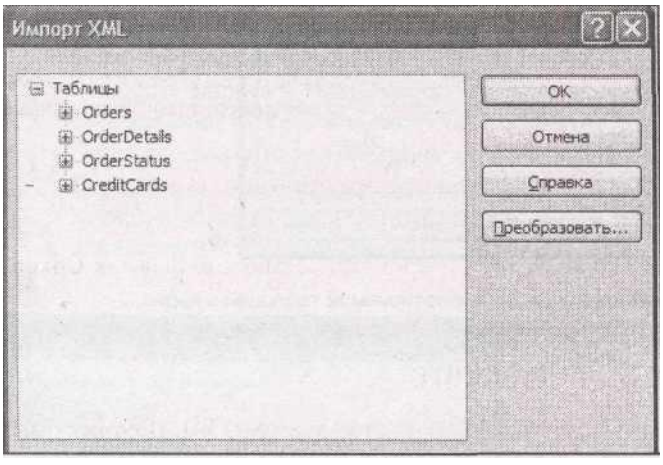


Рис. 19.14. В данном примере Access корректно устанавливает, что ваш файл схемы определяет структуру таблиц **Orders, OrderDetails, OrderStatus** и **CreditCards.** Можно раскрыть каждую таблицу и увидеть содержащиеся в ней поля

4. Щелкните мышью кнопку ОК.

С помощью данных схемы программа Access создает новую пустую таблицу с корректной структурой. Теперь можно заполнять ее данными.

Примечание

Если таблица с таким же именем уже существует, Access добавит в конец имени число, чтобы отличать новую таблицу (например, **Products1, Products2** и т. д.).

5. Щелкните мышью кнопку **Закрыть** (Close), чтобы вернуться в окно программы Access.

6. Выберите на ленте **Внешние данные →Импорт → Импорт XML-файла** (External Data → Import → XML File).

Теперь, когда ваши таблицы созданы, можно импортировать реальные данные.

7. Задайте имя XML-файла, содержащего данные, которые хотите импортировать, и щелкните мышью кнопку ОК.

Программа Access отобразит структуру таблицы на основе XML-данных вашего файла, Эта структура должна точно соответствовать структуре таблицы, которую вы хотите создать или в которую хотите добавить данные.

8. Выберите один из вариантов операции импорта.

- Выбор переключателя **добавить данные в таблицы** (Append Data to Existing Table(s)) заставляет программу Access найти таблицу с таким же именем и добавить в нее все данные. Этот вариант применяется, если используемая таблица уже существует.
- Выбор переключателя **структура и данные** (Structure and Data) создает таблицу и затем заполняет ее данными.
- Выбор переключателя **только структура** (Structure Only) создает несуществующую до этого момента таблицу, но не импортирует никаких данных.

Примечание

Если в процессе импорта вам нужно создать таблицу, всегда лучше применить файл схемы для ее создания (как описано в пунктах 1-5), поскольку в схеме содержатся более точные сведения о типах данных.

9. Щелкните мышью кнопку ОК.

Access заполнит таблицы данными из вашего XML-файла.

Если вы хотите повторять процесс импорта несколько раз, установите флажок **Сохранить шаги импорта** (Save export steps).

19.5. *Сбор информации по электронной почте*

В мире пользователей электронной почты гораздо больше, чем знатоков БД. Поэтому было бы очень кстати иметь способ получения данных от других людей с помощью сообщений электронной почты. Специалисты, работавшие над программой Access, думали также. Они добавили еще одно средство передачи данных в Access из других источников: данные можно собирать по электронной почте.

Вот как действует это средство.

1. Вы выбираете таблицу, которая нуждается в данных.

2. Определяете круг пользователей, способных предоставить эти данные. (Необходим список адресов электронной почты, которые можно вводить вручную или извлечь из таблицы в вашей БД.)

3. Программа Access отправляет им всем электронное сообщение с формой, которую они должны заполнить. Форма позволяет им ввести данные, формирующие одну запись (хотя адресат при необходимости может заполнить одну и ту же форм}' несколько раз). Для того чтобы увидеть пример того, как может выглядеть электронное сообщение, перейдите к рис. 19.16.

4. Заполненная форма возвращается вам по электронной почте.

5. Для каждого полученного по электронной почте сообщения программа Access создает в вашей таблице отдельную запись.

Часто средство сбора данных с помощью электронной почты применяется для получения сведений о людях. Например, у вас может быть таблица со списком контактов. Вы можете отправить каждому человеку электронное сообщение и попросить его предоставить личные

данные (адрес, номер телефона и т. д.). Конечно, с помощью этого средства можно собрать информацию и другого типа, например, список компонентов, которые люди хотят принести, для организации обеда в складчину.

У средства сбора данных с помощью электронной почты есть несколько базовых правил.

- Для отправки электронного сообщения и получения ответа необходимо использовать программу Microsoft Outlook 2007 (входящую в состав пакета Office 2007). Таким образом, если программа Outlook еще не настроена, следует сделать это прежде, чем двигаться дальше. Ваши адресаты могут пользоваться любой программой электронной почты, которая им нравится.
- Данные можно только вставлять, но не обновлять. (Есть одно исключение. Можно обновить таблицу, если каждая запись содержит в поле адрес электронной почты отправителя, поскольку в этом случае программа Access может установить, какую запись обновлять, сопоставляя адрес электронной почты отправителя с адресом, хранящимся в таблице.)
- Если пользователи включают неверные данные (например, помещают текст в числовое поле), программа Access не может создать запись. Затем вы должны самостоятельно выяснить, в чем проблема, и исправить ее.
- Возможно, придется потратить какое-то время на просмотр данных, предоставленных другими людьми. Небрежность людей при заполнении форм на компьютере печально известна. Они могут ввести свои имена и фамилии только строчными буквами, пропустить важную информацию, допустить орфографические ошибки или непристойные шутки и т. д.

В следующих разделах вы увидите, как создать нужное сообщение электронной почты и затем поместить данные в вашу таблицу.

19.5.1. *Создание сообщения электронной почты*

Прежде всего (при условии, что вы уже установили программу Outlook и настроили вашу учетную запись электронной почты в ней) укажите таблицу, куда хотите вставлять данные. В следующем примере вы увидите, как добавить предполагаемых кандидатов в таблицу **Bachelors** (холостяки) БД Dating Service (служба знакомств). Если вы хотите испытать это средство со своими друзьями, БД Dating Service можно найти на странице "Missing CD" Web-сайта **www.missingmanuals.com.**

Вот, как это делается.

1. Выберите на ленте **Внешние данные → Сбор данных → Создание электронного сообщения** (External Data → Collect Data → Create E-mail).

На экране появится мастер. В первом окне мастера перечислены все шаги, необходимые для получения ваших данных.

2. Для того чтобы двигаться дальше, нажмите кнопку **Далее** (Next).

На втором шаге можно выбрать тип формы, которую хотите использовать.

3. Выберите вариант **HTML-форма** (HTML form) и щелкните мышью кнопку Далее.

Этот вариант сообщает программе Access о необходимости использовать HTML-теги в ее сообщении электронной почты. С помощью этих тегов Access может создать форму с привлекательным внешним видом и полями ввода, в которые адресат сможет ввести данные.

Второй оставшийся вариант (Microsoft Office InfoPath form) отключен, если на вашем компьютере не установлено приложение InfoPath. Оно включено в состав только некоторых редакций пакета Office и чаще всего используется в больших компаниях. Несмотря на то, что InfoPath - отличная программа, у нее есть один серьезный недостаток - для того чтобы применить форму InfoPath, у всех ваших адресатов на компьютерах должна быть установлена программа InfoPath. По этой причине вариант HTML, как правило, выгодней.

4. На следующем шаге запрашивается необходимость сбора новых данных или обновления существующих. Выберите подходящий вариант и щелкните мышью кнопку **Далее.**

Обычно выбирается вариант **Собирать только новые данные** (Collect new information only). Именно он подходит для БД Dating Service, поскольку вы хотите получить информацию, которую нужно добавить в таблицу **Bachelors** как отдельную запись для каждого отправителя.

Если выбирается вариант **Обновлять существующие данные** (Update existing information),

в вашей таблице должны содержаться адреса электронной почты отправителей. Например, этот подход можно применять, если в таблице **Bachelors** есть набор записей, которые вы хотите обновить. Каждый холостяк получит электронный адрес, который можно использовать для изменения текущих данных о себе.

Подсказка

Вы, конечно, также можете применять вариант обновления, если адреса электронной почты есть в другой таблице, которая связана с заполняемой таблицей. Например, можно было бы обновить текущий статус каждого проекта из таблицы **Projects,** если в ней есть поле **Project-ManagerlD,** указывающее на запись в таблице **ProjectManagers.** которая в свою очередь содержит адрес электронной почты.

5. Выберите поля, которые хотите собрать.

Для добавления поля выберите его в списке **Поля в таблице** (Fields in table) и затем щелкните мышью кнопку >. Или щелкните мышью кнопку » для переноса всех полей за один шаг.

Примечание

Вы не увидите поле **Код** (ID) с типом **Счетчик** в списке **Fields in table** (Поля в таблице). Программа Access знает о том, что она должна генерировать это значение самостоятельно, поэтому не просит никого предоставить это значение. Также не отображаются многозначные поля или поля с типом **Вложение,** поскольку Access не может создать формы для данных этих типов.

6. При желании задайте более понятные имена для ваших полей.

Например, подпись "Your favorite food is" ("Ваша любимая еда") может оказаться понятнее, чем имя поля **FoodPreference** (пищевые предпочтения). Для изменения подписи выберите ее в списке и измените поле ввода, появляющееся ниже.

Можно также установить флажок **Только чтение** (Read-only), чтобы пользователи не могли изменить значение поля. Этот вариант имеет смысл, только если вы просите пользователей обновить записи. В этой ситуации, возможно, на форме есть данные для просмотра, но не для изменения.

7. При желании измените порядок следования полей.

Для переноса поля выделите его в списке включенных в форму полей и воспользуйтесь кнопками со стрелкой, направленной вверх или вниз. Когда программа Access создаст форму электронной почты, она поместит на ней поля в том же порядке.

8. Щелкните мышью кнопку **Далее** (Next).

На экране появляется последнее окно мастера.

9. Выберите папку, в которой Access сохраняет ответы после того, как обработала их.

Обычно программа Access хранит ответы в папке Outlook, названной **Ответы для сбора данных Access** (Access Data Collection Replies). Но можно применить любую папку, какую захотите. Для замены папки щелкните кнопкой мыши ссылку **Ответы для сбора данных Access.** Запустится программа Outlook и выведет на экран диалоговое окно **Обзор** (Select Folder), в котором можно выбрать любую имеющуюся папку (или щелкнуть мышью кнопку **Создать папку** (New) для создания новой папки). После выбора нужной папки щелкните мышью кнопку ОК.

Подсказка

Если вы планируете выполнить несколько операций импорта для разных таблиц, есть смысл использовать разные папки.

10. Если хотите применить автоматическую обработку, установите флажок **Автоматически обрабатывать ответы...** (Automatically process replies...). Если хотите обработать ответы вручную, перейдите к пункту 12.

Если применяется автоматическая обработка, программа Outlook связывается с Access, чтобы выяснить, получила ли она ответ. Затем программа Access сразу добавляет или обновляет

соответствующую запись. Подобная система действует до тех пор, пока файл вашей БД остается на одном и том же месте, с тем же именем **и** без защиты паролем.

Для ручной обработки требуется больше усилий, но это по-настоящему безопасный вариант. В этом случае можно просмотреть каждый ответ, прежде чем добавлять запись. Вы также точно знаете, сколько ответов получили и можете проверить ошибки до того, как данные попадут в вашу таблицу. Таким образам, ручная обработка - лучший выбор.

Для ручной обработки требуется больше усилий, но это по-настоящему безопасный вариант. В этом случае можно просмотреть каждый ответ, прежде чем добавлять запись. Вы также точно знаете, сколько ответов получили и можете проверить ошибки до того, как данные попадут в вашу таблицу. Таким образам, ручная обработка - лучший выбор.

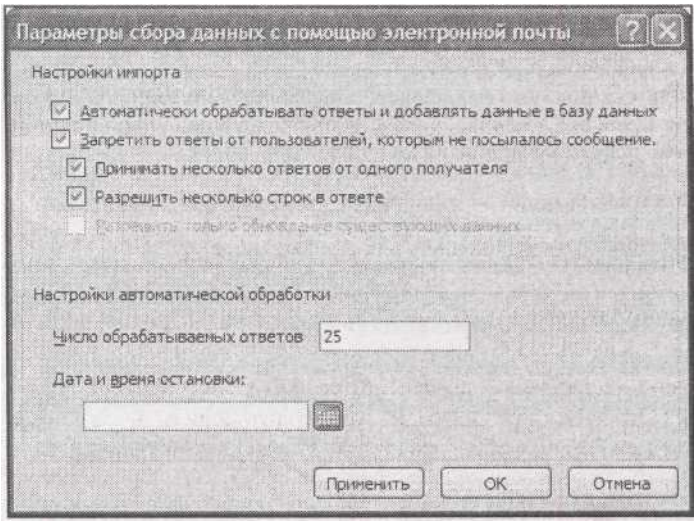


Рис. 19.15. Если вы выбрали автоматическую обработку ответов, в этом диалоговом окне можно задать ряд параметров

11. Если используется автоматическая обработка, щелкните кнопкой мыши ссылку **Задать свойства, управляющие автоматической обработкой ответов** (Set properties to control the automatic processing of replies) для отображения диалогового окна **Параметры сбора данных с помощью электронной почты** (Collecting Data Using E-mail Options) (рис. 19.15). Задайте нужные параметры и щелкните мышью кнопку OK.

Вы можете управлять следующими параметрами.

- □ Флажок **Запретить ответы от пользователей, которым не посылалось сообщение**
- (Discard replies for those to whom you did not send the message) позволяет игнорировать сообщения от людей, которым вы не посылали писем по электронной почте.
- Флажок **Принимать несколько ответов от одного получателя** (Accept multiple re plies from each recipient) позволяет отправителям отвечать столько раз, сколько они захотят. При каждом получении сообщения программа Access добавляет запись в таблицу. Это имеет смысл, если вы, скажем, собираете список вещей, которые ваши друзья хотят продать на групповой распродаже ненужных домашних пожитков. Данный вариант лишен смысла, если вы накапливаете персональные данные группы холостяков, поскольку каждому из них отводится одна запись.
- Флажок **Разрешить несколько строк в ответе** (Allow multiple rows per reply) действует, только если применяется программа InfoPath. С ее помощью можно представить в одной форме данные, предназначенные для нескольких записей (если этот параметр установлен).
- Флажок **Разрешить только обновление существующих данных** (Only allow updates to existing data) действует, только если выполняется обновление (см. пункт *A)*. В этом случае можно установить флажок для того, чтобы помешать пользователям добавлять новые записи. И снова этот параметр доступен только при использовании программы InfoPath.
- Поле **Число обрабатываемых ответов** (Number of replies to be processed) позволяет остановить обработку после получения определенного числа ответов. С этого момента

программа Access игнорирует все ответы (если вы не выберете ручную обработку ответов, как описано в следующем разделе).

- Поле **Дата и время остановки** (Date and time to stop) позволяет остановить обработку в определенный момент времени. Ответы, которые придут позже, будут игнорироваться, хотя их можно обработать вручную.

12. Щелкните мышью кнопку **Далее** (Next).

В следующем окне выводится запрос о том, как вы хотите предоставлять адреса электронной почты.

13. Выберите вариант предоставления адреса электронной почты и затем нажмите кнопку **Далее.**

Выберите вариант **Ввод адресов электронной почты в Microsoft Office Outlook** (Enter the email addresses in Microsoft Office Outlook), если хотите вводить адреса электронной почты ваших получателей (или выбирать их из Адресной книги). Затем переходите к пункту 15.

Выберите вариант **Использовать адреса электронной почты, хранящиеся в поле в базе данных** (Use the email addresses stored in a field in the database), если хотите извлекать адреса электронной почты из таблицы.

Если вы обновляете таблицу, то не увидите этого этапа. Вы всегда должны извлекать адреса из таблицы.

14. Если адреса электронной почты предоставляются из таблицы, нужно сообщить программе Access, какие таблицу и поле использовать для этого. Затем щелкните мышью кнопку **Далее** (Next).

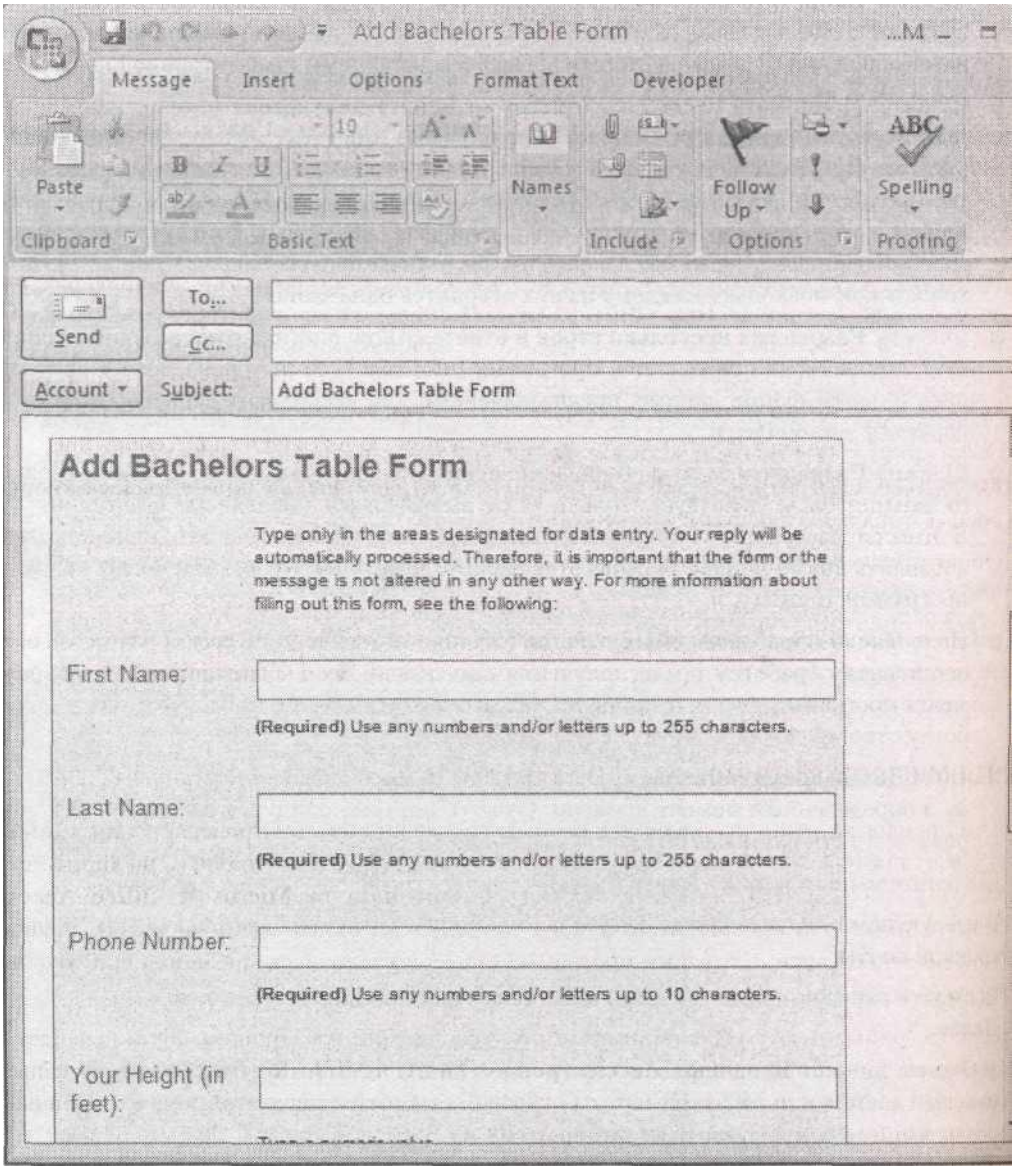


Рис. 19.16. На рисунке приведена форма для таблицы **Bachelors.** Вы заметите, что Access автоматически определяет обязательные поля и вставляет в каждую форму некоторые

уточнения, касающиеся приемлемых типов данных

Можно извлекать адреса из текущей таблицы (если выполняются обновления записей) или другой связанной таблицы (в случае обновления или добавления записей). Если информация обновляется в таблице **Bachelors,** можно использовать поле **Email** в этой таблице. Если же создается список проектов, каждый из которых связан с записью о менеджере проекта, адреса электронной почты можно извлечь из таблицы **ProjectManagers** и разрешить пользователям создавать связанные записи в таблице **Projects.**

15. Настройте до мелочей электронное сообщение, которое собираетесь отправлять, и щелкните мышью кнопку Далее (Next).

Можно изменить текст в поле **Тема** (Subject) и **Введение** (Introduction).

16. Вы добрались до последнего этапа. Щелкните мышью кнопку **Создать** (Create) и приготовьтесь к отправке вашего сообщения.

После щелчка мышью кнопки Создать программа Access создает форму и загружает ее в новое сообщение, которому не терпится отправиться в путь (рис. 19.16).

Если был выбран вариант извлечения адресов электронной почты из таблицы, вы увидите эти адреса в полях To... (Кому...) Cc... (Копия...). В противном случае строка To (Кому...) остается пустой, и ваша задача - заполнить ее правильными адресами. (Вставляйте столько адресов, сколько нужно, отделяя один от другого точкой с запятой.) Если вы хорошо знакомы с программой Outlook, можно включить в этот перечень список рассылки. Вы также можете внести последние изменения в ваше сообщение.

17. Как только корректные адреса электронной почты внесены, щелкните мышью кнопку **Send** (Отправить) для пересылки сообщения.

Оно отправлено. Ваша работа завершена (до тех пор, пока кто-то не получит его и не пошлет ответ).

Для заполнения ответа адресату нужно щелкнуть мышью кнопку **Reply** (Ответить), ввести значения в поля ввода и затем щелкнуть мышью кнопку **Send** (Отправить) для возврата вам заполненной формы.

19.5.2. *Ручная обработка ответов*

Если вы предпочли ручную обработку, необходимо периодически проверять папку **Inbox** (Входящие) программы Outlook для поиска ответов. Когда ответ найден, щелкните его правой кнопкой мыши и выберите команду **Export data to Microsoft Office Access** (Экспорт данных в Microsoft Access). Она появляется, только если щелкнуть правой кнопкой мыши сообщение, которое программа Outlook распознает как заполненную форму Access (рис. 19.17).

Если программа Access успешно импортирует сообщение, электронная почта перейдет в папку **Ответы для сбора данных Access** (Access Data Collection Replies) (или в ту папку, которую вы задали в пункте 9). Программа Outlook отобразит подтверждающее сообщение, которое извещает о том, что все прошло хорошо.

Если найдено сообщение, которое программа Access не может обработать, вам придется решать, как с ним быть. Проблемы могут возникнуть из-за значений, не прошедших проверку на значения, дублирующихся значений полей, в которых дублирование запрещено, и значений, не соответствующих заданным типу данных или размеру поля.

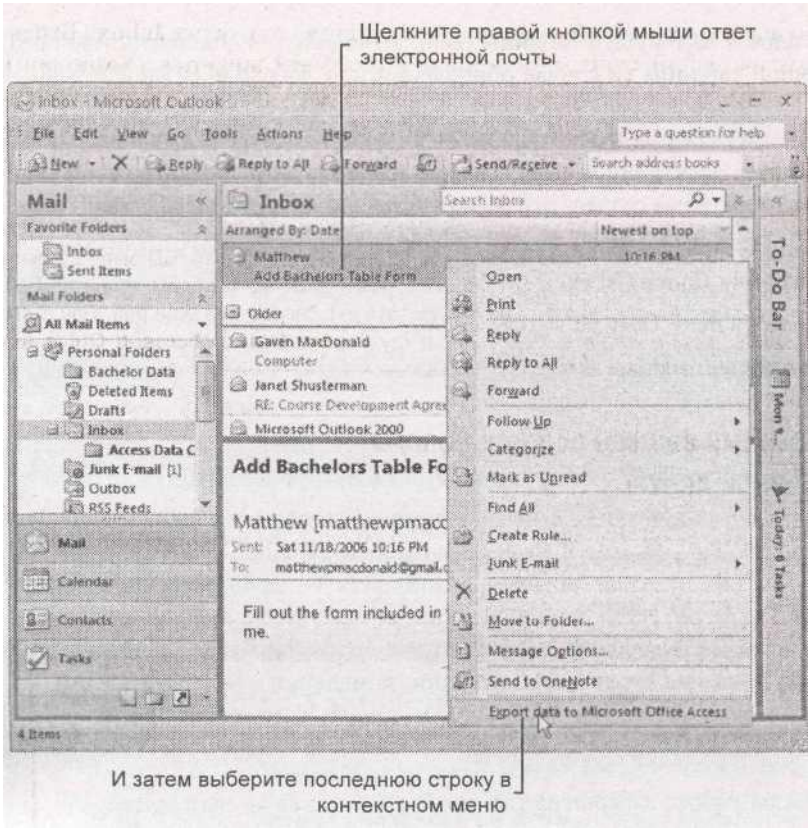


Рис. 19.17. Выберите этот вариант контекстного меню, и текущее сообщение отправится в программу Access, которая поместит его в новую запись в таблице **Bachelors**

Далее перечислены возможные способы устранения проблем в сообщениях, которые Access не может обработать.

- *Удалить сообщение и забыть о нем.* Вы можете применить этот подход, если обнаружили явно некорректное сообщение или сообщение, дублирующее уже имеющуюся в таблице запись.
- *Запросить корректировку.* Отправить форму адресату и попросить его попробовать еще раз.
- *Ввести корректные данные вручную.* Если вы можете исправить неверные данные, проблему можно устранить самостоятельно. В этом случае используйте лист данных программы Access для добавления записи, которую нужно было бы создать.

19.5.3. *Автоматическая обработка ответов*

Если был выбран вариант автоматической обработки, больше никаких действий предпринимать не нужно. По мере поступления ответов в вашу папку Inbox (Входящие) программа Access вставляет данные в таблицу без какого-либо уведомления вас об этом. Неплохо почаще проверять таблицу, чтобы убедиться в том, что вставленные данные не содержат явных ошибок. Нужно также просматривать папку Inbox (Входящие) в программе Outlook для обнаружения сообщений, которые Access не смогла обработать, например те, в которых содержатся некорректные данные.

Вы поймете, что сообщение не было обработано, если увидите его в папке Inbox (Входящие) с красным квадратиком или пустым значением в столбце Categories (Категории). В этой ситуации можно попробовать применить все варианты устранения проблем, описанные в предыдущем разделе.

Примечание

Можно применить еще один метод лечения проблематичных электронных сообщений. Если запись не была обработана из-за временно возникшей проблемы (например, БД была открыта в этот момент с монопольным доступом или был недоступен диск, на котором хранится БД), можно попросить программу Access попытаться обработать это сообщение еще раз. Для этого щелкните его правой кнопкой мыши и выберите команду **Export data to Microsoft Office Access** (Экспорт данных в Microsoft Access).

19.5.4. *Управление параметрами вашего сбора данных с помощью электронной почты*

После того как вы отправили сообщение, вся информация об адресате, заданных ему вопросах и т. д. сохраняется в вашей БД.

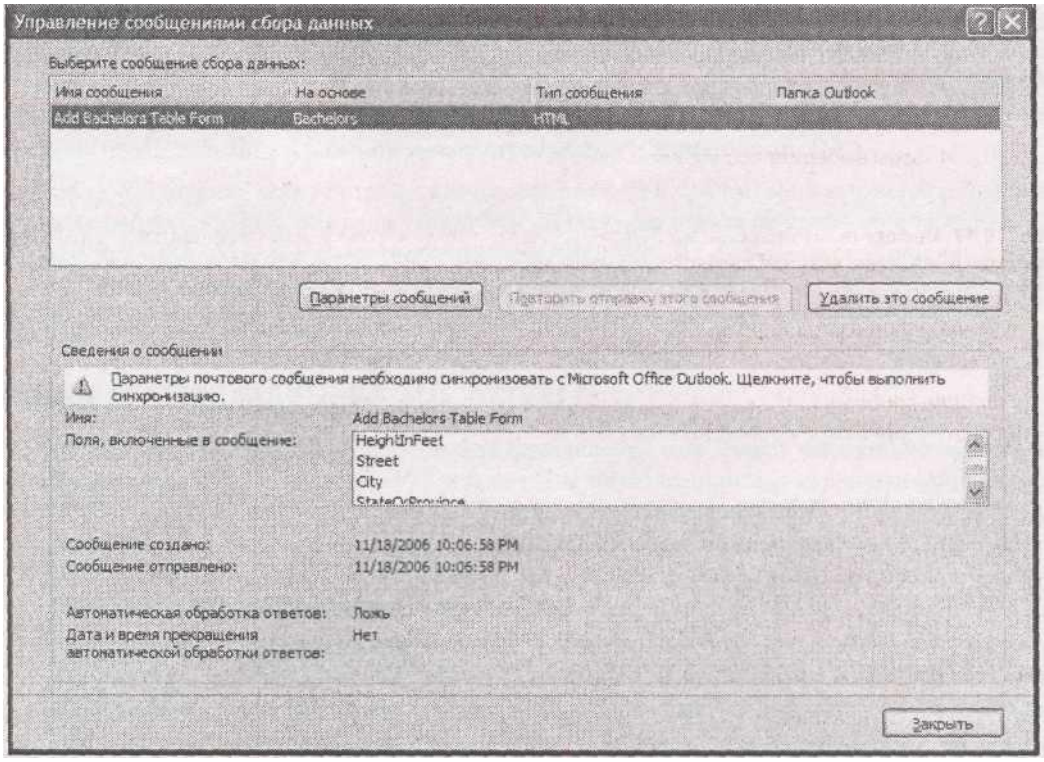


Рис. 19.18. Щелкните мышью кнопку **Параметры сообщений** для вывода на экран диалогового окна, в котором включается или отключается автоматическая обработка и корректируется способ ее функционирования. Щелкните мышью кнопку **Повторить отправку этого сообщения** для отправки вашей формы другой группе пользователей. Наконец, щелкните мышью кнопку **Удалить это сообщение,** когда получите данные, и оно вам больше не требуется

Программе Access нужны эти сведения для того, чтобы знать, как обрабатывать ответы.

Вы можете выполнить несколько действий для обеспечения гладкости процесса. Например, можно отправить сообщения по электронной почте большему числу пользователей, включить или отключить автоматическую обработку ответов и удалить всю информацию электронной почты. Для выполнения любой из этих задач выберите на ленте **Внешние данные → Сбор данных → Управление ответами** (External Data → Collect Data → Manage Replies). На экране появится окно **Управление сообщениями сбора данных** (Manage Data Collection Messages), показанное на рис. 19.18. (Название команды - **Управление ответами -** вводит в заблуждение. На самом деле с ее помощью вы ничего не делаете с ответами. Вместо этого вы настраиваете параметры исходного сообщения электронной почты.)

Подсказка

Даже если сбор данных завершен, нет причины удалять настройки электронной почты. Почему бы не сохранить их на случай, если когда-нибудь в дальнейшем вы решите повторить сбор тех же данных?

20. Глава 20. Подключение Access к SQL Server

В *главе 18* вы рассмотрели, как коллективно использовать вашу заслуживающую награды БД совместно с другими людьми. Для кого-то это блаженное состояние программы Access. Группы пользователей могут сотрудничать, бизнесмены - следить за ежедневным рабочим процессом и все трудятся с удовольствием ныне и присно. Для других многопользовательская поддержка БД - долгая головная боль, поскольку программа Access просто не в состоянии удовлетворить всех, кто хочет внести изменения в одно и то же время.

В этой главе вы познакомитесь с другим подходом, который позволяет преодолеть ограничения Access и совместно использовать ваши БД с гораздо более многочисленными группами пользователей, у которых появится возможность существенно повысить интенсивность такого использования. Для этого нужно подключить программу Access к SQL Server, мощной серверной БД корпорации Microsoft. (Вернитесь *к разд. "Access или SQL Sewer?" во введении,* чтобы вспомнить разницу между серверными и клиентскими БД.)

Такое объединение обеспечит максимум возможностей. Вы сможете применять сверхнадежный процессор БД (каковым является SQL Server) без отказа от удобного пользовательского интерфейса, облегчающего выполнение разнообразных задач (каковой является программа Access). Самое замечательное заключается в том, что можно начать использование версии SQL Server, не потратив ни рубля.

20.1. *Нужно ли переходить на SQL Server?*

Как вы уже знаете, нет общего правила для определения, кто может успешно использовать многопользовательскую поддержку, а кто нет. Вы познакомились с предельными характеристиками - компания Fortune 500 с тысячами сотрудников, вероятно, не сможет использовать многопользовательскую поддержку Access, а команда со штатом 5 человек, занимающаяся дизайном интерьера, не столкнется ни с какими проблемами - но для многих пользователей следует учитывать множество факторов.

Успех коллективного использования зависит от того, сколько пользователей нуждаются в одновременном внесении изменений, какова тенденция изменения объема обновлений, как долго записи остаются в режиме редактирования и каков общий объем хранимых данных (например, огромными полями типа **Memo** и **Вложение** в многопользовательской БД управлять гораздо труднее, чем текстовыми и числовыми полями).

Некоторые факторы находятся вне пределов досягаемости программы Access - например, медленная или ненадежная сеть компании - что может разрушить коллективное использование БД. В *разд. "Как действует многопользовательская поддержка в Access" главы 18* дано несколько важных признаков, позволяющих определить, когда многопользовательской поддержки недостаточно. Однако если БД Access устанавливается для маленькой компании, возможно, следует самостоятельно протестировать режим коллективного использования БД.

Если вы впервые сталкиваетесь с многопользовательским применением БД, некоторые характерные симптомы предупредят вас о том, что БД не справляется. Далее перечислены некоторые ключевые признаки опасности.

- *Ваши корректировки часто перекрываются с чьими-то еще.* В этой ситуации программа Access каждый раз запрашивает, как устранить конфликт *(см. разд. "Обработка конфликтов редактирования" главы 18).* Это самая распространенная проблема, и хотя она не разрушит вашу БД, конфликтующие корректировки могут привести к тому, что ваши изменения будут постоянно искажаться или стираться кем-то, одновременно с вами обращающимся к БД.
- *Вы не можете редактировать нужные вам записи.* Эта проблема возникает, когда программа Access применяет блокировки *(см. разд. "Применение блокировок для предотвращения наложения обновлений" главы 18)* для предотвращения одновременных изменений. Блокировки позволяют Access избежать проблем наложения корректировок, но за это приходится платить. Вся остальная обработка, использующая заблокированную запись, полностью прерывается. И снова эта ситуация не очень опасна - как последствия от приема тройной дозы аспирина.
- *Данные повреждаются.* Эта ситуация одновременно и не столь частая, и очень серьезная.

Несмотря на редкость ее возникновения (она становится все реже с каждым выходом новой версии программы Access), иногда она все еще случается. Вы понимаете, что у вас появилась проблема, когда в поле оказываются искаженные данные или когда Access выводит нелепое сообщение об ошибке (например "Слишком много индексов" ("Too many indexes")) при попытке открыть объект БД. В *разд. "Повреждение данных" главы 18* дается объяснение проблемы и несколько советов, как ее обезопасить.

Примечание

Порча данных может произойти не по вине программы Access. Например, если у некоего сотрудника пропало сетевое соединение в середине процесса сохранения изменения, Access может оставить БД с многопользовательской поддержкой в неопределенном или противоречивом состоянии. Единственный способ предотвращения подобных проблем - применение мощной программы управления БД, работающей на сервере, которая выполняет всю работу. (Именно эта идея лежит в основе SQL Server. Когда используется SQL Server, никто не изменяет БД непосредственно. Вместо этого пользователи создают вежливые запросы к постоянно работающему процессору БД SQL Server, который затем делает эту работу безопасным и контролируемым образом.)

Итак, что же делать, если выяснилось, что многопользовательская поддержка не дает нужных результатов? Самый лучший вариант - использовать мощную серверную программу управления БД, как SQL Server. Применение SQL Server, несомненно, добавит сложностей (вы потратите больше времени на се установку и проверку и наладку ее работы), но она обеспечит нерушимую поддержку безопасного и эффективного совместного использования БД.

20.1.1. *Как работает SQL Server*

Прежде чем вступить на территорию SQL Server, необходимо узнать немного больше о принципах работы этой программы. На рис. 20.1 показано взаимодействие программ SQL Server и Access. В данном примере есть несколько пользователей, одновременно обращающихся к БД SQL Server, причем каждому из них на помощь приходит собственная копия программы Access.

Примечание

Это реальный принцип работы серверных БД, включая конкурирующие программные продукты, такие как Oracle и DB2. Однако у других БД нет отличной интеграции с программой Access, о которой вы узнаете в этой главе, поэтому клиентам приходится использовать другую клиентскую БД (обычно специально разработанное приложение).

Рис. 20.1. SQL Server хранит БД с таблицами (и иногда запросами). БД Access играет роль

клиентской части, содержащей все остальные объекты других типов (отчеты, формы, макросы и модули программного кода)

Данный рисунок может показаться знакомым - в конце концов, в той или иной степени он напоминает принцип действия многопользовательской поддержки в Access. Каждый пользователь получает копию клиентской БД с формами и отчетами, а серверная БД (действительно хранящая данные) находится на другом компьютере (сервере) и доступна всем.

Но есть и существенная разница. На территории SQL Server у отдельных клиентских БД меньше работы. Вместо самостоятельной модификации БД они связываются с программой SQL Server (которая не что иное, как Windows-программа, работающая в фоновом режиме на серверном компьютере). По сути, вы переводите программу Access на менее квалифицированную работу, понижаете в должности. Теперь она отвечает за привлекательный внешний вид, макросы и распечатки, но освобождена от действительно тяжелой работы (добавления, удаления и обновления записей).

На профессиональном уровне.

Важнейшие причины перехода на SQL Server

Есть множество оснований любить SQL Server. Но когда почитатели, фанатично преданные программе Access, изменяют ей, у них, как правило, в голове одна из следующих причин.

- *Масса пользователей.* Как вы помните, Access не слишком хорошо справляется, если нескольким сотням пользователей необходимо совместно использовать один файл БД.
- *Огромное количество данных.* Программа Access не разрешает создавать БД больше 2 Гбайт. Полная версия SQL Server позволяет создавать БД, которые заглатывают весь жесткий диск целиком.
- *Производительность.* По мере роста вашей БД вы, возможно, заметите, что она стала медленнее, чем обычно, извлекать данные. Несмотря на то, что индексы могут помочь до некоторой степени, программа SQL Server способна сделать гораздо больше. Она хранит недавно использовавшуюся информацию в огромном пуле оперативной памяти и раздает ее порциями всем, кто в ней нуждается. Один этот метод сберегает массу времени.
- *Реальная защита.* Как вы узнали из *разд. "Защита базы данных" главы 18,* программа Access не предлагает многоуровневой модели безопасности, позволяющей защитить от конкретных пользователей определенные объекты БД. (Она применялась, но корпорация Microsoft выбросила это средство из Access 2007, поскольку оно было недостаточно безопасным.) У программы SQL Server пуленепробиваемая защита, которая обеспечивает столько уровней, сколько захотите.
- *Транзакции.* В сложных системах управления БД многие задачи состоят из отдельных операций с БД, выполняющихся последовательно, одна за другой. Например, денежный перевод 500 долларов включает два связанных действия: один счет получает кредит 500 долларов, а другой - дебет 500 долларов. Программа SQL Server позволяет поместить эту последовательность действий в транзакцию, что гарантирует отмену транзакции при аварийном завершении одного из указанных действий. Другими словами, даже если ударит молния и ваш сервер перезагрузится в середине обработки транзакции, SQL Server сумеет восстановить систему до состояния, в котором она была перед началом перевода денег (и вы никогда не лишитесь 500 долларов в мгновение ока).

Несмотря на то, что все эти характеристики замечательны, большая их часть не рассматривается в данной книге. Для того чтобы узнать больше, нужно найти книгу, посвященную SQL Server.

20.1.2. *Более дешевая версия SQL Server*

В данный момент вас, вероятно, интересует, сколько стоит часть, расположенная в центре рис. 20.1 - процессор БД SQL Server. Корпорация Microsoft устанавливает цену с помощью сложной схемы лицензирования, которая повышает цену в зависимости от числа одновременных пользователей БД. Обычно она достигает тысяч долларов и нередко для больших компаний доходит до 20 000 долларов и больше в год. Но прежде чем с досадой пропустить эту главу, следует узнать кое-что: абсолютно бесплатная версия SQL Server выставлена и ждет вас. Как ни странно, у нее почти та же мощность, что и у программы, стоящей тысячи долларов

и требующей от вас заложить своего первенца корпорации Microsoft.

Эта версия называется SQL Server 2005 Express Edition, и в следующем разделе вы узнаете, как ее устанавливать. Если сравнить эту версию с полной, выяснится, что у нее есть три следующие ограничения.

■ Данная версия поддерживает только один процессор (компьютерный процессор). Дополнительные ЦПУ создаются для сверхмощных компьютеров, и это ограничение мешает программе SQL Server Express быть столь же мощной, как ее старшая сестра, версия не Express.

■ Версия Express может использовать только 1 Гбайт оперативной памяти. Если на вашем сервере больший объем памяти, используйте его для чего-нибудь еще.

■ Максимальный объем создаваемой вами БД - 4 Гбайт. В этом нет проблемы, т. к. программа Access ограничивает объем файлов БД 2 Гбайт. Если избегать хранения изображений и других данных большого объема, на какое-то время этого будет достаточно.

Большего внимания заслуживает то, что нет денежных расходов. SQL Server Express - это полнофункциональная версия программы SQL Server с точно таким же мощным процессором БД или движком. Если необходимо преобразовать вашу БД Access, данная программа - отличный выбор.

Примечание

SQL Server Express предоставляет замечательные средства, помогающие создавать таблицы и управлять вашими БД. Несмотря на то, что можно загрузить из Интернета бесплатное средство управления с сайта корпорации Microsoft (рис. 20.2), у вас для этого уже есть программа Access. Это клиентская часть с большими возможностями, способная выполнить все, что вам нужно.

Часто задаваемые вопрос.

Можно ли доверять корпорации Microsoft?

Почему корпорация Microsoft отдает что бы то ни было даром?

Смекалистые компьютерные пользователи подозрительно относятся ко всему, что кажется слишком хорошим, чтобы быть правдой. Они боятся, что предложение SQL Server корпорацией Microsoft может быть хитроумной недобросовестной торговой тактикой - другими словами, достаточной, чтобы привлечь вас к применению SQL Server, но не удовлетворяющей ваши потребности.

К счастью, для беспокойства нет оснований. Если вы решите использовать программу SQL Server Express, то с успехом можете оставаться верным ей годами и не ощущать необходимости в переходе на коммерческую версию.

Но почему же Microsoft предлагает программное обеспечение, которое не принесет никакого дохода? Все просто - они рассчитывают на крупный улов. Маленькая компания может начать с версии SQL Server Express, а затем вырасти в большое предприятие, готовое заплатить дополнительную сумму за более мощную версию программы. Это особенно справедливо в отношении компании, применяющей SQL Server для снабжения данными так называемого Web-приложения (сайт интернет-торговли, например).

Если этот Web-сайт станет вторым eBay, компании, обслуживающей его, потребуется серьезная рабочая лошадка (например, серверный компьютер с несколькими ЦПУ и тоннами памяти). Для поддержки такого оборудования они вынуждены будут перейти на полную версию программы SQL Server.

И, наконец, предложение бесплатной версии SQL Server помогает привлечь большее количество людей к разработке высококлассных систем БД с помощью программы SQL Server. Технические специалисты, возможно, познакомятся с SQL Server Express и полюбят эту программу, а потом порекомендуют компаниям-толстосумам ее полную версию.

20.2. *Приступая к работе с SQL Server 2005 Express*

Прежде чем начать применять SQL Server Express, необходимо установить программу. Этот процесс чрезвычайно прост, но продолжителен и включает загрузку из Интернета нескольких очень больших файлов: один объемом 22 Мбайт, а другой - 54 Мбайт. Если вы все еще пользуетесь модемным соединением, это займет у вас несколько отсчитываемых в уме часов.

Примечание

Все, что вы узнаете в данной главе о программах Access и SQL Server Express, также применимо к полной версии SQL Server. Но если у вас есть эта версия, вам, конечно же, не нужен этап загрузки из Интернета, описанный здесь, вместо него вы можете вставить ваш установочный DVD-диск и начать установку.

20.2.1. *Установка SQL Server Express*

SQL Server устанавливают на компьютер, на котором планируется поместить серверную БД. Обычно это компьютер в сети, который никто не использует для другой работы. (Если кто-то работает на этом компьютере, есть риск, что они выключат его, перезагрузят, займутся другой работой или сделают что-то, что повлияет на возможность получения нужной информации всеми остальными пользователями.) На компьютеры, применяющие программу Access для работы с клиентской БД, устанавливать SQL Server не нужно, но конечно на них должна быть копия Access.

Однако если вы до сих пор продолжаете разрабатывать и настраивать вашу БД, возможно, вам захочется сначала опробовать ее на собственном компьютере. В этом случае программу SQL Server следует установить на ваш компьютер. Затем, когда вы будете готовы к совместному использованию БД, вы установите SQL Server на серверный компьютер и переместите туда вашу БД (как описано в *главе 21).* На самом деле, если вы не знаете программу SQL Server, может быть лучше всего начать с ее установки на собственный компьютер. Вам нужно будет настроить несколько перегруженных деталями установочных параметров для того, чтобы предоставить другим пользователям доступ к SQL Server с других компьютеров. Возможно, появится желание увидеть, как все работает до того, как начнете возиться с упомянутыми деталями.

У программы SQL Server очень скромные требования к системе. (Поразительно, но они менее жесткие, чем требования к работе программы Access.) Точные спецификации можно найти на сайте www.microsoft.com/sql/editions/express/sysreqs.mspx. Вы увидите, что программу SQL Server можно запустить на любом более или менее современном компьютере, но придется убедиться в том, что у вашей операционной системы есть последние обновления и пакеты исправлений. Например, на компьютерах под управлением Windows XP должен быть установлен Service Pack 2.

Подсказка

Для проверки наличия последних обновлений ОС щелкните мышью кнопку **Пуск** (Start) и выберите **Обновление Windows** (Windows Update).

После того как выбрано место установки программы SQL Server, и вы убедились в том, что компьютер может ее поддерживать, выполните следующие действия.

1. Откройте любимый Web-обозреватель и перейдите на сайт загрузки .NET.

Перед установкой SQL Server необходимо установить используемый программой компонент .NET Framework 2.0. Его можно найти с помощью средств поиска на Web-сайте **http://msdn.microsoft.com/netframework.** Или лучше использовать секретный укороченный URL-адрес **http://tinyurl.com/drj86.**

Примечание

Если вы работаете под управлением ОС Windows Vista, у вас уже есть компонент .NET Framework 2.0. Даже если у вас нет Windows Vista, другое приложение, возможно, уже установило его на ваш компьютер. Если вы предполагаете, что он установлен, перейдите в Панели управления (Control Panel) в группу **Администрирование** (Administrative Tools) и поищите пиктограмму, названную **Настройка конфигурации Microsoft .NET Framework 2.0** (Microsoft .NET Framework 2.0 Configuration). Если вы ее нашли, поздравляю - можно переходить непосредственно к пункту 4.

2. После того как компонент .NET Framework 2.0 найден, установите его. Файл установки

довольно большой, около 22 Мбайт.

3. После завершения загрузки файла установки запустите его и с помощью мыши пройдите через все этапы, предлагаемые мастером установки.

Не беспокойтесь - никаких решений принимать не придется.

4. После завершения установки перейдите на Web-страницу **www.microsoft.com/sql/editions/express.**

На этой странице много сведений о программе SQL Server Express. Найдите ссылку, которая позволяет загрузить SQL Server Express и затем щелкните ее кнопкой мыши. Она помечена текстом "Get SQL Server Express" ("Загрузить и установить").

5. Щелкните в стороне кнопкой мыши, чтобы найти загружаемый файл SQL Server Express (рис. 20.2).

В процессе установки корпорация Microsoft предложит вам зарегистрироваться у нее. Если вы регистрируетесь, то получаете новости от корпорации Microsoft (которые полезны, если вы хотите узнавать о последних обновлениях SQL Server). Но это совсем не обязательно.

6. Когда найдете ссылку на загружаемый файл для SQL Server (см. рис. 20.2), щелкните ее кнопкой мыши и установите программу.

Для загрузки SQL Server предлагается огромный файл, 54 Мбайт.

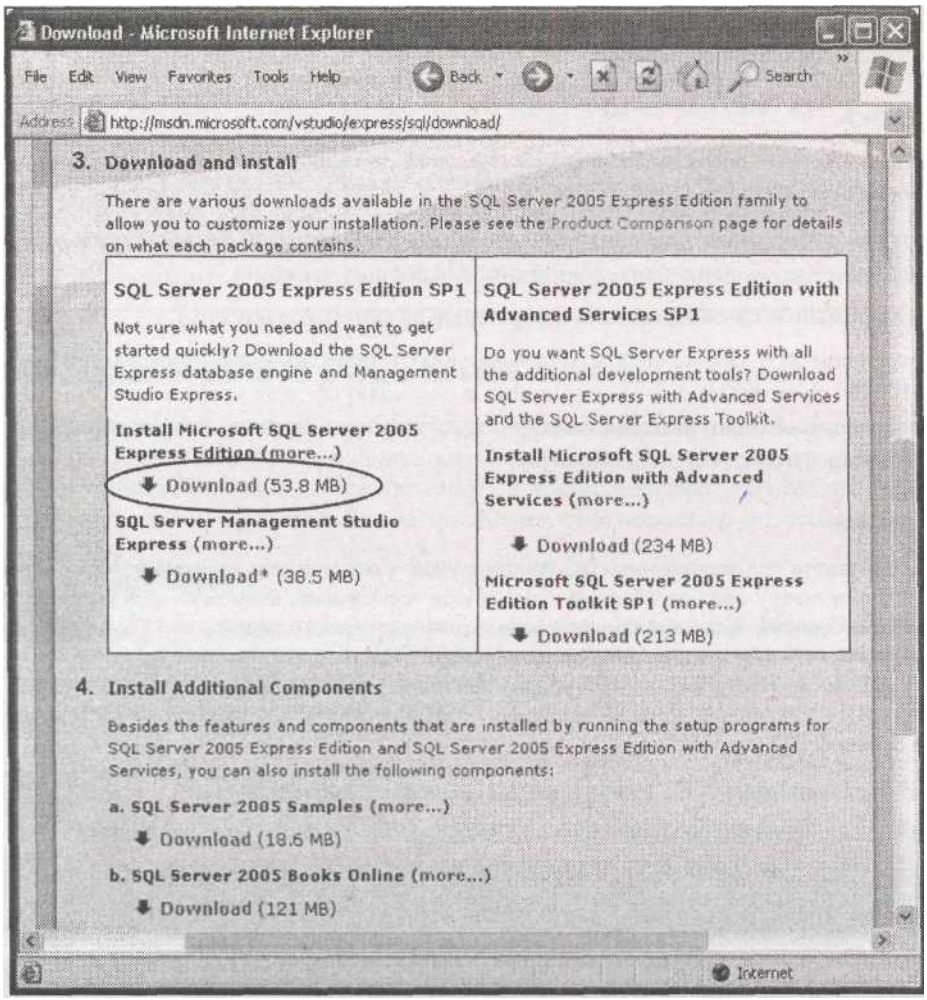


Рис. 20.2. У вас есть варианты. Вы можете загрузить не только SQL Server 2005 Express (обведенная ссылка), можно также получить отличное средство управления, названное SQL Server Management Studio (ссылка расположена ниже), некоторые примеры и напичканный информацией файл справки, названный SQL Server 2005 Books Online (самая нижняя ссылка)

7. После того как загрузка файла установки программы завершится, запустите его.

Прежде чем установка начнется, нужно пройти несколько простых шагов. Вот чего вам следует ждать.

□ Прежде чем устанавливать что бы то ни было, программа установки проверит конфигурацию системы. Она обследует ваш компьютер и затем сообщит, подходит ли он для

установки SQL Server (рис. 20.3).

□ Программа попросит ввести ваше имя. Эта часть установки очень типична, но следует оставить установленным флажок **Hide advanced configuration options** (Скрыть дополнительные параметры конфигурации), чтобы не выводились низкоуровневые параметры, которые вы не хотите менять.

Рис. 20.3. На этом компьютере может выполняться программа SQL Server, но выводится предупреждающее сообщение, уведомляющее о том, что у устаревшего процессора Pentium III недостаточно мощности для обеспечения оптимальной производительности

- Вам будет предложено выбрать средства, которые хотите установить. Оставьте стандартные установки - в этом случае инсталлируется все, что вам нужно.
- Следующий вопрос касается автоматической отправки корпорации Microsoft сообщений об ошибках. Не ждите, что они помогут вам. Это средство разработано только для того, чтобы помочь Microsoft улучшить будущие версии SQL Server за счет выявления проблем, которые в настоящее время нервируют пользователей программы.

И в конце процесса установки вам будет предложена более сложная часть. Программа SQL Server спросит о типе безопасности, который вы хотите применять.

8. Выберите ваш режим аутентификации (рис. 20.4). У вас есть два варианта.
   - Выбор переключателя **Windows Authentication Mode** (Проверка подлинности Windows) означает, что программа SQL Server принимает решение о предоставлении пользователю разрешения на пользование БД на основе персональной пользовательской учетной записи в ОС Windows. Это лучший и самый безопасный подход. К сожалению, он связан с дополнительной работой в дальнейшем, гарантирующей другим пользователям получение разрешений на использование вашей БД *(см. следующий раздел)*
     - Выбор переключателя **Mixed Mode...** (Смешанный режим...) означает, что программа SQL Server разрешает пользователям обращаться к БД, если у них есть корректная учетная запись Windows (как было описано ранее) или они могут предоставить имя пользователя и пароль, определенные вами. Если выбирается **Mixed Mode...** (Смешанный режим...), вы должны предоставить специальную учетную запись программы SQL Server, называемую sa (system administrator, системный администратор). Любой пользователь, зарегистрировавшийся с этим именем пользователя и паролем, получает полный контроль над всеми БД, хранящимися на сервере. Смешанный режим не устраняет трудности дополнительной настройки, но еще и не столь безопасен. (Например, потому, что вы вынуждены распространить в офисе эту информацию имя пользователя и пароль, которые должны будут применять пользователи.)

Примечание

Если вы не знакомы с понятиями "пользователи" и "группы" ОС Windows и нет сетевого администратора, который бы помог вам, лучше выбрать вариант **Mixed Mode...** (Смешанный режим...). Это не слишком безопасный подход, но это единственный легкий способ разрешить другим людям зарегистрироваться в БД *(см. следующий раздел)*.

Рис. 20.4. Вариант **Windows Authentication Mode** обеспечивает наилучшую безопасность. Но он подразумевает некоторую настройку, позволяющую сообщить программе SQL Server о тех, кому она может доверять

9. Начиная с этого момента, установка продолжительна, но незатейлива. Сейчас самое время взять чашечку кофе.

После завершения установки следует снова запустить обновление Windows (Windows Update) (щелкните мышью кнопку **Пуск** (Start) и затем выберите команду **Обновление Windows** (Windows Update)). Этот шаг необходим, т. к. у программы SQL Server Express есть два пакета исправлений (сведения на момент написания этих строк), и средство **Обновление Windows** (Windows Update) устанавливает их автоматически.

Примечание

Пакеты исправлений программы SQL Server Express очень важны для пользователей ОС Windows Vista. Без них сверхстрогая модель безопасности Windows Vista запретит вам делать что бы то ни было с вашими БД SQL Server.

Когда второй круг обновлений пройден, наконец, можно приступать к работе. Если было решено установить программу SQL Server на ваш компьютер, вы можете начать использовать ее прямо сейчас. Перейдите к *разд. "Создание БД SQL Server вручную" далее в этой главе,* чтобы вернуться в доброжелательные объятья Access.

Если же вы решили установить SQL Server на другой компьютер, для завершения установки вам нужно выполнить еще несколько дополнительных действий. Подробности приведены в следующем разделе.

20.2.2. *Подключение SQL Server к сети*

Когда программа SQL Server устанавливается впервые, она может использоваться только на текущем компьютере, т. е. другие компьютеры не могут зарегистрироваться на сервере и начать использовать любые его БД.

На первый взгляд это сводит на нет основную цель применения SQL Server. (На второй взгляд тоже.) Однако корпорация Microsoft знает, что если они выпускают программный продукт, широко открытый остальному миру с его хакерами, взломщиками и всякого рода околокомпьютерными злоумышленниками, кто-то где-нибудь да достучится до него. По этой причине программа SQL Server применяет самый безопасный подход - она ограничивает доступ к себе текущим компьютером до тех пор, пока вы не дадите ей разрешение принимать внешние вызовы.

Для того чтобы открыть SQL Server для внешнего мира, необходимо изменить два конфигурационных параметра. Если вы работаете под управлением ОС Windows XP или Vista, также нужно настроить брандмауэр Windows, чтобы он пропускал SQL Server. Самые свежие инструкции по установке можно найти в статье из Базы знаний (Knowledge Base) на Web-странице **http://support.microsoft.com/kb/914277.**

После внесения этих изменений другие пользователи, наконец, смогут связаться с программой SQL Server и попробовать зарегистрироваться. Однако дело еще не сделано. SQL Server все еще может отказать пользователям, т. к. программа не предоставляет доступа тем, кому не доверяет.

Кому же доверяет SQL Server Express? Далее приведена вся подноготная.

■ Когда программа SQL Server устанавливается впервые, она настроена на доверие администратору компьютера, на котором проходила установка. (Технически быть администратором означает, что ваша учетная запись Windows принадлежит группе **Администраторы** (Administrators).)

■ Если вы при установке выбрали аутентификацию в режиме **Mixed mode...** (Смешанный режим...), доступ будет разрешен тому, кто предоставит имя пользователя и пароль sa, заданные во время установки.

Если вы хотите, чтобы программа SQL Server доверяла другим пользователям, придется выполнить кое-что еще. Обычно вы должны гарантировать, что каждый пользователь, нуждающийся в работе программы SQL Server, принадлежит одной из Windows-групп *(группа* - это коллекция пользователей, имеющая осмысленное название, например, **Гости** (Guests), Администраторы (Administrators) ЛюбителиБД (DatabaseLovers) и т. д.). Создание группы - это задача установки ОС Windows, поэтому для ее выполнения необходимо проконсультироваться с сетевым администратором. После того как дело сделано, нужно сообщить программе SQL Server о необходимости доверять вашей группе. Для этого можно применять несколько методов, но легче всего загрузить из Интернета бесплатное программное средство SQL Server Management Studio (см. рис. 20.2). Дополнительные сведения можно найти в справке SQL Server Management Studio Help (рассчитанной на технически грамотных пользователей) или поискать книгу, посвященную администрированию в программе SQL Server.

Примечание

В данный момент вы, возможно, удивлены тем, что программа SQL Server так усложняет жизнь. Причина кроется в том, что она разработана в расчете на максимальную гибкость. Когда вы просто пытаетесь разрешить пользователям работать с вашей БД, модель безопасности программы кажется до нелепости сложной, но она необходима, когда вам нужно четко управлять разрешениями для определенных действий с БД, предоставляемыми разным пользователям.

20.3. *Создание БД SQL Server*

Вы прошли сквозь долгий и изнурительный процесс установки. Сейчас пора пожинать плоды вашего труда и приступить к созданию вашей первой БД SQL Server.

Программа Access предоставляет два способа создания БД SQL Server:

■ можно взять обычную БД Access и преобразовать ее. Программа создает нужные вам таблицы и передает все данные в программу SQL Server;

■ можно использовать программу Access для создания с нуля новой БД SQL Server.

Как правило, преобразование - самый легкий вариант, поскольку разрешает разработать таблицы с помощью хорошо знакомых вам средств, а затем передать данные. (Как вы увидите,

создание в программе Access таблицы SQL Server аналогично, но чуть отличается от процесса создания обычной таблицы Access. Это похоже на чувство, которое возникает, когда вы проснулись утром и обнаружили, что кто-то перевернул вашу коллекцию CD-дисков в ящике для белья. Ничего не пропало, но все не там, где вы привыкли находить.)

У метода прямого создания тоже есть свои достоинства. Важнее всего то, что он предоставляет больший контроль, поскольку лишен этапа преобразования. БД Access неточно соответствуют БД SQL Server (например, используемые типы данных похожи, но слегка отличаются). Если создавать БД с самого начала в программе SQL Server, вы избежите любых потенциальных проблем преобразования.

В следующих разделах вы познакомитесь с обоими подходами.

20.3.1. *Преобразование БД*

У программы Access есть удобный Мастер преобразования в формат SQL Server (Upsizing wizard), который превращает любую БД Access в БД SQL Server. Этот мастер аналогичен мастеру разделения БД - когда вы закончите, у вас будет серверная и клиентская части БД. Клиентская часть - файл Access, содержащий ваши формы, отчеты, макросы и программный код. Серверная часть - это данные (и обычно запросы), которые хранятся в надежных руках программы SQL Server.

Вот как взять в оборот Мастер преобразования.

1. Откройте БД, которую хотите преобразовать.

Выберите на ленте **Database Tools → Move Data → SQL Server** (Работа с базами данных → Переместить данные → SQL-Server)[1].

2. На экране появится первое окно Мастера преобразования (рис. 20.5).

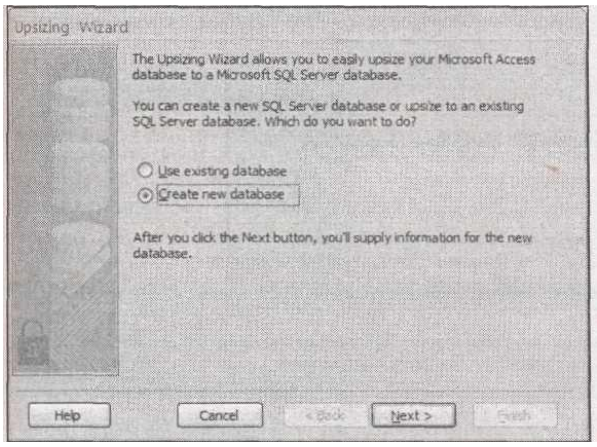


Рис. 20.5. Обычно мастер преобразования применяется для переноса данных из файла БД Access в замечательную новую БД SQL Server

3. Выберите переключатель **Create new database** (создать базу данных) и затем нажмите кнопку **Next** (Далее).

Если вы уже создали БД SQL Server на компьютере с программой SQL Server (например, с помощью другого средства управления БД), можно было бы выбрать переключатель **Use existing database** (использовать существующую базу данных) для переноса ваших таблиц Access в эту БД. Но почти всегда имеет смысл создавать новую БД. Помимо всего прочего процессор БД, такой как SQL Server, может хранить практически неограниченное число БД.

4. На следующем этапе (рис. 20.6) необходимо сообщить программе Access, где искать сервер вашей БД. Сначала введите в поле в верхней части окна имя сервера.

Имя сервера состоит из имени компьютера, на котором запущена программа SQL Server, за которым следует обратный слэш, а затем слово SQLEXPRESS. Если компьютер, выполняющий программу SQL Server, назван FudgeServer, вы найдете свою БД на FudgeServer\SQLEXPRESS. Если вы подключаетесь к полной версии программы SQL

Server(не к версии Express), вторая часть имени, как правило, не нужна, достаточно Fudge-Server. Обратитесь за помощью к вашему администратору БД.

[1] *В данной главе рассматривается нерусифицированная версия программы Access. - Ред.*

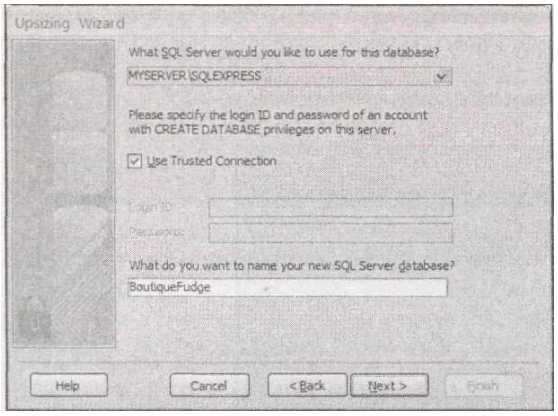


Рис. 20.6. В этом окне программа Access собирается подключиться к компьютеру, названному MYSERVER, на котором выполняется программа SQL Server Express

Примечание

Для выяснения имени вашего компьютера, найдите пиктограмму **Мой компьютер** (My Computer) (на вашем рабочем столе или в программе Проводник (Windows Explorer)), щелкните ее правой кнопкой мыши и выберите строку Свойства (Properties). Далее перейдите на вкладку **Имя компьютера** (Computer Name). Вы увидите имя вашего компьютера и полезную кнопку **Изменить...** (Change), которой можно воспользоваться для изменения этого имени.

5. Оставьте установленным флажок **Use Trusted Connection** (Доверительное соединение).

Этот шаг сообщает программе Access о необходимости подключения с помощью учетной записи Windows. Однако если вы хотите подключиться с другим именем пользователя и паролем, сбросьте упомянутый флажок и введите нужную информацию в поля ввода, расположенные в нижней части окна.

6. Введите имя вашей БД и щелкните мышью кнопку **Next** (Далее).

Пользуйтесь теми же правилами, которым вы следовали при именовании объектов БД - имена должны быть короткими, без пробелов и знаков пунктуации.

Примечание

Когда создается БД, программа SQL Server не сообщает вам имя реального файла БД (как правило, это несколько файлов). Имена файлов не так важны. Вам следует знать имя, которым вы наградили вашу БД (например, BoutiqueFudge). За кадром программа SQL Server хранит ваши данные в соответствующих файлах, поэтому вам беспокоиться об их именах, не стоит.

7. Выберите таблицы, которые хотите перенести в вашу БД (рис. 20.7), и щелкните мышью кнопку **Next** (Далее).

Программа Access не разрешит выбрать запросы, которые вы хотите перенести в БД. Вместо этого она перенесет все запросы, связанные с выбранными вами таблицами. Другие объекты, такие как формы и отчеты, никогда не переносятся.

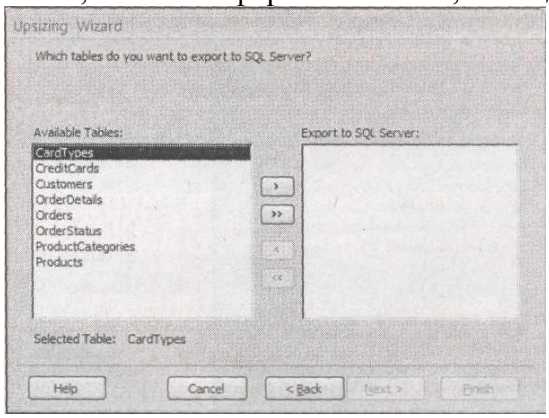


Рис. 20.7. Щелкните мышью кнопку > для переноса одной таблицы в список **Export to SQL Server** или кнопку » для переноса всех таблиц

8. Следующие этапы позволят управлять способом создания ваших таблиц в программе SQL Server (рис. 20.8). Измените нужные параметры и щелкните мышью кнопку **Next** (Далее).

В *главе 2* вы узнали об индексах, а в *главе 4* познакомились со значениями по умолчанию и условиями на значения. Обычно, если вы потратили время на определение этих составляющих БД в программе Access, вам захочется сохранить их и в экспортируемых в SQL Server таблицах, поэтому оставьте соответствующие флажки установленными. Далее перечислены остальные параметры, которые можно использовать.

- Флажок **Table relationships** (связи таблиц). Оставьте его установленным, что гарантирует сохранение в программе SQL Server связей между таблицами, которые вы определили. В этом случае у вас есть два варианта для выбора. Переключатель **Use DRI** (DRI), где DRI - сокращение для data relational integrity (ссылочная целостность данных), - обычная практика, обеспечивающая невозможность создания подчиненных или дочерних записей, ссылающихся на несуществующую главную или родительскую запись. Переключатель **Use triggers** (Триггеры) заставляет программу SQL Server применять менее распространенные средства каскадного удаления и каскадного обновления записей.
- Раскрывающийся список **Add timestamp fields to tables?** (Добавлять поля штампа времени в таблицы?) позволяет создавать поле с текущими датой и временем - дополнительное поле, основная цель которого фиксировать момент времени внесения изменения. Данное поле иногда применяется для запрета накладывающихся изменений, т. к. оно позволяет проверить, не сделал ли кто-то еще изменений в записи с тех пор, как вы последний раз ее просматривали. Обычно поля с датой и временем не добавляются. Лучше вставить их в таблицу позже, когда вы решите, что хотите воспользоваться этой возможностью.

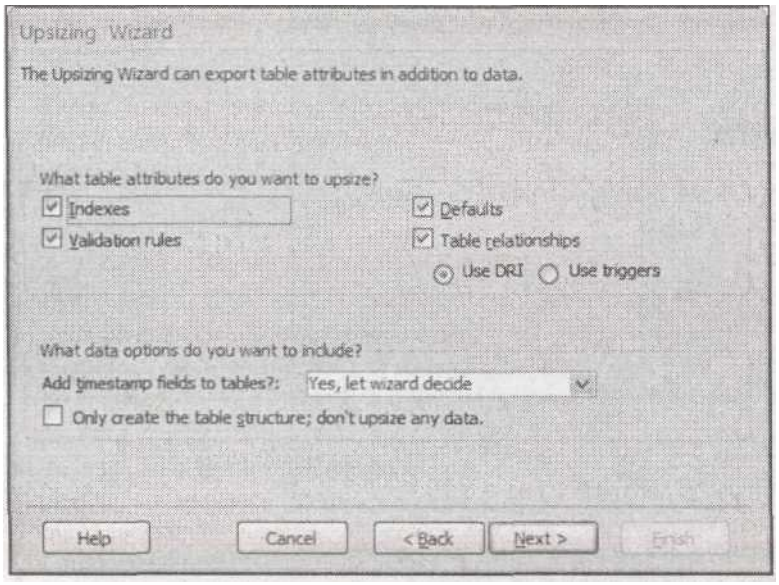


Рис. 20.8. Если вас пугают все параметры в этом окне, просто щелкните мышью кнопку **Next,** чтобы оставить все, как есть. Стандартные установки - как правило, то, что вам нужно

- Флажок **Only create the table structure...** (создать только структуру таблицы...). С помощью этого параметра можно создать все таблицы в программе SQL Server, но не переносить данные. Этот вариант удобен, если в данный момент файл БД Access тестируется и в нем много не реальных, а тестовых данных.

9. На последнем этапе решается, как поступить с исходным файлом Access. Обычно выбирается вариант **Create a new Access client/server application** (Создать новое приложение Access).

Далее описывается назначение каждого из трех возможных вариантов.

- Вариант **Create a new Access client/server application** (создать новое приложение Access). Если выбран этот вариант, создается файл Access специального типа, назы-

ваемый проектом Access. Вы заметите разницу, т. к. у этого файла расширение adp. Данный файл - клиентская БД, содержащая все ваши отчеты, формы, программный код и специальный набор ссылок, позволяющих взаимодействовать с таблицами, находящимися на сервере. (Исходный файл БД Access остается, но вы, наверное, просто удалите его, потому что все ваши данные есть в программе SQL Server.)

- Вариант **Link SQL Server tables to existing application** (связать таблицы SQL Server с существующим приложением). Этот вариант аналогичен созданию проекта Access за исключением того, что он модифицирует файл текущей БД, превращая его в клиентскую БД. Все таблицы, которые только что были перенесены, переименовываются добавлением в конец имени слова "local" (локальная) (таким образом, таблица **Products** становится таблицей **Products_iocal).** Кроме того, вы получите новый набор связанных таблиц с исходными именами. Каждая связанная таблица позволяет обращаться к соответствующей таблице в программе SQL Server. После того, как убедитесь, что связи таблиц действуют, вероятно, стоит удалить локальные таблицы, чтобы не путаться. (Дополнительную информацию о связанных таблицах *см. в примечании "На профессиональном уровне. Проекты Access по сравнению со связанными таблицами" далее в этом разделе.)*
- Вариант **No application changes** (не изменять приложение). При выборе этого варианта данные переносятся, но ваша БД Access не изменяется. Это означает наличие двух наборов данных: один набор в файле БД Access и другой в файле SQL Server (который можно просматривать и корректировать в программе Access). Такой результат обычно не то, что вам нужно.

10. Щелкните мышью кнопку Next (Далее).

Если создается проект Access (как описано в предыдущем пункте), программа Access запрашивает, хотите ли вы открыть новый проект клиентской БД прямо сейчас или сохранить открытой старую БД. Обычно появляется желание открыть новый файл, чтобы можно было в нем работать.

11. Щелкните мышью кнопку **Finish** (Готово) для того, чтобы начать процесс преобразования.

Программа Access подключается к SQL Server и начинает переносить все данные. Во время ее работы вы видите индикатор выполнения процесса (рис. 20.9).

Когда Access закончит, программа выведет на экран отчет со сводными данными о завершенном процессе. Отчет можно напечатать или выбрать на ленте **Print Preview → Close Preview → Close Print Preview** (Предварительный просмотр → Закрыть → Закрыть окно предварительного просмотра) для возврата к вашей БД.

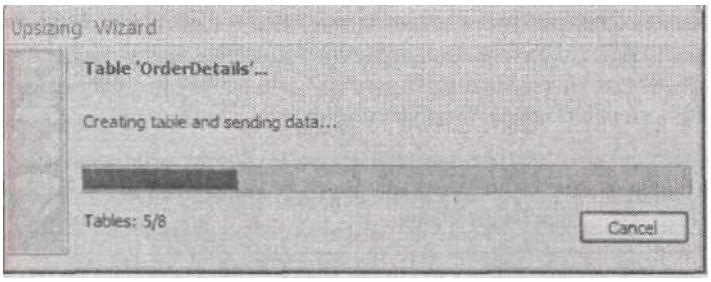


Рис. 20.9. Процесс может занять какое-то время, зависящее от количества данных

Процесс преобразования обычно выполняется без проблем. Но не все данные переживут преобразование. Далее перечислены ингредиенты, которые могут быть потеряны.

- *Поля гиперссылок.* Они превращаются в обычные текстовые поля.
- *Поля типа Вложение.* Они становятся обычными текстовыми, полями, но в этих полях содержатся только имена файлов, которые вы поместили в БД как вложения.
- *Многозначные поля.* Вы получите список значений, разделенных точкой с запятой. Такой результат содержит верные данные, но в форме, которую не может использовать программа SQL Server. Любые отношения или запросы, использующие эти данные, потерпят неудачу.

Если создается проект Access (а не просто связанные таблицы), ваши запросы тоже преобразуются. В результате потеряются следующие составляющие, поскольку программа SQL Server их не поддерживает:

■ запросы на изменение *(см. главу 8)* с параметрами;
■ перекрестные запросы;
■ запросы, ссылающиеся на значения, хранящиеся в форме. (Форма находится в вашей клиентской БД, поэтому после переноса запроса у него не будет доступа к данной информации.);
■ нестандартные запросы, которые были созданы в режиме SQL-команды, а не в Конструкторе, например запросы на объединение.

На профессиональном уровне.
Проекты Access по сравнению со связанными таблицами

Проекты Access и связанные таблицы выглядят как два похожих варианта. Они позволяют создавать клиентские БД Access, работающие с данными, хранящимися в БД SQL Server. Но у этих вариантов есть несколько важных различий.

• *Обновляемостъ.* Несмотря на то, что для изменения данных можно применять оба варианта, структуру связанной таблицы изменять нельзя. Таким образом, если нужно добавить поля, установить отношения и т. д., необходимо использовать проект Access.
• *Обработка запросов.* Когда создается проект Access, ваши запросы преобразуются в объекты SQL Server и хранятся на сервере. Когда создается связанная таблица, запросы остаются в клиентской БД. Трудно сказать, какой подход лучше. Обычно у проектов Access более высокая производительность обработки запроса, поскольку большую часть работы выполняет сервер. (Это особенно справедливо, если выполняется запрос, который отбирает несколько записей из большой таблицы.) Связанные таблицы позволяют продолжать применять хорошо знакомый конструктор запросов программы
Access и устраняют возможность ошибок преобразования запроса.
• *Тип файла.* Связанные таблицы помещаются в обычный файл Access. (Вы даже можете поместить их рядом с обычными таблицами.) Проект Access всегда должен сохраняться в специальном файле с расширением adp.

Вы узнали о связанных таблицах в *главе 19.* В оставшейся части данной главы предполагается, что создается проект Access.

Подсказка

Хотите вернуться назад другим путем и переслать данные из SQL Server в Access? Если так, воспользуйтесь средствами импорта и экспорта в программе Access, о которых вы узнали в *главе 19.*

После завершения процесса преобразования можно продолжать работать с таблицами так же, как вы делали это прежде. Например, можно открывать ваши таблицы в **Режиме таблицы,** редактировать записи и применять ваши формы и отчеты. Разница в том, что теперь ваша копия Access связывается с программой SQL Server для получения нужных данных и вносит изменения.

Во время редактирования данных и применения форм и отчетов вы не заметите разницы между новой преобразованной БД и вашей исходной БД (за исключением, возможно, некоторого снижения быстродействия). Однако разница станет заметной, когда вы создадите новую таблицу или запрос, либо когда вы измените дизайн существующей таблицы или запроса. Все потому, что таблицы и запросы на самом деле хранятся в БД SQL Server, а эти БД спроектированы несколько иначе, чем их дубликаты в Access.

Вы начнете рассматривать различия в *разд. "Добавление объектов в БД SQL Server" далее в этой главе.* Но сначала стоит рассмотреть, как управлять только что созданной БД SQL Server.

20.3.2. *Управление вашей БД*

На территории Access легко удалять, перемещать, создавать резервную копию БД. Нужно всего лишь найти соответствующий файл с расширением accdb и использовать средство управления файлами, например программу Проводник ОС Windows.

Программа SQL Server работает иначе. Как вы уже узнали, она обрабатывает файлы за кадром, не раскрывая имен этих файлов. Но даже если вы знаете, где искать файлы БД, вы мало что сможете с ними сделать. Например, если нужно перенести БД SQL Server с одного

сервера на другой, простая операция вырезания и вставки в Проводнике не сработает. Исходный сервер будет продолжать искать БД, которую вы перенесли, а новый сервер будет продолжать ее игнорировать.

Если вы хотите выполнять задачи управления, такие как удаление, перенос или копирование БД, необходимо работать с программой SQL Server, т. к. она может перемещать файлы и изменять каталог БД. Один из вариантов - загрузить из Интернета бесплатное программное средство SQL Server Management Studio (см. рис. 20.2), которое поможет выполнять широкий круг задач администрирования. (Например, это средство понадобится, если у вас грандиозные планы, и вы начинаете настраивать параметры безопасности SQL Server для того, чтобы дать разрешения одним пользователям и отказать другим.) Однако самые распространенные задачи управления можно выполнять и прямо в программе Access. Секрет кроется в применении кнопки **Office → Server** (Office → Сервер) (меню **Server** (Сервер) появляется, только если открыт проект Access).

У вас в меню есть следующие варианты.

■ **Connection** (Подключение). Выводит на экран диалоговое окно **Data Link Properties** (Свойство связи с данными), в котором можно изменять параметры подключения для вашего файла проекта Access. Обычно к этому окну обращаются при возникновении проблем подключения к вашей БД SQL Server. Например, если кто-то переместил или переименовал БД SQL Server или изменили имя пользователя и пароль, нужные вам для регистрации, именно в этом окне можно обновить ваши параметры. Задайте новый сервер, новое имя БД или сведения о пароле и щелкните мышью кнопку ОК для повторного подключения.

Подсказка

Если вы открыли файл проекта Access и не видите никаких таблиц, а в строке заголовка появляется сообщение "Disconnected" ("Нет соединения"), программа Access не смогла подключиться к вашей БД. Если имя БД или сервера были изменены недавно, выберите кнопку **Office → Server → Connection** (Office → Сервер → Подключение) для устранения проблем.

■ **Server Properties** (Свойства сервера). Отображает диалоговое окно с базовыми сведениями о вашей БД, включая программу, которая выполняется (SQL Server), ее версию (у SQL Server 2005 номер версии 9), расположение сервера и текущей БД. В этом окне нельзя изменять никакие данные.

■ **Link Tables** (Связь с таблицами). Позволяет добавить одну или несколько связанных таблиц к вашему текущему проекту Access. Как правило, суть заключается в добавлении связанных таблиц из другой БД (или даже другого сервера БД). Таким образом, вы можете видеть все, используя один файл Access.

■ **Back Up SQL Database** (Резервная копия базы данных). Создает резервную копию вашей БД SQL Server в файле с расширением dat, который можно поместить куда угодно (на серверный или на свой компьютер). Специалисты SQL Server обычно предпочитают применять средство, позволяющее создавать резервные копии автоматически, например SQL Server Agent (это средство входит только в полную версию SQL Server).

■ **Restore SQL Database** (Восстановление базы данных Microsoft SQL Server из резервной копии). Берет файл с расширением dat, который был создан командой **Office → Server → Back Up SQL Database** (Office → Сервер → Резервная копия базы данных), и создает заново соответствующую БД SQL Server.

■ **Transfer Database...** (Перенос базы данных...). Позволяет переместить БД с одного компьютера на другой. (На обоих компьютерах должна быть программа SQL Server.) Программа Access сохраняет новое расположение сервера в вашем файле проекта, поэтому можно продолжать использование БД, находящейся на ее новом месте. Это средство

■ удобно, если вы тестировали БД SQL Server на своем компьютере и теперь хотите перенести ее на сетевой сервер, где любые пользователи смогут обращаться к ней.

■ **Copy Database File...** (Копирование базы данных...). Если вы запускаете программу SQL Server на своем компьютере, этот вариант позволяет создать копию файла БД, которую можно взять на другой компьютер. (Если вы подключены к копии программы SQL Server на другом сервере, эта команда работать не будет.) Обычно легче использовать команду **Office → Server → Transfer Database...** (Office → Сервер → Перенос базы данных...), поскольку она делает все за один шаг.

■ **Drop SQL Database** (Удаление базы данных SQL Server). Стирает БД в программе SQL Server и удаляет ее.

■ **Set Logon Password** (Задание пароля входа в систему). Если применяется смешанный режим аутентификации, эта команда позволяет изменить пароль SQL Server.

20.3.3. *Создание БД SQL Server вручную*

Программа Access разрешает создать пустую новую БД SQL Server и затем добавить в нее таблицы, которые нужны. Вот как это делается.

1. Выберите **Office → New** (Office → Создать) (или просто запустите программу Access, не открывая файл БД, и щелкните кнопкой мыши пиктограмму **Blank Database** (Новая база данных)).

На экране появится окно **Getting Started** (Приступая к работе).

2. В правой части окна щелкните кнопкой мыши пиктограмму папки, расположенную рядом с именем файла БД.

На экране вы увидите диалоговое окно **New Database** (Файл новой базы данных).

3. В списке **Save as type** (Тип файла) выберите строку **Microsoft Office Access Projects (*.adp)** (Проекты Microsoft Office Access (*.adp)).

4. Выберите папку, в которую хотите поместить клиентскую БД (adp-файл), введите имя в поле **File name** (Имя файла) (например, ZooAnimals.adp) и затем щелкните мышью кнопку OK.

Программа Access вернет вас на страницу **Getting Started** (Приступая к работе) с вашей информацией на панели **New Project** (Новый проект) в правой части окна.

5. Щелкните мышью кнопку **Create** (Создать), чтобы узаконить выполненные действия.

Программа Access поинтересуется вашим желанием использовать существующую БД SQL Server.

6. Щелкните мышью кнопку No (Нет) для создания собственной новой БД.

Появится окно с вопросом о местонахождении вашего сервера и имени создаваемой БД. Этот этап вы уже видели раньше в мастере преобразования БД (см. рис. 20.6).

7. Введите расположение сервера вашей БД и имя БД, которую хотите создать, и затем щелкните мышью кнопку **Finish** (Готово).

Программа Access создаст пустую БД SQL Server (и adp-файл вашего проекта). После небольшой задержки на экране появится окно Access с пустой областью переходов.

8 Когда будете готовы к созданию вашей первой таблицы, переходите к следующему разделу.

20.4. *Добавление объектов в БД SQL Server*

Самая замечательная черта поддержки программы SQL Server в Access заключается в возможности работы в знакомом окне программы Access, даже когда вы работаете с совершенно другим процессором БД. Но за это удобство приходится платить. Как вы увидите в следующих разделах, создание объектов БД для SQL Server не так наглядно, как их создание в программе Access.

20.4.1. *Создание таблицы*

Создать таблицу можно в любом проекте Access (файл с расширением adp), будь то новая БД, созданная вами с нуля, или существующая БД, с которой вы работаете, в любом случае процесс одинаков.

Сначала выберите на ленте **Create → Tables → Table Design** (Создание → Таблицы → Конструктор таблиц). В режиме таблицы создать таблицу нельзя. Всегда следует начинать в режиме **Design View** (Конструктор). Можно также обычным способом редактировать имеющуюся таблицу в конструкторе. Просто щелкните ее правой кнопкой мыши и выберите режим **Design View** (Конструктор).

Когда на экране появится окно конструктора, вы заметите, что оно выглядит несколько иначе, чем, конструктор для обычных таблиц Access. К счастью, работает режим в основном так же. Вы добавляете список полей сверху вниз и настраиваете имя, тип данных и размер

каждого поля. (В программе SQL Server они называются столбцами, а не полями, но на самом деле это не имеет значения.)

У каждого поля есть пять столбцов информации (рис. 20.10), которые вы должны заполнить.

■ Column Name (Имя столбца). Этот столбец идентифицирует поле (так же как в обычной таблице Access). Во избежание неприятностей не используйте пробелы и специальные символы.

■ Data Type (Тип данных). Этот столбец определяет тип данных, которые может хранить поле (так же как это делается в обычной таблице Access). Но набор типов данных, предоставляемый программой SQL Server, отличается от набора типов, используемых программой Access.

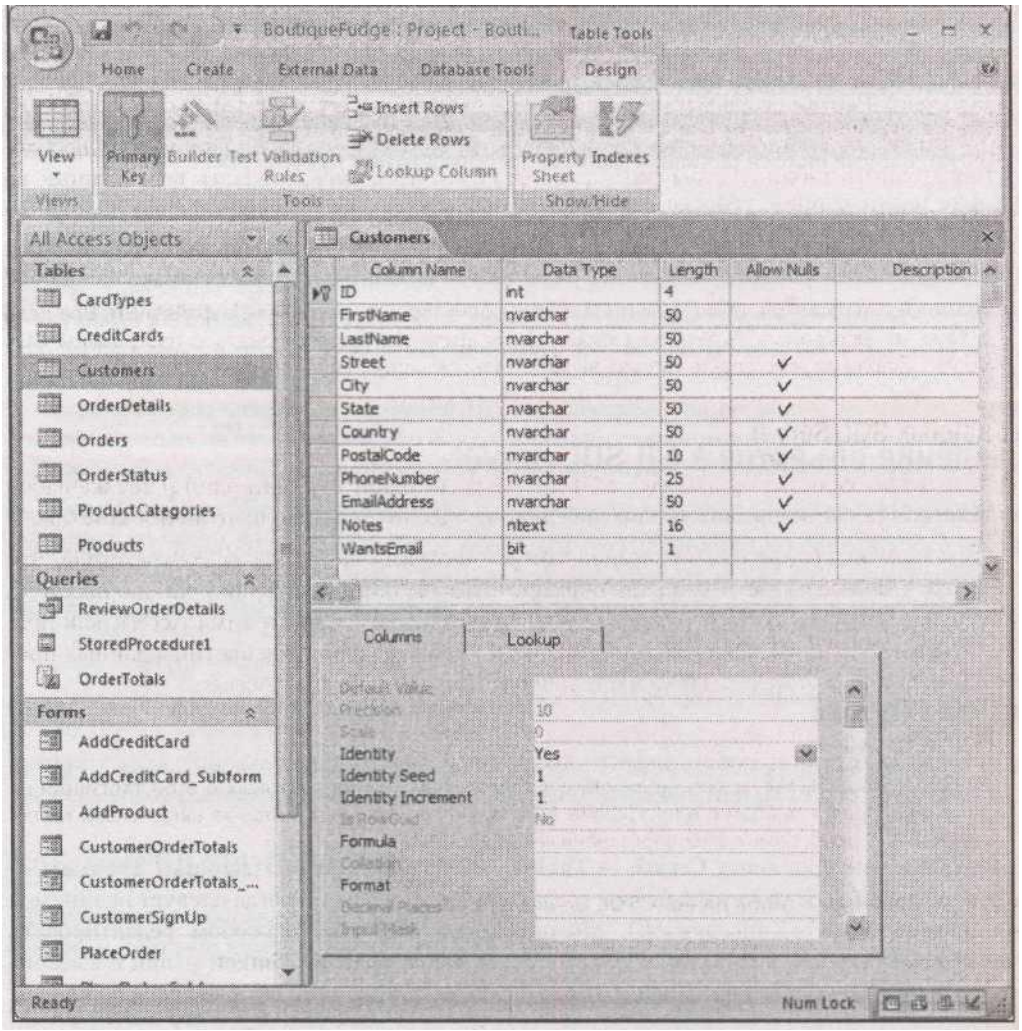


Рис. 20.10. В этом примере показан режим **Design View** в стиле SQL Server для уже знакомой таблицы **Customers** из БД BoutiqueFudge

■ **Length** (Длина). Этот столбец в грубом приближении соответствует свойству **Field Size** (Размер поля) в Access. Для основанных на тексте типов данных длина равна количеству допустимых символов. Для большинства типов данных размер поля служит лишь для сведения; он отображает количество байтов, занятых значением поля, и не может быть изменен.

■ **Allow Nulls** (Разрешить пустые значения). Этот столбец соответствует свойству **Required** (Обязательное поле) в программе Access. Если в столбце установлен флажок, вы сообщаете программе SQL Server о том, что пустые значения разрешены, т. е. пользователь БД может пропустить это поле.

■ **Description** (Описание). В этом столбце приводится описание вашего поля на обычном английском. Заполнять его или нет - ваше дело.

При проектировании таблицы также необходимо задать первичный ключ. Обычно для этого используется поле, хранящее автоматически сгенерированный код. Для обозначения поля как первичного ключа щелкните его кнопкой мыши для выделения и затем выберите на ленте

Table Tools | Design → Tools → Primary Key (Работа с таблицами | Конструктор → Сервис → Ключевое поле). Вы увидите пиктограмму ключа, отображаемую у левого края строки.

Когда работа с таблицей завершена, закройте ее. Программа Access напомнит о необходимости сохранить таблицу и выбрать для нее имя. Затем можно начать ввод данных на привычном листе данных программы Access, который нисколько не изменился.

Примечание

Возможно, вы заметили, что проекты Access медлительнее, чем обычные файлы БД. Все дело в том, что программе Access за кадром приходится взаимодействовать с программой SQL Server, запрашивая создание таблиц, выполнение операций над данными и т. д.

Типы данных SQL Server

Было бы чудесно, если бы программы SQL Server и Access применяли один и тот же набор типов данных. Но у этих приложений разное происхождение и порой их отличия заметны.

К счастью, между большинством типов данных программ есть близкое соответствие. Это означает, что у большей части типов данных Access есть соответствующий тип данных SQL Server, очень близкий к типу Access. (Когда БД преобразуется, программа Access, как пра- вило, способна подобрать хорошее соответствие.) В табл. 20.1 приведены типы данных про- граммы SQL Server, которые вы получаете для разных типов данных Access.

Примечание

У программы SQL Server есть еще много типов данных, не приведенных в этой таблице и не имеющих близких аналогов в программе Access. Но типы данных, которые включены в таблицу, - несомненно, самые распространенные.

Таблица 20.1. Сравнение типов данных SQL Server и Access

Тип данных Access	*Эквивалент SQL Server*
Текстовый (Text)	nvarchar (способен хранить до 4000 символов, в отличие от Текстового типа данных Access, у которого верхний предел составляет 255 символов)
Поле Memo (Memo)	ntext
Числовой (Number) (со значением в свойстве Размер поля (Field Size) - Целое (Integer))	smallint
Числовой (Number) (со значением в свойстве Размер поля (Field Size) - Длинное целое (Long Integer))	int
Числовой (Number) (со значением в свойстве Размер поля (Field Size) - Одинарное с плавающей точкой (Single))	real
Числовой (Number) (со значением в свойстве Размер поля (Field Size) -Двойное с плавающей точкой (Double))	float
Числовой (Number) (со значением в свойстве Размер поля (Field Size) -Действительное (Decimal))	decimal
Числовой (Number) (со значением в свойстве Размер поля (Field Size) - Байт (Byte))	tinyint
Числовой (Number) (со значением в свойстве Размер поля (Field Size) - Код репликации (ReplicationID))	uniqueidentifier
Дата/время (Date/Time)	datetime
Денежный (Currency)	money
Счетчик (AutoNumber)	int (с параметром Identity равным Yes (Да))
Логический (Yes/No)	bit
Гиперссылка (Hyperlink)	nvarchar
Вложение (Attachment)	nvarchar (сохраняется только имя файла)

Поля типа Счетчик

Возможно, вы заметили, что у программы SQL Server нет типа данных **Счетчик.** Но не дайте себя одурачить, решив, что нет способа получить это невероятно полезное средство в программе SQL Server. Просто задавать его придется немного иначе.

1. Когда создаете поле ID (Код), задайте для него тип данных int.
2. На вкладке **Columns** (Столбцы), расположенной под списком полей, задайте свойство **Identity** равным *Yes* (Да).

Значение параметра **Identity** - это имя поля типа **Счетчик** в программе SQL Server. Это имя программа присваивает автоматически и гарантирует его уникальность.

3. Вы также можете задать свойства **Identity Seed** (Начальное значение IDENTITY) и **Identity Increment** (Приращение IDENTITY).

- Значение **Identity Seed** (Начальное значение IDENTITY) - начальное значение. Это свойство - досадное упущение программы Access, которая всегда начинает считать с 1.
- Значение **Identity Increment** (Приращение IDENTITY) - величина, на которую программа SQL Server увеличивает очередное значение. Например, если **Identity Increment** - 5, вы увидите числа 1, 6, 11, 16 и т. д. Конечно, программа SQL Server может по разным причинам, как и программа Access, пропустить очередное значение.

Подстановки

Конструктор SQL Server лишен удобного мастера подстановки, который применяет программа Access. Вместо него вы должны выбрать поле, куда хотите поместить подстановку, щелкнуть кнопкой мыши вкладку **Lookup** (Подстановка) в нижней части окна конструктора (рис. 20.11) и затем заполнить все данные подстановки. Далее приведены ключевые параметры, необходимые для создания подстановки.

■ В поле параметра **Display Control** (Тип элемента управления) следует задать *Combo Box* (Поле со списком), раскрывающийся список, позволяющий выбрать нужное значение.

■ В поле параметра **Row Source Type** (Тип источника строк) нужно задать *Tables/Views/Functions* (Таблицы/представления/функции), если хотите создать подстановку, применяющую данные из связанной таблицы. (Если хотите предоставить просто список значений, можно использовать значение *Value List* (Список значений).)

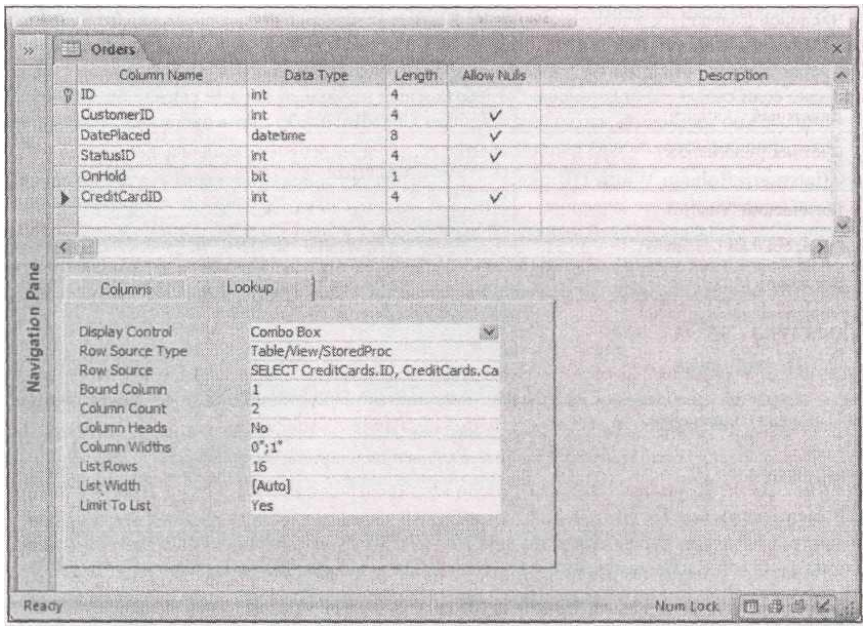


Рис. 20.11. Подготовленная полностью подстановка для поля **CreditCardID** в таблице **Orders**

■ В поле параметра **Row Source** (Источник строк) указываются данные для подстановки. Если данные извлекаются из другой таблицы, применяется SQL-команда SELECT, которая извлекает два поля - поле с описательной информацией и поле со значением ID (Код). Например, можно использовать команду SELECT ID, ProductName FROM Products ORDER BY ProductName для создания отсортированного по названию товара списка

подстановки, который получает ID и название каждого товара из таблицы **Products.**

Подсказка

Если вы не хотите писать оператор SELECT самостоятельно, щелкните кнопкой мыши в этом поле и затем щелкните мышью кнопку с многоточием, открывающую окно запроса, в котором можно выбрать таблицу и поля. Это окно запроса немного отличается от **Конструктора** запросов программы Access, который вы использовали до этого.

■ Параметр **Bound Column** (Связанный столбец) обозначает, какой столбец (из параметра **Row Source** (Источник строк)) должен быть добавлен в поле, когда выбрано значение из списка значений подстановки. Например, если первое поле в вашей команде SELECT - ID (как в предыдущем примере), нужно задать значение *1.*

■ В параметре **Column Count** (Число столбцов) задается количество столбцов, отображаемых в списке подстановки. Обычно задается значение *2* (для отображения обоих столбцов), но ширина первого столбца с номером ID задается бесконечно малой, поэтому вы его практически не видите.

■ Параметр **Column Heads** (Заголовки столбцов) определяет, выводить ли заголовки в первой строке столбцов в списке подстановки. Обычно задается значение *No* (Нет). Однако если создается подстановка, отображающая несколько порций связанной информации, можно применить заголовки столбцов для того, чтобы легче было понять, что есть что, при просмотре списка подстановки.

■ Параметр **Column Width** (Ширина столбцов) задает ширину каждого столбца в списке подстановки. Каждое значение в дюймах отделяется точкой с запятой. Например, значение *0 ";1"* скрывает из вида первый столбец и делает второй столбец шириной 1 дюйм.

К сожалению, создание подстановки не формирует связь между двумя таблицами. Если вы хотите создать отношение, необходимо сделать это самостоятельно, как описано в следующем разделе.

Примечание

Когда БД преобразуется, программе Access хватает изобретательности, чтобы сохранить все ваши подстановки.

Отношения

Как вы узнали в *главе 5,* у каждой заслуживающей уважения БД есть множество отношений, В программе Access существуют два способа быстрого построения отношения: с помощью схемы данных и созданием подстановки в поле. Но в проекте Access ни одно из этих средств не доступно. Вместо этого вам придется определять отношения вручную в окне Конструктора для вашей таблицы.

Вот как это делается.

1. Откройте подчиненную или дочернюю таблицу в **Конструкторе.**

У этой таблицы есть поле, связанное с родительской таблицей. (Например, **Products -** дочерняя таблица для **ProductCategories.** Поле **ProductCategorylD -** связующее звено, которое присутствует в таблице **Products.)**

2. Выберите на ленте **Table Tools | Design → Show/Hide → Property Sheet** (Работа с таблицами | Конструктор → Показать или скрыть → Страница свойств).

Это действие выводит на экран диалоговое окно **Properties** (Свойства) (рис. 20.12), которое выглядит совсем не так, как **Property Sheet** (Окно свойств), которое вы применяли раньше в БД Access.

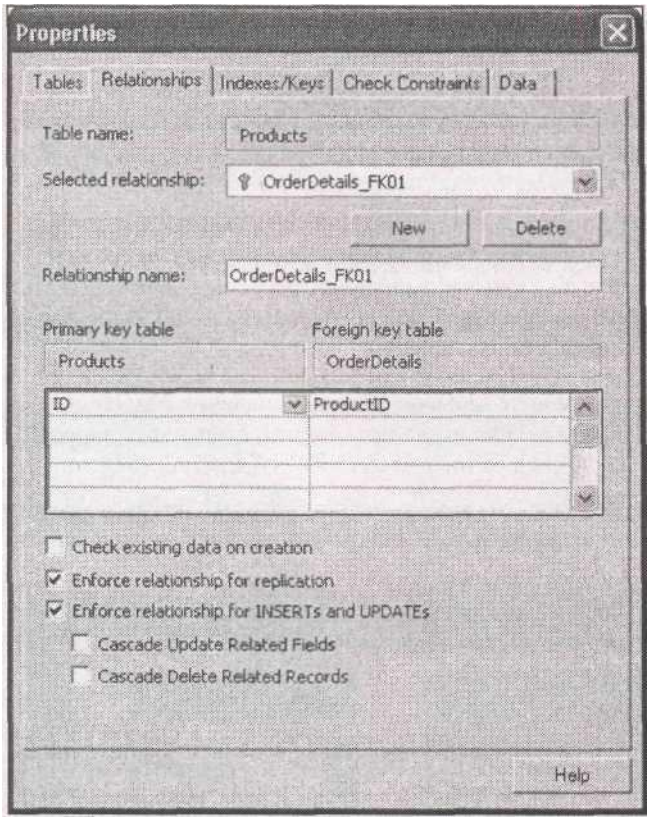


Рис. 20.12. Показано установленное отношение, связывающее таблицу **OrderDetails** (как дочернюю) с таблицей **Products** (как родительской). В каждой записи таблицы **OrderDetails** поле **ProductID** указывает на заказанный товар

3. Щелкните кнопкой мыши вкладку **Relationships** (Схема данных).
4. Щелкните кнопкой мыши кнопку **New** (Создать) для определения нового отношения.
5. В списке, расположенном под заголовком **Primary key table** (Таблица первичного ключа), выберите родительскую таблицу.
6. В первой строке, расположенной под именем таблицы, выберите однозначно определенное поле в родительской таблице.

(В этой области вкладки есть несколько строк, поскольку можно создавать отношения, основанные на нескольких полях, хотя это делается редко.)

7. В списке, находящемся под заголовком **Foreign key table** (Таблица внешнего ключа), выберите дочернюю таблицу.

В первой строке, расположенной ниже, выберите поле в дочерней таблице, которое указывает на связанную родительскую запись.

8. Если хотите убедиться в том, что существующие данные удовлетворяют данному отношению, установите флажок **Check existing data on creation** (Проверять существующие данные при создании).

Если не хотите проверять имеющиеся записи на соответствие правилам данного отношения, не устанавливайте этот флажок. Если в вашей таблице еще нет данных, ваш выбор не играет роли.

9. Установите флажок **Enforce relationships for INSERTS and UPDATES** (Обеспечить отношение для INSERT и UPDATE), если хотите обеспечить ссылочную целостность при добавлении и изменении записей.

Это действие не позволит нарушать правила отношения при добавлении и редактировании записей. Например, будет запрещена вставка дочерней записи, указывающей на несуществующую родительскую запись. Если вы решите не применять ссылочную целостность, можно использовать один из вариантов, расположенных ниже, для переключения на каскадные обновления или удаления.

10. Когда закончите, закройте окно.

20.4.2. *О запросах*

Таблицы данных - не единственные объекты БД, хранящиеся в БД SQL Server. Ваша БД SQL Server может также содержать запросы, бесконечно полезные процедуры для поиска (или изменения) нужных записей.

У объектов, которые приверженцы программы Access называют запросами, совсем иное существование в SQL Server. Там, где вы видите запросы, программа SQL Server видит три объекта разных типов.

- View (Представление). *Представление* - приблизительный аналог запроса на выборку (select query) - он выбирает записи (возможно, из связанных таблиц) и отображает их на листе данных.
- User-defined function (Пользовательская функция). *Пользовательская функция* аналогична запросу на выборку с параметрами. (Как вы уже знаете, параметры позволяют запрашивать порцию информации сразу перед выполнением запроса. Эту информацию затем можно применить для отбора записей и выполнения вычисления.)
- Stored procedure (Хранимая процедура). *Хранимая процедура* - тяжеловес среди объектов БД SQL Server. Она может выполнять целый ряд задач, таких как выбор записей, фиксация изменений, выполнение транзакций. Вы не будете использовать большую часть этих возможностей, когда создадите хранимую процедуру в программе Access. Вместо этого вы будете применять хранимые процедуры для создания аналога запроса на изменение в SQL Server, который фиксирует в БД одиночную операцию обновления, добавления или удаления.

В следующем разделе вы попытаетесь приложить руки к созданию базового представления.

20.4.3. *Создание представления*

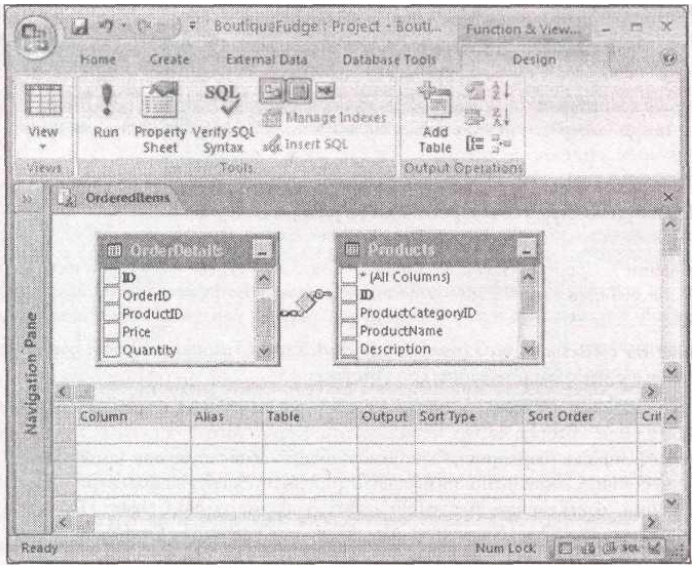
Создание представления во многом напоминает создание классического запроса Access, выбирающего группу записей. Далее перечислены действия, демонстрирующие создание представления и наделение его всеми важными чертами, включая сортировку, фильтрацию и выражения.

1. Выберите на ленте **Create → Other → Query Wizard** (Создание → Другие → Мастер запросов).

На экране появится диалоговое окно New Query (Новый запрос), предлагающее помощь в создании запросов SQL Server разных типов. В данном случае нужно просто создать обычный запрос, отбирающий некоторые полезные данные.

2. Выберите режим Design View (Конструктор) и щелкните мышью кнопку OK. Отображается диалоговое окно **Add** Table (Добавить таблицу).
3. Выберите таблицу (или таблицы, которые хотите использовать) и затем щелкните мышью кнопку **Add** (Добавить), чтобы включить ее в ваш запрос. Когда закончите, нажмите кнопку **Close** (Закрыть).

На экране появится Конструктор запросов (query designer). Теоретически он работает более или менее так же, как Конструктор запросов программы Access с теми же параметрами.



Однако выглядит он немного иначе (рис. 20.13).

Рис. 20.13. Когда в запрос добавляется несколько таблиц, и у них есть определенные в БД отношения, **Конструктор** запросов включает объединенные строки. В данном примере запрос отображает список заказанных товаров с дополнительной информацией о товарах из таблицы **Products**

4. Выберите поля, которые хотите включить в результаты запроса.

Для включения поля в ваши результаты, установите флажок, находящийся рядом с полем. Поля вставляются в перечень в нижней части окна, каждое в отдельную строку (рис. 20.14). Это похоже на окно **Конструктора** запросов в Access, но повернутое вокруг своей оси. (Программа Access создает отдельный столбец для каждого поля в запросе.)

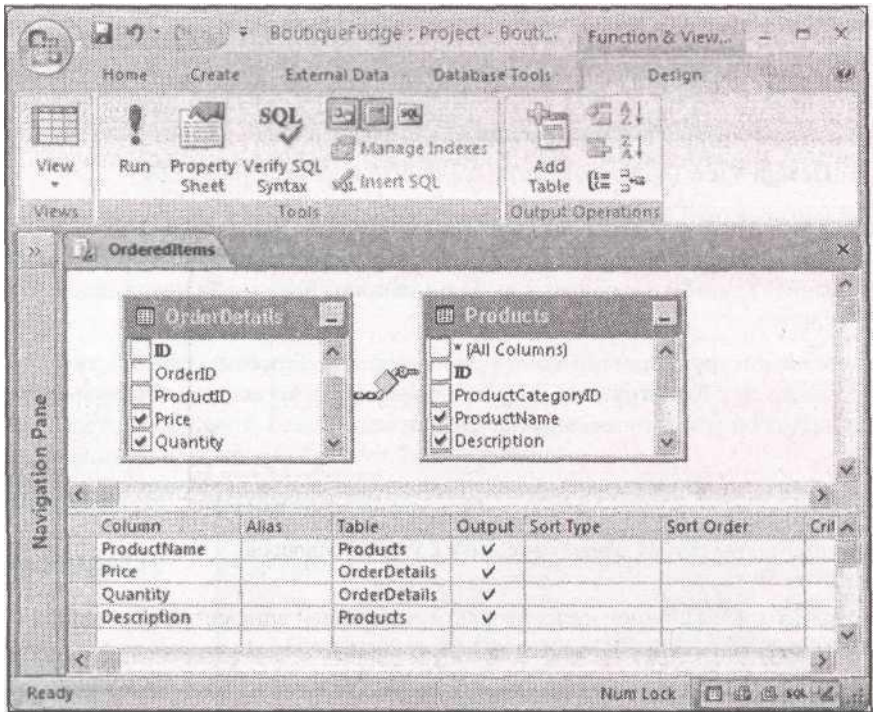


Рис. 20.14. Данный пример демонстрирует четыре поля, все с установленным флажком в столбце **Output,** что означает их отображение в таблице результатов

К этому моменту у вас есть полнофункциональный запрос. Но, возможно, вы захотите добавить сортировку, фильтрацию и вычисляемые выражения, как описывается далее.

5. Если хотите отсортировать поле, выберите в поле **Sort Type** (Тип сортировки) вариант *Ascending* (по возрастанию) или *Descending* (по убыванию).

Если хотите сортировать по нескольким полям, задайте параметр **Sort Type** (Тип сортировки) для каждого поля, которое хотите использовать. Кроме того, надо указать число в столбце **Sort Order** (Порядок сортировки), чтобы сообщить **Конструктору** запросов, какую сортировку выполнять первой. Например, если сортируется группа фамилий и имен сначала по фамилии, а затем по имени, следует задать в параметре **Sort Order** (Порядок сортировки) число 1 для поля **LastName** (фамилия) и число 2 для поля **FirstName** (имя).

Подсказка

Если хотите использовать поле для сортировки или фильтрации, но не хотите выводить его в таблице результатов, сбросьте флажок в столбце **Output** (Вывод).

6. Если хотите применить фильтрацию, задайте условие отбора в столбце **Criteria** (Условия) в строке соответствующего поля.

Для числовых значений выполнить эту задачу довольно просто, поскольку выражения для фильтра точно такие же, как в программе Access. Можно использовать те же операции (например, знаки +, -, /, * для вычислений и знаки =, < и > для сравнения чисел). Но если хотите работать с текстом или датами, ознакомьтесь с синтаксическими отличиями в программах SQL Server и Access в *примечании "Для тех, кто понимает. Синтаксические*

различия" далее в этом разделе.

Условия отбора можно применять к любому количеству полей. Если хотите задать несколько разных условий и отобразить записи, удовлетворяющие любому из них, дополнительные условия определите в столбце Or (Или), как показано на рис. 20.15.

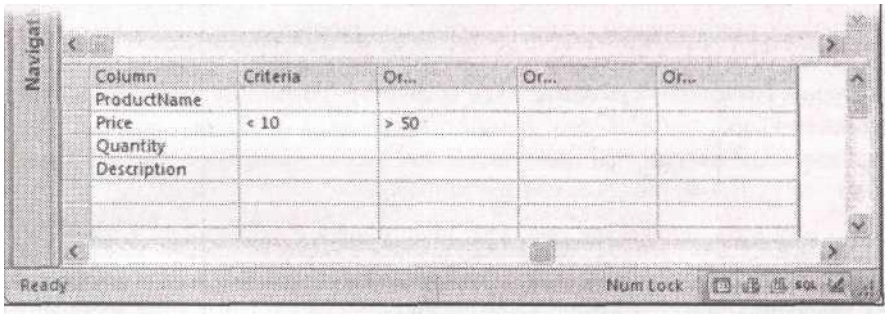


Рис. 20.15. Это поле соответствует любым ценам менее 10 и более 50 долларов

7. Если хотите применить вычисляемое поле, добавьте его в конец списка, заполнив поля в столбцах **Column** (Столбец) и **Alias** (Псевдоним) (рис. 20.16).

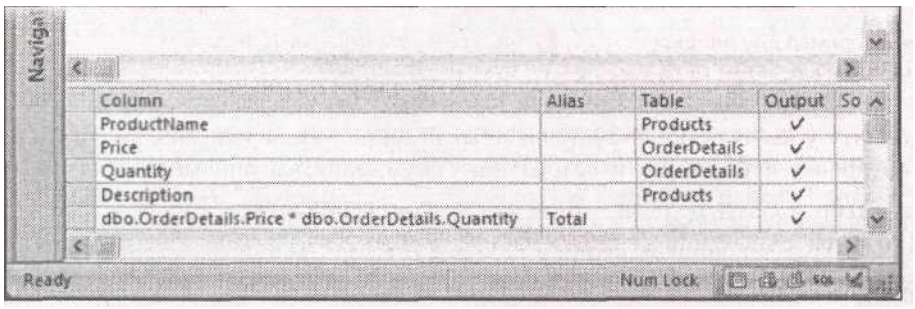


Рис. 20.16. Вы заметили, что в именах полей применяются странные имена, состоящие из трех частей (имя **Price** становится **dbo.OrderDetails.Price).** Это имя означает, что поле **Price** находится в таблице, названной **OrderDetails,** которая была создана владельцем БД **(dbo)** (Программа SQL Server применяет в выражениях такие имена, чтобы устранить возможную неоднозначность. Но не беспокойтесь - **Конструктор** запросов преобразует обычные имена в трехчастные автоматически.)

Синтаксис вычисляемого поля в **Конструкторе** запросов SQL Server несколько отличается. Вместо синтаксической записи **ColumnName: Expression** (ИмяСтолбца: Выражение) имя поля помещается в поле **Alias** (Псевдоним), а выражение в поле **Column** (Столбец).

Несмотря на то, что можно ввести выражение, используя просто имена полей, **Конструктор** запросов автоматически преобразует их в трехчастные имена. На рис. 20.16 показано, что вы увидите, если введете простое вычисляемое выражение Price*Quantity, вычисляющее стоимость одной строки заказа.

Если хотите создать вычисляемое выражение, использующее текст или даты, прежде познакомьтесь с синтаксическими отличиями в программах SQL Server и Access.

Для тех, кто понимает.

Синтаксические различия

Если запрос создается с применением проекта Access, он становится на самом деле объектом БД SQL Server. Именно программа SQL Server хранит запрос и выполняет его. Это важно, поскольку разновидность языка SQL (структурированный язык запросов), которая применяется в программе Access, немного отличается от букета, который вы найдете в программе SQL Server. Эти незначительные различия могут заставить споткнуться большую часть искушенных создателей запросов.

К счастью, можно избежать множества проблем, просто зная о нескольких ключевых различиях.

• В текстовых значениях используют одинарные кавычки (апострофы), а не двойные. Таким образом, для поиска записи какого-либо товара по имени применяйте строку

'Maple Magic', а не"Maple Magic".

- В датах также применяют одинарные кавычки, не знаки решетки. Поэтому ищите заказы, сделанные до '1/30/2008', а не до #1/30/2008#.
- При слиянии двух фрагментов текста используйте символ +, а не символ &. Итак, для получения полного имени в вычисляемом выражении применяйте строку FirstName + ' ' + LastName,а не строку FirstName &+ " " & LastName.

Вы не можете применять функции Access. У программы SQL Server есть собственная библиотека функций и, несмотря на то, что многие из них такие же или похожи на знакомые и любимые вами функции Access, различия присутствуют в изобилии. Самый безопасный подход - найти функцию, которую хотите использовать, в справочном руководстве SQL Server Books Online (см. на рис. 20.2, как его загрузить из Интернета). В этом случае вы сможете проверить функцию и увидеть, действует ли она так же, как ее дубликат в программе Access.

8. Когда создание запроса закончено, щелкните правой кнопкой мыши заголовок вкладки и затем выберите строку **Datasheet View** (Режим таблицы) (или выберите на ленте **Function & View Tools | Design → Tools → Run** (Работа с функциями и представлениями Конструктор → Сервис → Выполнить)).

Программа Access предложит сохранить ваш запрос. Когда вы сделаете это, то увидите ваши результаты на привычном листе данных, так же как в случае выполнения запроса в БД Access. Затем можно напечатать или отредактировать полученные результаты.

21. Глава 21. Подключение Access к SharePoint

Даже в компаниях с самым неблагоприятным психологическим климатом, людям приходится уживаться друг с другом. Предприятия, у которых есть эффективные способы коллективного использования информации, будь то графики встреч, высокоприоритетные задачи или внутриофисные переговоры, больше преуспевают, чем те, которые не заботятся об этом.

Возможно, читая *главу 18,* вы догадались, что можете применить программу Access для коллективного использования такого рода данных. Для этого нужно всего лишь создать подходящую БД, поместить ее в расположение с общим доступом и убедиться в том, что у всех участников на компьютерах установлена программа Access. Но вам не придется делать ничего из только что перечисленного, если использовать SharePoint - программный продукт корпорации Microsoft, который специально разработан для взаимодействия групп сотрудников в офисе. Лучше всего, если ваша компания приобретет ОС Windows Server 2003, в которую включена базовая версия SharePoint, содержащая все, что вам нужно. (Под именем Microsoft Office SharePoint Server 2007 или сокращенно MOSS продается и версия SharePoint, усиленная дополнительными средствами Office.)

Примечание

Если вы не являетесь счастливым обладателем системы Windows Server 2003 и предпочли бы потратить 1000 долларов (а именно во столько она вам обойдется) на отпуск на морском берегу, можете дальше не читать. Вам лучше разработать для сотрудничества собственные БД *(см. главу 18)* или воспользоваться бесплатной версией программы SQL Server *(см. главу 20).*

Программа SharePoint отлично работает и без Access - вам потребуется только Web-обозреватель Internet Explorer. С его помощью вы сможете зарегистрироваться на узле SharePoint вашей рабочей группы, просмотреть свежую информацию, загрузить документы и отредактировать списки данных. Для большинства пользователей SharePoint этого более чем достаточно. Но если у вас под рукой есть копия программы Access, появляются две дополнительные возможности. Вы можете:

- передавать данные в программу SharePoint и из нее. Это средство очень полезно, если одни сотрудники вашей компании применяют программу Access, а другие - SharePoint. Конечно, заботиться о том, чтобы все получили свежие данные - целиком ваша задача;
- использовать программу Access как клиента для SharePoint. Этот метод аналогичен методу, применявшемуся в *главе 20* для взаимодействия с программой SQL Server. Он позволяет работать с таблицами данных в знакомой рабочей среде Access, а хранить данные на сервере SharePoint. Преимущество в том, что ваши данные одновременно доступны гораздо большему числу людей, и даже тем, у кого нет программы Access, информация доступна на Web-страницах вашего узла SharePoint.

В данной главе вы узнаете немного больше о программе SharePoint и проверите на практике оба описанных метода.

Примечание

В отличие от программы Access сервер SharePoint может одновременно поддерживать практически неограниченное число пользователей. Объясняется это скрытым применением программы SQL Server, мощного программного обеспечения для БД, которое рассматривалось в *главе 20.*

21.1. *Основные сведения о SharePoint*

Нет, это не специальный сленг, обозначающий новый способ. SharePoint - это серверная программа, помогающая группам пользователей сотрудничать, коллективно используя данные и документы с помощью централизованного Web-сайта или узла.

SharePoint - несколько необычное программное обеспечение. Несмотря на то, что это один из самых быстроразвивающихся программных продуктов в истории корпорации Microsoft, большинство обычных пользователей никогда не слышали о нем, и даже его многолетние

поклонники с трудом описывают, что именно он делает. К счастью, основная идея программы SharePoint довольно проста. Сначала группа пользователей собирается вместе и создает Web-узел SharePoint. Этот узел располагается на сервере в сети вашей компании. В процессе установки решается, кому предоставить доступ к узлу и какие действия разрешить.

Подсказка

Обычно ваш сервер SharePoint не доступен в Интернете, но если вы хотите предоставить пользователям возможность работать с ним из дома, ситуацию можно изменить - просто сообщите об этом вашему интернет-провайдеру, предоставляющему услуги по размещению Web-сайтов (Internet hosting company).

После того как узел SharePoint установлен, все члены рабочей группы могут получить к нему доступ. Процесс регистрации прост - запустите Web-обозреватель и перейдите на узел группы. Обычно нет необходимости вводить имя пользователя и пароль, поскольку Internet Explorer автоматически регистрирует вас, используя текущую учетную запись пользователя (которую вы применяли для регистрации в сети, когда запускали свой компьютер в начале рабочего дня). Однако если для доступа к серверу SharePoint требуется другое имя пользователя и пароль, при переходе к узлу Internet Explorer выведет на экран окно регистрации. (Сетевые администраторы могут оказать неоценимую помощь при возникновении подобных проблем.)

На узле SharePoint вы увидите настраиваемую страницу со сводкой последних новостей, анонсов и полезных ссылок (рис. 21.1).

Примечание

Корпорация Microsoft применяет тысячи узлов SharePoint для координации работы собственных групп, включая группу создателей программы Access.

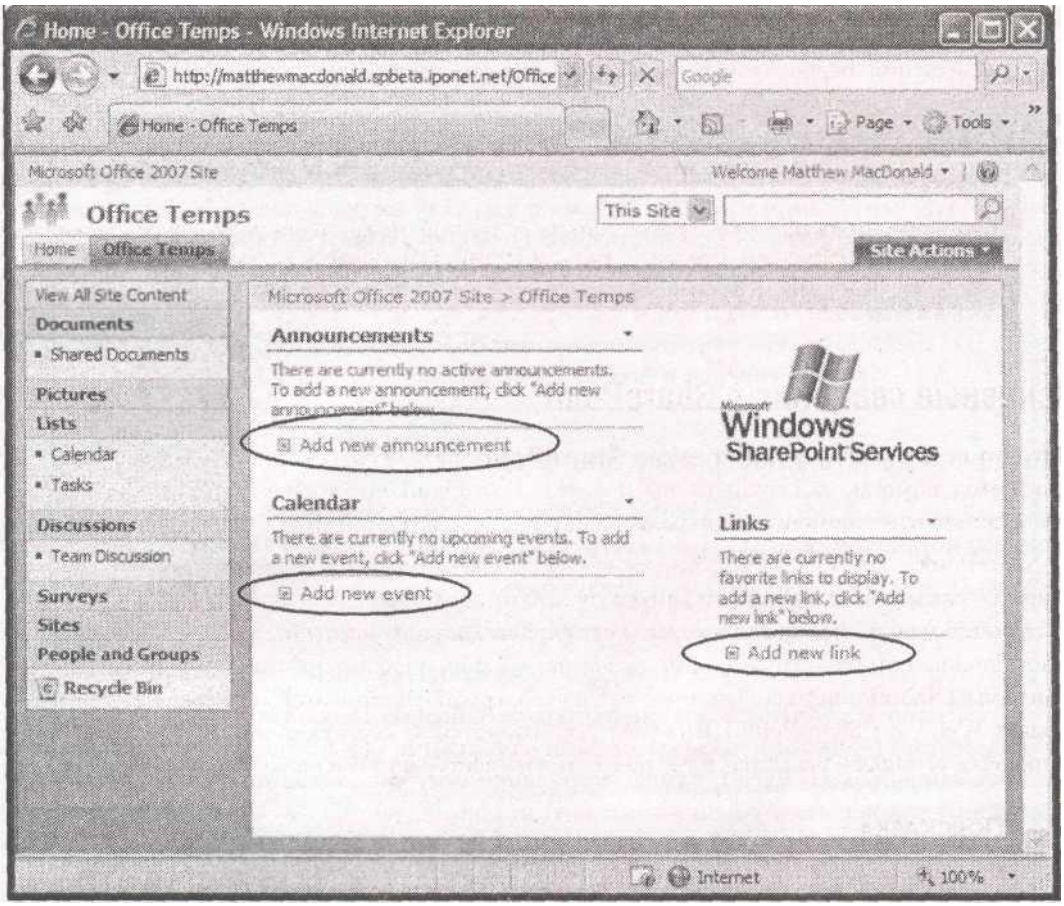


Рис. 21.1. В данном примере отображается стандартный узел SharePoint без какой-либо настройки. Вы можете переходить к разным областям с помощью панели, расположенной в левой части страницы, или использовать одну из ссылок "Add..." (обведены) для создания нового извещения, новой встречи или ссылки. Другие члены группы могут

зарегистрироваться и увидеть добавленные вами элементы

Часто задаваемый вопрос.
Путаница, связанная с SharePoint

Как быть с другими версиями программы SharePoint?

Версия SharePoint, включенная в состав ОС Windows Server 2003, известна под названием Windows SharePoint Services (или WSS для любителей аббревиатур). Возможно, вы слышали о двух других программных продуктах, которые кажутся подозрительно похожими на Windows SharePoint Services.

• Microsoft Office SharePoint Portal Server (называемый осведомленными людьми SPS). Это не бесплатный продукт с дополнительными возможностями по сравнению с SharePoint Services.

Его ключевые особенности - способы интеграции разных узлов групп, возможность размещения персональных узлов всех членов группы и поддержка другого программного продукта Microsoft - BizTalk, способного автоматизировать управление рабочим процессом в огромных компаниях.

• Microsoft Office SharePoint Server 2007 (или MOSS). Это обновление SharePoint Portal Server. Данная программа играет в жизни ту же роль, но с небольшими отличиями. В нее включено дополнительное функциональное средство, которое раньше продавалось отдельно как Microsoft Content Management Server (сервер Microsoft управления контентом).

Эта глава посвящена исключительно программе SharePoint Services, которая предоставляет все средства, применяемые в Access.

21.1.1. *Что можно делать в программе SharePoint*

Освоение основных функциональных возможностей узла SharePoint не займет много времени. Вы можете просмотреть весь узел группы, как любой другой Web-сайт.

Узел SharePoint позволяет делать следующее:

- держать в поле зрения важные даты с помощью календаря группы;
- помещать сообщения на доску группового обсуждения (team discussion board);
- коллективно использовать документы пакета Office из Document Library (библиотека документов) (например, написанные вами в программе Word отчеты и электронные таблицы из программы Excel). Разные сотрудники могут предоставить отредактированные версии, а руководители групп - отклонить не понравившиеся им;
- назначить задачи разным сотрудникам и узнать, когда они будут выполнены;
- увидеть список всех членов вашей группы и разослать сообщения по электронной почте;
- совместно пользоваться ссылками на полезные Web-страницы;
- создавать и редактировать списки, хранящие разнообразные данные (рис. 21.2). Например, можно применять список для хранения жалоб важных клиентов, на которые следует ответить, или список продуктов, которые сотрудники приносят на обед в складчину.

Именно в последней задаче в действие вступает программа Access. Несмотря на то, что создать в SharePoint список с данными и управлять им можно с помощью вашего Web-обозрвателя, вероятно, вам захочется применить список в Access. Например, у вас есть форма, запрос или программная процедура, которые должны учесть эти данные. Или вам гораздо удобнее редактировать данные в привычном интерфейсе программы Access.

Список SharePoint аналогичен таблице Access. Несмотря на то, что оба термина ссылаются на одно и то же, у списка SharePoint больше ограничений. Он не поддерживает текст большого объема или условия на значения. И хотя в нем разрешены подстановки, список не позволит использовать отношения для защиты данных.

Из этого следует, что узел SharePoint - не слишком удачное место для хранения важной деловой информации, такой как списки клиентов, каталоги товаров и счета. Но он вполне подходит для размещения неофициальных списков или информации для конкретного случая, которую нужно передать коллегам. Например, списки SharePoint хороши для хранения офисных номеров телефонов или списка возможных добровольных участников спортивной команды компании.

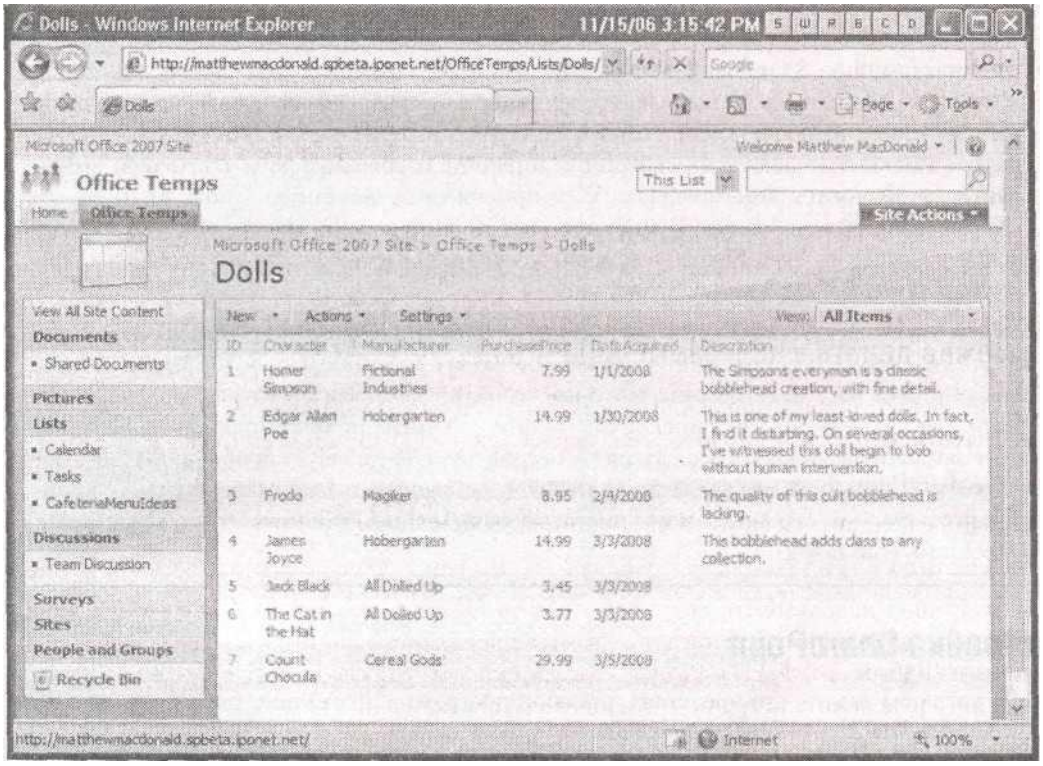


Рис. 21.2. В данном примере показан список SharePoint, дублирующий непримечательную таблицу **Dolls** из БД Bobblehead

На профессиональном уровне.

Установка SharePoint

Прежде чем вы сможете создать узел SharePoint, необходимо убедиться в том, что программное обеспечение SharePoint Services установлено должным образом. В данной книге не описывается полный процесс установки, поэтому, если хотите его опробовать, заручитесь поддержкой находящегося по соседству сетевого администратора.

Далее приведено несколько рекомендаций, которые помогут вам подготовиться.

• Прежде всего, для использования SharePoint вам нужен компьютер под управлением ОС Windows Server 2003. Если у вас последняя редакция (именуемая Windows Server 2003 R2, выпуск 2), у вас уже есть все нужные компоненты.

• Если у вас нет последнего выпуска Windows Server 2003, не бойтесь - у вас есть право на бесплатный дополнительный модуль, устанавливающий программу SharePoint Services. Укажите в строке адреса вашего Web-обозревателя **www.microsoft.com/windowsserver2003/technologies/sharepoint** для получения нужного вам загружаемого файла.

Этот сайт - замечательная отправная точка, если нужна дополнительная информация о программе SharePoint Services или вы столкнулись с необычной проблемой.

• Легче всего установить программу SharePoint, запустив мастер Configure Your Server (настройка вашего сервера) и выбрав роль (role) SharePoint Services. Ваш сервер будет настроен на установку нескольких ключевых компонентов, в том числе IIS (Internet Information Server, информационный сервер Интернета) (программное обеспечение, преобразующее компьютер в Web-сервер) и ASP.NET (Active Server pages, технология активных серверных страниц) (программное обеспечение, позволяющее запускать динамические Web-приложения, например, узлы SharePoint). Кроме того, если на вашем компьютере нет полной версии программы SQL Server, мастер Configure Your Server установит усеченную версию, которую можно использовать для хранения данных SharePoint.

Если вся эта возня с установкой кажется слишком трудным занятием (или вы не можете воспользоваться сетью компании), можно заплатить кому-нибудь, кто разместит узел SharePoint для вас. Есть фирмы, которые выделят немного места на мощном Web-сервере для хранения ваших списков SharePoint, документов SharePoint и т. д. Если это звучит заманчиво, можно подписаться на бесплатную 30-дневную пробную версию, позволяющую применять все средства SharePoint, описанные в этой главе; подробную информацию см. на

странице **www.microsoft.com/technet/windowsserver/sharepoint/ V2/techinfo/trial.mspx.**

21.2. *Настройка SharePoint*

Теперь, когда вы знаете, что представляет собой программа SharePoint, пора брать ее в оборот. Все серверы SharePoint запускаются с одним основным узлом или узлом верхнего уровня (home site). Несмотря на то, что для сотрудничества можно использовать этот узел, большинство пользователей предпочитает создавать дополнительные узлы для каждой рабочей группы. (У вашей компании может быть лишь один узел группы или несколько тысяч узлов.)

Подсказка

Вы увидите отображение узла верхнего уровня в программе Internet Explorer в конце процесса установки SharePoint.

21.2.1. *Создание узла рабочей группы*

Вот как создается новый узел группы.

1. Перейдите на ваш узел верхнего уровня SharePoint.

Помните о том, что программа SharePoint сообщит вам URL в конце процесса установки. Если вы разместили свой узел SharePoint на Web-сервере, компания, обеспечивающая размещение, должна предоставить вам URL-адрес.

2. Щелкните мышью кнопку **Site Actions** (Действия узла) (рис. 21.3) и затем выберите команду **Site Settings** (Параметры узла).

На экране появится страница управления узлом с огромным списком параметров, которые можно изменять.

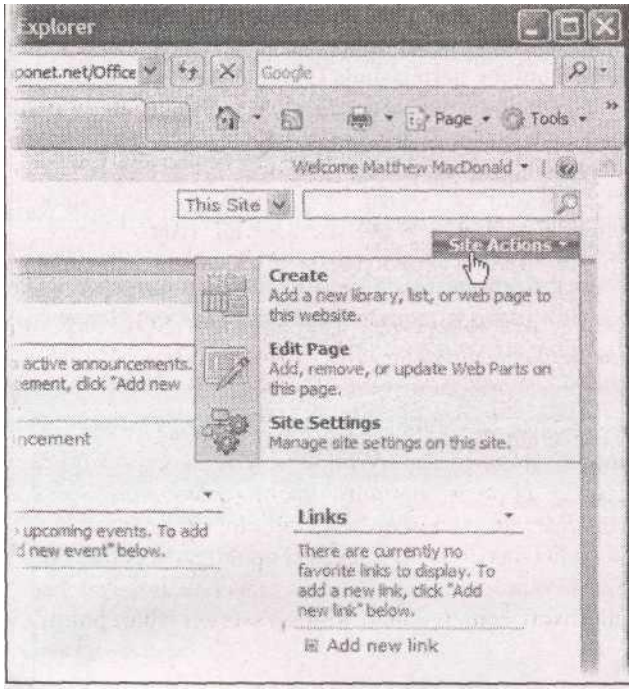


Рис. 21.3. Кнопка **Site Actions** располагается в правом верхнем углу любой страницы SharePoint. Она дает возможность быстро настроить узел или создать новый элемент (такой как список или Web-страница)

3. В разделе **Site Administration** (Администрирование узла) щелкните кнопкой мыши ссылку **Sites and workspaces** (Узлы и рабочие области).

На экране появится страница с перечислением всех узлов и рабочих областей документов на текущем сервере SharePoint. (Рабочие области документов позволяют совместно использовать файлы, например, документы Word и электронные таблицы Excel. У узлов тоже есть такое средство, но, кроме того, они обладают дополнительными элементами, такими как списки и совместно используемый календарь.)

Эту страницу можно использовать для просмотра и удаления других узлов. Сначала список узлов пуст, поскольку новая установка программы SharePoint стартует с единственным узлом верхнего уровня.

4. Щелкните кнопкой мыши команду **Create** (Создать) для построения нового узла.

На экран выведется страница, на которой можно настроить ваш новый узел (рис. 21.4).

5. Введите все данные об узле.

- В разделе **Title and Description** (Название и описание) определяется способ представления узла на главной странице.
- Раздел **Web Site Address** (Адрес Web-узла) позволяет выбрать URL, которым будут пользоваться для перехода на данный узел группы. Этот адрес состоит из двух частей: URL узла верхнего уровня (обычно содержащий имя сервера) и идущей следом настраиваемой части, идентифицирующей группу.
- Раздел Permissions (Разрешения) позволяет выбрать способ аутентификации пользователей (другими словами, как программа SharePoint определяет, разрешать ли им войти). Выберите переключатель **Use unique permissions** (Использовать собственные разрешения), если хотите четко управлять доступом к этому узлу группы (это самый гибкий подход). В противном случае разрешения для вашего нового узла будут такими же, как разрешения для узла верхнего уровня.
- В разделе **Navigation** (Переходы) можно выбрать, будет ли ссылка на данный узел отображаться на узле верхнего уровня.
- В разделе **Template** (Выбор шаблона) можно выбрать первоначальный вариант макета для вашего узла. Шаблон Team Site (узел группы) - неплохой выбор. После создания узла можно настраивать его, сколько душе угодно.

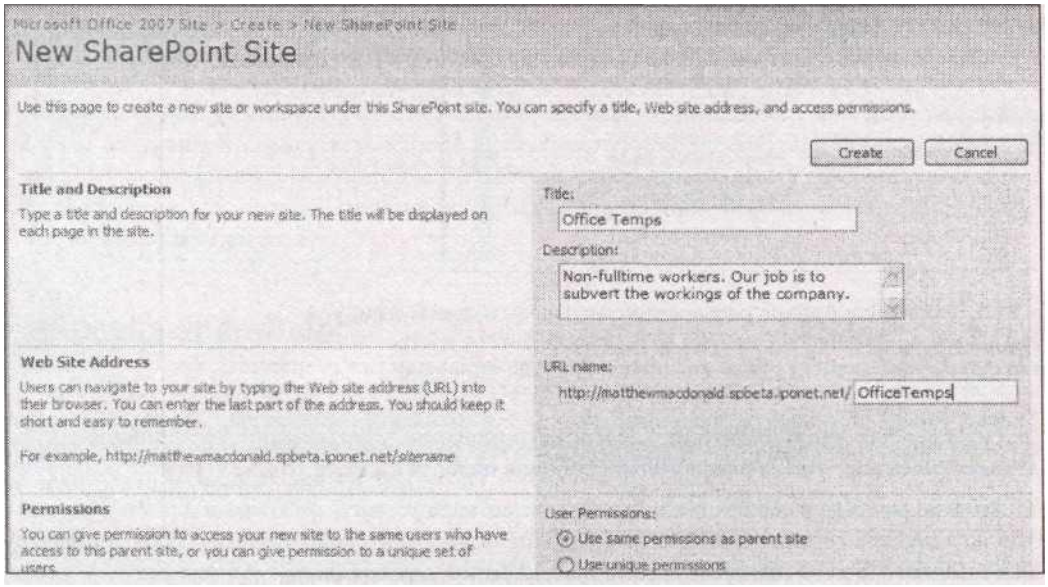


Рис. 21.4. Здесь можно ввести информацию для нового узла группы **Office Temps**

6. Щелкните мышью кнопку **Create** (Создать).

Во время создания узла на экран будет выводиться сообщение **Operation in Progress** (Операция выполняется).

Если вы выбрали переключатель **Use unique permissions** (Использовать собственные разрешения) в пункте 5, программа SharePoint отобразит новую страницу, позволяющую задать пользователей, которым разрешено использовать ваш узел.

7. Укажите группы, которым разрешается использовать узел (рис. 21.5). Узел SharePoint могут применять три сорта пользователей:

- **Visitors** (Посетители) могут читать информацию, оставленную другими пользователями, но не могут ничего менять;
- **Members** (Участники) входят в состав группы; они могут редактировать информацию в существующих списках, но не могут создавать новые списки.
- **Owners** (Владельцы) - это суперучастники; они добавляют и удаляют других пользователей, создают списки и изменяют параметры узла.

Для назначения этих разных уровней доступа различным пользователям применяются

группы. *Группы* - это компонент системы безопасности ОС Windows, позволяющий управлять одновременно большим числом пользователей. Основная идея состоит в том, что в одной группе может быть столько пользователей, сколько нужно. Самое замечательное свойство групп - их сверхгибкость. Если в компании появляется новый сотрудник, вам не нужно изменять параметры настройки узла SharePoint. Вместо этого вы просто добавляете этого сотрудника в нужную группу, и программа SharePoint точно знает, что ему разрешено, а что нет.

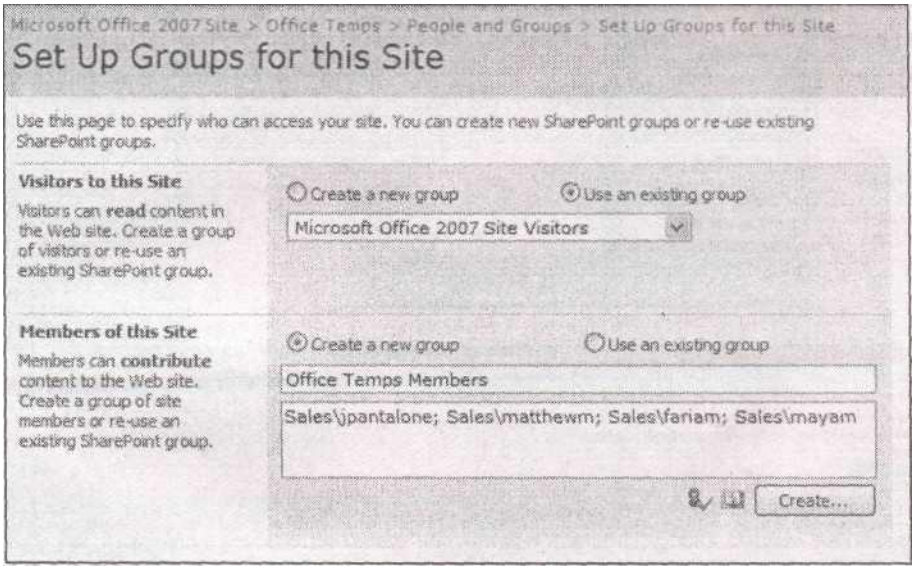


Рис. 21.5. Если у вас уже есть группа посетителей, выберите переключатель **Use an existing group** и затем введите имя группы. В противном случае выберите переключатель **Create a new group** и предоставьте список имен пользователей, отделенных точкой запятой. В данном примере новая группа называется **Office Temps Members** и создается с четырьмя пользователями

Примечание

Очень важно получить корректные имена пользователей и групп. И снова вам, возможно, придется связаться с сетевым администратором. Вероятно, потребуется задавать компьютер или домен, используемый каждым пользователем для регистрации. Таким образом, пользователь jpantalone регистрируется в домене Sales, и чтобы сделать это понятным для программы SharePoint, следует добавить имя пользователя Sales\jpantalone.

После того как данные пользователей введены, щелкните мышью кнопку ОК.

Это действие завершает процесс. Поздравляю! У вас появился новенький узел SharePoint для группы, с которым можно поиграть (рис. 21.6).

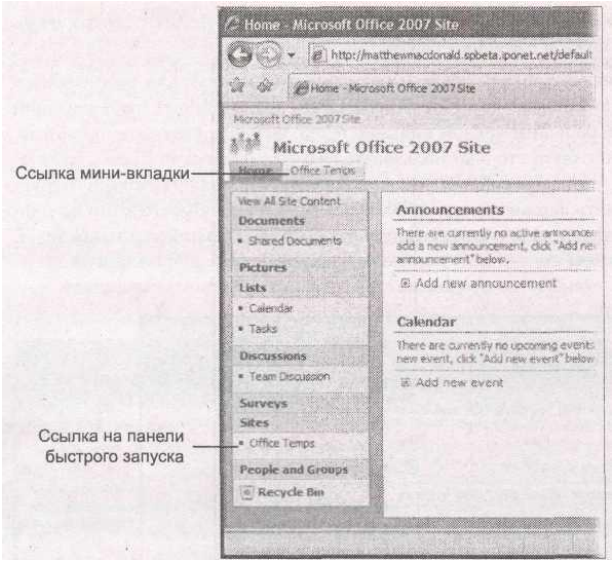


Рис. 21.6. В зависимости от выбранных вариантов в разделе **Navigation** ваш узел для быстрого доступа может отображаться в виде мини-вкладок в верхней части страниц SharePoint

Подсказка

Нового пользователя ОС Windows можно создать, не покидая страницы установки SharePoint. Просто щелкните мышью кнопку **Create** (Создать) под одним из списков пользователей. Для удобства вставки программа SharePoint может отправить по электронной почте приглашение удачливому новому участнику группы, как только вы предоставите его адрес электронной почты.

21.2.2. *Настройка вашего узла*

Узлы SharePoint до смешного легко настраиваются. И все потому, что они формируются из многочисленных компонентов с собственным содержимым, которые корпорация Microsoft называет Web-частями (Web Parts). Если хотите изменить страницу, просто добавьте новые Web-части, удалите имеющиеся или измените порядок их расположения на странице.

Для того чтобы убедиться в этом, перейдите на домашнюю страницу вашего узла группы и щелкните мышью последовательность ссылок **Site Actions → Edit Page** (Действия узла → Изменить страницу). Страница переключится в режим редактирования, как показано на рис. 21.7.

Примечание

Когда узел изменяется таким способом, ваши корректировки влияют на всех. Любой владелец узла может настроить узел SharePoint своей группы.

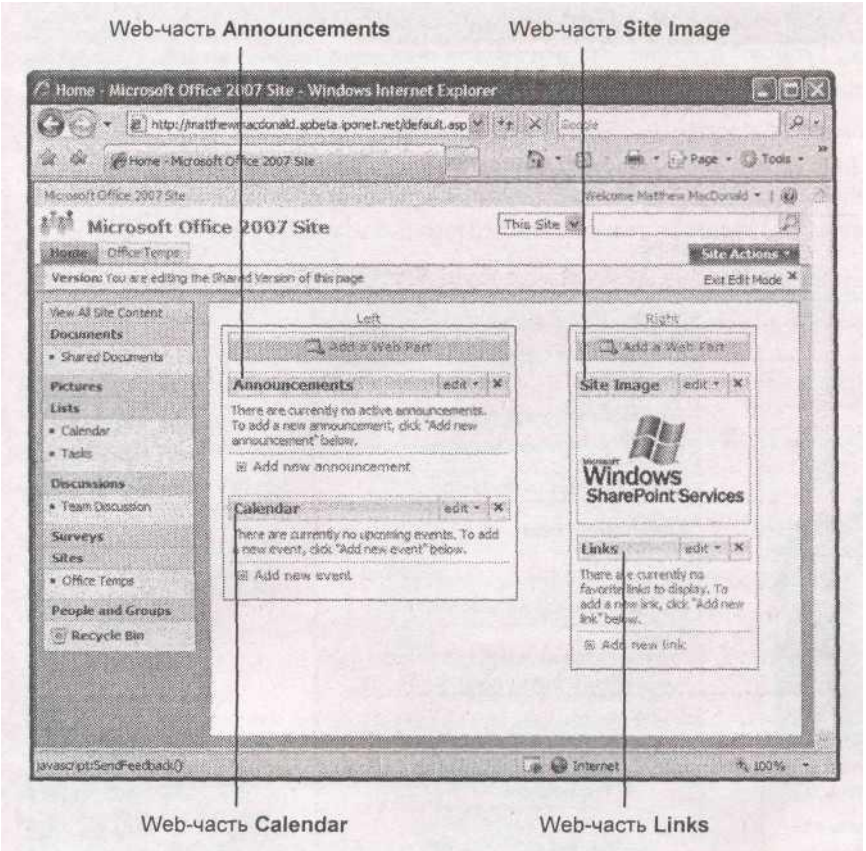


Рис. 21.7. В режиме редактирования ясно видны отдельные области вашей страницы и местоположение каждой Web-части. Эта ничем не примечательная страница состоит из четырех Web-частей

В режиме редактирования можно изменить многое. Вы можете следующее.

■ *Переместить Web-часть.* Просто щелкните кнопкой мыши полосу заголовка (например, текст "Announcements" (извещения)) и перетащите его с нажатой кнопкой мыши в другое

место на странице.

■ *Разрешить сворачивание Web-части.* Когда Web-часть свернута, на странице виден только заголовок Web-части, что позволяет освободить значительное место. Если пользователи, работающие на странице, решат применить Web-часть, они могут для вывода на экран Web-части просто щелкнуть кнопкой мыши направленную вниз стрелку, расположенную рядом с ее заголовком.

■ *Удалить Web-часть.* Для того чтобы закрыть Web-часть, щелкните кнопкой мыши пиктограмму ? в правом верхнем углу Web-части. Вы всегда сможете вернуть на экран Web-часть позже.

■ *Изменить Web-часть.* В разных Web-частях можно настраивать различные параметры, в том числе отображение Web-части, включенные в нее элементы, их поведение и т. д. Для изменения Web-части щелкните мышью кнопку edit (Правка) в правом верхнем углу Web-части и выберите **Modify Shared Web Part** (изменить совместно используемую Web-часть) ("совместно используемую" означает, что вся группа пользователей видит эту Web-часть, поэтому вы изменяете ее на свой страх и риск.) На рис. 21.8 показан пример.

Добавить Web-часть. У программы SharePoint есть библиотека полезных Web-частей, как показано на рис. 21.9. (И, безусловно, инициативные программисты могут добавить собственные Web-части.) Для выбора новой Web-части щелкните мышью кнопку **Add Web Part** (Добавить Web-часть) в том разделе страницы, где хотите ее отобразить.

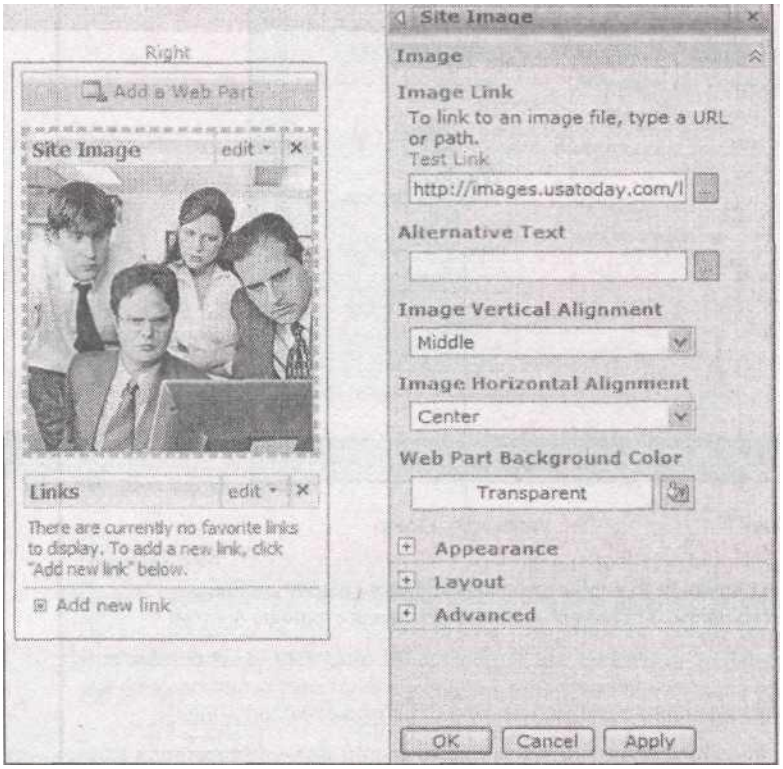


Рис. 21.8. Когда Web-часть изменяется, вокруг нее появляется штриховая рамка. Параметры этой Web-части отображаются на панели справа. В данном примере Web-часть **Site Image** снабжается новым изображением

21.3. *SharePoint и Access*

Теперь, когда вы увидели программу SharePoint в целом, пора довести до совершенства ее средство, названное *списками.* Вкратце списки SharePoint разработаны, чтобы помочь вам отслеживать самую разную информацию, которую необходимо коллективно использовать вместе с вашей группой. Некоторые из ключевых средств программы SharePoint, такие как ссылки, задачи, извещения, контакты и даже календарь, - на самом деле заранее сформированные списки.

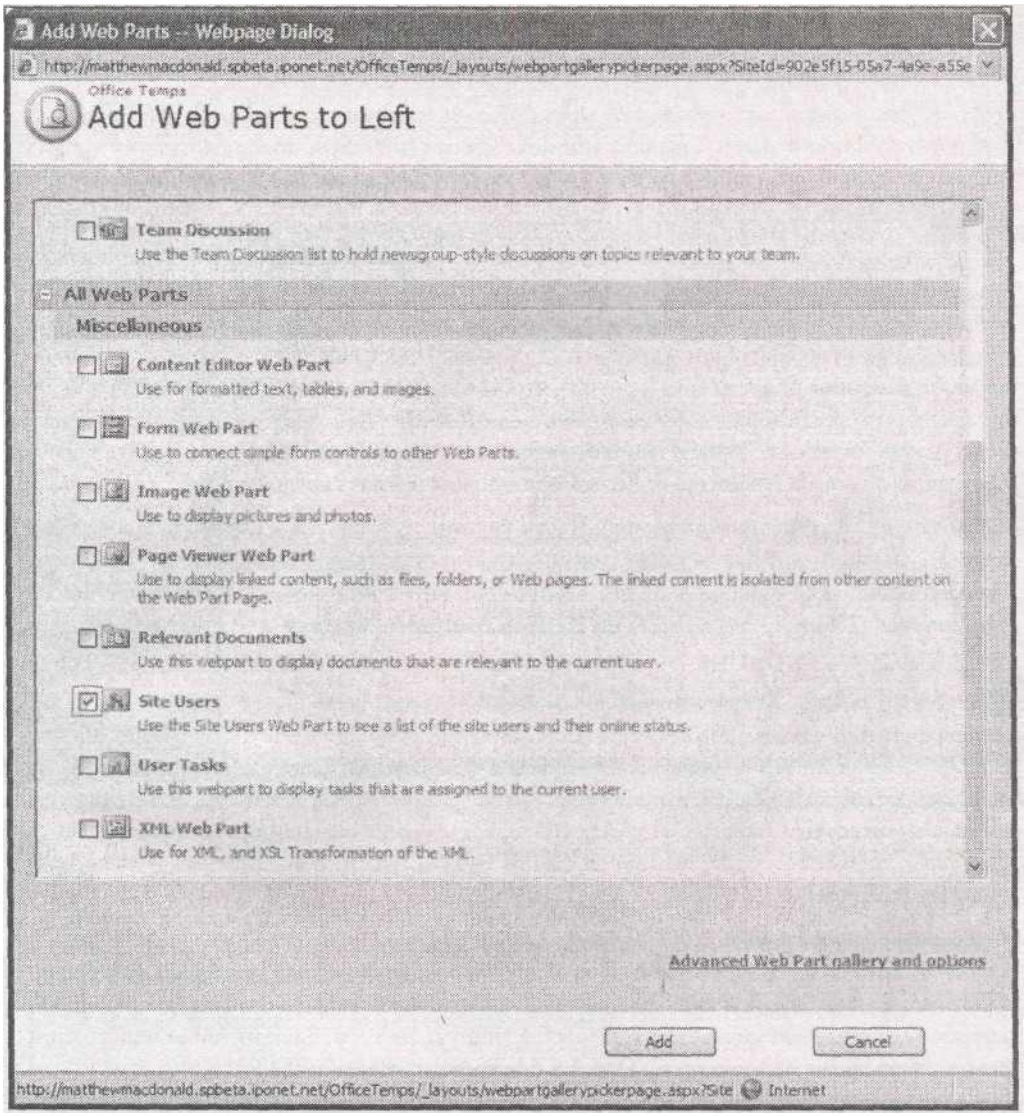


Рис. 21.9. В данном примере Web-часть добавляется в раздел, названный **Left.** К Web-частям, первоначально не включенным в домашнюю страницу узла группы, относятся Web-часть **Site Users** (перечисляющая членов группы и отображающая текущего пользователя), **User Tasks** (отображающая только те задачи, которые назначены текущему пользователю) и **Team Discussion** (предоставляющая для беседы форум в стиле доски сообщений)

На профессиональном уровне.

Пять интересных инструментов программы SharePoint, которые стоит опробовать

Остаток данной главы посвящен средствам программы SharePoint, которые действуют в сочетании с Access, а именно спискам.

Однако вы не должны прекращать изучение SharePoint в этот момент. Если вы все еще заинтригованы, проверьте действие следующих инструментов программы SharePoint.

- *Назначить собрание с помощью календаря.* В Web-части **Calendar** (Календарь) щелкните кнопкой мыши ссылку **Add new event** (Добавить новое событие). Можно ввести название, описание, место и время вашего собрания. Вы даже можете создать рабочую область для хранения заметок.
- *Получить немедленные уведомления с помощью оповещения.* **Alerts** (Оповещения) позволяют уведомлять пользователей, когда что-то меняется на узле SharePoint (появилось новое событие, назначенная задача, извещение или что-нибудь еще). Например, можно добавить оповещение в календарь. Просто щелкните кнопкой мыши заголовок **Calendar** (Календарь) для того, чтобы открыть представление календаря, и затем выберите последовательность **Actions → Alert Me** (Действия → Оповещать меня). Программа попросит указать тип изменений, которые вас интересуют, и задать адреса электронной почты, по которым SharePoint должна послать уведомления.

• *Совместно использовать документ.* Часто группы нуждаются в пересылке спецификаций, отчетов и других деловых документов по цепочке рецензирования. Программа SharePoint облегчает эту задачу. Просто щелкните кнопкой мыши ссылку **Shared Documents** (Общие документы) на **Панели быстрого запуска** для перехода в центр документов, где можно просмотреть все, что уже есть там, и загрузить свою работу.

• *Назначить задачу.* Когда есть работа, которая должна быть сделана, помогает наличие эффективного способа координации действий сотрудников. В SharePoint это означает щелчок кнопкой мыши ссылки **Tasks** (Задачи) на **Панели быстрого запуска** для просмотра списков невыполненных задач. Можно создать и назначить новые задачи с множеством важной отслеживающей информации (например, приоритетом, статусом, процентом завершенности, датой начала и датой завершения и даже вложенным файлом).

• *Совместно использовать контактную информацию.* Нуждается ли ваша группа в поддержании постоянных контактов с одними и теми же людьми? В программе SharePoint есть заранее подготовленный список **Contacts** (Контакты), созданный специально для этой цели. Для того чтобы глянуть на него, щелкните кнопкой мыши ссылку **Contacts** (Контакты) на **Панели быстрого запуска.**

Подсказка

Для того чтобы просмотреть все списки узла группы, найдите раздел Lists (Списки) на Панели быстрого запуска, расположенной слева. Щелкните кнопкой мыши заголовок Lists (Списки) на этой панели для отображения более подробного перечня списков (рис. 21.10).

21.3.1. *Формирование списка*

Создать список с помощью интерфейса Web-страницы SharePoint так же просто, как таблицу в программе Access. Вот как это делается.

1. Выберите последовательность **Site Actions → Create** (Действия узла → Создать). На экране появится страница управления узла.

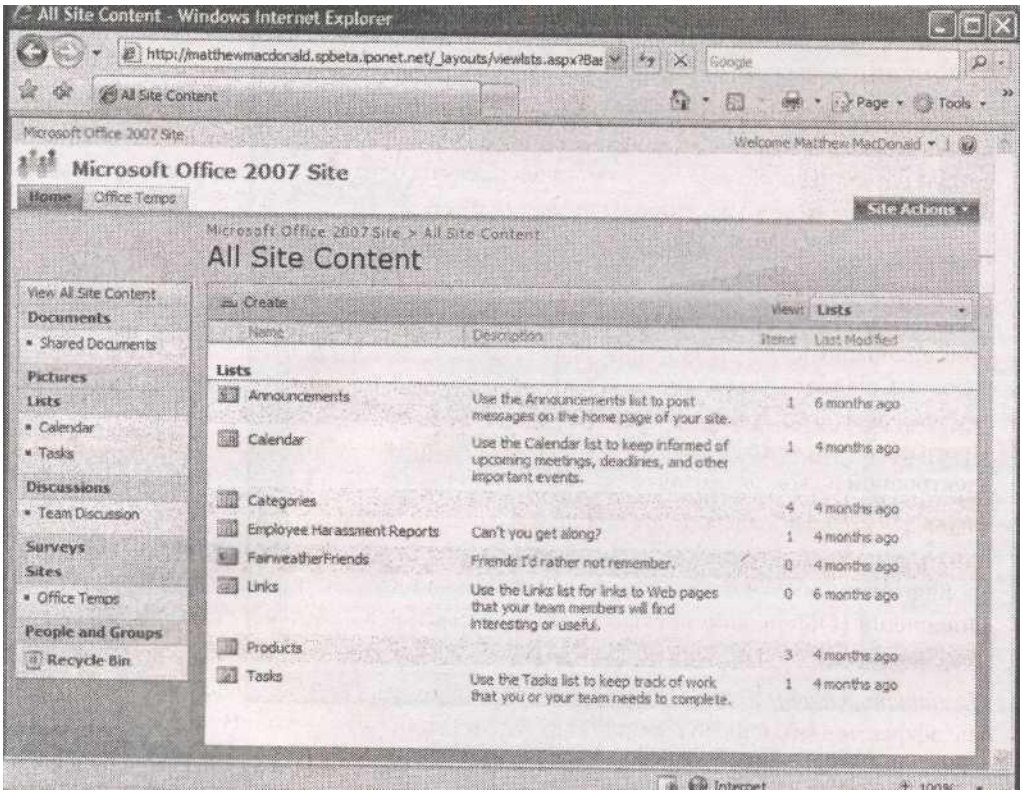


Рис. 21.10. На этой странице показаны все списки узла, число элементов в каждом списке и время последнего сделанного изменения

2. В разделе **Custom Lists** (Настраиваемые списки) щелкните кнопкой мыши ссылку **Custom List** (Настраиваемый список).

На экране отобразится страница создания списка.

3. Введите имя и описание вашего списка.

Например, можно создать список, названный CafeteriaMenuIdeas (соображения о меню кафетерия), содержащий элементы, которые сотрудники хотели бы видеть в продаже в буфете компании. Или использовать список **DodgeballTeamAssignment** (хитрости распределения в бейсбольной команде), чтобы узнать, кто собирается сразиться с шефом.

4. Решите, должен ли список отображаться на панели быстрого запуска.

Если вы решили поместить его на **Панель быстрого запуска,** список будет виден на всех страницах. В противном случае, чтобы найти его, придется перейти в раздел списков (щелкнуть мышью заголовок **Lists** (Списки)).

5. Щелкните мышью кнопку **Create** (Создать).

Программа SharePoint создаст список и мигом перебросит вас на страницу ввода элементов списка, где вы сможете начать заполнять список данными (рис. 21.11) или изменять его структуру.

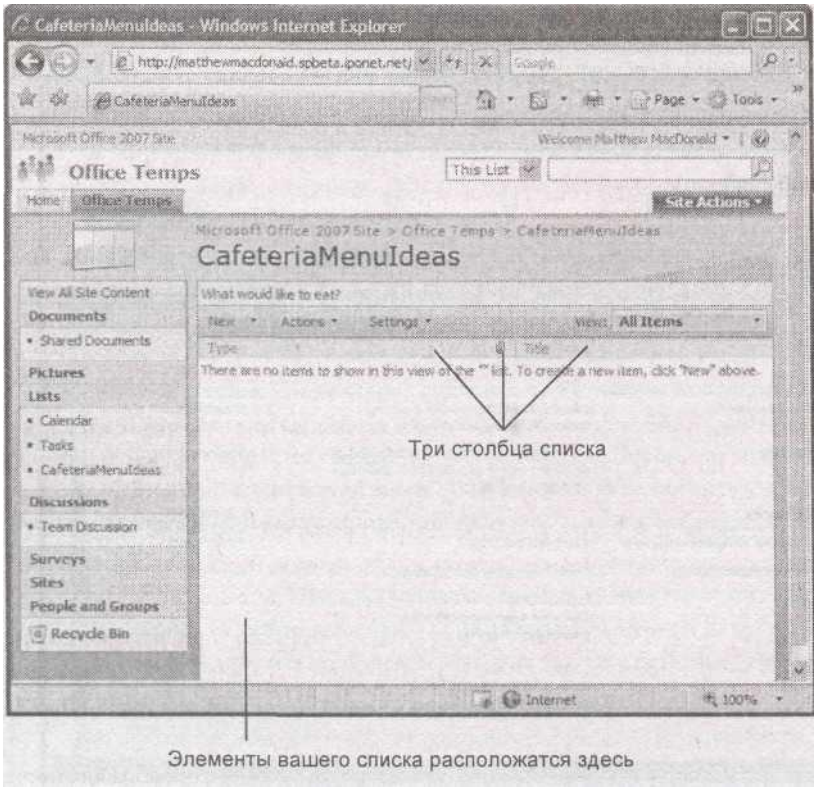


Рис. 21.11. Эту страницу используют всегда при добавлении данных в список или изменении структуры списка (добавлением или изменением столбцов)

6. Создайте нужные вашему списку столбцы.

У всех списков сначала есть три столбца: **Type** (Тип) (используется программой SharePoint для различения объектов разных типов), **Attachments** (Вложения) (позволяет присоединить любые файлы, связанные с элементом) и **Title** (Название) (описательная текстовая строка). Первые два столбца обязательные, а столбец **Title** (Заголовок) можно удалить (как, вы узнаете в пункте 7).

Примечание

У всех списков SharePoint на самом деле есть несколько дополнительных столбцов, которые программа Access держит за кадром. Например, у каждого элемента списка есть уникальный скрытый ID (Код) и столбцы, отслеживающие автора вставленного элемента и время последнего изменения, внесенного в элемент.

Для добавления столбца выберите последовательность **Settings → Create Column** (Параметры → Создать столбцы). Введите всю информацию, необходимую для нового столбца, включая его имя, описание и тип данных (рис. 21.12). Можно указать максимальную длину данных, значение по умолчанию и определить, является ли поле обязательным (и не может

быть пустым).

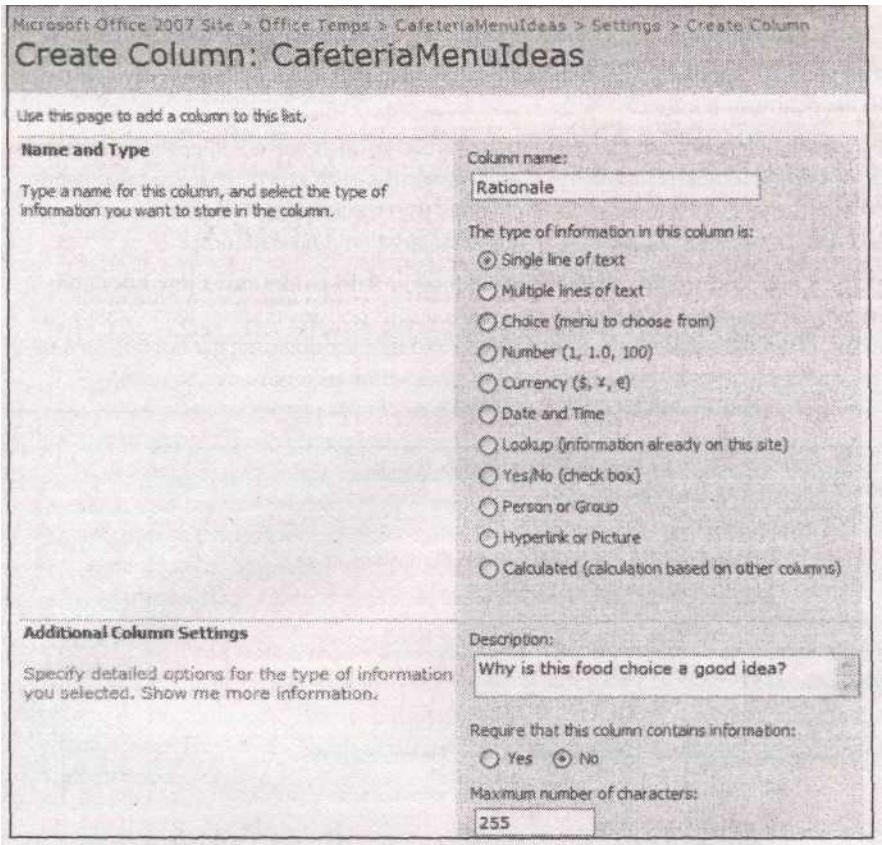


Рис. 21.12. Типы данных в списках SharePoint соответствуют типам данных в таблицах Access (хотя у вас не так много вариантов для выбора, как в программе Access). Если вы хотите извлечь значения из другой таблицы, создайте столбец подстановки и укажите список, на который хотите сослаться

Примечание

Программа SharePoint не так строга в отношении ссылочной целостности, как Access. Например, если вы создали столбец подстановки, связанный с другим списком, а затем удалили последний, программа SharePoint просто очистит все связанные значения.

7. При желании можно изменить параметры вашего списка, щелкнув кнопкой мыши последовательность **Settings → List Settings** (Параметры → Параметры списков).

Этот шаг открывает страницу параметров списка (рис. 21.13), на которой можно выполнить множество полезных настроек.

Малоизвестная или недооцененная возможность.

Представления SharePoint = запросы Access

На жаргоне SharePoint *представление* - это настраиваемый способ отображения данных в списке. Представления могут выводить подмножество полного набора столбцов, применять фильтры для ограничения списка только интересующими вас строками.

Можно также использовать сортировку, группировку и итоги. По существу представление SharePoint играет ту же роль, что и универсальный запрос на выборку программы Access, который вы освоили в *главе 6.*

Создать новые представления для вашего списка можно, выбрав во время просмотра списка **Settings → List Settings** (Параметры → Параметры списков). Этот шаг приведет к отображению страницы, битком набитой параметрами списка. У вновь созданных списков только одно представление, называемое *представлением по умолчанию.*

В процессе создания нового столбца вы увидите флажок **Add to default view** (Добавить в представление по умолчанию). Именно в представлении по умолчанию вы первый раз открываете список. Если не добавить ваш столбец в это представление, вы вообще его не

увидите (если не создадите новое представление с включенным в него столбцом).

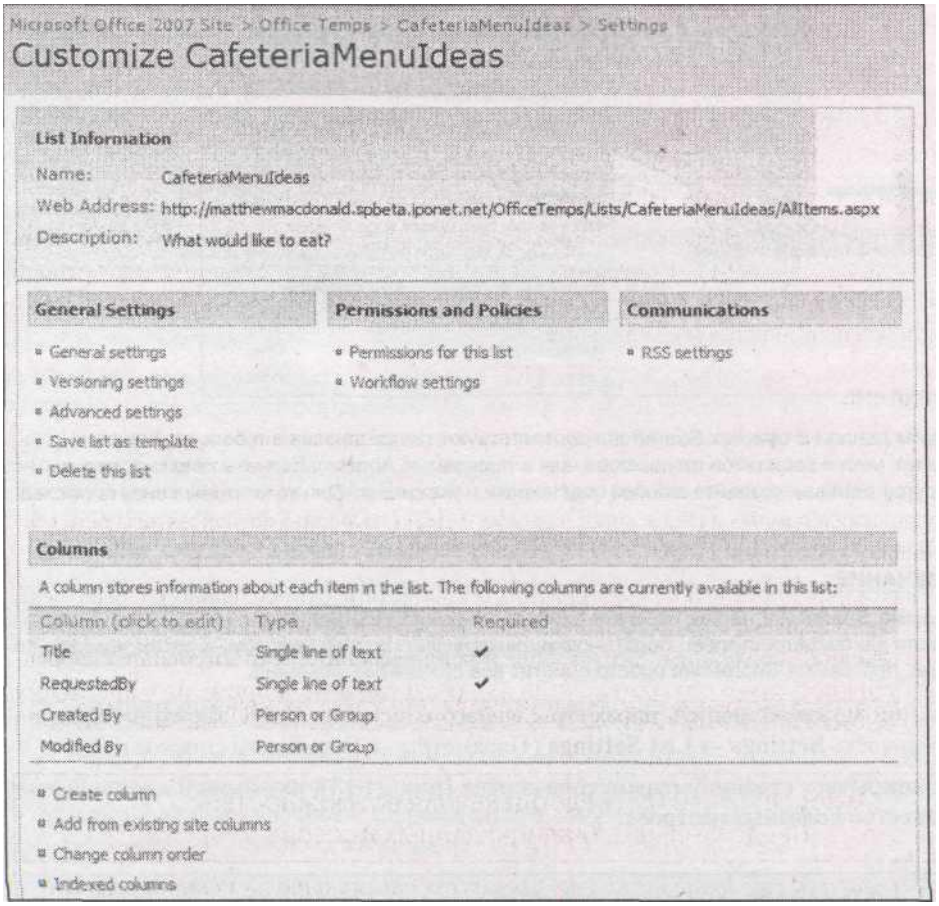


Рис. 21.13. В первом разделе страницы параметров списка находятся ссылки, позволяющие настраивать различные параметры. Ниже располагается раздел, позволяющий просматривать текущие столбцы, удалять ненужные и добавлять новые. Наконец, в нижней части страницы (не показанной на рисунке) находится раздел, позволяющий просматривать представления и создавать собственные

К самым полезным ссылкам на странице параметров списков относятся следующие.

- **General settings** (Общие параметры) позволяет изменять информацию, предоставленную в пунктах 3 и 4.
- **Advanced settings** (Дополнительные параметры) позволяет определить, ограничен ли пользователь только чтением и изменением собственной введенной информации. Можно отключить средство, обеспечивающее вложение файлов (позволяющее пользователям подсоединять собственные файлы к элементу), и инструмент работы с папкой (позволяющий пользователям создавать вложенные папки для большей упорядоченности элементов списка).
- **Delete this list** (Удалить этот список) - эта ссылка не нуждается в объяснениях.
- **Permissions for this list** (Разрешения) позволяет управлять пользователями данного списка, которым разрешено редактирование списка, и разрешенными им действиями со списком. Первоначально владельцы узла группы обладают разрешением на полный контроль (т. е. они могут делать что угодно, включая изменение параметров списка), у обычных участников есть разрешение на участие (contribute permission) (т. е. они могут добавлять, редактировать и удалять элементы списка), а посетители обладают разрешением на чтение (т. е. они могут только просматривать существующую информацию).

8. Когда нужные столбцы созданы, можно вводить данные.

Для добавления новой записи щелкните кнопкой мыши ссылки **New → New Item** (Новый → Новый элемент). Программа SharePoint отобразит страницу, на которой можно ввести значения для всех ваших столбцов (рис. 21.14). Вы также можете выбрать элемент для редактирования или удаления.

Примечание

Программа SharePoint разработана для одновременного использования большим числом пользователей (или параллельного выполнения операций), поэтому вполне возможно, что вы не будете единственным пользователем, редактирующим список. Для того чтобы увидеть последние добавления и изменения, сделанные другими людьми, выберите последовательность ссылок **Actions → Refresh Data** (Действия → Обновить данные).

Средства параллельной обработки у программы SharePoint относительно слабые. Если одновременно два пользователя редактируют один и тот же элемент списка, тот, кто пытается сохранить изменения последним, получает сообщение об ошибке, отклоняющее корректировку. (Ситуация несколько улучшится, если для редактирования применить программу Access.)

Малоизвестная или недооцененная возможность.
Представление таблицы данных Access

Если вы хотите видеть па экране несколько элементов списка одновременно и редактировать их все сразу на листе данных в стиле программы Access, можно воспользоваться представлением таблицы данных Access (Access Web Datasheet), специализированной надстройкой Web-обозревателя, усовершенствующей программу SharePoint. Представление таблицы данных Access, в отличие от других страниц SharePoint, доступно, только если у вас на данном компьютере установлен пакет Office 2007.

Представление таблицы данных Access выглядит как обычный лист данных Access, встроенный в ваш Web-обозреватель. Можно переходить от строки к строке, от столбца к столбцу, вносить изменения куда угодно, перемещать столбцы с места на место с помощью кнопки мыши и применять сортировку посредством направленных вниз стрелок, расположенных рядом с заголовками столбцов. Вы также можете быстро добавить или удалить столбцы (снова щелкнув правой кнопкой мыши заголовок столбца). Если вам нравится программа SharePoint, но не хватает комфорта, предоставляемого программой Access, представление таблицы данных Access поможет вам почувствовать себя как дома.

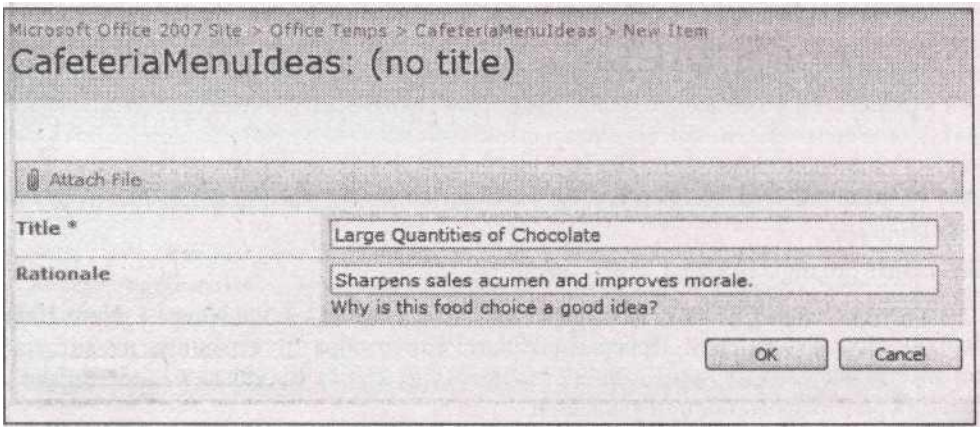


Рис. 21.14. На данной странице приведен новый элемент для списка CafeteriaMenuldeas

21.3.2. *Экспорт таблицы в SharePoint*

Есть и другой способ создания списка SharePoint. Можно начать с таблицы Access и экспортировать ее в программу SharePoint. Недостаток этого подхода заключается в необходимости небольшого преобразования типов данных Access в типы данных SharePoint. Некоторые важные детали (например, условия на значения и маски ввода) могут быть потеряны, поэтому настраивать любой из этих параметров в программе Access не имеет смысла. Тем не менее экспорт таблицы из Access - прекрасный выход, если у вас есть кое-какие данные, которые необходимо передать на узел группы, чтобы больше людей могли ими воспользоваться.

Процесс экспорта в программу SharePoint в основном такой же, как процесс экспорта данных других типов, с которыми вы познакомились в *главе 19*. Вот как он работает.

1. Откройте файл БД Access.
2. В области переходов выделите таблицу, которую хотите экспортировать.

Если экспортируется подчиненная таблица, программа Access автоматически экспортирует и все связанные родительские таблицы. Например, если вы экспортируете таблицу **Products,** вместе с ней отправляется и таблица **ProductCategories.**

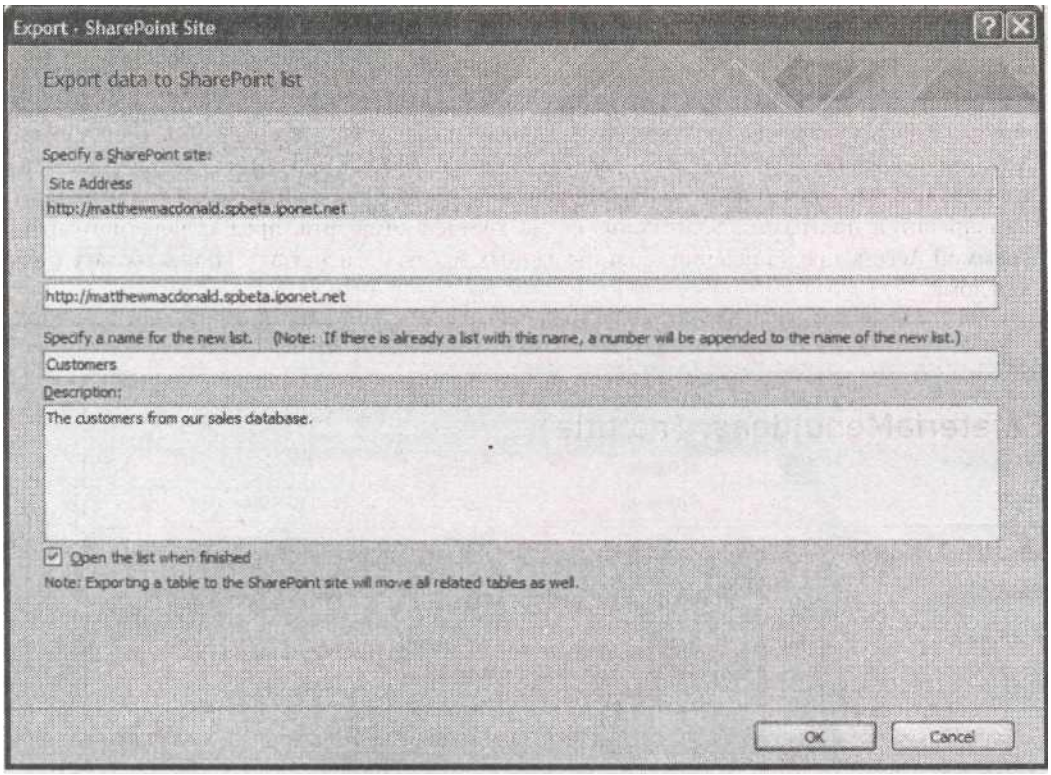


Рис. 21.15. В данном примере программа Access готова копировать таблицу **Customers** в программу SharePoint

3. Выберите на ленте **External Data → Export → SharePoint List** (Внешние данные → Экспорт → Список SharePoint).

Запускается мастер SharePoint Export wizard (Экспорт - узел SharePoint) (рис. 21.15).

4. Введите URL-адрес узла SharePoint для группы, заголовок для списка и (при желании) описание.

Это базовые параметры списка.

5. Если хотите увидеть список в программе SharePoint, когда процесс будет завершен, установите флажок **Open the list when finished** (Открыть список по окончании экспорта)

Всегда неплохо просмотреть список после операции пересылки, чтобы убедиться в том, что он функционирует так, как вы ожидали.

6. Щелкните мышью кнопку ОК.

Если для доступа к узлу SharePoint нужен пароль, следует ввести его сейчас. Затем программа Access создает новый список SharePoint и заполняет его данными. Ваша БД Access в любом случае не изменяется.

Если флажок **Open the list when finished** (Открыть список по окончании экспорта), упомянутый в пункте 5, установлен, программа Access по завершении процесса откроет окно Web-обозревателя для отображения нового списка (рис. 21.16).

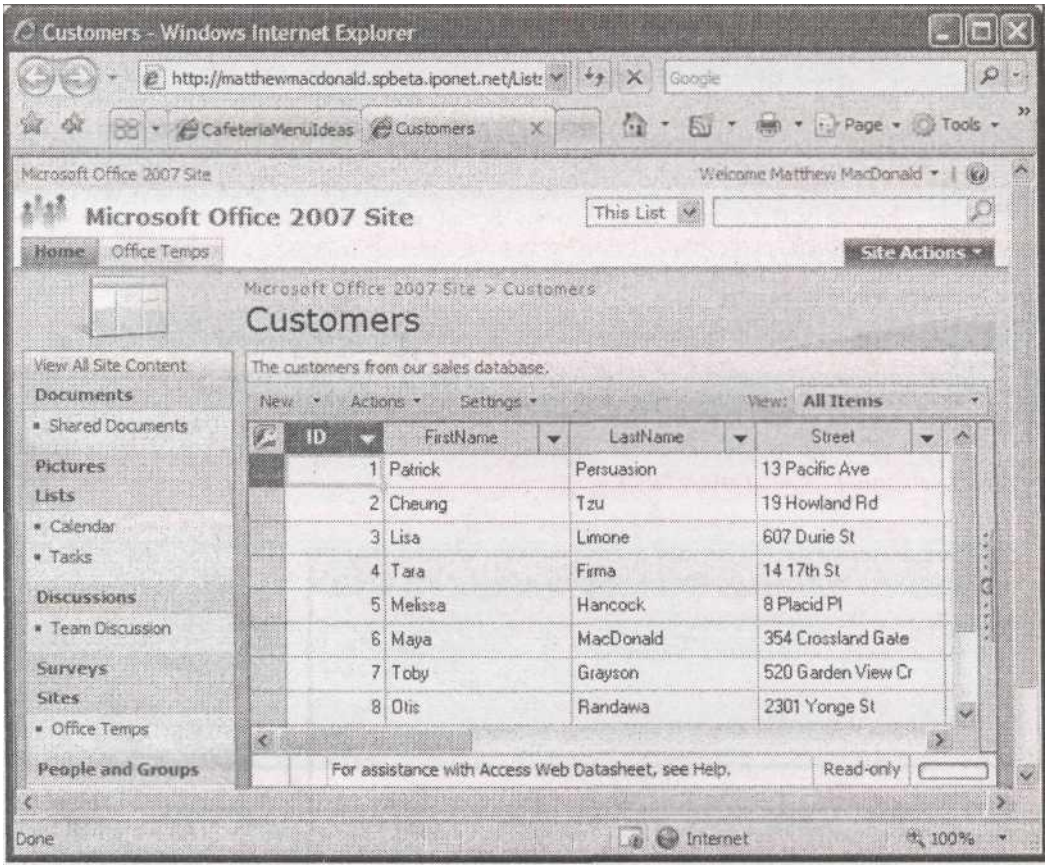


Рис. 21.16. Экспортированная из Access таблица **Customers** представлена как список SharePoint. Когда программа Access открывает список для просмотра, она автоматически применяет представление таблицы данных Access, что делает отображение похожим на интерфейс листа данных Access

Когда экспорт завершен, программа Access предлагает сохранить шаги экспорта. Если вы соглашаетесь, эту же операцию экспорта можно будет повторить позднее (возможно, для переноса самой свежей копии данных на сервер).

Примечание

Когда данные экспортируются в программу SharePoint, создается копия этих данных. Это означает, что при редактировании списка SharePoint ваша БД никак не меняется. Аналогично, если изменяется БД, эти изменения не вносятся в список до тех пор, пока вы не экспортируете их все снова. Если это не то, что вам нужно, рассмотрите возможность сохранения данных в программе SharePoint и управление ими в программе Access с помощью связанных таблиц. В следующем разделе приводится дополнительная информация об этом методе.

21.3.3. *Импорт данных в Access*

Есть два варианта для импорта данных в программу Access. Можно использовать процесс импорта, описанный в *главе 19*. Он позволит сохранить шаги импорта и повторить их в дальнейшем. Но есть и другой способ, более удобный, поскольку не требует указания URL-адреса списка SharePoint. Можно выполнить экспорт непосредственно с узла SharePoint вашей группы.

Примечание

Этот вариант доступен, только если у вас на рабочем компьютере установлена программа Access. Если вы переходите на Web-страницы SharePoint на чьем-либо компьютере, не имеющем установленной программы Access, экспорт выполнить не удастся.

Вот как действует этот метод.

1. На узле группы найдите список, который хотите импортировать.
2. Выберите последовательность ссылок **Actions → Open with Microsoft Access** (Действия → Открыть с помощью Microsoft Access).

На экране появится диалоговое окно, позволяющее выбрать БД и режим копирования или связывания данных (рис. 21.17).

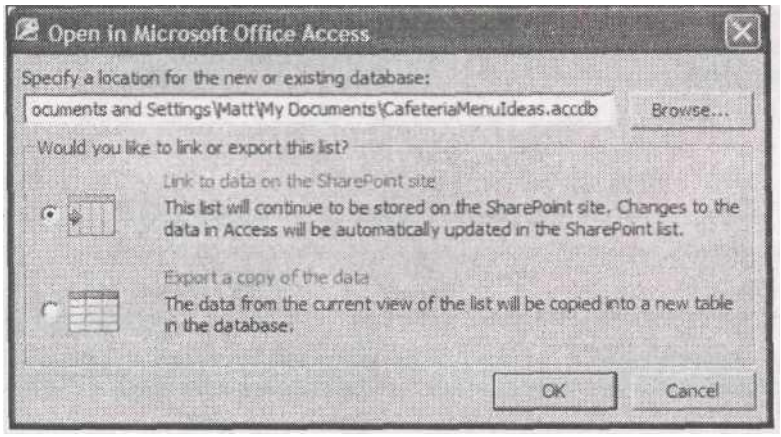


Рис. 21.17. В данном примере таблица **CafeteriaMenuldeas** экспортируется с узла SharePoint. Программа Access даже пока не открыта

3. Введите имя БД, которую хотите использовать.

Если задан несуществующий файл БД, программа Access создаст его. (Это обычный вариант.) Если же указывается существующая БД, Access добавляет таблицу в эту БД.

4. Выберите импорт копии списка или создание связанной таблицы.

В случае связанной таблицы данные всегда хранятся в программе SharePoint. Программа Access применяется для их модификации. В этом варианте существует единственная копия данных и все выполняют корректировки в одном и том же месте.

В случае создания копии у вас появляются два отдельных набора данных (список SharePoint и таблица в вашей БД), которые могут изменяться независимо друг от друга. Вы не сможете синхронизировать эти две порции данных. Преимущество такого подхода в том, что для внесения изменений нет необходимости сохранять подключение к серверу SharePoint.

5. Когда закончите, щелкните мышью кнопку **ОК**.

На вашем компьютере откроется программа Access с выбранной в пункте 3 базой данных. Возможно, потребуется повторная регистрация на узле SharePoint. Затем связанная или скопированная таблица создается и вносится в вашу БД (рис. 21.18).

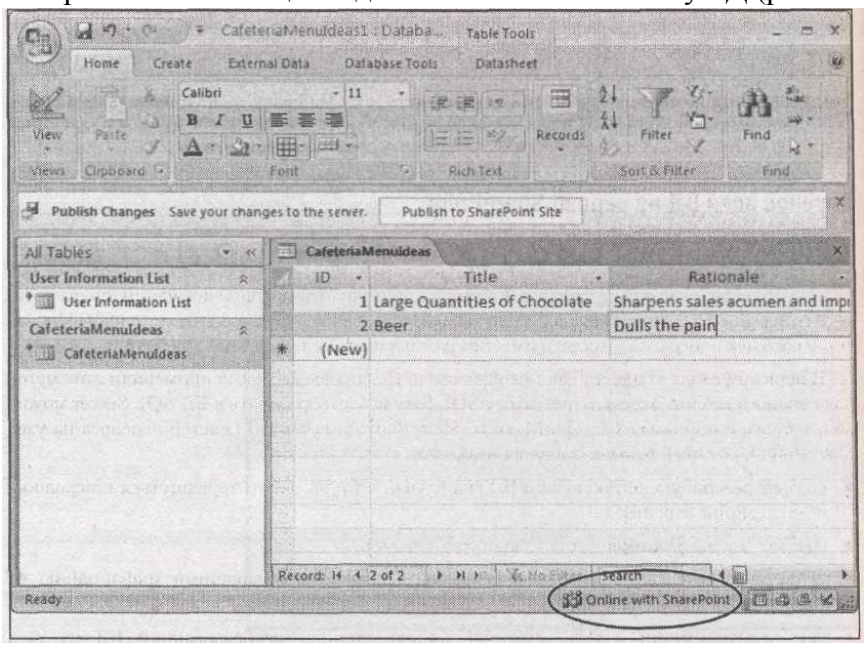


Рис. 21.18. Теперь таблицу CafeteriaMenuldeas можно редактировать прямо в программе

Access. Сообщение "Online with SharePoint" (подключение к SharePoint) в правом нижнем углу строки состояния Свидетельствует о том, что все нормально - можно вносить изменения и они будут немедленно сохранены на сервере

Вместе с выбранной (в пункте 1) таблицей программа SharePoint также экспортирует таблицу с именем **User Information List** (список сведений о пользователе), которая включается в вашу БД. Эта таблица содержит участников вашего узла SharePoint, что важно, т. к. у всех списков SharePoint есть два скрытых поля (Created By (кем создано) и **Modified By** (кем изменено)), обозначающих создателя элемента и пользователя, внесшего последние изменения в элемент. Вам не нужно беспокоиться об этих подробностях, поскольку Access поддерживает их автоматически (несмотря на то, что их можно увидеть на листе данных, выбрав на ленте последовательность команд **Home → Records → More → Unhide Columns** (Главная → Записи → Дополнительно → Отобразить столбцы)).

Когда начнете использовать вашу БД, вы заметите на панели сообщений в верхней части окна (рис. 21.18) кнопку **Publish to SharePoint Site** (Опубликовать на узле SharePoint). Щелкните мышью эту кнопку для сохранения копии файла БД Access в библиотеке документов (Document Library) на узле SharePoint, откуда другие пользователи смогут ее загрузить.

Примечание

Программа SharePoint не поддерживает формы и отчеты Access. Вы можете вставить эти объекты в вашу БД, но у пользователей, применяющих Web-страницы SharePoint, нет никаких возможностей использовать их. Если хотите предоставить свои формы и отчеты для коллективного использования, необходимо раздать копии вашей связанной БД всем пользователям, имеющим программу Access, или использовать кнопку **Publish to SharePoint Site** (Опубликовать на узле SharePoint). Если нет необходимости в коллективном использовании клиентской части БД, то и для применения этого средства нет причин.

21.3.4. *Перенос всей БД на сервер SharePoint*

Зачем останавливаться на одной таблице? С помощью программы Access можно преобразовать всю БД в набор списков SharePoint. Это замечательный способ преобразования БД. Например, если у вас есть удачная БД, которая применяется в вашей компании, но вы хотите убедиться в том, что она способна поддерживать большее число пользователей (включая тех, у кого нет программы Access), имеет смысл передать ее программе SharePoint.

В первый раз вы увидели, как преобразовать БД в *главе 20,* когда применяли этот метод для переноса таблиц Access в программу SQL Server. Преобразовать в БД SQL Server можно почти также с помощью мастера Move to SharePoint Site Wizard (мастер переноса на узел SharePoint), который решает следующие задачи:

- создает резервную копию вашей БД (на случай если вы захотите вернуться к исходной, не-SharePoint версии);
- создает список SharePoint для каждой таблицы в БД;
- удаляет ваши таблицы и заменяет их связанными, получающими информацию от SharePoint. В этом случае все данные находятся "в умелых руках" сервера SharePoint;
- при необходимости может загрузить копию данной преобразованной БД на узел SharePoint. Другие пользователи программы Access, возможно, захотят ее использовать, если им понадобятся ваши запросы, формы, отчеты или программные процедуры.

Предложенная далее последовательность действий проведет вас через весь процесс.

1. В программе Access откройте БД, которую хотите преобразовать.
2. Выберите на ленте **External Data → SharePoint Lists → Move to SharePoint** (Внешние данные → Списки SharePoint → Переместить на Web-узел SharePoint).

Запустится мастер Move to SharePoint Site Wizard (мастер переноса на узел SharePoint) (рис. 21.19).

3. Введите URL-адрес вашего узла SharePoint для группы.
4. Если хотите дать возможность пользователям программы Access применять ваши формы и отчеты, установите флажок **Save a copy of my database to the SharePoint site and create**

shortcuts to my Access forms and reports (Сохранение копии базы данных на узле SharePoint и создание ярлыков на формы и отчеты Access).

В этом случае другие пользователи могут загрузить вашу БД и использовать другие содержащиеся в ней объекты. При этом не будет проблем с синхронизацией, поскольку загруженная БД применяет связанные таблицы. Это означает, что все данные всегда хранятся на сервере SharePoint, независимо от того, где вы вносите изменения, в программе Access или на Web-страницах SharePoint.

Если вы не хотите разбираться с этим пунктом (в вашей БД могут быть только таблицы или в вашей компании никто не пользуется программой Access), сбросьте флажок и переходите к пункту 5.

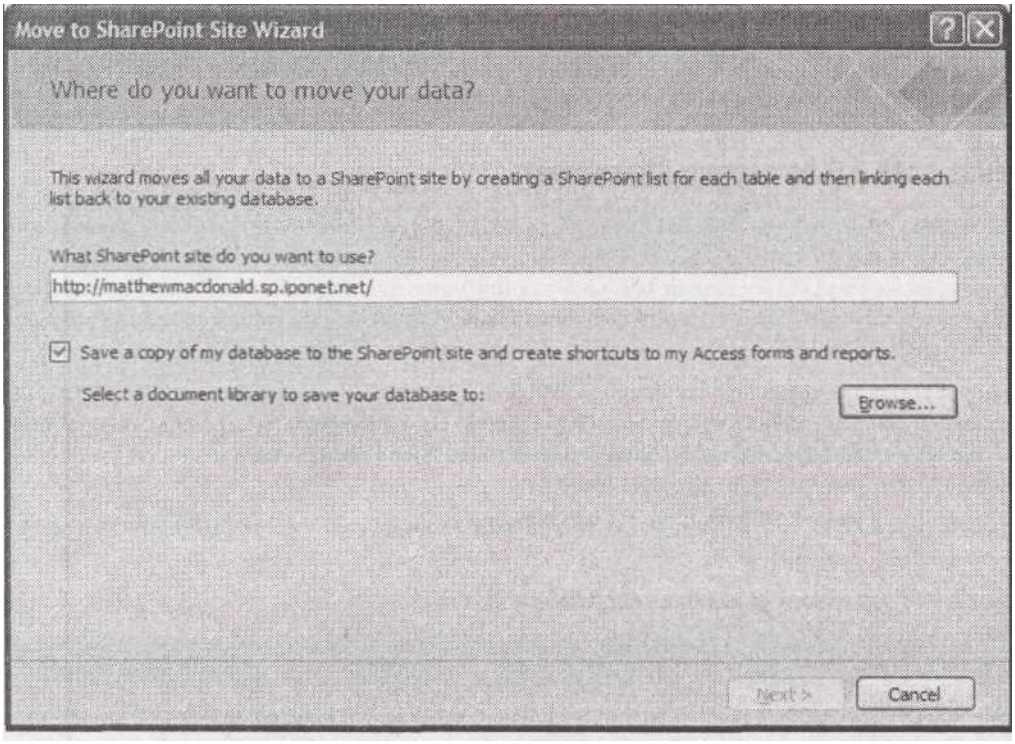


Рис. 21.19. Этот одношаговый мастер облегчает перенос целой БД с ценной информацией на узел SharePoint

5. Щелкните мышью кнопку **Browse** (Обзор) и укажите место хранения загружаемой копии вашей БД.

Если у вас для этой цели не создана никакая новая библиотека документов, необходимо использовать общедоступный раздел **Shared Documents** (Общие документы). До тех пор пока вы не выберите место хранения, кнопка **Next** (Далее) остается недоступной.

6. Щелкните мышью кнопку **Next** (Далее).

Если для доступа к узлу SharePoint нужен пароль, теперь необходимо его ввести. Затем программа Access начнет процесс передачи, который может занять какое-то время в случае БД большого объема. Индикатор выполнения процесса поможет вам узнать, сколько осталось работы.

Когда Access закончит преобразование, вы увидите завершающее окно подтверждения.

7. Для отображения конкретных действий программы Access установите флажок **Show Details** (Показать подробности) (рис. 21.20).

Если во время процесса преобразования возникают какие-либо проблемы, программа Access создает таблицу, названную **Move to SharePoint Site Issues** (перейти к неполадкам узла SharePoint). Каждая запись этой таблицы описывает проблему и причину ее возникновения.

Примечание

Программа SharePoint не поддерживает строго целостность данных. Если вы публикуете БД, применяющую это свойство, то в результате получите несколько предупреждающих сообщений в таблице **Move to SharePoint Site Issues.**

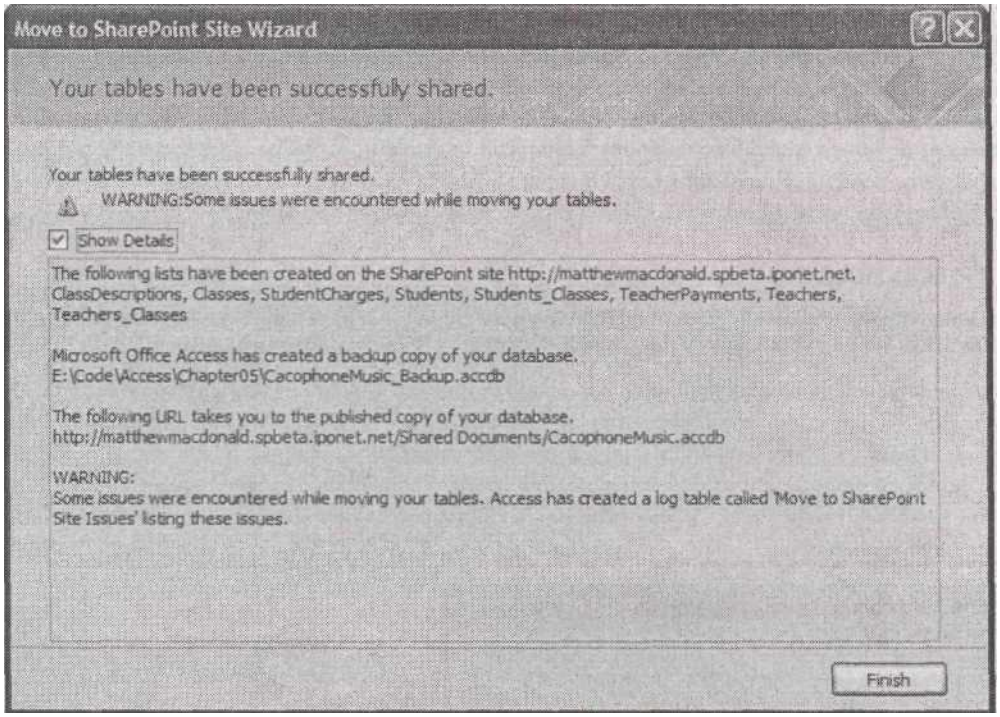


Рис. 21.20. В данном примере программа Access сгенерировала восемь списков, создала резервную копию БД и опубликовала БД в расположении **Shared Documents.** Access также отметила несколько проблем в таблице **Move to SharePoint Site Issues.** Это просто предупреждения об отсутствии поддержки в программе SharePoint ссылочной целостности

8. Щелкните мышью кнопку **Finish** (Готово).

Вы заметите, что ваша БД изменилась. Все таблицы изменены на связанные таблицы, подключенные к соответствующим спискам SharePoint. (Визуальный признак - желтая пиктограмма таблицы со стрелкой, появляющаяся рядом с каждой таблицей в области переходов программы Access.)

Малоизвестная или недооцененная возможность.

Параметры списков SharePoint

Программа Access предоставляет легкий способ доступа к нескольким часто используемым параметрам SharePoint. Для их просмотра щелкните правой кнопкой мыши связанную таблицу и выберите подменю **SharePoint List Options** (Параметры списка SharePoint). На экране появятся команды, позволяющие изменить таблицу, настроить ее разрешения или задать оповещения, уведомляющие об изменении конкретных данных. При выборе одного из этих параметров программа Access запускает ваш Web-обозреватель и переводит его на соответствующую страницу SharePoint.

21.3.5. *Редактирование данных SharePoint в Access*

Как только вы фиксируете изменение в связанной таблице (например, выполнив редактирование и перейдя к другой строке), программа Access отправляет новые значения на сервер SharePoint. Единственное, что вы не можете менять, - структура таблицы. Для этого придется использовать программу SharePoint. (Один из быстрых способов перехода к нужной Web-странице - щелчок таблицы правой кнопкой мыши в программе Access и выбор последовательности команд **SharePoint List Options → Modify Columns and Settings** (Параметры списка SharePoint → Изменить столбцы и параметры списка SharePoint).)

Подсказка

Для отображения самой свежей информации в вашей таблице в любое время можно выбрать на ленте **Home → Records -+ Refresh All** (Главная → Записи → Обновить все).

Если не повезет, вы можете начать изменять запись, когда ее редактирует кто-то другой. Если вы закончите свою корректировку первым, то ничего не узнаете о возникшем конфликте. (Отменено будет изменение другого пользователя.) Если вы окажетесь проигравшей стороной и завершите корректировку после того, как кто-то уже изменил запись, то получите показанное на рис. 21.21 сообщение, которое позволяет решить, что делать.

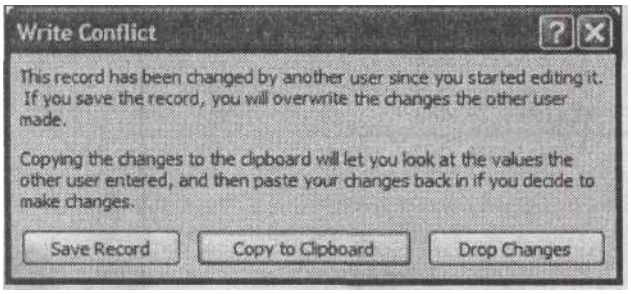


Рис. 21.21. Это сообщение извещает вас о том, что кто-то уже изменил запись, с которой вы работаете. Можно щелкнуть мышью кнопку **Save Record** и вслепую записать собственные изменения поверх изменений другого пользователя (что всегда рискованный шаг) или кнопку **Drop Changes** и отменить свою корректировку. Но самый интересный вариант - кнопка **Copy to Clipboard,** которая позволяет скопировать ваши значения в буфер обмена ОС Windows и отменяет вашу корректировку. Затем вы сможете просмотреть текущую запись и вставить в нее часть или все ваши изменения

Подсказка

Когда используется кнопка **Copy to Clipboard** (Копировать в буфер), программа Access копи-рует всю строку. Если вы хотите вставить просто пару значений, можно вставить всю запись в другую программу (например, текстовый редактор) и затем скопировать только нужные вам значения.

21.3.6. *Внесение изменений в автономном режиме*

Единственный недостаток применения связанных таблиц - необходимость подключения к серверу SharePoint. Если нужно внести изменения в нестандартной ситуации - например, когда применяется портативный компьютер дома или гроза нарушила работу вашей сети - удача отвернется от вас.

Или нет? Оказывается, программа SharePoint предоставляет связанным таблицам дополнительную поддержку, позволяя использовать списки SharePoint без подключения. Благодаря этому средству можно использовать связанные таблицы автономно, внести изменения и затем применить их позже, когда подключитесь к серверу в следующий раз.

Для работы с БД в автономном режиме выберите на ленте **External Data → SharePoint Lists → Work Offline** (Внешние данные → Списки SharePoint → Автономный режим). Теперь вы отключились от сервера и не увидите изменений других пользователей, а они не увидят ваших.

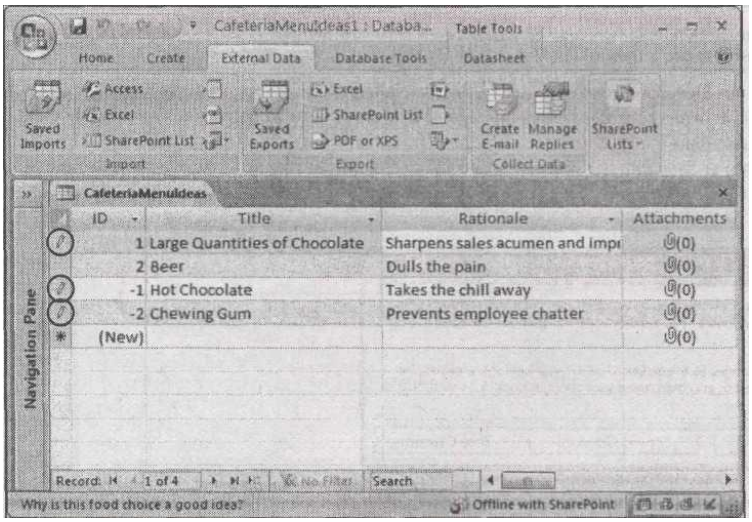


Рис. 21.22. В данном примере две новые записи добавлены и одна запись отредактирована. Значение поля ID во всех новых записях временно задается отрицательным числом, поскольку программа Access не может получить эти данные, пока не подключится к серверу SharePoint и не попросит его сгенерировать новое значение. Пиктограммы редактирования (обведены) указывают на то, что три записи все еще в режиме редактирования

Когда ваша таблица используется в автономном режиме, рядом с каждой измененной вами записью остается пиктограмма редактирования (рис. 21.22). Это означает, что таблица остается в режиме редактирования, поскольку изменения еще должны быть перенесены на сервер SharePoint.

В автономном режиме можно возобновить подключение к серверу SharePoint тремя способами.

- Выбрать на ленте **External Data → Lists → Synchronize** (Внешние данные → Списки SharePoint → Синхронизировать) для получения самых свежих данных и фиксации сделанных вами изменений. Когда процесс завершится, вы снова вернетесь в автономный режим.
- Выбрать на ленте **External Data → SharePoint Lists → Work Online** (Внешние данные → Списки SharePoint → Связать списки) для синхронизации вашей БД и затем переключения в оперативный режим.
- Выбрать на ленте **External Data → SharePoint Lists → Discard Changes** (Внешние данные → Списки SharePoint → Отменить изменения) для отказа от ваших изменений. В этом 'случае у вас есть два варианта: применить команду **Discard All Changes** (Отменить все изменения) для того, чтобы просто отбросить все, что вы сделали, или использовать команду **Discard All Changes and Refresh** (Отменить все изменения и обновить) для того, чтобы отменить все изменения, а затем получить самые свежие данные с сервера SharePoint. В любом случае вы останетесь в автономном режиме.

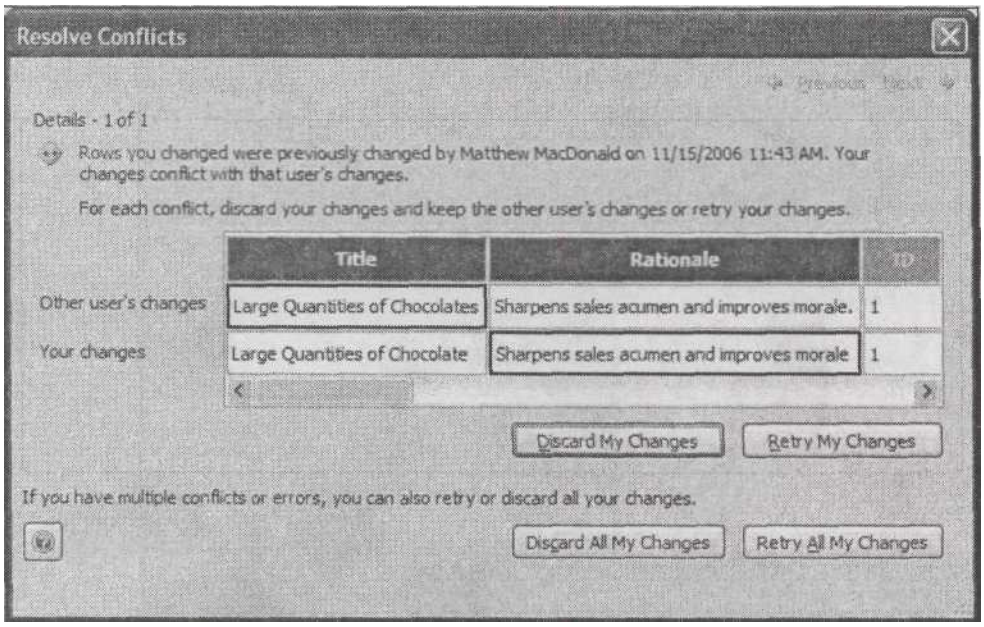


Рис. 21.23. В данном примере программа Access обнаружила запись, которая уже корректировалась кем-то. У вас есть на выбор два варианта: **Discard My Changes** для сохранения записи в текущем состоянии или **Retry My Changes** для внесения своей корректировки, даже если она запишется поверх самых последних изменений

Когда вы повторно подключитесь к серверу SharePoint, программа Access попытается внести по очереди все сделанные вами во время работы в автономном режиме изменения. Этот процесс идет нормально до тех пор, пока не встретится запись, которая изменена кем-то еще. В этом случае возникает проблема, т. к. программа Access не знает, должна ли она внести ваши изменения и уничтожить работу другого пользователя или оставить запись в потенциально противоречивом состоянии.

Программа Access обрабатывает этот конфликт лучше, чем обычные конфликты оперативного режима (см. рис. 21.21). Вместо того чтобы просто сообщить о возникшей проблеме, она

отображает реальные конфликтующие значения. Например, на рис. 21.23 видно, что текущий пользователь удалил точку в конце значения поля **Rationale** (обоснование) в то время, как другой пользователь добавил букву "s" в слово "Chocolates." В результате поле осталось с двумя конфликтующими значениями.

Примечание

Средство внесения изменений в автономном режиме работает лучше всего, если применяется только в течение коротких периодов времени, поскольку есть риск столкновения ваших изменений с корректировками других пользователей. Программа Access обнаружит эти проблемы во время следующей синхронизации вашей БД (см. рис. 21.23), а решить их не всегда легко. Если возможно, не вносите изменения при отсутствии подключения к серверу.

22. Приложение Настройка Панели быстрого доступа

В предыдущих версиях программа Access разрешала своим фанатам перемещать панели инструментов, реорганизовывать кнопки и даже нарушать порядок элементов главного меню. Бесшабашные пользователи могли преобразовать Access так основательно, что никто больше не мог пользоваться их компьютерами и рекомендации в книгах, подобных данной, становились бесполезными.

Версия Access 2007 стала строже к настройке. Если вы не хотите пачкать руки серьезным программированием на специальном языке, лента - запретная зона. Программа Access разрешает настраивать одну крошечную часть полезной площади экрана, **Панель быстрого доступа** (Quick Access toolbar).

Это ограничение может показаться существенным, но на самом деле это разумный компромисс. Те, кто любит изменять и улучшать свои рабочие места (вы знаете, к какому типу относитесь сами), получают как дополнение все необходимые времясберегающие средства ускорения. Остальные могут расслабиться. Не важно, на каком компьютере вы работаете, лента всегда на месте со своей удобной унификацией и тщательно организованными вкладками.

Примечание

Вы можете добавить собственную вкладку на ленту. Но это определенно нелегкий процесс - в действительности он под силу опытным программистам, кого не смущает серьезный программный код. Стандарт для настройки ленты называется RibbonX и требует смеси из XML-языка и языка программирования ядра системы, например C#. Технари могут начать с введения, представленного на Web-странице **http://msdn2.microsoft.com/en-us/library/ms406046.aspx.**

22.1. *Панель быстрого доступа*

Вы уже видели **Панель быстрого доступа** (Quick Access toolbar), известную знатокам Access как QAT. Эта панель крошечного размера расположена над лентой. На ней есть только пиктограммы, но можно переместить указатель мыши поверх пиктограммы, если хотите увидеть краткое описание назначения кнопки.

Когда вы начинаете знакомиться с программой Access, **Панель быстрого доступа** - единственное место с кнопками для быстрого сохранения текущего объекта БД и отмены или повторения последней команды. Программа Access предоставляет полный контроль над этой областью экрана, включая возможность добавить новые кнопки. Самый быстрый путь для вставки кнопок - щелчок кнопкой мыши по направленной вниз стрелке, показанной на рис. П.1.

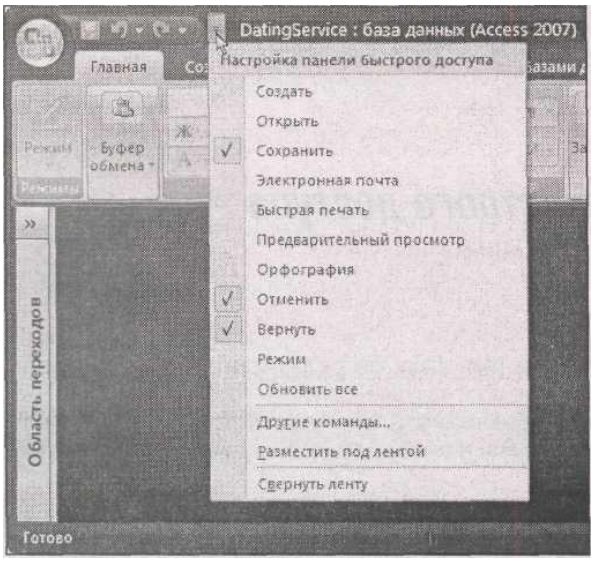


Рис. П.1. Когда вы щелкаете кнопкой мыши направленную вниз стрелку, расположенную на **Панели быстрого доступа,** программа Access отображает список часто используемых

команд, которые можно добавить простым щелчком кнопки мыши. К ним относятся команды создания новой БД, открытия существующей БД, отправки текущего объекта БД (того, который выделен в области переходов) на принтер без каких-либо дополнительных вопросов, отправки данных текущего объекта БД по электронной почте и запуска проверки правописания. Но чтобы увидеть все возможности, следует выбрать команду **Другие команды...**

Примечание

Если вам не нравится местоположение **Панели быстрого доступа,** программа Access предлагает еще один вариант. Щелкните кнопкой мыши направленную вниз стрелку и выберите команду **Разместить под лентой** (Show Below the Ribbon) для перемещения панели под ленту и сокращения пути пробега вашей мыши.

Могут быть две причины для добавления кнопок на **Панель быстрого доступа.**

■ Облегчение доступа к команде, которая часто используется. Если она расположена на **Панели быстрого доступа,** не нужно запоминать сочетание клавиш или переключаться на другую вкладку на ленте.

■ Для получения доступа к команде, которой нет на ленте. У программы Access есть небольшой набор непопулярных команд, которые можно применять, но они не хранятся на ленте. Многие из этих команд - наследие прежних версий Access. Если у вас есть давно потерянное любимое средство Access, которое пропущено, оно может стать доступным благодаря дополнительным кнопкам на **Панели быстрого доступа. (В** следующем разделе описывается, как познакомиться с полным набором доступных кнопок.)

Любители клавиатуры могут также с легкостью запускать команды, представленные на **Панели быстрого доступа,** благодаря средству Access "Клавиатурные подсказки" (KeyTips) *(см. разд. "Использование ленты с помощью клавиатуры" во введении).* Когда нажимается клавиша <Alt>, программа Access отображает числа, наложенные поверх каждой команды на **Панели быстрого доступа** (начиная с 1 и далее по возрастанию). Затем можно нажать цифру для запуска нужной команды. Таким образом, на **Панели быстрого доступа,** показанной на рис. П.1, комбинация клавиш <Alt>+<1> сохраняет открытый в данный момент объект БД, <Alt>+<2> запускает команду **Отменить** (Undo) и т. д.

Подсказка

Если вы хотите добавить команду, повторяющую что-либо, уже представленное на ленте, есть средство ускорения: найдите команду на ленте, щелкните ее правой кнопкой мыши и затем выберите строку **Добавить на панель быстрого доступа** (Add to Quick Access Toolbar).

22.2. *Добавление кнопок*

Для добавления кнопки на **Панель быстрого доступа** выполните следующие действия.

1. Щелкните кнопкой мыши направленную вниз стрелку на **Панели быстрого доступа** и затем выберите команду **Другие команды...**

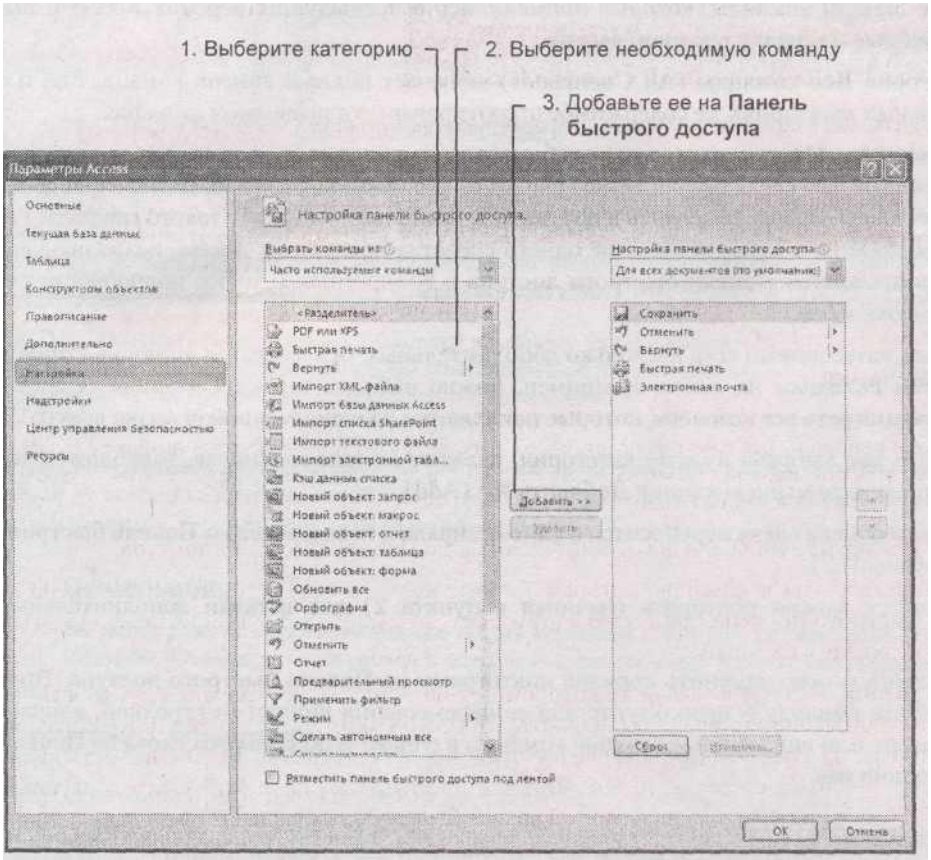


Рис. П.2. У раздела **Настройка** в окне **Параметры Access** есть две области. Список слева позволяет выбрать команду, которую хотите добавить. Список справа отображает команды, которые в данный момент находятся на **Панели быстрого доступа**

Откроется диалоговое окно **Параметры Access,** и вы будете направлены в раздел **Настройка** (Customize) (рис. П.2).

2. Выберите категорию из списка **Выбрать команды из** (Choose commands from).

Библиотека команд, которые можно добавить на **Панель быстрого доступа,** огромна. Для того чтобы легче было найти то, что нужно, в программе Access все команды разделены на категории. Многие категории перекрываются - Access просто предоставляет их для облегчения поиска нужной команды. Далее перечислены первые предлагаемые варианты.

- Категория **Часто используемые команды** (Popular Commands) предлагает краткий список команд, любимых профессионалами Access. Если вы пытаетесь получить доступ к популярному инструменту, возможно, вы найдете его здесь.
- □ Категория **Команды не на ленте** (Commands Not in the Ribbon) содержит все оставшиеся команды, которые корпорация Microsoft не считает достаточно полезными для размещения их на ленте. В этот список входят некоторые команды, замененные или частично дублирующиеся другими командами, команды, включенные в другие диалоговые окна, и команды, которые применялись в предыдущих версиях Access и выброшенные на свалку в данной версии.
- Категория **Все команды** (All Commands) включает полный список команд. Как и в остальных категориях, ее содержимое отсортировано в алфавитном порядке.
- В категории **Макросы** (Macros) отображаются все макросы из открытой в данный момент БД. Но тут есть проблема: если добавить макрокоманду на **Панель быстрого доступа,** она не будет работать в других БД, поскольку у них нет такого макроса. Решение кроется в применении еще одного средства программы Access, позволяющего настроить состав **Панели быстрого доступа** в конкретных БД. Все подробности *см. в следующем разделе.*

Под этими категориями есть несколько дополнительных, относящихся к кнопке **Office** и различным вкладкам на ленте. Например, можно выбрать строку **Создание** (Create), чтобы просмотреть все команды, которые появляются на одноименной вкладке ленты.

3. После того как выбрана нужная категория, укажите команду в списке, расположенном ниже, и щелкните мышью кнопку **Добавить** » (Add).

Команда из списка слева переносится в список справа и помещается на **Панель быстрого**

доступа (рис. П.3).

4. Этот процесс можно повторять (начиная с пункта 2) для вставки дополнительных команд.

При желании можно изменить порядок пиктограмм на **Панели быстрого доступа.** Просто выберите команду и используйте для ее перемещения кнопки со стрелкой, направленной вверх или вниз. Самые верхние команды в списке отображаются слева на **Панели быстрого доступа.**

Подсказка

Если вы чудовищно перенастроили **Панель быстрого доступа** и хотите вернуться к более спокойной жизни, просто щелкните мышью кнопку **Сброс** (Reset).

5. Когда закончите, щелкните мышью кнопку ОК для возврата в программу Access с переделанной **Панелью быстрого доступа.**

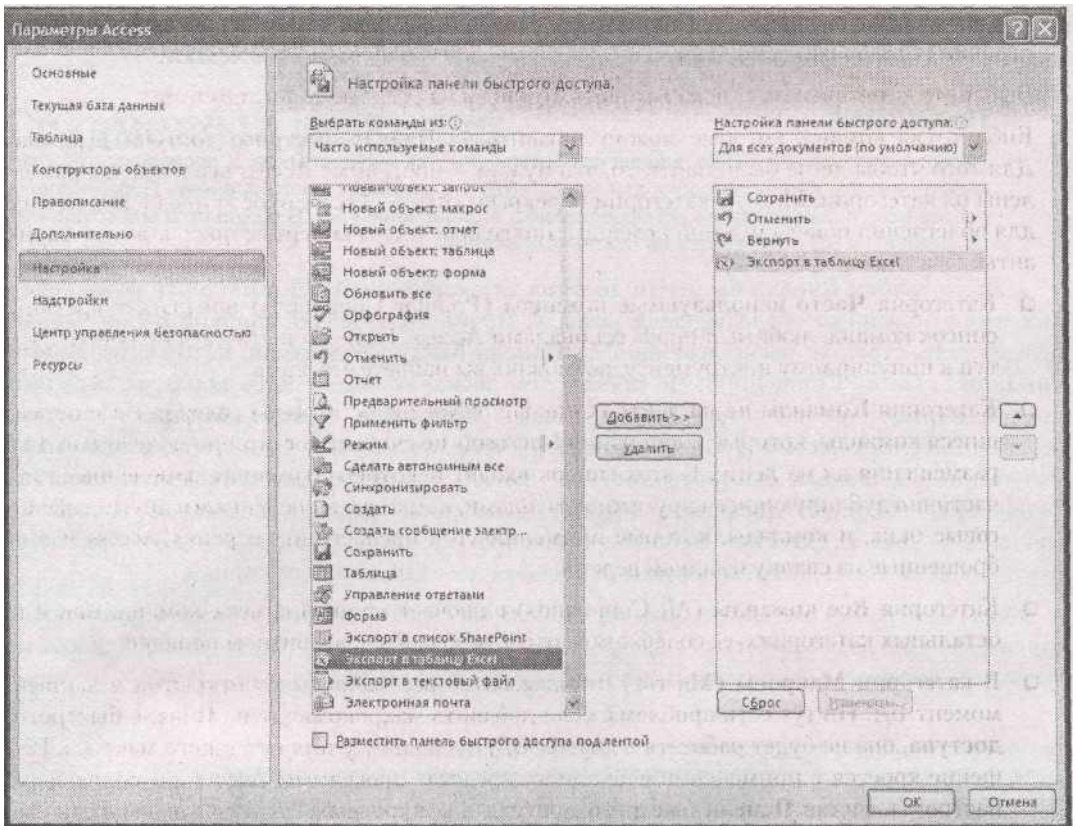


Рис. П.3. В данном примере команда **Экспорт в таблицу Excel** добавляется на **Панель быстрого доступа,** так что вы сможете быстро экспортировать содержимое текущей таблицы без отыскивания ее на ленте

Вставка на **Панель быстрого доступа** - недолговечное вложение. Для того чтобы избавиться от команды, которая вам больше не нужна, щелкните ее правой кнопкой мыши и выберите команду **Удалить с панели быстрого доступа** (Remove from Quick Access Toolbar).

Примечание

Вы могли заметить соблазнительную кнопку **Изменить...** (Modify), которая позволяет изменить название команды и пиктограмму. К сожалению, она работает только с макрокомандами.

22.3. *Настройка конкретных БД*

У вас есть кнопка или две, которыми вы пользуетесь постоянно, но только в конкретной БД? В этом случае, возможно, нет смысла настраивать **Панель быстрого доступа** стандартным образом. Если вы сделаете это, то получите дополнительные кнопки на панели для всех БД, включая те, в которых эти кнопки бесполезны.

У программы Access есть замечательное средство, способное помочь в такой ситуации.

Панель быстрого доступа можно настроить для отдельной БД. В этом случае, когда открывается такая БД, нужные вам кнопки отображаются на **Панели быстрого доступа.** Когда БД закрывается (или открывается другая БД в отдельном окне), кнопки исчезают.

Примечание

Настройка отдельных БД не лишена как достоинств, так и недостатков. К последним относится необходимость такой настройки для каждой БД, что отнимает много времени. Достоинство - запись всех ваших настроек непосредственно в файл вашей БД. В результате они сохраняются, даже если БД открывается на чужом компьютере.

Для настройки **Панели быстрого доступа** для конкретной БД выполните те же действия, что и описанные в предыдущем разделе. Начните со щелчка кнопкой мыши направленной вниз стрелки, расположенной на **Панели быстрого доступа,** и выберите **Другие команды...** (More Commands). Но прежде чем добавлять какие-либо команды, выберите другой вариант в раскрывающемся списке **Настройка панели быстрого доступа** (Customize Quick Access Toolbar), который отображается прямо над списком команд, включенных в **Панель быстрого доступа.** Вместо варианта **Для всех документов (по умолчанию)** (For all documents (default)) выберите вариант с именем вашей БД (например, **Для C:\MyFiles\SecretSanta.accdb).** В этом случае вначале отображается пустой список команд. Затем обычным способом добавляете в него кнопки.

Когда программа Access отображает **Панель быстрого доступа,** она объединяет стандартные кнопки (в соответствии с настройкой, описанной в предыдущем разделе) с любыми кнопками, которые вы определили для текущей БД. На рис. П.4 показан пример.

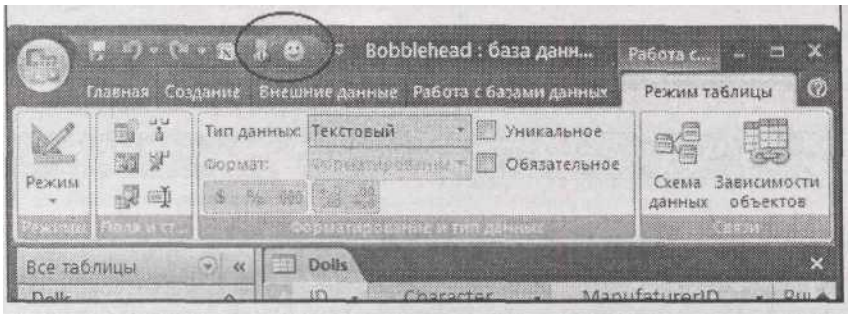


Рис. П.4. Относящиеся к конкретной БД кнопки (обведены) всегда выводятся после стандартных кнопок и немного отличаются от них внешне (более темный цвет фона)

Настройка панели быстрого доступа для конкретной БД - полезный прием. Он очень удобен при работе с макросами, поскольку позволяет создать одну БД, содержащую набор полезных макросов, и удобные кнопки для их запуска. Например, можно создать макросы, выводящие на экран определенные формы вашей БД, а затем добавить их на **Панель быстрого доступа.** В этом случае пользователь сможет быстро перемещаться в вашей БД без применения области переходов.

23. Предметный указатель

Оглавление (по книге)