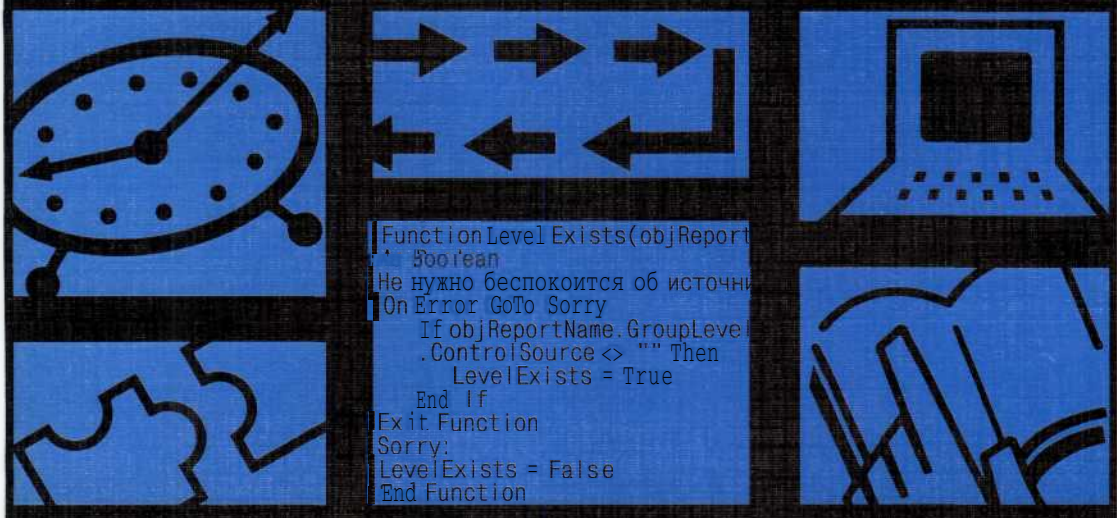
Программирование Access 2002

В ПРИМЕРАХ



que

КУДИЦ-ОБРАЗ

Боб Вилларриал

Я посвящаю эту книгу моим родителям, Морей и Лью Виллариал, чья поддержка и содействие сопровождали меня на протяжении всей моей жизни.

Bob Villareal

Access 2002 Programming

BY EXAMPLE

que

Боб Вилльариал

Программирование Access 2002

В ПРИМЕРАХ

Перевод с английского

КУДИЦ-ОБРАЗ
Москва • 2003

Вилльариал Б.
Программирование Access 2002 **в примерах: Пер. с англ. - М.: КУДИЦ-ОБРАЗ, 2003. - 496 с.**

Эта книга позволяет читателю не просто быстро начать писать программы с помощью интегрированной среды разработки MS Access 2002 но и правильно разрабатывать свои приложения баз данных, используя всю мощь MS Access 2002.

Она научит читателя фундаментальным основам, таким как нормализация баз данных, создание запросов, работа с объектами и оптимизация, а также создание пользовательских форм и отчетов с помощью программных средств. Эти умения позволят читателю добиться большей эффективности, производительности и профессионального владения средствами Access VBA. Главы книги включают не только полезные советы, примечания, предупреждения и ссылки, но также содержат резюме, которые дают место каждой главы в общей перспективе.

ISBN 0-7897-2594-0
ISBN 5-93378-059-6

Вилльариал Боб
Программирование Access 2002 **в примерах.**

Учебно-справочное издание

Корректор М. Матекин
Перевод с англ. П.А. Казанцев, А.А. Ковытин
Научный редактор к. ф.-м. н. Ю. Голубь

Лицензия ЛР № 071806 от 02.03.99. НОУ «ОЦ КУДИЦ-ОБРАЗ»
119034, Москва, Гагаринский пер., д. 21, стр. 1. Тел.: 333-82-11, ok@kudits.ru

Подписано в печать 15.11.2002.
Формат 70x100/16. Бум. офсетная. Печать офс.
Усл. печ. л. 40. Тираж 3000. Заказ Зак. 7-11

Отпечатано с готовых диапозитивов в ООО
"Типография ИПО профсоюзов Профиздат".
109044, Москва, Крутицкий вал, 18.

ISBN 0-7897-2594-0 © QUE, 2002
ISBN 5-93378-059-6 © НОУ «ОЦ КУДИЦ-ОБРАЗ», 2003

Authorized translation from the English language edition, entitled ACCESS 2002 PROGRAMMING BY EXAMPLE, 1st Edition by VILLAREAL, BOB, published by Pearson Education, Inc, publishing as Que, Copyright © 2002 by Que Publishing.

Авторизованный перевод с англоязычного издания, озаглавленного ACCESS 2002 PROGRAMMING BY EXAMPLE, 1st Edition by VILLAREAL, BOB, опубликованного Pearson Education, Inc. под издательской маркой Que, Copyright © 2002 by Que Publishing.

Программирование на Access 2002 в примерах

Торговые знаки

Предупреждение и отказ от ответственности

Об авторе

Боб Вилльариал работает с микрокомпьютерами с 1981 года и имеет почти десятилетний опыт программирования на VBA как в Access, так и в Excel. Боб написал множество приложений по отслеживанию и управлению и разрабатывает базы данных для крупной страховой компании более 10 лет.

Боб также занимается свободным программированием и преподавательской деятельностью. Он является сотрудником ежемесячного Web-журнала, *«Inside Microsoft Access», а* также преподавателем Microsoft Access в Общественном колледже в Талсе, штат Оклахома. Боб получил научную степень бакалавра по системам связи в университете в Талсе.

Выражение благодарности

В первую очередь хотелось бы поблагодарить Лоретту Йэтс, редактора по приобретениям для издательства Que, которая оказала огромную мотивацию и содействие при написании этой книги. Другим человеком, оказавшим большое содействие является Грег Перри, преуспевающий пирсоновский автор, который помог мне хорошо сориентироваться при написании. Я бы не сделал этого без Сью Хоббс, редактора по разработке, чей талант и поддержка были очень полезны на всем пути от написания до издания. Ларри Стил, технический редактор, в меру критиковал меня по делу и в меру поощрял. Его вклад определенно помог спланировать эту книгу. И в заключение я хотел бы поблагодарить моих друзей и семью, которые, казалось, были почти так же, как и я, заинтересованы в издании этой книги.

Скажите нам, что вы думаете!

Как читатель этой книги, вы наш наиболее важный критик и комментатор. Мы ценим ваше мнение и хотим знать, что мы делаем правильно, что мы могли бы сделать лучше, где бы вам хотелось, чтобы мы публиковались, и любые другие слова мудрости, которые вы бы хотели направить в наш адрес.

Как издатель в корпорации Que, я буду рад принять ваши комментарии. Вы можете отправить их по факсу, по электронной почте или просто написать письмо мне, чтобы я знал, что вам понравилось или не понравилось в этой книге, а также, что мы можем сделать для того, чтобы наши книги были еще лучше.

Пожалуйста, учтите, что я не могу вам помочь с техническими проблемами, относящимися к теме этой книги, и что из-за огромного числа писем, получаемых мною, я не смогу ответить на все послания.

В письме, пожалуйста, укажите название этой книги и ее автора, а также ваше имя и номер телефона или факса. Я внимательно просмотрю ваши комментарии и поделюсь ими с автором и редакторами, которые работали над этой книгой.

Факс: 317-581-4666

E-mail: feedback@quepublishing.com

Почтовый адрес: David Culverwell
Que Publishing
201 West 103rd Street
Indianapolis, IN 46290 USA

Введение

Хотя Microsoft Access и является мощной платформой для разработки баз данных, все же требуется значительное время для того, чтобы полностью охватить ее возможности и продемонстрировать ее силу. Как и при любом другом изучении, вам необходимо сначала понять основы разработки баз данных, чтобы заложить прочный фундамент, на котором можно строить.

Эта книга не претендует на полноту. Даже более «продвинутые» книги о разработке в Microsoft Access не могли бы претендовать на то, что они описывают все, что нужно знать об Access. Что действительно является целью этой книги, так это ввести вас в курс дела процесса разработки баз данных. Но как же это осуществить? Мы собираемся дать вам ясные, легкие для понимания примеры, и вы узнаете многое об Access в этом процессе.

Почему «в примерах»?

Есть известная поговорка: «Лучше один раз увидеть, чем сто раз услышать». Ну вот, например, разговоры о теории гольфа. Вы можете обсуждать, как нужно держать клюшку, стойку и как наносить удары. Все это необходимо. Но вот когда вы видите, как кто-то подходит к мячу и загоняет его в ямку, то многое можно понять об основах удара из этого наблюдения.

Большинство людей учатся, воспринимая визуально. Конечно, у вас будет множество изображений диалоговых окошек и других графических объектов, чтобы помочь вам воспринять визуально применяемые методы, но у вас также будут примеры «настоящих» ситуаций, которые часто встречаются в бизнесе или дома. Другие книги о компьютерах тоже претендуют на то, что в них приведены настоящие примеры. Но большинство наших примеров будут непосредственно из мира бизнеса - мира, в котором компьютеры необходимы каждый день. Другими словами, вы не найдете здесь много «теории», за исключением случаев, когда мы будем говорить об основах.

Это означает, что после того, как изложен новый принцип, у вас сразу же будет практический пример того, как применить этот принцип. Используя метод обучения на примерах, мы стараемся начинать с простого, а затем постепенно переходим к более сложному. Таким образом, вы никогда не почувствуете переполнения знаниями. Вы изучите каждую тему чуть ли не сразу. Каждая глава является сама по себе законченной; в то же время с каждым новым изученным методом вы будете надстраивать свое здание знаний, отталкиваясь от базовых сведений.

Цели этой книги

То, чего я хотел достичь написанием этой книги, объясняется очень просто. Я хотел представить книгу, которая легко воспринимается и которую легко применить на практике. Вот здесь-то и возникают трудности. Я хорошо осведомлен о том, что существуют и другие книги по Access, но многие из них трудно воспринимаются, а применить их на практике еще труднее. Другие содержат

множество методов, но не имеют связнности изложения, которая помогает вам сложить все части воедино. Эта книга поможет вам ответить на вопрос «Зачем?», стоящий за этими методами, так, что вы поймете, какой метод лучшим образом подходит для какой ситуации и какое отношение различные методы имеют друг к другу. Часто используются сравнения, чтобы помочь вам понять различия между разными терминами Access.

Я также надеюсь дать вам ряд полезных инструментов. Начинающий программист может встретить множество препятствий и ловушек. Опытный программист скажет: «Если бы я знал тогда то, что я знаю сейчас, я бы сделал куда больше». Я хочу дать вам эти инструменты сейчас, и вам не нужно будет бродить по Web-сайтам и меню помощи в поисках ответов. Все становится очень логичным, когда вы знаете, что вы делаете. Если даже на первый взгляд кажется, что один из инструментов бесполезен или неважен, просто подождите. Рано или поздно вы натолкнетесь на ситуацию, в которой вам потребуется точно такой же метод, описанный в этой книге.

Что, мы предполагаем, вы знаете

Предполагается, что вы начинающий разработчик. Было бы полезно иметь кое-какие основные знания по Microsoft Access и свободно обращаться с Windows, но даже и это не является полностью необходимым, потому что книга рассказывает и о некоторых основах Access. Также предполагается, что вы серьезно настроены на изучение большей части Microsoft Access во всех отношениях. Эта книга позволит подняться с уровня начинающего до уровня среднего (а в некоторых случаях и «продвинутого») пользователя Access.

Что, мы предполагаем, вы не знаете

Разумеется от вас не требуется быть экспертом Microsoft Access. Никаких знаний о макросах, Visual Basic for Applications (VBA) или программировании в целом не требуется. Мы предполагаем, что вы совсем еще новичок. Поэтому любые знания любого языка программирования, которые, может, у вас и имеются, будут только плюсом, но, естественно, не необходимы.

Исходные файлы и как их получить

Чтобы воспользоваться множеством примеров, представленных в этой книге, сначала создайте папку AccessByExample в разделе C вашего жесткого диска. Затем, зайдите на www.QuePublishing.com и скачайте файл AccExamp.exe в папку AccessByExample. В скачиваемый файл включены несколько заархивированных баз данных, поэтому это может занять кое-какое время, в зависимости от используемого вами типа связи с Интернетом. Эти базы данных и файлы используются на протяжении всей книги в качестве практического применения обучения на примерах.

Условные обозначения, используемые в книге

ЗАМЕЧАНИЯ - поясняют или дополняют идею каждой главы.

ПОДСКАЗКИ - дают короткое пояснение или решение, которое либо относится к вопросу, либо предлагает альтернативу общепринятому методу. Подсказки экономят время и удобны.

ПРЕДОСТЕРЕЖЕНИЯ - предупреждают о потенциальных проблемах или ошибках, которые могут возникнуть по мере выполнения множества практических упражнений или примеров на протяжении всей книги.

Соглашения о названиях

Некоторые используют соглашение о названиях, принятые для Reddick VBA для названия объектов Access. Например, все таблицы имеют приставку tbl, тогда как все запросы имеют приставку qry. Так, все таблицы Customer будут называться tblCustomer. Это удобно и вот почему. Access не позволяет иметь таблицы и запросы с одним и тем же именем. К тому же сразу можно сказать, что за объект перед вами. Разумеется, вы можете использовать эти названия по своему усмотрению. Это соглашение часто используется в этой книге, а особенно при описании методов.

Другое распространенное соглашение о названиях заключается в использовании прописных букв для отделения слов и в то же время для исключения разрывов в названиях объектов. Например, вместо использования Last Name для названия поля следует использовать LastName. Это помогает избежать необходимости использования скобок в критериях запроса, так же как и в VBA. Вместо написания [Customer]![Last Name] в качестве параметра выражения, просто пишите Customer!LastName - и скобки будут вставлены автоматически. Access допускает пробелы в полях, таблицах, формах, отчетах и названиях запросов. Тем не менее, за исключением редких случаев, вы будете использовать это соглашение.

Часть I
Таблицы и запросы

1

Планирование и проектирование вашей базы данных

Представляете ли вы себе строительство дома без чертежа или сборку автомобиля без схемы? Здравый смысл подсказывает нам, что такое невозможно. Более того, это только говорит нам, что необходимо потратить время на проектирование вашей базы данных перед тем, как создавать ее. Это сэкономит время на продолжительном этапе разработок, потому что вам удастся избежать неприятностей, связанных с исправлением ошибок, свойственных плохо разработанным приложениям. Эта глава посвящена тому, как применять принципы базы данных в ваших разработках. В частности, вы узнаете:

- о получении данных из внешних источников;
- почему важны реляционные таблицы;
- о дизайне базы данных и нормализации;
- почему следует избегать повторяющихся данных;
- об основахнормализации;
- о первой, второй и третьей нормальной форме;
- как сделать так, чтобы Access автоматически выполнял нормировку для вас.

Продумайте ваши данные перед тем, как создавать вашу базу данных

В этой книге рассказано о том, как получать данные из внешних источников, что относится к более широкому кругу, чем просто покупатели, клиенты, глобальная сеть, электронная почта, внешняя универсальная ЭВМ или источники в Интернете. Внешние источники могут быть и внутри вашей компании или даже внутри вашего отделения. Термин «внешний» обозначает все, что находится вне Microsoft Access. Что вы делаете с данными, которые вы лично не проверили на согласованность, точность и полноту? Даже если вы сгенерировали данные сами (скажем, с помощью такой программы, как Microsoft Excel), как вы узнаете о том, что они согласованы с правилами «нормализации баз данных» (обсуждаемых в разделе «Избежание повторения данных» позже в этой главе) для создания баз данных, во время импортирования их в Access?

А конкретней, всегда ли ZIP-коды полны и точны? Всегда ли запятая отделяет город от штата? Нужно ли разбирать (отделять) данные по отдельным полям?

Были ли данные проверены? Например, все ли номера социального обеспечения имеют одинаковое число знаков? Microsoft Access обеспечивает проверку данных, введенных в базу данных Access, а что же насчет данных, введенных в другие пакеты программного обеспечения?

Спокойствие: у Microsoft Access есть множество инструментов для того, чтобы помочь вам проверить данные даже «в соответствии с обстоятельствами». В этой книге описаны такие методы для подготовки данных, которые стали бы частью совместимого и хорошо спланированного приложения. Кроме инструментов для анализа и очистки данных при их подготовке к использованию в приложении, существуют также принципы разработки, которые должны быть четко уяснены и применены для того, чтобы убедиться в связности и полноте данных. Так что перед тем, как заняться «внешним», давайте-ка заглянем внутрь.

Зачем нужны реляционные таблицы?

Давайте поговорим о разработке базы данных. Почему же так важно иметь реляционные таблицы в базе данных? Почему бы не использовать их подобно тому, как мы используем обычную табличную программу[1] - в двумерных таблицах? Разве и так нельзя вести расчеты и добавлять записи?

Все потому, что «настоящие» ситуации устроены совсем не так. Сколько покупателей лишь закажут один товар? Сколько работников будут лишь зачислены на один проект? Сколько студентов будут посещать лишь один курс? Мы живем в трехмерном мире типа «один ко многим». И хотя двумерность сойдет для обычной табличной программы, базы данных все же имеют более жесткие конструкции; эта дополнительная размерность может быть огромным преимуществом.

База данных, построенная на двумерном файле, является двумерной моделью. Что это означает? Если вы представите себе таблицу как электронную таблицу с рядами в виде записей и полями в виде колонок, получится, что вы смотрите как бы на одну сторону этой таблицы, так сказать. Реляционная база данных добавляет третье измерение. Теперь вы можете представить себе ее длину, ширину и глубину. Ряды - это ширина, колонки - это длина, а зависимость - это глубина (см. рис. 1.1).

Years : Table

Region	1997	1998	1999	2000
SW Region	150,000	195,000	253,500	329,550
Central Region	145,000	188,500	245,050	318,565
Southern Region	175,000	227,500	295,750	384,475

Record: 1 of 3

Рис. 1.1. Эта трехмерная электронная таблица показывает область как ширину, колонку с годами как длину и реляционную последовательность данных о годах как глубину

[1] В качестве обычной табличной программы можно рассмотреть программу Excel. В ней можно вести записи и расчеты, но она создана для плоских или двумерных таблиц. - *Научн. ред.*

На первый взгляд все это выглядит как двумерная модель. Фактически это и есть двумерная модель, если вы представите ее в виде строк и столбцов. Но финансовые данные в столбцах трехмерны, если правильно это себе представить, как показано на рис. 1.2.

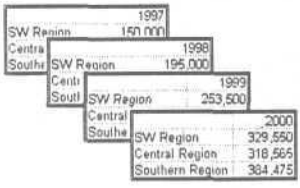


Рис. 1.2. Осмыслить таблицу с данными лучше всего, посмотрев на нее с трехмерной точки зрения

Заголовки столбцов имеют одно общее свойство: все они обозначают годы. В базе данных вам хотелось бы хранить общую информацию в категориях, называемых *полями*. Более подробно это будет обсуждаться позже, но дизайн на рис. 1.1 и 1.2 не является хорошей моделью базы данных.

Например, если бы вы хотели подвести итог по финансам в юго-западной области, вы бы легко могли подставить формулу в клетку справа в обычной табличной программе. Но в базе данных вам требуется вычисляемое поле для всех четырех столбцов. В любом случае подстановка этих итогов в поле обычно не является лучшим способом разработки базы данных. А если вам требуется подытожить все поля с финансовыми данными, то это, хотя и возможно, является наиболее быстрым или эффективным способом задания запроса. Пример на рис. 1.3 - более правильная модель базы данных, хотя все еще с недостатками. Заметьте, что годы размещены в поле с названием Year (Год), а финансовые данные помечены как Sales (Продажи).


Sales : Table

Region	Year	Sales
SW Region	1997	150,000
Central Region	1997	145 ДМ
Southern Region	1997	175,000
SW Region	1998	196,000
Central Region	1998	188,500
Southern Region	1998	227,500
SW Region	1999	253,500
Central Region	1999	245,050
Southern Region	1999	295,750
SW Region	2000	329,550
Central Region	2000	318,565
Southern Region	2000	334,475

Record: 1 of 12

Рис. 1.3. Лучшая, но все же с недостатками трехмерная таблица, показывающая Year и Sales как отдельные поля

Теперь по крайней мере пользователь знает, с какой информацией он имеет дело. Пользователь может сделать запрос по группе и легко найти итог для каждого года или области или комбинации того и другого. Единственная проблема

состоит в том, что в этом примере показана излишняя (повторяющаяся) информация. Наоборот, в каждом поле должны быть различные значения. Единственным способом установления уникального поля в предыдущем случае является объединение первых двух полей в одно. Это более совершенный способ.

Разработка базы данных и нормализация

К этому моменту вы уже должны были бы знать, что база данных - это совокупность информации, относящейся к определенной теме. Но как организовать эту информацию с максимальной эффективностью? Правильная разработка базы данных - вот ключ для решения этой задачи.

Таблица - сердце базы данных. Это то место, на которое прямым или косвенным образом направлена большая часть вашей работы над базой данных. Но то, что у вас имеется несколько таблиц, связанных друг с другом в вашей базе данных, еще не означает, что они готовы для функционирования базы данных.

То, что нужно рассмотреть при разработке базы данных, включает:

- назначение базы данных;
- необходимые таблицы и поля;
- необходимые связи между таблицами;
- указание, как избежать излишних данных;
- указание, как убедиться в согласованности данных.

Когда вы обдумаете эти пункты, вы быстро придете к мысли о том, как полезна нормализация данных, которая есть процесс упрощения разработки базы данных, целью которого является получение максимальной эффективности и согласованности. Одни говорят, что нормализация - это процесс нахождения и удаления излишних данных и аномалий в базе данных, связанных с ними. Другие утверждают, что нормализация есть процесс разбиения таблиц на более мелкие компоненты с целью оптимизации. Некая комбинация этих определений почти что верна. Чтобы убедиться в правильности разработки базы данных и структуры таблицы, необходимо во что бы то ни стало избежать двух изъянов разработки: избыточности и несогласованности.

Если у вас есть таблица с излишним числом полей или с большим количеством излишних (повторяющихся) данных - это сигнал, что требуется нормализация. Разбивая ваши таблицы на более мелкие таблички и обеспечивая связь между схожими табличками, вы выполняете несколько задач, которые включают:

- избежание повторяющихся данных, как внутри самой таблицы, так и внутри связанных(схожих) табличек;
- максимизацию скорости и эффективности вашей базы данных;
- обеспечение того, что одинаковые данные никогда не будут помещены более чем в одно поле;
- экономию дискового пространства путем избежания излишних данных;

- обеспечение согласованности и единообразия данных, будь то входные или выходные данные;
- создание дизайна, который гораздо более удобен, чем таковой с ненормализованными данными, для поддержания и управления;
- создание механизма для поисков по условию в запросах, формах или отчетах.

Теперь давайте рассмотрим эти вопросы на примере.

Избежание повторения данных

Если вы храните данные в более чем одном мете (будь то одна и та же таблица или две и более), вы создаете *избыточность данных.* Почему же избыточность так нежелательна при разработке базы данных? Скажем, у вас имеется поставщик, который едет из Хьюстона в Даллас, и 50 строк с данными, содержащими их адреса; то тогда вам нужно менять все 50 строк. Это то, что называется *аномалией при обновлении.* Чтобы поправить такое положение, если вы удаляете часть, относящуюся к этому поставщику, в нереляционной модели, вы также удаляете поставщика. К счастью, существует лучший подход для достижения оптимального дизайна базы данных.

Здесь мы введем термин *внутренняя избыточность,* который означает избыточность внутри таблицы, тогда как термин *внешняя избыточность* будет означать одинаковую информацию (неключевые поля), которая содержится более чем в одной таблице. Внешняя избыточность свойственна электронным таблицам. Это потому, что не допускается хранение информации в нескольких таблицах, бланках или рабочих журналах.

Таблица на рис. 1.4 показывает примеры избыточности.

Сначала представьте себе таблицу на рис. 1.4 без первого столбца. И что теперь вы с ней будете делать? Когда возраст Джейн изменится, вам потребуется поменять целых три записи, вместо того чтобы поменять лишь одну. Без первого столбца вы ни с чем не сможете связать эту информацию, потому что у вас нет «крючка», или точки, к которой вы прикрепляете информацию; поэтому информация становится фактически бессмысленной. Первый столбец - единственный уникальный столбец в таблице. Но даже с первым столбцом дизайн функционально небезупречен.

Project ID	Name	Age	SSN
1	Jane	50	444-44-4444
2	Jane	50	444-44-4444
3	Jane	50	444-44-4444

Рис. 1.4. В этой таблице с избыточной информацией есть только одно уникальное поле - Project ID

Столбец Project ID обеспечивает «крючок». Это ключевое (уникальное) поле может быть связано с другой таблицей, называющейся Projects, которая может содержать любую информацию о каждом проекте. Все, что вам требуется, чтобы устранить избыточность, - это присвоить Джейн уникальный Employee ID (но-

мер служащего) (ибо Джейн не очень-то уникальное имя) и поместить ее в таблицу Employee (Служащие). Вы даже можете использовать номер ее социального обеспечения, так как вам известно, что он-то уж точно уникален. Затем вы можете связать ее с таблицей Clients (Клиенты), таблицей Prospects (Потенциальные клиенты) или другими таблицами; список можно продолжать и продолжать. Посмотрите на результаты устранения избыточности на рис. 1.5.

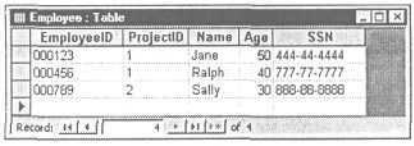


Рис. 1.5. В этой таблице Джейн имеет свой уникальный Employee ID (ID служащего) и помещена в таблицу со своими сослуживцами

Основы нормализации

В 1969 году человек по имени E. F. Codd изобрел модель реляционной базы данных. Для достижения оптимальной структуры он создал *нормальные формы* - последовательность правил, которые вы применяете к вашей базе данных с каждым уровнем нормальной формы, улучшая общий дизайн.

Если бы вы хотели собрать базу данных служащих для отслеживания их клиентов, вам следовало бы устроить поля так, как показано на рис. 1.6.

На рис. 1.6 - ненормализованная таблица. Первая проблема, которая кроется в *полях,* очевидна. А что если вам требуется отследить более чем двух клиентов на каждого служащего? А точнее, что если Стэнли, у которого уже есть два клиента, приобретет еще одного? Добавление еще одного поля - плохой способ, потому что это делает таблицу более сложной и неэффективной.

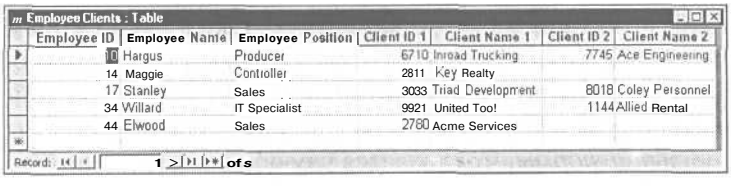


Рис. 1.6. Эта ненормализованная таблица имеет внутреннюю избыточность и повторяющиеся поля

В *первой нормальной форме* не должно быть *повторяющихся групп* (столбцов), поэтому нам придется считаться с тем фактом, что у нас есть два повторяющихся поля. Чтобы изменить ситуацию, у вас не должно быть нескольких полей в одной и той же категории. Это-то и есть поле для большинства случаев - категория. Имея два поля с ID клиентов и два поля с именами клиентов, затруднительно выбирать, сортировать и отслеживать информацию правильно. Как

правило, нет никакой нужды в двух или более полях с одинаковой информацией. Проблемы можно подытожить следующим образом:

- нулевые значения, если у служащего есть только один клиент;
- нельзя отследить более двух клиентов;
- повторяющиеся группы (проблематично отследить);
- одинаковый тип данных в более чем одном поле.

Единственное, что можно сказать хорошего о таблице на рис. 1.6, - это то, что информация о служащих в ней уникальна.

Первая нормальная форма - устранение повторяющихся групп

Как решить проблему повторяющихся групп или другие проблемы, связанные с ненормализованными таблицами, и в то же время гарантироват, что вы сможете отслеживать клиентов таким образом, каким вы намереваетесь это делать? Посмотрите на рис. 1.7, на котором показана та же информация, но только в другом представлении, и с одним лишь исключением.

First Normal Form : Table

Employee ID	Employee Name	Employee Position	Client ID	Client Name	State Code	State
34	Willard	IT Specialist	1144	Allied Rental	MO	Missouri
44	Elwood	Sales	2780	Acme Services	LA	Louisiana
14	Maggie	Controller	2811	Key Realty	TX	Texas
17	Stanley	Sales	3033	Triad Development	OK	Oklahoma
10	Hargus	Producer	6710	Inroad Trucking	AR	Arkansas
10	Hargus	Producer	7745	Ace Engineering	KS	Kansas
17	Stanley	Sales	8018	Coley Personnel	NM	New Mexico
34	Willard	IT Specialist	9921	United Tool	OK	Oklahoma

Record: 9 of 9

Рис. 1.7. Хотя первая нормальная форма решает кое-какие проблемы, все же она несовершенна

На рис. 1.7 добавлен код штата для каждого клиента. С таким добавлением таблица может содержать любое количество клиентов на каждого служащего. Но решение одной проблемы породило другую, которая таится в *записях.* Таблица теперь содержит внутреннюю избыточность информации о служащих. Вы уже достигли *первой нормальной формы,* что является улучшенным дизайном по сравнению с тем, с которого вы начали, но до *оптимального* дизайна еще далеко. Вам все же нужно установить *первичный ключ.*[2] Хотя информация о служащих в ненормализованной таблице и уникальна, все же новый дизайн содержит излишнюю информацию о служащих. С другой стороны, заметьте, что одно поле отслеживает ту же информацию, которую раньше отслеживали два поля. В этом заключается преимущество этой формы перед предыдущей, но все же она не идеальна.

[2] Здесь имеется в виду ключевое поле. Обычно определение «первичный» опускается. - *Науч. ред.*

Первичный ключ

Первичный ключ - это поле (или ряд полей), которое однозначно идентифицирует каждую запись в таблице. Access не допускает повторных значений в поле первичного ключа. Более того, вам обязательно нужно фиксировать значение каждой записи поля первичного ключа для того, чтобы быть уверенным в том, что у вас нет записей без значений. Это и есть то, чего требует реляционная модель. И в этом есть смысл. Разумеется, что вы не хотите дублировать номера заказчиков, если вы желаете идентифицировать заказчика. Access не требует создания первичного ключа в каждой таблице, но это настойчиво рекомендуется, так как если вы сделаете это, то вам же и лучше. Ситуации могут различаться, но это основополагающий принцип.

Из-за того, что поле Employee ID неуникально, вам нужно создать *составное* (два поля действуют как одно) поле, которое выполняет функцию ключа для создания уникальности. В этом случае вы могли бы соединить поля Employee ID и Client ID. Заметьте, что если вы скомбинировали эти два поля в одно, то это создаст уникальное поле. Другими словами, два поля, соединенные вместе, не будут повторять друг друга. А чего бы это им повторять друг друга? Зачем нужно дважды отслеживать служащего с таким же клиентом? Даже если первая нормальная форма позволяет установить первичный ключ, тот факт, что он является составным ключом, - индикатор того, что вторая нормальная форма еще не достигнута.

Если вы разобьете эту таблицу на две, вы все равно не решите проблему избыточности, т. е., пока вы не дойдете до следующего уровня. У вас могут возникнуть ситуации, когда избыточности практически невозможно избежать. Предположите, что в предыдущем примере вместо того, что у служащего есть много клиентов, у одного клиента, использующего множество служащих, также есть другие клиенты. Такой вариант, который в целом возможен, порождает избыточность. Но помните, что мы здесь всего лишь определяем понятие *«норма»*. Вероятно, это потому, что они называются *нормальными* формами.

Аналогичное издание - справочник для разработчиков *«Access 97»,* написанное Паулом Литвиным, Кеном Гетсом и Майком Гильбертом (Access 97 Developer's Handbook, написанный Paul Litwin, Ken Getz и Mike Gilbert) и изданное издательством Sybex, утверждает, что существуют ситуации, когда возможна *денормализация.* Для этих редких случаев, существуют основные правила, которым нужно следовать, чтобы правильно «нарушать правила». В эти правила включают описание каждого шага процесса денормализации и вносят необходимые поправки в применение, дабы избежать отклонений и несовместимости.

Несовместимые данные

Избыточность создает отклонения, а отклонения создают *несовместимость*. Предположим, что у вас есть порядковые номера, для которых заказчиков нет. Это случается, когда таблица с заказчиками не связана[3], а число заказчиков изменилось или информация о заказчике удалена из таблицы заказов. Эти заказы, становятся как бы «потерянными данными», которые «живут» где-то сами по себе, занимая место, и не являются связанными. Это потому что заказ *зависим* от заказчика.

Другой тип несовместимости, упомянутый выше, получается, если вы вводите данные в нескольких местах. Имя может быть написано правильно в одном месте, а в другом нет. Здесь нормализация минимизирует ошибку человека. Но с помощью первой нормальной формы вы все еще не устранили избыточность или несовместимость.

Табл. 1.1 показывает, что может быть сделано с помощью первой нормальной формы, а что нет.

Таблица 1.1. Возможности первой нормальной формы

То, чего вы не можете	То, что вы можете
Устранить избыточность.	Разбить таблицу на меньшие таблицы.
Устранить несовместимость.	Установить первичный ключ (только составной).
Иметь как повторяющиеся группы столбцов, так и более одного значения на пересечении строки и столбца (как, например, клетки в электронной таблице). Иметь несоставной первичный ключ.	Устранить повторяющиеся группы.

Вторая нормальная форма и устранение избыточности

Для того чтобы таблица имела вторую нормальную форму, для начала ее надо привести к первой нормальной форме. Еще одно условие состоит в том, что каждый неключевой столбец должен быть полностью зависим от всего первичного ключа в целом. Это означает, что если у вас есть составной первичный ключ, то другие неключевые поля должны полностью, а не частично зависеть от первичного. Рис. 1.8 иллюстрирует полную зависимость во второй нормальной форме.

[3] Здесь автор имеет в виду, что две таблицы - таблица «заказчики» и таблица «заказы», должны быть связаны друг с другом. *Науч. ред.*

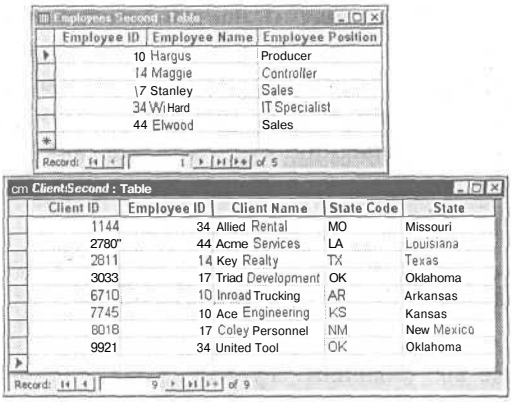


Рис. 1.8. Чтобы получить вторую нормальную форму, никаких частичных зависимостей быть не должно

Частично зависимые элементы

Частично зависимые элементы лишь зависят от части первичного ключа, а не от всего первичного ключа в целом. Поэтому, если у вас есть составной ключ, есть хорошая вероятность того, что у вас имеется частичная зависимость и первая нормальная форма. Заметьте, что на рис. 1.8 поле Employee Name полностью зависит от поля Employee ID, а не отчасти зависит от него и еще какого-то поля, создавая таким образом композицию. Так, вы можете заключить, что рис. 1.8, который является примером второй нормальной формы, не содержит каких-либо частичных зависимостей. Мы также можем утверждать, что зависимость во второй нормальной форме будет *автоматическая,* если она содержит первичный ключ, состоящий из одного показателя (а не составной!).

Обратите внимание на уменьшение повторения. Записи Employee Ids (ID служащих) 34, 10 и 17 сохраняются один раз, а не два, как это было в предыдущей форме, показанной на рис. 1.7.

Зависимость «один ко многим»

Таблица Employees (таблица служащих) является первичной таблицей и *первой* в зависимости «один ко многим». Скажем, у вас есть один служащий со множеством клиентов. Так как поле Employee ID есть в обоих таблицах, то оно называется *общим* полем. Тем не менее поле Employee ID в таблице Clients (связанной) называется *внешним* ключом. Также примите во внимание то, что унифицированность не так уж сильно важна в идентичности имен полей; гораздо более важно то, чтобы данные в этих полях совпадали. Связывание полей Employee ID в двух приведенных в качестве примера таблицах и есть то, что устанавливает зависимость «один ко многим». Вы можете пронаблюдать зависимости между таблицами в окне Relationships.

> Чтобы узнать больше об окне Relationships, см. раздел «Ссылочная целостность» в гл. *2* «Соединения и каскады».

Вам может показаться, что вторая нормальная форма есть наивысшая форма, ибо она имеет много преимуществ, но это вовсе не так. На самом деле «кодд» (Codd) определил третью нормальную и несколько других форм более высокого уровня. В этой главе рассказано о третьей нормальной форме, ибо другие формы не привносят ничего особо положительного по сравнению с третьей. Для получения третьей нормальной формы вам все еще требуется исправить кое-какие недостатки дизайна.

Третья нормальная форма - устранение неключевых столбцов, не зависящих от ключа

Что же насчет данных о штате, которые вы добавили? Тут есть только одна повторяющаяся запись. А правильно ли это? В третьей нормальной форме вы ищите неключевые столбцы, которые независимы от первичного ключа таблицы. И не путайте связь с зависимостью. Ключ State Code (Код штата) связан с данными о клиенте, но не зависит от них функционально. Это называется *переходной зависимостью.* Не удивительно то, что у вас имеются повторяющиеся значения, так как два или более клиента могут быть из одного и того же штата. Поле State Code может действовать само по себе в справочной таблице и без данных о клиенте. Можно рассмотреть этот вопрос и с другой стороны: поле State (Поле штата) зависимо от неключевого поля (State Code). Если компания переезжает и код штата меняется; то штат меняется тоже. Это не соответствует третьей нормальной форме.

Во второй нормальной форме вы устранили частичные зависимости; в третьей же нормальной форме вы устранили переходные зависимости. Это существенное различие. Информация о коде штата должна быть перемещена в новую таблицу и использоваться как справочная таблица.

В третьей нормальной форме вы устранили избыточность, с двумя лишь исключениями. Первое - это поле Employee ID. Сохранение зависимости «один ко многим» возможно. Второе - это поле State Code в таблице Client ID. Это также возможно, так как его можно использовать для справки, если потребуется. Сторона «многие» зависимости «один ко многим», как и сторона «один» зависимости «многие к одному» являются возможными значениями полей, которые можно повторять, так как мы говорим лишь об одном поле. Основная задача состоит в том, чтобы сохранить место путем использования одного поля таким образом, чтобы связать столько информации в связанных таблицах, сколько вам потребуется. Заметьте, что на рис. 1.9 справочное поле State Code в таблице Clients сделано как связное поле.

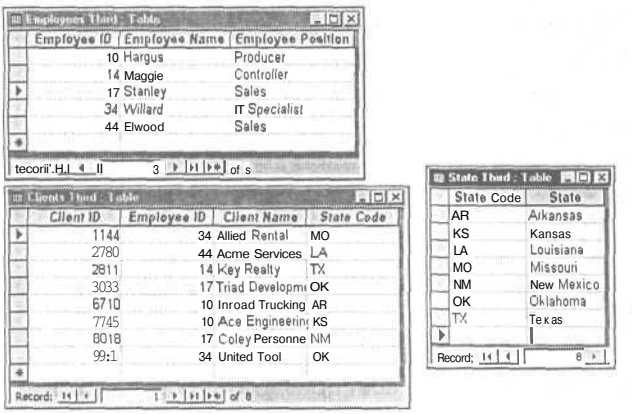


Рис. 1.9. Этот пример третьей нормальной формы иллюстрирует устранение переходных зависимостей

Нормализация в Access

Теперь вы готовы к тому, чтобы приступить к изучению практической части. Это будет легко и занимательно. Скачайте ADBE.zip с www.QuePublishing.com. Разархивируйте файлы в папку AccessByExample на диске C. Откройте базу данных Normalize.mdb из папки AccessByExample с помощью таблицы CustOrder, показанной на рис. 1.10. Эту таблицу проанализирует мастер анализа таблиц(Table Analyzer Wizard), который работает, следуя принципам нормализации, изложенным в этой главе.

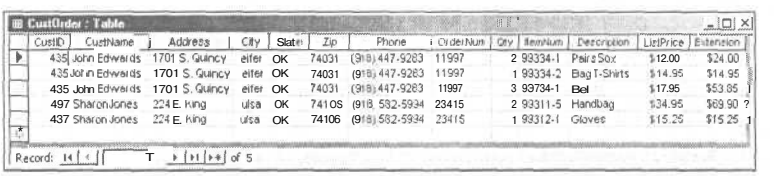


Рис. 1.10. Это таблица CustOrd, которая будет проверена анализатором таблиц

Чтобы Access автоматически нормализовал вашу таблицу, выполните следующие операции:

1. Выберите Tools, Analyze Table из меню Access, чтобы открыть первую страницу мастера анализа таблиц, как показано на рис. 1.11.

ЗАМЕЧАНИЕ. Пусть термин «Анализ» вас не смущает. Мастер анализа таблиц делает больше, чем просто анализирует таблицу. Вообще-то он разбивает вашу таблицу на более мелкие таблички, если это требуется. Будьте спокойны: ваша первоначальная всегда сохраняется.

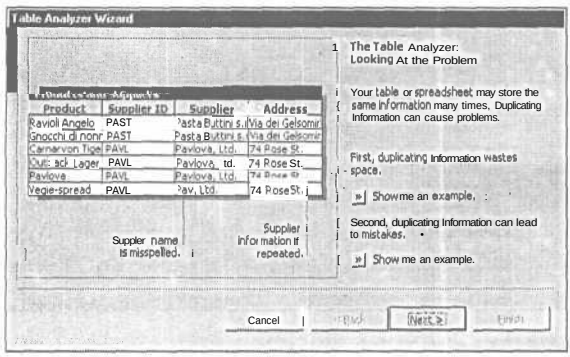


Рис. 1.11. Первая страница мастера анализа страниц дает примеры повторения, которого вы стремитесь избежать

2. Первая страница мастера анализа таблиц содержит кнопки, которые вы можете нажимать, чтобы увидеть примеры повторения. Нажимайте кнопки Show Me An Example (Покажи мне пример). Ознакомившись с информацией, которая появится после нажатия кнопок, щелкните по кнопке Next (Далее). Следующая таблица, показанная на рис. 1.12, похожа на первую, она чисто учебная. Нажмите Next после просмотра примеров и проследуйте на страницу мастера № 3.

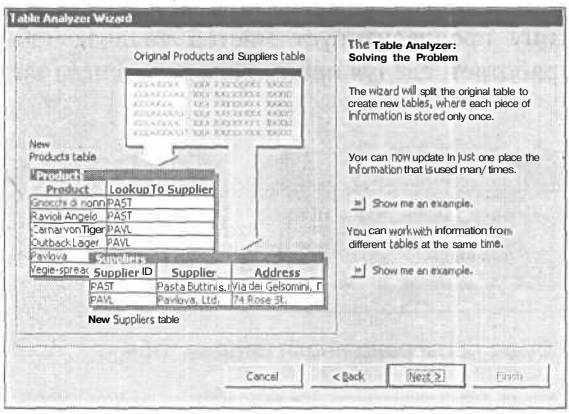


Рис. 1.12. Вторая страница мастера анализа таблиц показывает, что одну таблицу можно разбить на две или более таблиц

3. На стр. мастера № 3 выберите CustOrder, а затем нажмите Next, чтобы перейти на страницу мастера № 4. Вы можете предоставить мастеру возможность решить самому (или сами сделать выбор), какие поля в какие таблицы поместить. В этом упражнении пусть мастер решит сам. Выберите этот вариант и нажмите Next, чтобы открыть страницу мастера № 5.
4. Страница мастера № 5 - диаграмма, показывающая таблицы, поля и зависимости визуально. Щелкните по полю CustName в табл.3 и перетащите его назад в таблицу 2, как показано на рис. 1.13.

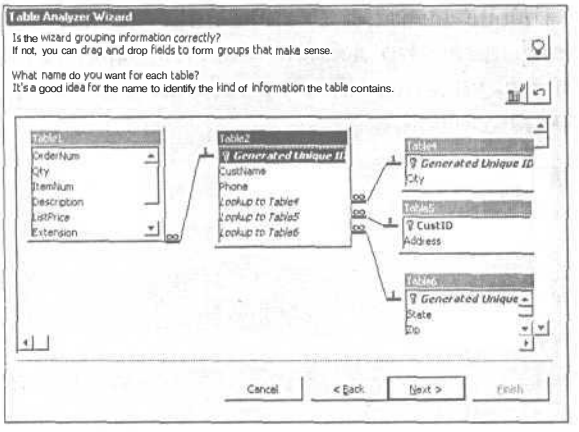


Рис. 1.13. Вы можете перемещать поля между таблицами на ар. мастера анализа таблиц № 5

5. Таким же образом переместите каждое поле из таблицы 4 через таблицу 6 в таблицу 2. Вы можете использовать горизонтальную и вертикальную полосы прокрутки. Перерасположите поля в правильном порядке: CustID, OrderNum, CustName, Address, City, State, Zip и Phone. Просто нажмите на любое из них, держите, не отпуская, кнопку мыши и перетащите его на нужное место, чтобы все было так, как на рис. 1.14.

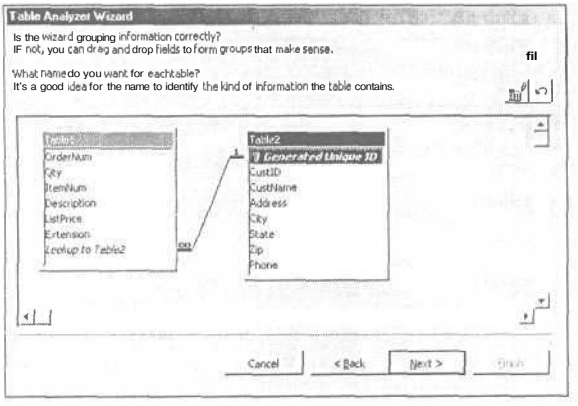


Рис. 1.14. Когда поля перерасположены, можете переименовать таблицы

6. Теперь дважды щелкните по заголовку таблицы 2 и дайте ей название - Customers. Таким же образом, дайте название таблице 1 - Orders. Поместите таблицы горизонтально, перетаскивая их за строку заголовка, так, чтобы таблица Customers была слева. Оставьте зависимость «один ко многим», которую Access заранее определил для вас, и нажмите Next для перехода на страницу мастера № 6.
7. Когда страница 6 откроется, у вас будет возможность идентифицировать первичные ключи в обеих таблицах. Щелкните по CustID в таблице Customers,

а затем щелкните по кнопке Set Unique Identifier (Установить уникальный идентификатор), показанной на рис. 1.15. Это должно удалить первичный ключ Generated Unique ID (Созданный уникальный ID) в таблице Customers. Нажмите Next для перехода на страницу мастера № 7.

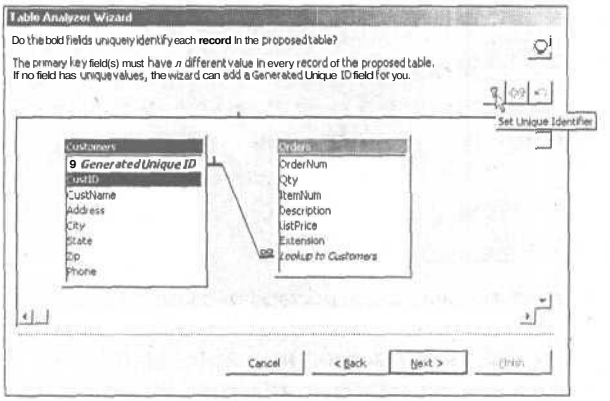


Рис. 1.15. Вы можете установить первичный ключ на стр. мастера анализа таблиц № 6

8. На стр. *1* спрашивается, нужен ли вам запрос. Нажмите No, «Не создавать запрос», а затем нажмите Finish (Завершить). Закройте окно помощи после прочтения, чтобы увидеть две таблицы, показанные на рис. 1.16. Вот и все!

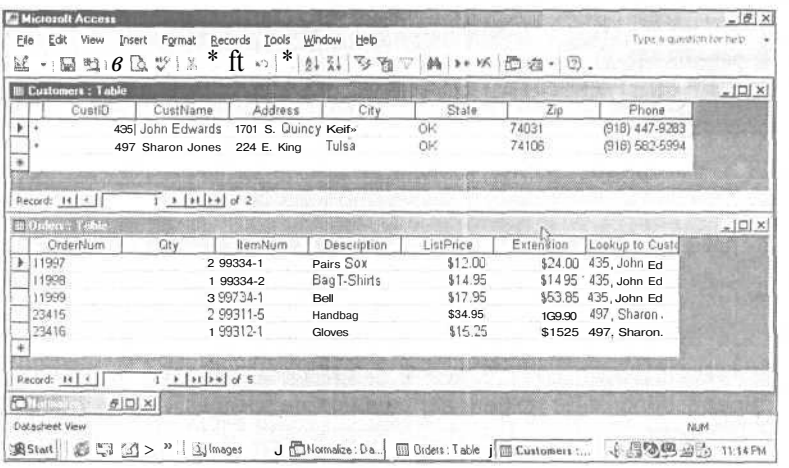


Рис. 1.16. Мастер анализа таблиц создал для вас две таблицы

Что дальше?

В этой главе были изложены основы проектирования базы данных. В следующей главе вы узнаете больше о зависимостях между таблицами. Примеры соединений, каскадов и ссылочной целостности дадут вам практические навыки работы с этими важными объектами.

2

Объединения и каскады

В гл. 1 рассматривались основные принципы разработки баз данных. Вы уже знаете, что, возможно, вам потребуется разбить большую таблицу на более мелкие, чтобы улучшить эффективность и обеспечить целостность данных. В этой главе больше внимания уделяется связям между таблицами. Чтобы правильно понять, каким образом Access управляет связями между таблицами, вам нужно понять объединения и каскады. Конкретнее вы узнаете:

- что такое объединения и почему следует их использовать,
- что такое ссылочная целостность,
- что такое каскадное обновление и удаление,
- как задавать ссылочную целостность и каскады в окне «Связи» (Relationships).

Что такое объединения и зачем их использовать? Прежде чем мы ответим на эти вопросы, вам нужно понять, где их использовать. Основные моменты Access, в которых вы используете линии объединений, встречаются при проектировании запросов, к которым можно обращаться через формы и отчеты, а также через окно Relationships. Впрочем, вы можете использовать объединения в любой другой части рабочей среды Access, применяя SQL (структурированный язык запросов).

Для более подробного рассмотрения SQL см. раздел «Введение в SQL» в гл. 4.

Вы используете линии объединений, которые просто показывают, что происходит в кулуарах SQL, в конструкциях запросов, чтобы получить различные типы реляционных результатов, таких, как условный поиск или предварительное сравнение. Вы используете линии объединений в окне связей, чтобы установить более постоянные (хотя их можно менять) глобальные объединения на связях между таблицами. Термин *«глобальные»* используется для объединений, которые влияют на многие объекты Access, такие, как формы, отчеты и запросы. Например, когда вы проектируете форму в мастере форм и выбираете из двух таблиц поля, которые объединены в окне связей, Access автоматически предложит вам создать подформу для второй таблицы, чтобы стало возможным отображение связи типа «один ко многим». Это удобный способ вводить заказы для клиентов, потому что вам с одного взгляда видно клиента и его заказы.

Возвращаясь к исходному вопросу, *объединения Access* есть операции SQL, которые позволяют вам считывать записи из двух таблиц, между которыми существует связь, заданная путем связывания двух столбцов этих таблиц. Эти связи бывают типа «один к одному», «один ко многим» или «многие ко многим». С функциональной точки зрения объединение сопоставляет записи двух таблиц. Общее поле или поле объединения должно присутствовать в обеих таблицах и содержать согласующие данные. Объединение, как правило, является частью запроса *select* (выбрать).

Зачем вам применять объединения? Access не заставляет вас их использовать. Впрочем, они являются необходимостью для людей, которые относятся серьезно к наиболее эффективному использованию возможности Access работать с реляционной моделью данных. Хоть и существует возможность разрабатывать мощные запросы, использующие более одной таблицы, *не применяя* объединения, вы можете попасть впросак, если не знаете, как правильно составлять такие типы запросов.

Неправильно спроектированный запрос, использующий две или более таблицы без объединений, может порождать то, что называется декартовым произведением, что редко является полезным или желательным. Декартово произведение названо в честь французского математика и мыслителя Рене Декарта. Хотя обычно декартово произведение получается без применения объединений, но, если соединить каждую строку одной таблицы с каждой строкой другой таблицы, получится декартово объединение, которое по существу и является декартовым произведением. При нормальных обстоятельствах зачем вам может понадобиться такое объединение?

Если вы не сделаете объединение между таблицами, то эти типы запросов будут отображать все возможные комбинации из каждой таблицы, которой касается этот запрос. Например, две 60-строчные таблицы, заданные таким образом в запросе, дадут результат размером 3600 строк. Это происходит потому, что 60 на 60 равно 3600. Без объединения или условного оператора where (где) в запросе, содержащем две или более таблицы, вы легко можете запутаться в результатах. Короче говоря, объединения обычно являются самым простым, наиболее эффективным методом сравнения и связывания таблиц.

Если вы позволите Access нормализовать ваши таблицы, он создаст объединения в окне связей. Поскольку в ваших же интересах освоить это окно, прочтите следующий обзор того, что оно может вам предложить.

1. Из папки AccessByExample откройте базу данных Northwind (NWIND.MDB или NORTHWIND.MDB). Вы также можете скопировать ее с компакт-диска Access/Office.
2. Когда вы увидите вводный экран Northwind, нажмите OK, а затем нажмите на кнопку Display Database Window (Отобразить окно базы данных).
   Если, находясь в окне Access, вы нажмете кнопку «связи» (которая находится двумя кнопками левее знака вопроса), вы попадете в окно связей. Взгляните на таблицы, состоящие в связи «один ко многим».
   Заметьте, что там присутствуют линии объединений, соединяющие различные таблицы. Также заметьте «1» со стороны «один» связи и лежащую на боку цифру 8, представляющую сторону «многие»этой связи, как показано на рис. 2.1.
3. Используйте двойной щелчок мыши на линии объединения между полями **Клиенты и Заказы,** чтобы открыть диалоговое окно редактирования связей.
4. Нажмите кнопку **Тип объединения.** Появится диалоговое окно свойств объединения, как показано на рис. 2.2.

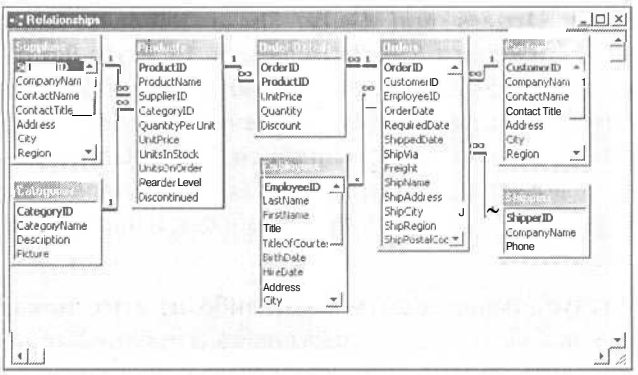


Рис. 2.1. Линии объединения в окне связей дают графическое представление связи типа «один ко многим» между таблицами

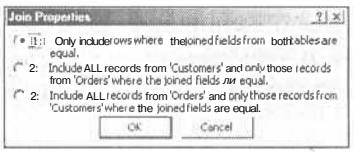


Рис. 2.2. Диалоговое окно свойств объединения содержит три опции, которые соответствуют трем типам объединений, имеющимся в Access

Это то же самое диалоговое окно, которое вы видите, когда соединяете таблицы в запросе. Для вас важно понять, что означает каждая из этих опций, чтобы вы могли правильно настраивать связи между таблицами, независимо от того, где они задаются, в окне связей или в запросе.

Опции этого диалогового окна включают:

1: **Only Include Rows Where the Joined Fields from Both Tables are Equal** (Включать только те строки, в которых объединенные поля обеих таблиц совпадают).Такое объединение называется *внутренним объединением* или *объединением по эквивалентности.* Это просто значит, что значения, находящиеся в объединенных полях обеих таблиц, должны быть равны или совпадать для того, чтобы можно было считать их запись. Это наиболее общий тип объединений. Он также может использоваться для поиска значений.

2: **Include ALL Records From Customers and Only Those Records from Orders Where the Joined Fields are Equal** (Включать ВСЕ записи из заказчиков и только те записи из заказов, в которых объединенные поля эквивалентны). Это левостороннее внешнее объединение. Это означает, что включаются все записи таблицы, расположенной слева, а записи правой таблицы включаются только в том случае, если их объединенное поле содержит значения, совпадающие со значениями в объединенном поле левой таблицы.

3: **Include ALL Records from Orders and Only Those Records from Customers Where the Joined Fields are Equal** (Включать ВСЕ записи из заказов и только те записи из заказчиков, в которых объединенные поля эквивалентны). Это правостороннее внешнее объединение. Это означает, что включаются все записи таблицы, расположенной справа, а записи левой таблицы включаются только в том случае, если их объединенное поле содержит значения, совпадающие со значениями в объединенном поле правой таблицы.

Итак, в каких случаях вам следует использовать какой-либо из этих типов объединений? Предположим, у вас есть таблица заказчиков и таблица заказов, к которой вы хотите направить запрос. Таблица «Заказчики» является «одним» в соотношении «один ко многим» и находится слева, а таблица «Заказы» является «многими» этого соотношения и находится справа. Предположим также, что вы хотите отображать только тех заказчиков, у которых есть заказы. В этом случае вы используете внутреннее объединение. Если же вы хотите отображать все соответствующие заказы и всех заказчиков, независимо от наличия у них заказов, то вам следует использовать левостороннее внешнее объединение.

5. Закройте базу данных Northwind.

Так как большинство людей обладают преимущественно визуальным восприятием, далее мы приводим наглядный пример, который поможет вам представить, что происходит во внешнем объединении.

1. Запустите Microsoft Access. Если он уже запущен, переключитесь в окно базы данных (Database).
2. Выберите в меню пункты **Файл** (File), **Новый** (New) и **Пустая база данных** (Blank Database). Назовите эту базу данных Joins [4] и разместите ее в папке AccessByExample.
3. В пункте **Объекты** (Objects) выберите **Таблицы** (Tables); затем нажмите **Создать таблицу** (Create Table) в пункте **Вид проекта** (Design View) на панели инструментов окна базы данных, чтобы открыть вид проекта новой таблицы.
4. После того как вы попадете в **Конструктор таблицы** (Table Design), введите Value в названии первого поля и нажмите клавишу Tab. Введите а в поле типа данных (Data Type) вы увидите, как появится слово **Автономер** (Autonumber), как показано на рис. 2.3. Щелкните на ячейке в столбце **Имя поля** (Field Name) в следующем за Value ряду.
5. В следующем поле наберите Результат (Result) и сделайте его текстовым полем размера 20. Нажмите F6, чтобы попасть в раздел **Свойства поля** (Field Properties) и поле **Размер поля** (Field Size).

[4] Данная база иллюстрирует типы объединения в базе, поэтому она названа автором «Объединения». - *Науч. ред.*

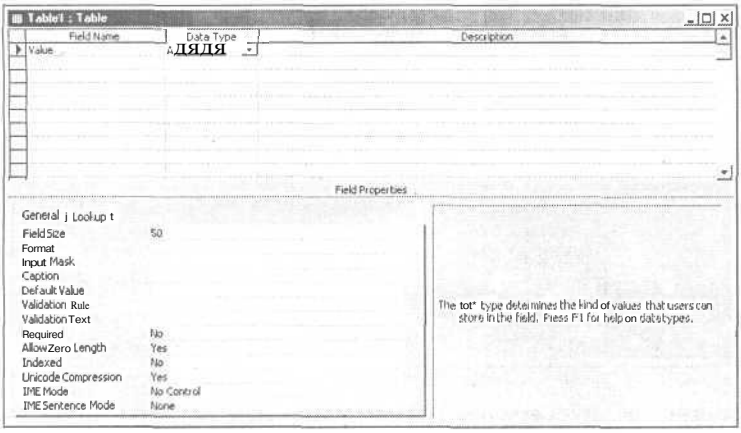


Рис. 2.3. Когда вы наберете «а» в столбце Тип данных, автоматически вставляется Автономер

6. Щелкните в поле **Значение,** а затем нажмите кнопку **Первичный ключ** (Primary Key) в верхней части экрана, как показано на рис. 2.4, чтобы сделать поле Значение первичным ключом. Теперь можно вводить данные.

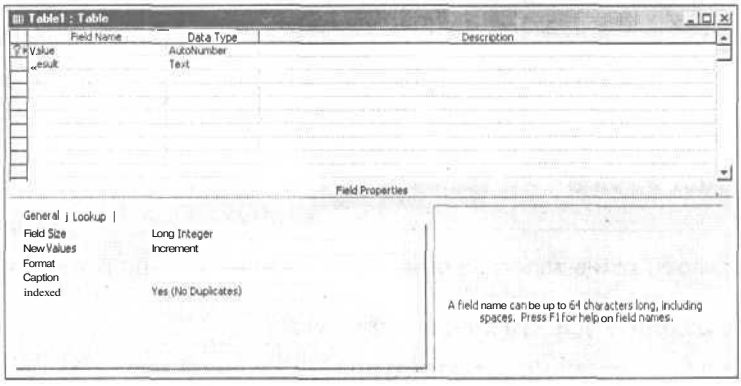


Рис. 2.4. При использовании кнопки Первичный ключ предполагается, что вы вводите только уникальные значения в поле, для которого используется эта кнопка

7. Нажмите кнопку Вид в панели инструментов ниже меню **Файл,** как показано на рис. 2.5. Когда Access предложит вам сохранить таблицу перед открытием, нажмите **Да** (Yes) и назовите файл «А».
8. Поле **Значение** автоматически заполняется последовательными значениями из-за автонумерации. Введите в поле **Результат** (Result) не соответствует B (doesn't match B).
9. Находясь в этом поле, при помощи курсора опуститесь ниже, к следующей записи и нажмите Ctrl+' (апостроф) одновременно, чтобы дублировать предыдущую запись; затем сделайте то же самое со следующей записью.

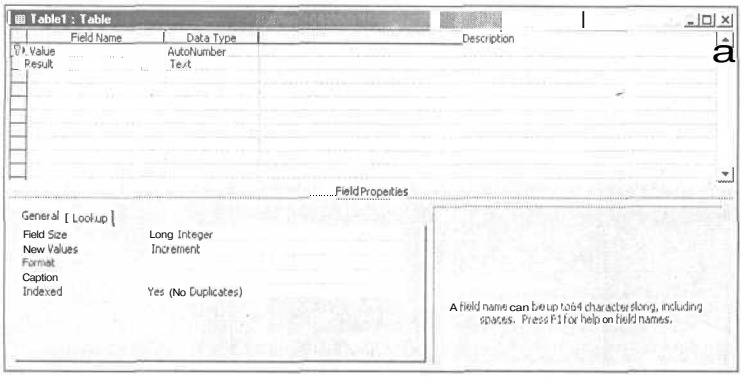


Рис. 2.5. Кнопка Вид сохраняет шаг, открывая таблицу сразу же после ее сохранения

10. В следующих трех полях введите соответствует B (matches B). У вас должно получиться шесть записей, которые выглядят так, как на рисунке *2.6.*

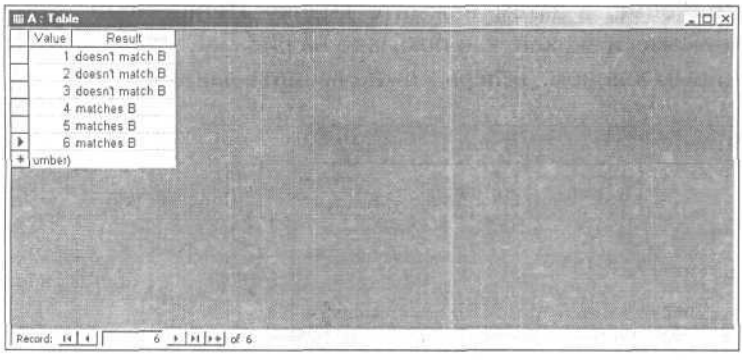


Рис. 2.6. Таблица A показывает, какие записи соответствуют записям таблицы B

11. Закройте таблицу и сохраните все изменения в проекте.
12. В окне **Базы данных** под таблицами щелкните правой кнопкой мыши на таблицу A выберите **Копировать** (Copy).
13. Нажмите кнопку **Вставить** (Paste) в панели инструментов. Когда будет запрошено имя, введите B.

ПОДСКАЗКА. Вы также можете щелкнуть правой кнопкой мыши на заголовок окна **Базы данных** и выбрать пункт **Вставить.**

14. Выбрав таблицу B, нажмите кнопку **Конструктор** в панели инструментов.
15. Поменяйте тип данных в поле **Значение** на Число (Number) и нажмите кнопку **Первичный ключ,** чтобы удалить первичный ключ.
16. В разделе **Свойства поля,** поменяйте **Индексированное поле** внизу на Да (Дублирование OK) (Yes (Duplicates OK)). Нажмите кнопку Вид и сохраните изменения.

17. Вы уже должны быть готовы вводить данные, но в этом случае вы измените данные. Выделите первые три ряда, щелкнув на селектор строк слева; затем, удерживая, перетащите их на три строки ниже, как показано на рис. 2.7.

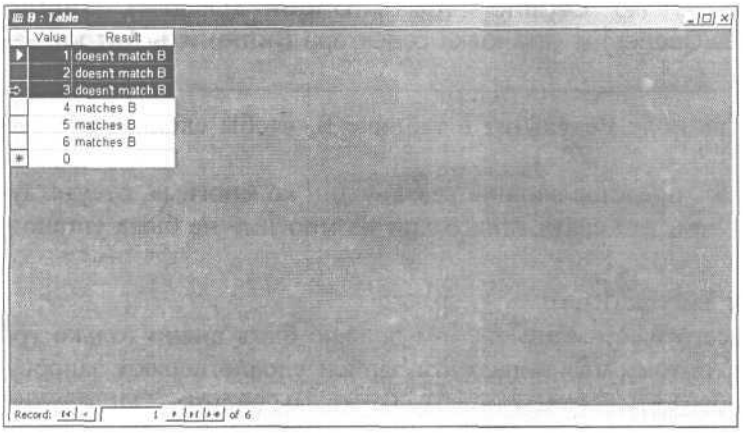


Рис. 2.7. Передвиньте первые три строки таблицы B в новое положение

18. Нажмите Ctrl+X или кнопку **Вырезать** (Cut) на панели инструментов, а затем нажмите Да в окне сообщения, показанном на рис. 2.8. Выберите строку целиком из позиции **Новая запись** (New Record) (*), а затем нажмите **Вставить** на панели инструментов.

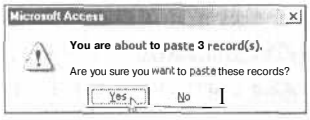


Рис. 2.8. Меняем таблицу B для возможности объединения с таблицей A

19. Поменяйте значения от 1 до 3 в столбце **Значение** (Value) на значения от 7 до 9.
20. Поменяйте B в поле **Результат** каждой записи на A. числа в поле **Значение** (Value) должны возрастать от 4 до 9. Текст в поле **Результат** должен быть либо «соответствует A», либо «не соответствует A». Закройте таблицу.

Теперь вы готовы создать запрос, чтобы посмотреть, что же происходит собъединениями.

1. В пункте **Объекты,** выберите **Запросы** (Queries), а затем **Новый** (New) чтобы открыть окно **Новый запрос** (New Query).
2. Щелкните на **Обзор проекта** (Design View), а затем OK, чтобы открыть диалоговое окно **Показать таблицу** (Show Table). Выберите A, а затем нажмите кнопку Добавить (Add). Щелкните два раза на B так, чтобы стало видно оба метода, для того чтобы добавлять таблицы. Выберите **Закрыть** (Close).

Диалоговое окно **Свойства объединения** появится, как показано на рис. 2.9, позже в этой главе. Заметьте, что поле **Значение** соединяется автоматически.

ЗАМЕЧАНИЕ. Вы можете включать и выключать опцию автообъединения путем выбора пунктов меню **Сервис** (Tools), **Опции** (Options), **Таблицы/Запросы** (Tables/Queries) и установки селектора **Включить Автообъединение.**

3. Щелкните два раза на поле **Результат** в таблице B, чтобы скопировать его контейнер запроса.
   Заметьте, что "знак 8", представляющий связь «один ко многим», отсутствует. Это объясняется тем, что связь типа «один ко многим» не была установлена в окне связей.
4. Щелкните на значок **Запуск** (Run).
   Хотя в таблице B всего шесть записей, вам должно быть видно только три подходящие записи, потому что только эти записи удовлетворяют запросу. Все они должны отображать соответствие A. То, что вы видите, - внутреннее объединение, или объединение по эквивалентности. Таким образом, это соответствие типа «один к одному».
5. Нажмите кнопку **Вид,** чтобы вернуться к таблице проекта.
6. Щелкните по выпадающему меню, находясь в поле Result: строка, сменить таблицу на A, а затем щелкните по значку Run (Запуск). Вы должны были получить три записи, в которых написано «Соответствует B».
7. После того как вы вернетесь в **Обзор проекта** щелкните правой кнопкой мыши на линию между таблицами, а затем выберите пункт **Свойства объединения,** чтобы открыть диалоговое окно **Свойства объединения.** Вы также можете дважды щелкнуть по линии, но эта линия должна быть жирной. Если это не так, щелкните правой кнопкой мыши.
8. Выберите п. 2 (рис. 2.9). Прочтите, что написано в п. 2. Вы должны включить ВСЕ записи из таблицы A и только те записи из таблицы B, в которых объединенные поля эквивалентны. Это внешнее левостороннее объединение.

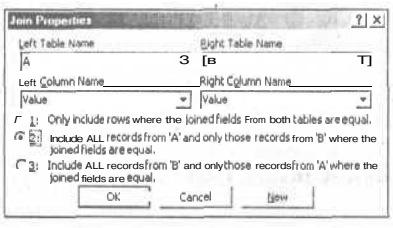


Рис. 2.9. Второй пункт, который включает все записи из таблицы слева, представляет внешнее левостороннее объединение

9. Нажмите OK и запустите запрос. Заметьте, что теперь вам видны все записи, а не только те записи, которые совпали.

10. После того как вы вернетесь в **Вид проекта,** щелкните правой кнопкой на линии объединения, выберите свойства объединения и поменяйте их на п. 3. Перезапустите запрос.
    Теперь вы должны увидеть, три записи с «соответствует B» в поле **Результат** и три пустые записи.
11. Вернитесь в Вид проекта, смените свойства обратно на п. 2 и нажмите OK. Щелкните на значок **Сохранить,** а затем назовите запрос Join. Закройте запрос нажатием X.
12. Нажмите кнопку **Связи** (Relationships), изображенную на рис. 2.10, а затем нажмите кнопку **Показать таблицу** (жирный желтый знак плюс) в окне **Связи,** чтобы выбрать таблицы.

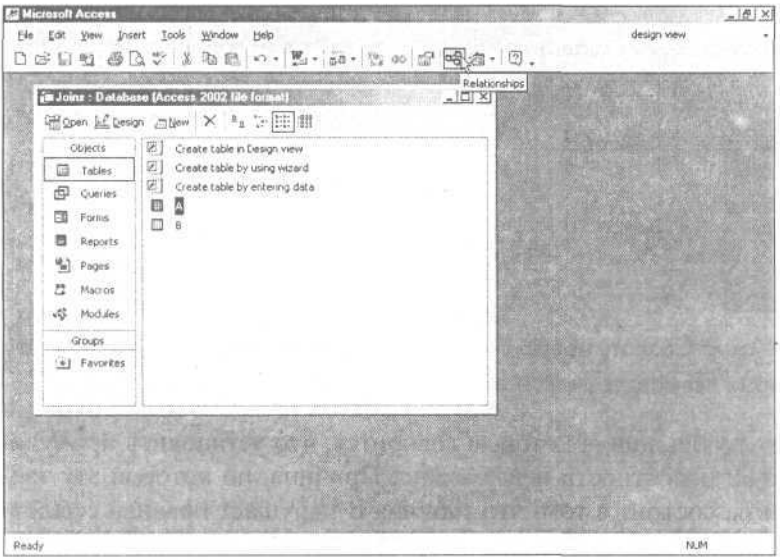


Рис. 2.10. Кнопка **Связи** открывает окно **Связи,** в котором вы можете задавать связи в вашей базе данных

13. Выберите таблицы A и B так же, как вы делали это в диалоговом окне разработки запросов.
14. Нажмите и, удерживая, перетащите поле **Значение** из таблицы A в таблицу B, таким образом попадая в диалоговое окно **Редактирования связей,** как показано на рис. 2.11.
15. Нажмите кнопку **Создать новую** (Create New), чтобы войти в диалоговое окно **Создать новую,** как показано на рис. 2.12. Вы можете щелкнуть по соответствующему выпадающему меню, чтобы ввести имена таблиц и столбцов.
16. Нажмите кнопку **Отмена** (Cancel), а затем щелкните флажок **Принудительная ссылочная целостность** (Enforce Referential Integrity). Нажмите кнопку **Создать.**

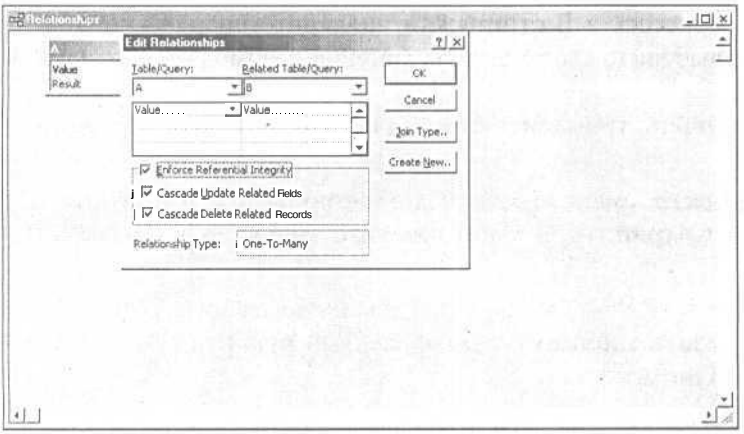


Рис. 2.11. Диалоговое окно Редактирования связей позволяет вам задавать связи между таблицами

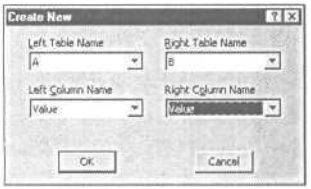


Рис. 2.12. Диалоговое окно Создать новую отображает названия таблиц слева и справа, а также поля, которые являются основой объединения

17. Отобразится окно сообщения, в котором говорится, что установить принудительную ссылочную целостность невозможно. Причина, по которой эту таблицу создать нельзя, состоит в том, что таблица B нарушает правила ссылочной целостности.
18. Нажмите кнопку OK в ответ на это сообщение, а затем кнопку **Отмена.** Закройте окно связей, затем нажмите кнопку **Нет** чтобы не сохранять изменения.

Ссылочная целостность

Как показывает рис. 2.13, записи с 4-й по 6-ю являются единственными подходящими записями. Если вы соедините эти две таблицы левосторонним объединением, то вы получите три *«подвисшие»* записи в таблице B. Если A — таблица заказчиков, а B - таблица заказов, то у вас получатся заказы без заказчиков.

Ссылочная целостность определяет, что нельзя добавить в таблицу значение с внешним ключом, если в таблице, на которую делается ссылка, не существует соответствующего значения. Попросту говоря, прежде чем значение можно будет ввести в таблицу B, соответствующее ему значение должно существовать в таблице A. Если значение в таблице меняется, соответствующее значение в таблице B не может стать подвисшим. На рис. 2.13 показано нарушение ссылочной

целостности. Если вам нужно левостороннее объединение, то в таблице В нет значений, соответствующих значениям с 1-го по 3-е таблицы А.

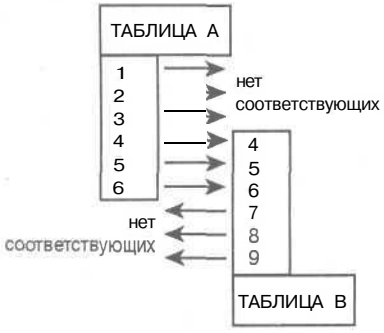


Рис. 2.13. На этой диаграмме видно, что в таблице В нет записей, соответствующих записям с 1-й по 3-ю таблицы А

Ссылочную целостность не следует путать с целостностью сущностей-объектов, которая гласит, что первичные ключи не могут содержать нулевые значения. Это имеет настоящий смысл, если вы считаете, что не существует способа получить уникальные значения, а первичный ключ может быть пустой.

Заметьте, что на рис. 2.13 обе таблицы имеют несовпадающие значения. Если А - главная таблица, то несовпадающие значения допустимы. Но как быть с таблицей В? Таблица В содержит повисшие записи в левостороннем объединении.

Что же вам надо сделать, чтобы решить эту дилемму? Вы должны изменить записи таблицы В таким образом, чтобы они соответствовали таблице А. Для этого проделайте следующее:

1. Откройте таблицу В.
2. Замените значения с 7-го по 9-е на значения с 4-го по 6-е. Это дублирует поле **Значения,** но это приемлемо в связанной таблице.
3. Замените записи в столбце **Результат,** в которых в данный момент написано не соответствует А [doesn't match А), на соответствует А(matches А). (Вы можете дублировать с использованием Ctrl+' так же, как вы это делали раньше.) Выйдите и сохраните таблицу.

Теперь еще раз попробуйте включить принудительную ссылочную целостность.

1. Откройте окно связей (Relationships).
2. Нажмите кнопку **Показать таблицы,** щелкните два раза на таблицы А и В, чтобы поместить их в окно. Нажмите **Закрыть.**
3. Нажмите и, удерживая, перетащите поле **Значение** из таблицы А в таблицу В.
4. Установите флажок **Принудительная ссылочная целостность,** а затем установите галочки напротив пунктов **Последовательное обновление связан-**

ных **полей и Последовательное удаление связанных записей** в диалоговом окне Редактирования **связей.**

5. Нажмите кнопку **Создать** и закройте окно связей.
6. Нажмите Да, чтобы сохранить изменения.

Поздравляем! Теперь вы устранили проблему и установили ссылочную целостность. Если вам это не удалось, вернитесь назад и повторите все шаги. И следите за тем, чтобы вводить все правильно.

Каскады

Что такое каскады? Вы можете представлять это так. Если вы поменяете значение в одной таблице, оно автоматически последовательно изменяется или обновляется, независимо от того, сколько записей привязаны к той записи в другой таблице. Аналогично, если вы удалите значение в одной таблице, соответствующие записи другой таблицы также удаляются. Первое работает на уровне поля, а второе - на уровне записей. Таким образом, когда вы меняете значения первичного ключа или удаляете записи в главной таблице, Access автоматически вносит необходимые изменения в связанные таблицы так, чтобы сохранялась ссылочная целостность.

Если вы поменяете номер заказчика, вам нужно будет обновить его заказы соответствующим образом. Иначе его заказы не синхронизированы. Точно так же, если вы удалите этого заказчика, вам нужно будет удалить его заказы. Это еще одно отличие от разработки запросов.

Теперь давайте посмотрим на это правило в действии, для чего продемонстрируем каскады на примере. Для начала рассмотрим свойство типа данных Автономер в поле **Значение** в таблице А. Вам нужно это сделать, потому что вы не можете менять автонумеруемое поле; по крайней мере вы не сможете сделать этого, не влезая в дебри программирования.

1. При выбранной таблице А нажмите кнопку **Конструктор.**
2. В столбце **Тип данных** поля Значение щелкните по выпадающему списку и выберите **Число.** Оставьте размера поля **Длинное целое** (Long Integer).
   Вы должны получить сообщение о том, что тип данных изменить нельзя, потому что это часть одной или более связей. Решение состоит в том, чтобы временно удалить связь. Нажмите OK в ответ на сообщение и закройте таблицу не сохраняя.
3. Откройте окно **связей,** щелкните по линии объединения между А и В и нажмите клавишу Delete. Ответьте Да на последующий вопрос и закройте окно **связей.**

Теперь мы можем поменять ключ **Значение** в таблице А на Номер (Длинное целое).

1. Нажмите кнопку **Проект** в таблице А, поменяйте тип данных поля **Значение** на **Число,** а затем закройте и сохраните таблицу.

2. Нажмите кнопку **Связи,** чтобы открыть окно связей.
3. Выберите **Тип объединения.** Выберите второй пункт, чтобы включить все записи таблицы A, и только соответствующие записи таблицы B. Нажмите OK.
4. Установите галочку **Принудительная ссылочная целостность,** а затем установите галочки напротив пунктов **Последовательное обновление связанных полей и Последовательное удаление связанных записей.**
5. Нажмите **Создать.** Закройте окно связей.
6 Щелкните дважды по таблице A. Поменяйте значение 4 в таблице A на 7, а затем откройте таблицу B.

   Обратите внимание на две вещи. значения 4 больше нет. Две записи были заменены на Значение 7. Причина, по которой вы можете продолжить изменения, состоит в том, что вы выбрали последовательное обновление связанных полей.
7. Откройте таблицу A и нажмите кнопку развертывания (+), иногда также называемую drill down button, находясь в поле **Значение** 7, чтобы посмотреть связанные с ним записи. Нажмите кнопку свертывания (-), чтобы вернуть окно в нормальное состояние. Затем щелкните по селектору слева от значения 7, нажмите кнопку delete и ответьте Да на вопрос о последовательном удалении. Закройте таблицу.
8. Откройте таблицу B.

Две подобные записи со значением 7 в столбце **Значение** таблицы B были автоматически удалены, когда вы удалили запись со значением 7 в столбце **Значение** таблицы A, как показано на рис. 2.14. Это произошло благодаря принудительному последовательному удалению.

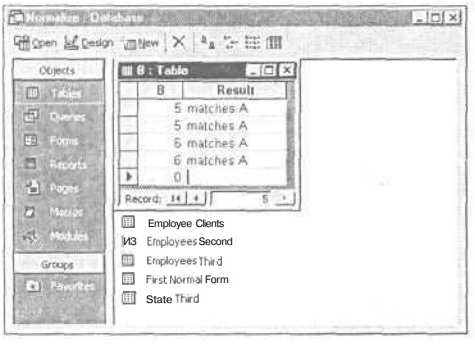


Рис. 2.14. Если вы удалите значение с одной стороны связи, последовательное удаление гарантирует, что все подобные значения со стороны множества для этой связи также будут удалены

Распечатка содержимого окна связей

По мере того как вы создаете все большее и большее количество связей в вашей базе данных, становится все труднее просматривать их все без прокрутки. Access дает простой способ получить печатную копию содержимого окна связей, на которой будут отображаться все связи вашей базы данных. Графические результаты оформляются в форме отчета, который можно сохранить. Следующие пункты показывают, как это сделать.

1. Нажмите кнопку **Связи,** затем **Файл, Печать связей.**
2. В виде проекта создается отчет. Щелкните по значку принтера в верхней части экрана, чтобы распечатать связи.
3. Access спросит, хотите ли вы сохранить отчет. Если вы выберете сохранить, он будет находиться в отчетах в разделе **Объекты** в окне **Базы данных.**

Что дальше?

Теперь вы лучше стали понимать объединения и каскады. В следующей главе основное внимание будет уделяться таблице, сердцу базы данных. Вы узнаете внутреннее строение таблицы путем ознакомления с тем, что собой представляют типы данных. Вы также узнаете, как создавать таблицы и добавлять в них данные.

3
Таблица - душа базы данных

Во гл. 2 вы узнали о связях между таблицами. Настоящая глава раскрывает суть таблиц и их структуру, состоящую из полей. Таблица - душа любой базы данных, потому что практически все, что происходит в базе данных, крутится вокруг таблицы. Когда вы выполняете запрос, он основывается на таблице или на другом запросе, основанном на таблице. Когда вы открываете форму, она строится на основе таблицы или запроса. Общий смысл понятен. Конкретнее вы узнаете:

- о таблицах и полях, из которых состоят таблицы;
- о преимуществах жесткости таблицы;
- какопределить, какие типы данных следует использовать в поле;
- о форматах полей;
- как проверить ваши форматы.

Поля - элементы построения таблиц

Если таблица - душа базы данных, то о поле можно сказать, что это душа таблицы. Запись - это просто совокупность полей. Если вы будете искать определение поля в Интернете, вам будет очень затруднительно найти такое определение, которое адекватно описывает, что такое поле, а не что оно делает. Некоторые говорят, что поле - это наименьший элемент базы данных, но едва ли это определение является наглядным. С другой стороны, то, что поле очень мало по сравнению с размерами современных баз данных, позволяет определить его таким образом.

Клетка - это наименьший органический элемент человеческого тела, а ученые только начинают понимать значение клетки для диагностики и лечения большого количества заболеваний человека. Короче говоря, поле имеет критическое значения для здоровья базы данных. *Поле* - это просто категория или объединение общепринятых значений. Вы не будете писать название города там, где нужно писать название штата, так же как вы не будете писать фамилию в поле имени. Эти категории должны сохраняться тщательно, чтобы соблюдать точность.

Хотя в Excel включена проверка правильности, в электронной таблице проще обойти эту проверку с неточностями и несовместимостью, чем в базе данных. Фактически это одно из основных преимуществ баз данных. Хотя базы данных обладают гораздо более жесткой структурой, чем электронные таблицы, они гораздо менее склонны к ошибкам. Вдобавок базы данных могут работать с большими объемами данных намного быстрее и с гораздо меньшими усилиями, чем электронные таблицы, особенно если их правильно настроить и эксплуатировать.

Жесткая структура окупается за счет вынужденной правильности ввода. Это в основном происходит на уровне поля. Так как поле - это место, где вы рассчи-

тываете, обновляете, изменяете, обрабатываете и сравниваете данные, то неудивительно, что Access поддерживает соответствующий набор проверок правильности, которые вы можете применять к полям для сохранения точности. Например, предположим, что вы вводите номера социального страхования в таблицу. Если вы введете восемь вместо девяти знаков, вам необходимо знать, что одного знака не хватает. Шаблонный ввод Access укажет на это по ходу проверки правильности расстановки тире.

Нормализация (совместно со здравым смыслом) поможет вам определить, в каких таблицах размещать какие поля. Здравый смысл подсказывает, что надо содержать в таблице все поля, относящиеся к одному и тому же предмету. Вам не нужен номер телефона работника в таблице заказчиков, например. Но сколько же информации должно содержать каждое поле?

Сторонники строгости в базах данных говорят, что совокупная информация (несколько категорий) не должна находиться в одном поле. Например, название города, штата, а также индекс не должны быть в одном поле. Так же имя и фамилия не должны появляться в одном и том же поле. Конечно же, Access обладает инструментами для того, чтобы достаточно просто осуществлять разделении (разбор) и сведение (соединение) данных. Но вы должны стараться удерживать данные поля в одной категории насколько это возможно.

Хорошим эмпирическим правилом является содержание в таблице только тех данных, которые относятся к одному предмету, а в поле - только тех данных, которые относятся к одной категории. Если вам приходится нарушить эти правила, удостоверьтесь, что на это есть достойная причина. Хороший тестовый пример представляет собой список адресов. Некоторым людям нравится подход типа «адрес1, адрес2». Поступая таким образом, вам не придется беспокоиться о досадных пустых строчках, которые появляются в отчетах. С таким подходом вы вводите адрес подобно тому, как это сделано далее:

Заказчик:	Ralph Jones
Адрес 1:	2742 North Cedar
Адрес2:	Suite 200
Адрес3:	Indianapolis, IN 46290

На первый взгляд не существует логических причин нарушать эти правила. Вам не нужно беспокоиться о пустой строчке, если нет второго адреса. Вы просто вводите столько строчек с адресами, сколько вам нужно. Но что будет, если вам понадобится упорядочить их по почтовому индексу? Или если вам нужно будет просмотреть список заказчиков из Индианы? Тут вы не сможете обойтись без программирования. Очевидно, что вы решили одну проблему и получили другую. Это еще одна причина, по которой необходимо действовать согласно принципам хорошего дизайна, и это применимо не только к организации данных в каждом поле, но и к типу данных в каждом поле.

Типы данных Access 2002

Как вам определить, какой тип данных нужно использовать в каждом поле? Access позволяет вам использовать текстовый тип данных (по умолчанию) для всех полей, если вы этого хотите. Но в данном случае этот подход неблагоразумен по нескольким причинам. Вы не сможете должным образом сортировать числовые данные, а также посчитать количество дней между датами. И это всего лишь верхушка айсберга.

Ответ находится в знании причин, по которым вы используете те или иные типы данных. Давайте посмотрим на все это в действии на примере. На рис. 3.1 показано применение текстового типа данных. Там также показаны другие типы данных, которые вы можете использовать.

Рис. 3.1. Выпадающее меню в виде таблицы показывает возможные типы данных

Текстовый

Текстовый - наиболее часто используемый в Access тип данных. Он превосходно подходит для хранения адресов. В поле шириной по умолчанию 50 символов вмещается практически любой адрес. Этот тип также подходит для полей с краткими описаниями. Вы также можете использовать это поле для числовых данных, не требующих расчетов, таких, как телефонные номера и почтовые индексы. Но что если вам потребуется поле длиной более 255 символов? Тогда вам следует перейти к описанию следующего типа данных.

Поле Memo

Тип Memo предназначен для полей, длина которых превосходит 255 символов. Хорошим примером является длинное поле описания. В это поле можно ввести целый абзац или даже больше без проблем. Поле Memo может хранить до

65,535 символов, что приближенно равняется 32 страницам текста. Если вы хотите хранить форматированный текст или длинные документы, предпочтительнее будет тип OLE (технология связывания и внедрения объектов).

Числовой

После того как вы выберете тип **Числовой,** можно будет установить несколько параметров в части **Размер поля** раздела **Свойства поля** в окне **Вид проекта.** В табл. 3.1 дается описание характеристик полей.

Таблица 3.1. Числовые параметры полей

Параметр	Размер числа	Байт	Описание
Байт	0-255	1	Занимает меньше всего места
Целое	От -32,768 до 32,767	2	Целые числа без десятичных знаков
Длинное целое	От -2,147,483,648 до 2,147,483,647	4	Большие целые числа без десятичных знаков
Одинарное с плавающей точкой	Большие числа до семи знаков после запятой	4	Больше, чем длинное целое
Двойное с плавающей точкой	Большие числа до 15 знаков после запятой	8	Больше, чем одинарное
Код репликации	Глобально универсальный идентификатор	16	Создает уникальный идентификатор
Действительное	Числа от -10^28-1 до 10^28-1. До 28 десятичных знаков	12	До 28 десятичных знаков

Вы можете ввести до 27 знаков в левой части десятичного числа с одинарной точностью, испбльзуя стандартное форматирование. Двойная точность дает практически неограниченное число знаков. Скажем так: если вы не ученый и не статистик, вы даже близко не подойдете к числу разрядов, которое обеспечивает двойная точность.

Некоторые могли бы возразить: «А почему бы просто не использовать везде двойную точность?» Они объясняют это тем, что, поступая таким образом, они никогда не выйдут за пределы числа. Единственная проблема, связанная с таким подходом, - это нерациональное использования памяти. Если у вас есть автонумеруемое поле, которое, как заранее известно, не будет содержать значение больше, чем 2 млрд., нет никаких причин использовать двойную точность. В таком случае вполне подойдет тип **Длинное целое.**

Даже если у вас есть избыток лишнего места, нерациональное расходование ресурсов - плохая привычка. Когда ваши базы данных чересчур разрастутся, то выполнение вычислений, запросы, а также их компоновка будут занимать боль-

ше времени. Даже если ограничения по объему вас не заботят, то должна заботить скорость.

Просто помните, что тип целых чисел - это целые числа без десятичных знаков. Если вам не нужны знаки после запятой и ваши числа не очень большие, пожалуйста, используйте целые числа.

Дата/время

Вы можете вводить даты с 1 января 100 года по 31 декабря 9999 года. Access предлагает несколько различных форматов дат, но вы также можете вводить свои собственные форматы. Например, в качестве определенного вами формата вы можете ввести формат `М.Д.ГГГГ`, который будет возвращать дату в виде 01.21.2001. Косая черта, тире, запятые или любые другие разграничители на ваш выбор могут автоматически вставляться с помощью шаблона ввода. Вы можете сортировать эти даты с самой поздней по самую раннюю, если выберете порядок по возрастанию, а также вычислите временной диапазон между двумя периодами.

Чтобы узнать больше о форматировании дат на основе пользовательских настроек, см. «Тестирование и использование пользовательских форматов» далее в этой главе.

Денежный

Вы можете выбрать денежный тип данных, если вы планируете выполнять вычисления над полем, которое содержит числа, в левой части которых не более 15 знаков, а справа от запятой не более четырех знаков. Тогда как одинарная и двойная точность требуют вычислений с плавающей точкой, денежный тип данных использует более быстрые вычисления с фиксированной точкой.

Счетчик

Вы можете указать последовательную нумерацию или случайную нумерацию в целых числах с использованием этого типа данных либо выбрав соответствующий пункт в свойстве **Новое значение** этого поля. Access автоматически вставит значение в это поле для каждой записи. Это гарантирует универсальность поля. Впрочем, если вы удалите одну из последовательных записей, этот тип поля не запомнит и не перенумерует удаленное значение. Это значение будет просто отсутствовать.

Да/нет [5]

Этот тип данных ограничивает ввод значениями **да/нет, включено/выключено или истина/ложь.** Он занимает всего лишь один символ. На-

---

[5] В русифицированных версиях Access этот тип поля обычно называется *логическим. - Науч. ред.*

пример, если вы хотите знать, был ли оплачен счет, вы можете создать поле **Оплачено,** которое принимает значения да или нет. Вот вы и пришли к полю-переключателю, который может принимать только два значения.

Объект OLE

Вы можете использовать этот тип поля для ввода объектов из других программ, таких как изображения, фотографии, рисунки, диаграммы, сообщения голосовой почты, звуковые файловы, электронные таблицы и электронные документы. Если у вас есть таблица работников и вы хотите вставить туда фотографии всех работников, это как раз подходящий тип данных.

Гиперссылка

Это буквенно-цифровое поле, которое используется как гиперадрес для связи с Интернетом. Впрочем, вы не ограничены вводом только лишь URL (Унифицированный указатель информационного ресурса). Вы можете связываться с адресом электронной почты, документом Microsoft Office, таблицей, запросом, формой, отчетом или со страницей доступа к данным. Чтобы использовать гиперссылочный тип для электронной почты, введите mailto:// с последующим адресом. Например, введите mailto://JohnSmith@hotmail.com.

Мастер подстановок

Этот мастер создает поле подстановок, которое поможет вам, если у вас имеется длинный список кодов. Таким образом, вы сможете сразу увидеть, какое значение соответствует какому коду. Если вы хотите, например, узнать код для конкретного заказчика и находитесь в другой таблице, используйте этот тип для поисказаказчиков.

Чтобы узнать другие способы подстановок значений, см. раздел «Как пользоваться DAO (объект доступа к данным) и ADO(интерфейсы доступа к данным) для управления данными» в гл. 12 «Средства, подсказки и методы Access Visual Basic».

Тестирование и применение пользовательских форматов

Предположим, у вас есть нестандартный формат чисел или дат, который вы хотели бы использовать. Табл. 3.2 и 3.3 познакомят вас с символами, употребляемыми для создания пользовательских форматов чисел и дат.

Таблица 3.2. Форматы дат

Символы	Значение символов
/ (косая черта), - (тире) или . (точка)	Разделительные элементы значений даты

Символы	Значение символов
d	Численное значение дня месяца без добавления нуля для однозначных чисел
dd	Численное значение дня месяца с добавлением нуля для однозначных чисел
ddd	Три первые буквы дня недели (с Sun (Воскресенья) по Sat (Субботу))
dddd	День недели полностью (с Sunday (Воскресенья) по Saturday (Субботу))
w	День недели(1-7)
ww	Неделя года (1-53)
m	Численное значение месяца в году без добавления нуля для однозначных чисел
mm	Численное значение месяца в году с добавлением нуля для однозначных чисел
mmm	Три первые буквы дня месяца (с Jan (Янв) по Dec (Дек))
mmmmm	Полное название месяца (с January (Января) по December (Декабрь))
yy	Последние две цифры года (с 01 по 99)
yyyy	Полное значение года (с 2001 по 9999)

Таблица 3.3. Форматы чисел

Символ	Значение символа
.(точка)	Разделитель целых и десятых
, (запятая)	Разделитель тысяч
0	Число, если число; ничего, если пусто (null); 0, если нуль
#	Число, если число; ничего, если пусто; ничего, если нуль
$	Буквально символ $
%	Процентное значение, умноженное на 100 и дополненное знаком процента

Теперь давайте продемонстрируем способ проверить форматы, прежде чем вводить их в таблицу, форму или отчет. После того как вы попробуете применить некоторые из этих форматов, вы увидите больше смысла в таблицах.

Проверка форматов

В следующем примере используются старые основные условные обозначения. Знак вопроса (?) означает **Вывод.** Вы просто говорите: «Напечатать следующее на экране». В функциях запятые отделяют аргументы.

Аргумент - это значение, передаваемое функции, которое может быть другой функцией, полем, константой, переменной или выражением. Константа - это просто численное или буквенное значение, которое, в отличие от переменной, не изменяется. Переменная действует как контейнер, содержащий изменяющиеся

значения. *Выражение* в Access может содержать комбинацию полей, функций, констант и операторов, которые вместе составляют формулу. Вот пример выражения: [InvoiceAmt] * .08, где [InvoiceAmt] поле, * оператор, .08 константа.

В функции формат(дата, «мм/дд/гг») format(date, "mm/dd/yy") формат является функцией. Дата - это другая функция, использованная в качестве аргумента. Во втором аргументе мм, дд и гг использованы в качестве переменных, в то время как косая черта использована как константа.

1. Находясь в окне **База данных** Access, нажмите Ctrl+G. Это перенесет вас в окно Immediate Visual Basic.
2. Введите ? format(date, "dddd, mmmm dd, yyyy") и нажмите Enter. Для дня святого Патрика 2001 года эта функция должна вернуть субботу 17 марта 2001 года. Ваша дата будет отличаться, но формат должен быть подобен этому[6].
3. Введите ? format(date, "m/d/yyyy") и нажмите Enter. Эта функция должна вернуть 3/17/2001 для дня святого Патрика. Если бы ваша дата была 12/02/2001, функция вернула бы 12/2/2001. Взгляните на табл. 3.2.
4. Введите ? format(date, "dd-mmm-yy") и нажмите Enter. Эта функция должна вернуть 17 марта 01 или нечто, близкое к вашей дате.
5. И наконец, вводите ? format(date, "mm.dd.yyyy.") и нажмите Enter. Эта функция должна вернуть 03.17.2001 для этой даты.
6. Теперь что касается некоторых числовых форматов. Введите ? format(2000.00, "#,##0") и нажмите Enter. Эта функция должна вернуть значение 2,000.
7. Введите ? format(00, "0.00") и нажмите Enter. Эта функция должна вернуть значение 0.00. Заметьте, что первый нуль исчез. Введите ? format(00, "#.##") и нажмите Enter. В чем разница? В последнем случае формат не обозначен, но он дает вам представление о том, что там происходит.
8. Введите ? format(9864, "$#,##0.00") и нажмите Enter. Эта функция должна вернуть значение $9,864.00. Закройте окно Immediate.

Использование окна Immediate для тренировки дало вам несколько преимуществ:

- теперь вы уверенно можете вводить свои форматы, потому что понимаете, как они работают;
- Вы получили общее представление об окне Immediate, которое вы будете использовать далее для проверки функций;
- Вы понимаете, как работает встроенный формат функций Access.

[6] Во всех данных примерах, введя функцию, вы получите дату ваших упражнений в формате, который указывает автор, а не дату дня святого Патрика. - *Науч. ред.*

Построение таблицы в Access 2002

Далее мы шаг за шагом построим эту таблицу с самого начала, с использованием различных средств контроля качества и разных типов данных:

1. Откройте базу данных Claims.mdb в папке AccessByExample.
2. В разделе **Объекты,** выберите пункт **Таблицы;** затем нажмите **Новая,** чтобы открыть диалоговое окно **Новая таблица.**
3. Выберите второй пункт, **Вид проекта,** когда появится следующее окно, а затем щелкните OK, чтобы открыть конструктор таблицы.
4. После того как вы введете ClaimNumber, перейдите с помощью клавиши Tab в столбец **Тип данных** (Data Type). Заметьте, что автоматически появляется стрелка, направленная вниз. Щелкнув на этой стрелке, вы можете выбрать любой тип данных. На этот раз оставьте тип по умолчанию.
5. Нажмите кнопку **Первичный ключ,** чтобы сделать поле первичным ключом, а затем щелкните на **Размер поля** в нижней части окна (или нажмите F6).
6. Установите размер поля равным 14, а затем щелкните на пустом месте в колонке **Название поля** в ClaimNumber. Введите ДП(DOL)(дата потери) в следующее поле.
7. С помощью клавиши tab перейдите в столбец Data Type, но на этот раз начните ввод с буквы д, и вы увидите, как появится **Дата/время.** Подтвердите этот тип данных, перейдя при помощи клавиши Tab в колонку Description (описание) и наберите «Дата потери», как показано на рис. 3.2.

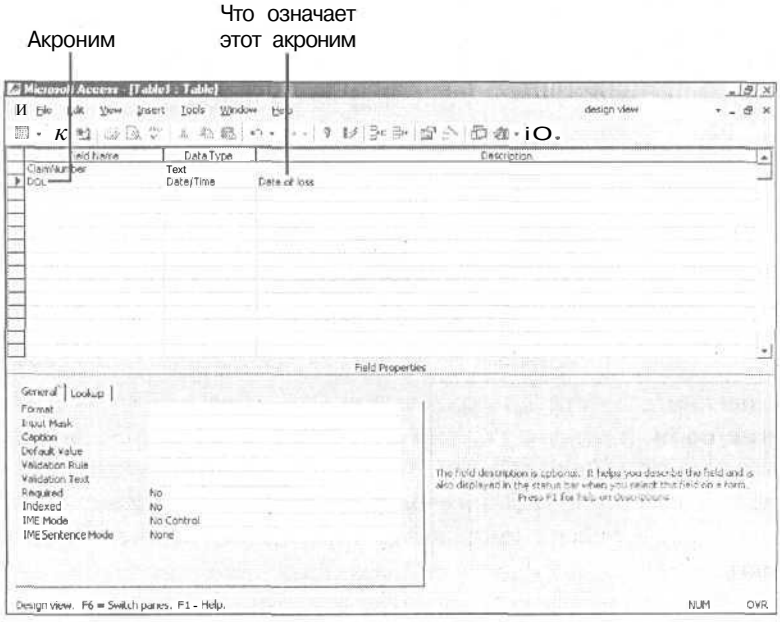


Рис. 3.2. В полях, содержащих акроним, можно ввести в качестве напоминания, что он означает

8. Нажмите F6, чтобы войти в раздел **Свойства поля.** Щелкните по выпадающему списку в поле формата; затем выберите пункт **Краткий формат даты.** Щелкните в поле **Условие на значение** в разделе **Свойства ноля.**

ЗАМЕЧАНИЕ. Условие на значение используется в качестве основного принципа, по которому проверяется содержание поля, когда пользователь его покидает. Это одна из тех вещей, которые отделяются от базы данных. Намного проще предотвратить ввод неправильных данных пользователем на этапе вводе, чем возвращаться и исправлять все ошибки после этого.

9. Щелкните по конструктору выражений(...), а затем введите не > дата() (not > date()) в окне. Нажмите OK.

10. Выберите Условие на текст и наберите Введите дату, которая не превосходит сегодняшнюю дату в диалоговом окне, как показано на рис. 3.3.

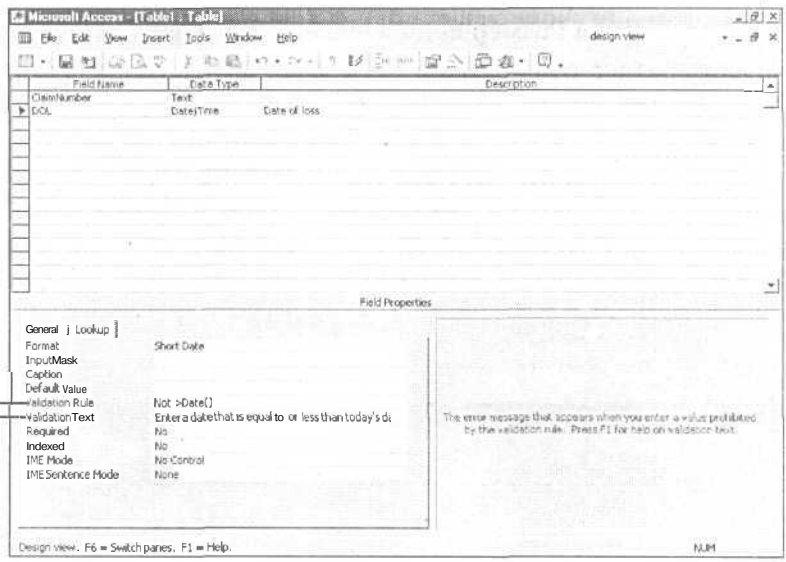


Рис. 3.3. Вы можете использовать Условие на значение для гарантии правильности ввода данных, а Условие на текст для того, чтобы сообщить пользователю, как правильно вводить данные

11. Нажмите F6 или щелкните по пустой строчке под DOL. Введите Претендент в столбце **Название поля,** а затем с помощью курсора перейдите в следующее поле. Заметьте, что по умолчанию автоматически устанавливается **Текстовый тип данных** с длиной поля 50 символов (если вы не установили другой размер поля, используемый по умолчанию, в разделах **Сервис, Опции, Таблицы/Запросы).**

12. В столбце **Название поля** внутри **Претендента** введите Адрес. Затем выберите **Обязательное поле** в разделе **Свойства поля.** (Будет введен размер поля по умолчанию.) Заметьте, что появляется стрелка, направленная вниз. Вы-

берите Да в выпадающем списке. Это означает, что вам обязательно придется вводить информацию в это поле.

ПОДСКАЗКА. Двойной щелчок мышью также переключает состояние да/нет.

13. Введите Город в пустой строчке ниже «Адреса», а затем при помощи курсора перейдите к пустому ряду ниже. Введите Штат, нажмите F6 и поменяйте **Размер поля на 2.**
14. В пустом ряду под **Штатом** введите Индекс, нажмите F6 и поменяйте **Размер поля** на 10. Щелкните в пустую строчку под индексом, введите Статус. Нажмите F6, чтобы перейти в окно **Размер поля** и поменяйте его на 1.
15. Щелкните в поле **Значение** по умолчанию и введите 0 для открытого. Это значение будет автоматически вставляться в новую запись, хотя его можно будет поменять.
16. Введите НСО (SSN - номер социального обеспечения) в пустой строчке под Статусом. Поменяйте его на 11, щелкнув в окне **Размер поля.**
17. В разделе **Свойства поля,** щелкните по маске ввода, а затем по многоточию(...) сбоку. Нажмите Да, когда вас спросят, нужно ли перед этим сохранить таблицу, и сохраните таблицу с именем Claims. Откроется **Создание масок ввода.** Выберите пункт **Номер социального страхования** и попробуйте ввести собственный номер социального страхования, как показано на рис. 3.4.

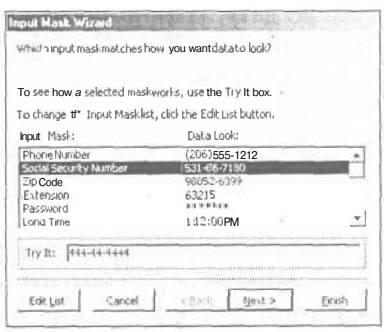


Рис. 3.4. Создание масок ввода позволяет вам проверить маску, находясь в окне Вид таблицы

18. Щелкните **Далее** и завершите работу **Создания масок.**
19. В поле НСО введите ОписаниеПретензии (ClaimDescription). Нажмите Tab и поменяйте тип данных на Memo.
20. Находясь в **Описании претензии,** введите Размер Претензии (ClaimAmount). Нажмите Tab и поменяйте тип данных на **Числовой,** затем щелкните в **Размер поля** и поменяйте его на **Двойное с плавающей точкой.**
21. Под **Размер поля** нажмите **Формат поля** и поменяйте его на **Стандартный.**

22. Под **Формат поля** поменяйте **Число десятичных знаков** на 0, а затем нажмите кнопку **Сохранить.**
23. Нажмите кнопку **Вид.** Это фактически средство быстрого вызова. Вместо того чтобы выходить из вида проекта и открывать таблицу, вам нужно совершить всего одно нажатие, чтобы приготовиться к вводу данных.

Ввод данных

Так как самым важным компонентом любой базы данных являются сами данные, не следует недооценивать важность ввода данных. Если данные проверены должным образом, количество ошибок сводится к минимуму.

В следующем примере вы будете играть роль оператора по вводу данных, чтобы исследовать созданные вами для каждого поля правила проверки. Ваша задача - ввести данные соответственно рис. 3.5, нажимая клавишу tab после каждого ввода.

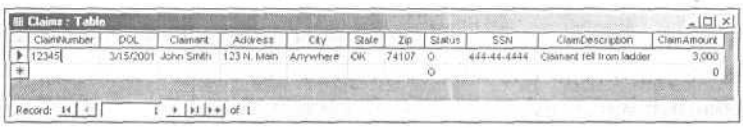


Рис. 3.5. Ввод данных в новую таблицу для исследования проверки условий и типов данных

1. Введите 12345 в поле **Номер претензии,** а затем нажмите Tab.
2. Когда вы дойдете до колонки ДП, введите завтрашнюю дату. Например, если сегодня 3/15/2001, введите 3/16/2001. Нажмите Tab. Вы должны получить сообщение, которое вы ввели в **Условие на текст поля** в окне **Конструктор таблиц.** Вернитесь и введите сегодняшнюю дату.
3. Введите John Smith в поле **Претендент** и нажмите клавишу Tab. В поле Адрес перейдите к следующему полю, не вводя данных. Ничего не происходит. Вернитесь в поле Адрес и введите 123 N. Main.

ЗАМЕЧАНИЕ. Если вы не вводите данные в поле, вы создаете нулевой ввод. Причина, по которой ничего не происходит, когда вы нажимаете Tab, состоит в том, что свойство Обязательное поле включено на уровне таблицы by Jet. Впрочем, вы получите сообщение об ошибке, если попытаетесь выйти из таблицы или перейти к следующей записи.

4. В столбце Город введите Любой (Anywhere), а в столбце **Штат** введите OK. В столбце **Индекс** введите 74107, а в столбце **Статус** нажмите Tab. Там должно автоматически появиться «О», потому что **Статус** - это поле по умолчанию.
5. Введите SSN. Наберите только 444-44-444 и нажмите Tab. Вы должны были получить сообщение о том, что введенное вами значение не подходит. Нажмите OK; затем вернитесь и добавьте последнюю 4.

6. В столбце Описание претензии введите Претендент упал с лестницы и сломал ногу. В этом столбце вы можете печатать и печатать. У вас не закончится место, потому что это поле типа Memo. Нажмите Tab.
7. Введите 3000.00 в столбце **Размер претензии** и щелкните на столбце **Описание претензии.** Два знака после запятой должны были исчезнуть.

ЗАМЕЧАНИЕ. Число между 2999.50 и 2999.99 округляется до 3000. С другой стороны, любое число между 2999.01 и 2999.49 округляется до 2999. Так как это число было форматировано стандартным форматом без десятичных знаков, два нуля после запятой опускаются.

8. Выйдите из таблицы, нажав X в правом верхнем углу заголовка.

ВНИМАНИЕ! Если вы попытаетесь ввести еще одну запись с таким же номером претензии, вы получите сообщение об ошибке. Это происходит потому, что поле **Номер претензии** является первичным ключом.

Объяснения и ограничения

Зачем нужно налагать столько ограничений на только что созданную вами таблицу? Вы хотите избавиться от ошибок в данных, насколько это возможно. Вы обнаружите, что в Access есть средства проверки данных после ввода, но почему бы не проверять их в процессе ввода? Пользователи не получают никаких надоедливых сообщений до тех пор, пока они не введут неправильные данные. Если текущий месяц март, то достаточно легко в качестве номера месяца нечаянно ввести 4 вместо 3. Access не позволит осуществить такой ввод с теми правилами проверки, которые вы установили. Об этой ошибке будет сообщено пользователю, и он сможет продолжить ввод, когда ошибка будет исправлена.

Хотя нельзя до конца избавиться от ошибок человека, вы можете свести к минимуму шансы появления этих ошибок. Когда происходит ввод больших объемов информации, скорость ввода также можно увеличить. Например, если оператор по вводу данных вводит номера социального страхования, то автоматическая вставка разделительных элементов в поле ввода может увеличить скорость набора. Даже если это увеличит скорость всего лишь на долю секунды для каждой записи, эти доли складываются за долгое время набора.

Таблица - это основа

Возвращаясь к первоначальному вопросу, выясним, почему же таблица все-таки душа базы данных. Потому что таблица - это основа всего в базе данных. Запросы, формы, отчеты, макросы и модули - все крутится вокруг таблицы. Точно так же как таблица должна иметь отношение к одному предмету, база данных должна иметь отношение к одной теме. Например, в пределах базы данных о клиентах у вас может быть таблица **Работники,** в которой говорится, какие работники работают с какими клиентами. Но вы также можете включить таблицу **Проекты.** Тема базы данных по-прежнему клиенты, но в этой теме есть

много аспектов. Вы также можете смотреть на это с той точки зрения, что база данных имеет отношение к одной главной теме, а таблицы, которые в нее входят, относятся к ее аспектам.

В Access добавлено еще одно средство для дальнейшего описания вашей базы данных. Теперь вы можете объединять в группы ваши таблицы, отчеты и формы, относящиеся к одной и той же или *к* похожим категориям. Хотя поле - это тоже категория, это более широкая категория. Точно так же как поле является категорией общепринятых значений, группа Access является категорией общепринятых объектов. Например, вы можете объединить все отчеты, связанные со счетами, в одну группу или все таблицы, связанные с проектами, в другую. Вы можете сочетать или согласовывать таблицы, отчеты, формы и модули в группах. Вы определяете общность этих объектов.

Все это сводится к организации и распределению информации по категориям. Таблица в данном случае - лучшее средство. Но таблицы также должны быть организованы. Формы в основном помогают вам с вводом информации, а отчеты помогают с выводом информации,

Что дальше?

В следующей главе вы узнаете, как с наибольшей выгодой использовать таблицы при помощи запросов. Видите ли, база данных связана не только с организацией информации. Она также связана с проверкой, управлением, сравнением, подведением итогов и прогнозированием информации. Запросы - это мощное средство, которое позволит вам все это осуществить.

4
Использование возможностей запросов

В гл. 3 вы узнали, как создать таблицу и подготовить ее к вводу данных. Настоящая глава посвящена тому, почему же запросы так бесценны, когда дело доходит до обновления и управления данными.

В этой главе вы узнаете, как

- создавать запросы, используя сетку запросов Access;
- использовать типы запросов;
- использовать язык SQL;
- сортировать и группировать;
- использовать шаблоны.

Запросы задают вопросы

В предыдущей главе вы уяснили, что таблица - это душа базы данных. Хорошее планирование поможет вам организовать данные вашей таблицы. Но организация - это не единственная задача базы данных. Вам может понадобиться сравнить таблицу за прошлый год с таблицей за текущий год. Или, возможно, вы захотите обновить данные одного поля по всей таблице. Или, возможно, вам понадобится подсчитать налог с продаж по всем счетам, имеющимся в таблице. И этот список можно продолжить. Иначе говоря, вам нужно средство, которое способно выполнять проверку и управление данными множеством различных способов.

Что такое *запрос*? Это просто программа, которая задает вопрос, подходящий к табличным данным. Вы можете считать запрос описанием записей, которые вам нужно получить из табличных данных. Это описание обычно включает *признак* (признаки), который является условием (условиями), определяющим выбор записей. Это условие в запросе принимает форму выражения, которая описывалась в гл. 3. Сколько было исков в штате Миссури на сумму более $5,000 за период между 1 июля 1994-го и 30 июня 1995 года? Это типичный вопрос, который легко можно превратить в условие запроса.

Запросы обращаются к записям

Запросы - это идеальное средство для управления таблицами. Пока что основное внимание уделялось вопросам, касающимся полей и таблиц. А что насчет записей? Благодаря запросам записи выходят на передний план. Таблица - это набор записей, поскольку запись - это набор полей. Возможно, вам не понадобиться отображать все записи или поля конкретной таблицы. С помощью запроса вы можете выбрать, какие поля и записи нужно отображать.

Запросы производят *подмножества*, которые, по сути, есть записи, удовлетворяющие условиям запроса. Таблица - это набор записей; подмножество - это

набор элементов набора записей. Возможно, вам не нужно отображать в таблице все штаты. Что если вы хотите рассмотреть только записи штата Техас? Или если вы хотите отобразить итоговые записи для всех штатов? Запросы эффективно выполняют эти требования.

Подмножества отображаются в виде *таблиц,* которые выглядят точно так же, как таблицы данных. Однако есть одно важное различие: таблицы запросов являются временными, а таблицы данных являются постоянными. Таблицы данных представляют данные, хранимые на диске или на подобном ему запоминающем устройстве. Временный характер таблиц запросов работает в вашу пользу, потому что вы можете совместить две или более таблицы в запросе, тем самым получив большую таблицу, которая не хранится на диске и, таким образом, не занимает места. Она может быть приведена к виду отчета или формы.

Сетказапросов

Строение сетки запросов похоже на строение обычной таблицы. Вы можете выделять и перемещать поля в этой сетке, пользуясь теми же методами, которые применяются и для обычных таблиц. Но для начала вы должны выбрать, какие таблицы должны участвовать в запросе. Оттуда вам нужно щелкнуть два раза или перетащить нужные вам поля в сетку. Порядок выбранных вами полей должен быть такой же, как и у отображаемых результатов. В отличие от простой таблицы вы не видите отдельных записей, но имеете возможность описать, какие именно поля или записи вы хотите получить.

Реализовать модель QBE (запрос на примере) намного проще, чем писать напрямую код SQL. Писать код SQL вручную совсем не обязательно, потому что QBE создает код SQL за вас. В окне конструктора вы переходите с помощью курсора к соответствующим полям, вводите образец **Условия отбора** и щелкаете по значку Run(Запуск), чтобы получить результат. На рис. 4.1 показан пример условия отбора QBE, готового к запуску.

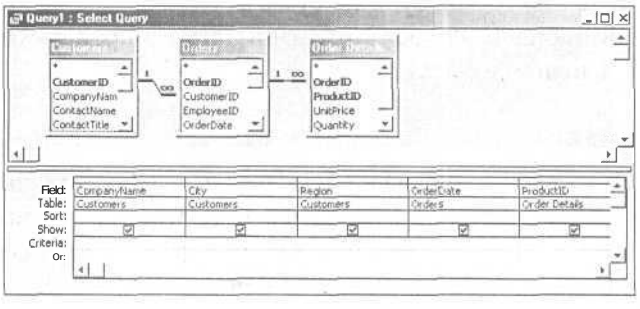


Рис. 4.1. Чтобы задать ваши условия отбора в окне конструктора запроса, воспользуйтесь запросом на примере

О типах запросов

Хотя Access показывает шесть типов запросов в его выпадающем меню, вы можете разделить их по трем категориям: запросы на выборку, запросы действия и перекрестные запросы (запросы на перекрестные табличные данные). Запросы на выборку создают списки, представленные в виде таблиц. Эти таблицы могут быть (а могут и не быть) обновляемыми в зависимости от ряда причин. Запрос на выборку сам по себе не меняет данные в таблице. Он лишь позволяет вам рассмотреть с различных точек зрения и, возможно, подредактировать данные в списке (после запуска запроса) в зависимости от ситуации. Вы можете поместить результаты запроса в форму или отчет. Это наиболее часто используемый тип запроса.

Более детально об обновлении таблиц рассказано в разделе «О динамических множествах и таблицах, лежащих в их основе» далее в этой главе.

Запрос действия, наоборот, меняет набор записей с помощью всего одной операции. Существует четыре типа запросов действия: на создание таблицы, с обновлением, с добавлением, с удалением. Запрос на создание таблицы создает новую таблицу из одной или более таблиц в зависимости от параметров запроса. Запрос на обновление изменяет данные в полях таблиц. Запрос на добавление копирует (добавляет) записи из одной таблицы в другую. Запрос на удаление удаляет записи из таблицы в зависимости от условия отбора.

Перекрестный запрос можно себе представить как способ взглянуть на таблицу с разных точек зрения. Он выполняет сложные вычисления со значениями полей базы данных, используя одно или более полей, в качестве заголовков строк и данные одного поля, в качестве заголовков столбцов.

Типы запросов

Какой же тип запроса следует использовать? Все зависит от того, что вы хотите сделать. Вам следует быть более осторожными с запросами действия, поскольку они могут менять большие объемы информации практически без усилий. И отменить эти изменения невозможно. На самом деле перед запуском запроса действия Access выводит сообщение, предупреждающее, что данные невозможно восстановить. Но если вам требуется лишь задать вопросы и распечатать данные, то используйте запрос на выборку. Давайте рассмотрим различные типы запросов более подробно.

Запрос на выборку

В основе этого запроса лежит оператор SQL. Оператор SELECT в SQL, который запускает запрос на выборку, является движущей силой SQL. Слово «выборка» используется потому, что запросы выбирают записи. Раз уж мы на этом остановились, давайте заглянем внутрь запроса на выборку, чтобы посмотреть на SQL-код, который им управляет. Следующие действия познакомят вас как

с сеткой запросов, так и с SQL. Вы можете представить их себе как два способа просмотра запроса перед тем, как его запустить.

1. В папке AccessByExample откройте базу данных Claims.
2. В меню **Объекты** (Objects) в окне базы данных выберите опцию **Запросы** (Queries) для просмотра списка запросов, как показано на рис. 4.2.
3. Нажмите кнопку **Новый** (New), чтобы открыть диалоговое окно **Новый запрос** (New Query), как показано на рис. 4.3.
4. Убедитесь, что опция **Конструктор** (Design) выбрана, а затем нажмите OK. Откроется диалоговое окно **Добавление таблицы** (Show Table), как показано на рис. 4.4.

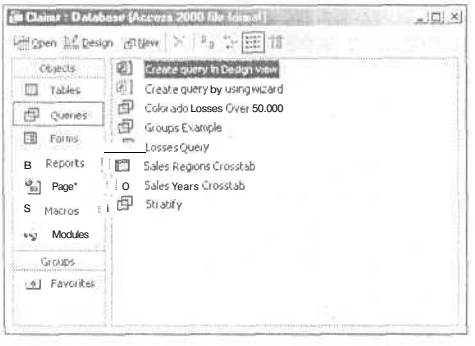


Рис. 4.2. Когда вы выберете Запросы (Queries) в меню Объекты (Objects) в окне базы данных, вы увидите список имеющихся запросов

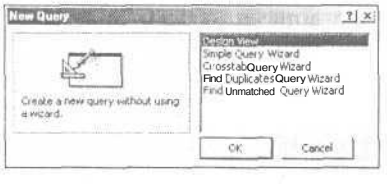


Рис. 4.3. Для того чтобы начать, вы также можете выбрать опцию Конструктор (Design view) из списка опций в диалоговом окне Новый запрос (New Query) или выбрать мастер из списка

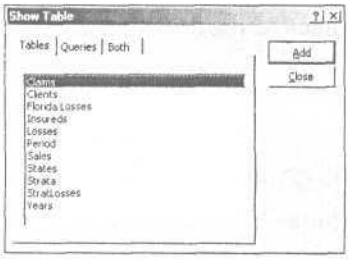


Рис. 4.4. Диалоговое окно Добавление таблицы (Show Table) выдает список таблиц и запросов, которые могут быть добавлены в окно Конструктор (Design)

5. Если нужно, выберите закладку **Таблицы** (Tables). Два раза щелкните по таблице Losses [7], нажмите кнопку **Закрыть** (Close), для закрытия диалогового окна Show Table и выведите таблицу Losses (рис. 4.5).

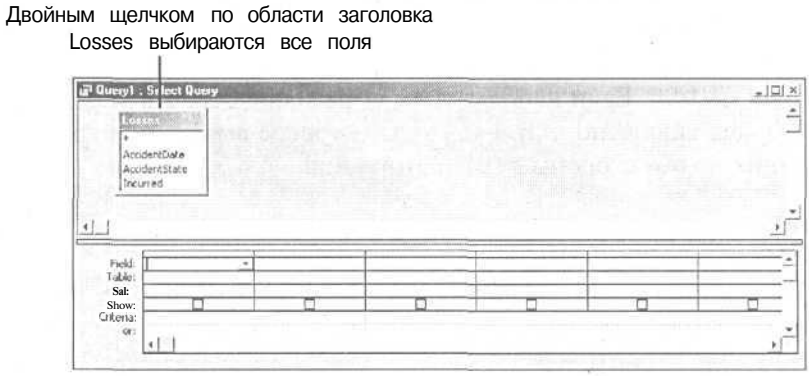


Рис. 4.5. После того как вы добавили таблицу в окно конструктора запроса, вы можете выбрать любое поле, чтобы добавить его в сетку создания запроса

6. Два раза щелкните по строке заголовка таблицы Losses для того, чтобы выбрать все поля.
7. Нажмите и, не отпуская, перетащите выбранные в таблице поля, как показано на рис. 4.6, в сетку запросов.

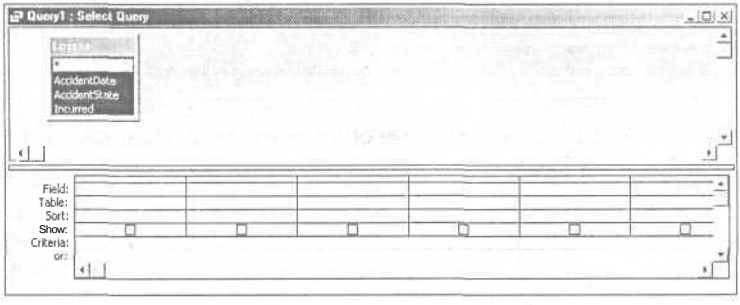


Рис. 4.6. Когда поля выделены, перетащите их в сетку запросов

8. В строке **Условия отбора** (Criteria) поля AccidentDate наберите >=7/1/97 AND <7/1/98 для того, чтобы отобразить записи, значения дат которых больше или равны 7/1/97 или меньше 7/1/98. Заметьте, что значки «решетки» (#), требуемые Access для дат, подставляются автоматически после того, как вы переходите к другому полю.

ПРЕДОСТЕРЕЖЕНИЕ. Хотя значки # вставляются автоматически после введения в сетку запросов даты или выражения с датой в качестве **условия от-**

[7] Название таблицы буквально означает Потери, или Убытки. - *Науч. ред.*

бора, помните, что сетка запросов - это единственное место в Access, в котором значки «решетки» подставляются автоматически. Поэтому стоит привыкнуть к тому, чтобы вводит эти значки вручную.

Теперь у вас есть условие отбора по интервалу времени, как показано на рис. 4.7. Любые даты, попадающие в этот интервал от 7/1/97 до 6/30/98, будут включены. Тем не менее дата 7/1/98 включена не будет, так как в Условии отбора указано <, т. *б.* меньше 7/1/98. Если использовать выражение between 7/1/97 and 6/30/98, то 6/30/98 будет включено, потому что это строгое выражение граничных значений, стоящих по обе стороны от оператора And.

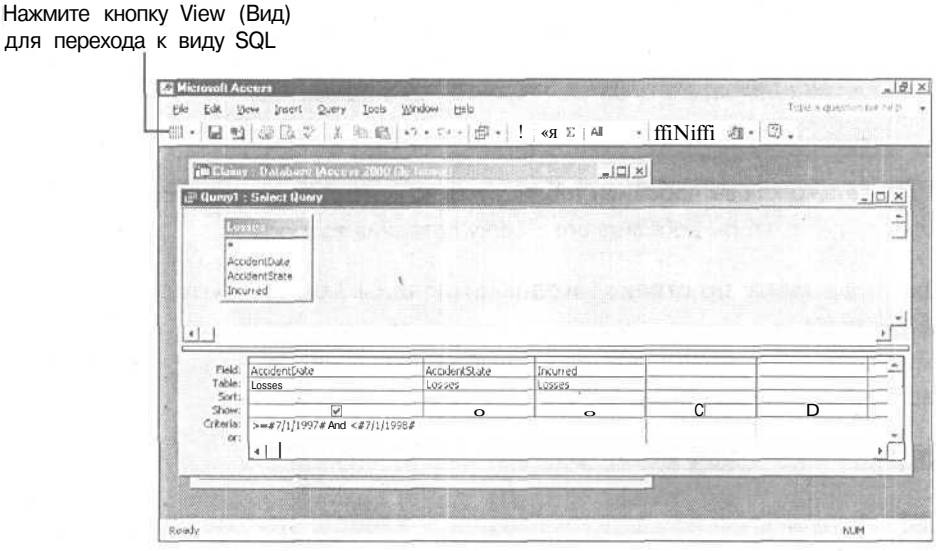


Рис. 4.7. Вы можете ввести выражение с датами, которые определяют те записи, которые приходятся на этот временной интервал между двумя датами и которые будут извлечены

Чтобы просмотреть запрос в виде SQL, выберите View, а затем SQL View. Обратите внимание на временной интервал в строке **Условия отбора,** как показано на рис. 4.8.

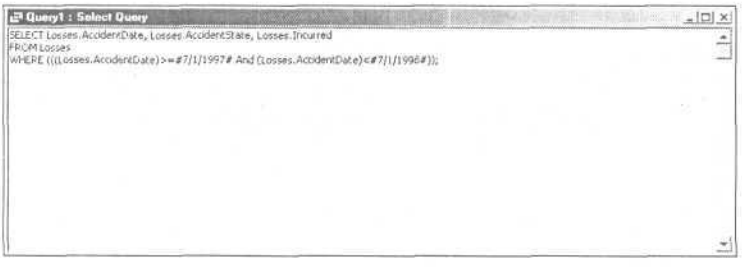


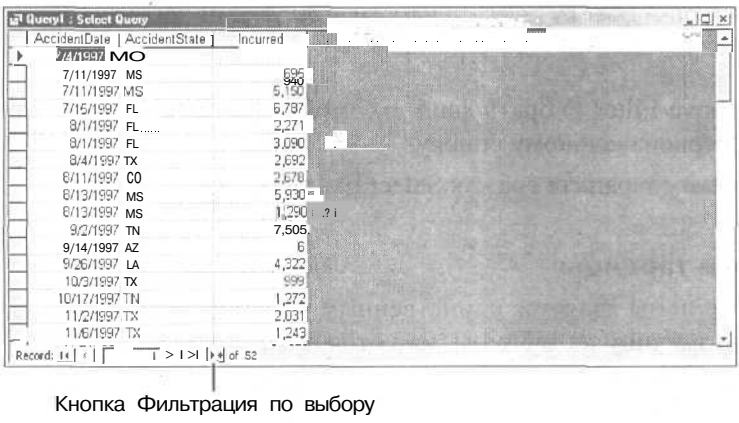
Рис. 4.8. Представление запроса по интервалу времени в виде SQL показывает SQL-выражение в окне просмотра конструктора запроса (Design view)

Теперь вы можете взглянуть на результаты запроса, запустив его. Чтобы научиться управлять данными после запуска запроса, выполните следующее:

1. Перейдите обратно в окно конструктора запроса, щелкнув мышью по View, Design.

ПОДСКАЗКА. Вы также можете запустить запрос из окна его просмотра. Например, вы можете щелкнуть по Run или Datasheet View (Просмотр таблицы) в меню View, чтобы посмотреть на результаты запроса, независимо от того, находитесь ли вы в окне просмотра проектирования запроса или в окне просмотра в виде SQL.

2. Нажмите кнопку Run для запуска запроса. Вы должны увидеть результаты запроса такие, как на рис. 4.9. Это именно тот список, о котором говорилось ранее и который был показан в окне просмотра таблицы (Datasheet view).



Кнопка Фильтрация по выбору

Рис. 4.9. Вы можете изменить параметры просмотра данных даже после того, как запрос был запущен

Допустим, что вы хотите увидеть только убытки в Колорадо. Где бы вы ни встретили повторяющиеся «схожие» значения в поле в окне просмотра таблицы (как в таблице, так и в результатах запроса), вы можете *отфильтровать* эти значения, убирая таким образом все, кроме того, что вы хотите увидеть. Это очень удобный вариант для поиска значений. Вместо того чтобы постоянно нажимать кнопку Find Next (Искать далее), просто отфильтруйте значения, которые вы хотите увидеть, будь вашей целью их редактирование или всего лишь просмотр. Последующее применение фильтра к результатам запроса сужает область поиска записей.

Следующие действия показывают простоту применения промежуточного фильтра, с помощью которого вы можете сконцентрировать поиск только по одному штату.

1. Щелкните по любому значению CO в поле AccidentState (обратитесь к рис. 4.9, если требуется).

2. Нажмите кнопку Filter By Selection (Фильтровать по выбранному). Заметьте, что результаты отфильтрованы (как показано на рис. 4.10), что показывается внизу окна.

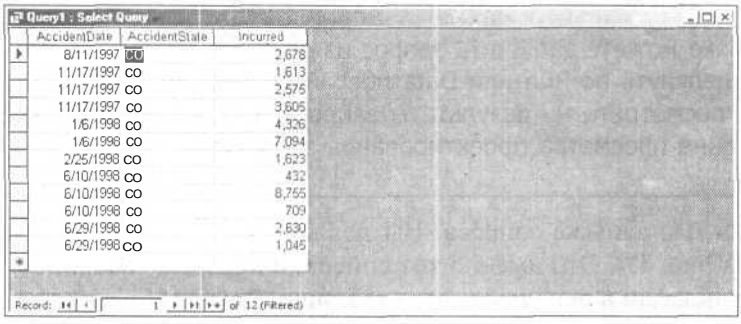


Рис. 4.10. После нажатия кнопки Filter By Selection в таблице вы увидите только записи с Колорадо

3. Нажмите кнопку Remove Filter (Убрать фильтр), чтобы отказаться от фильтрации и вернуться к первоначальному списку.
4. Закройте запрос, а затем сохраните его как Select Example.

Запрос на создание таблицы

Запрос на создание таблицы является единственным типом запроса в Access, который создает другую таблицу из таблицы, на которой основан запрос. Получаемая таблица обычно меньше исходной. Например, допустим, что ваш начальник подходит к вам и говорит, что ему нужна таблица с убытками, превышающими или равными $10,000, для того, чтобы поместить данные из нее в электронную таблицу для последующего статистического анализа. Ему нужна сортировка по дате (начиная с последних исков), и ему также нужно видеть штат. Понимает он это или нет, но он просит вас создать запрос, который помещает результаты, основанные на его условиях отбора, в таблицу.

Чем меньше таблица, тем быстрее ее обработка. Это может быть очень полезно при работе с большими таблицами. Чтобы сделать запрос на создание таблицы, выполните следующие действия:

1. В меню **Объекты** (Objects) выберите Запросы (Queries).
2. Щелкните два раза по **Создать запрос** (Create query) в окне конструктора, а затем щелкните два раза по таблице Losses(Убытки) в появившемся диалоговом окне **Добавление таблицы** (Show Table). Закройте окно Show Table.
3. Два раза щелкните по строке заголовка таблицы Losses, чтобы выделить все поля.
4. Нажмите, и, не отпуская, перетащите выбранные в таблице поля в сетку запроса.

5. Наведите указатель или нажмите на поле Incurred (Понесенные убытки) в строке **Условия отбора** (Criteria) сетки запроса.
6. Наберите >=10000 (четыре нуля) к качестве условия.
7. В строке сортировки поля AccidentState (Состояние убытков) щелкните по выпадающему меню, а затем выберите Ascending (По возрастанию).
8. Щелкните по выпадающему меню кнопки Query Type (Тип запроса) и выберите Make Table Query (Запрос на создание таблицы), чтобы открыть диалоговое окно **Создание таблицы** (Make Table), как показано на рис. 4.11.

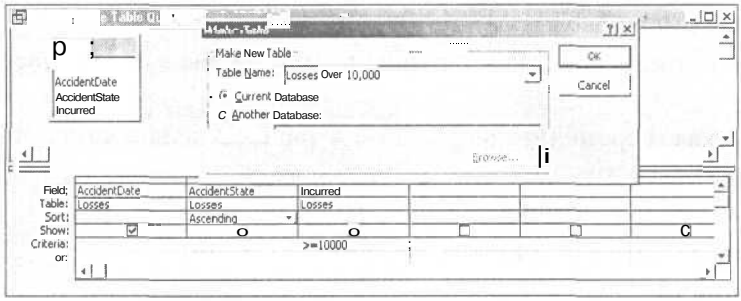


Рис. 4.11. Диалоговое окно запроса на создание таблицы генерирует другую таблицу, основанную на введенных вами в сетку запросов условиях отбора

9. В текстовом окне Table Name (Название таблицы) наберите Losses Over 10,000 (Убытки, превышающие 10, 000), а затем нажмите OK.
10. Запустите запрос. Откроется окно сообщения, которое извещает вас о том, что вы собираетесь вставить 47 строк в новую таблицу, предупреждая, что отменить изменения нельзя.
11. Нажмите Yes для завершения работы запроса; затем закройте его и сохраните как Make Table Example.
12. Выберите **Таблицы** (Tables) и щелкните два раза по Losses Over 10,000, чтобы посмотреть на результаты запроса.

Запрос на добавление

Запрос на добавление копирует записи из одной таблицы в другую, «добавляя» их с конца (за последней записью). Важно помнить, что записи из первоначальной таблицы при этом не удаляются. Чтобы продемонстрировать это, предположим, что ваш начальник сообщает вам, что филиал компании во Флориде продан. Эта информация не нужна в анализе исков, но ему хотелось бы сохранить ее в отдельной таблице, выделив из таблицы Losses (Убытки), для последующего занесения в архив. Для этого потребуется два запроса. Сначала, поместить копии этих записей в другую таблицу, а затем удалить их из первоначальной таблицы для завершения извлечения, что подобно вырезанию и вставке их в другую таблицу.

1. Выберите **Таблицы** (Tables) в меню **Объекты** (Objects); затем правой кнопкой мыши щелкните по таблице Losses и выберите Copy (Копировать).
2. Нажмите кнопку Paste (Вставить) и выберите Structure Only (Только структура) в Paste Options (опции вставки).
3. Назовите таблицу Florida Losses и нажмите OK.
4. В меню **Объекты** (Objects) выберите **Запросы** (Queries).
5. Дважды щелкните по Create Query (Создать запрос) в окне конструктора запроса. Когда появится диалоговое окно **Добавление таблицы** (Show Table), выберите таблицу Losses, дважды щелкнув по ней. Закройте диалоговое окно **Добавление таблицы** (Show Table).
6. Дважды щелкните по строке заголовка таблицы Losses, чтобы выделить все поля.
7. Нажмите и, не отпуская, перетащите выбранные в таблице поля в сетку запроса.
8. Наберите FL в строке **Условия отбора** для поля AccidentState.
9. Щелкните по выпадающему меню кнопки **Тип запроса** (Query Type).
10. Выберите **Добавление** (Append) в качестве вашего типа запроса. Когда появится диалоговое окно **Добавление** (Append), выберите таблицу Florida Losses (Убытки во Флориде) из выпадающего меню Table Name (Имя таблицы) и нажмите OK.

Правильные имена полей автоматически вставляются в строку Append To из таблицы Florida Losses в сетку (рис. 4.12). Это происходит потому, что имена полей подходят. Вы делаете добавление в пустую таблицу, но вы также легко добавить в таблицу с записями. Ни номера полей, ни имена полей в обеих таблицах могут не совпадать. Вы даже можете сделать преобразование некоторых типов данных, но будьте осторожны с этим. Например, вы можете создать другую таблицу, которая имеет такую же структуру, как и таблица Florida Losses, за исключением того, что поле Incurred в ней будет текстовым, а не числовым. Access позволяет добавлять в эту таблицу из таблицы Florida Losses и наоборот. Но учтите, что при этом вы можете потерять десятичные разряды.

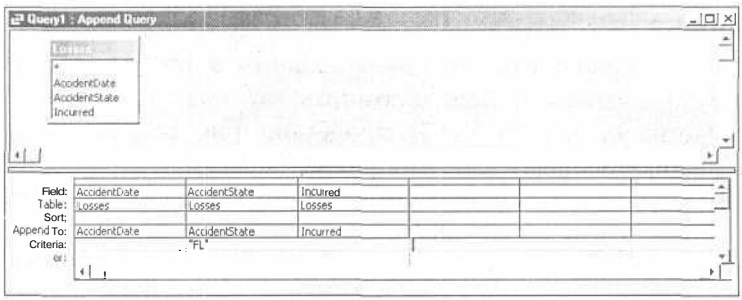


Рис. 4.12. Так как вы ввели FL в качестве условия отбора запроса на добавление, в таблицу Florida Losses добавляются только записи, касающиеся Флориды

Когда вы запустите запрос, вы начнете копирование записей, относящихся к Флориде из таблицы Losses table в только что созданную таблицу Florida Losses. Помните, что на настоящий момент вы скопировали лишь структуру таблицы Losses и назвали ее Florida Losses. Поэтому таблица Florida Losses пуста, но готова к добавлению в нее записей. Это выполняется следующим образом:

1. Запустите запрос. Появится окно сообщения, спрашивающее о том, уверены ли вы, что хотите добавить выбранные строки. Нажмите Yes.
2. Нажмите кнопку Save(Сохранить) и сохраните запрос, как Append Example. Не выходите из запроса.

Теперь, когда вы добавили данные, вам нужно удалить их из первоначальной таблицы. Эти два объединенных запроса: тот, который вы только что запустили (запрос на добавление), и тот, который вы сейчас запустите (запрос на удаление), - образуют скорее операцию переноса, а не копирования данных из одной таблицы в другую.

Запрос на удаление

Запрос на удаление удаляет записи из таблицы в соответствии с условиями. В нашем случае у вас уже есть правильное условие отбора для запроса на удаление; вам осталось лишь поменять тип запроса. Запрос удалит все те записи, которые вы только что скопировали (добавили) в другую таблицу, завершив, таким образом, всю эту операцию.

1. Щелкните по выпадающему меню кнопки Тип запроса (Query Type).
2. Поменяйте тип запроса на **Запрос на удаление** (Delete Query).
3. Запустите запрос. Опять появится окно сообщения, спрашивающее о том, уверены ли вы, что хотите удалить выбранные строки, как показано на рис. 4.13. Нажмите Yes.

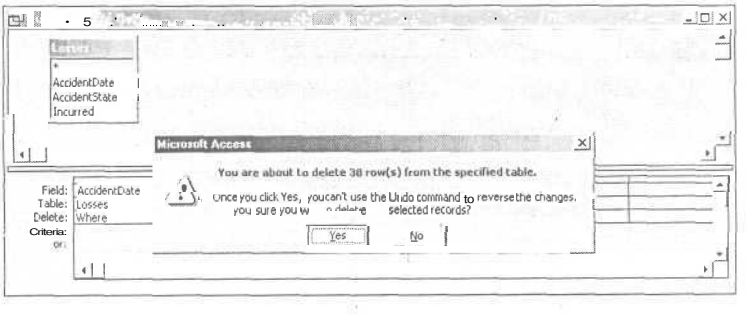


Рис. 4.13. Как и все запросы действия, запрос на удаление выводит сообщение, предупреждающее о том, что изменения отменить нельзя

4. В меню **Файл** (File) выберите Save As (Сохранить как) и сохраните запрос как Delete Example. Нажмите OK и закройте запрос.

Запрос на обновление

Запрос на обновление изменяет *существующие* в исходной таблице данные, в то время как другие запросы действия делают изменения путем либо добавления или удаления, либо создания полностью новой таблицы. Поэтому запросы на обновление являются мощным инструментом, который позволяет вам очень быстро менять огромные объемы данных.

Тем не менее учтите, что если первоначальное состояние таблиц может быть легко восстановлено после применения других запросов действия, то после применения запроса на обновление восстановление таблиц может занять время. И так как это действие не может быть отменено, полезно взять себе за правило копирование таблицы, в которую вы хотите внести изменения, каждый раз, когда вы собираетесь запустить этот запрос. Следующие действия показывают, как создать запрос на обновление с двумя таблицами.

1. Сделайте копию таблицы Losses, назвав ее Losses Backup. Так же, как вы делали в примере на добавление, выберите **Копировать** (Copy) правым щелчком мыши. Нажмите кнопку **Вставить** (Paste), но на этот раз примите значения Structure и Data по умолчанию, когда вы переименовываете таблицу в Losses Backup.
2. Выберите опцию **Запросы** (Queries) в меню **Объекты** (Objects) и дважды щелкните мышью по **Создать запрос** (Create Query) в окне проектирования запроса.
3. На этот раз, когда появится диалоговое окно **Добавление таблицы** (Show Table), дважды щелкните мышью по таблице States и таблице Losses Backup перед тем, как закрыть диалоговое окно **Добавление таблицы** (Show Tables).
4. Как показано на рис. 4.14, выберите и, не отпуская, перетащите поле Code из таблицы States в поле AccidentState в таблицу Losses Backup, а затем отпустите левую кнопку мыши. Так образуется связь между двумя полями, обозначенная графически линией, соединяющей эти два поля в двух данных таблицах.

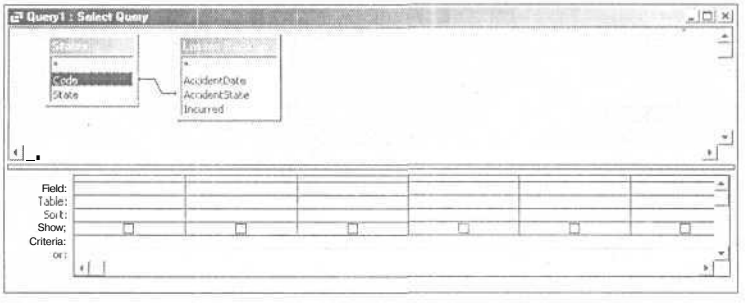


Рис. 4.14. Перетаскивание поля Code из таблицы States в поле AccidentDate из таблицы Losses Backup создает связь между этими таблицами

5. Щелкните два раза по полю AccidentState в таблице Losses Backup, чтобы перенести его в сетку запроса.

6. Выберите Update Query (Запрос на обновление) из выпадающего меню кнопки **Тип запроса** (Query Type). Заметьте, что строка **Обновление** (Update To) вставляется автоматически.
7. В строке **Обновление** (Update To) наберите states! state и нажмите кнопку «стрелка вниз» на клавиатуре. Заметьте, что скобки подставляются автоматически, как показано на рис. 4.15.

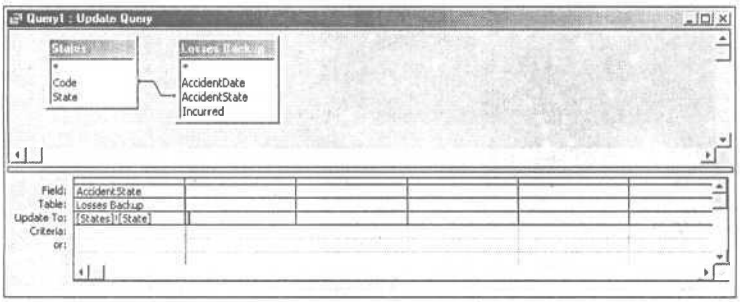


Рис. 4.15. В запросе на обновление может быть использована связь для того, чтобы изменить данные в поле таблицы справа, основываясь на данных в таблице слева

Вы даете запросу задание найти соответствие полю Code из таблицы States полю AccidentState (Сокращения) в таблице Losses. На рис. 4.16 показано, какая информация содержится в таблице States. Это подтверждает то, что имена полей могут не совпадать, но данные в связи должны совпадать обязательно. Вы как бы говорите компьютеру: «Когда коды штатов совпадают, замени полное название штата сокращенным названием». Если код штата есть CO, программа подставит Colorado.

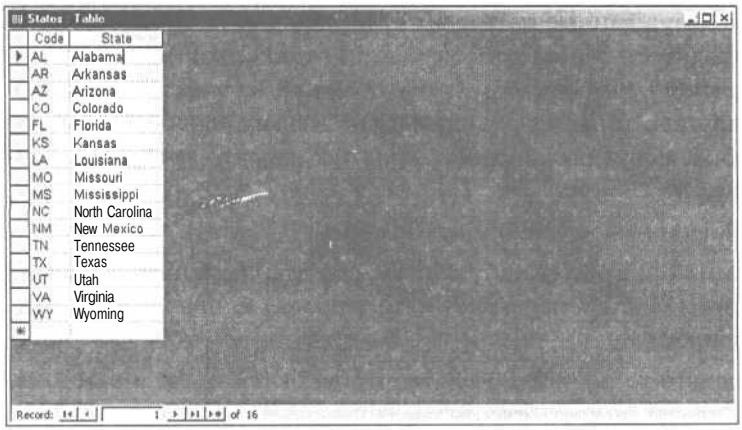


Рис. 4.16. Информация в таблице States, которая используется для обновления информации в таблице Losses Backup

8. Запустите запрос, а затем закройте его, сохранив как Update Example.

9. Выберите **Таблицы** (Tables), а затем дважды щелкните по таблице Losses Backup для просмотра результатов, как показано на рис. 4.17.

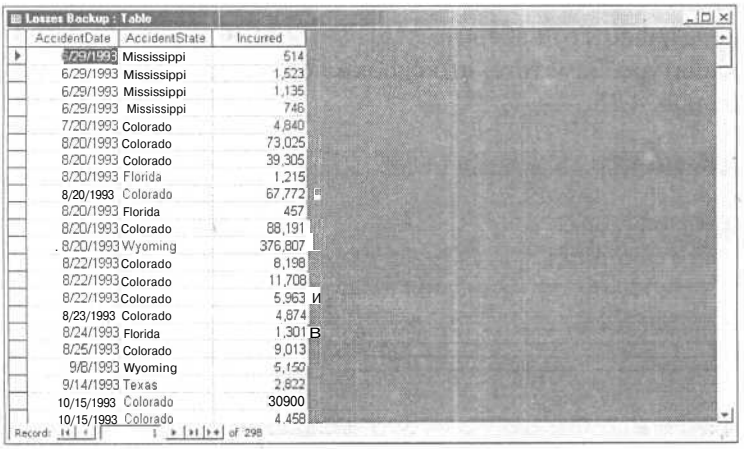


Рис. 4.17. Поле AccidentState в таблице Losses Backup после обновления полными названиями штатов

Перекрестный запрос

Этот запрос уникален по нескольким причинам. Никакой другой запрос не превращает данные поля таблицы в заголовки столбца. Этот запрос является как бы противоположным команде в Excel - Paste Special (Специальная вставка), Transpose (Транспонировать), которая превращает столбцы в строки, а строки в столбцы, оставляя данные нетронутыми, но подгоняя под новые координаты. Перекрестный запрос использует один или более столбцов для заголовков строк, что слева, при этом используя данные из других столбцов для заголовков столбцов, что сверху. Он использует обобщенные функции, такие, как Sum (Суммирование) или Count (Подсчет), выдавая результат, похожий на электронную таблицу. Никакой запрос другого типа так радикально не меняет способ просмотра данных. Строки и столбцы могут быть перестроены так, как это требуется в каждом конкретном случае.

1. В меню **Объекты** (Objects) выберите опцию **Запросы** (Queries).
2. Выберите опцию **Новый запрос** (New), а затем дважды щелкните по опции Crosstab Query Wizard (Мастер перекрестного запроса) в диалоговом окне New Query (Новый запрос).
3. На стр. 1 мастера щелкните по таблице Sale, чтобы выбрать ее, а затем нажмите **Далее** (Next), чтобы открыть стр. *2.*
4. На стр. 2 мастера, выберите поле Region в качестве заголовка строки, щелкнув по нему два раза, а затем нажмите Далее (Next).

5. На стр. 3 выберите поле Year в качестве заголовка столбца (оно уже должно быть выделено) и нажмите **Далее** (Next) (рис. 4.18).

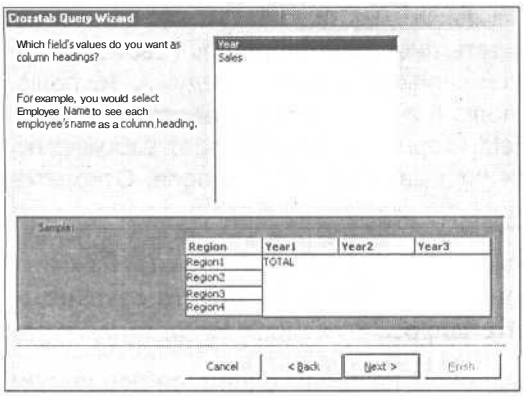


Рис. 4.18. С помощью мастера перекрестного запроса легко выбирать заголовки столбцов

6. На стр. 4 поле Sales уже выделено, но вы выберите опцию Sum (Суммирование) в меню Functions (Функции), что справа, как показано на рис. 4.19, а затем нажмите **Далее** (Next).

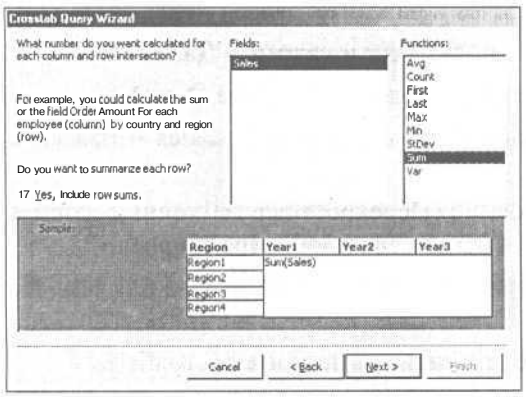


Рис. 4.19. Выбрав опцию Sum для поля Sales, вы производите общее суммирование

7. Примите установки по умолчанию в View Query (Просмотр запроса), но уберите подчеркивание имени запроса, подставив Years, чтобы выглядело, как Sales Years Crosstab (Перекрестные табличные данные **Продажи/годы),** а затем нажмите Finish(Готово), чтобы посмотреть, что получилось.
8. Нажмите View (Вид) (голубой треугольник), чтобы перейти в окно просмотра проектирования запроса. В строке **Поле** (Field) в сетке выберите Sales, а затем щелкните правой кнопкой мыши и выберите пункт Properties (Свойства). В поле для форматирования наберите #,##0. Закройте лист Field Properties (Свойства поля)

9. Щелкните правой кнопкой мыши по полю строки столбца Total of Sales (Общая сумма продаж) и выберите Properties. В поля для. форматирования наберите #, ##0.

ПОДСКАЗКА. Вместо того чтоб закрывать лист Field Properties (Свойства поля) и одно поле, а затем щелкать правой кнопкой мыши по следующему полю, вы можете щелкнуть по следующему полю в сетке, которое вы хотите отформатировать (предполагая, что лист Field Properties не закрывает собой сетку запроса), оставляя при этом открытым лист свойств первого поля. Откроется лист свойств следующего поля.

10. Нажмите Run, чтобы посмотреть результаты форматирования; затем нажмите **Сохранить** (Save) (так как вы уже присвоили имя, то вы просто сохраняете отформатированную версию) и закройте запрос.

Хотя следующий перекрестный запрос похож на предыдущий, он использует другие поля, давая при этом совершенно другой результат. Следующие действия помогут вам создать запрос.

1. В меню **Объекты** (Objects) выберите опцию **Запросы** (Queries).
2. Щелкните по опции New, затем по опции Crosstab Query Wizard (Мастер перекрестного запроса) и после нажмите OK.
3. На стр. 1 Мастера, выберите таблицу Sales и нажмите Далее (Next).
4. На стр. 2 дважды щелкните мышью по полю Year и нажмите **Далее** (Next).
5. На стр. 3 оставьте поле Region выделенным и нажмите **Далее** (Next).
6. На стр. 4 выберите опцию Sum в меню Functions для поля Sales и нажмите **Далее** (Next).
7. Назовите запрос как Sales Regions Crosstab (Перекрестные табличные данные **Продажи/области)** и нажмите Finish (Готово) для просмотра запроса.
8. Чтобы отформатировать поля, выполните действия с 8-го по 11-е для запроса Sales Regions Crosstab.
9. Сравните два запроса, обратив особое внимание на заголовки столбцов.
10. Сравните ваши результаты с результатами, изображенными на рис. 4.20.

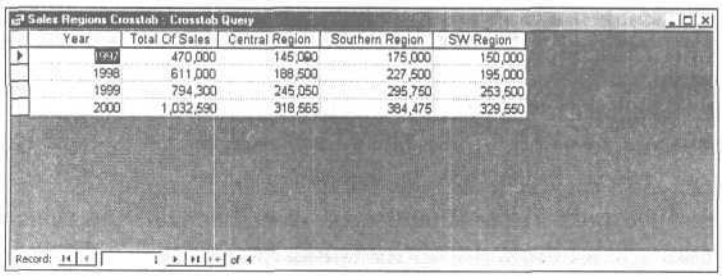


Рис. 4.20. Другой вариант результатов перекрестного запроса, в котором табличные данные превращаются в заголовки столбцов

Введение в SQL

Так как вы уже ознакомились с тем, что можно сделать с помощью SQL, давайте рассмотрим этот вопрос поподробней. SQL является стандартом, для которого любая реляционная система управления базой данных (СУБД) высшего уровня имеет свою версию. Если вы изучите этот стандарт, то будет не трудно начать изучение и использование версий SQL других систем базы данных.

Вы можете создавать запросы прямо в SQL, но если вы хотите, чтобы ваш запрос можно было исполнить, то необходимо следовать строгому синтаксису (правилам языка). Одним из отличительных свойств Access SQL является его способность интерпретировать запросы обоими способами. Это означает, что вы можете набрать SQL-выражения прямо в окне SQL и сразу же видеть их эквивалент в окне проектирования запроса по образцу, до запуска запроса. Более того, вы можете создать запрос по образцу и сразу же увидеть его аналог в SQL.

Вдумайтесь в то, что это значит. Эксперт по SQL может очень быстро изучить Access QBE, создавая SQL-запросы и сверяясь с его аналогом. И наоборот, пользователь, освоивший QBE, может начать изучать и применять SQL, сверяясь с эквивалентными командами SQL.

Другое преимущество SQL состоит в том, что вы можете вырезать и вставлять код SQL в Visual Basic лишь с небольшими изменениями. Это не единственный случай использования SQL в Access, как вы увидите далее, но это есть мощное средство, с помощью которого можно последовательно запускать множество запросов, что, в свою очередь, может быть использовано для автоматизации важных повторяющихся задач.

Начинающий, вероятно, найдет SQL несколько пугающим, но основы его довольно просты. Лучше всего начать с оператора SELECT. В табл. 4.1 приведены основные правила синтаксиса SQL. Курсивом напечатаны дополнительные пояснения только для этого примера, которые не являются настоящими машинными командами.

Таблица 4.1. Объяснение операторов SQL

Оператор	Аналог оператора	Функция
SELECT	*Field* list	Поля, которые нужно включить
FROM	Table list	Таблицы или запросы, из которых нужно выбрать записи
WHERE	Expression	Задание ограничений (условий)
ORDER BY	Field list or Expression	Порядок сортировки

Проще простого, правда? Если вы хотите просмотреть только CustomerName и CustomerPhone columns из таблицы Customer, используйте следующие выражения:

```
SELECT CustomerName, CustomerPhone
FROM Customer
```

Если вы хотите видеть все поля и записи таблицы Claims, введите следующее SQL предложение:

```
SELECT*FROM Claims
```

В каких случаях нужно использовать скобки в запросах

Если в названиях полей у вас есть пробелы или не буквенно-цифровые символы, то вам необходимо заключить названия полей в квадратные скобки, как это сделано в следующем примере:

```
[Customer#], [Customer Name], [Customer Phone]
```

Ниже приведены правила использования запросов.

- Поля, таблицы и ключей сортировки должны быть отделены при помощи запятых.
- В конце выражения должна стоять точка с запятой.
- Если в названиях полей, таблиц или ключах сортировки есть пробелы, то нужно заключить их в скобки.
- Чтобы включить название таблицы, за ним нужно поставить название поля, объединив их при помощи точки (например, Customer.Phone).

Используя SQL, вы определяете то, что хотите получить в итоге. В отличие от сетки проектирования запроса вы прописываете, а не собираете ваш запрос. Следующие действия помогут вам создать запрос SQL.

1. В меню **Запросы** (Queries) щелкните два раза по опции Create query (Создать запрос) в окне конструктора (Design view).
2. Нажмите Close(Закрыть) в диалоговом окне **Добавление таблицы** (Show Table).
3. Нажмите кнопку **Вид** (View) и выберите режим SQL.
4. Поместите курсор после символа «T» в SELECT и наберите

```
SELECT AccidentDate
FROM Losses;
```

5. Запустите запрос. Нажмите кнопку Вид (View) (голубой треугольник), чтобы открыть окно **Конструктор** (Design view) (QBE).
6. Щелкните по выпадающему меню кнопки Dbl (View) и выберите режим SQL View. Отредактируйте операнды SELECT следующим образом:

```
SELECT AccidentDate, Incurred
FROM Losses
WHERE Incurred >=1000
ORDER BY Incurred DESC;
```

7. Запустите запрос. Нажмите кнопку **Вид** (View), а затем закройте не сохраняя.

Зачем нужны логические операторы And и Or в сетке запроса?

Если вы хотите создать запрос cj множеством условий, то вы должны понять разницу между логическими операторами And (И) и Or (ИЛИ). Хотя эти операторы могут соединять более двух условий, простоты ради давайте рассмотрим

только два. Начнем с того, что оператор And отбирает только те записи, для которых *оба* условия в запросе с двумя условиями выполнены. Оператор Or, наоборот, отбирает записи только тогда, когда условия выполнены для одной *или* другой записи в запросе с двумя условиями. Другими словами, в запросе с использованием оператора Or должно быть выполнено хотя бы одно условие.

Вот тут-то и таится небольшой подвох. Скажем, в таблице Losses вы хотите видеть убытки в Миссисипи и Колорадо. Вы можете подумать, что для извлечения этих записей можно набрать лишь CO And MS в столбце AccidentState. С таким условием вы не получите ничего. Почему? Чтобы понять это, рассмотрите процесс по шагам - от записи к записи. SQL проверяет *каждую* запись на предмет выполнения обоих. Естественно, он ничего не найдет, так как нет ни одной записи, в которой бы были *оба* значения Colorado *и* Mississippi в поле AccidentState. Поэтому вам зачастую придется использовать два или более поля для задания условий к оператору And в сетке запроса.

Извлечение двух значений из одного и того же поля

Так как же вам извлечь оба штата в предыдущем примере? Вы можете использовать оператор Or, так как вам нужны оба значения из одного и того же поля. В согласии с правилами действия оператора Or по крайней мере одно из условий было верно. В этом случае оба условия были верны, хотя проверка каждой записи в отдельности не производилась. Итак, нам необходимо использовать оператор Or. В отличие от оператора And оба условия в одной и той же записи могут быть верны, потому что условия находятся в разных полях.

Из предыдущих рассуждений легко догадаться, что вам следует использовать оператор Or в одном-единственном поле сетки запроса, а оператор And - в двух или более различных полях. Тем не менее это верно не всегда. Если придерживаться нашего случая с двумя условиями, то при проектировании запроса вы можете использовать оператор Or в двух полях, если разместить второе условие во втором поле в другой строке. Перед тем как идти дальше, давайте проиллюстрируем это.

Создание запроса с оператором And

Если у вас имеются условия в двух или более полях одной и той же строки условий в сетке запроса, то используйте оператор And. Выполните следующие действия, чтобы создать этот тип запроса.

1. В меню **Запросы** (Queries) два раза щелкните по опции Create query(Создать запрос) в окне конструктора (Design view).
2. Два раза щелкните по таблице Losses, а затем нажмите кнопку Close (Закрыть).
3. Перенесите все поля в сетку запроса, дважды щелкнув мышью по строке заголовка таблицы Losses и перетащив все поля в сетку.
4. Наберите CO в строке **Условия отбора** (Criteria) поля AccidentState.

5. Кнопкой табуляции перейдите к столбцу Incurred и в строке условий наберите >= 50000 (четыре нуля).
6. Нажмите кнопку Run для запуска запроса. Так вы извлечете записи по Колорадо с понесенными убытками, превышающими или равными $50,000. Не выходите из запроса.

Так мы поставили условие And, набрав два условия в двух различных полях. Это-то и составляет операцию логического умножения AND («И»). На рис. 4.21 заметьте, что оба условия находятся в одной и той же строке.

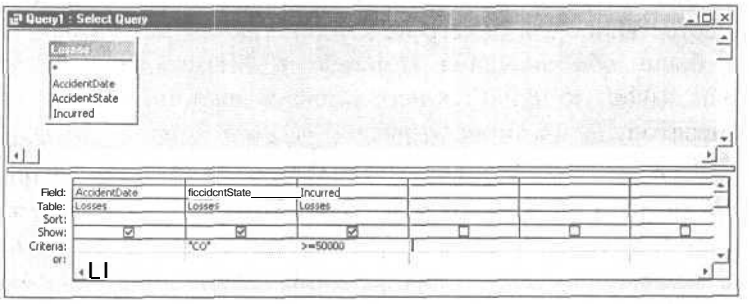


Рис. 4.21. Два условия в двух разных полях образуют условие And

Для проверки нажмите **Вид** (View) и выберите режим SQL. Обратите внимание на оператор AND в задании условия, как показано на рис. 4.22.

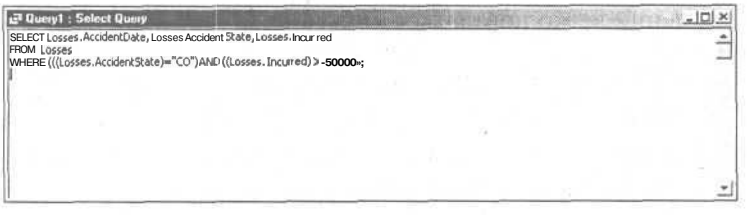


Рис. 4.22. Вы можете убедиться в том, что у вас есть условие And, просмотрев запрос в режиме SQL

Помните, что в запросе с оператором And оба условия должны быть верны для *каждой записи*. Вы легко можете переделать этот запрос в запрос с оператором Or, который работает со многими столбцами.

Следующие действия показывают, как переделать запрос с оператором And в запрос с оператором Or, работающим с несколькими столбцами, изменив лишь одно поле. Это поможет вам понять, как создавать такой запрос.

1. Активизируйте окно конструктора, а затем выберите CO в строке **Условия отбора** (Criteria). Скопируйте с удалением это условие отбора в буфер, нажав Ctrl+X.
2. Переместите курсор вниз на строку Or поля AccidentState и сделайте вставку из буфера, нажав Ctrl+V, как показано на рис. 4.23.

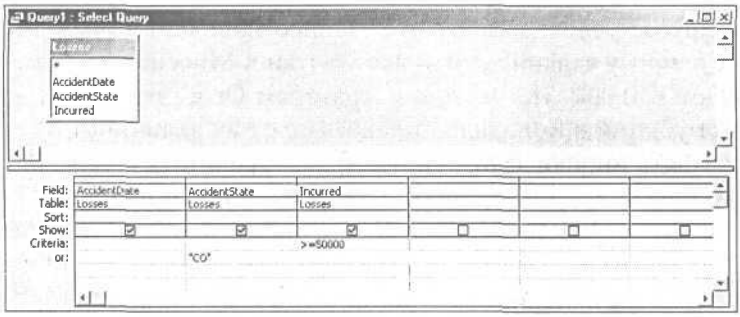


Рис. 4.23. Вы можете создать запрос с оператором Or, используя два поля, внеся одно из условий в строку Или сетки конструктора

3. Запустите запрос. Из запроса не выходите.

Теперь вы направляете запрос на поиск записей, которые или относятся к Колорадо *или* убытки в которых больше либо равны $50,000. Этот случай сильно отличается от предыдущего. Убытки могут иметь место в Колорадо, но могут быть не равными или не превосходить $50,000. И наоборот, убыток может быть равным или превосходить $50,000, но иметь место не в Колорадо. Другой возможный случай может заключаться в том, что убыток может быть равным или превосходить $50,000 и иметь место в Колорадо. По крайней мере одно из условий должно выполняться. Сделайте проверку в режиме SQL, чтобы убедиться в наличии условия с оператором Or. Также возможно создание запроса, в котором будет как условие с оператором And, так и условие с оператором Or. Следующие действия преобразовывают запрос с оператором Or в запрос с операторами And и Or.

1. Активизируйте окно конструктора (Design view) и наведите курсор на пустое поле в строке Условия отбора (Criteria) над условием CO.
2. Наберите MS в качестве условия. Заметьте, что он будет находиться в той же строке, что и условие >=50000, как показано на рис. 4.24.

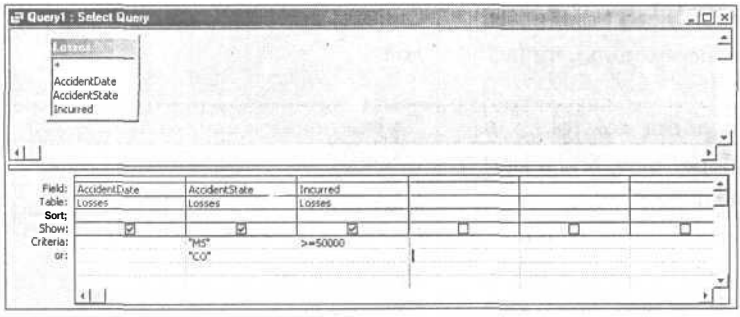


Рис. 4.24. Несколько условий этого запроса образуют запрос с операторами And и Or

3. Запустите запрос.

Так, было добавлено другое условие, делая этот запрос запросом с операторами And и Or. Вы даете запросу задание найти все убытки в Миссисипи, которые равны или превышают $50,000. Но часть с оператором Or в этом условии говорит запросу найти все убытки в Колорадо, независимо от их величины. Сделайте проверку в режиме SQL кнопки Вид (View), чтобы убедиться в наличии условия с операторами And и Or. Закройте запрос, назвав его And Or Example.

Создается впечатление что в Access нет правил без исключений. Операции с And и Or являются тому наглядным примером. Вообще говоря, если вы помещаете два условия в одно и то же поле, но в разные строки сетки, вы создаете условие с оператором Or. Если вы помещаете два условия в одну и ту же строку, но в разные поля, вы создаете условие с оператором And. Тем не менее с помощью подзапросов и функций можно создать условия с оператором And в одном и том же поле, а условия с оператором Or - в разных полях одной и той же строки.

Чтобы узнать больше о подзапросах, см. раздел «Применение подзапросов при использовании возможностей запросов» в гл. 16.

Даже без этих программ вы можете просто поместить несколько многократных объявлений с оператором Or в одной и той же строке. Например, в качестве условия в столбце AccidentState вы могли бы набрать TX Or FL, а не прописывать эти значения в разных строках сетки.

Понятие о подстановочных знаках

Могут возникнуть случаи, когда вам не требуется точного соответствия извлекаемых записей. Представьте, например, что в таблице вам нужно видеть все штаты, которые начинаются с букв «AR». После анализа табл. 4.2 давайте используем подстановочные знаки в примерах.

Таблица 4.2. Примеры символов подстановочных знаков

Подстановочный знак	Пример	Операция
*	tr* выберет слова train (поезд), translate (переводить), try (попытка) и т. д.	Сличает любое количество символов
?	?at выберет bat, cat, rat и т. д.	Сличает любой единичный символ
[]	[bfk] выберет bind, find и kind, но не mind, hind, mind	Сличает любой единичный символ в скобках
!	[!bfk] выберет mind, hind и wind, но не bind, find, kind	Символы, находящиеся в скобках, не сличаются
-	[b-k] выберет bind, find и kind, но не mind, hind, mind	Сличает любой символ, находящийся в указанном диапазоне
#	#92 выберет 192, 892, 492 и т. д.	Сличает любую единичную цифру

Выборка записей с использованием подстановочных знаков

Вероятно, что наиболее удобно использовать подстановочные знаки, когда вы не помните имени отдельного лица или названия компании. Предположим, что вы искали конкретную компанию, но не могли припомнить полного ее названия, хотя знали, что начиналось оно с «United». В качестве условия вы можете просто набрать United*, и запрос извлечет United Trucking, United Rental и т. д. Если вам нужны все слова из четырех букв начинающиеся с «S»,, то в качестве условия вы можете набрать "S???" (с кавычками), так как «S» является одной из четырех букв. Оставшиеся три буквы могут быть любыми. Следующие действия помогут вам создать запрос с использованием подстановочных знаков.

1. В меню **Запросы** (Queries) в окне конструктора (Design view) дважды щелкните по опции **Создать Запрос.**
2. Дважды щелкните по таблице States, а затем выберите **Закрыть** (Close).
3. Поместите оба поля Code и State в сетку запроса, дважды щелкнув по ним.
4. В строке Условия отбора (Criteria) поля Code наберите "?S" (с кавычками).
5. С помощью кнопки табуляции перейдите в поле State и заметьте, что оператор Like вставляется автоматически. Запустите запрос.
   Вы должны увидеть записи о Миссисипи и Канзасе, так как второй буквой в их кодах является "S".
6. Нажмите Вид (View), чтобы вернуться в окно конструктора (Design view).
7. В столбце Code выделите условие и удалите его.
8. С помощью кнопки табуляции перейдите к строке условий поля State, наберите AR* и нажмите кнопку табуляции.
9. Запустите запрос. Вы должны увидеть оба штата - Арканзас и Аризону, как показано на рис. 4.25.
10. Закройте запрос без сохранения.

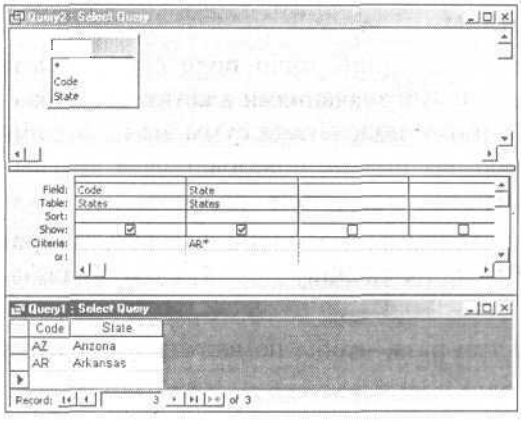


Рис. 4.25. Так как условие, задаваемое подстановочным знаком, начинается с букв «AR», запросом извлекаются оба штата - Арканзас (Arkansas) и Аризона (Arizona)

Понятие о группировке и сортировке

Оператор Group By в запросах Access дает возможность быстро и легко выполнять вычисления сумм, таких, как итоговая сумма, в группах со схожими значениями. Функция DSUM в Excel является аналогом Sum, подсчитываемой с помощью оператора Group By в запросе Access. Однако эта функция громоздка по сравнению с ее простым аналогом в Access. В Excel вы можете сортировать и вычислять промежуточные суммы, но опять-таки в Access эта функция более рационализирована.

Заметьте, что в табл. 4.3 присутствуют два убытка, как для Флориды, так и для Колорадо. Так как поле AccidentState содержит похожие значения, а поле Incurred содержит значения, которые вы хотите добавить в итоговую сумму, поместите оператор Group By в поле AccidentState, а функцию вычисления суммы Sum - в поле Incurred. Вам не нужно группировать значения в поле Incurred, так как в этом примере нет схожих значений.

вам также не нужно пытаться суммировать поля с текстовыми значениями, такие, как AccidentState.

Таблица 4.3.Таблица-образец для оператора

Accident State	Incurred
FL	500
FL	493
CO	209
CO	1,215

Если вы желаете подсчитать все значения, просто поменяйте функцию вычисления суммы Sum на функцию Count. Давайте рассмотрим это немного подробнее на примере.

Созданиезапросассуммированием

Вы можете сосчитать общую сумму, имея лишь одно поле с финансами в сетке проекта. Если вы добавите поля с такими значениями в сетку, вы сможете подсчитать общие суммы. Впрочем, одними подсчетами сумм вы не ограничены. Как показывает следующий пример, вы можете пользоваться и другими обобщенными функциями, такими, как пересчет или взятие среднего.

1. В меню **Запросы** (Queries) дважды щелкните по опции **Создать запрос** (Create Query) в окне конструктора. Выберите таблицу Losses, а затем нажмите **Закрыть** (Close).
2. Дважды щелкните по полю Incurred три раза, чтобы поместить его в сетку проекта.
3. Нажмите кнопку Totals (Суммы), чтобы появилась строка Total.

4. Замените первый оператор Group By на оператор Sum, щелкнув по элементу «стрелка вниз», который появится, когда вы щелкнете по первому экземпляру поля в строке Total и выберете оператор Sum.
5. Выполните предыдущее действие над оставшимися двумя экземплярами поля, но выбирая операторы Count и Average (в таком же порядке) для этих полей, как показано на рис. 4.26.

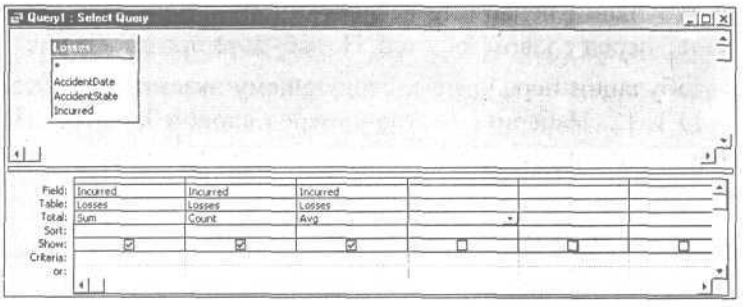
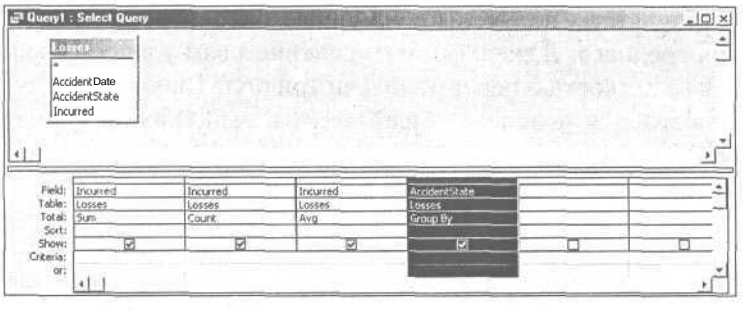


Рис. 4.26. Вы можете вставлять несколько экземпляров одного и того же поля, но с разными обобщенными функциями

6. Запустите запрос, чтобы посмотреть, что получилось. Вы должны увидеть сумму, пересчет и среднее в одной записи. О форматировании пока не беспокойтесь.
7. Нажмите кнопку **Вид** (View), чтобы вернуться в окно Конструктор. Затем дважды щелкните по полю AccidentState, чтобы поместить его в сетку.
8. Поместите курсор чуть выше поля AccidentState и дождитесь, пока он не примет вид стрелки, показывающей вниз. Нажмите левую кнопку мыши, чтобы выделить поле, как показано на рис. 4.27.



Рисунке 4.27. Выделив поле AccidentState, вы можете переместить его в другое место

9. Щелкните и, не отпуская, перетащите поле на первую позицию в таблице. Оставьте оператор Group By. Запустите запрос, а затем вернитесь в окно.
10. Правой кнопкой мыши щелкните по первому экземпляру поля Incurred в строке Field. В меню выберите опцию Properties (Свойства).

11. В Format наберите #,##0, а затем нажмите X для закрытия окна.
12. Если поле Incurred не выделено, нажмите на клавиатуре кнопку HOME, находясь в поле, чтобы поместить курсор перед первым символом названия поля. Если это поле все таки выделено, щелкните мышью перед символом «I» в названии «Incurred».
13. Наберите Sum: перед словом Incurred. Не забудьте про двоеточие.
14. С помощью клавиши табуляции перейдите к следующему экземпляру поля Incurred. Наберите Count: перед словом Incurred. Не забудьте про двоеточие.
15. С помощью клавиши табуляции перейдите к следующему экземпляру и повторите действия пп. 11 и 12. Наберите Average: перед словом Incurred. Не забудьте про двоеточие. Ваш запрос должен выглядеть так, как показано на рис. 4.28.

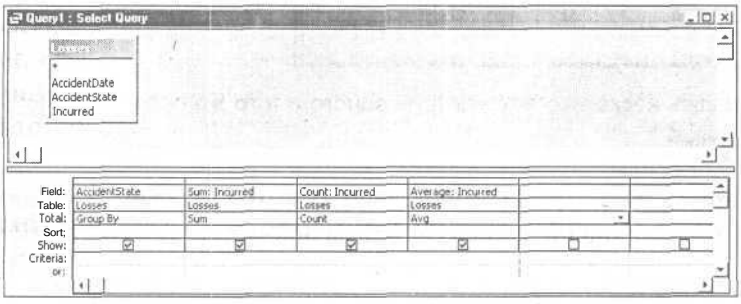


Рис. 4.28. Схожие значения из поля AccidentState становятся основой для групповых вычислений со значениями поля Incurred

16. Запустите запрос, а затем сохраните его как Groups Example.

Обратите внимание на следующие моменты, касающиеся только что созданного вами запроса. Все результаты получены с помощью операторов суммирования, пересчета и взятия среднего. Даже с форматированием вам удалось очень быстро создать запрос. Вы с легкостью переименовали три поля (прямо на ходу). До этого вы создали условия для подсчета общей суммы, выполнения общего пересчета и взятия среднего для одного и того же поля. Вы преобразовали запрос в резюме по каждому набору условий, добавив всего лишь поле.

Сортировка полей в запросах

Вы можете производить сортировку в запросе Access во время и после его запуска. Сортировка результатов в таблицах является очень простой операцией. Вы можете просто нажать кнопку сортировки по возрастанию на инструментальной панели, находясь в выделенном столбце, если вам угодно произвести сортировку по возрастанию в этом столбце. В сетке запроса вы должны щелкнуть по соответствующему полю в строке сортировки и затем выбрать способ сортировки - по возрастанию или убыванию. Затем вы должны запустить за-

прос. Хотя сортировка в сетке запроса не так уж проста, но она обладает большей функциональностью.

Итак, мы уже подошли к обсуждению того, что вы хотите сделать(с помощью сортировки). Если вам нужен простой и быстрый способ сортировки одного или более полей, то используйте метод сортировки в таблицах. Но если у вас есть несмежные поля или два и более поля, для которых выбраны разные порядки сортировки, то вам следует использовать метод сортировки в запросах.

Перед тем как вы проанализируете эти методы сортировки на примере, убедитесь в том, что вы понимаете, что такое сортировка нескольких полей. В таблице 4.4 показан пример многоуровневой сортировки, или сортировки внутри сортировки. Первый уровень, или поле первичной сортировки, - это поле AccidentState. Вы можете сказать, что это значения в этом поле отсортированы в алфавитном порядке, так как записи, начинающиеся с «О», идут перед записями, начинающимися с «Т». Вторичное поле сортировки - это поле Incurred. Хотя записи в поле AccidentState идут по возрастанию, все же значения поля Incurred идут по убыванию. Может быть, вы захотите, чтобы сначала шли самые большие убытки в каждом штате. Так как у вас есть повторяющиеся значения в столбце AccidentState, то в столбце Incurred вы можете произвести субсортировку, или вторичную сортировку.

Таблица 4.4. Сортировка внутри таблицы сортировки

AccidentState	Incurred
OK	5,000
OK	4,000
OK	3,000
TX	7,000
TX	6,000
TX	5,000

Сортировка во время и после работы запроса

Многоуровневая сортировка предназначена для установления порядка, в котором вы хотите видеть ваши записи. Как показывает следующий пример, вторичная сортировка (сортировка внутри сортировки) может иметь порядок, обратный порядку первичной сортировки. Выполните следующие действия, чтобы «на примере» увидеть, как производить сортировку во время и после работы запроса.

1. В меню **Запросы** (Queries) дважды щелкните по опции **Создать запрос** (Create Query) в окне проекта. Выберите таблицу Losses и нажмите кнопку **Закрыть** (Close).
2. Дважды щелкните по полям AccidentState Incurred, чтобы поместить их в сетку. В строке **Сортировка** поля AccidentState выберите опцию Ascending (По возрастанию).

3. В строке **Сортировка** поля Incurred выберите опцию Descending (По убыванию). Теперь у вас есть многоуровневая сортировка, как показано на рис. 4.29.

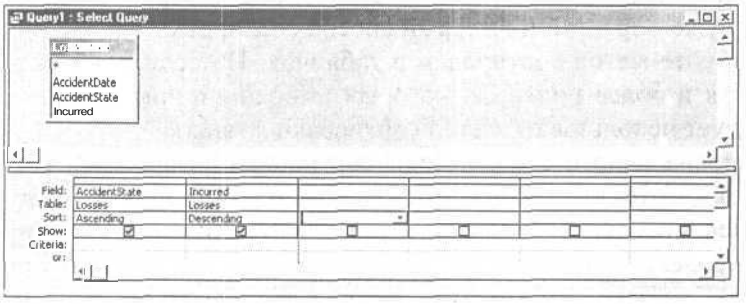


Рис. 4.29. В этом запросе с многоуровневой сортировкой для поля AccidentState выставлен порядок сортировки по возрастанию, а для поля Incurred - по убыванию

4. Запустите запрос.
5. Выделите оба столбца, поместив курсор чуть выше столбца AccidentState. Когда курсор примет вид «стрелка вниз», выделите столбец и, щелкнув левой кнопкой мыши, не отпуская, переместите курсор на столбец Incurred. Если вы все сделали правильно, оба столбца должны быть выделены.
6. Нажмите кнопку сортировки по возрастанию на инструментальной панели, чтобы отсортировать поля. Заметьте, что вторичная сортировка - сортировка по возрастанию, как показано на рис. 4.30.

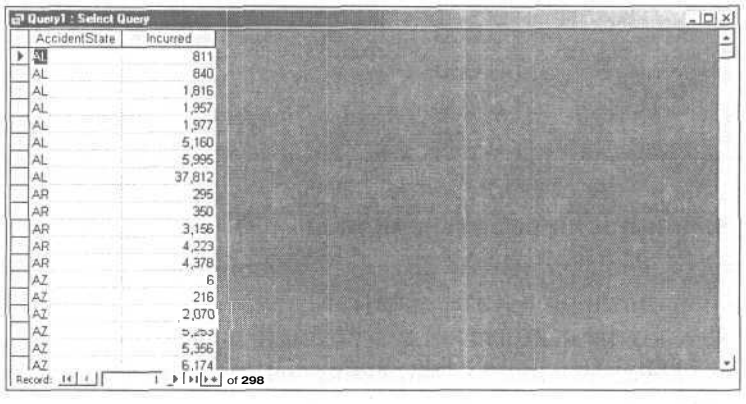


Рис. 4.30. Вы можете изменить результаты сортировки даже после запуска запроса с помощью кнопок сортировки по возрастанию и убыванию, находящихся на инструментальной панели

Оба способа отсортировали оба поля, но второй способ отсортировал оба столбца в порядке возрастания. Если бы в вашей таблице было 10 полей и первое поле, которое надо было бы отсортировать, было в первом столбце, а второе поле в последнем, то был бы смысл задать сортировку в окне конструктора, так

как столбцы *несмежные*. И хотя вы можете перемещать поля в обоих окнах, легче будет использовать окно конструктора, особенно если порядки сортировки разные.

Другим важным моментом является то, что в обоих окнах самый левый столбец - поле первичной сортировки. Это означает, что поля должны быть расположены так, чтобы поля, которые вы хотите отсортировать, были расположены в сетке конструктора, а также в сетке запроса справа налево.

Понимание динамических множеств и таблиц, лежащих в их основе

Как описано выше, запрос на выборку извлекает записи в зависимости от тех условий, которые вы задали в сетке запроса. Эти записи называются *динамическим множеством*. Получающаяся таблица с результатами запроса выглядит так же, как и таблица с исходными данными, но имеет другие характеристики.

Динамическое множество, как это следует из названия, динамично и временно. Таблица, наоборот, статична и постоянна. *Динамическое множество* показывает то, что есть в таблице, когда запрос запущен. Когда запрос закрывается, динамическое множество «испаряется». Динамическое, согласно толковому словарю, имеет тенденцию к изменению. Если данные в таблице изменяются во время исполнения запроса, динамическое множество отображает то, что в ней сейчас находится. Важно помнить то, что, когда вы сохраняете запрос, вы в то же время сохраняете и конструктор запроса, который генерирует динамическое множество, но не само динамическое множество.

Сохраняя проект, а не записи, вы экономите место на жестком диске. Более того, динамическая природа запроса гарантирует то, что записи, принимающие участие в запросе, всегда новые. Эти записи даже можно обновлять, но все эти преимущества не обходятся даром. Динамическое множество не самый быстрый тип записи.

Динамические множества против «моментальных снимков» в борьбе за оптимизацию запроса

Хотя динамическое множество в запросе является типом записи по умолчанию, но «моментальный снимок» быстрее. Например, если вы создаете записи для того, чтобы заполнить список элементов на выборку, то вам следует использовать «моментальный снимок», если вас устраивает атрибут «только для чтения» (необновляемый), присущий этому типу записи. Или, например, вы делаете запрос в огромной таблице с той лишь целью, чтобы переслать результаты в отчет. Здесь вам следует подумать об использовании типа записи - «моментальный снимок». Только убедитесь в том, что вам не нужно редактировать результаты.

О заполнении списка элементов на выбор подробно написано в разделе «The Power of Controls Revisited» гл. 7.

Чтобы поменять тип записи конструктора запроса, откройте окно конструктора, щелкните правой кнопкой мыши по пустому месту над сеткой и выберите опцию Properties (Свойства)[8]. Щелкните по элементу «стрелка вниз» в окне Recordset type (Тип набора записей) и выберите **Динамический набор** (Snapshot).

Что дальше?

Теперь, когда вы посвящены в возможности запросов, у вас есть шанс постичь возможности элементов управления, которые могут использовать возможности запросов. Элементы управления являются компонентами формы. Формы могут использовать динамические множества, взятые из запросов, чтобы производить поиск и вычисления. Главным образом они используются для ввода. Об этом и многом другом вы сможете прочитать в следующей главе.

[8] Кроме способа, указанного автором, можно найти опцию Свойства следующим способом. Опция Свойства находится в окне Конструктор в меню, появляющемся после нажатия команды меню Вид. - *Науч. ред.*

Часть II
Формы для ввода и отчеты для вывода данных

5

Изучаем формы и элементы управления

В гл. 4 книги «Использование возможностей запросов» вы научились создавать и использовать разные виды запросов. Теперь пришло время научиться создавать механизмы для пользовательского ввода, применяющие запросы разных видов. Они называются *формами*.

В этой главе особое внимание уделяется формам, элементам управления и событиям. В частности, мы изучаем:

- различия между формами и отчетами;
- события и как их использовать;
- как использовать помощь мастеров по созданию форм;
- как исследовать и использовать свойства форм;
- что такое элементы управления и как эффективно использовать различные их типы;
- как использовать запросы в формах и элементах управления.

Формы в сравнении с отчетами

Вообще говоря, формы служат для ввода, а отчеты для вывода. Это не значит, что вы не можете делать распечатку, используя форму. Практически нет границ ее разносторонней применимости. Вы можете использовать форму как главную точку отправления вашего приложения. Форма предоставляет возможности меню, позволяет открывать другие файлы, например другие формы и таблицы, распечатывать отчеты и делать множество других вещей. Как сказано выше, вы можете собирать вместе многочисленные таблицы, используя запрос, который вводится в форму.

Форма дает вам широкий спектр функций просмотра информации. Если этого недостаточно, то вы легко можете запрограммировать различные объекты на выполнение рутинных задач. Например, простым щелчком мыши вы можете установить фильтр, просмотреть многочисленные значения, сосчитать многочисленные поля, распечатать текущую запись, выбрать объекты из перечня в меню и т. д.

Просмотр данных в виде таблицы (Datasheet view, Table view) отображает их в виде электронной таблицы. Это прекрасный способ просмотра данных в лю-

бом виде, но что делать, если вы хотите увидеть детали записи с большим числом полей?

При изображении данных в виде таблицы вам необходимо делать прокрутку вправо, пока не увидите искомые поля. В это время поля слева будут исчезать. Вы можете перемещать поля, но это скорее приведет к недоразумениям, нежели к улучшению таблицы.

Напротив, столбчатые формы, сделанные в виде вращающихся картотек (rolodex-style) изображают все поля и записи одновременно до тех пор, пока полей не будет настолько много, что придется переходить на вторую страницу. Но и в этом случае вам необходимо приводить их в порядок. В отдельных случаях форма может напоминать отчет, но все равно между ними есть четкие различия.

Красота формы заключается в способности выводить, управлять, создавать и фактически изменять каждый объект в базе данных. Красота отчета - в его способности динамически представить данные, связанные с запросами и таблицами. Главная функция формы - ввод данных. Главная функция отчета - вывод данных. Оба они сильно зависят от *элементов управления,* которые можно определить так: это любые объекты, например текстовое поле, список, прямоугольник, или командная кнопка, которые вы располагаете в форме. Оба они могут выглядеть как электронная таблица или страница отчета в файле. Они связаны с таблицами и запросами через свойства источника записи. Оба они сортируют, фильтруют данные, обрабатывают многочисленные таблицы, выполняют вычисленияипрограммы.

Так что же делает форму уникальной? В то время как отчет связан с печатью, форма связана с заполнением (набором). Форма может уменьшить число ошибок, сделанных при вводе данных. Она может направлять пользователя вдоль цепочки действий при помощи справочных меню, подсказок и командных кнопок. Например, имеется иногда пропускаемое свойство элемента управления, называемое *текстом подсказки.* Для большей информации об элементах управления (например, о кнопке) поместите курсор мыши рядом с объектом; появится справка, которая включает в себя полную информацию о том, что управление делает или как это активизируется. Когда вы убираете курсор в сторону от объекта, справка исчезает.

Форма - это обычно то место, где происходит программирование большой базы данных. Эти программы - управляемые события. Это означает, что объектно-ориентированное программирование активизируется, когда происходит некоторое событие. Например, событие «Щелчок мыши» происходит, когда вы нажимаете на клавишу. Это выполняется независимо от программы, привязанной к событию.

Определение событий

Если вы работали с формами Access или отчетами, вы активизировали события. Каждый раз, когда вы открываете или закрываете форму или отчет, вы активизируете события On Open и On Close. Событие On Activate происходит, когда форма становится активным окном. В отличие от многих других свойств

объекта свойства события первоначально являются пустыми без заданных по умолчанию значений. Если вы устанавливаете свойство события в макросе, макрос загружается, когда это событие происходит. Скажем, в форме вы имеете кнопку, помеченную Print Record. Если вы щелкаете по этой кнопке, когда форма открыта (не в виде конструктора), событие On Click активизирует и загружает макрос или процедуру, приложенные к событию. Эта процедура печатает текущую запись вашей формы. Если вы хотите составить список выражений Visual Basic, выполняемых когда происходит конкретное событие, вы пишете процедуру для приложения к событию. Не удивительно, что она называется *процедурой обработки события.*

Управляющие события

События также происходят в элементах управления на формах. Это может быть очень полезно для полей подстановки (lookups) и выбираемых списков (pick lists). Например, предположим, что вы имеете подстановочное значение, которое вы только что изменили. Вы можете иметь другую ячейку, которую нужно обновить для соответствия обновленному полю. Например, если покупатель решил купить две вещи вместо одной, поле Extension (которое вычисляется умножением цены на количество) должно измениться, чтобы отобразить новое значение.

Если вы хотите перейти к конкретной записи или полю, когда открывается форма, используйте событие формы On Open. Вы можете использовать это событие во время открытия формы, чтобы показать или скрыть панель инструментов. Помните, что событие On Open не происходит при активизации формы, если она уже открыта. В этой ситуации происходит событие Activate. Если у вас есть записи, которые вы хотите выделить изменяя цвет каждого текстового поля в форме, которая имеет некоторое значение поля, используйте событие On Current. Например, скажем, вы следите за студентами-медиками, которые имеют активный или неактивный статус. Вы можете использовать это событие, чтобы выделить элементы управления на записях каждого неактивного студента в отличие от остальной части студентов. Это лишь несколько примеров того, что можно делать с событиями формы.

Использование мастера при разработке формы

Вы обнаружили, что мастера Access могут быть вашими друзьями. Они легко проводят вас через процедуры, которые иначе могли бы оказаться довольно трудными, сильно экономя ваши силы и время. Это не значит, что мастера - это всегда лучший способ. Но это значит, что освоить мастер - это хорошая идея, чтобы полностью использовать функциональные возможности вашей базы данных.

Суть в том, что мастера могут экономить Ваше время, создавая формы, в особенности если они реляционные (relational). Например, мастер может заниматься внутренним процессом установки реляционной формы типа автоматического размещения подчиненной формы в главной форме и автоматического раз-

мещения выбранных полей в надлежащем месте. Кроме того, вы можете выбрать эстетически приятный стиль формы из списка форматов установки.

Чтобы использовать мастер формы, выполните следующие действия:

1. Откройте базу данных магазина музыки (Music Store) из папки AccessByExample на вашем жестком диске.
2. В окне базы данных, в меню **Объекты,** выберите **Формы,** как показано на рис.5.1.

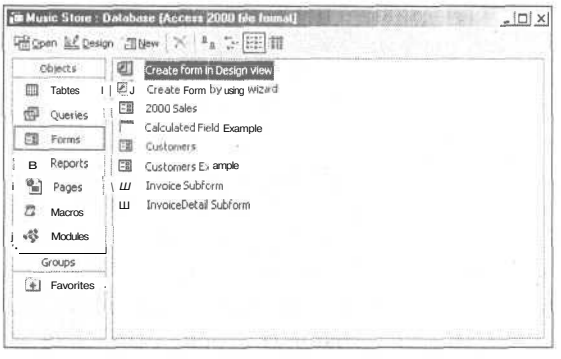


Рис. 5.1. Нажав Формы, вы увидите список форм в окне Базы данных

3. Чтобы открыть мастер, два раза щелкните по опции **Создание формы с помощью мастера** (Create form by using wizard).
4. На стр. 1 мастера, в пункте **Таблицы и запросы,** щелкните кнопку **стрелка вниз** и выберите Customers, как показано на рис. 5.2.

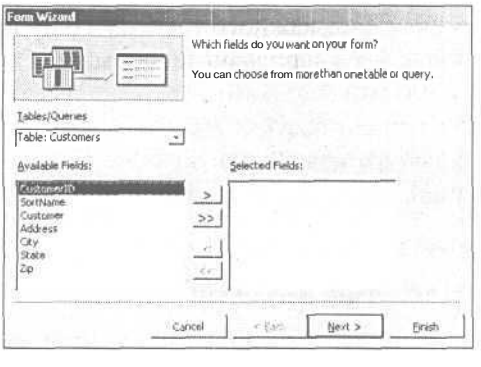


Рис. 5.2. Вы выбираете вашу таблицу или запрос и названия полей из таблицы или запроса на первой странице мастера формы

5. Щелкните кнопку >>, чтобы перенести все поля из блока **Доступные поля** в блок **Выбранные поля.**

ЗАМЕЧАНИЕ. Имеются четыре кнопки между блоком **Доступные поля** и блоком **Выбранные поля,** которым Microsoft официально не дала названия. Эти кнопки имеют следующие значения:
> добавляет выбранное поле,
> > добавляет все поля,
< удаляет выбранное поле,
< < удаляет все поля.

6. Нажмите **Далее,** чтобы открыть следующую страницу мастера.
7. Выберите **В один столбец** как показано на рис. 5.3, а затем щелкните **Далее,** чтобы открыть стр. 3 мастера.

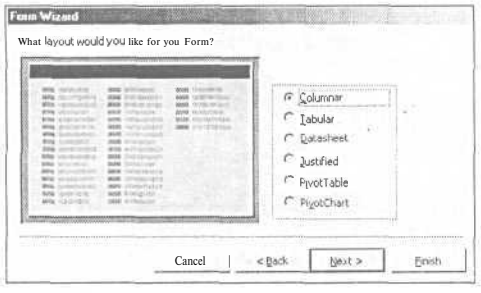


Рис. 5.3. Тип расположения «В один столбец» отображает каждую запись как страницу в форме

8. На стр. 3 примите заданный по умолчанию Стандартный, щелкнув Далее.
9. Наберите Form Customers Example в блоке под Задайте имя формы и нажмите Готово. При этом откроется форма для просмотра и ввода данных, как поясняет переключатель. Оставьте форму открытий для следующего примера.

Это все. Заметьте, что большинство полей размещено вертикально, а не горизонтально, как они были бы представлены в виде электронной таблицы. Пока вы здесь, давайте опробуем несколько способов просмотра записей. Это - комбинации «быстрых клавиш», которые дают простой способ передвижения по набору записей. Далее вы знакомитесь с некоторыми из этих комбинаций, показанных на рис. 5.4.

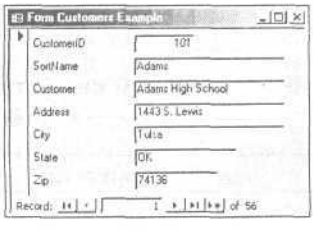


Рис. 5.4. Вы можете использовать быстрые клавиши, чтобы передвигаться по форме, созданной мастером форм

1. В форме, оставленной открытой с последнего упражнения, нажмите Ctrl+End. Заметьте белый блок номера записи внизу формы. Он указывает, что вы находитесь в последней записи. Заметьте также, что вы находитесь в последнем поле.
2. Нажмите Ctrl+Home. Это немедленно переносит вас к первой записи и к первому полю таблицы.
3. Нажмите End. Заметьте, что вы находитесь в первой записи и в последнем поле записи.
4. Нажмите F5. Это перенесет вас к блоку номера записи.
5. Наберите 55 и затем нажмите Enter. Теперь вы готовы сделать запись 55, как показано на рис. 5.5. Оставьте форму открытой для следующего примера.

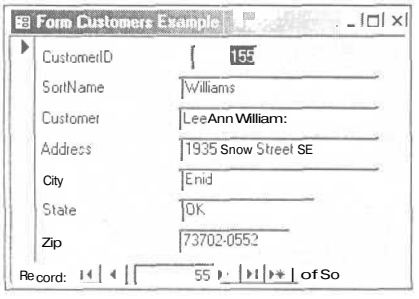


Рис. 5.5. После нажатия F5, чтобы войти в блок номера записи, вы можете набрать номер записи, которую вы хотите просмотреть. После нажатия клавиши Enter запись будет найдена

Вы также можете использовать клавиши PAGE UP и PAGE DOWN, чтобы передвигаться по записи или делать записи, а также вы можете использовать мышь, щелкая по навигационным кнопкам внизу формы.

Исследование свойств вашей формы

Каждый объект в Access, включая непосредственно базу данных, имеет свойства. Ваша форма не исключение. Имеются различные категории свойств формы. Они представлены вкладками в окне свойств. Некоторые свойства относятся к данным, другие - к форматированию формы, третьи - к событиям.

Табл. 5.1 показывает, что представляют собой категории свойств, к которым можно обращаться, щелкая вкладки в листе свойств.

Таблица 5.1. Объяснение категорий свойств

Категория	Описание
Формат (Format)	Свойства, которые принадлежат способу отображения объекта
Данные (Data)	Свойства/ которые принадлежат данным объекта, независимо от того, изображение ли это данных или то, каким способом они получены

Категория	Описание
Событие (Event)	Свойства, которые принадлежат событиям и связанным с ними процедурам
Другое (Other)	Свойства, которые принадлежат характеристикам объекта или разным его признакам
Все (АН)	Все категории или все свойства объекта

Вполне возможно, что вы не до конца понимаете то, какие конкретные свойства имеются в виду, но все, что вам нужно сделать для этого, - это нажать F1, чтобы получить мгновенную справку об этом свойстве. Давайте исследуем окно свойств формы.

1. Щелкните кнопку **Вид** (View), чтобы перейти в режим конструктора (Design view), все еще находясь в образце формы покупателей. Щелкните правой кнопкой мыши в области заголовка формы, чтобы открыть меню быстрого вызова команд, и щелкните **Свойства** (Properties) как показано на рис. 5.6[9].

Рис. 5.6. Находясь в конструкторе (Design view), вы можете щелкнуть правой кнопкой мыши в области заголовка формы, чтобы исследовать ее свойства

2. Щелкните вкладку **Данные** (Data) листа свойств, чтобы просмотреть свойства в этой категории.
3. Щелкните раскрывающийся список свойств «Источник записей» (Record Source), как показано на рис. 5.7. Заметьте, что в базе данных перечислена каждая таблица и запрос.
4. Закройте окно свойств и форму.

[9] Эту же команду Свойства вы можете найти в команде меню Вид. - *Науч. ред.*

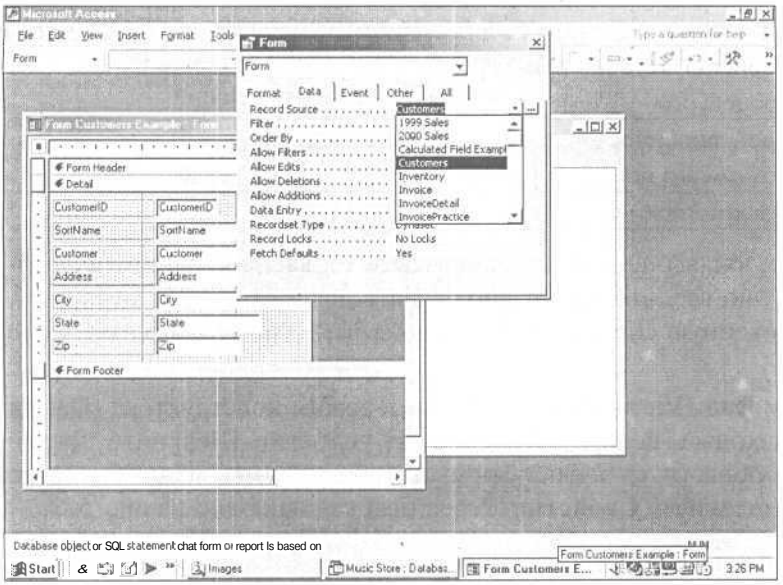


Рис. 5.7. Каждая доступная таблица и запрос перечислены в раскрывающемся списке Record Source

ЗАМЕЧАНИЕ. Если у вас возникают трудности с просмотром имен таблиц, лист свойств может быть расширен таким образом, чтобы имена таблиц или запросов находились в более широких полях.

Работа со свойством источника записи (Record Source Property)

Свойство источника записи может легко быть изменено программированием кнопки для активизации различных таблиц или запросов как источников записи. Главное, чтобы таблица или запрос имели ту же самую структуру, что и текущий источник записи. Например, вообразите, что у вас есть коммерческая таблица за 1999 год. Таблица 2000 года, вероятно, имеет те же самые поля, которые имеет таблица и за 1999 год, но они могут быть заполнены различными данными.

Создание командной кнопки (Command Button)

Просмотреть предыдущие годовые записи одним нажатием клавиши мыши, запрограммировать форму, установить прошлогоднюю таблицу как источник записи. Принимая во внимание то, что «2000 таблица» называлась «2000 Продажами», а «1999 таблица» называлась «1999 Продажами», и меняя источник записи на «1999 таблицу», просто сделайте кнопку события On Click и наберите

```
Me.RecordSource = "1999 Sales"
```

После конфигурирования кнопки прошлые годовые данные таблицы - лишь на «расстоянии щелчка мыши». Это может быть очень полезно.

ЗАМЕЧАНИЕ. После того как новый источник записи установлен, последующие щелчки на той же самой кнопке для прошлогодней таблицы не будут что-либо изменять, потому что установка источника записи завершена.

Чтобы сконфигурировать кнопку, выполните следующие действия:

1. В разделе **Объекты** (Objects) щелкните **Формы** (Forms), чтобы увидеть список форм.
2. Щелкните «2000 Продажи» (2000 Sales), затем щелкните **Конструктор** (Design) в панели инструментов окна базы данных, чтобы открыть форму в виде конструктора.
3. На панели инструментов в конструкторе нажмите **Панель элементов** (Toolbox tool).
4. Убедитесь, что палочка Мастер (Control Wizard's wand) наверху набора инструментов не выбрана, и щелкните по значку командной кнопки, как показано на рис. 5.8. Держите курсор неподвижным около кнопки в течение нескольких секунд, чтобы прочитать текст подсказки (ScreenTip) для полной уверенности.

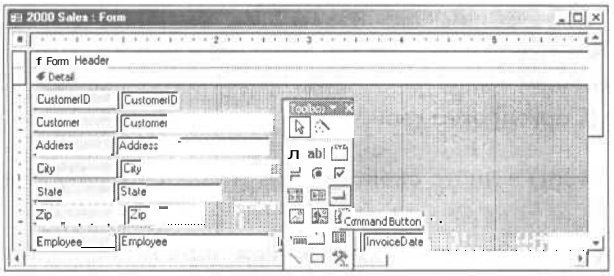


Рис. 5.8. Выберите командную кнопку в наборе инструментов, а затем поместите объект в форму

5. Ваш курсор изменит вид на кнопочный, показанный в правом нижнем углу командной кнопки на рис. 5.9. Справа от адресного текстового поля зажмите левую кнопку мыши и формируйте прямоугольник, перемещая курсор из левого верхнего угла в правый нижний (см. рис. 5.9).

ЗАМЕЧАНИЕ. Когда ваш курсор изменится на кнопочный, вы можете щелкнуть по области справа от поля адреса. Должна появиться кнопка. Этот способ быстрее, но также надо хорошо знать, как сформировать блок, потому что неизвестно, сколько у вас будет свободного места.

6. С выбранной командной кнопкой, нажмите F4. Щелкните вкладку **Все** (АИ). В свойстве имени напечатайте cmdChangeYear.
7. Наберите 1999 Sales в блоке свойства **Подпись** (Caption), чтобы заменить подпись, автоматически созданную для кнопки.

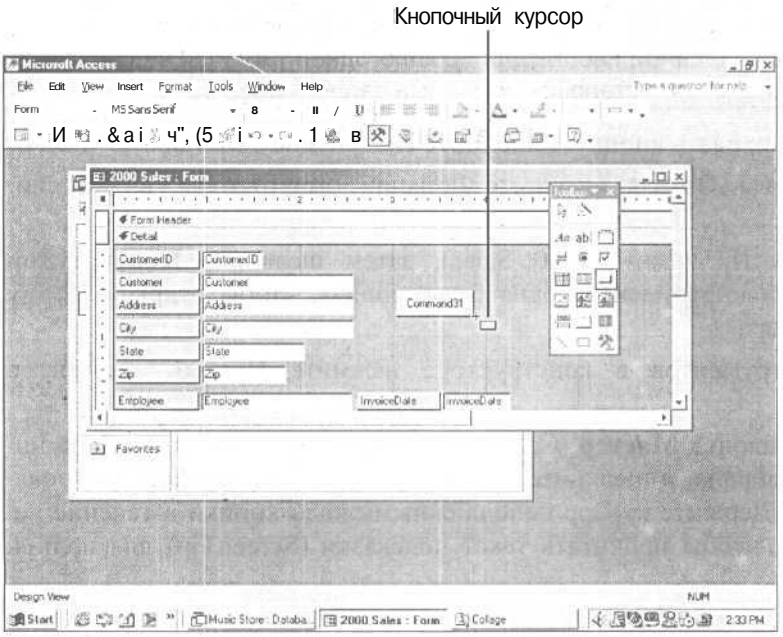


Рис. 5.9. Подпись (Command31), которая была произведена, когда была создана кнопка, может быть изменена перепечатыванием подписи непосредственно или созданием новой подписи в свойстве «Подпись» (Caption) окна свойств

8. Щелкните вкладку **Событие** (Event) в листе свойств и затем выберите **Нажатие кнопки** (On Click).
9. Щелкните раскрывающийся список и выберите **Процедура обработки событий** (Event Procedure). Щелкните кнопку **Создать** (Build) и напечатайте следующий код, как упомянуто ранее:

```
Me.RecordSource = "1999 Sales"
```

10. Щелкните кнопку **Close.** Закройте лист свойств, а затем сохраните и закройте форму.
11. Дважды щелкните форму 2000 Sales, которую вы только изменили. Щелкните кнопку 1999 Sales и смотрите за числом после "@" справа от навигационных кнопок внизу формы. Оно должно измениться на 32, потому что источник записи формы изменился. Заметьте, что текущая запись также изменилась.
12. Закройте форму.

Выбор таблицы для источника записи - не единственная доступная ему опция. Вы также можете выбрать существующий запрос или создать запрос «на лету» в то время, когда вы находитесь в блоке свойства источника записи. Следующие инструкции помогут вам изучить другие опции источника записи.

1. Выберите образец формы Customers и щелкните **Конструктор** (Design) на панели инструментов базы данных.

2. Щелкните правой кнопкой мыши по черному квадратику слева от горизонтальной линейки и выберите «Свойства» (Properties). Это дает тот же самый результат, что и щелчок правой кнопкой мыши в области заголовка формы.

ЗАМЕЧАНИЕ. Имеется много способов обратиться к листу свойств формы. В дополнение к двум методам, упомянутым в шаге 2, вы можете также щелкнуть правой кнопкой мыши по большой серой области на правой стороне формы в конструкторе. Вы можете испробовать эти опции, чтобы увидеть, какая работает лучше всего.

3. Нажмите вкладку **Данные** (Data) листа свойств и выберите **Источник записей** (Record Source).
4. Щелкните кнопку Создать (Build) с правой стороны блока, чтобы вызвать **Построитель запросов** (Query Builder), показанный на рис. 5.10.

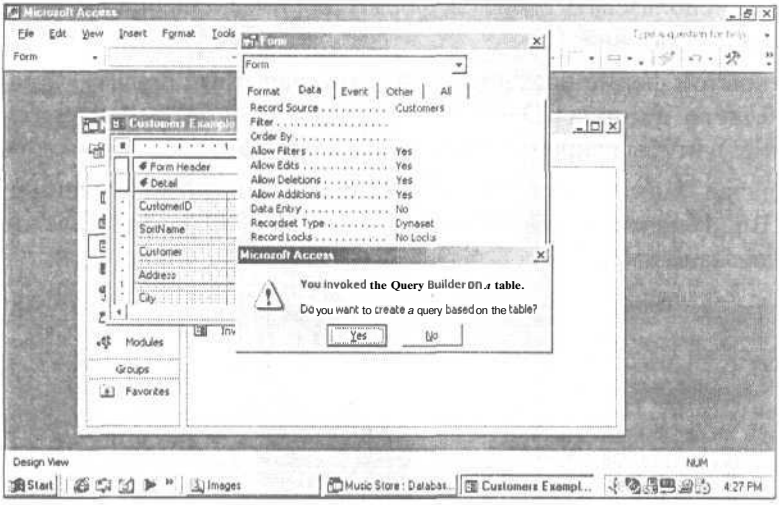


Рис. 5.10. Вызов построителя запросов дает возможность вам создать запрос и использовать это как источник записи «на лету»

5. Вызовите построитель запросов, щелкнув Да. Это открывает целый новый диапазон возможностей. Мы можем создавать запрос здесь или использовать код SQL для источника записи.
Обратите внимание на то, что если имеется уже выбранная таблица или запрос, когда построитель запросов вызван, то запрос будет отнесен к объекту, если вы ответите Да.

ЗАМЕЧАНИЕ. Поскольку разметка этого проекта запроса имеет тот же самый вид, который вы видите, когда открываете существующий запрос или создаете новый, он должен выглядеть знакомым. Различие в том, что этот запрос остается с формой. Когда вы закрываете построитель запросов, вы можете сохранить запрос.

6. Закройте запрос, не сохраняя его, щелкнув Нет в ответ на запрос системы сохранить изменения. Оставьте форму открытой для следующего примера.

Вы можете связаться с предварительно созданным запросом или создавать его «на лету», используя метод, который вы только что применили. Это некоторые из доступных вам возможностей. Вы можете использовать те же самые методы, чтобы изменить источник записи отчета, таким образом, чтобы форма и отчет соответствовали. Это все можно осуществить нажатием кнопки мыши.

Изменение цвета формы

Имеется много причин, почему люди изменяют цвет формы. Некоторые меняют цвет, чтобы сделать форму более красивой. Другие используют цвета для выделения. Например, если они хотят выделить один раздел или объект формы. Вы можете также использовать цвета, чтобы сделать условное форматирование данных в полях. Чтобы использовать свойства формы для изменения цвета, придерживайтесь следующих указаний:

1. В оставленной открытой форме последнего примера щелкните правой кнопкой мыши по **Области данных** (Detail section) формы, и затем выберите **Свойства** (Properties), чтобы открыть лист свойств этого раздела.
2. Щелкните по вкладке **Макет** (Format) в листе свойств и затем щелкните кнопку **Создать** (Build) напротив **Цвета фона** (Back Color). Откроется цветовая палитра, как показано на рис. 5.11.

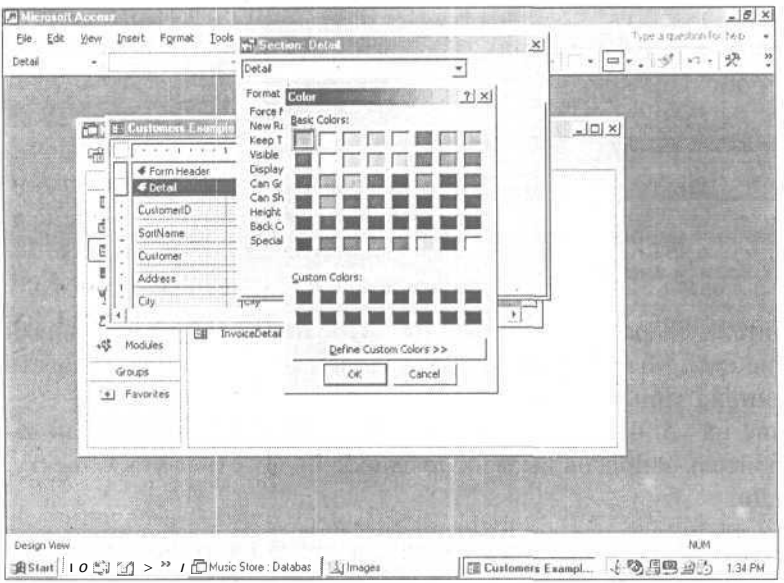


Рис. 5.11. Вы можете выбрать из палитры цвета заднего фона свойство назначать цвет вашей формы

ПОДСКАЗКА. Проще и удобнее всего изменить цвет области данных можно, просто щелкнув правой кнопкой мыши по области данных и затем выбрав **Цвет заливки/фона** (Fill/Back Color). Появляется раскрывающийся список с цветной палитрой для упрощенного выбора цвета.

3. В цветной палитре выберите 5-й блок в последней строке перед разделом **Дополнительные цвета** (Custom Colors) и нажмите OK.
4. Закройте лист свойств.
5. Заметьте, что цвет формы изменился. Щелкните X в окне, чтобы закрыть форму без сохранения каких-либо изменений (рис. 5.12).

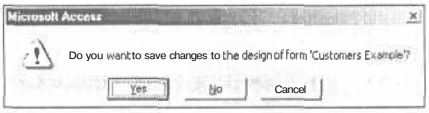


Рис. 5.12. Появится сообщение, напоминающее вам команду сохранения формы, но предоставляющее возможность закрытия без сохранения

Информацию относительно изменяющихся цветов объекта через VBA см. в разделе «Изменение свойств объекта через VBA» в гл. 7

Введение в элементы управления

Почему объекты форм и отчетов называют элементами управления? Чем управляют элементы управления? Говоря простым языком, они контролируют данные, будь это способ их изображения или управления ими через выражения, запросы, макросы и процедуры.

Возможно создать форму на «пустом месте», добавляя объекты к пустой форме. Объекты, чем бы они ни были: текстовыми полями, списками, полями со списком, кнопками, переключателями или любыми другими объектами в форме, - называются *элементами управления*. К ним обращаются из набора инструментов формы, который имеет значок для раскрытия или свертывания.

Текстовое поле может быть заменено на подпись (label), список или поле со списком. Они - лишь различные формы одного и того же. Другими словами, их можно преобразовать в любой из вышеупомянутых типов блока. Этот блок может быть связанным или несвязанным (bound, unbound). Связанный блок отличается от несвязанного блока тем, что у него есть источник данных (control source), который может быть полем (field) или выражением. Эти выражения могут включать функции, поля, операторы или константы. Функции могут быть встроенными функциями Access или индивидуальными функциями, созданными пользователем. Другие средства управления также могут быть связанными, но элементы управления, подобные текстовому полю, чаще всех бывают связанными, потому что они созданы с применением мастеров.

Когда вы используете мастер, чтобы создать форму, мастер автоматически размещает текстовые поля для всех полей, которые вы выбрали, и устанавливает

источник записи в таблице или запросе, который был выбран вами. Это означает, что все текстовые поля связаны с именами полей (fields), указанными в подписях (labels).

Несвязанное средство управления, с другой стороны, может использоваться, чтобы отобразить текст типа подписи, или оно может использоваться как контейнер, чтобы хранить элементы, выбранные из списка. Например, если вы имеете список элементов инвентаря и вы хотели бы выбрать из того списка элементы, которые были отозваны поставщиком. Это называется *списком для выбора* (pick list), и несвязанный элемент управления идеально подходит для этого видаприложения. •

Различные типы элементов управления

Когда вы исследуете различные типы элементов управления, находящихся в наборе инструментов, то быстро обнаруживаете, что можете делать с формой практически все. Табл. 5.2 показывает типы элементов управления, доступных в форме.

Таблица 5.2. Опции средств управления набора инструментов для формы

Элемент управления	Функция
Присоединенная рамка объекта	Содержит объект OLE или внедренное изображение
выключатель	Показывает квадрат с отметкой, если «включено», и пустым блоком, если «выключено»
Поле со списком	Раскрывающийся список опции значений для текущей поля из другого источника
Кнопка	Используется для вызова макроса или процедуры
Рис.	Изображает растровую картинку
Надпись	Изображает текст
Линия	Изображает одиночную линию с изменяемой толщиной
Список	Изображает список опций для выбора
Свободная рамка объекта или диаграмма	Содержит объект OLE или внедренное изображение, не привязанные к таблице
Кнопка выбора вариантов	Отображает точку в кружке, когда опция включена
Группа переключателей	Содержит множество кнопок выбора вариантов, флаговых кнопок или переключателей
Прямоугольник	Используется для красоты или для акцента. Может быть закрашен или оставаться пустым
Подчиненная форма	Отображает другую форму в пределах текущей формы
Вкладка	Может отображать многочисленные страницы, подобно папке файла, для экономии места

Надпись	Используется для ввода данных. Может быть связанной или несвязанной
Переключатель	Используется как переключатель с двумя состояниями: вверх или вниз

Группа опций позволяет пользователю легко выбирать значения полей из группы элементов. Предположим, что вы хотите выбрать между различными методами отгрузки. Или возможно вы хотите зарегистрировать кредитную карточку, которую ваш клиент предъявил при покупке.

Группа переключателей

Группа переключателей объединяет многочисленные кнопки - переключатели, флаговые кнопки так, чтобы они функционировали вместе как одно целое. Идея состоит в том, чтобы выбирать один элемент из многочисленных элементов управления, чтобы хранить в поле или использовать в процедуре для управления потоком данных. Если вы хотите выбрать многочисленные элементы, вы можете либо использовать список, либо обводить несколько отдельных кнопок прямоугольником, чтобы был эффект группы переключателей. В последнем случае вы обрабатываете каждую кнопку отдельно с точки зрения программирования.

Следующие инструкции помогут вам понять, как работает группа опций:

1. Откройте базу данных **Борей** (Northwind). Зажмите клавишу SHIFT, чтобы во время открытия пропускать открывающиеся окна. Под **Объектами,** выберите **Формы.**
2. Дважды щелкните форму **Заказы,** чтобы ее открыть.
3. Форма **Заказы** показывает различные методы отгрузки, как показано на рис. 5.13. В разделе **группы опций** с именем **Доставка** (Ship Via) выберите вашего грузоотправителя, щелкая одну из трех рамок: **Почта** (Speedy), Иное (United), **Ространс** (Federal). Свободно щелкайте и другие рамки, но оставьте помеченным **Почта** (Speedy) перед закрытием формы.

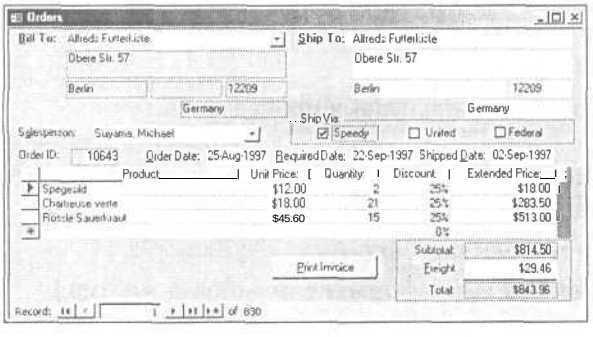


Рис. 5.13. Группа переключателей - группа элементов управления, которые позволяют вам выбирать одну опцию из списка опций

4. Закройте форму. Под **Объектами** щелкните **Таблицы,** а затем дважды щелкните таблицу **Заказов.**
5. Смещайтесь вправо, пока не увидите ячейку **Доставка** (Ship Via). Важно понять, что данные, выбранные в группе переключателей **Доставка** (Ship Via) сохранены в этом поле. Мастер группы переключателей позволяет вам выбрать в зависимости от того, хотите ли вы хранить выбор Группы переключателей в поле таблицы или только программно использовать данные.
6. Закройте таблицу и оставьте базу данных открытой для следующего примера.

То, что вы только что видели, - хороший способ сделать переключатели пользователя для маленького числа элементов. Если вы имеете большое количество элементов, то предпочтительно использование списка или комбинированного раскрывающегося списка. Группу переключателей быстро и просто установить используя мастер, который автоматически активизируется, когда вы размещаете группу на форме. В предыдущем типичном упражнении поле Ship Via было источником данных (control source) для группы переключателей по имени ShipVia.

Полесосписком

Поле со списком - средство управления, которое дает пользователю возможность выбирать из списка значения, но вместо щелчка кнопки-переключателя пользователь уже выбирает из раскрывающегося списка. База данных «Борей» (Northwind) имеет пример раскрывающегося списка с несколькими элементами на выбор.

Находясь в Northwind, откройте форму **Сотрудники** (Employees) и щелкните **Личные данные** (Personal Info). Этот элемент управления, который напоминает папку с файлами, называется *вкладкой.* Затем щелкните **Обращение** (Title of Courtesy). Источник данных этого поля был создан с помощью *списка значений,* который состоит из списка заголовков типа «господин» или «госпожа», отделенных точками с запятой. Это может легко быть преобразовано в гораздо больший список, изменив источник строки на таблицу или запрос. Закройте форму.

Изучение процедур обработки событий

Событие подобно «пусковой установке» для ваших процедур. Всякий раз, когда вы обновляете объект, происходит событие, дающее возможность изменить другие объекты. Например, если вы хотите найти значение, основанное на другом значении, используйте событие After Update, как показано далее:

1. Откройте базу данных магазина музыки из папки AccessByExample.
2. **Под Формами** дважды щелкните **Создание формы с помощью мастера.**
3. На стр. 1 мастера, в разделе **Таблицы и запросы,** выберите **Таблица:InvoicePractice**.
4. Щелкните кнопку >>, чтобы перенести все поля в сторону «выбранные».

5. Нажмите **Далее,** чтобы открыть 2-ю страницу мастера.
6. На стр. 2 выберите **Табличный** (Datasheet) и щелкните Далее.
7. На стр. *3* выберите **Стандартный и** щелкните **Далее.**
8. На стр. 4 назовите форму InvoiceDetail, выберите **Изменить макет формы** и щелкните Готово.
9. Щелкните правой кнопкой мыши PartNumber в области данных и выберите **Преобразовать элемент в** (Change To), затем выберите **Поле со списком** (Combo box) (рис. 5.14.)

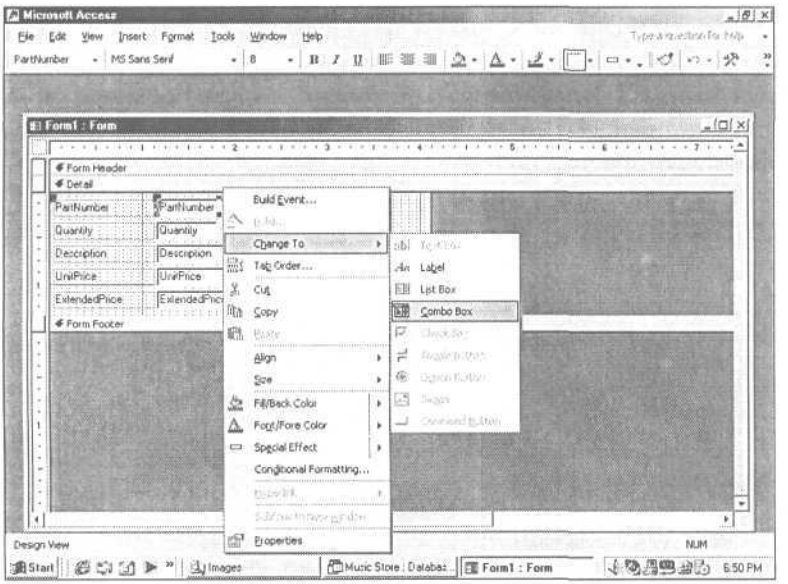


Рис. 5.14. Вы можете заменить текстовое поле на поле со списком, щелкнув правой кнопкой мыши текстовое поле и выбрав Change To

10. Щелкните снова правой кнопкой мыши PartNumber и на сей раз выберите **Свойства,** чтобы открыть список свойств.
11. Убедитесь, что вы находитесь на вкладке **Данные** (Data), и щелкните кнопку **Создать** (Build) напротив свойства **Источник строк** (Row Source), чтобы открыть диалоговое окно **Добавление таблицы** (Show Table).
12. В диалоговом окне **Добавление таблицы** (Show Table) выберите Inventory, щелкнув **Добавить** (Add), и нажмите **Закрыть** (Close).
13. Выберите PartNumber, Instrument, Model и UnitPrice в таблице, дважды щелкнув поля в таблице Inventory, как показано на рис. 5.15.

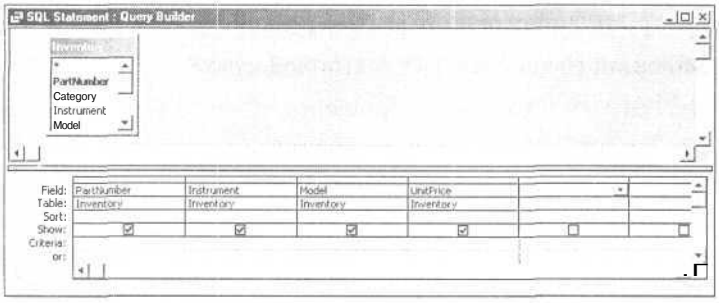


Рис. 5.15. Выбор полей двойным щелчком в Построителе запросов (Query Builder) для свойства источника строк

14. Щелкните X, чтобы закрыть **Построитель запросов.** Когда появится окно сообщения, как показано на рис. 5.16, щелкните Да, чтобы сохранить изменения, сделанные в выражении SQL, и обновить свойство.

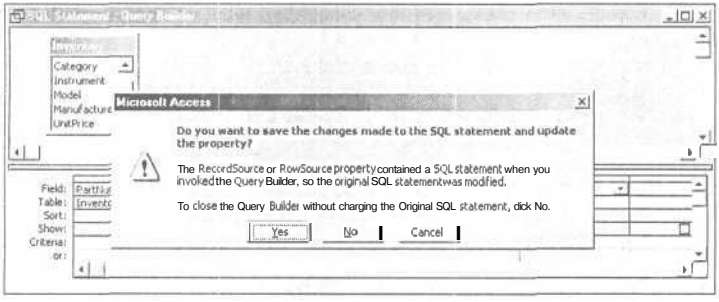


Рис. 5.16. Нажатие Да (Yes) в сообщении от Построителя запросов гарантирует, что запрос будет сохранен в форме

15. В поле свойства **Ширина столбцов,** которое находится на вкладке **Макет,** напечатайте .7; 1.5; 1; 0 и затем нажмите **Ввод.** Заметьте, что дюймовые индикаторы автоматически вставлены.

ЗАМЕЧАНИЕ. Это свойство устанавливает ширину столбца для полей, которые будут показаны, когда список раскрывается. Если для первого поля записано нулевое (0) значение (например, 0 "; 0.9"; 0.7 '), то вы не будете видеть это поле. Второе поле будет показываться первым. Как бы то ни было, правильные данные (PartNumber) будут вставлены в подчиненную (underlying) таблицу. Чтобы быть уверенным в том, что все ваши поля будут показаны в полной мере, измените **Ширину списка** на сумму всех ширин столбцов. Например, если три столбца имели ширину 1.5, свойство **Ширина списка** должно быть 4.5. Это позволит расширить видимую ширину за пределы правого края поля со списком.

16. Присвойте свойству **Число столбцов** (Column Count) значение 4, чтобы соответствовать числу столбцов, которые находятся в запросе (Рис. 5.17), а свойству **Ширина списка** - значение 3.2.

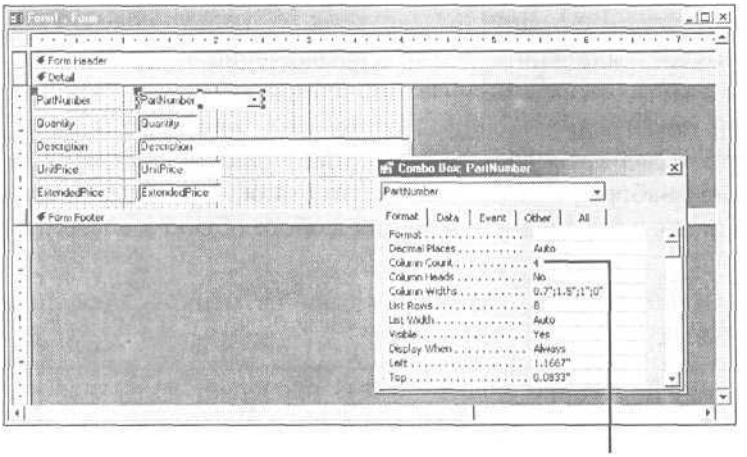


Рис. 5.17. Вы должны изменить свойство Число столбцов, чтобы отразить число столбцов в таблице запросе

17. Щелкните вкладку **Событие** (Event) и затем щелкните кнопку **Создать** (Build) в событии **После обновления** (After Update).

18. Выберите **Программы** (Code Builder) и щелкните OK. Напечатайте следующий код в этом событии:

```
Dim sDesc As String, dPrice As Double
sDesc = Me.PartNumber.Column(1) & " " & Me.PartNumber.Column(2)
dPrice = Me.PartNumber.Column(3)
Me.Description = sDesc
Me.UnitPrice = dPrice
```

Объясняется это так: В первой строке вы объявляете ваши переменные как переменные типа «Строка» и «Десятичное число двойной точности» (string и double). Вторая строка связывает или сочетает три части данных. Первая часть данных (Me. PartNumber. Column(1)) содержит зарезервированное слово Me. Вы можете использовать это слово в модулях формы, чтобы получить ссылку на текущую форму. В данном случае вы получаете ссылку на свойство Column элемента управления PartNumber вашей формы. Поскольку вы установили свойство **Число столбцов** равным четырем и установили запрос на выбор этих полей, вы можете теперь использовать это свойство в модуле, чтобы получить данные и присвоить их вашей переменной.

Вторая часть данных - просто пробел (). Третья часть данных (Me. Partnumber. Column(2)) - третий столбец в запросе. Важно помнить, что свойство столбца нулевое, что означает, что вы начинаете считать с нуля. Первый столбец - О,

второй столбец - 1, и т. д. Поэтому, Me. PartNumber. Column(1) - в действительности второй столбец.

Когда вы складываете все это вместе, вы получаете значение, подобное Flugelhorn FE747, что является названием инструмента и модельным номером. Переменная Price устанавливается почти таким же образом, за исключением того, что вам нужно заполучить только одно значение.

Затем вы просто назначаете эти переменные, найденные из другой таблицы, полям в подчиненной (underlying) таблице. Переменные действуют подобно временным контейнерам для хранения, которые отыскивают и быстро передают данные. Когда вы набирали, вы, вероятно, заметили, что после печати точки (.) перед вами появляется список опций для любого объекта, в котором вы находитесь.

19. Выберите **Отладку, Компиляцию,** магазин музыки из меню и щелкните кнопку **Сохранить** (Save).

ЗАМЕЧАНИЕ. При компилировании модуля Access проверяет коды ошибок. Закройте окно.

20. Когда вы вернетесь к списку свойств элемента управления PartNumber, щелкните по элементу управления Quantity, оставляя открытым список.

21. Выберите событие **После обновления** (After Update) и щелкните кнопку **Создать** (Build). Щелкните **Построитель: Программы.**

22. Введите следующий код:

```
Dim sExt As String
sExt = Me.Quantity * Me.UnitPrice
Me.ExtendedPrice = sExt
```

23. Закройте окно свойств и затем закройте форму. Щелкните Да (Yes), когда спрашивается сохранить изменения.

24. Дважды щелкните форму InvoiceDetail, чтобы загрузить ее.

25. Щелкните кнопку «Новая запись» (New Record) в селекторе записей в нижней части формы, чтобы добавить новую запись. (Подобная кнопка находится на панели инструментов.)

26. Щелкните на ниспадающем списке в поле PartNumber и выберите вторую строку, как показано на рис. 5.18.

27. Заметьте, что поле Discription (Описание) автоматически вставилось, так же как и UnitPrice (Цена единицы товара).

28. Перейдите к полю Quantity (Количество), наберите 2, а затем нажмите ввод. Обратите внимание, что правильное значение автоматически вставляется в ячейку ExtendedPrice (Общая цена).

29. Закройте форму.

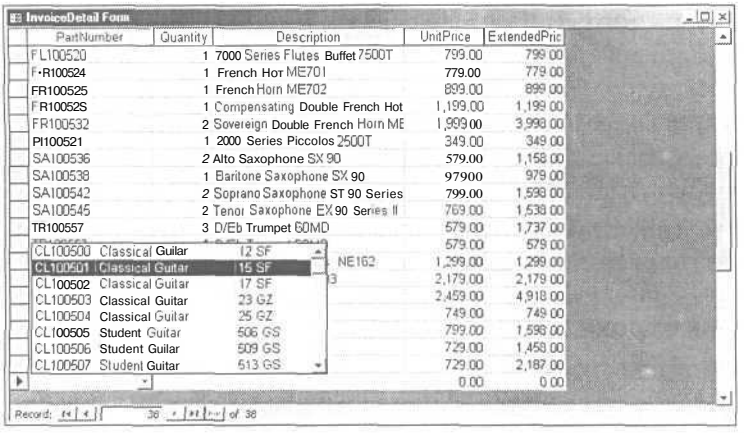


Рис. 5.18. Значения полей подстановок (lookup field) автоматически вводятся, когда выбирается значение из ниспадающего списка

Как управлять элементами управления

Итак, вы вкусили возможности, которыми обладает форма. Очевидно, их гораздо больше. Но прежде чем вы пойдете дальше, необходимо уяснить некоторые важные свойства элементов управления. Давайте сначала рассмотрим свойства списка и поля со списком.

Свойство «Имя»

Новички легко путают свойство «Данные» (Control Source) со свойством «Имя», в особенности из-за того, что они часто имеют одно и то же название. Свойство «Имя» - лишь только название для обращения к элементам управления. Оно обеспечивает возможность ссылаться на несвязанный (unbound) элемент управления. Не имеется вообще никакой связи с подчиненными (underlying) данными, использующими это свойство.

Свойство «Источник данных»

Напротив, свойство **Источника Данных** (Control Source) в действительности связано с подчиненными данными. Вы можете связать это свойство с ячейкой, полученной из списка, созданного любой таблицей, выбранной в свойстве источника записи формы. Вы также можете щелкнуть кнопку Build, чтобы ввести выражение в это свойство.

Свойство «Тип источника строк»

Это свойство определяет источник записи данных строки для управления. Доступные опции включают:

- Таблицу/запрос. Тип данных - таблица или запрос. Работает совместно со свойством «Источник строк», которое указывает имена таблиц, запросов или SQL-выражения.

- Список значений. Вместо того чтобы запрашивать данные, вы указываете здесь, что вы внесете ваши собственные данные в свойстве «Источник строк», отделенные точками с запятой. Они обычно состоят из короткого списка значений. Элемент управления отображает вносимые туда данные.
- Список полей. Вместо изображения записей выбор этого типа обозначает, что вы запрашиваете список полей из таблицы или запроса, указанных в свойстве «Источник строк». Это - простой способ напечатать список полей для любой структуры таблицы, которая вам нужна.

Свойство «Источник строк»

Следует отметить, что вы также можете заполнять список или поле со списком, используя код, ссылаясь на свойство «Источника строк». Когда вы выбираете таблицу или запрос в качестве «Источника строк» в элементе управлении формы, то поле со списком или список отображают набор опций, используя данные таблицы или выбранного запроса. Когда пользователь выбирает значение из списка, элемент управления изображает это значение до тех пор, пока пользователь не изменит его.

Например, если вы хотите позволить пользователю выбирать из набора кодов в списке Inventory формы Invoice, используйте поле со списком или список, чтобы обеспечить список для поиска (lookup). Пользователь может и не помнить код, но, если он видит список кодов с соответствующими элементами, выбор становится простым. Как вы видели, событие After Update обеспечивает способ изменения других полей в форме в зависимости от того, какое значение выбрано в элементе управления. Так что это - хороший способ находить значения для заполнения других полей.

Свойство «Число столбцов»

Хотя это свойство может быть легко пропущено, оно не менее важно, чем другие. Если вы хотите отобразить четыре столбца, но оставите все остальные параметры как для одного, то вы отобразите только один столбец. Если вы изменяете установку на четыре, вы по-прежнему должны должным образом изменить свойство ширины столбца, чтобы уместить все четыре столбца. Убедитесь, что использовали числовое значение.

Свойство «Ширина столбцов»

Каждое значение, введенное здесь, устанавливает ширину столбцов в элементе управления и должно быть отделено точкой с запятой. Используя предыдущий пример, предположим, что вы хотите установить для всех четырех столбцов одно-дюймовое значение. Вы просто вводите 1; 1; 1; 1. Если вы напечатаете 2 вместо этого, первый столбец будет двухдюймовым, а остальные однодюймовыми по умолчанию. Если вы опускаете значение, то программа принимает заданную по умолчанию ширину в 1 дюйм. Если вы хотите скрыть столбец, вы устанавливаете ширину, равную нулю. Если вы оставите это свойство пустым, ширина всего элемента управления будет поровну поделена между столбцами.

Свойство «Присоединенный столбец»

Это свойство определяет, в каком столбце содержится значение, подлежащее возвращению, когда элемент выбран из списка. Скажем, присоединенный столбец установлен равным единице, а первый столбец скрыт, потому что ширина столбца установлена в 0. Отображенный столбец будет второй ячейкой, а возвращенный столбец - первой. Если бы первое поле было PartNumber, а второе поле было бы Part, то вы видели бы название части (Part), отображенное в элементе управления, но номер части был бы помещен в подчиненной таблице.

Этот метод мог бы быть удобным для подстановки (lookup) значений, но возможны некоторые недоразумения, потому что поле будет скорее всего подписано как PartNumber. Однако это легко исправляется простой заменой метки на Part. Таким образом, вы скрываете возвращенное значение связанного столбца и отображаете подстановочный столбец. По сути дела, мастер подстановок в конструкторе таблицы предлагает то же самое. Пользователь ни о чем не будет догадываться, а надлежащая информация будет помещена в таблицу. Если вы хотите возвратить название части (Part), вы должны установить значение свойства на 2.

Создание вычисляемого элемента управления

Связанный или несвязанный (bound и unbound) элемент управления может легко быть преобразован в вычисляемый (calculated) элемент управления. Просто щелкните правой кнопкой мыши элемент управления и выберите **Свойства** (Properties). Щелкните кнопку **Создать** (Build) свойства **Данные** (Control Source), чтобы запустить построитель выражений (expression builder). Предположим, что в таблице, к которой вы обращаетесь имеются поля City (Город), State (Государство), Zip (Почтовый индекс); тогда вы можете напечатать выражение (или использовать построитель для составления выражения), подобное следующему:

```
[City] & ", " & [State] & " " & [Zip]
```

Это выражение использует амперсанд (&) - оператор для связи значений полей с литерами (константами). Тот факт, что City, State и Zip находятся в скобках, говорит о том, что все они - значения полей. Запятая и пробел - литеральные значения. Вместе они составляют значение, которое может быть частью адреса. Если поля имели бы числовые значения, то вы могли бы также легко создавать выражение, которое включает операторы сложения, вычитания, умножения или деления.

Другой способ создать вычисляемый элемент управления

Вы можете также создать вычисляемый элемент управления, используя запрос. Если в сетке запроса у вас имеется таблица Customers (Клиенты) из базы данных магазина музыки и поля Customer и Address уже находятся в сетке, то справа от поля Address есть пустое место (empty spot). В строке **Поле,** напечатайте CSZ: [City] & ", " & [State] & " " & [Zip;, затем щелкните раскры-

вающийся список кнопки **Новый объект** (New Object), как показано на рис. 5.19, и выберите **Автоформа** (AutoForm). Программа создает ячейку с именем CSZ. Запрос и форма под названием Calculated Field Example находятся в той же базе данных. Заметьте, что источник данных этой ячейки - CSZ (а не выражение, записанное в запросе).

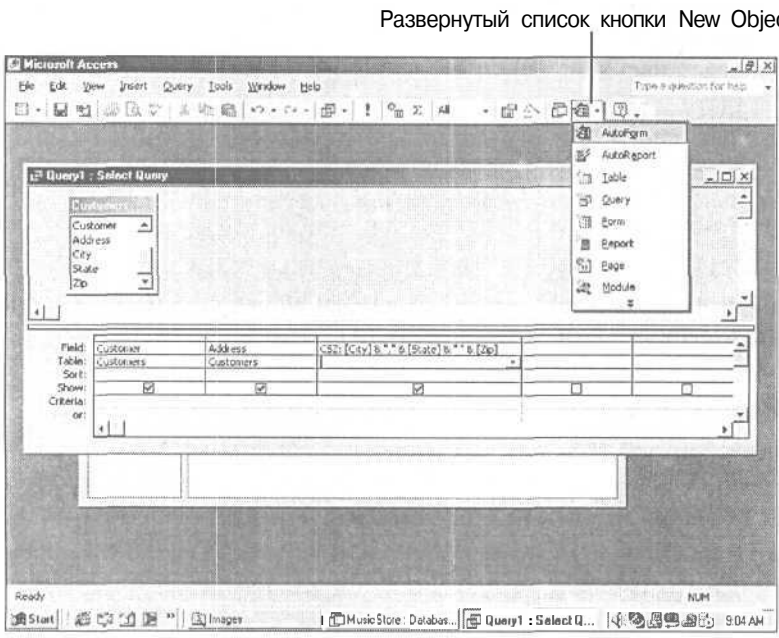


Рис. 5.19. Используйте кнопку New Object, чтобы создать вычисляемую ячейку в форме

Создание командной кнопки

За исключением событий Before Update и After Update, кнопка имеет в основном те же самые события, которые имеет и другое средство управления. Событие On Click, возможно, одно из наиболее часто используемых событий этой кнопки. Щелчок этой кнопки вызывает действие, которое может храниться в виде макроса или модуля.

Печать записи

Обычное желание пользователя - распечатать текущую запись в форме. Хотя в Access никогда не было этой возможности, нетрудно реализовать ее с использованием процедуры. Следующие шаги ознакомят вас с процессом создания кнопки для печати текущей записи.

1. Откройте базу данных магазина музыки из папки AccessByExample. Под Объектами, щелкните Forms, чтобы рассмотреть список форм.

2. Щелкните Customers и **Конструктор** (Design) в панели инструментов базы данных, чтобы открыть форму Customers в виде конструктора.
3. В виде конструктора щелкните значок **Панель элементов** на панели инструментов.
4. Подведите курсор на элемент управления, которой напоминает кнопку. Наблюдайте появившуюся всплывающую подсказку командной кнопки - просто чтобы удостовериться.
5. Убедитесь, что палочка мастер (рядом со стрелкой выбора объектов) не активизирована. Щелкните кнопку и, когда курсор поменяется на «кнопкоподобный», переместите его к области данных формы возле отметки 4.5".
6. Зажмите клавишу мыши и переместите курсор с левого верхнего в правый нижний угол, чтобы образовался прямоугольник приблизительно 1 дюйм по ширине.
7. Щелкните правой кнопкой мыши по объекту, а затем выберите Properties, как показано на рис. 5.20, чтобы открыть лист свойств.

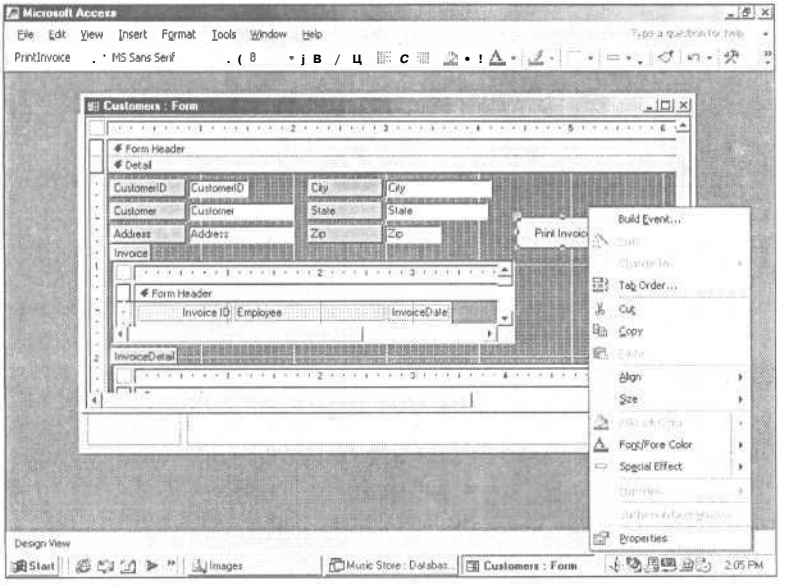


Рис. 5.20. Чтобы запрограммировать командную кнопку, вы должны щелкнуть по ней правой кнопкой мыши и выбрать Properties, чтобы открыть список свойств

8. Щелкните вкладку АН в листе свойств. Убедитесь, что вертикальная полоса прокрутки на правой стороне листа свойств находится наверху.
9. Измените подпись на Print Invoice.
10. Измените имя на cmdPrintInvoice (без пробела).
11. Щелкните вкладку **Событие** (Event) в списке свойств.

12. Щелкните кнопку Build справа в строке события **Нажатие кнопки** (On Click).

ПОДСКАЗКА. Чтобы избегнуть всей этой волокиты с использованием списка свойств, просто щелкните правой кнопкой мыши командную кнопку, а затем выберите событие **Обработка событий** (Build Event) - первый пункт. Хотя вам все-таки придется выбрать **Программы** (Code Builder) на этом этапе, такой способ позволит сократить несколько шагов. Это особенно удобно для редактирования процедур, потому при использовании **Построителя** (Build Event) вам не дают возможности выбирать событие, которое вам нужно при создании новой процедуры. Если вы хотите обойти необходимость выбирать **Программы** (Code Builder) из списка, выберите **Сервис, Параметры, Формы и отчеты** в окне базы данных. Затем поставьте галочку в пункте «Всегда использовать процедуры обработки событий» (Always Use Event Procedures).

13. Выберите **Программы** и затем щелкните OK.

14. Напечатайте следующий код (исключая первые и последние строки) в окне модуля:

```
Private Sub PrintInvoice_Click()
On Error GoTo Exit_PrintInvoice_Click

 Dim ReportName As String, InvID As String, InvWhere As String

    'Получает invoice ID из подчиненной формы и записывает его в переменную
InvID = Me.Invoice_Subform.Form.Controls![Invoice ID]
    'Создает выражение для оператора where, объединяя строки и записывая результат в переменную
InvWhere = "[InvoiceID]=" & InvID
    ReportName = "Invoice"

 DoCmd.OpenReport ReportName, acViewNormal,, InvWhere
Exit_PrintInvoice_Click:
Exit Sub
End Sub
```

ЗАМЕЧАНИЕ. Убедитесь, что вы не повторяете первые и последние строки (private sub и end sub) в этом коде. Они включены в код только с целью справки. Вам не нужно печатать эти строки, так как они вставлены до и после позиции курсора с целью облегчить вам начало печати между строками.

15. Щелкните Debug, Compile Music Store и сохраните изменения. Закройте окно свойств и форму.

16. Под **Формами,** дважды щелкните Customers, чтобы открыть форму. Нажмите F5, чтобы перейти в поле выбора записи (внизу экрана маленькая строчка **За-**

пись:, ее трудно увидеть - *Пер.*). Наберите 23, чтобы выделилась Irvine Elementary.

17. Щелкните кнопку Print Invoice, чтобы распечатать счет Irvine Elementary; затем закройте форму.

Как вы можете обратиться к элементу управления подчиненной формы? Строчка, начинающаяся с `InvID`, показывает вам, как это сделать. Вам нужен Invoice ID, чтобы обратиться к тому же самому ID в отчете. Следующая строчка устанавливает оператор `where`. Все после знака равенства может быть помещено в свойство «Фильтр» отчета. `ReportName` - это просто переменная для хранения имени отчета. Метод `OpenReport` очень мощный, потому что вы можете применять фильтр или оператор `where` для выбора записей.

На данном этапе не стоит сильно волноваться относительно понимания каждой строчки в этой процедуре. Вы детальнее изучите код в последующих главах. Здесь вам лишь предложена идея того, что вы можете делать с событиями, привязанными к коду.

Обращение с базовыми (underlying) записями

Чтобы должным образом исследовать, как базовые (underlying) записи относятся к форме, лучше всего начать со свойства «Источник записей». И таблицы и запросы, связанные с формами через свойство источника записей, используют наборы обновляемых записей. Поля, доступные в форме, зависят от таблицы или запроса, которые привязаны к форме. Их список может быть расширен многими способами. Один из них состоит в том, чтобы добавить другую таблицу к запросу на выборку, присоединенную к форме. Другой способ состоит в том, чтобы создать подстановочный элемент управления (lookup control), зависящий от полей в другой таблице. Эти поля тогда станут доступными посредством программного кода, используя свойство Column.

Многочисленные табличные запросы как источник записей

Если ваша форма привязана к запросу на выборку, основанному на одной таблице, не должно быть никаких проблем с редактированием записей. Но когда запрос основан больше чем на одной таблице, это становится более интересным. Если вы хотите редактировать присоединенную ячейку с одной стороны связи «один ко многим», вы должны обеспечить ссылочную целостность и каскадное обновление в окне связей. Это - плюс, так как вы хотите сохранить целостность ваших связей.

Если каскадное обновление установлено, то вы не можете добавлять или изменять данные в присоединенном поле *(join* field) со стороны «многие». А когда вы изменяете данные присоединенной поля (join field) со стороны «один», это автоматически изменяет соответствующую ячейку со стороны «многие». Единственное исключение - поле **Счетчик** (Autonumber). Вы не можете изменить эту ячейку Join ни с какой стороны, если только вы не удалите временно связи и не замените Autonumber на Number с последующим восстановлением связей.

Имена полей

Было показано, что данные, отображенные в элементе управления, не обязаны быть такими же, что и данные в базовой (underlying) таблице. Также следует отметить, что подписи к полям (field labels) в форме не должны иметь те же самые имена, которые имеют и сами базовые поля (underlying field data). Например, если вы имеете поле с именем PartNum, вы можете запросто расширить имя в подписи: Part Number. Это не затрагивает основные данные, а просто является справкой, для пользователя.

Проектирование реляционной формы

Мастер формы делает проектирование реляционной (relational) формы простой задачей. Чтобы спроектировать реляционную форму с нуля, вы должны создать подчиненную форму для связанной таблицы. Мастер может создавать для вас подчиненную форму и разместить ее в надлежащей позиции в форме.

Как использовать мастер формы для создания формы

Как упомянуто ранее, мастер формы может автоматически решать рутинные задачи. Следующее упражнение показывает, как мастер может сэкономить Ваше время, решая задачи, связанные с реляционными формами, типа конфигурирования подчиненной формы.

Следующие этапы ознакомят вас с созданием реляционной формы при помощи мастера формы.

1. Откройте базу данных магазина музыки из папки AccessByExample. Под **Объектами** выберите Forms.
2. Дважды щелкните Create Form By Using Wizard, чтобы открыть первую страницу мастера формы.
3. На стр. 1 мастера в самораскрывающемся списке **Таблицы и запросы** выберите Table:Customers.
4. Рядом с **Доступными полями** щелкните кнопку >>, чтобы переместить все поля в **Выбранные поля.**
5. Выделите Sort Name в **Выбранных полях** и щелкните кнопку <, чтобы вернуть ее обратно к **Доступным полям.**
6. Оставайтесь на стр. 1 диалогового окна Form Wizard. В самораскрывающемся списке **Таблицы и запросы** выберите Table:Invoice (см. рис. 5.21).
7. Рядом с **Доступными полями** нажмите кнопку >>, чтобы переместить все поля к столбцу **Выбранные поля,** как показано на рис. 5.21.
8. Щелкните Invoice.CustomerID под **Выбранными полями** и щелкните кнопку < чтобы вернуть ее обратно к **Доступным полям.**
9. В самораскрывающемся списке **Таблицы и запросы** выберите Table:InvoiceDetail.
10. Нажмите кнопку >>, чтобы перенести все поля к столбцу **Выбранные поля.** Щелкните InvoiceDetail.InvoiceID, затем кнопку <, чтобы вернуть ее обратно к блоку **Доступных полей.** Щелкните Далее, чтобы открыть стр. 2 мастера.

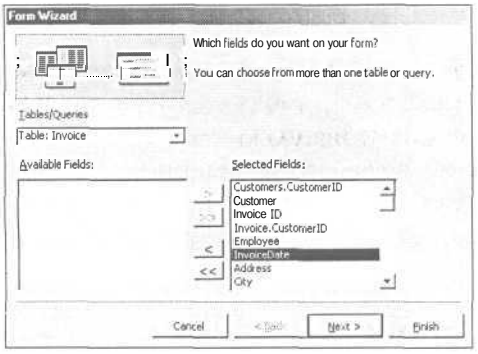


Рис. 5.21. Мастер формы проведет вас через выбор объектов и опций форматирования для вашей формы

11. Когда на стр. 2 вы увидите вопрос «В каком виде представить данные?», щелкните **Далее**, подтвердив выбор «Customers». Тем самым вы также подтверждаете выбор **Форма с подчиненными формами** в нижних двух опциях.
12. Стр. 3 мастера спрашивает: «выберите внешний вид подчиненной формы:». Поскольку для обеих подчиненных форм уже выбран **Табличный** (Datasheet), нажмите **Далее** (Next).
13. На стр. 4 мастера оставьте выбранный **Стандартный** и затем нажмите Далее.
14. Оставьте предложенные имена для форм и щелкните **Готово,** чтобы открыть форму, показанную на рис. 5.22.
15. Поиграйтесь с формой, а затем закройте ее.

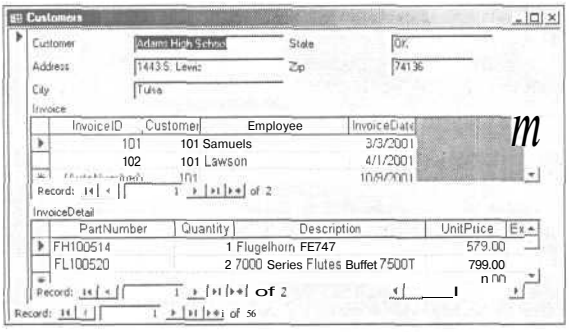


Рис. 5.22. Мастер формы создает реляционную форму с подчиненными формами, автоматически вложенными в главную форму

Всего за несколько минут вы создали реляционную форму, основанную на трех таблицах. Подчиненные формы связаны должным образом, потому что таблицы, на которых они основаны, связаны в окне «Схема данных» (Relationships). Хотя эта форма могла быть создана вручную, мастер сберег Ваше время, обрабатывая размещение подчиненной формы, ссылки (links) и выполняя другие вспомогательные работы для вас.

Что дальше?

Эта глава представила вам некоторые из основ проектирования формы и обслуживания. Вы узнали о свойствах и событиях форм, и свойствах и событиях элементов управления. Вы научились создавать командную кнопку и вычисляемый элемент управления. Вы также узнали о присоединенном (attaching) коде к событию в элементе управления или форме.

В гл. 6 применяются некоторые из этих концепций к отчету при знакомстве с его многосторонностью. Если в главе 5 внимание было сфокусировано на способах структурирования и ввода данных, то в следующей главе сделан акцент на способе представления данных для просмотра и их печати. Вполне достаточное место отведено разделам и элементам управления отчетом.

6
Анализ отчетов

В предыдущей главе вы прочитали о формах и элементах управления. В этой главе говорится о различных компонентах отчетов, таких, как элементы управления и разделы (sections). Вы научитесь группировать и сортировать элементы управления внутри разделов (sections). Более подробно вы узнаете:

- в чем разница между электронными таблицами (spreadsheets) и отчетами;
- как создавать отчеты с самого начала;
- как разделы и элементы управления работают вместе;
- как управлять элементами управления в отчетах;
- как делать вычисления, подобные производимым в электронных таблицах, в отчетах;
- как контролировать группировку и сортировку;
- как подводить итоги (total) внутри групп;
- как спроектировать реляционный (relational) отчет при помощи мастера.

Сравнение отчетов и электронных таблиц

Отчеты служат для форматирования выходных данных. Вы можете выводить данные на любое устройство, подключенное к вашему компьютеру, будь это стандартный монитор или принтер или более замысловатое устройство, такое, как проектор или Web-сервер. Отчеты также служат для большей гибкости. Одни и те же данные могут быть просмотрены при помощи различных типов отчетов, поэтому вы можете анализировать их с разных точек зрения. И наоборот, вы можете использовать один и тот же отчет для различных источников данных, таких, как разные таблицы, имеющие одинаковую структуру, или динамические ресурсы, такие, как запросы и процедуры. Свойства источника записей могут быть программно изменены для взаимодействия с широким спектром источников данных.

Отчеты имеют такие же функциональные возможности, как запросы и формы, по крайней мере в одном: они динамичны. Это означает, что, когда рассматриваемые данные изменяются, отчет формирует сам себя в соответствии с изменениями. До известной степени отчеты становятся живыми документами. В отличие от форм они *отражают,* но *не влияют* на изменения в рассматриваемых данных. Формы могут делать и то и другое. Однако это не означает, что вы не можете выполнять вычисления, «на лету» производить условное форматирование, группировать и сортировать данные и выполнять целую кучу других операций (используя формы. - *Пер.*). Но эти операции влияют только на то, каким образом отражаются данные. Если вы хотите сделать некие постоянные изменения в исходных (underlying) данных, оформляйте их в объектах, предназначенных под ввод, таких, как форма или таблица.

Эти функциональные возможности дают вам большое преимущество перед электронными таблицами (spreadsheet). Когда вы добавляете данные в электронную таблицу, вы также должны добавить или скопировать форматирование, который поставит таблицу в соответствие с используемыми данными. После того как отчет создан и уточнен в соответствии с вашими спецификациями в Access, вы можете использовать его снова и снова, будучи уверены, что получите согласующиеся результаты раз за разом.

Хотя обновление полей данных в связанной с отчетом таблице не составляет проблем, изменение структуры таблицы - это совсем другое дело. Если в Access 97 вы переименовывали поле в основной таблице, отчет, основанный на этой таблице с переименованным полем, не мог больше распознавать это поле. В Access 2002, если вы переименовываете поле в основной таблице, изменение отражается в отчете. Эта функциональная возможность была добавлена в Access 2000. Однако если вы уничтожаете поле в основной таблице, поле в отчете остается, но уже не распознается.

Основные функции отчетов

Если вы посмотрите на блоки, из которых построен отчет, то увидите, что основными составляющими являются разделы (sections) и элементы управления. Разделы организуют поток данных, обрабатывая одну запись в единицу времени. Итак, следует рассмотреть организацию как в пространстве, так и во времени. Например, участок нижнего колонтитула не будет обработан до тех пор, пока не будет напечатана последняя запись в отчете. Таким же образом запись первой группы не будет обработана до тех пор, пока не будет напечатан заголовок первой группы (подразумевается, что он один).

Более подробно об элементах управления см. в гл. 5.

Проще говоря, разделы помогут вам увидеть ваши данные в любом порядке и в любом формате, каком бы вы ни захотели. Вас никто не заставляет их использовать, но они могут гораздо больше, чем просто сделать ваш отчет действительно привлекательным. Они могут суммировать данные для промежуточных и конечных сумм (subtotals и grand totals). Вы также можете использовать их для создания титульных страниц, сносок и т. д.

Текстовое поле (text box) является наиболее важным элементом управления в отчете, поскольку это основной строительный блок записи отчета. Когда вы создаете отчет при помощи мастера отчетов, выбранные вами поля вставляются в виде текстовых полей. Когда вы вставляете в отчет номера страниц, даты или времена, вы также используете текстовые поля. Если вы проводите вычисления, итоги и промежуточные суммы, вы снова применяете текстовые поля. Другие типы элементов управления, которые интенсивно используются в отчетах, - это подписи, подчиненные формы, линии (lines), блоки(boxes) и рисунки (images).

Для более подробной информации об элементах управления отчета см. дальше в этой главе, раздел «Осуществление контроля за элементами управления отчета».

Расположение элемента управления в разделе определяет где и как часто он будет печатаеться в отчете. Например, если вы располагаете текстовое поле в нижнем колонтитуле группы (group footer section) отчета, вы можете оформить его в виде промежуточной суммы, используя такое выражение, как =Sum([UnitPrice]). Эта промежуточная сумма печатается после каждого группового раздела, когда вы запускаете отчет. Если вы помещаете дату в нижнем колонтитуле страницы (page footer section), она печатается один раз внизу каждой страницы.

Для более подробной информации о разделах см. дальше в этой главе раздел «Понимание важности разделов».

События помогают вам управлять как разделами, так и элементами управления. В зависимости от ситуации ваше приложение реагирует на события в отчете, которые генерирует Access или пользователь. Вы можете применять как события отчета, так и события разделов для создания ваших собственных процедур, которые обеспечат вам больший контроль и гибкость практически каждого объекта в отчете. Четыре события отчета (report events) происходят, когда пользователь производит такое действие над отчетом, как открытие или закрытие отчета. Другие три события, такие, как On No Data, генерируются приложением. Например, если источник записи не производит записи, срабатывает событие On No Data.

Напротив, события раздела (section events) генерируются до того, как Access форматирует или печатает каждую запись (в области данных, details section) или раздел (все другие разделы). Например, событие On Format происходит до того, как Access сформирует каждую запись отчета или раздел. Это означает, что вы можете указать Access, как форматировать каждую запись или раздел перед тем, как они будут напечатаны. Вы можете даже использовать события для того, чтобы напечатать условные сообщения в разделах. Например, вы можете показать или скрыть сообщение, поздравляющее продавца с вырученной прибылью, в зависимости от его продаж.

Для более подробной информации о событиях см. раздел «Подробнее про объектные события» в гл. 7

Два основных типа отчета - это *в столбец* (columnar) и *табличный*(tabular). Отчет в столбец подходит для всех типов форм заполнения стандартного бланка, таких, как заявления или счета. Они используются для печати результатов после того, как формы заполнены. Табличные отчеты выглядят скорее как широкоформатные таблицы, поскольку они обычно имеют структуру в виде строк и столбцов. На рис.6.1 показаны оба типа отчетов рядом.

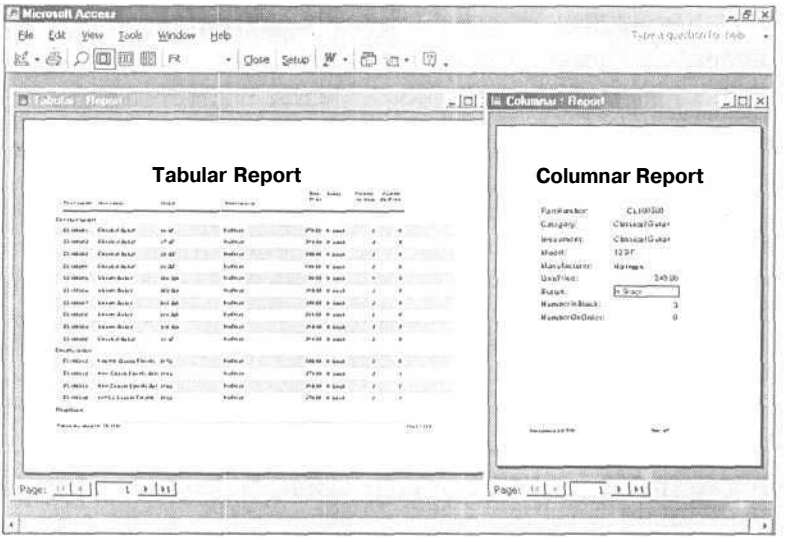


Рис. 6.1. Когда типы отчетов в столбец и табличный представлены рядом бок о бок, вы можете увидеть различия двух стилей

Создание отчета с самого начала

Вы можете создавать отчет различными способами, один из которых - это мастер отчетов, который действует в большинстве своем так же, как и мастер форм. Однако если ваш задуманный дизайн отчета не похож на большинство стандартных стилей отчетов, вам следует использовать метод построения отчетов с самого начала. Используя этот метод, вы начинаете построение с чистого листа и добавляете различные компоненты, такие, как разделы и элементы управления, вручную. Хотя этот метод более труден, вы многому можете научиться непосредственно в процессе работы.

Для того чтобы снарядить вас несколькими практически полезными средствами, давайте рассмотрим следующее упражнение, показывающее на примере, как строить отчет в столбец.

1. Откройте базу данных Music Store из папки AccessByExample.
2. В меню **Объекты** (Objects) зайдите в раздел **Отчеты** (Reports) для того, чтобы увидеть список отчетов, а затем щелкните мышью два раза на **Создание отчета в режиме конструктора** (Create Report in Design View) для того, чтобы открыть чистый бланк отчета как показано на рис. *6.2.*

   Панель инструментов на рис.6.2 похожа на панель инструментов в окне конструктора формы. Однако кнопку, следующую за кнопкой **Панель элементов** (Toolbox), вы не найдете на панели инструментов формы. Это кнопка **Группировка и сортировка** (Grouping and Sorting). Вы заметите также разделы **Верхний колонтитул** (Page Header) и **Нижний колонтитул** (Page Footer) в окне конструктора. Вы можете включить или отключить их, но если они включены, то все, что ни находится в них, будет выводиться один раз на

каждой странице. Слева от кнопки **Панель элементов** на панели инструментов находится кнопка **Список полей** (Field List). Обратите внимание, что она окрашена серым цветом, это означает, что клавиша неактивна.

Важно помнить, что где бы вы ни находились, в форме или в отчете, если у вас не установлен источник записей, кнопка **Список полей** (Field List) не будет активирована, что будет отражено серым цветом кнопки.

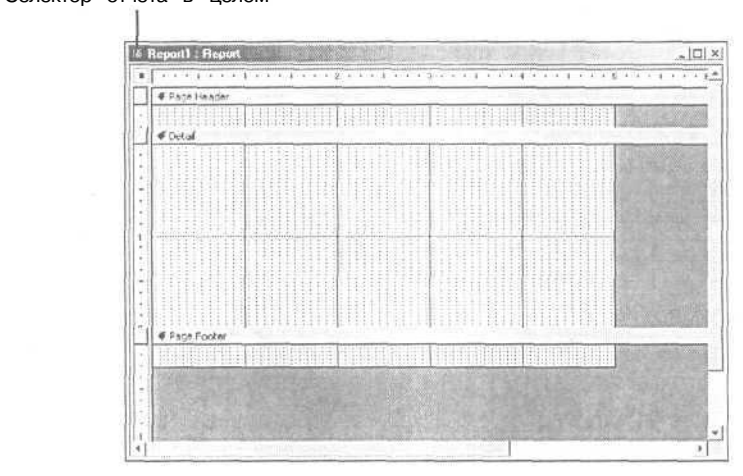


Рис. 6.2. Обратите внимание на панель инструментов и кнопку «группировка и сортировка» (grouping and sorting) в окне конструктора с пустым бланком отчета

ПОДСКАЗКА. Открыть список свойств можно и другим способом, нежели нажатием правой клавиши мыши: можно использовать кнопку **Свойства** (Properties) на панели инструментов. Помните, что используя этот способ, вы открываете список свойств того объекта, который находится под курсором. Например, если вы щелкнете по области выделения отчета (Report Selector, черный квадратик на пересечении двух линеек), а затем кнопку **Свойства** (Properties), это избавит вас от необходимости нажатия правой клавиши мыши перед выбором опции **Свойства.** Щелчок мышью либо по строке заголовка (Title bar), либо по области выделения отчета (Report Selector) (см. рис. 6.2), либо по серой области справа от отчета позволит вам выделить отчет.

3. Щелкните мышью по области выделения отчета (Report Selector), затем на **Свойства** (Properties) и выберите закладку **Данные** (Data), расположенную в списке свойств.
4. В поле **Источник записей** наберите `SELECT * FROM Inventory WHERE Inventory.NumberInStock>4;` и нажмите клавишу Enter. Откроется окно **Список полей,** при этом активируется кнопка **Список полей.** Теперь вы можете включать и отключать блок **Список полей** (Field List) используя одноименную клавишу.

5. После закрытия списка свойств, измените размер окна **Список полей** (Field List) (если необходимо), поместив курсор над одним из четырех краев поля и увидев, что он превратился в двойную стрелку. Затем нажмите клавишу мыши и, удерживая ее, растягивайте поле до тех пор, пока не увидите все пункты. Вы также можете двигать окно **Список полей** (Field List) путем перемещения его заголовка на новое место.
6. Нажмите на поле PartNumber (first). Удерживая нажатой клавишу Shift, щелкните мышью на поле Manufacturer, как показано на рис. 6.3, для того чтобы выбрать первые пять пунктов в **Списке полей** (Field List).

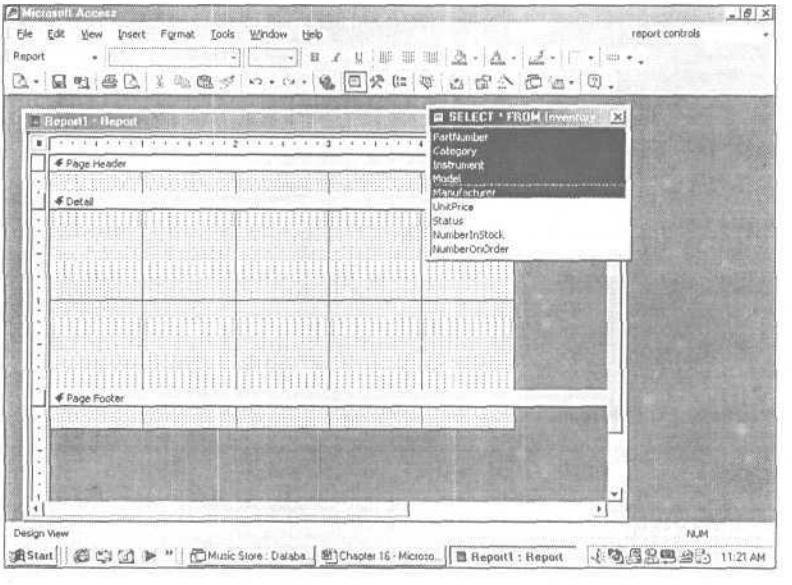


Рис. 6.3. Используйте **Список полей** (Field List) для выбора нескольких полей, которые могут быть вставлены в область данных

7. Поместите курсор над выделенными полями; затем перетащите их в нижнюю левую часть раздела области данных (Detail Section), как показано на рис. 6.4.
8. Щелкните мышью на поле Unit Price, а затем, удерживая клавишу Shift, на поле NumberOnOrder для того, чтобы выбрать эти пункты и все пункты, находящиеся между ними.
9. Поместите курсор на выделенные поля и щелкните мышью, затем, удерживая, потяните курсор в правую нижнюю сторону области данных, как показано на рис. 6.5.
10. Щелкните мышью на текстовом поле PartNumber, затем поместите курсор как можно ближе к верхней левой части буквы «P», пока курсор не превратится в руку с вытянутым указательным пальцем.
11. Щелкните мышью, и, удерживая, растяните текстовое поле PartNumber вправо до тех пор, пока подпись к PartNumber не покажется полностью.

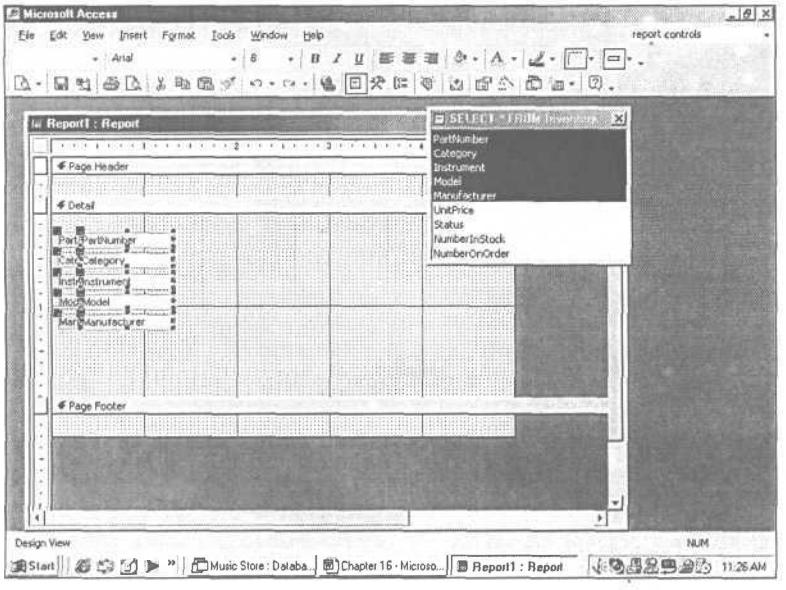


Рис. 6.4. Текстовые поля, вставленные в область данных, являются полями из подчиненного (underlying) запроса

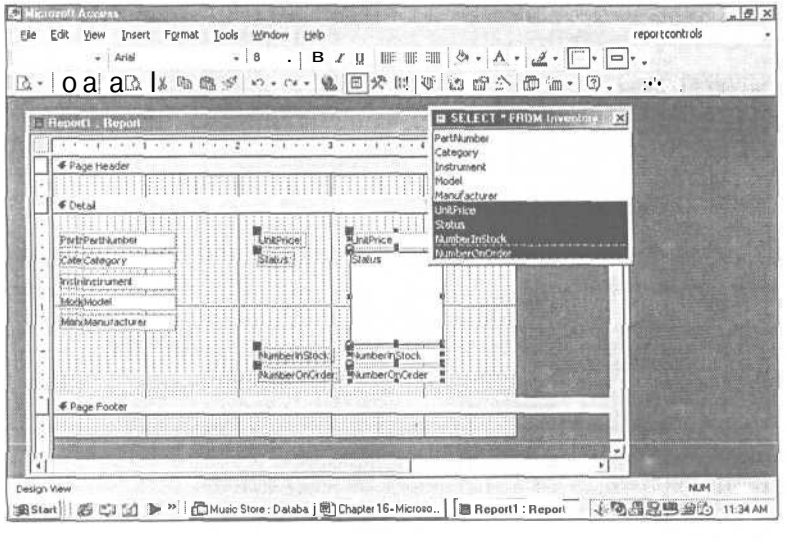


Рис. 6.5. Область данных заполнена второй группой полей из списка полей

12. Повторите шаг 11 для текстовых блоков **Категория** (Category), **Инструмент** (Instrument), **Модель** (Model) и **Производитель** (Manufacturer). Картинка на вашем экране должна быть похожа на рис. 6.6. Оставьте эту базу данных открытой.
13. Закройте окно **Список полей** (Field List).

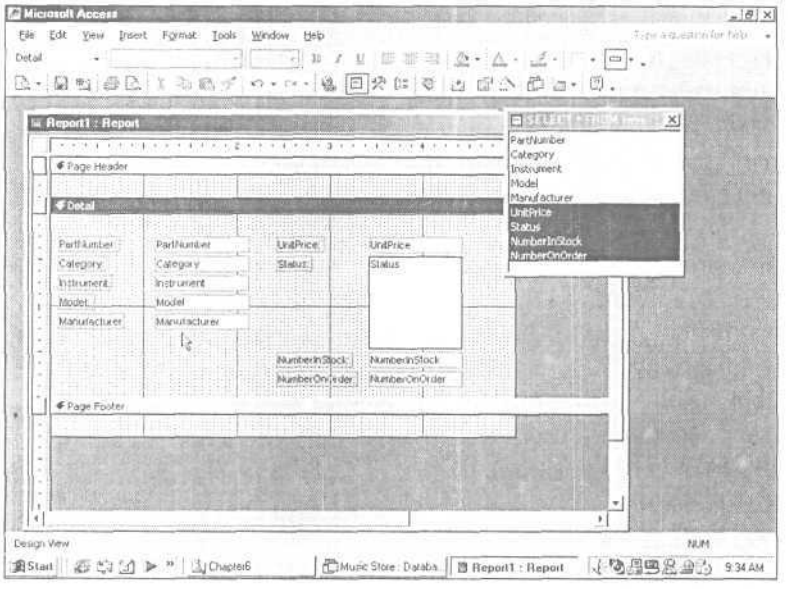


Рис. 6.6. Текстовые поля слева в области данных после их отделения от подписей к ним

Элементы управления для разделов верхнего колонтитула (Page Header) и нижнего колонтитула (Page Footer)

вы можете использовать разделы верхнего колонтитула и нижнего колонтитула для того, чтобы вставить элементы управления, которые будут повторяться на каждой странице. Например, если вы хотите, чтобы название отчета и дата были вверху каждой страницы вашего отчета, а время и номер страницы внизу, сделайте следующее:

1. Щелкните мышью на опции меню **Вставка** (Insert), **Дата и время** (Date and Time) для того, чтобы открыть диалоговое окно для установки даты и времени.
2. Отмените опцию **Формат времени** (Include Time). Это гарантирует, что вы получите только дату. Нажмите OK. Дата появится в области данных. Перетащите ее в правую часть раздела верхнего колонтитула (Page Header).

ПРЕДОСТЕРЕЖЕНИЕ. Если вы пользуетесь клавишей стрелка вверх для перемещения текстового окна вверх, это увеличит размер верхнего колонтитула (Page Header), но если вы перетаскиваете его с помощью мыши, размер раздела не изменится. Используйте курсор в виде руки вместо курсора в виде указательного пальца для перемещения по разделам при помощи мыши.

ПОДСКАЗКА. Вы также можете набрать =Date() в текстовом окне, чтобы получить тот же результат.

3. Нажмите кнопку **Панель элементов** (Toolbox) и выберите инструмент для установки подписей **(Надпись,** Label), который выглядит как прописная и строчная буквы «Аа», выделенные курсивом.

4. Для того чтобы нарисовать подпись слева в разделе верхнего колонтитула (Page Header), щелкните мышью и, удерживая, растяните из верхнего левого в правый нижний угол. Образуется небольшой прямоугольник.
5. Наберите Inventory Status Report в окне подписи. Оставьте окно выделенным для совершения следующего шага.
6. В ниспадающем окне **Размер шрифта** (Font Size) панели инструментов форматирования выберите 14. Из меню выберите **Формат** (Format), **Размер** (Size), **По размеру данных** (To Fit), чтобы подогнать размер поля к выбранному в панели инструментов. Картинка на вашем экране должна соответствовать рис. 6.7.
7. Щелкните мышью на меню **Вставка** (Insert), **Номера страниц** (Page Numbers) из панели инструментов. В диалоговом окне выберите опции **Страница N из M, Нижний колонтитул, выравнивание по правому краю, и Отображать номер на первой странице,** как показано на рис. 6.8. Нажмите OK.

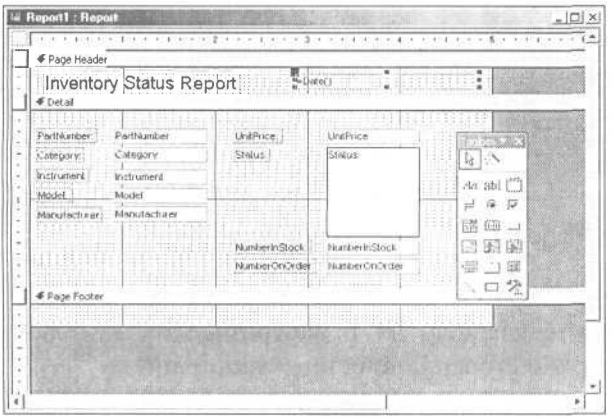


Рис. 6.7. Используйте раздел «Верхний колонтитул» для вставки и форматирования текущей даты и прочих атрибутов заголовка отчета

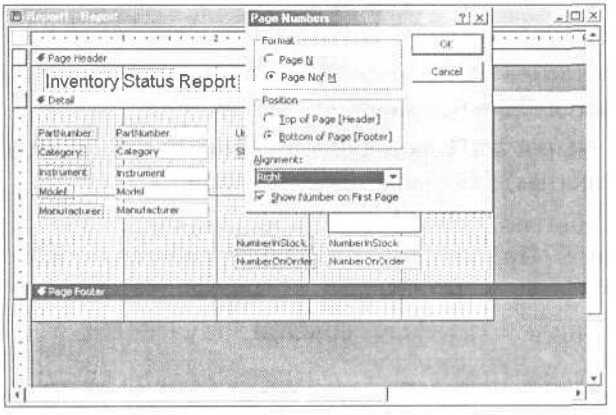


Рис. 6.8. Диалоговое окно **Номера страниц** (Page Numbers) поможет вам расположить и выровнять номер страницы в нижнем колонтитуле отчета

8. Вберите инструмент **Текстовое поле** (Text Box) из панели элементов (Toolbox) и нарисуйте текстовое поле слева в разделе нижнего колонтитула (Page footer). Щелкните мышью на подпись, прикрепленный к текстовому полю, и нажмите клавишу Delete, чтобы удалить его.
9. Наберите `="Report printed at: " & Time()` в текстовом окне (не забудьте поставить обе кавычки), как показано на рис. 6.9.

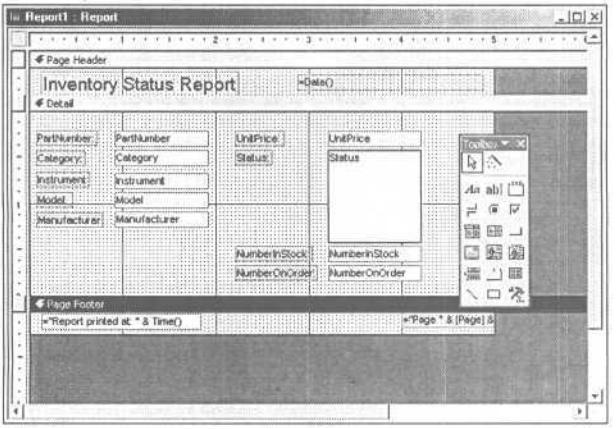


Рис. 6.9. Раздел Нижний колонтитул (Page Footer) - самое подходящее место для номеров страниц и даты или времени

10. Выбрав текстовое поле, нажмите клавишу F4 и наберите txtTime в поле свойств **Имя** (Name). Закройте окно Property, но оставайтесь в отчете.

Что делать, если не найдено никаких записей

Одной из наиболее распространенных проблем, с которой сталкиваются пользователи, - это взаимодействие с отчетами, которые не находят никаких записей. К счастью, событие On No Data дает вам способ разрешения этой проблемы.

1. Нажмите кнопку **Предварительный просмотр** (Print Preview). Убедитесь, что никаких данных не печатается. Закройте окно просмотра.
2. Щелкните правой клавишей мыши на серой области справа от отчета и выберите меню **Свойства** (Properties) для открытия листа свойств.
3. Щелкните мышью на опции **Событие** (Event) и выберите поле On No Data. Из ниспадающего меню выберите **Процедура обработки событий** (Event Procedure) и нажмите клавишу **Создать** (Build).
4. Вставьте следующий программный код между линиями Private Sub и End Sub:

```
MsgBox "No records were retrieved.", vbOKOnly
Cancel = True
```

Закройте окно **Модуль** (Module) и список свойств.

5. Снова нажмите **Предварительный просмотр** (Print Preview). Теперь вы должны увидеть сообщение как на рис. 6.10. Нажмите OK и закройте окно предварительного просмотра.

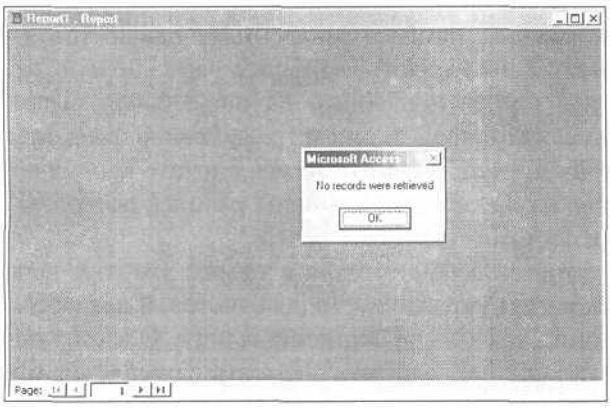


Рис. 6.10. Процедура, привязанная к событию On No Data, создает это сообщение

6. Щелкните правой клавишей мыши на области выделения отчета (Report Selector), выберите **Свойства,** и выберите вкладку **Данные** (Data).
7. Щелкните мышью на свойстве **Источник записей** (Record Source) и нажмите клавишу **Создать** (Build). Обратите внимание, что вы находитесь в окне конструктора запросов (Query Design).
8. В строке **Условие отбора** (Criteria) измените >4 на >2. Закройте построитель запроса (Query Builder) и нажмите **Yes,** чтобы сохранить и закрыть запрос; затем закройте список свойств.
9. Нажмите снова Print Preview. На этот раз вы должны увидеть данные. Обратите внимание, что поле Status является элементом «список». Закройте окно предварительного просмотра.
10. Щелкните правой клавишей мыши на текстовом поле Status (возможно, опечатка: в предыдущем абзаце говорилось, что этот элемент управления - список. - *Пер.)* и выберите **Преобразовать элемент в** (Change To), **Поле** (Text Box). Нажмите в последний раз Print Preview и закройте окно предварительного просмотра.
11. Щелкните мышью на поле NumberInStock, затем щелкните, зажав клавишу Shift, на поле NumberOnOrder.
12. Поместите курсор поверх полей. Когда курсор превратится в руку, переместите поля NumberlnStock и NumberOnOrder прямо под полем Status.
13. Расположите курсор над верхним краем раздела нижнего колонтитула (Page Footer). Когда курсор превратится в двойную стрелку, щелкните клавишей и, удерживая, растяните раздел прямо под поле **Производитель** (Manufacturer).
14. Закройте и сохраните отчет как Inventory Status Report.

Контроль над элементами управления отчетом

В отличие от элементов управления формой вы не можете вводить данные в элементы управления отчетом. Поэтому они используются в демонстрационных целях. Даже когда вы применяете элементы управления отчетом для вычисления или манипулирования информацией, это все равно имеет целью только показ данных. Обычно текстовые поля и подписи - наиболее часто используемые элементы управления отчетом. Ориентированные на визуальный вывод данных элементы управления, такие, как блоки, линии и графические элементы управления, являются также полезными, когда вы пытаетесь сделать ваш отчет эстетически приятным. В этой главе на примерах объяснены методы, позволяющие воспользоваться всеми возможностями текстовых полей.

Панель элементов для форм и отчетов в общем одна и та же. Это означает, что вы используете те же самые элементы управления и для отчетов, и для форм, хотя есть отличия как в конструкции, так и в функционировании. Одно из основных отличий вы заметите, когда будете сравнивать элементы управления отчетами с элементами управления формами - это отсутствие событий (events), хотя и имеется в наличии вкладка События (Events). Существуют другие способы программирования и обращения к элементам управления в отчете. Например, когда вы щелкаете правой кнопкой мыши на различных разделах отчета, вы получаете доступ к вкладке События для обращения к элементам управления в этом разделе. Вы также можете использовать возможности условного форматирования, подробно описанные дальше в этой главе, для форматирования элементов управления. И если этого недостаточно, вы можете использовать события в свойствах отчета, для того чтобы прилагать программы к элементам управления отчетом.

Есть еще один способ управления элементами управления в отчете. В качестве источника данных (Control Source) вы можете использовать имя другого элемента управления (Control Name). Это простой способ произведения вычислений в отчете, подобный электронным таблицам. Давайте рассмотрим этот метод на примере.

Произведение вычислений подобно электронным таблицам в элементах управления отчетом

Существует несколько способов набора выражений в элементах управления отчетом. Вы можете щелкнуть правой клавишей мыши для получения доступа к свойству источника данных (Control Source) в списке свойств (Property), или вы можете щелкнуть мышью прямо в текстовом поле и начать печатать. Вы, должно быть, удивитесь: «Почему бы просто не набирать каждый раз формулу прямо в окне?» Этот метод правилен, но необходимо учесть, что вам все равно придется щелкнуть правой клавишей мыши на элементе управления для того, чтобы проверить другие свойства. Кроме того, нажатие правой клавиши мыши на элементе управления предлагает различные опции для придания лучшего вида вашим формулам.

Существует два способа обращения к элементам управления в отчете. Вы можете обратиться к ним по их имени или по их источнику данных, который обычно является именем поля. Если источник данных вдруг является выражением, то вы должны использовать имя для обращения к этому элементу управления. Это дает вам преимущество, обеспечивая механизм настройки вычислений, подобный электронным таблицам. Следующие шаги покажут вам, как это сделать.

1. Убедитесь, что база данных Music Store открыта с предыдущего примера.
2. В меню раздела **Объекты** (Objects) выберите графу **Отчеты** (Reports). В ней выберите отчет Invoice Practice report и затем нажмите **Конструктор** из панели инструментов базы данных.
3. Разверните окно отчета на весь экран и используйте вертикальную полосу прокрутки для прокрутки вниз. Обратите внимание на три подписи (labels): Sub Total (Предварительный итог), Tax (Налог) и Total (Итог) соответственно.
4. Щелкните правой клавишей мыши на текстовом поле, расположенном после подписи Sub, Total и выберите **Свойства** (Properties) для открытия списка свойств.
5. Наберите `=sum(ExtendedPrice)` в свойстве источника данных **(Данные,** Control Source). Переместите курсор вниз и обратите внимание, как автоматически вставляются скобки и надпись превращается в `=sum([ExtendedPrice])`, как показано на рис. 6.11.

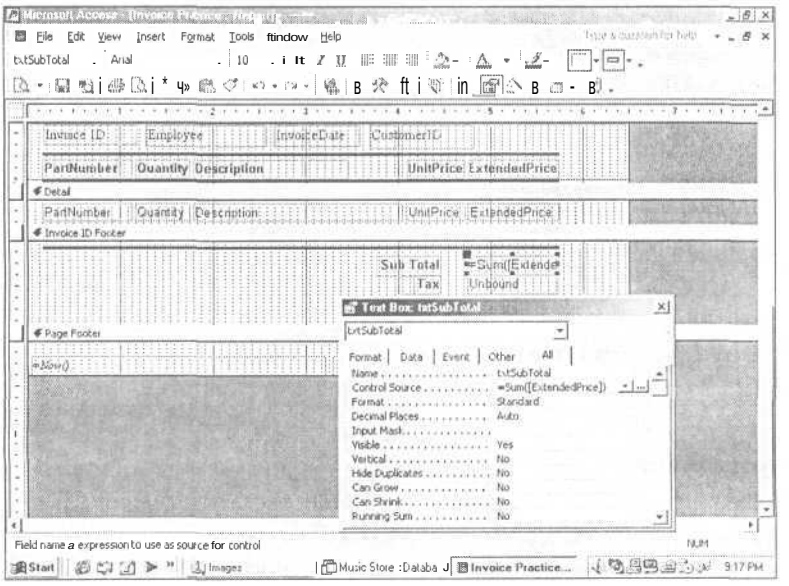


Рис. 6.11. Вы можете написать выражение прямо в свойство Control Source, которое ссылается на источник данных другого поля

Вы обращаетесь к полю ExtendedPrice путем обращения к его источнику данных. Имя элемента управления txtExtension, а не ExtendedPrice. Выражение =sum(ExtendedPrice) используется с целью получения итога для поля ExtendedPrice.

ПОДСКАЗКА. Соглашением Венгерской конвенции наименований было решено помещать приставку txt перед именем элемента управления для того, чтобы отличать имена, определяющие текстовые поля. Для поля со списком (combo box) используется приставка cbo и т. д. Когда вы видите приставку txt перед управляющим именем, вы сразу же понимаете, какого рода этот элемент управления. Вы также знаете, что это не имя поля. Access не заставляет вас пользоваться принятыми наименованиями, но применение их в такого рода ситуациях имеет смысл. Помните, что, когда вы создаете форму или отчет, используя мастера, созданные этой программой элементы управления следует приводить к принятому конвенцией виду вручную.

6. Находясь в строке источника данных **(Данные,** Control Source), щелкните мышью по текстовому полю, следующему за подписью Tax. (вам, возможно, понадобится передвинуть окно свойств, в котором вы находитесь, для того, чтобы увидеть текстовое поле.) Теперь наберите =txtExtension * .08 в свойстве Control Source и нажмите стрелку вниз (рис. 6.12).

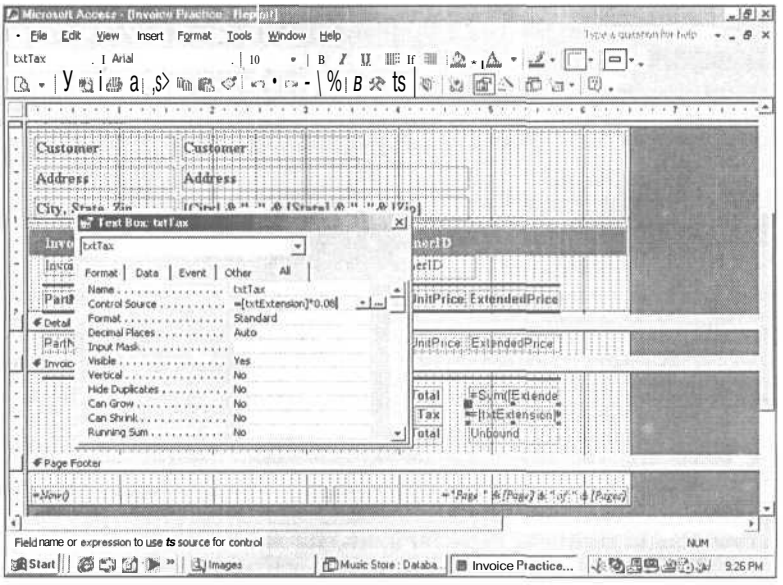


Рис. 6.12. Вы можете использовать имя элемента управления (Control Name) для ссылки на другой элемент управления в выражении

7. Закройте окно свойств (Property) элемента управления.

8. Теперь щелкните клавишей мыши прямо на текстовом поле (вероятно, имеется в виду поле Total. - *Пер.*). (После того как вы медленно щелкните дважды в текстовом поле, надпись Unbound должна исчезнуть.) Наберите в окне `=txtSubTotal + txtTax`.

ПОДСКАЗКА. Вы также можете использовать кнопку **Создать** (Build) свойства источника данных (Control Source) для построения ваших выражений. Это делается так: щелкните мышью на неприсоединенном (unbound) элементе управления. Затем нажмите кнопку **Свойства** (Properties). Потом нажмите кнопку Build напротив свойства **Данные** (Control Source). Откроется построитель выражений (Expression Builder) (рис. 6.13.)

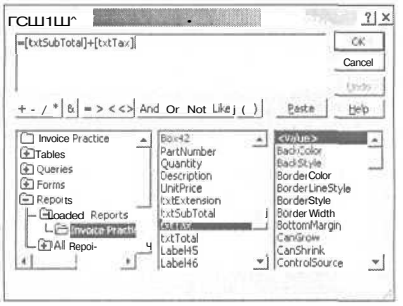


Рис. 6.13. Вы можете использовать Expression Builder для построения ваших выражений

В этом упражнении вы открыли для себя два способа обращения к элементам управления. Второй метод, ссылка на имя (Control Name), а не на источник данных (Control Source), похож на способ адресации ячейки в электронной таблице. Набрать `=txtSubTotal + txtTax` проще, чем `=sum([ExtendedPrice]) + sum([ExtendedPrice]) *.08` для элемента управления txtTotal. Если бы формула была еще сложнее, вы бы получили еще больше выгоды.

9. Убедитесь, что вы отменили все операции с элементами управления и щелкните мышью кнопке **Предварительный просмотр** (Print Preview) для того, чтобы увидеть отчет. Щелкните мышью на области, содержащей итоговую сумму, для изменения размеров этой области. Закройте окно предварительного просмотра.
10. Сохраните и закройте отчет.

Управление сортировкой и группировкой

Одной из основных нужд для подведения итогов в отчете является группировка схожих величин. Например, если бы вы занимались страхованием, вас могли бы заинтересовать все иски за определенный год. Если бы вы занимались продажами, вы могли бы заинтересоваться общим объемом продаж по региону. Группировка дает вам эту возможность. Если вы группируете по штатам, вы можете захотеть рассортировать эти группы. Внутри каждой группы

для одного штата вы можете рассортировать почтовые индексы (ZIP codes). Это многоуровневая сортировка, которую можно рассматривать как сортировку внутри сортировки.

Если вы хотите сгруппировать все значения для определенного года в запросе, вы, очевидно, столкнетесь с проблемой, если дата является полем, по которому нужно проводить группировку. У вас не будет схожих значений в этом поле, поскольку даты будут распределены по всему году. Существует простой способ решения такой дилеммы. Вы можете создать вычисляемое поле в запросе, которое бы выглядело так: SalesYear: Year([DateField]); затем сгруппируйте это поле и привяжите запрос к отчету. Это так просто! По этому же принципу строится сортировка по месяцу; просто используйте функции месяца вместо функций года.

Есть еще хорошие новости. Вам не нужно создавать запрос с вычисляемым полем для группировки поля дат в отчете. Когда вы группируете по любой дате в отчете, Access имеет возможность группировки по месяцам, кварталам, годам и т. д.

Для получения дополнительной информации о группах см далее в этой главе раздел «Создание групп в отчетах».

Для того чтобы узнать больше о группировке данных по временным диапазонам, см. раздел «Использование запроса на примере таблицы для конструкций 'If-Then'» в гл. 16.

Создание групп в отчетах

Перед тем как вы попробуете создать группы на примере, взгляните на опции, предлагаемые в диалоговом окне сортировки и группировки (Sorting and Grouping), как показано на рис. 6.14.

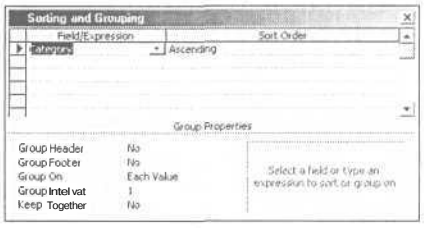


Рис. 6.14. Вы можете устанавливать опции группировки и сортировки, используя диалоговое окно Sorting and Grouping

Следующие четыре таблицы (которые вы можете использовать позже для справочных целей) предлагают обзор функциональных возможностей диалогового окна Sorting and Grouping. В табл. 6.1 объяснен раздел «Свойства группы» (Group Properties) диалогового окна сортировки и группировки (Sorting and

Grouping) путем объяснения вариантов выбора для различных свойств; в табл. 6.2 объяснен пункт **Группировка** (Group On); в табл. 6.3 разобрана опция **Интервал** (Group Interval) и ее взаимодействие с Group On; табл. 6.4 показывает, что означают различные опции в меню «Не разрывать» (Keep Together).

Таблица 6.1. Пункты диалогового окна Sorting and Grouping

Свойство	Действие	Варианты
Заголовок группы (Group Header)	Включает или выключает верхний колонтитул группы	Да, Нет
Примечание группы (Group Footer)	Включает или выключает нижний колонтитул группы	Да, Нет
Группировка (Group On)	Определяет значение для группировки	См. табл. 6.2
Интервал (Group Interval)	Работает со свойством Group On	См. табл. 6.3
Не разрывать (Keep Together)	Сохраняет части группы (включая разделы) на одной странице	См. табл. 6.4

Таблица 6.2. Свойство «Группировка» (Group On)

Тип данных	Параметр	Группируемые им записи
Текст	Каждое значение	С одинаковым значением в поле или выражении
	Префиксный символ	С одинаковым n (числом) символов в поле или выражении
Дата/время	Каждое значение	С одинаковым значением в поле или выражении
	Год	Одного года
	Квартал	Одного квартала
	Месяц	Одного месяца
	Неделя	Одной недели
	День	Одного дня
	Час	Одного часа
	Минута	Одной минуты
Счетчик, деньги, число	Каждое значение	С одинаковым значением в поле или выражении
	Интервал	Со значениями в указанном интервале

Таблица 6.3. Свойство «Интервал» (Group Interval)

Тип поля данных	Варианты опции Group On	Установки опции Group Interval
Все	Каждое значение	Равно 1
Текст	Префиксные символы	Равно 3 для группировки по первым трем символам (например, Манчестер, Манфорд и Мансфилд были бы объединены в группу)
Дата/время	Неделя	Равно 2 для группировки данных за две недели
Дата/время	Час	Равно 12 для группировки данных за 12 ч

Таблица 6.4. Свойство «Не разрывать» (Keep Together)

Параметр	Описание
Нет (No)	Печатает группу без сохранения заголовка группы, области данных и нижнего колонтитула группы на одной странице
Вся группа (Whole Group)	Печатает заголовок группы, область данных и нижний колонтитул группы на одной странице
С первыми данными (With First Detail)	Печатает заголовок группы на странице, а также, если может, первую запись области данных

Например, управление музыкальным магазином решило добавить итоговые суммы в отчет Inventory. Теперь, поскольку у вас есть шанс изучить диалоговое окно Sorting and Grouping, вы можете исследовать его применение на примере. Следующий пример поможет вам в создании групп в отчетах, обучая по ходу работы, как сортировать и размещать элементы управления внутри группы.

1. Убедитесь, что база данных Music Store открыта.
2. В меню Reports щелкните мышью на опции Inventory Practice Report, а затем на **Конструкторе,** для того чтобы открыть отчет в виде конструктора.
3. Щелкните правой клавишей мыши на разделе отчета **Верхний колонтитул** (Page Header) и выберите опцию **Сортировка и группировка** (Sorting and Grouping) из выпавшего меню или нажмите кнопку Sorting and Grouping на панели инструментов для открытия диалогового окна.
4. Щелкните по ниспадающему списку справа в столбце **Поле/выражение** Field/Expression и выберите поле Category. Откроются свойства группы (Group Properties).
   Обратите внимание, что значение «По возрастанию» автоматически вставляются в колонку **Порядок сортировки** (Sort Order).

5. Щелкните по ниспадающему списку в строке **Заголовок группы** (Group Header) в разделе свойств группы (Group Properties) и выберите **Да** (Yes). Вы можете также щелкнуть дважды на строке Group Header для выбора Yes.
   Обратите внимание, что заголовок раздела **Категория** (Category) вставлен в ваш отчет, как показано на рис. 6.15. Сделайте то же самое для опции «Примечание группы» (Group Footer). Это поместит раздел группы в ваш отчет, в котором вы можете разместить элементы управления в зависимости от того, в каком разделе вы находитесь. Например, если вы находитесь в разделе заголовка, вы можете разместить идентификатор (наподобие текстового окна) для разделения и идентификации групп. Если вы находитесь в разделе нижнего колонтитула, вы можете разместить ярлыки, текстовые окна или вычисляемые поля (наподобие частных итогов). Конечно, вы можете размещать линии, блоки или графические объекты в любом разделе.

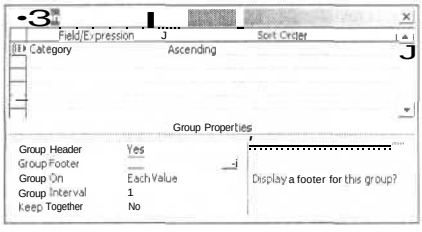


Рис. 6.15. Когда вы выбираете параметр Yes в свойствах Примечания группы (Group Header), новый раздел вставляется в ваш отчет

6. Вызовите ниспадающий список во втором ряду колонки **Поле/выражение** (Field/Expression) и выберите опцию **Статус** (Status).
7. Вызовите ниспадающий список в колонке **Порядок сортировки** (Sort Order) в ряду **Статус** (Status) и выберите **По убыванию** (Descending).
   Статус рассортирован в порядке уменьшения, поскольку имеет смысл увидеть в каждой группе все пункты по порядку; но, поскольку буква «i» для in stock (на складе) стоит перед буквой «o» для on order (под заказ), соответствующий порядок сортировки будет обратным или восходящим.
8. Закройте диалоговое окно **Группировка и сортировка** (Grouping and Sorting).
9. Выберите раздел **Заголовок группы 'Категория'** (Category Header).
10. Активируйте инструментальную панель (Toolbox) и выберите инструмент **Поле** (Text Box). Ваш курсор превратится в текстовое поле с надписью ab внутри и знаком плюс, как показано на рис.6.16.
11. Щелкните мышью на правом верхнем углу раздела **Заголовок группы 'Категория'** (Category Header). Вы также можете нажать клавишу мыши и, удерживая, растянуть область для того, чтобы нарисовать окно, но в этом случае проще щелкнуть мышью на нужной области и позволить Access нарисовать ее.
12. Щелкните на подписи, прикрепленной к тестовому полю, и нажмите клавишу Delete для того, чтобы удалить ее. (Если подпись не видна, передвиньте текстовое поле вправо.)

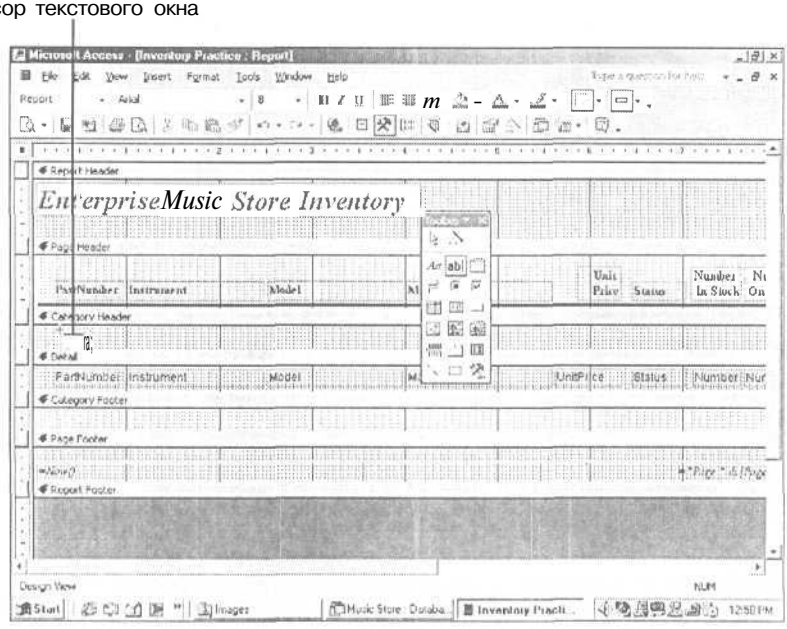


Рис. 6.16. Курсор превратится в текстовое окно, когда вы выберете инструмент Поле (Text Box) из панели элементов (Toolbox)

13. Щелкните правой клавишей мыши на несвязанном (unbound) текстовом окне и выберите **Свойства** (Properties) из выпавшего меню. Нажмите клавишу **Все** (АИ).
14. Вызовите ниспадающее окно в строке источника данных (Данные, Control Source) и выберите поле Category.
15. Наберите `txtCategory` в окне свойства **Имя** (Name). Закройте окно свойств.
16. Убедитесь, что текстовое окно все еще выбрано, и выберите из меню **Формат** (Format), **Размер** (Size), **По размеру данных** (To Fit). Ваше текстовое окно теперь должно идеально подходить тексту. Также нажмите кнопку **Полужирный** (Bold).
17. Закройте панель элементов.
18. Переместите текстовое окно Category, удерживая нажатой клавишу Ctrl и нажимая клавишу **левая стрелка** на клавиатуре до тех пор, пока поле не окажется слева от текстового поля Part number в области данных (Detail Section) и даже за линией, которая отделяет подписи от полей в разделе **Верхний колонтитул** (Page Header), как показано на рис. 6.17.
19. Нажмите кнопку **Предварительный просмотр** (Print Preview). Теперь вы легко можете быстро переместиться в любую область отчета щелкнув на ней клавишей мыши, как показано на рис. 6.18.
20. Нажмите кнопку **Закрыть** (Close) в окне предварительного просмотра (Print Preview), но оставайтесь в отчете.

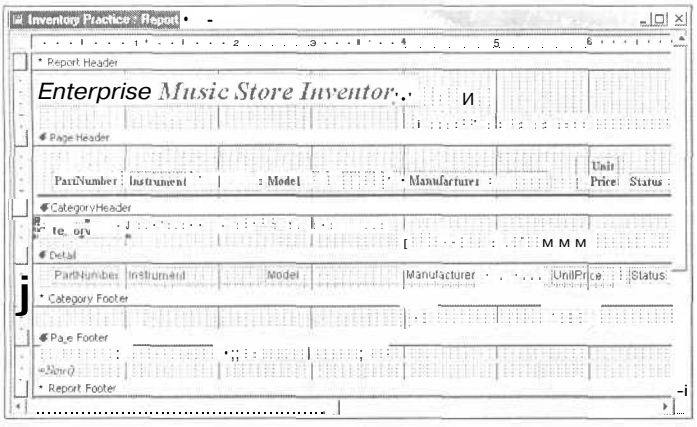


Рис. 6.17. Правильное положение поля Category после форматирования и перемещения

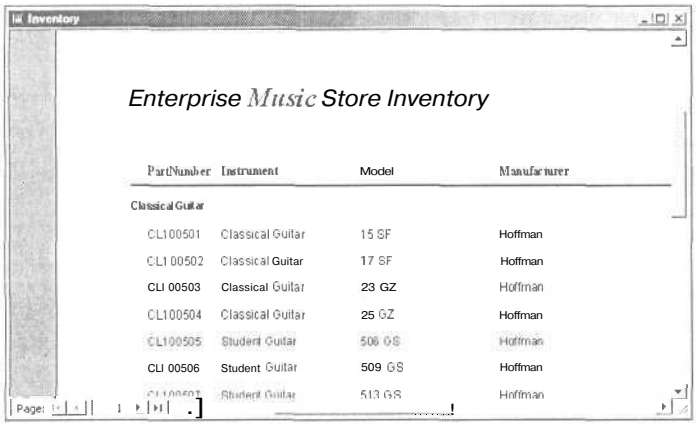


Рис. 6.18. Предварительный просмотр печати (Print Preview) показывает, как с помощью поля Категория (Category) разделить группы

Подведение итогов в группах

Обычной задачей для пользователей является подведение итогов для групп. Например, если вы хотите сгруппировать товары по категориям, вам нужно разместить элементы управления в разделе группы, суммирующие каждую категорию. В следующем примере описан процесс подведения итогов.

1. Нажмите на горизонтальную полосу прокрутки отчета, находящуюся в нижней части окна конструктора, пока не увидите подпись Extended Price (Общая цена), как показано на рис. 6.19.
2. Щелкните на несвязанном (unbound) текстовом поле в области данных под подписью. Убедитесь, что появились ручки для растягивания (handles, черные квадратики на границе элемента управления. - *Пер.*). Снова щелкните выбранное поле. (Возможно, это может потребовать небольшой практики, но

слово Unbound должно исчезнуть, позволяя начать печатать с самой левой позиции курсора).

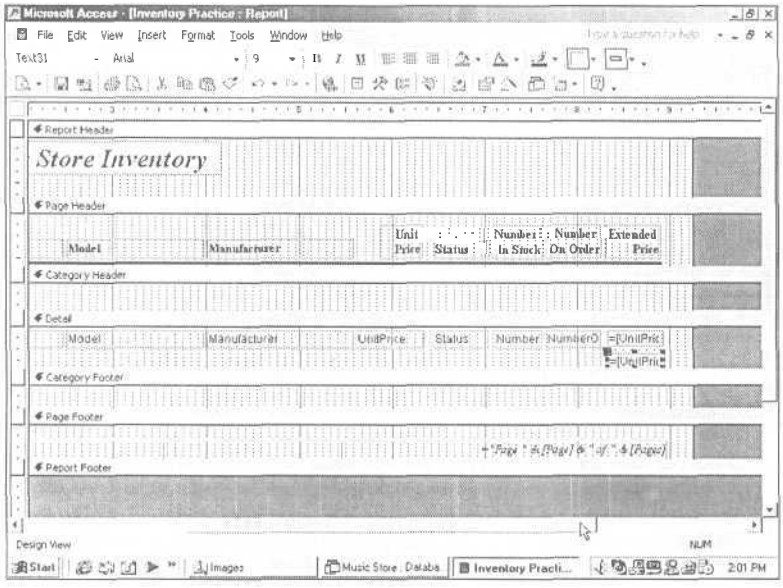


Рис. 6.19. Нажмите на горизонтальную полосу прокрутки, чтобы увидеть подпись Extended Price

3. Наберите - UnitPrice* NumberInStock и нажмите Ввод (Enter). Не следует печатать скобки, поскольку они ставятся автоматически, если имена полей написаны правильно. Посредством этой команды вы перемножаете поля UnitPrice и NumberlnStock ссылаясь на источники данных (Control Source) обоих текстовых полей.
4. Щелкните правой клавишей мышки на вычисляемом поле из шага 3 и выберите пункт **Свойства** (Properties). Выберите вкладку **Макет** (Format) и выберите строку **Формат поля** (Format), затем в ниспадающем списке выберите опцию **Стандарт** (Standard).
5. Закройте меню свойств.
6. Снова нажмите **Предварительный просмотр** (Print Preview). Обратите внимание на то, что там присутствует Extended Price.
7. Закройте предварительный просмотр печати.
8. Щелкните мышью на текстовом поле в области данных под подписью ExtendedPrice, если оно еще не выделено.
9. Нажмите комбинации клавиш Ctrl+C и Ctrl+V для копирования и вставки текста. Раздел должен расшириться для того, чтобы подстроиться к новому элементу управления, как показано на рис. 6.20.
10. Переместите ваш курсор в нижнюю часть копируемого текстового окна, подождите, пока курсор превратится в символ открытой руки.

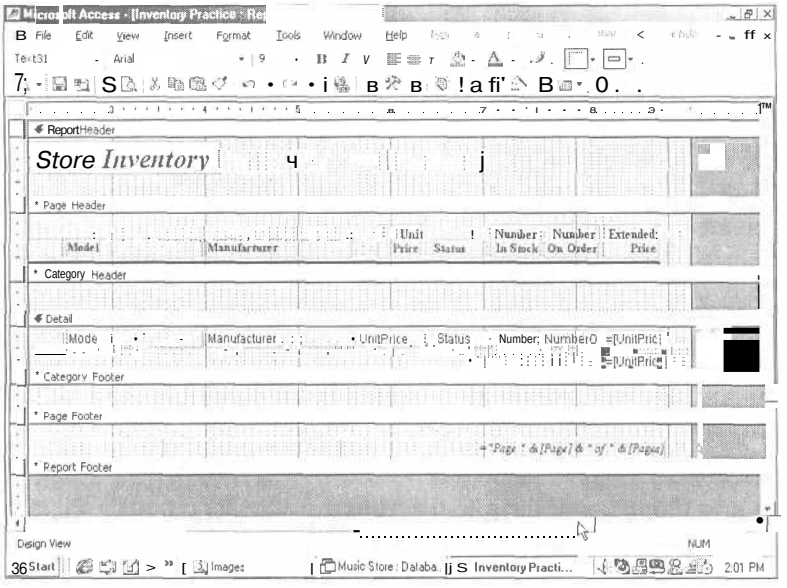


Рис. 6.20. Наиболее быстрый способ создать элемент управления - это просто скопировать и вставить созданный ранее

11. Нажмите клавишу и, удерживая, переместите окно вниз в раздел **Примечание группы 'Category'** (Category Footer) прямо под первоначальным текстовым полем.

    Вы также можете щелкнуть правой кнопкой мыши на окне, выбрать опцию **Копировать** (Copy) из меню быстрого вызова, щелкнуть правой клавишей на разделе **Примечание группы 'Category'** (Category Footer) и выбрать **Вставить** (Paste). Помните, что элемент управления при использовании этого метода будет располагаться в левом верхнем углу того раздела, в который вы собираетесь его копировать.

12. Нажмите клавишу и, удерживая, переместите верхнюю часть раздела **Примечание группы 'Category'** (Category Footer) в ее первоначальную позицию.

13. Выберите копируемое окно и зайдите в него.

    Наберите `=Sum([UnitPrice]*[NumberInStock])`. Вы добавляете функцию суммирования Sum к предыдущей формуле. Нажмите клавишу Enter для подтверждения новой формулы.

ЗАМЕЧАНИЕ. Если, находясь в окне **Свойства** (вы можете просто щелкнуть мышью на кнопке **Свойства** при выделенном элементе управления), вы захотите увидеть больше текста в вычисляемом поле не нажимая клавишу **Создать** (Build), вы можете просто щелкнуть правой клавишей на свойстве источника данных **(Данные,** Control Source) и выбрать опцию **Масштаб** (Zoom). Эта опция позволяет увидеть больше текста, чем кнопка Build. Кроме того, эта процедура работает на некоторых свойствах, которые не имеют кнопки Build.

14. Выберите инструмент **Надпись** (Label) из панели элементов для создания подписи в разделе **Примечание группы 'Category'** (Category Footer) под текстовым полем **Производитель** (Manufacturer). Нарисуйте окно подписи прямо под текстовым полем. Наберите Totals в окне и нажмите Enter.
15. Выберите текстовое поле Manufacturer, а затем нажмите на панели инструментов кисточку **Формат по образцу** (Format painter). Щелкните на подписи Totals для копирования формата.

ЗАМЕЧАНИЕ. Если вам нужно форматировать более одного объекта, щелкните дважды на кнопке Format Painter и форматируйте столько объектов, сколько вам надо. Нажмите на нее снова, когда понадобится отключить эту опцию.

16. Выбрав подпись Totals, нажмите клавишу **Полужирный** (Bold), чтобы выделить полужирным шрифтом подпись.
17. Выберите текстовое поле в области данных под подписью UnitPrice.
18. Нажмите Ctrl+C и Ctrl+V для копирования и вставки текстового поля.
19. Переместите курсор в нижнюю часть скопированного поля и подождите, пока курсор превратится в значок открытой руки.
20. Нажмите и, удерживая, переместите поле вниз в раздел **Примечание группы 'Category'** (Category Footer) прямо под первоначальным текстовым полем.
21. Нажмите и, удерживая, передвиньте верхнюю часть раздела **Примечание группы 'Category'** (Category Footer) в его первоначальное положение.
22. Наберите `=Sum([UnitPrice])` в текстовом поле. Нажмите Enter для подтверждения новой формулы.
23. Оставляя выделенным текстовое поле, содержащее это выражение, нажмите клавишу Shift и щелкните текстовое поле UnitPrice. Теперь, когда у вас выделены оба поля, одно прямо под другим, выберите **Формат** (Format), **Выровнять** (Align), **По правому краю** (Right) из меню, как показано на рис. 6.21.
24. Выберите оба текстовых поля щелчком мыши с удержанием клавиши Shift под ярлыком Extended Price и нажмите **Формат** (Format), **Выровнять** (Align), **По правому краю** (Right) в меню.

ПОДСКАЗКА. Вы также можете выбрать элементы управления щелчком на линейке разметки прямо над ними (найдите черную стрелку), чтобы выбрать вертикально стоящие поля, и слева от них, чтобы выбрать горизонтально стоящие поля. Другой вариант - это просто щелкнуть мышью и, удерживая, передвинуть при помощи курсора выбора объектов (Select Objects cursor) из пустой области чуть выше и левее выбираемых объектов в область чуть ниже и правее (обычное натягивание рамки. - *Пер.*). Обратите внимание на рамку вокруг текстовых окон на рис. 6.22. Когда вы отпустите клавишу мыши, объекты будут выбраны.

25. Нажмите кнопку **Предварительный просмотр** (Print Preview) для просмотра отчета. Закройте Print Preview, сохраните отчет как Grouping Example, затем закройте его.

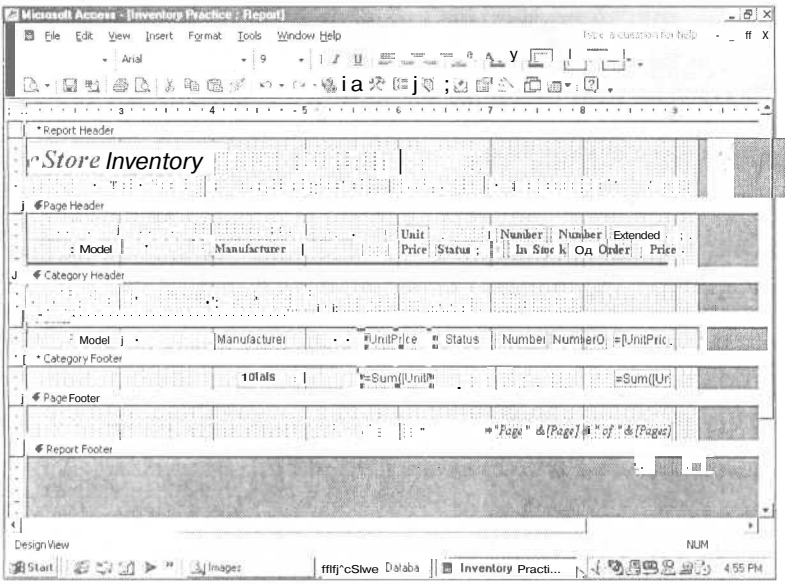


Рис. 6.21. Вы можете сделать выравнивание текста отчета по правому краю, чтобы быть уверенными в том, что числа будут выровнены в линию

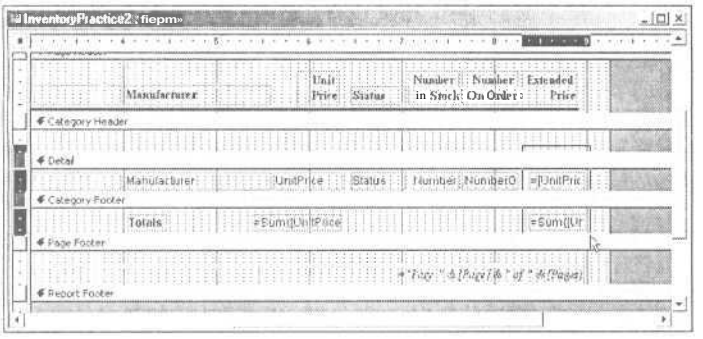


Рис. 6.22. Рисование рамки вокруг текстовых окон при помощи курсора выбора объектов (Select Objects cursor) (указатель в виде стрелки) производит выбор заключенных в нем объектов после того, как вы отпустите клавишу мыши

Понимание важности разделов

Стоит запомнить такую вещь: вы не обязательно должны использовать каждый тип раздела в каждом отчете. Кто-то может даже удивиться: «Для чего их вообще использовать?» Важно понять, что правильное понимание того, для чего нужны разделы, чрезвычайно необходимо, чтобы пользователь смог должным образом организовать свой отчет. Если вы хотите, чтобы титульная страница печаталась *только один* раз в начале отчета, используйте раздел отчета (report section). Но если вы хотите, чтобы оглавления полей печатались на каждой стра-

нице, используйте раздел «Верхний колонтитул» (page header section). Группы позволят вам отделить каждую группу общих значений некоторым пространством друг от друга, что может быть использовано для подведения итогов или других обобщенных вычислений.

В табл. 6.5 представлен пример того, что вы можете сделать при помощи разделов.

Таблица 6.5. Управляющие разделы

Раздел	Когда печатается	Примеры использования
Заголовок отчета (report header)	Один раз в начале отчета	Заголовок отчета, логотип компании, титульный лист, дата
Верхний колонтитул страницы (Page header)	Один на страницу	Заголовок страницы, оглавления полей, заголовок отчета, номер страницы, дата
Заголовок группы	Один на группу	Заголовок группы
Область данных (Detail Section)	Каждая запись	выделенные поля
Примечание группы (Group footer)	Один на группу	Промежуточные суммы, подсчеты, вычисления среднего и т. д.
Нижний колонтитул страницы (Page footer)	Один на страницу	Номер страницы, имя отчета, дата
Нижний колонтитул отчета	Один раз в конце отчета	вычисление полного итога, заметки

Некоторые важные свойства являются общими для каждого раздела, за исключением раздела страницы. Последующее обсуждение прояснит в некотором роде этот вопрос.

Свойство «Не разрывать» (Keep Together)

Предположим, записи в определенной группе начинают печататься в направлении сверху вниз на текущей странице, причем разместить их всех на одной странице невозможно. Когда это свойство настроено соответствующим образом, то вы получаете гарантию, что записи, принадлежащие одной группе, не будут расщеплены на две страницы. Раздел группы имеет два места для установки свойства «Не разрывать» (Keep Together). Поскольку значением по умолчанию для свойства Keep Together является Yes в разделе группы, вы должны удостовериться, что свойство «Не разрывать» в окне «Сортировка и группировка» установлено для «Всей группы» (Whole Group). В противном случае группа так и будет распределяться по двум страницам.

Свойства Can Grow, Can Shrink

Элементы управления также имеют возможность установки этих свойств. Если было установлено значение Yes для свойства элемента управления Can Grow (Позволить расширяться), раздел, которому она принадлежит, может увеличиться, чтобы подогнаться к большому полю, такому, например, как поле Memo, имеющее значительное количество данных. Подчиненный отчет дает другой хороший пример функциональных возможностей свойства Can Grow. Сложно определить, сколько вертикального пространства вам нужно в подотчете, поскольку количество записей может меняться. Однако включение свойства Can Grow приводит к тому, что элемент управления и раздел будут соответствовать требуемому пространству.

Вы можете включить свойство Can Shrink (Позволить сужаться), если хотите уменьшить размер раздела так, чтобы он соответствовал меньшему количеству текста. Вы также можете использовать это свойство, чтобы избавиться от этих раздражающих пустых адресных строк в элементах управления.

Понимание области данных (Details Section)

Представьте, что бы случилось, если бы вы убрали элементы управления в этом разделе. Вы бы остались лишь с промежуточными и общими суммами для подведения итогов. Иногда это может быть предпочтительнее. В этом случае вам, вероятно, лучше подвести итоги в запросе, для того чтобы послать результаты в отчет, который бы мог использовать область данных (Details Section). Единственная разница между этими двумя схемами заключается в принадлежности элементов управления. В первом случае они принадлежат к примечаниям группы (Group Section footer). Во втором случае они принадлежат к области данных (Details section), поскольку итоги уже подведены.

Область данных (Details Section) в первую очередь с записями. В столбчатом виде (rolodex-style, в виде вращающейся картотеки) вы с большой вероятностью столкнетесь с элементами управления полей для записей с соответствующими подписями. Но в табличном виде вы бы, вероятно, поместили подписи заголовков в раздел страницы, а элементы управления полей в область данных.

Для информации о том, как программировать область данных см. раздел «Добавление чередующихся серых и белых полосок в отчет» в гл. 7.

Событие On Open

В отчетах это событие происходит до того, как отчет предварительно просмотрен или напечатан. Это первое событие, которое активируется, что дает вам возможность производить некоторые служебные действия. Например, вы можете использовать это событие для изменения источника записей, в зависимости от того, какой источник записей был выбран в форме. Скажем, вы изменили источник записей в форме с записей за 2001 год на записи 2000 года. Очевидно, вы бы хотели, чтобы источник записей в отчете соответствовал вашей форме. Вы бы

могли создать другой отчет, но, поскольку структуры таблицы или запроса являются схожими, в этом нет необходимости.

Вы можете нажать кнопку **Свойства** (Properties), находясь в **Конструкторе отчета** (Design mode). После того как свойства отчета открыты, вы получаете доступ к этому событию. Вы также можете получить доступ к свойствам отчета просто щелчком правой клавиши мыши на серую область справа от отчета.

Работа со свойством источника записей (Record Source)

Вы также можете настроить источник записи на SQL-запрос посредством данного свойства. Это дает вам возможность использовать переменные для изменения табличных имен. Это делает запрос динамичным источником записей для этого отчета.

Есть еще выгода от использования динамического источника записей, и она применима как для формы, так и для отчета. Если вы работаете с огромными базами данных, имеющими сотни тысяч записей, вы можете избежать замедления системы, связанного с открытием формы или отчета, связанных с одной огромной таблицей. Разбейте таблицы мелкие таблички и получайте доступ к ним динамически. Таким образом, пользователь получает доступ к одной группе записей сразу.

Разработка реляционного отчета с помощью мастера

Поскольку вы научились разрабатывать реляционную (relational) форму с помощью мастера в прошлой главе, вы можете почти также использовать его для разработки отчета. Вы можете разработать отчет с нуля или построить отчет на реляционном запросе, но для большинства целей вы можете также воспользоваться преимуществом мастера отчетов (Report Wizard). Следующие шаги покажут вам, как это сделать.

1. Убедитесь, что база данных Music Store открыта.
2. В разделе **Объекты** (Objects) выберите **Отчеты** (Reports) для просмотра списка отчетов и опций отчета.
3. Выберите **Создание отчета с использованием мастера** (Create Report by Using Wizard).
4. В строке **Таблицы и запросы** (Tables/Queries) откройте ниспадающий список и выберите **Таблица:Customers**.
5. В списке **Доступные поля** (Available Fields) нажмите >> для того, чтобы переместить все поля в окно **Выбранные поля** (Selected Fields).
6. Выберите поле Sort Name в окне **Выбранные поля** и нажмите <, чтобы вернуть его обратно в поле **Доступные поля.**
7. Оставайтесь в этом диалоговом окне. В строке **Таблицы и запросы** нажмите стрелку вниз и выберите **Таблица: Invoice.**

8. В списке **Доступные поля** (Available Fields) нажмите >> для того, чтобы переместить все поля в окно **Выбранные поля** (Selected Fields).
9. Нажмите Invoice.CustomerID в окне **Выбранные поля** и нажмите <, чтобы вернуть его обратно в поле **Доступные поля.**
10. Повторите шаги 7 и 8 для **Таблица:InvoiceDetail**.
11. Выберите поле InvoiceDetail.InvoiceID и нажмите < для возвращения его в окно **Доступные поля.** Нажмите **Далее** (Next).
12. На стр. 2 мастера в ответ на запрос **Выберите вид представления данных** (How do you want to view your data?) нажмите **Далее,** подтвердив выбор Customers (by Customers).
13. На стр. 3, которая является страницей группировки уровней, нажмите Далее для принятия группировки уровней по умолчанию.
14. На стр. 4, которая является страницей порядка сортировки, нажмите **Далее** для отмены сортировки в данном отчете.
15. На стр. 5, в которой выбирается тип размещения данных, выберите **По левому краю 1** (Align Left 1) (оставить значение по умолчанию **Книжная** (Portrait) в блоке **Ориентация** (Orientation), как показано на рис. 6.23) и нажмите **Далее.**
16. В списке стилей отчета, после просмотра различных стилей, оставьте стиль **Деловой** (Corporate) и нажмите **Далее.**
17. На следующей странице назовите отчет Customer Invoices и нажмите Готово.
18. Нажмите **Закрыть** (Close) для того, чтобы просмотреть отчет в экране конструктора. Закройте отчет.

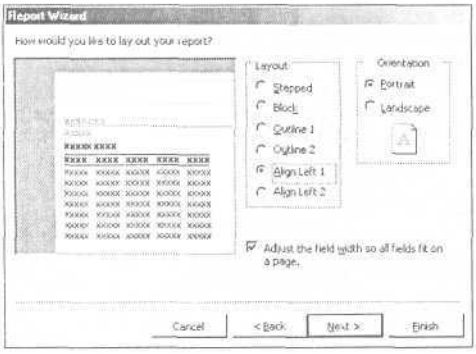


Рис. 6.23. Выбор макета для вашего отчета при использовании мастера отчетов

Условное форматирование

Условное форматирование дает вам возможность изменить формат элемента управления в форме или в отчете, основанном на значении элемента управления. Если вы хотите изменить цвет элемента управления на красный, когда ее значение превысит 2,000, вы можете это сделать используя условное форматирование. Раньше, когда функция условного форматирования еще не была добавлена

в Access, это можно было сделать использую программный код. Теперь вам не надо быть программистом, чтобы воспользоваться этим преимуществом. Следующие шаги покажут вам, как добавлять условное форматирование в отчет.

1. Находясь в базе данных Music Store откройте отчет Inventory Practice в окне конструктора.
2. Выберите текстовое окно UnitPrice.
3. Нажмите **Формат** (Format), а затем **Условное форматирование** (Conditional Formatting), расположенные в меню, или щелкните правой клавишей мышки и выберите **Условное форматирование** (Conditional Formatting) из появившегося меню быстрого вызова.
4. Щелкните на поле справа от строки **между** (between) в разделе **Условие 1** (Condition 1).
5. Наберите 1000 и нажмите Tab. Наберите 1999 в окне после «и».
6. Нажмите кнопку В для выбора полужирного шрифта и посмотрите, как изменился текст в большом белом окне под разделом **Условие 1.**
7. Снова нажмите кнопку **Добавить** (Add) для того, чтобы добавить следующее условие. В разделе **Условие** раскройте ниспадающий список **между.**
8. Выберите **больше** и нажмите Tab. Наберите 2000.
9. Щелкните по ниспадающему окну, за кнопкой А, как показано на рис. 6.24.
10. Выберите красный (третий квадратик вниз от черного). Нажмите OK для подтверждения форматирования.

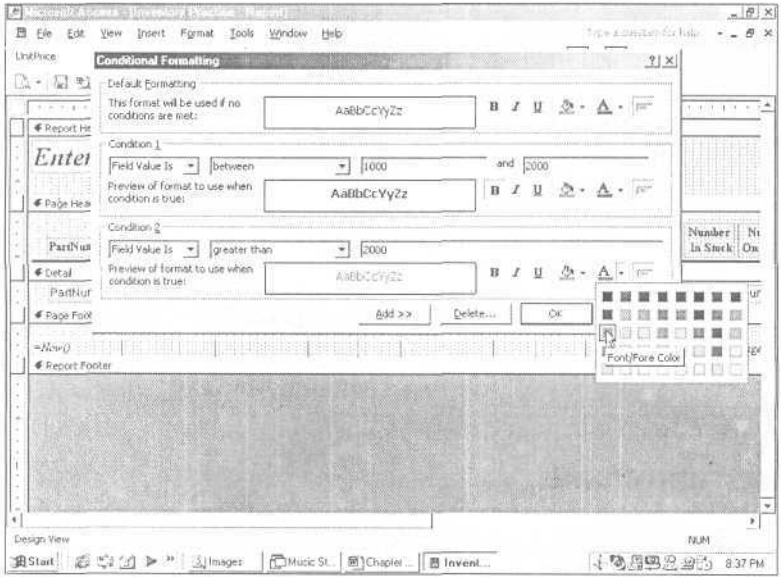


Рис. 6.24. Вы можете использовать опции условного форматирования в отчете не пользуясь программированием

11. Нажмите **Предварительный просмотр** для того, чтобы увидеть отчет, как показано на рис. 6.25.
12. Нажмите указатель следующей записи для просмотра последующих страниц, чтобы удостовериться, что условное форматирование Unit Price прошло удачно. Закройте окно предварительного просмотра, а потом и отчет.

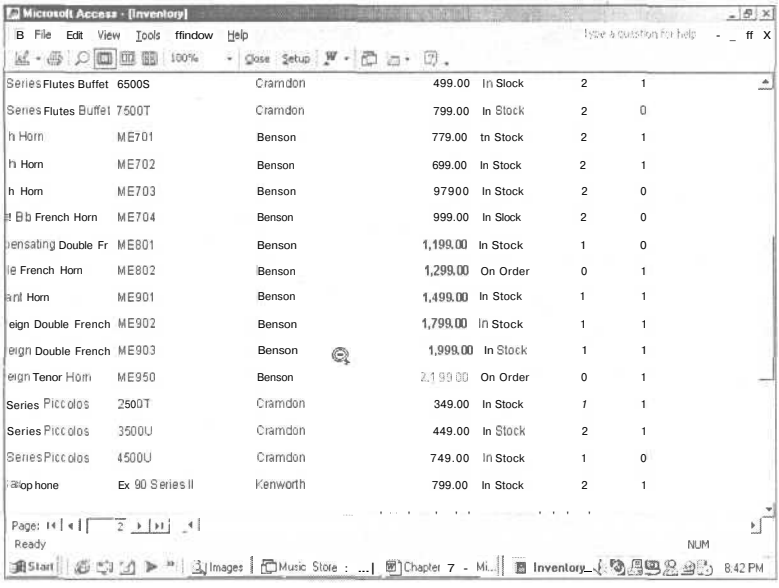


Рис. 6.25. Используйте опцию предварительного просмотра печати (Print Preview) для проверки результатов условного форматирования

Создание групп элементов управления (Group of Controls)

Access 2002 имеет возможность группировать элементы управления. Это удобно по нескольким причинам. Вам не нужно больше выбирать и перевыбирать группы элементов управления. Теперь вы можете сгруппировать их так, чтобы их можно было передвигать, изменять размер и форматировать как единое целое. Они остаются сгруппированными даже после того, как вы выйдете из отчета (при условии, что вы сохранили отчет) и снова откроете его.

1. Находясь в базе данных Music Store, откройте отчет Invoice Practice в окне конструктора (Design view). Покрутите вниз мышью или курсорными клавишами, пока не увидите подписи Sub Total, Tax и Total.
2. Выберите текстовое поле Sub Total справа от подписи Sub Total. Нажмите клавишу Shift и, удерживая ее, выберите текстовые поля Tax и Total.
3. Выберите **Формат** (Format), **Группировать** (Group).
4. Обратите внимание, что ярлыки окружены рамкой. Эту рамку можно перемещать, изменять ее размер и форматировать. Групповое окно не будет печа-

таться на вашем принтере или во время предварительного просмотра печати (Print Preview).

5. Раскройте ниспадающее окно Font/Fore Color (Цвет текста) на панели инструментов. Выберите красный цвет и посмотрите, как изменился шрифт.
6. Щелкните мышью мимо окна группы, а затем снова на нем. Окно, как вам покажется, исчезнет, но оно снова появится, когда вы снова на нем щелкнете. Однако вы по-прежнему можете выбирать отдельные текстовые поля внутри группы и форматировать их индивидуально.
7. Поместите мышью курсор в нижнюю часть окна, пока не появится вместо курсора символ открытой ладони.
8. Нажмите клавишу и, удерживая ее, переместите окно группы слева от подписей Sub Total, Tax и Total, потом отпустите. Обратите внимание, что три текстовых поля перемещаются как группа.
9. Нажмите кнопку **Отменить** (Undo) (или сочетание клавиш Ctrl+Z. - *Пер.)* один раз, чтобы отменить ваше последнее изменение.
10. Закройте отчет, а затем и базу данных.

Хотя в этой главе есть масса полезной информации, нет необходимости в углубленном рассмотрении основ отчета. Для дополнительной информации по рассмотренным в этой главе вопросам см. книгу издательства Que: R. Jennings. *Special Edition Using Access 2002.* ISBN 0-7897-2510-X.

Что дальше?

В этой главе вы узнали о преимуществах отчета над электронными таблицами. Вы узнали, как группировать, сортировать разделы и работать с ними. Вы прочувствовали, что такое условное форматирование и группирование элементов управления. Обе эти функции могут быть использованы также и в формах. Следующая глава поможет вам разобраться в объектах. В ней рассмотрены также вопросы использования событий и языка Visual Basic для автоматизации вашей базы данных.

Часть III
Автоматизация базы данных с помощью программ

7
Изучение объектов

В последних двух главах вы узнали о формах и отчетах, оба из которых являются объектами, которые сильно зависят от элементов управления, также являющихся объектами. В этой главе обсуждается, что такое объекты и что они могут для вас сделать. В частности, вы узнаете:

- что такое объекты;
- что такое объекты в Access;
- какое отношение к объектам имеют методы, свойства и события;
- как назвать объекты;
- как обращаться к объектам;
- как изменять свойства объекта вручную и при помощи VBA;
- как управлять событиями объекта;
- как работать с событиями объекта на примере;
- как программировать с использованием моделей объекта;
- как запрограммировать элемент управления.

Определение объектов

Поиск определения понятия *объект* может дать любой результат, начиная с «взаимосвязанных данных» и заканчивая «нечто, придуманное кем-то из Microsoft, чтобы разработать и продать побольше языков программирования». Объекты сравнивались с чем угодно - от «черных ящиков» до зверей. Нельзя говорить, что объекты - это все, на чем можно щелкнуть мышью, потому что объекты не всегда можно увидеть. Но в целом можно сказать, что объект - это автономный (самодостаточный) программный компонент, который может быть выбран и изменен либо непосредственно пользователем, либо косвенно, программным путем. В Access объекты представляют собой элементы приложения, такие, например, как таблица, запрос, форма, отчет, макрос, модуль или элементы управления.

В Access используются сотни объектов различных типов. Некоторые (такие, как формы, отчеты, элементы управления и модули) можно изменить, щелкнув на них мышью. Эти объекты отображают данные, хранимые в базе данных. Иные объекты используются для получения, хранения и обработки самих дан-

ных. Третьи же облегчают написание программы на VBA, который является частью Visual Basic. Вы даже можете создавать свои собственные объекты, обозначающие реальных людей, реальные предметы и географические места.

Объекты включают в себя их собственные данные, методы (процедуры) либо и то и другое одновременно. Как и реальные объекты, они обладают формой и содержанием. *Форма* - это то, как представлено содержание. Часть содержания описывает форму. Например, свойства объекта имеют отношение к тому, как объект представляется или как себя ведет. Какой размер и цвет шрифта будет использовать объект? Будет ли он видимым или невидимым? Будет ли он выводить все поля? Будет ли его параметрам присвоено значение по умолчанию? Другая часть содержания объектов - это непосредственно пользовательские данные и процедуры.

Объект характеризуется состоянием, поведением и идентификатором.

- *Состояние* относится к данным, которые содержит объект, или к самому объекту. Например, длина и ширина поля определяют состояние.
- *Поведение* относится к методам объектов, что в Access означает подпроцедуры и функции-процедуры. Поведение позволяет объектам взаимодействовать между собой.
- Под *идентификатором* понимается уникальный способ обращаться к объектам, в основном по имени. Предположим, у вас есть много полей на форме. Как отличить их одно от другого?

Объекты взаимодействуют с событиями. Однако объекты могут быть активными или пассивными; они могут инициировать событие или ответить на него. Например, открытие объекта «форма» инициирует событие объекта «форма» On Open. Когда по кнопке на поле щелкают мышью, запуская событие On Click, объект «отчет» может ответить распечаткой текущего содержания формы. События - это тот механизм, который позволяет объектам влиять друг на друга. Объекты могут изменять другие объекты и могут сами быть изменены. Они даже позволяют пользователю уничтожать объекты.

Способность хранить в себе и скрывать информацию об объекте (такую, например, как внутренние данные или программный код) называется *инкапсуляцией*. В Access объекты «форма» и «отчет» обычно инкапсулируют свойства, методы и события. Последние могут быть определены следующим образом:

- Свойства - обозначают определенные атрибуты (характеристики) объекта. Это может быть как графический атрибут, такой, как размер и цвет, так и характеристика поведения, например «скрытый».
- Методы - обозначают службы (процедуры), обеспечиваемые объектом. Например, объект DoCmd обладает методами, отвечающими за большинство макрокоманд, такими как DoCmd.OpenForm.
- События - обозначают явления, на которые объект реагирует, или явления, которые объект порождает. Например, событие After Update происходит после обновления данных элемента управления.

Определение свойств объектов

Свойства объектов - это индивидуальные характеристики, или атрибуты, объектов. Эти свойства могут быть изменены вручную или программным путем. И сейчас пришло время более внимательно рассмотреть следующие вопросы: как называть объекты и обращаться к ним, а также как узнавать и изменять свойства объектов?

Для получения дополнительной информации об изменении свойств объектов см. раздел «Изменение свойств объектов» ниже в этой главе.

Названия объектов

Если бы вы встретились с классом, состоящим из 101 объекта, вы не продвинулись бы далеко в его изучении и использовании, не зная, как называть эти объекты (т. е. как к ним обращаться «по имени-отчеству». – *Пер.*).Следовательно, перед тем как продолжить рассмотрение свойств объектов, мы должны сделать небольшое отступление (которое вообще-то и не является отступлением в полном смысле этого слова), чтобы убедиться, что вы понимаете, как обращаться к объектам.

Вы уже видели, что в запросах скобки автоматически вставляются для имен объектов, не имеющих пробелов. Вы должны самостоятельно расставлять скобки, если в вашем объекте содержатся пробелы. Но что если ваша форма или запрос имеют больше чем одну таблицу с одинаковыми именами полей? Как обращаться к ним? Если у вас есть таблицы Customer и Order, обе имеющие поле CustID, вы можете обращаться к нему следующим образом:

```
[Customer]![CustID]
```

Но если у вас есть форма, называющаяся CustForm и другая форма, называющаяся CustOrder, имеющие одинаковые имена полей, вы должны обращаться к ним посредством зарезервированного слова forms вот так:

```
Forms![CustForm]![CustID]
```

Сравнение оператора «восклицательный знак» и оператора «точка»

Если вы хотите правильно идентифицировать объекты, вы должны использовать правильный синтаксис. Когда следует использовать оператор «восклицательный знак» (!), а когда - оператор «точка» (.)? Непосвященного пользователя наверняка введет в заблуждение эта разница. Оба они операторы-идентификаторы. Оба служат для описания взаимосвязи между коллекциями и объектами и вызова последних. Так в чем же разница?

Вообще говоря, вы должны использовать оператор «восклицательный знак» для обозначения взаимосвязи между созданными объектами, такими, как таблицы, формы, отчеты или элементы управления, которые становятся частью коллекций. Оператор «точка», наоборот, обозначает взаимосвязь между объектом

и его свойством, методом или коллекцией. Следующий кусок кода иллюстрирует вышесказанное:

```
With rs 'Добавляет их в таблицу названий таблиц
   .AddNew
   ![TableName] = td.NAME
   .Update
   .Bookmark =.LastModified
End With
```

Конструкция With

Конструкция With предоставляет вам эффективный способ обращаться сразу ко множеству свойств объекта. Заметьте, что вместо того, чтобы каждый раз повторять переменную rs для каждого обращения к объекту, вы сразу используете операторы «точка» или «восклицательный знак» в начале каждой внутренней строки. Следовательно, конструкция With ускоряет написание кода программы. Кроме того, когда код уже написан, его проще читать и исправлять.

Переменная rs обозначает множество записей из таблицы названий таблиц. Оператор «восклицательный знак» в ![TableName] обращается к полю в таблице, *но* оператор «точка» в td.Name обращается к названиям всех таблиц в коллекции TableDef. Хотя в обоих случаях они обращаются к созданным объектам, в первом случае оператор используется для того, чтобы обратиться к полю и добавить в него данные, а во втором случае оператор используется, чтобы обратиться к свойству коллекции и скопировать данные из него. Оператор «точка» в.Update используется для обращения к методу.

Изменение свойств объектов

Вы можете узнавать и изменять свойства объектов двумя способами. Вы можете изменить свойства объекта вручную, щелкнув на нем правой кнопкой мыши и выбрав «свойства», либо можно изменить свойства при помощи программного кода. Например, вы можете щелкнуть правой кнопкой мыши на элементе управления и ввести текст в поле «текст подсказки» данного элемента управления. То же свойство можно изменить при помощи программного кода. Этот метод дает вам возможность изменять текст сообщения-подсказки в зависимости от условий убрать его вообще, если это будет нужно. Другими словами, используя программный код, вы получаете большую гибкость. Процесс происходит «на лету», во время выполнения процедуры; напротив, если вы не используете первый метод, вам придется все бросать, открывать форму или отчет в окне конструктора (Design View) и модифицировать элемент управления или другой объект, который вы собирались изменить.

Изменение свойств при помощи VBA

Хотя вы уже представляете себе, как использовать программный код для изменения свойств объектов, кое-что еще нужно прояснить. Существует несколько спо-

собов обращения к свойствам объекта при помощи программного кода. Вы можете делать это используя ключевое слово Me. Также вы можете обращаться к свойствам объекта используя иерархию объектов. Предположим, что форма открыта. Следующий пример показывает два способа, которыми вы можете воспользоваться для обращения к элементу управления ctlPhone на форме frmCustomer:

```
'Неявное обращение
Forms!frmCustomer!ctlPhone.name
'Явное обращение
Forms!frmCustomer.Controls!ctlPhone.name
```

Заметьте, что в неявной форме обращение сначала идет к коллекции форм, затем идет имя формы, затем имя элемента управления, затем свойство. Другой способ обращаться к объектам и их свойствам заключается в использовании объектных переменных. Следующий пример показывает, как объявлять и присваивать значение объектным переменным:

```
Dim frm as Form
Dim ctl as Control

Set frm = Forms!frmCustomers
Set ctl = frm.Controls!strCompanyName
```

Объектные переменные не ссылались ни на какой объект, пока им не присвоили значение. Ключевое слово Set «указывает» на объект, к которому обращаются. После того как это произведено, вы можете обращаться к объекту и его свойствам следующим образом:

```
VarProperty = ctl.name
```

Предположим, на вашей форме есть поле, называемое State. Если вы хотите приписать ему значение, вы можете просто добавить строчку следующего содержания в свою программу:

```
[State] = "CO"
```

Однако вы также можете обращаться к этому полю используя ключевое слово Me, как показано ниже:

```
Me.[State] = "CO"
```

Это ключевое слово особенно полезно, если вы не помните точно название поля или другого объекта, к которым обращаетесь. Как сказано выше, когда вы наберете Me в окне модуля, появится список. Не имеет значения, различаются ли имена поля и элемента управления, - вы увидите оба варианта, набрав ключевое слово Me.

Вы можете обращаться к тому же самому объекту используя объектную переменную. Преимущество этого метода состоит в том, что вы можете обращаться и изменять каждый элемент управления формы одновременно, если это необходимо, используя переменную. Все, что вам нужно сделать, - это объявить переменную как переменную элемента управления. Затем можно использовать выражение With для обращения к объекту, на который ссылается переменная, как показано ниже:

```
Dim MyControl as Control
With MyControl
.ForeColor = 255
End With
```

Выражение With разобрано более подробно в гл. 5, раздел «Изучение форм и элементов управления», но вы должны помнить, что эти различные методы обращения к объектам - в вашем арсенале. Когда вы привыкнете к этим методам, вы наверняка будете использовать их снова и снова.

Подробнее про объектные события

В этой главе вы уже познакомились с событиями как в теории, так и на примерах. На первый взгляд события могут показаться чем-то мистическим. Что инициирует их? Как их можно использовать? Названия некоторых событий сами за себя отвечают. Например, On Click случается, когда вы нажимаете и отпускаете левую кнопку мыши на объекте. Но как насчет On Focus? Чем оно инициируется? Табл. 7.1 и 7.2 должны помочь ответить на эти вопросы[10].

Таблица 7.1. Когда происходят события

Событие	К чему применяется	Когда инициируется
События данных		
AfterDelConfirm	Формы	После подтверждения или отмены удаления записи
AfterInsert	»	После добавления новой записи в базу данных
AfterUpdate	Формы, элементы управления	После обновления данных в записи или в элементе управления
BeforeDelConfirm	Формы	После удаления одной или нескольких записей, но перед появлением диалогового окна и после события Delete
BeforeInsert	Формы	Когда вы написали первый символ в новую запись, но перед добавлением записи в таблицу
BeforeUpdate	Формы, элементы управления	Перед обновлением данных элемента управления или записи
OnChange	Элементы управления	Когда изменяется содержимое блока текст или текстовой части комбинированного блока

[10] В первом столбце табл. 7.1 и 7.2 перечислены функции, вставляемые в программу для создания того или иного события. Автор на протяжении всей книги не делает различий между событием и функцией, создающей событие. - *Науч. ред.*

Событие	К чему применяется	Когда инициируется
OnCurrent	Формы	Когда право ввода перемещается на запись, делая ее активной, или когда вы запрашиваете форму
OnDelete	»	При удалении записи, перед тем как удаление было подтверждено или произведено
OnDirty	»	При изменении содержимого формы или текстовой части комбинированного блока
OnNotInList	Элементы управления	Когда в комбинированный блок водится значение, не присутствующее в его списке
OnUpdated	То же	Когда данные объекта OLE были модифицированы
События-ошибки и временные события		
OnError	Формы, отчеты	При генерации «ошибки Jet во время исполнения», когда вы находитесь на форме или на отчете
OnTimer	Формы	При истечении установленного временного интервала
События фильтров		
OnApplyFilter	Формы	Когда вы нажимаете «Применить фильтр для форм»
OnFilter	»	Когда вы нажимаете «Фильтр форм»
События права ввода (Focus Events)		
OnActivate	Формы, отчеты	Когда форма или отчет становится активным окном
OnDeactivate	» »	Когда иное окно становится активным, но до того, как текущее потеряет право ввода
OnEnter	Элементы управления	Перед тем как элемент управления действительно получит право ввода перед событием GotFocus
OnExit	То же	Как раз перед тем, как элемент управления потеряет право ввода в пользу другого элемента управления, и перед событием LostFocus
OnGotFocus	Формы, элементы управления	Когда элемент управления или форма без активных или доступных элементов получает право ввода
OnLostFocus	То же	Когда форма или элемент управления теряет право ввода

Событие	К чему применяется	Когда инициируется
События клавиатуры		
OnKeyDown	Формы, элементы управления	Когда вы нажимаете любую клавишу на клавиатуре в то время, а право ввода находится на элементе управления или на форме
OnKeyPress	То же	Когда вы нажимаете и отпускаете клавишу или их сочетание, а право ввода находится на элементе управления или на форме
OnKeyUp	»	Когда вы отпускаете клавишу, а право ввода находится на элементе управления или на форме
События мыши		
OnClick	Формы, элементы управления	Когда вы нажимаете и отпускаете (щелкаете) левую кнопку мыши на элементе управления
OnDblClick	То же	Когда вы дважды нажимаете и отпускаете (щелкаете) левую кнопку мыши на элементе управления или на подписи к нему
OnMouseDown	»	Когда вы нажимаете кнопку мыши, а указатель находится на форме или на элементе управления
OnMouseMove	»	Когда вы перемещаете указатель мыши над формой, разделом формы или над элементом управления
OnMouseUp	»	Когда вы отпускаете нажатую кнопку мыши, а указатель находится на форме или на элементе
Оконные события		
OnClose	Формы, отчеты	Когда форма или отчет закрыты и убраны с экрана
OnLoad	Формы	Когда форма открыта и ее записи отображены на экране; перед событием Current, но после события Open
OnOpen	Формы, отчеты	Когда форма открыта, но перед тем, как первая запись отображена; после открытия отчета, но до того, как он напечатан

Событие	К чему применяется	Когда инициируется
OnResize	Формы	Когда размер формы изменяется; также при первом отображении формы
OnUnload	»	Когда форма закрыта и все ее записи выгружены, но перед событием Close
BeforeScreenTip	»	Перед тем, как ScreenTip (всплывающая подсказка) отображается для элемента в окне «Сводная диаграмма» или «Сводная таблица»

Таблица 7.2. Новые события сводной таблицы и сводной диаграммы

События	Доступно в сводной Таблице	Доступно в сводной Диаграмме
OnConnect, OnDisconnect	Да	Нет
BeforeQuery, Query	»	»
AfterLayout, BeforeRender, AfterRender, AfterFinalRender	Нет	Да
DataChange	Да	Нет
DataSetChange	Нет	Да
PivotTableChange	Да	Нет
SelectionChange, ViewChange	»	Да
CommandEnabled, CommandChecked, CommandBeforeExecute, CommandExecute	»	»
KeyDown, KeyPress, KeyUp, MouseDown, MouseMove, MouseUp, MouseWheel, Click, DblClick	»	»

Право ввода может находиться только в одном месте. Если вы вводите данные в поле для ввода (text box), строчка данных появляется, только если поле для ввода имеет право ввода. Для элементов управления события Enter и GotFocus происходят в следующем порядке: Enter, GotFocus.

Событие On Click, вероятнее всего, будет приоритетным для командной кнопки. Некорректно назначать различные процедуры событиям On Click и On Dbl Click одной и той же кнопки: Access в этой ситуации, вероятнее всего, проинтерпретирует двойной щелчок мышью как одинарный, инициируя событие Click.

Следует заметить, что, когда вы ищете какое-либо событие в списке свойств, его название будет содержать пробелы, чтобы облегчить чтение. Например, событие OnOpen в VBA обозначается как On Open в списке свойств.

Добавление чередующихся серых и белых полосок в отчет

События имеют смысл только в связи с поведением того объекта, в который они инкапсулированы. Вы не открываете комбинированный блок (combo-box), поэтому нет нужды в том, чтобы у КОМБИНИРОВАННОГО БЛОКА было событие On Open. Однако вы форматируете область данных в отчете. Следовательно, поскольку область данных является объектом, она обладает событием On Format. Это событие позволяет вам форматировать каждую запись по-разному, если это вам необходимо.

Следующее упражнение, использующее простую процедуру, весьма полезно в трех отношениях. Во-первых, оно дает вам почувствовать силу Visual Basic for Applications (VBA). Во-вторых, оно стимулирует мыслительный процесс, давая вам понять, что такое область данных на самом деле. И в-третьих, оно дает вам представление, как работать с событиями.

1. Откройте отчет Inventory Practice в демонстрационной базе данных Music Store и откройте ее в виде конструктора (Design View).
2. Щелкните правой кнопкой мыши на области данных и выберите Свойства в появившемся меню. Откроется список свойств.
3. Щелкните на закладке События. Выбрав [Процедура обработки события] из ниспадающего окна события On Format, щелкните кнопку Создать для открытия окна модуля, показанного на рис 7.1.

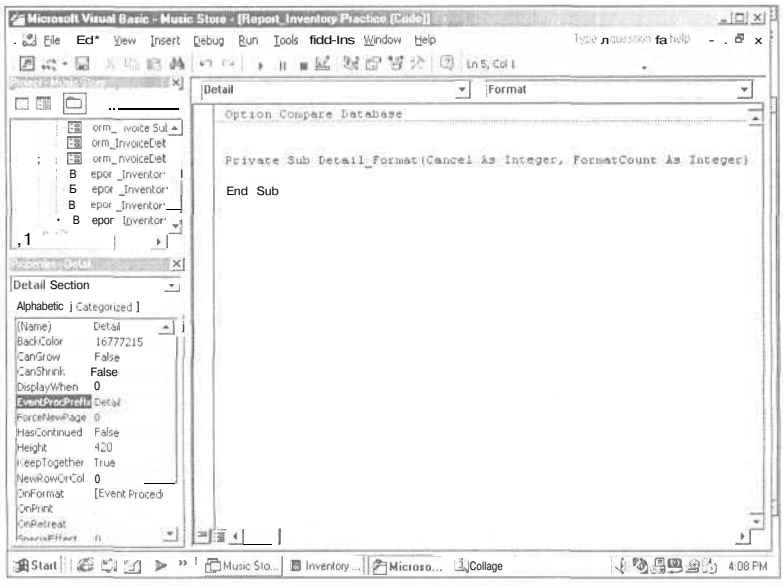


Рис.. 7.1. Щелкните кнопку **Создать** для открытия окна модуля, в котором вы можете набирать код программы на Visual Basic

4. Введите следующий код в окне от входной точки и до строки End Sub. Оставьте пока окно открытым.

```
'Устанавливает константы
Const White = 16777215
Const Gray = 12632256

'Сообщает процедуре, какой цвет нужно установить
'в зависимости от состояния переменной BackForth
Select Case BackForth
 Case True
  Me.Detail.BackColor = White
 Case False
  Me.Detail.BackColor - Gray
End Select

'Изменяет состояние переменной из True (истина) на False (ложь) и обратно
'каждый раз, когда выполняется процедура (для каждой записи)

BackForth = Not BackForth
```

Использование события для приписывания цвета области данных

вы можете использовать функции QBColor или RGB для присвоения цвета константам White и Gray, но диапазон цветов, задаваемых ими, не достаточно велик. Однако если вам просто нужен один из базовых цветов, эти функции подходят. Типичное выражение с использованием QBColor: Blue = QBColor(2). Для RGB: Red = RGB(255, O, O). В табл. 7.3 и 7.4 представлены значения аргументов, необходимых для получения стандартных цветов.

Таблица 7.3. Таблица цветов для функции QBColor

Номер	Цвет
0	Черный
1	Синий
2	Зеленый
3	Голубой
4	Красный
5	Пурпурный
6	Желтый
7	Белый
8	Серый
9	Светлый синий
10	Светлый зеленый
11	Светлый голубой
12	Светлый красный
13	Светлый пурпурный
14	Светлый желтый
15	Светлый белый

Таблица 7.4. Таблица цветов для функции RGB

Цвет	Красный	Зеленый	Синий
Черный	0	0	0
Синий	0	0	255
Зеленый	0	255	0
Голубой	0	255	255
Красный	255	0	0
Пурпурный	255	0	255
Желтый	255	255	0
Белый	255	255	255

Если у вас есть числа, которые не будут изменяться, вы можете приписать их константам (Const) для экономии памяти. Константы представляют собой постоянные, или неменяющиеся, величины, такие, как символьные или численные знаки. Вы можете спросить: «Почему бы просто не записывать значения непосредственно в текст программы?» Если у вас есть длинная строка или большое число, которое вы не хотели бы повторять в программном коде снова и снова, вам нужно использовать константы вместо этих чисел и строк. Другая причина, по которой стоит использовать константы, состоит в том, что они делают программу более легкой для понимания. Гораздо проще понять Gray, чем 12632256.

Существуют предопределенные (встроенные) константы, обозначающие цвета, которые вы можете использовать в своих процедурах. Предположим, вы хотите получить зеленые и белые полоски вместо серых и белых. Для этих цветов вы можете использовать встроенные константы vbWhite и vbGreen. Также доступны vbBlack, vbRed, vbYellow, vbBlue, vbMagenta и vbCyan для соответствующих цветов (черный, красный, желтый, синий, пурпурный и голубой). Причина, по которой они не были использованы в вышеописанной процедуре, состоит в том, что константа vbGreen соответствует зеленому цвету, который слишком темен для данного случая.

Объявление глобальных переменных

На первый взгляд кажется, что в этой программе нет переменных. Однако, ключевым моментом во всей процедуре является использование переменной BackForth. Она чередует свои значения между True и False. Эта переменная объявлена где-то выше как глобальная (общедоступная) переменная. Она в совокупности с выражением select case обеспечивает простой способ чередования цветов.

Поскольку процедуры из списка свойств запускаются снова и снова при обработке данных записи, вы должны обращать особое внимание при их написании. Например, если вы написали процедуру длиной всего в одну строчку, которая должна была бы увеличивать значение переменной x на единицу: x = x + 1. На самом же деле переменная x будет равняться единице для каждой новой за-

писи, вместо того чтобы постепенно увеличиваться (1, 2, 3 и т. д.). Поэтому вам нужно объявить глобальные переменные (иногда называемые *общедоступными*) вне процедуры в стандартном модуле. В Access глобальные переменные могут быть использованы во всех модулях.

Для получения дополнительной информации о глобальных переменных см. раздел «Что такое область действия (scope)?» в гл. 10.

По указанным выше причинам вы должны инициализировать (установить в первый раз) переменную BackForth где-то в другом месте, но не в списке свойств. Потому что, если вы установите ей значение False в списке свойств, она так и будет равняться False для каждой записи, каждый раз, когда запускается процедура. Оператор Not изменяет значение переменной на противоположное. Если она была True, станет False, и наоборот.

Подпроцедура, которую вы начали писать, еще не завершена. Нужно позаботиться еще о двух вещах. Вам нужно объявить глобальную переменную и в разделе **Верхний колонтитул** (Page Header) присвоить ей значение False.

Выполните следующие пункты, чтобы сделать завершающие штрихи в разработке процедуры.

1. В меню модуля выберите **Отладка** (Debug), **Компилировать Music Store** (Compile Music Store), как показано на рис. 7.2, и нажмите **Сохранить** (Save).

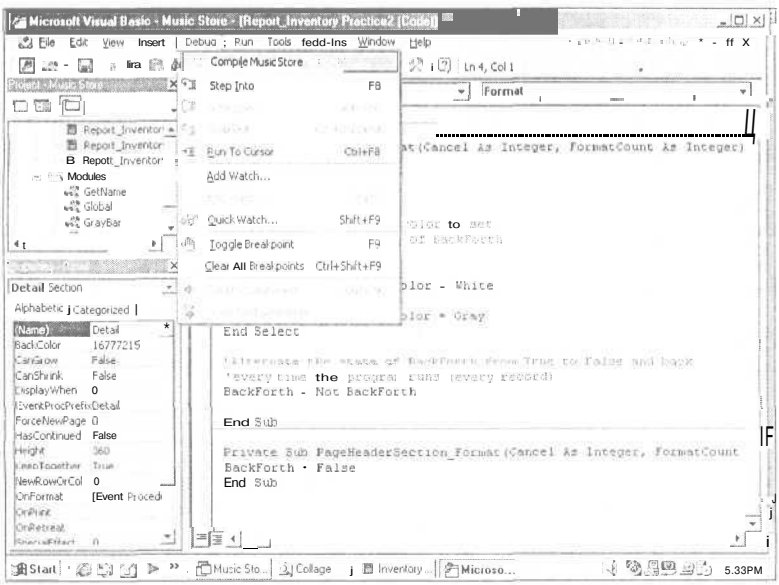


Рис. 7.2. Выберите Отладка (Debug), Компилировать Music Store (Compile Music Store) для проверки вашего программного кода

2. Нажмите **Вставка, Модуль** (Insert, Module), чтобы открыть новый стандартный модуль, который не показан в списке модулей отчета. Поскольку вы не

выбрали **Модуль класса** (Class Module) из меню **Вставка** (Insert), вы вставили стандартный модуль.

3. Введите следующее выражение после строки Option Compare Database сверху в модуле:

   ```
   Public BackForth As Boolean
   ```

4. Нажмите **Сохранить** (Save). Когда появится сообщение **Сохранить изменения в следующих объектах?**, нажмите Да. Сохраните модуль как basGlobal. Объявление переменной как глобальной (Public) делает ее открытой для использования другими модулями. Булевы (Boolean) переменные могут иметь только два значения - True (истина) и False (ложь).
5. Сверните окно **Модуль** и щелкните мышью на разделе «Верхний Колонтитул» (Page Header) отчета.
6. Вы должны увидеть **Список свойств,** как показано на рис. 7.3. Если это не так, нажмите F4.

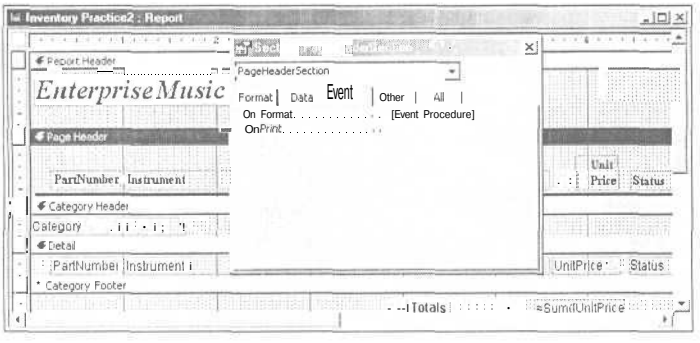


Рис. 7.3. Вы можете свернуть окно Модуль (Module) и создать другую процедуру в разделе Верхний колонтитул (Page Header) отчета

7. Выберите закладку **Событие** (Event), если нужно.
8. Выбрав [Процедура обработки события] из ниспадающего окна события On Format, щелкните кнопку **Создать** и введите следующее выражение:

   ```
   BackForth = False
   ```

9. Закройте окно VBA, закройте **Список свойств** и вернитесь обратно к отчету.
10. Нажмите **Предварительный просмотр** (Print Preview), чтобы насладиться результатом вашей работы. Вы должны увидеть чередующиеся белые и серые полоски, как показано на рис. 7.4.
11. Закройте предварительный просмотр, сохраните и закройте отчет.

Рис. 7.4. Нажмите Предварительный просмотр (Print preview), чтобы увидеть отчет с чередующимися полосками и проверить работоспособность процедуры

Вы, наверное, удивляетесь: «Откуда вам известны эти номера для цветов?» Хороший вопрос. Вам не нужно лазить по справочной системе, руководствам и Интернету, чтобы получить их. Если вы хотите, чтобы чередовался, например, светло-голубой вместо серого, есть простой способ найти номер для него.

Все, что вам нужно сделать, - это щелкнуть правой кнопкой мыши на любом элементе управления в «Конструкторе» (Design View), выбрать **Свойства,** и нажать кнопку **Создать** напротив свойства «Цвет фона» (Back Color), чтобы выбрать цвет. Нажав на кнопку «Определить цвет» (Define Custom Colors), вы получите возможность сделать цвет ярче или темнее. После нажатия кнопки OK выделите мышью получившийся номер и нажмите Ctrl+C, чтобы скопировать его, как показано на рис. 7.5. Потом просто вставьте номер в программный код при помощи иконки **Вставить** или сочетанием клавиш Ctrl+V. Вы также можете просто щелкнуть правой кнопкой мыши на элементе управления и выбрать «Цвет заливки/фона» (Fill/Back Color), но палитра там несколько иная.

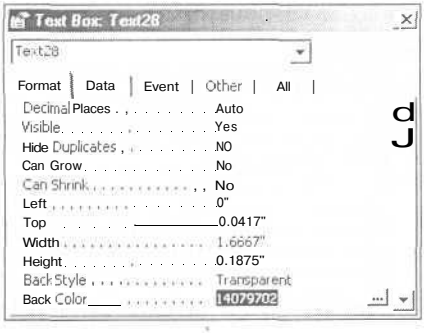


Рис. 7.5. Вы можете скопировать номер цвета из «Цвета фона» элемента управления и использовать его в своей процедуре

Методы объектов

Метод - это процедура, которая производится над объектом. В Access есть свои встроенные методы, либо вы можете создавать ваши процедуры, используя VBA. Вы можете посмотреть методы объекта при помощи обозревателя объектов (object browser) либо любым вышерассмотренным способом.

Модели объектов

Модель объекта обеспечивает структуру доступа к объекту и его модификации. В Microsoft Access 97 использовалась модель, называемая DAO (Data Access Objects, объекты доступа к данным). Начиная с Access 2000, ADO (ActiveX Data Objects, объекты данных ActiveX) стала новым стандартом для Access, причем библиотеки DAO все еще доступны. Хотя ADO более мощна и гибка, чем DAO, DAO все-таки выигрывает в производительности, особенно при работе с Microsoft Jet. Однако если вы планируете иметь доступ к SQL-серверу или базам данных Oracle, ADO более предпочтительна. Кроме того, поскольку Microsoft рассматривает ADO в качестве будущего в технологии доступа к данным, было бы мудрым решением для тех, кто желает оставаться на острой грани, разделяющей эти две модели, воспользоваться преимуществами последней.

Поскольку модель объектов ADO основана на DAO, у них есть много общего. Те, кто уже знаком с DAO, не испытают больших трудностей изучая ADO. В этой книге представлены примеры программ, основанных на обеих моделях. Разумно будет акцентировать внимание только на тех из них, которые используют подходящую вам модель.

Для получения дополнительной информации о моделях объектов см. гл. 8.

Установка ссылок в библиотеке ссылок

Если ваша программа на Visual Basic ссылается на объекты в других приложениях, вам необходимо создать или активизировать ссылку на библиотеки объектов тех приложений. В списке ссылок показаны все библиотеки, зарегистрированные операционной системой. Он также позволяет вам выбирать библиотеки DAO и ADO. Если вы планируете использовать DAO, вы должны установить на нее ссылку, так как она не устанавливается по умолчанию в Access 2002.

Следующее упражнение поможет вам установить ссылку на библиотеку:

1. Находясь в окне «База данных» в Access, нажмите **Вставка, Модуль** для открытия окна **Модуль.**
2. В окне **Модуль** выберите **Инструменты** (Tools), **Ссылки** (References) для открытия диалогового окна **Ссылки,** как показано на рис. 7.6.
3. Четвертая сверху строчка должна быть «Библиотека ADO».
4. Прокрутите список вниз до «Microsoft DAO 3.6 Object Library».
5. Чтобы установить ссылку, просто щелкните по ней мышью.

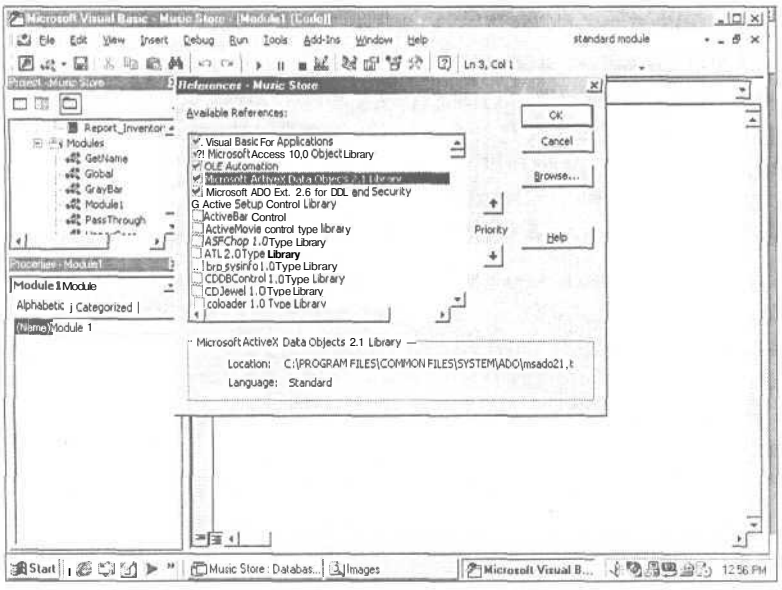


Рис. 7.6. Вы можете использовать библиотеку ссылок, чтобы установить ссылку на нужную вам модель объектов

6. Нажмите **Отмена,** чтобы закрыть диалоговое окно **Ссылки,** и нажмите F2, чтобы просмотреть объекты выбранных вами библиотек (рис. 7.7.)

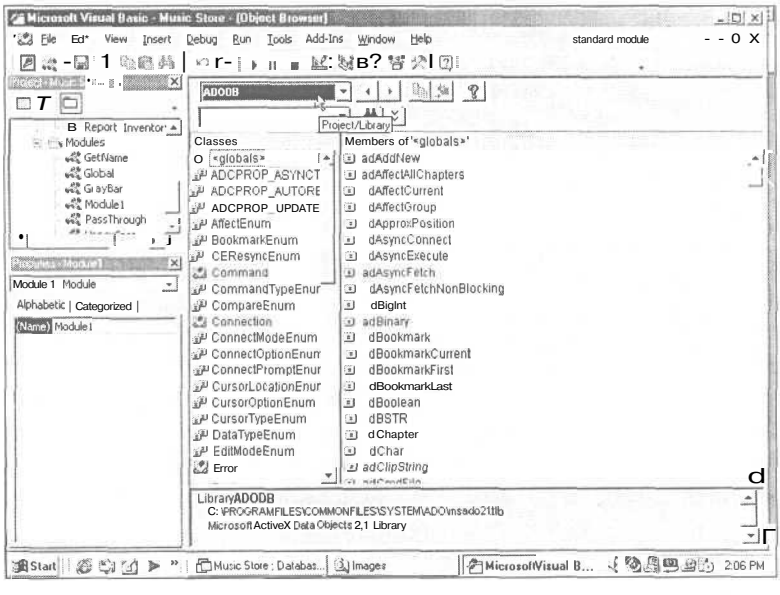


Рис. 7.7. Нажав F2 в окне Модуль, вы можете просмотреть объекты из библиотек, которые вы выбрали

Установка ссылок для других приложений

Если приложение, на которое вы хотите ссылаться, не показано в списке, вы можете нажать на кнопку **Обзор** (Browse) и найти библиотеки объектов вручную (файлы *.olb и *.tlb). Помните, что ссылки, не отмеченные галочкой, не используются в вашем проекте, но могут быть добавлены в любой момент.

Возвращаясь к возможностям элементов управления

В двух предыдущих главах были кратко представлены те возможности и та мощь, которые станут доступными любому, кто уделит время для изучения внутреннего функционирования элементов управления. Пришло время продолжить начатое и копнуть чуть глубже, чтобы разобраться в этой будоражащей ум теме. Мы начнем с обсуждения различных способов, которыми элемент управления может отображать данные.

Неприсоединенный элемент управления все же может отображать данные посредством своего свойства «Источник строк» (Row Source). Вы можете привязать источник строк к таблице и производить доступ к ее полям посредством свойства Column (дословно - «столбец», используется только в VBA и макросах, поэтому не переводится. - *Пер.)* элемента управления. Вы даже можете указать, к какой строке вы хотите обращаться, так как к свойству Column можно обращаться как к двумерной таблице. Если вы введете всего один аргумент, процедура будет интерпретировать это так, что вы хотите получить доступ к определенному столбцу. Например, если вы введете `MyControl.column(1)` как источник данных в другом элементе управления, вы будете запрашивать второй столбец (помните, что первый - нуль) элемента управления MyControl, у которого источником строк должна быть таблица по крайней мере с двумя полями. Однако если вы введете `MyControl.column(0, 1)`, вы будете запрашивать вторую строчку (первая, опять-таки нуль) в первом столбце этого же элемента управления. Помните всегда, что в этих свойствах нумерация начинается с нуля.

Мог возникнуть вопрос: «Почему бы просто не использовать свойство «Источник строк» (Row Source) вместо свойства «Данные» элемента управления (Control Source) для таблиц и запросов?» Во-первых, обычные элементы управления «Поле для ввода» (Text Box) вообще не имеют свойства «Источник строк». Но даже для списка (List Box) и поля со списком (Combo Box), если не использовать свойства «Данные» элемента управления, пользователь не сможет ввести данные в таблицу или запрос. Если вы создадите пустую форму, поместите неприсоединенный элемент управления (не присоединенный к таблице или запросу посредством своего свойства «Данные») на нее и свяжете его с таблицей посредством свойства «Источник строк», вы сможете видеть данные, но не сможете вводить новые без применения программирования.

«Источник строк» - замечательный способ получения данных из таблиц или запросов, которые не связаны с формой посредством свойства «Источник записей». Однако полученные данные могут быть использованы в большинстве случаев только для поиска. Элемент управления может искать данные для своего

источника данных (Control Source) либо для других элементов управления, использующих свойство Column. Список или поле со списком (List Box и Combo Box) не обязаны быть присоединенными для того, чтобы работал метод, использующий свойство Column. Но помните, что ни список, ни поле со списком, не являющиеся присоединенными, не будут отображать сделанный вами выбор после того, как форма будет закрыта и открыта заново.

Три источника

Три свойства в форме или отчете могут стать источниками данных таблиц или запросов. Все эти три источника могут работать совместно. Например, свойство формы «Источник записей» (Record Source) может быть привязано к таблице или запросу. Как только вы установили источник записей формы или отчета на таблицу или запрос, их поля становятся доступны форме или отчету посредством источника данных (Control Source) любого поля для ввода (Text Box), списка (List Box) или поля со списком (Combo Box), которые вы разместили, например, на форме. Источник строк (Row Source) может стать источником данных для источника «Control Source» того же самого элемента управления.

Можно посмотреть на это с другой точки зрения: вы можете использовать свойство «Данные» (Control Source) как для отображения данных из таблицы исходных данных, так и для вставки данных в нее. «Источник строк» (Row Source), напротив, только извлекает и отображает данные, если только вы не используете программирование. «Источник записей» (Record Source) доступен для всех форм и отчетов, а свойства «Данные» (Control Source) и «Источник строк» (Row Source) доступны только для определенных типов элементов управления. Табл. 7.5 поясняет разницу между этими тремя свойствами.

Таблица 7.5. Свойства, используемые для источников данных

Свойство	Источник данных	Является свойством объекта	Доступно для следующих объектов
Источник записей (Record Source)	Таблица или запрос	Форма или отчёт	Форма или отчёт
Данные (Control Source)	Поле или выражение	Элемент управления	Поле для ввода, список, поле со списком (Text Box, List Box, Combo Box)*
Источник строк (Row Source)	Таблица или запрос	Элемент управления	Список и поле со списком, но не поле для ввода*

* И другие элементы управления

На рис. 7.8 изображен пример использования свойства «Источник строк» (Row Source), в котором оно заполнено перечнем значений, разделенных точками с запятой. Обратите внимание, что в свойстве «Тип источника строк» (Row Source Type) выставлено «Список значений» (Value List).

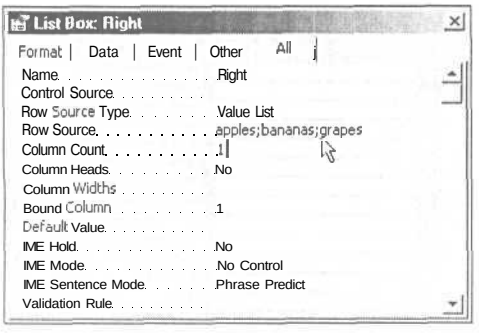


Рис. 7.8. Используйте свойство «Источник строк» (Row Source) в качестве списка значений либо набирая перечень, разделенный точками с запятой, либо создавая их программно

Выбираемый список

выбираемый список, показанный на рис. 7.9, является прекрасным примером, демонстрирующим возможности элементов управления. Элемент управления «Список» предоставляет для форм превосходный способ выбора компонентов из перечня. Для каких целей это вам может понадобиться? Воспользуйтесь воображением. Вы можете выбирать отчеты для распечатки. Вы можете выбрать регионы, являющиеся ключевыми по продаже вашей продукции, на которых нужно сфокусироваться. Вы можете выделить лучших работников для поощрения.

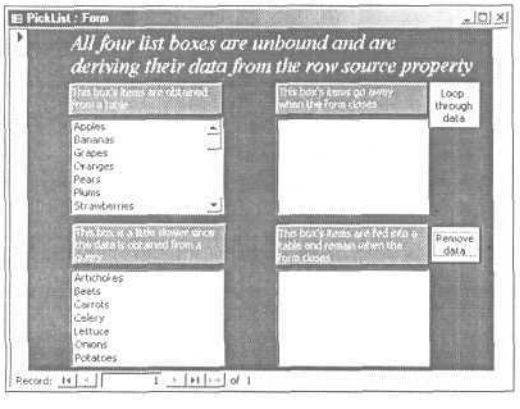


Рис. 7.9. Используйте форму «Выбираемый список» (Pick List) для переноса компонентов из одного списка в другой

Интересным моментом в данной форме является то, что все четыре списка (List Box) являются непривязанными. Свойство «Данные» для элемента управления (Control Source) нигде не установлено, также не установлен источник записей (Record Source) для формы. Тем не менее вы можете как извлекать данные, называемые Fruit, Vegetables и Grocery, так и добавлять их в эти таблицы. Это осуществляется просто выделением нужных компонентов (щелкнув на них мышью) и перемещением их из одного списка в другой.

В качестве простой демонстрации давайте рассмотрим бакалейный список овощей и фруктов. Он демонстрирует работу источника строк.

1. В папке AccessByExample откройте базу данных FormsAndControls.
2. В разделе **Объекты** выберите **Формы.** Откройте форму «PickList», дважды щелкнув мышью на ней.
3. В списке, отмеченном «This box's items are obtained from a table», выберите три любых фрукта, щелкнув мышью на первых двух по одному разу, а на последнем — дважды. Заметьте, что двойной щелчок переносит компоненты в правый список.
4. Нажмите кнопку «Loop Through Data» (пролистать данные). Вы сможете последовательно просмотреть каждый компонент.
5. Закройте форму и откройте ее заново. Обратите внимание, что компоненты в правом верхнем списке исчезли.

В упражнении, которое вы только что выполнили, выбранные вами компоненты, которые должны быть добавлены, включаются в свойство элемента управления, называемое ItemsSelected (дословно - «Выбранные компоненты» - *Пер.*). При помощи программы, которую вы можете изучить далее, событие «двойной щелчек мыши» инициирует процедуру, которая пробегает по выделенным компонентам и копирует их в свойство «Источник строк» (Row Source) правого списка. Тип источника строк (Row Source Type) для этого списка должен быть установлен как «Список значений» (Value List). Хотя данные были введены процедурой в источник строк как список значений, доступ к ним (вероятно, после нажатия кнопки Loop Through Data - *Пер.)* происходит при обращении к свойству Column элемента управления.

И это понятно - зачем добавлять компоненты в список, если к ним потом нельзя получить доступ. Если бы ваш выбираемый список был отчетом, вы возможно, захотели бы вывеси этот список на печать. Поскольку список значений ограничен по размеру и значения содержатся в памяти только тогда, когда форма открыта, этот метод скорее подходит для коротких списков, от которых не требуется большой сложности и которые быстро создавать. Рис. 7.10 изображает программу, которая управляет первым из тестируемых нами выбираемых списков.

Перед тем как переходить к следующему выбираемому списку, давайте взглянем на программу, спрятанную «за кулисами». Построчный анализ этой программы приведен сразу же за ее исходным кодом.

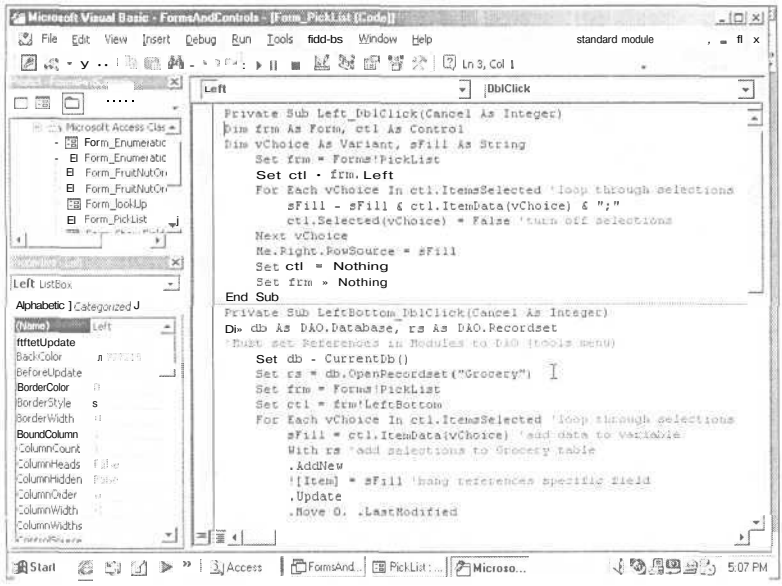


Рис. 7.10. Вы можете изучить программу, управляющую работой списка, чтобы узнать, что происходит «за кулисами»

1. Открыв базу данных FormsAndControls, выберите форму PickList *и* нажмите **Конструктор.**
2. Щелкните мышью на верхнем левом списке и нажмите F4 для открытия списка свойств.
3. Выберите вкладку **События** (если она еще не выделена) и нажмите кнопку **Создать** напротив события **Двойное нажатие кнопки** (On Dbl Click), убедившись, что установлена [Процедура обработки событий].

   Вы должны увидеть следующую программу:

```
Private Sub Left_DblClick(Cancel As Integer)
Dim frm As Form, ctl As Control
Dim vChoice As Variant, sFill As St   ring
  Set frm - Forms!PickList
  Set ctl = frm.Left
  For Each vChoice In ctl.ItemsSelected 'пробегает по выбранным компо-
нентам
    sFill = sFill & ctl.ItemData(vChoice) & ";"
    ctl.Selected(vChoice) = False 'снимает выделение
  Next vChoice
Me.Right.RowSource = sFill
  Set ctl = Nothing
  Set frm = Nothing
End Sub
```

Проанализируем эту программу. Процедура сначала использует ключевое слово Set, чтобы установить указатели на объектные переменные для формы и элемента управления. Дальше она пробегает коллекции выделенных компонентов (ItemsSelected) с целью определить, какие фрукты были выделены щелчком мыши. Переменная fillCtl (возможно, опечатка в оригинале; должна быть переменная sFill. - *Пер.)* заполняется последовательно по одному компоненту, в качестве разделителя между компонентами ставится точка с запятой. В списке значений, указанном в источнике строк, должен быть перечень, выглядящий так: Компонент1; Компонент 2; Компонент 3. К свойству RowSource (Источник строк) обращаются после имени правого списка, который называется Right.

Ключевое слово Me доступно для любой процедуры в модуле классов. Вы можете использовать его для обращения и изменения элементов управления. Наберите Me, затем точку в модуле классов, таком, как, например, для формы, и вы увидите список объектов и процедур, доступных в этой форме. Процедура могла бы с тем же успехом использовать другую объектную переменную для обращения к правому списку. Ключевое слово Me используется для заполнения правого списка путем присвоения значения свойству элемента управления Right. Последнее, что делает процедура, - убирает выделение. Когда это происходит, вы видите, что все подсвеченные компоненты возвращаются к своему нормальному состоянию. Таким образом подготавливается плацдарм для следующей попытки, если того захочет пользователь.

Теперь давайте поэкспериментируем со следующим выбираемым списком. В отличие от предыдущего этот список является перманентным. Помните, что понятие неизменности на компьютерном жаргоне означает «сохранность» или неизменность до тех пор, пока *вы* это не измените. Кроме того, эта программа немного по-другому обращается с выборкой. Она ждет, пока вы не удалите выбранный список, чтобы снять выделение в нижнем левом списке.

Следующие инструкции помогут вам выбрать компоненты из второго выбираемого списка.

1. Закройте окно модуля и список свойств, оставшиеся от предыдущего примера. Закройте и откройте форму заново либо (тот же результат, но быстрее) нажмите кнопку Вид в конструкторе.
2. Выделите три овоща из нижнего левого списка, дважды щелкнув на последнем (рис. 7.11). До сих пор все идет почти так же, как и в прошлый раз, за исключением того, что двойной щелчок не снял выделение.
3. Нажмите кнопку Remove Data. Обратите внимание, что выделение в левом нижнем списке снялось, так что теперь вы можете снова выбирать.
4. Выберите три различных компонента, дважды щелкнув на последнем, как и раньше. Закройте и откройте форму заново. Заметьте, что данные в правом нижнем списке сохранились.
5. Нажмите кнопку Remove Data.

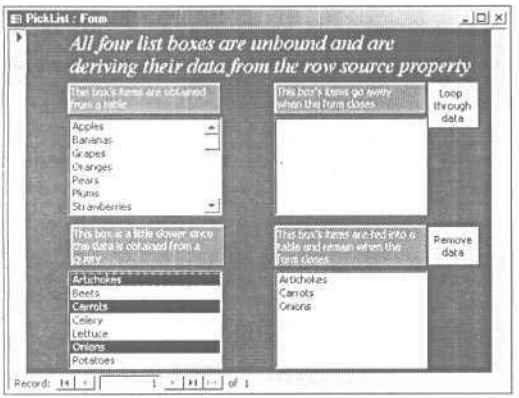


Рис. 7.11. Вы можете использовать источник строк элемента управления «Список», чтобы сохранить выбранные данные, когда форма закроется

Этот пример демонстрирует способ сохранения выбранного списка для дальнейшего использования. Данные сохраняются в таблице, называемой Grocery (Бакалея). Вы можете составить бакалейный список сейчас и использовать его позже. Вы также можете использовать выбранный список в условия отбора в запросе или в отчете. Теперь давайте взглянем «за кулисы» на управляющую списком программу.

1. Находясь в форме PickList, нажмите кнопку **Вид,** чтобы переключиться на вид **Конструктор.** Щелкните правой кнопкой мыши на левом нижнем списке и выберите **Свойства.**
2. Убедитесь, что выбрана закладка **События** и нажмите кнопку **Создать** напротив события **Двойное нажатие кнопки.** Обратите внимание: в комментариях указано, что должна быть установлена ссылка на DAO. Также обратите внимание на переменные типа Database и Recordset (дословно - «База данных» и «Совокупность записей». – *Пер.*).

Давайте разберем по винтикам эту программу:

```
Private Sub LeftBottom_DblClick(Cancel As Integer)
Dim db As DAO.Database, rs As DAO.Recordset
'Должна быть установлена ссылка на DAO (меню Tools в окне Module)
  Set db = CurrentDb()
  Set rs = db.OpenRecordset("Grocery")
  Set frm - Forms!PickList
  Set ctl = frm!LeftBottom
  For Each vChoice In ctl.ItemsSelected 'пробегает по выбранным
                                        'компонентам
    sFill = ctl.ItemData(vChoice) 'добавляет данные в переменную
    With rs 'добавляет выбранные компоненты в таблицу Grocery
    .AddNew
    ![Item] = sFill 'восклицательный знак позволяет обращаться
                    'к определенному полю
    .Update
    .Move 0,.LastModified
    End With
```

```
  Next vChoice
Set ctl = Nothing
Set frm = Nothing
rs.Close
db.Close
Set rs = Nothing
Set db = Nothing
Me.Refresh
End Sub
```

В первых двух строчках после объявления переменных (не считая комментариев) при помощи ключевого слова Set устанавливаются указатели на объектные переменные для к доступа табличным данным. Просто укажите имя, которое хотите присвоить таблице, либо задайте переменную для того, чтобы это имя мог ввести пользователь. Функция CurrentDb - это простой способ обращения к текущей базе данных, которой занимается пользователем. Другой способ обращения к текущей базе данных - при помощи DBEngine(0)(0). При использовании этого метода требуется обновление (Refresh), прежде чем вы сможете использовать коллекцию, связанную с ним.

Таблица не единственный объект типа Recordset, который вы можете открыть. Точно так же вы можете открыть Recordset для запроса, использовав инструкцию SQL совместно с методом OpenRecordset, но в данном случае вам просто нужен доступ к табличным данным.

В следующих двух строчках используется ключевое слово Set для указания объектных переменных на текущую форму и на элемент управления «Список». После того как процедура запросит выбранный компонент из элемента управления, она добавляет данные через таблицу исходных данных в список справа внизу. Синтаксическая конструкция после выражения With может быть использована снова и снова для добавления и обновления табличных данных. Поэтому она вставлена в цикл For Each. Выражение Refresh обновляет экран, чтобы вы могли видеть изменения.

Следует отметить, что тип источника строк (Row Source Type) правого нижнего списка установлен в значение «Таблица или запрос» (Table/Query), в отличие от «Списка значений» в предыдущем примере. Не имеет значения то, что элемент управления является непривязанным, поскольку вы не используете свойство «Данные» (Control Source). Если вы нажимаете кнопку Remove Data, инструкция SQL позаботится об удалении данных. Вы можете увидеть следующий код, щелкнув правой кнопкой по правому нижнему списку и рассмотрев событие «Нажатие кнопки» (On Click) для этого элемента:

```
DoCmd.SetWarnings False
DoCmd.RunSQL "Delete * from Grocery" 'очищает все записи в таблице
DoCmd.SetWarnings True
  Set frm = Forms!PickList
  Set ctl = frm!LeftBottom
  For Each vChoice In ctl.ItemsSelected
    ctl.Selected(vChoice) = False 'снимает выделение
  Next strChoice
Set ctl = Nothing
Set frm - Nothing
Me.Refresh
```

Использование списка для поиска

Традиционная возможность электронных баз данных - поиск. Вам может понадобиться найти адрес заказчика, или вы можете забыть инвентарный номер продукта. Следовательно, вам нужно увидеть список продуктов. Вам также может потребоваться увидеть сам продукт, вместе со своим кодом. В результате поиска вы сможете заполнить соответствующие поля элементов управления синхронизованнымиданными.

Давайте рассмотрим несколько методов, используемых элементами управления для поиска по таблицам. В первом способе используется запрос для поиска данных. Второй применяет пересчитываемые поля для получения искомых данных. Третий метод комбинирует поиск и пересчет. В первых двух методах источник данных для элементов управления (Control Source) расположен в поле Lookup. Это гарантирует сохранение данных после закрытия формы.

Следующие инструкции помогут вам познакомиться с этими различными способами.

1. Находясь в базе данных FormsAndControls, дважды щелкните по форме LookUpList, чтобы ее открыть.
2. Нажмите кнопку New Record (Новая запись) для добавления новой записи в таблицу исходных данных.
3. Щелкните по верхнему ниспадающему списку Zip (почтовый индекс) и выберите 49344. Обратите внимание, что в полях для результатов поиска автоматически появилось Shelbyville, Michigan (г. Шелбивиль, шт. Мичиган) - рис. 7.12.

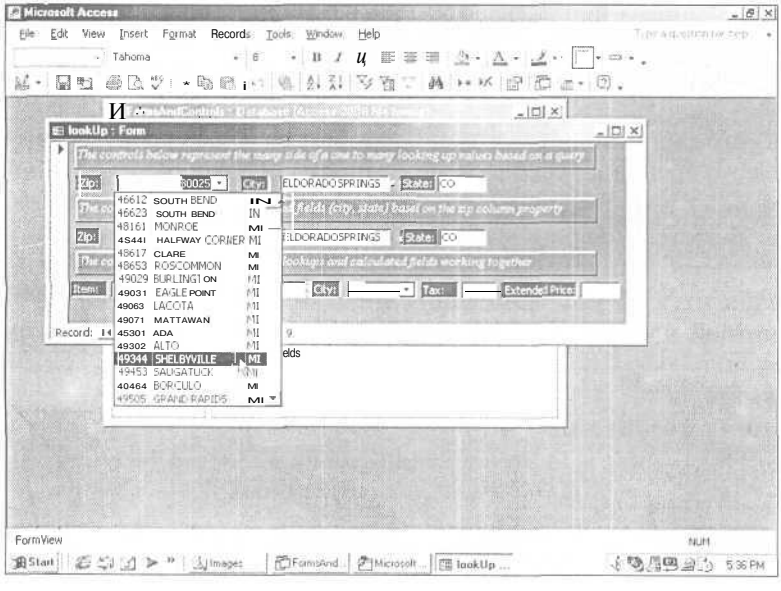


Рис. 7.12. Выбор компонента из списка поиска, автоматически расставляющий правильную информацию в соответствующие поля

Теперь давайте рассмотрим запрос, стоящий за первым методом поиска. Этот запрос - JoinZip - является источником записей для формы.

1. В разделе **Объекты** выберите **Запросы.**
2. Выделите JoinZip и нажмите **Конструктор.** Обратите внимание, что тип отношения между двумя таблицами - «один ко многим».
3. Нажмите иконку запуска (!), чтобы начать выполнение запроса.
4. Наберите 46526 в поле Zip новой записи (*) и нажмите Tab (рис. 7.13). Заметьте, что соответствующая информация ввелась автоматически.

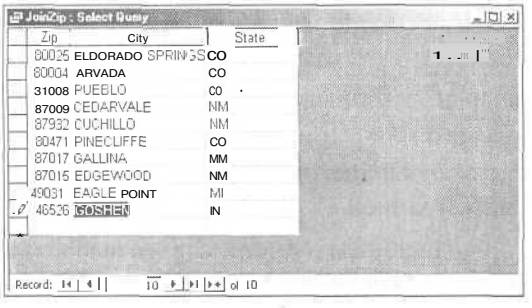


Рис. 7.13. Тестирование поискового запроса, обеспечивающего функционирование формы

Этот метод полезен и интересен сам по себе. Когда вы вводите информацию со стороны «многие», появляется соответствующая информация со стороны «один». Что действительно интересно, так это то, как этот метод работает. Информация о почтовом индексе фактически добавляется в таблицу исходных данных LookZip, олицетворяющую сторону «многие». Однако со стороны «один» ничего не добавляется. Эта сторона просто динамически *отражается.*

Причина, по которой сказанное выше существенно, состоит в том, что это является краеугольным камнем модели межтабличных отношений. В идеале нужно стремиться к тому, чтобы данные нигде не вводились больше чем в одном месте. Но вам приходится повторять данные в объединяемых полях для связывания данных в таблице. Вместо этого вы просто отображаете данные, которые уже существуют. Если вы хотите импортировать данные из 30 или 40 полей, берите и делайте. Вам не придется беспокоиться о повторении чего-либо: кроме тех полей, которые уже существуют, ничего не понадобится.

Некоторые пользователи баз данных применяют функцию Dlookup для запроса данных из других таблиц. Для этой функции тоже найдется область применения, но обычно существует лучший способ поиска данных. Копирование данных из одной таблицы в другую при поиске приводит к тому, что они оказываются введены более чем в одном месте. Более корректным методом будет динамическое отражение данных, особенно если количество полей с данными, которые нужно вставить, существенно.

При некоторых обстоятельствах использование метода поиска через запрос становится невозможно. Предположим, в качестве источника записей для формы уже установлен другой запрос. Предположим также, что этот запрос не является запросом для поиска или содержит не те поля данных, которые необходимы для вашего поиска. Даже если вы получите доступ к поисковому запросу через свойство источника строк (Row Source), вам по-прежнему нужен будет способ получения столбцов этого запроса. К счастью, существует другой метод динамического поиска данных, причем относительно простой. И хотя в данном случае представленная ниже группа элементов управления не является необходимой (так как первая группа также может отыскивать значения), эта группа иллюстрирует совершенно иной метод поиска, несмотря на то что он выглядит и кажется точно таким же.

1. Из базы данных FormsAndControls в разделе **Формы** дважды щелкните на форме LookUpList, чтобы открыть ее.
2. Нажмите кнопку New Record. Обратите внимание, что все поля на форме пусты, отображая готовность к вводу новой записи.
3. Щелкните по стрелочке на втором ниспадающем списке Zip и выберите 49031, как показано на рис. 7.14. В обеих группах должно появиться Eagle Point, Michigan.

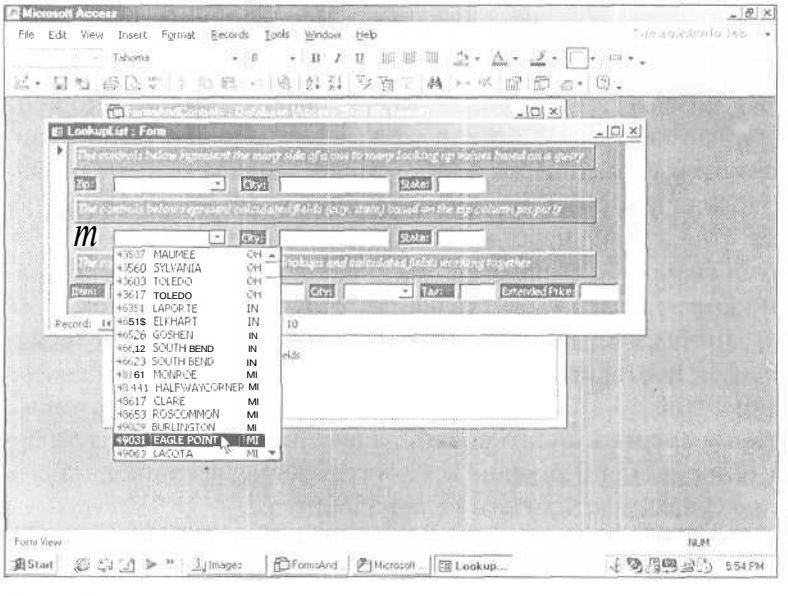


Рис. 7.14. Вы можете выбрать компонент и использовать свойство Column другого элемента управления для получения информации об этом компоненте

На данный момент невозможно заметить разницу между этими двумя методами. Пришло время найти эти различия, включив вид **Конструктор.**

1. Нажмите кнопку **Вид** для переключения в конструктор.
2. Щелкните правой кнопкой по полю со списком (Combo Box) Zip и выберите **Свойства.**
3. Выберите вкладку Все (АИ). Убедитесь, что вертикальная полоса прокрутки установлена в крайнее верхнее положение.
4. Обратите внимание, что число столбцов (Column Count) равно трем. Также заметьте, что источником строк (Row Source) служит таблица Zipcodcs.
5. Источником данных (Control Source) является таблица Zip.
6. Не закрывая список свойств, щелкните по полю со списком Zip во второй группе. Вам, возможно, понадобится передвинуть список свойств, перетянув его за заголовок, чтобы увидеть нужную часть формы.
7. Убедившись, что вы в самом верху списка свойств, заметьте, что свойства «Источник строк» (Row Source), «Данные» (Control Source) и «Число столбцов» (Column Count) ничем не отличаются от первого элемента, рассмотренного выше. Так в чем же разница?
8. Щелкните по полю City второй группы элементов. Обратите внимание на выражение, записанное в этом элементе управления (рис. 7.15). Закройте форму.

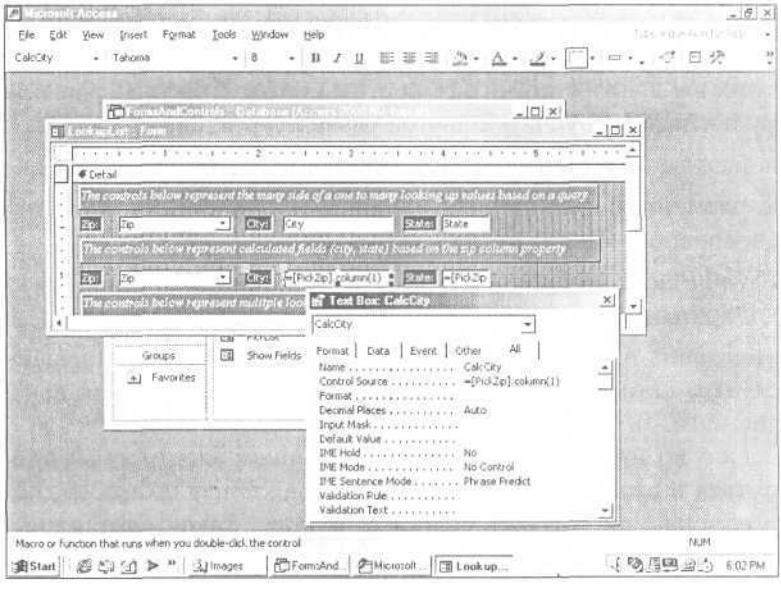


Рис. 7.15. Посмотрите на выражение, записанное в поле City второго ряда элементов, чтобы понять, как работает второй метод поиска

Различие между этими двумя методами заключается в выражениях, записанных в полях City и State второй группы. Этот метод работает, даже если к форме присоединен совершенно другой запрос, не способный производить поиск. Таблица должна иметь поле Zip, чтобы к нему можно было присоединить элемент

управления. В выражении `=[PickZip].column(1)` часть Pickzip - это имя элемента управления, а не его источника данных (Control Source). `Column(1)` обозначает второе поле таблицы. Для элемента State выражение совершенно аналогично, за исключением обращения к `Column(2)`.

В двух словах, первая группа элементов управления, предназначенных для поиска, должна иметь в качестве источника данных (Control Source) запрос JoinZip, в то время как вторая группа может работать и с другим запросом или с таблицей в качестве источника данных, лишь бы он имел поле Zip. Первый метод для поиска использует связи между таблицами и объединение полей таблицы; второй метод может работать без объединения, употребляет выражения. Оба метода имеют свои «за» и «против», в зависимости от ситуации.

Если вы выбрали в качестве источника данных (Control Source) таблицу, используйте второй метод. Если в качестве источника данных вы выбрали запрос, не использующий связи между таблицами, опять-таки применяйте второй метод. Но если в качестве источника данных у вас есть запрос, использующий связи и правильно сконфигурированный, даже если эти связи не определены в окне «Схема данных» (Relationships window), используйте первый метод.

Комбинированный вариант

Решение некоторых задач приводит к использованию как поиска, так и пересчитываемых полей. Когда вы покупаете продукт, вам нужно, во-первых, найти его цену, но, кроме того, вы должны посчитать местный налог с продаж. Давайте посмотрим, что делает последняя группа элементов управления в данной форме.

1. В разделе Формы дважды щелкните по форме LookupList, чтобы открыть ее.
2. В нижней группе элементов щелкните по ниспадающему списку элемента управления Item (дословно - «Вещь, компонент». - *Пер.*).
3. Выберите 15 inch Wall Plaque (дословно - «Настенное украшение, размер - 15 дюймов». – *Пер.*).Обратите внимание, что автоматически появилась цена.
4. Выберите название города Alden из ниспадающего списка элемента управления City. Заметьте, что автоматически появились налог и совокупная цена, как показано на рис. 7.16.

Работа с этим методом во многом схожа с использованием электронных таблиц; вы можете вернуться и ввести другой продукт, и совокупная стоимость будет посчитана с использованием того же налога с продаж. Такая комбинация поиска величин с последующим пересчетом иллюстрирует то, как поисковые элементы управления могут работать совместно с пересчитываемыми полями.

1. Нажмите кнопку **Вид** для переключения в вид Конструктор.
2. Опустите вертикальную полосу прокрутки вниз до упора, чтобы видеть последнюю группу элементов управления.
3. Обратите внимание на расстановку в группе. Сначала элемент управления, используемый для поиска, затем колонка, являющаяся результатом поиска. Находится цена, а затем налог.

4. Последний элемент управления справа собирает вместе все результаты. Выражение в нем =[price]+([price]*[tax]). Как price, так и tax являются именамиэлементовуправления.

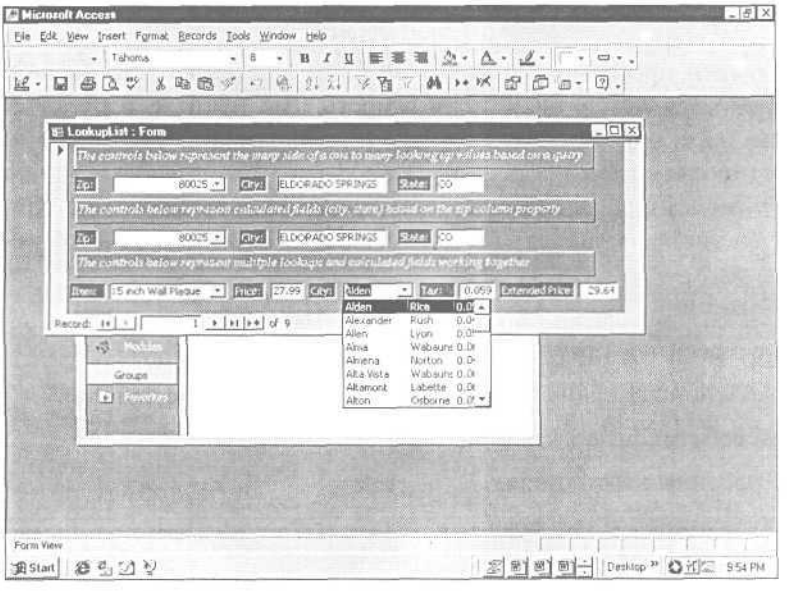


Рис. 7.16. Вы можете комбинировать возможности поиска и пересчитываемых полей

Что дальше?

Теперь, когда вы представляете, что такое объекты и их свойства, перейдем к следующей главе, которая поведет вас дальше к программированию на VBA, которое вы в конце концов будете применять к объектам. Эта глава поможет вам сделать первые шаги по написанию первой программы. Вы узнаете все о функциях и их возможностях.

8
Написание программы на Visual Basic for Applications

До сих пор вы изучали программные коды, но сами еще ничего не писали. В этой главе вам предоставляется такая возможность. Вы начнете с простых процедур, постепенно переходя к более сложным и мощным. Вы также научитесь проверять и тестировать свои процедуры. После того как вы поймете, что можно сделать при помощи сплава возможностей функций и запросов, не удивительно, что у вас внезапно появится мотивация начать написание своих, нетривиальных процедур.

В этой главе вы узнаете:

- как использовать встроенные строковые функции Access,
- как использовать «Окно проверки» (Immediate) для тестирования функций,
- как создавать свои собственные функции,
- как использовать массивы в функциях,
- как комбинировать функции и запросы.

Функции, подпрограммы и модули

Visual Basic for Applications (VBA, Visual Basic для приложений) - язык программирования, который устанавливается вместе с любым приложением Office, в частности с Access 2002. В VBA используется три типа программных структур: подпрограммы *(sub* procedures), функции *(function* procedures) и процедуры-свойства *(property* procedures). Можете считать их своим «хлебом и *маслом»* при разработке автоматизированной базы данных. Данная глава охватывает первые два из указанных типов.

Для получения дополнительной информации о процедурах см. раздел «Функции и подпрограммы» в гл. 10.

Место, где в Access пишется программный код VBA - это *модуль:* множество процедур VBA, хранящихся вместе как одно целое. Модули не запускаются; они содержат в себе запускаемые процедуры. В Access используется два вида модулей - стандартный (standard) модуль и модуль класса (class). Стандартные модули расположены в нижней строчке раздела **Объекты** в окне «База данных». В них вы можете размещать подпрограммы и функции, которые могут после этого использоваться любой другой процедурой в любом месте базы данных.

Модуль класса может содержать определение нового объекта. Поэтому не удивительно, что модули класса связаны с формами и отчетами, которые являются новыми объектами, создаваемыми вами. Однако вы можете создавать модули класса, которые не связаны с формой или отчетом. Этот тип модулей клас-

са дает вам возможность создавать и модифицировать свои собственные пользовательские объекты, с приминением программного кода.

вам, наверное, интересно, в чем различие между функцией и подпрограммой. *Функция (function* procedures) может получать и возвращать значения, обрабатывать данные и являться частью выражения. *Подпрограмма (sub* procedures) же, напротив, может получать значения, но не может значения возвращать. Она не может быть использована как часть выражения. Функции могут быть использованы в запросах, подпрограммах, пересчитываемых элементах управления и макросах. Подпрограммы чаще используют в классах модуля для форм и отчетов. Процедуры, которые связаны с формами и отчетами, еще называют *процедурами событий.*

Строковые функции Access

Простая причина, по которой нужно использовать встроенные строковые функции, состоит в том, что они очень полезны всем, кто хочет хорошо разбираться в проектировании баз данных Microsoft Access. Базы данных предназначены для хранения и управления данными. Запросы могут управлять данными, но имеют ограничения. Предположим, вы хотите выделить имя из ФИО (здесь и далее мы будем предполагать, что ФИО=[имя]+[иногда отчество или второе имя]+[фамилия]) сотрудника в столбце таблицы, имеющей 30.000 записей. Как вы можете сделать это в запросе без использования строковых функций? вы можете использовать SQL-строки в VBA, но даже в них лучшим способом извлечения данных является использование строковых функций. Вы также можете воспользоваться строковыми функциями в вашей собственной функции, разработанной для нужд вашего приложения.

Строковые функции не единственные полезные встроенные (внутренние) функции Access, но они, вероятнее всего, станут функциями, которые вы будете использовать чаще всего. Как и остальные встроенные функции, строковые функции имеют аргументы, заключенные в скобки. Некоторые аргументы являются обязательными, другие можно опустить. Каждый аргумент может принадлежать к своему типу данных. Например, в функции left, первый аргумент должен быть строкой, а второй - числом (длинное целое, long integer). Таким образом, ее синтаксис следующий:

```
Left(string, length)
```

Если вы наберете ? `left("Park Place", 4)` в окне проверки (Immediate), вам буден возвращено Park (первые четыре символа из строки) `Park Place` является первым аргументом, 4 - вторым. Заметьте, что число 4 не нужно заключать в кавычки, так как это число. Вы вводите два значения, а получаете только одно. Оба аргумента являются обязательными для функции left. Если вы попробуете ввести только один аргумент, вы получите ошибку при компиляции.

Аргументами могут быть константы, переменные или выражения. *Константа* — значение, которое не может быть изменено. *Переменная* — именованное место хранения данных, которое ведет себя как временное хранилище. Имя пе-

ременной должно начинаться с буквы алфавита. В соответствии со своим названием содержимое переменных часто меняется. *Выражение* может содержать другие функции, они могут образовывать «матрешки»: функция, а точнее, ее значение может являться аргументом другой функции.

«Окно проверки» (Immediate) в Access

Один из самых простых и наиболее полезных инструментов, с которыми вы столкнетесь при изучении Access, - «Окно проверки» (Immediate), являющееся одновременно окном с командной строкой и окном отладки. Оно является окном командной строки, потому что команда и процедура могут быть набраны прямо в нем в строчке и будут запущены после того, как вы нажмете «Ввод» (Enter). Окном отладки оно является потому, что отлаживаемые выражения, размещенные в вашей программе, будут отображаться в нем же. Также вы можете запускать в нем свои процедуры для проверки. Хотите ли вы проверить переменную, выражение или целую процедуру - в окне проверки можно сделать и то, и другое, и третье. После того как вы познакомитесь с окном проверки, оно станет для вас просто необходимым инструментом, без которого как без рук.

Давайте начнем с того, что сосчитаем количество букв в вашем имени.

1. Откройте базу данных Music Store из папки AccessByExample и нажмите Ctrl+G для открытия окна проверки Access.
2. Наберите ? `left(First&LastName, NumberOfLettersInFirstName)` и нажмите **Ввод** (Enter). Например, John Smith должен набрать ? `left("John Smith", 4)`. Обязательно заключите ваше ФИО в кавычки. Вы должны получить ваше имя (рис. 8.1).

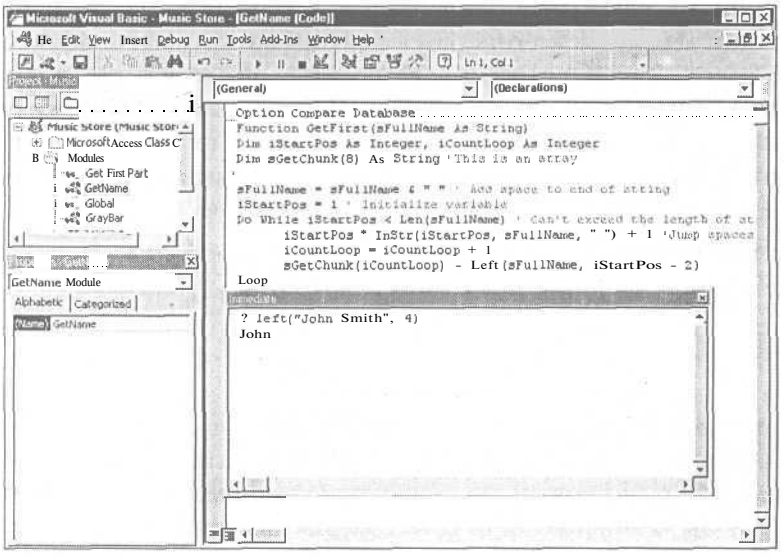


Рис. 8.1. Вы можете проверить строковую функцию left в окне проверки

Все замечательно, если вам нужно выделить только ваше имя, поскольку вы знаете, сколько в нем букв. Но предположим, что у вас список ФИО. Количество букв в имени, очевидно, не одно и то же для всех. Вам нужен способ, которым вы найдете положение пробела, затем отступите на одну позицию назад к последней букве в имени. Таким образом вы получите число, которое вам нужно. Строковая функция `InStr` в Access дает вам этот способ отыскания положения пробела.

Если вы вводите `InStr("John Smith", " ")`, вы приказываете: найти положение пробела в строке `"John Smith"`. Первый аргумент - это строка, в которой вы хотите искать. Второй - это символ или строка символов, которые вы хотите найти. Если бы вы набрали это в окне проверки, функция вернула бы результат «5», т. е. позицию пробела. Однако положение буквы «n» в слове John (оно же - число букв в имени), на одну позицию левее, т. е. «4».

Предположим, вы решили собрать обе функции вместе. Это будет выглядеть следующим образом:

```
Left("John Smith", InStr("John Smith", " ") -1)
```

Хотя это выражение потребовало для написания больше ударов по клавиатуре, оно того стоит. Теперь не имеет значения, из скольких символов состоит имя. Оно будет работать с "Elizabeth Williams" так же, как и с "John Smith" (рис. 8.2). Функция `InStr` стала вторым аргументом, генерируя число, которое будет изменяться в зависимости от того, сколько букв содержится в имени. Обратите внимание на «минус один» (`-1`). Таким способом вычитается один символ и мы передвигаемся назад к букве «п» в слове John.

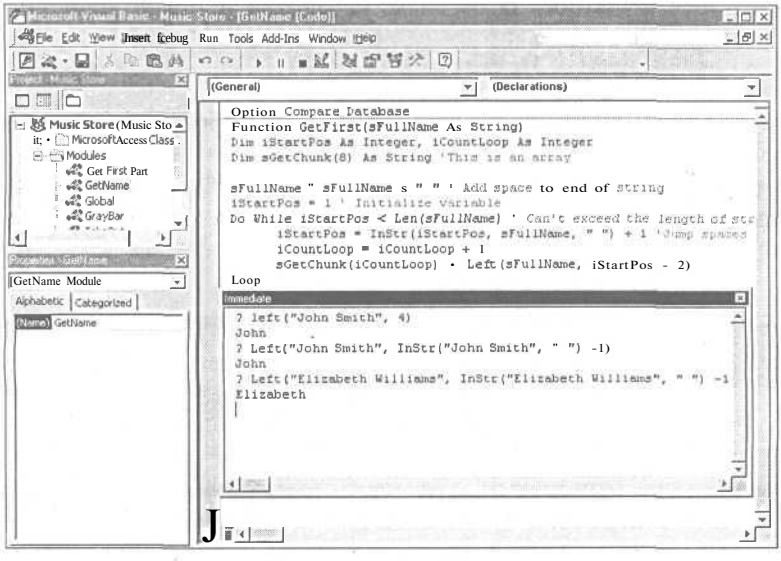


Рис. 8.2. Вложенные функции работают независимо от размеров имени и фамилии

Попробуйте написать свое имя вместо "John Smith" в предыдущем выражении в окне проверки. Не забудьте, что начинать нужно с вопросительного знака (?), а заканчивать - нажатием ввода (Enter). Вместо того чтобы печатать все это выражение каждый раз при использовании данной комбинации функций, почему бы не попробовать объединить все это в новую функцию.

Следующий пример демонстрирует вам, как можно написать свою собственную процедуру, которую затем можно будет использовать, когда понадобится.

1. В окне **База данных** (Database) базы данных Music Store, выберите **Вставка** (Insert), **Модуль** (Module).
2. В окне модуля после строчки объявления наберите следующее:
   ```
   Function Showame(StrVal As String) as String
   ```

После того как вы нажали Ввод (Enter), автоматически будет вставлена строчка End_Function. До этого вы объявили вашу процедуру как функцию. Вы также объявили, что вводимая переменная, которая является единственным аргументом, будет строковой. Для этой простой процедуры вам понадобится только эта одна переменная. В следующей строке наберите следующее:

```
StrVal = Left(StrVal, InStr(StrVal, " ") -1)
```

Заметьте, что мы использовали тот же метод, что и ранее, только вместо "John Smith" была подставлена переменная StrVal. Заметьте также, что StrVal используется три раза. Таким образом, переменная StrVal приобретает значение, состоящее из крайних левых символов, составляющих имя. В итоге процедура выделяет имя из ФИО.

Но остался еще один момент. Функция должна еще вернуть значение. Это делается так: пишется имя функции, затем знак равенства и новое значение. Взгляните на это:

```
ShowName = StrVal
```

Именно так! Просто, да? Если вы соберете все вместе, ваша программа должна будет выглядеть так:

```
Function ShowName(StrVal As String) as String
StrVal = Left(StrVal, InStr(StrVal, " ") -1)
ShowName = StrVal
End Function
```

Если вы хотите сэкономить на одном шаге, есть способ написать функцию, которая даже проще:

```
Function ShowName(StrVal As String) as String
ShowName = Left(StrVal, InStr(StrVal, " ") -1)
End Function
```

Процедура в этом примере отрезает правую часть выражения и возвращает значение, и это - в одной строчке, расположенной между Function и End Function. Единственное различие состоит в том, что вы присваиваете значение, возвращенное функцией left непосредственно пользовательской функции ShowName, вместо того чтобы сначала присваивать его переменной StrVal. Посмотрите результаты работы этой функции на рис. 8.3.

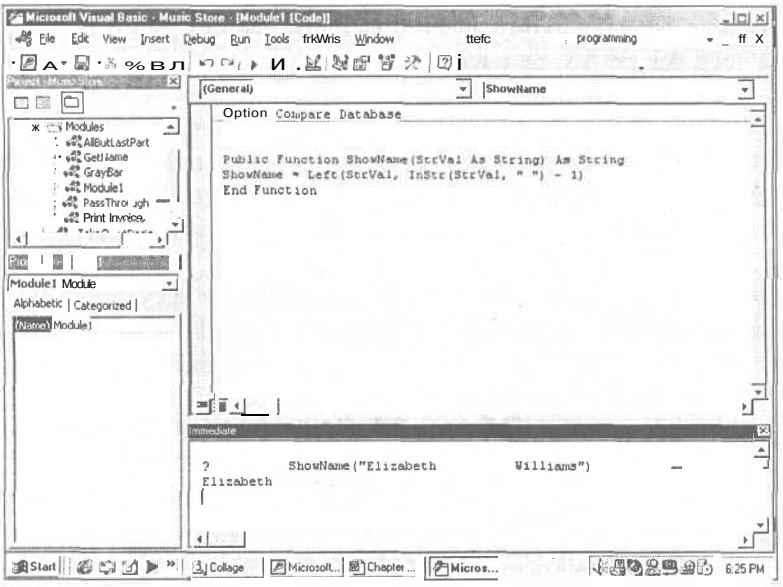


Рис. 8.3. Вы можете использовать вложенные функции снова и снова, если поместите их в пользовательскую функцию

Теперь давайте опробуем ее. Если окно проверки еще не открыто, нажмите Ctrl+G. В этом окне наберите (подставив свое ФИО вместо John Smith)

```
? ShowName("John Smith")
```

После нажатия ввода (Enter) вы должны увидеть свое имя. Теперь попробуйте другое имя, просто чтобы проверить, что все работает правильно. Если проверка прошла удачно, значит, вы только что написали свою первую функцию. Если ничего не получается, вернитесь и перепроверьте ваш программный код, нет ли там опечаток.

Это маленькая, но мощная функция. Однако есть небольшая проблема. Предположим, у человека двойное имя (например, Channing Creighton Capwell - Си-Си Кепвэл - из сериала «Санта Барбара». В этой ситуации функция, которую вы только что создали, вернет только первое имя. Вам нужна функция, которая знала бы, сколько слов содержится в строке, и отсеивала последнее (фамилию).

Совместное использование массивов и строковых функций

Функция, которую вы скоро увидите, считает слова; но это не главное. Она раскладывает строку по частям - по одному слову за шаг. Она использует тип переменных, который называется *массивом.* Массивы отпугивают некоторых людей, но на самом деле это достаточно простая вещь. Вместо того чтобы заполнять одну переменную одной строкой за шаг, использование массива позволяет запомнить сразу всю совокупность строк и обращаться к любой из них, когда понадобится.

Представляйте себе массив как группу ячеек в таблице. Если у вас информация находится в ячейках с A1 по A3, то у вас есть массив ячеек. Если A1 содержит FirstName (имя), A2 содержит MiddleName (отчество или второе имя) и A3 содержит LastName (фамилия), вы можете получить содержимое, щелкнув мышью на соответствующей ячейке. Обращение с переменной-массивом во многом схоже. Если имя массива Fill, вы можете объявить Fill следующим выражением: Dim Fill(2) as String, где «(2)» просто говорит процедуре выделить три места для хранения переменных. Их выделяется именно три, а не два, так как отсчет идет от нуля. В результате вы резервируете три блока памяти для трех элементов массива. Fill(O) - совершенно другая переменная, нежели Fill(1). Табл. 8.1 иллюстрирует, каким способом хранятся переменные в нашем примере.

Таблица 8.1. Пример хранения массива

Переменная	Хранимая информация
Fill(O)	FirstName (имя)
Fill(I)	MiddleName (отчество или второе имя)
Fill(2)	LastName (фамилия)

Вы можете легко заполнить все переменные массива при помощи цикла. В следующей процедуре в качестве исходной информации вы получаете порции слов. Цикл говорит вам, сколько слов содержится в строке. В каждом следующем элементе массива содержится на одно слово больше, чем в предыдущем. Рассмотрите этот программный код.

```
Function GetFirst(sFullName As String)
Dim iStartPos As Integer, iCountLoop As Integer
Dim sGetChunk(8) As String 'Это массив
sFullName - sFullName & " " 'Добавляет пробел в конце строки
iStartPos = 1 'Задает начальное значение переменной
Do While iStartPos < Len(sFullName) 'Не может превышать длины строки
   iStartPos = InStr(iStartPos, sFullName, " ") + 1 '«Прыгает» по пробелам
   iCountLoop = iCountLoop + 1
   sGetChunk(iCountLoop) = Left(sFullName, iStartPos - 2)
Loop
GetFirst - sGetChunk(iCountLoop - 1) 'Отсекает последнее слово
End Function
```

Переменная sGetChunk - это, очевидно, ваш массив. Хотя в нем отведено (или отмерено, dimensioned. - *Пер.)* девять элементов, вам скорее всего не понадобится так много слов. Если ваше имя - John Wilkes Booth, первый элемент массива (sGetChunk(1)) будет содержать John, второй - John Wilkes и т. д. Эта процедура устроена так, что нулевой элемент массива пропускается; таким образом, первый элемент массива, который вы заполняете, пронумерован как 1. Вы уже знакомы с функцией InStr; в этот раз впервые был использован первый, необязательный, аргумент. Таким образом, вы говорите функции InStr, откуда следует начать.

Синтаксис цикла do loop следующий:

```
Do...While [Condition is True]
Loop
```

Слова в скобках обозначают условие, обычно состоящее из выражения. (Дословный перевод всей конструкции: Делать. . . , Пока [Условие Истинно]; Следующий цикл. - *Пер.)* Пока это условие истинно, цикл выполняется. Когда условие прекращает быть истинным, процедура выходит из цикла и продолжает со следующей строки после Loop.

ПОДСКАЗКА. Вы также можете остановить выполняющийся цикл используя End Do.

Для получения дополнительной информации по различным типам циклов в VBA см. раздел «Создание процедур» в главе 10

В первой итерации цикла процедура ищет пробел начиная с первой позиции и устанавливает переменную iStartPos на одну позицию правее первого пробела (на первую букву следующего слова для поиска в следующей итерации цикла). Таким образом, она «прыгает» по словам. Если вы хотите увидеть, как процедура работает, из базы данных Music Store откройте модуль GetName в виде конструктора (Design view). Установите курсор на строке после последнего выражения Dim и нажмите F9. Так вы установите «точку останова» (breakpoint).

ЗАМЕЧАНИЕ. Также вы можете щелкнуть мышью на серой полоске, находящейся слева от строки, на которой вы собираетесь поставить точку останова (breakpoint) (рис. 8.4). На полоске появится большая точка, а сама строка будет подсвечена. Точка останова может быть выключена и включена снова также при помощи щелчка мыши; однако не пытайтесь установить точку останова на выражение Dim ни первым, ни вторым способом.

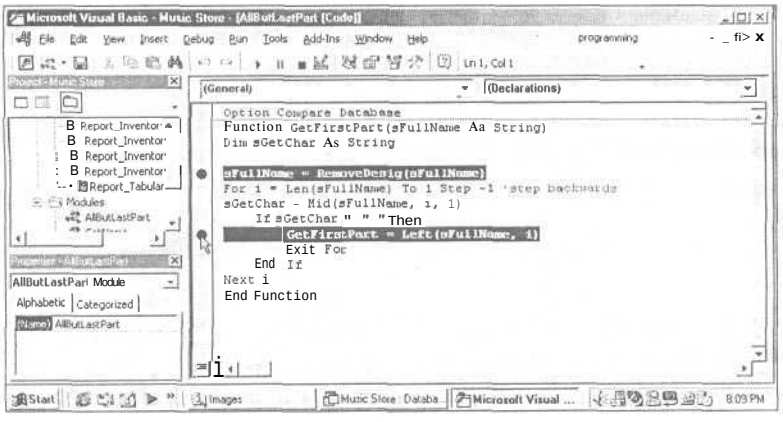


Рис. 8.4. Используйте мышь для установки точки останова, щелкая ей по вертикальной полоске в окне модуля

В следующем примере вам придется зайти «за кулисы» программирования на VBA и немного посмотреть на то, как процедура работает. Метод, изложенный здесь, может оказаться неоценимым, когда вы будете отлаживать свои собственные программы.

1. Нажмите Ctrl+G, чтобы открыть окно проверки. Наберите ? GetFirst("Mary Ann Mobley") и нажмите Ввод.

   Строка точки останова должна окрасится в желтый цвет (или в другой цвет, в зависимости от ваших настроек).

2. Поместите курсор на переменной sFullName. Всплывет табличка, на которой написано Mary Ann Mobley. Нажмите F8, которая является «горячей клавишей» для пункта меню **Шаг вперед** (Step Into). Вы можете производить пошаговое выполнение вашей программы, используя **Шаг вперед** (Step Into), или F8.

   Теперь над той же самой переменной высвечивается надпись Mary Ann Mobley с пробелом на конце (рис. 8.5).

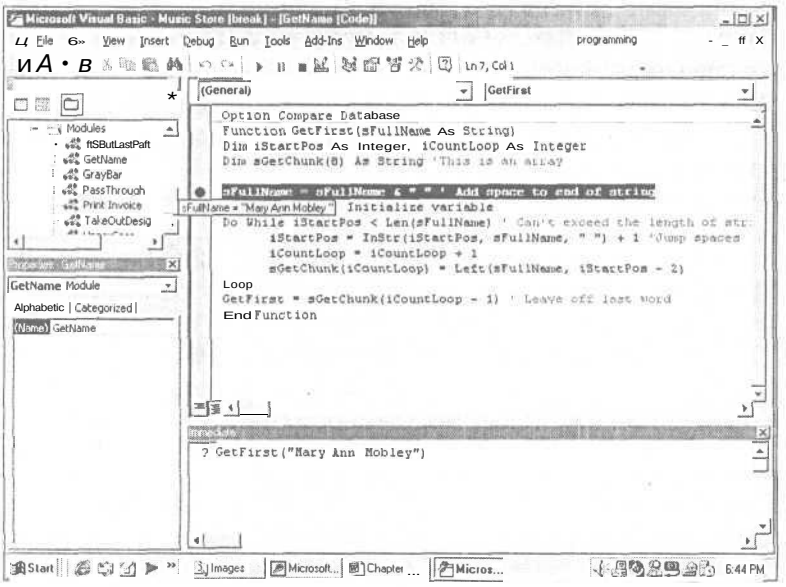


Рис. 8.5. Поместите курсор над переменной, чтобы наблюдать ее изменение во время выполнения вашей пользовательской функции

3. Нажмите F8 несколько раз, пока не окажетесь на строчке, следующей за выражением DoWhile.
4. Наведите курсор на переменную iStartPos. Ее значение должно быть равно 1. Нажмите F8, оставив курсор на том же месте. Теперь должно высветится 6.
5. Теперь переведите курсор на переменную iCountLoop и заметьте, что ее значение равно 0. Нажмите F8, посмотрите, что ее значение стало равно 1.
6. Наконец, переместите курсор на переменную sGetChunk и нажмите F8.

Заметьте, что переменной sGetChunk(iCountLoop) было присвоено строковое значение Mary. sGetChunk(iCountLoop) на самом деле - sGetChunk(1), потому что значение iCountLoop равно 1 в данной точке процедуры.

В следующей итерации цикла sGetChunk(iCountLoop) станет sGetChunk(2) и ему будет приписано значение Mary Ann.

В функции Left(sFullName, iStartPos - 2) делается отступ на две позиции от первой буквы (символа) второго слова (в данном случае «А» в слове Ann) к последнему символу предыдущего слова (в данном случае «у» в слове Mary); в результате функция left возвращает значение Mary.

Мы можете продолжить рассматривать переменные и как они изменяются с каждым новым циклом, если хотите, но это должно дать вам понимание того, что происходит в процедуре. Из этого маленького упражнения вы, кроме всего прочего, получили информацию о том, как находить ошибки в своих собственных процедурах.

Вы можете спросить: «Зачем добавлять пробел в изначальную строку?» Функция InStr используется для определения слов методом отыскания пробелов. Последнее слово не может быть определено без пробела на конце. Вы также можете спросить: «Каким образом sGetChunk(iCountLoop 1) отсеивает последнее слово?» Процедура как бы возвращается назад, к предыдущей итерации цикла, выбирая значение переменной массива до того, как последнее слово было добавлено. То есть, если было три слова, процедура выбирает два. Если было два слова - выбирает одно и т. д.

Очевидно, эта процедура все еще не совершенна. Что если парня зовут John Brown III? Функция вернет John Brown вместо John. Вы можете настроить процедуру, чтобы она исправляла различные аномалии, как эту, но в цели книги не входит дать вам решение на все случаи жизни; однако в ее цели входит дать вам средства, которые обеспечат вам быстрый старт на пути написания программ, решающих любые ваши задачи.

Если вы хотите проверить последнее слово в ФИО и посмотреть, является ли оно обозначением вроде "MD" или "III", первое, что вы должны сделать - это найти последнее слово. Функция GetFirst может быть переписана по-другому. Этим методом легко можно найти как фамилию, так и имя и отчество. Вы также можете использовать этот способ, который применяет отыскание последнего пробела в строке для выделения дописок к имени, таких, как "Jr." Следующий кусок кода может быть дополнен для нахождения последнего пробела в строке.

Для получения дополнительной информации о цикле For..Next см. раздел «Создание процедур» *в* гл. 10.

```
For i = Len(sFullName) To 1 Step -1
sGetChar = Mid(sFullName, i, 1)
Next i
```

На данный момент все, что вам нужно знать об этом коде, - это то, что он отступает в обратном направлении по одному символу за раз. Если вы можете делать такое, значит, все, что вам нужно, - это использовать выражение If для

проверки на пробелы. Как только положение последнего пробела в строке найдено, вы просто берете это число и употребляете его в качестве второго аргумента строковой функции left. Теперь у вас есть John Wilkes в John Wilkes Booth и John в John Booth. Вы можете переписать функцию GetFirst следующим образом:

```
Function GetFirstPart(sFullName As String)
Dim sGetChar As String

For i = Len(sFullName) To 1 Step -1 'в обратную сторону
sGetChar = Mid(sFullName, i, 1)
  If sGetChar = " " Then
    GetFirstPart = Left(sFullName, i)
    Exit For
  End If
Next i
End Function
```

Эта процедура делает то же самое, что и GetFirst, но она короче. Происходит выход из цикла, когда найден пробел после удаления последнего слова. Следующая функция удалит дописки к имени, такие, как "Jr." или "III":

```
Function RemoveDesig(sFullName As String)
Dim i As Integer, sGetChar As String
Dim sLast As String, sFirst As String

For i = Len(sFullName) To 1 Step -1
sGetChar = Mid(sFullName, i, 1)
  If sGetChar = " " Then
    sLast = Right(sFullName, Len(sFullName) - i)
    sFirst = Left(sFullName, i - 1)
    If sLast = "Jr." Or sLast = "Sr." Or sLast = "III" _
      Or sLast = "M.D." Or sLast = "R.N." Then
      RemoveDesig = sFirst
    Exit For
    Else
      RemoveDesig = sFullName
    Exit For
    End If
  End If
Next i
End Function
```

Вы можете вписать столько or (или), сколько необходимо для удаления всех приписок, которые вам не понравились. Были использованы некоторые из наиболее частых приписок. Вы можете добавить свои.

Если ваш список с «или» слишком велик, попробуйте привлечь способ обращения с этими «или» используя запрос. См. раздел «Using Query By Table Example For If Then Scenarios» в гл. 16.

В функцию GetFirstPart нужно теперь добавить строчку

```
sFullName = RemoveDesig(sFullName)
```

Теперь функция выглядит так:

```
Function GetFirstPart(sFullName As String)
Dim sGetChar As String

sFullName = RemoveDesig(sFullName)
For i = Len(sFullName) To 1 Step -1 'в обратную сторону
sGetChar = Mid(sFullName, i, 1)
  If sGetChar = " " Then
    GetFirstPart = Left(sFullName, i)
    Exit For
  End If
Next i
End Function
```

Теперь эти две процедуры объединены, потому что GetFirstPart вызывает RemoveDesig. Первое, что процедура делает, - это удаляет приписки в конце строки. Затем она находит первую часть из ФИО без приписок. Как сказано выше, этот метод не обязательно покрывает все возможные варианты имен. Что он действительно делает - это дает вам понимание того, как с ними обращаться.

Как и в случае с функцией GetFirst, вы можете протестировать эту процедуру, используя средства, находящиеся в окне модуля. Вы также можете вернуться к этим процедурам после получения дополнительных программистских навыков в следующей главе.

Объединение возможностей функций и запросов

Очевидно, вы хотите уметь больше, чем просто экспериментировать с функцией GetFirstPart в окне проверки. Следуйте следующим инструкциям, чтобы совместить указанную функцию с запросом.

1. В базе данных Music Store в разделе **Запросы** выберите **Создать запрос в конструкторе.** Откроется диалоговое окно **Добавление таблицы.** Дважды щелкните мышью на Employees и нажмите **Закрыть.**
2. Дважды щелкните на поле Employee таблицы Employees, затем щелкните правой кнопкой в строке **Поле** в пустом столбце справа от поля Employee. Выберите **Масштаб.**
3. Наберите Fname:GetFirstPart(Employee) и нажмите **ОК.** Скобки будут вставлены автоматически.
4. Дважды щелкните по правой разделительной черте текущего поля, чтобы увидеть все выражение (отступ по размеру).
5. Нажмите кнопку Запуск для выполнения запроса (рис. 8.6).
   Обратите внимание на Terry J. Turner, Jr. Функция преобразовала это имя в Terry J., как оно и предполагалось.
6. Закройте запрос без сохранения.

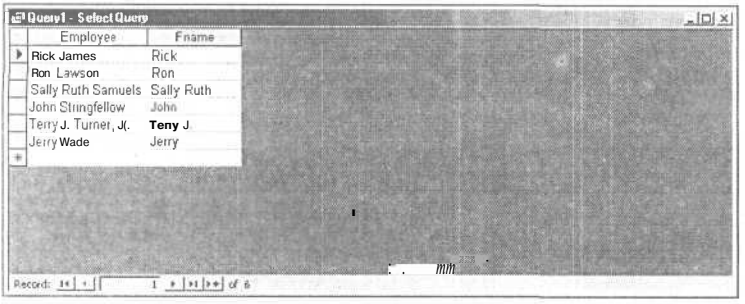


Рис. 8.6. Результат совмещения запроса и функции отображает возросшую функциональность

Вы написали свою первую программу. Вы узнали, как использовать окно проверки для тестирования ваших процедур. Вы представляете, что такое переменные и массивы, и вы поняли, как приятно видеть, что делают процедуры, наблюдая их в действии. Чем больше вы работаете с процедурами, тем проще они становятся. И чем проще они становятся, тем больше удовольствия вы получаете, работая с ними.

Что дальше?

В следующей главе вы узнаете в чем разница между макроязыком и VBA и как конвертировать макрос в VBA-код. Вы можете рассматривать конвертирование макросов как практическую помощь при изучении VBA. Вы также узнаете о различных способах выдачи сообщений пользователям.

9
Макросы, модули и сообщения

В гл. 7 вы узнали об объектах. В гл. 8 вы получили начальные навыки написания VBA-программ, используя функции. Эта глава объединяет VBA и объекты с целью показать, как они взаимодействуют. Повседневной задачей базы данных является открытие объектов базы данных, таких, как таблицы, формы и отчеты. Эта рутинная работа может быть автоматизирована благодаря программному коду.

Макросы также могут автоматизировать рутинные операции, такие, как открытие форм и печать отчетов, а создание их проще, чем написание программ. Поскольку вы обычно устанавливаете аргументы для макросов из списка возможных в окне «Макрос», вам не приходится запоминать сложный синтаксис. В этой главе вы узнаете, как использовать макрос для создания VBA-кода. Вы также узнаете, как эффективно использовать сообщения в Access. В частности, вы узнаете:

- в чем разница между макросами и модулями;
- когда использовать макросы, а когда модули;
- как использовать специальные макросы;
- как изучить VBA, конвертируя макросы в VBA-код;
- как использовать функцию MsgBox;
- как использовать функцию InputBox;
- как конвертировать макрос в VBA (на примере);
- как распечатать текущую запись используя макрос.

Visual Basic против макроязыка

В ранних версиях Access в качестве программной среды использовался Access Basic; однако в последние годы VBA заменил Access Basic, становясь стандартом для приложений Office, таких, как Access, Excel и PowerPoint. Такая унифицированность выгодна тем, что вам приходится изучать только один язык программирования для нескольких приложений. Однако помните, что вы должны сделать ссылки на библиотеки для тех приложений, которые должны быть объединены в программу.

И Visual Basic и макроязык - мощные средства, помогающие вам автоматизировать ваше приложение для Microsoft Access, но какое из них следует использовать? Как макросы, так и модули могут автоматизировать рутинные задачи. И те и другие могут обращаться к формам и отчетам и могут быть присоединены к их событиям.

Новички обычно находят написание макросов проще, чем программ на VBA. В макросе вы просто выбираете команду из списка и создаете последовательность действий, определенную пользователем. Эти действия выполняются последовательно, образуя программу. Однако вскоре после того, как вы начнете использовать макросы, их ограниченность станет очевидной.

Когда использовать VBA, а когда - макросы

Хотя макросы - мощное средство, существуют ситуации, в которых макрос просто ничего не сможет сделать. Предположим, вы хотите создавать объекты и манипулировать ими. В макросах не существует средств для эффективного выполнения таких задач; они также не могут управлять наборами данных по принципу «запись за записью». Вы также можете перемещать и находить отдельные записи из набора используя макрос, но вы не сможете управлять записями так, как вы могли бы сделать это в VBA, не вызывая функцию или запрос для осуществления этой работы. Поскольку модули компилируются, они выполняются быстрее, чем макросы, что делает их использование выгоднее в целях оптимизации. Используя VBA, вы можете также передавать аргументы, возвращать значения и использовать переменные в качестве аргументов.

Макросы полезны в простых задачах, таких, как открытие и закрытие форм, либо для маленьких приложений, не требующих сложной автоматизации. Вы также можете использовать макрос для создания приложения, в котором легко может разобраться (и модифицировать его) сторонний человек, незнакомый с программированием. Другой пример уместного использования макросов - создание прототипа программы, который в дальнейшем может быть расширен до полноценного приложения, использующего VBA. Хотя временные затраты на создание макросов обычно меньше, обратной стороной их использования являются уменьшение возможностей по сравнению с VBA. Поскольку отладочные средства для макросов ограниченны по сравнению со средствами для модулей, отладка и доработка макросов может оказаться сложнее.

Особые макросы Access

Два типа макросов используются для специальных целей -- это макрос AutoKeys и макрос AutoExec.

Если вы хотите произвести какое-либо действие при нажатии клавиши (или комбинации клавиш), макросы позволяют это сделать. Более того, вы можете привязать программный код к комбинации клавиш, используя макрокоманду RunCode. Можно создать один макрос для всех комбинаций клавиш, используемых в базе данных, но помните, что вы должны назвать его особым образом: AutoKeys, как это показано на рис. 9.1. Помните также, что нужно пользоваться столбцом «Имя макроса» (который вставляется нажатием иконки «Имена макросов» на панели инструментов) для размещения комбинаций клавиш.

Другой специальный тип макросов, называемый AutoExec, производит действие или серию действий, таких, как высвечивание меню или открытие формы, при открытии базы данных. После открытия базы данных Microsoft Access ищет макрос AutoExec и запускает его автоматически в случае обнаружения.

Visual Basic for Applications определенно преобладает над макросами, когда речь заходит об управлении ошибками. Вы можете использовать переменные и создавать циклические конструкции в VBA-программе. Вы можете выполнять инструкцию SQL из строки переменных VBA-программы. Вы можете использо-

вать VBA для работы с любыми типами коллекций объектов. Например, вы можете узнавать и изменять свойства объекта при помощи программного кода. Если вы хотите добавлять запись в таблицу, манипулировать элементами управления в формах и отчетах или создавать свои функции для настройки вышей базы данных, VBA - однозначно ваш выбор. Резюмируя, скажем, что VBA имеет больше возможностей, он более гибок, чем макроязык. И хотя вам придется потратить больше сил для обучения, оно того стоит.

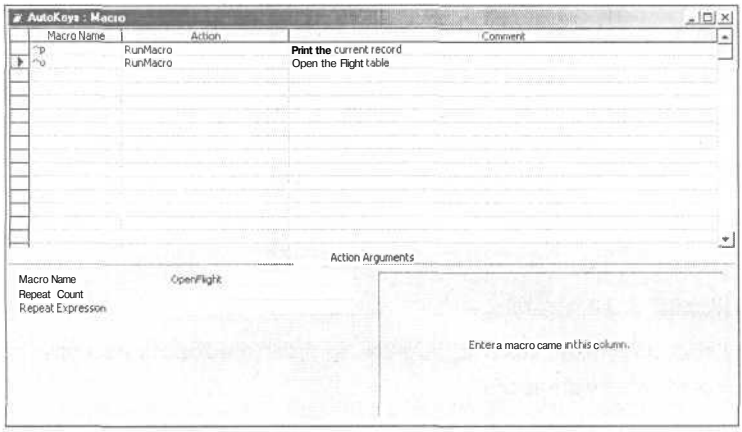


Рис. 9.1. Специальная группа макросов AutoKeys позволяет вам создать группу команд, выполняемых отдельно ото всех при нажатии клавиши или комбинации клавиш

Изучение VBA путем конвертирования макросов в VBA-код

Конвертирование макросов в VBA-программы производится очень просто. Это будет способствовать вашему росту в процессе обучения и поможет вам сделать первые шаги на почве программирования. Но таким способом вы можете дойти только до определенного уровня, поскольку существуют многие вещи, которые макросам просто не под силу. Тем не менее это хороший способ начать написание процедур.

Следующий пример демонстрирует вам, как создать макрос, открывающий форму. Далее вы узнаете, как конвертировать его в VBA-программу.

1. Откройте базу данных Convention из папки AccessByExample на вашем жестком диске.
2. В разделе **Объекты** выберите **Макросы** и нажмите **Создать** для открытия окна «Макрос», в котором вы можете создавать и редактировать макросы.
3. В первом ряду столбца команд[11], сверху окна, наберите ms и нажмите Tab. Заметьте, как появится команда MsgBox (рис. 9.2). (В русифицированном варианте команда называется Сообщение и выводится соответственно после набора со.– *Пер.*)

[11] Это окно называется «Макрокоманда» и отображает макрокоманды. - *Науч. ред.*

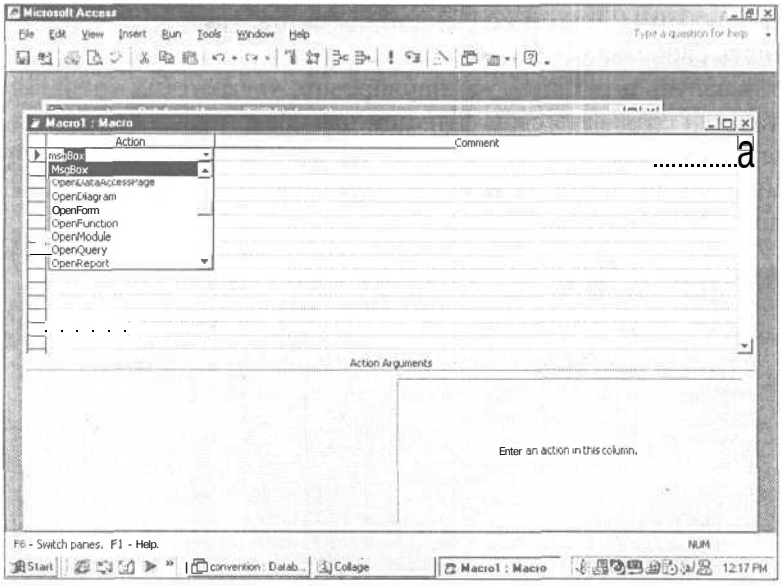


Рис. 9.2. Вместо того чтобы листать список команд, вы можете просто набрать несколько первых букв, и она допишется автоматически

4. Нажмите F6 и наберите Открытие таблицы Flight.
5. Щелкните на второй строчке столбца команд верхней секции окна «Макрос».
6. В меню «Окно» выберите «Слева направо». Скорректируйте размеры окон, чтобы видеть их оба в достаточной степени.
7. В окне «База данных» в разделе **Объекты** выберите **Таблицы.**
8. Перетащите мышью (наведите курсор, зажмите левую клавишу и тяните) таблицу Flight во вторую строку окна **Макрос.** Заметьте, что в нижней секции все настроилось правильно (рис. 9.3).
9. Выберите меню **Окно, Каскадом,** чтобы изменить размер окна **База данных** таким образом, чтобы это было удобно после закрытия окна **Макрос.**
10. Нажмите **Сохранить** (иконка дискеты). Сохраните как macOpenFlight и закройте окно **Макрос.**
11. В разделе **Объекты** выберите **Формы;** выделите Client и нажмите **Конструктор.**
12. Если панель элементов не открыта, откройте ее, кликнув на соответствующей иконке.
13. Убедитесь, что кнопка **Мастер** не нажата, и выберите элемент **Кнопка.** Курсор приобретет форму плюса с прямоугольником.

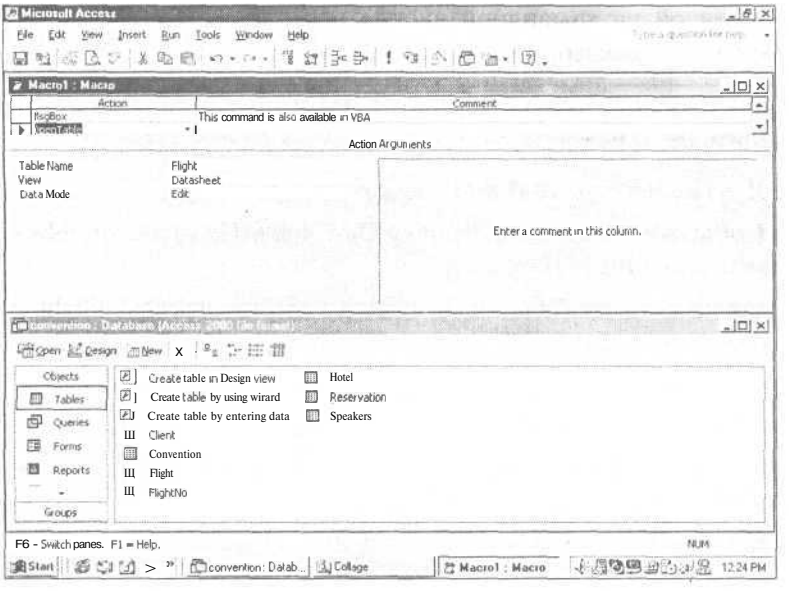


Рис. 9.3. Вы можете перетащить мышью вашу таблицу для создания команды OpenTable (Открыть таблицу)

ЗАМЕЧАНИЕ. Кнопка, изображающая волшебную палочку, находящаяся обычно в верхней части панели элементов, в зависимости от того, как вы настроили вашу панель инструментов, называется **Мастер.** Когда она активна, мастер помогает вам в процессе создания элемента управления, задавая различные вопросы. После завершения работы мастера ваш элемент управления настроен в соответствии с предоставленными вами ответами. Мастер поможет вам с полями со списками, простыми списками, простыми кнопками, подчиненными формами и т. д. Если вы предпочитаете создавать элемент управления вручную (как в данном случае), просто отключите кнопку **Мастер.**

14. Справа от надписи Ccity нарисуйте прямоугольник размерами примерно с надпись Ccity.

ЗАМЕЧАНИЕ. Командная кнопка (обычная нажимающаяся кнопка. - *Пер.*) - одна из тех, которые позволяют мгновенное создание. Вместо того чтобы рисовать прямоугольник, вы можете просто щелкнуть мышью в том месте, где вы хотите, чтобы ваша кнопка располагалась, и командная кнопка стандартных размеров будет размещена на вашей форме. Другие элементы управления, такие, как переключатели, также обладают такой способностью.

15. Наберите текст Открыть таблицу Flight на кнопке.

ЗАМЕЧАНИЕ. Если текст, набранный вами, не помещается на кнопке, можно воспользоваться опцией меню **Формат, Размер, По размеру данных. То** же самое можно сделать, щелкнув правой клавишей на кнопке и выбрав **Размер,** «По размеру данных».

16. Щелкните правой кнопкой на командной кнопке и выберите **Свойства** из появившегося меню быстрого вызова, появится список свойств.

ЗАМЕЧАНИЕ. Можно миновать меню быстрого вызова: щелкните мышью на кнопке, чтобы выделить ее, и нажмите F4; сразу появится список свойств.

17. Выберите вкладку **Другие** и в строке **Имя** наберите `cmdOpenFlight`.
18. Выберите вкладку **События** и выделите строчку On Click («Нажатие кнопки» - в русифицированном варианте. – *Пер.*).
19. Щелкните на ниспадающем списке события On Click и выберите macOpenFlight.
20. Закройте список свойств. Сохраните и закройте форму.
21. Проверьте работоспособность макроса: откройте предыдущую форму и щелкните по кнопке, которую вы только что создали.
22. Вы должны увидеть сообщение с надписью «Открытие таблицы Flight» (рис. 9.4). Нажмите OK - откроется таблица Flight.

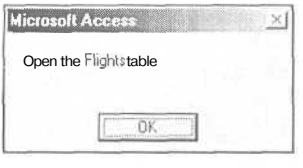


Рис. 9.4. При проверке вашего макроса перед открытием таблицы появляется сообщение

Функция MsgBox

Знаете вы или нет, но вы воспользовались одной из многих встроенных функций, доступных в Access. Функция MsgBox является полезным средством, позволяющим посылать сообщения пользователю. Другая подобная функция - InputBox, предоставляющая пользователю дополнительную возможность ввода информации. Когда MsgBox запущена, пользователь не может щелкнуть мышью ни в каком месте приложения Access, кроме как на кнопку OK. Например, пользователь не может выбрать что-либо из меню приложения, пока «висит» сообщение. Однако пользователь по-прежнему может воспользоваться операционной системой, в частности кнопкой Пуск (Start), открывающей доступ к другим приложениям.

Синтаксис для использования MsgBox (выражения в скобках обозначают необязательные аргументы):

```
MsgBox(prompt,[ buttons],[ title],[ helpfile, context])
```

Табл. 9.1 демонстрирует требования для каждого аргумента. Вы можете использовать функцию MsgBox с текстовой строкой в виде аргумента, чтобы послать простое сообщение, либо вы можете использовать ее, чтобы посмотреть, какое значение принимает какая-либо переменная в определенный момент времени.

Таблица 9.1. Аргументы MsgBox

Аргумент	Обязательность	Описание	Пример
Prompt	Обязательно	Строковое выражение	"Open Form"
Buttons	Необязательно	Числовое выражение	1
Title	»	Строковое выражение	"Warning"
Helpfile	»	»	"acmain9.chm"
Context	»	Числовое выражение	5003042

Следующая процедура позволяет пользователю воспользоваться подсказкой Access.

```
Private Sub cmdMsgBox_Click()
Dim Path As String, Button As Integer
Dim Title As String, HelpFile As String

Path = "C:\Program Files\Microsoft Office\Office\1033\"
Button = 64
Title = "System Message"
HelpFile = "acmain9.chm"
MsgBox "Click help to search Access help", _
Button + vbMsgBoxHelpButton, Title, Path & HelpFile, 0
End Sub
```

Эта подпрограмма использует все возможные аргументы MsgBox. Заметьте, что используется vbMsgboxHelpButton для создания кнопки, ответственной за вывод помощи. Аргумент Context определяет раздел в файле помощи, указанного в аргументе HelpFile. Таким образом, эти два аргумента взаимосвязаны.

Были созданы вспомогательные переменные для облегчения вписывания аргументов и для улучшения удобочитаемости программы. Путь присоединен к имени файла при помощи оператора амперсанда (&). Если вы хотите обеспечить помощь по ошибке «невозможно найти таблицу», измените путь на c:\windows\system (или другой, в зависимости от того, куда установлена ваша система), helpfile на Jeterr40.chm и context (0) на 5003024.

В табл. 9.2 показаны возможные значения аргумента *buttons*[12].

Таблица 9.2. Параметры Button

Тип кнопки (константа)	Значение	Результат
VbOKOnly	0	Отображает только кнопку OK
VbOKCancel	1	Отображает кнопки OK и Отмена
VbAbortRetrylgnore	2	Отображает кнопки Отмена, Повторить попытку, Продолжить (Abort, Retry, и Ignore)
VbYesNoCancel	3	Отображает кнопки Да, Нет и Отмена

[12] Button - кнопка, поэтому это макрокоманда, которая создает кнопку. - *Науч. ред.*

Тип кнопки (константа)	Значение	Результат
VbYesNo	4	Отображает кнопки Да и Нет
VbRetryCancel	5	Отображает кнопки Повторить попытку и Отмена (Retry и Cancel)
Тип иконки		
VbCritical	16	Отображает иконку Критическое сообщение
VbQuestion	32	Отображает иконку Вопрос
VbExclamation	48	Отображает иконку Предупреждение
VbInformation	64	Отображает иконку Информация
Кнопка, выделенная по умолчанию		
vbDefaultButton1	0	выделена первая кнопка
vbDefaultButton2	256	выделена вторая кнопка
vbDefaultButton3	512	выделена третья кнопка
vbDefaultButton4	768	выделена четвертая кнопка
Модальность		
VbApplicationModal	0	Пользователь должен ответить на сообщение, прежде чем продолжить работу в данном приложении
VbSystemModal	4096	Все приложения приостанавливаются до тех пор, пока пользователь не ответит на сообщение
Особые возможности		
VbMsgBoxHelpButton	16384	Добавляет кнопку Справка
VbMsgBoxSetForeground	65536	Делает окно сообщения активным окном
VbMsgBoxRight	524288	Текст выравнивается по правому краю
VbMsgBoxRtlReading	1048576	Текст печатается справа налево для иврита и арабского языков

Функция MsgBox возвращает значение в зависимости от сделанного выбора. Это очень удобно для программирования, как вы убедитесь дальше. Например, если вы хотите продолжать вашу программу в зависимости от ответа пользователя, вы можете выдать сообщение при помощи следующего кода:

```
Msgbox("вы хотите продолжить?", 4 + 32 + 256)
```

Если сложить вместе все три числа в этой функции, получится то же самое сообщение. Предположив, что вам предпочтительна именно такая комбинация констант, вы можете сложить эти числа вместе, записать полученную сумму и что она обозначает и в дальнейшем использовать ее снова и снова. Набирать од-

но число быстрее, чем три константы. Посмотрите, как 4, 32 и 256 сложены в следующем примере:

```
Msgbox("вы хотите продолжить?", 292)
```

Даже если вы уберете вопросительный знак в конце предложения, по-прежнему очевидно, что вы задаете вопрос. Вопросительный знак дописывается автоматически, поскольку использован аргумент 32. Взгляните на рис. 9.5. Комбинирование трех чисел приводит к трем изменениям в окне сообщения. Вы могли бы также подставить эквивалентные константы для каждой опции, чтобы сделать вашу программу удобной для понимания. Если вы набираете ваш код в окне **Модуль,** VBA предоставляет вам на выбор список констант функции MsgBox. Если бы вы использовали встроенные константы вместо чисел, программный код выглядел бы примерно так:

```
Msgbox("вы хотите продолжить?",vbYesNo + vbQuestion + vbDefaultButton2)
```

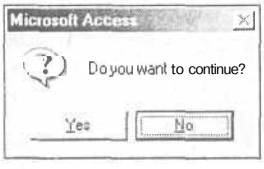


Рис. 9.5. Окно сообщения с вопросом может быть отображено путем использования функции MsgBox с соответствующими аргументами

ЗАМЕЧАНИЕ. В Access имеются константы (все они начинаются с букв «vb») для аргументов функции Msgbox, делающие вашу программу легче для понимания. Вы можете использовать как числа, так и константы либо и те и другие вперемешку.

Испробуйте первый вариант в окне проверки, нажав Ctrl+G и набрав ? MsgBox("вы хотите продолжить?", 4 + 32 + 256). Если вы выберете Да, будет возвращено 6. Если Нет - 7. Вы можете использовать эту методику в своей программе, что иллюстрирует следующий пример:

```
If MsgBox("вы хотите продолжить?", 4 + 32 + 256) - 6 Then
MsgBox ("Да")
Else
MsgBox ("Нет")
End If
```

Конечно, этот пример служит только лишь для проверки процедуры. После того как вы убедились, что она работает, вы можете вписать любой код вместо MsgBox (Да) и MsgBox (Нет). В табл. 9.3 показаны возвращаемые значения для каждого типа кнопок.

Таблица 9.3. Возвращаемые значения

Константа	Значение	Описание
vbOK	1	OK
vbCancel	2	Отмена (Cancel)
vbAbort	3	Отмена (Abort)
vbRetry	4	Повторить попытку (Retry)
vbIgnore	5	Продолжить (Ignore)
vbYes	6	Да (Yes)
vbNo	7	Нет (No)

Функция InputBox

Если не считать поля для ввода, функция InputBox на первый взгляд выглядит почти так же, как MsgBox. Однако она позволяет запрашивать у пользователя информацию, которая может быть записана в переменную. Синтаксис InputBox похож на MsgBox:

```
InputBox ([prompt], [title], [default], [xpos], [ypos], [ helpfile, context])
```

Аргументы, общие для этих двух функций (prompt, title, helpfile и context), работают аналогично. Аргумент default - необязательная строка или выражение, появляющиеся в текстовом поле при его появлении. Необязательные аргументы xpos и ypos задают координаты окна сообщения на экране. Следующая строчка кода демонстрирует типичное использование этой функции:

```
SSN = InputBox("Пожалуйста, введите номер вашей социальной страховки",,"000-00-0000")
```

Заметьте, что ответ пользователя записывается в переменную SSN. Эта возможность делает функцию InputBox очень полезной. Также заметьте, что дополнительная запятая между строками prompt и default служит для пропуска title. Изображение экрана после выполнения данного кода показано на рис. 9.6.

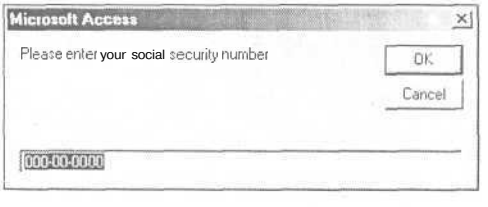


Рис. 9.6. Вы можете использовать функцию InputBox, чтобы приписать значение переменной

Конвертирование макроса

Теперь вы знаете немного о создании макросов, генерирующих сообщения. Теперь давайте прольем свет на конвертирование макросов в VBA-код. Первая хорошая новость: вы можете конвертировать любой имеющийся макрос в программу на VBA. Дополнительное преимущество, получаемое от конвертирования, - это то, что автоматически генерируется процедура обработки ошибок. Об этом вы подробнее узнаете, начав изучение процедур обработки ошибок.

Чтобы узнать больше об управлении ошибками, см. раздел «Determining which Errors Occur» в гл. 11.

1. В окне «База данных» в разделе **Макросы** выберите макрос, который должен быть конвертирован.
2. Из меню выберите **Сервис,** «Макрос», «Преобразовать макросы».
3. Убедитесь, что флажки «Добавить программу обработки ошибок» и «Добавить примечания макросов» установлены.

Вот и все! Заметьте, что новый модуль появился в разделе **Модули,** как показано на рис. 9.7.

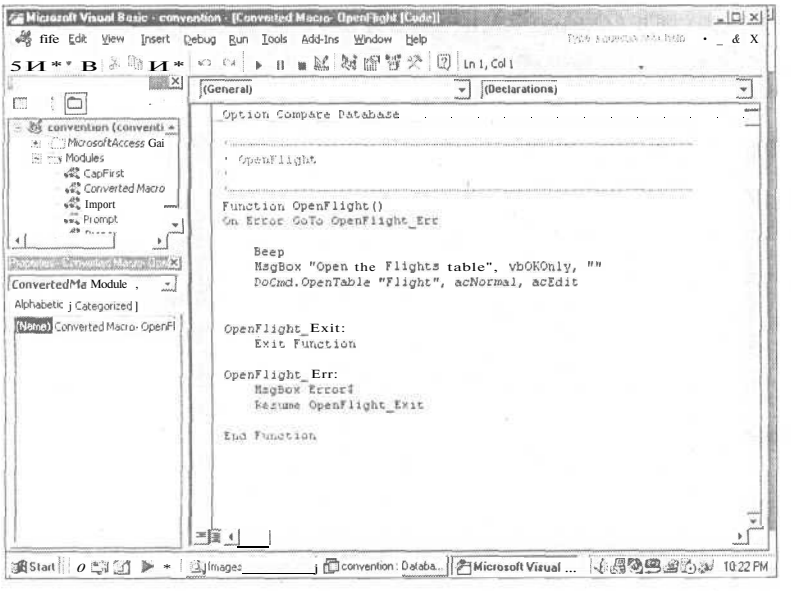


Рис. 9.7. Когда Access конвертирует макрос в процедуру VBA, добавляется программа обработки ошибок

Распечатка текущей записи при помощи макроса

Команда RunCommand (Выполнить команду в русифицированном варианте. - *Пер.)* может запускать встроенные команды из меню Access, его панели инструментов или меню быстрого вызова. Эта команда заменила команду DoMenuItem

из Access 97. Для нее требуется один обязательный аргумент, который можно выбрать из ниспадающего списка.

Вы можете использовать макрос для распечатки текущей записи в форме. Этот макрос может быть просто трансформирован в VBA-код. Вы можете использовать команду RunCommand_(выполнитьКоманду) для выбора текущей записи и затем распечатать ее.

Следующие шаги помогут вам создать такой макрос.

1. Находясь в разделе **Макросы** в окне «База данных», нажмите **Создать** для открытия нового окна «Макрос».
2. В столбце для команд в первой строке щелкните мышью на стрелке ниспадающего списка и выберите RunCommand (выполнитьКоманду). Вы можете набрать несколько первых букв вручную для ускорения поиска.
3. Нажмите F6, чтобы перейти в секцию «Аргументы команды» для RunCommand (ВыполнитьКоманду). Из ниспадающего списка аргумента «Команда» выберите SelectRecord.
4. В следующей строке столбца макрокоманд выберите команду Printout (Печать) из ниспадающего списка. Опять-таки, набрав р (п в русифицированной версии. – *Пер.*), вы быстрее отыщите команду.
5. Нажмите F6 для перехода в секцию «Аргументы команды» для макрокоманды Printout (Печать). В аргументе Print Range (Распечатать), выберите Selection (Фрагмент), как показано на рис. 9.8.

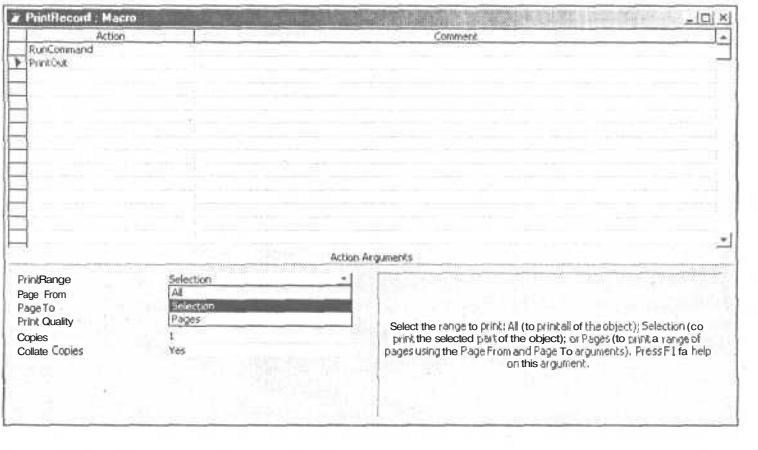


Рис. 9.8. При помощи всего лишь двух макрокоманд можно распечатать текущую запись на форме

6. Нажмите кнопку **Сохранить** на панели инструментов и назовите макрос macPrintRecord. Закройте его.
7. Откройте форму Client в виде конструктора. Создайте командную кнопку под кнопкой **Открытие таблицы Flight** и назовите ее Print Record.

8. Щелкните на нее и нажмите F4 чтобы открыть список ее свойств.
9. Переключитесь на вкладку **События** и выберите **Нажатие кнопки.** Из ниспадающего списка события **Нажатие кнопки** выберите macPrintRecord.
10. Закройте список свойств. Сохраните и закройте форму. Протестируйте макрос, открыв заново форму и нажав кнопку, которую вы только что создали.
11. Закройте форму.

Вы, возможно, вспомните, что в гл. 5 вы уже научились распечатывать текущую запись из формы; однако этот метод уникален в свете следующих причин. Во-первых, он не использует отчет для печати, как это было в гл. 5. Вместо этого он распечатывает текущую запись из формы, используя то оформление формы, какое вы создали. Во-вторых, он использует команду Printout вместо команды OpenReport для печати записи. В-третьих, он представляет вам команды RunCommand и Printout, открывающие большие возможности для вашей базы данных.

Если вы сохраните макрос macPrintRecord как модуль, ваш код будет выглядеть следующим образом:

```
Function macPrintRecord()
On Error GoTo PrintRecord_Err

   DoCmd.RunCommand   acCmdSelectRecord
   DoCmd.PrintOut acSelection,,,  acHigh,  1,  True


PrintRecord_Exit:
   Exit Function

PrintRecord_Err:
   MsgBox  Error$
   Resume PrintRecord_Exit
End Function
```

Макросы - это просто и здорово. Несмотря на ограничения, у них есть своя область применения. Возможно, их главное предназначение - помочь вам научиться писать программный код.

Что дальше?

В следующей главе вы узнаете, как настроить ваше приложение, используя написанные вами программы. VBA может предоставить вам такие гибкие средства разработки, что, возможно, единственным ограничением будет ваше воображение. Функции, подпрограммы, область действия, управление ходом программы, циклы - вот лишь некоторые темы из тех, что будут освещены. Если вы не можете найти встроенную функцию Access, удовлетворяющую вашим нуждам, вы можете создать свою.

10
Использование процедур для настройки вашей базы данных

В предыдущей главе вы узнали не только о макросах и модулях, но и о том, как посылать сообщения пользователю. Эта глава поведет вас дальше в область программирования, обучая вас, как использовать различные операторы управления ходом программы, являющихся частью VBA.

Вы уже имеете представление о функциях и их возможностях. Эта глава дает вам возможность продолжить изучение функций при помощи практически полезных примеров их применения. В примерах в конце главы используется много встроенных функций Access. В частности, вы узнаете:

- как использовать встроенные строковые функции Access;
- в чем разница между функциями и подпрограммами;
- что такое область действия и как ее использовать;
- как создавать процедуры;
- как использовать программные циклы;
- как использовать конструкции условий;
- как использовать функции для настройки вашей базы данных;
- как использовать функции для преодоления ограничений подстановочных знаков в запросах.

Изучение процедур

Одно из преимуществ базы данных Access состоит в том, что вы можете приспосабливать приложения для ваших нужд. Подпрограммы и функции могут обеспечить гибкость, позволяющую вам делать практически все, что пожелаете в смысле настройки и оптимизации вашей базы данных для получения большей эффективности. Отдача от повышенной продуктивности более чем покрывает то время, которое вы потратите на изучение того, как правильно разрабатывать и воплощать автоматизированное приложение.

Вместо того чтобы платить программисту за написание приложения с нуля, вы можете использовать модель базы данных Access как основу для вашей программы. Самой распространенной причиной того, что многие так не делают, - робость. Они боятся, что у них уйдет слишком много времени на выработку навыков, необходимых для выполнения работы.

Смотря на экраны подсказок и усовершенствования, созданные для удобства пользователя в последние годы, становится все труднее привести убедительный аргумент в пользу того, что Access слишком сложен. Но существует множество барьеров, связанных с обучением, которые мешают воспринимать все части как целое. В этой главе поставлена задача попытаться разрушить эти барьеры, созданные справочными системами Access и руководствами из Интернета. Это ни

в коем случае не умаляет значения этих методов; просто существуют некоторые важные вопросы, на которые трудно найти ответы.

Барьеры обучения обычно всплывают при возникновении какого-либо затруднения. Когда вы точно знаете, что вам нужно, но не знаете, как реализовать это в Access. Например, предположим, что вы хотите сделать заглавной первую букву в каждом слове в поле, притом оставив остальные буквы строчными. Попробуйте поискать встроенную функцию Access, которая может это осуществить. Такой функции не существует, но на данном этапе вы этого не знаете. Даже если бы она существовала, не зная, какие ключевые слова использовать для поиска, вы вряд ли ее найдете. Вместо того, чтобы часами искать по окнам подсказок и Web-страницам встроенную возможность Access, которая, возможно, и не существует, вы просто напишете вашу собственную программу, выполняющую необходимые операции.

Все, что вам нужно сделать, - это разработать пользовательскую функцию, позволяющую вам выполнить вашу задачу. Это означает, что вы можете настроить вашу базу данных, дополнив ее вашими собственными процедурами, которые в Access либо забыли включить, либо не посчитали их достаточно важными. Даже если существует утилита Access, делающая в точности то, что вы хотите, найти ее не всегда просто. Вы узнаете, как писать пользовательские процедуры, ниже в этой главе.

Когда вы осознаете возможности, которые становятся доступными посредством использования программного кода, вы увидите, что практически не существует ограничений на то, что вы сможете делать. Встроенные строковые функции Access помогут вам на этом пути. Две встроенные строковые функции, используемые чаще всего, - Mid и Instr. Строковая функция Mid имеет три аргумента, как показано в следующей строке:

```
Mid(string, start, [length])
```

Первый аргумент - это строка, которая может быть представлена переменной, константой или выражением. Второй аргумент (start) - это начальная точка строки, которую вы хотите получить. Третий аргумент (length) - количество символов, которые вы хотите получить. Этот аргумент является необязательным. Если вы его не впишите, вы просто получите остаток строки от начальной точки и до конца. В табл. 10.1 описаны аргументы для строковой функции Mid.

Таблица 10.1. Аргументы функции Mid

Аргумент	Обязательность	Тип данных	Описание
String	Обязателен	String	Строковое выражение, из которого возвращаются символы
Start	»	Long	Позиция первого символа из необходимого куска в строке. Если start больше, чем количество символов в строке, Mid возвращает пустую строку ("")

Аргумент	Обязательность	Тип данных	Описание
Length	Необязателен	Variant(long)	Количество символов, которые нужно вернуть. Если не указано или если в тексте меньше символов, чем length (включая символ в start), возвращаются все символы от начальной точки до конца строки

Функция InStr имеет четыре аргумента, но часто используется без первого и последнего аргумента. Первый аргумент - начальная позиция для вашего поиска. Второй аргумент - строка, в которой вы хотите искать. Третий аргумент - строка, которую вы хотите искать. И последний аргумент - способ сравнения строк. Эта функция возвращает численную позицию искомой строки в исходной строке. Синтаксис выглядит следующим образом:

```
InStr([start], string being searched, string searched for,[comparison type])
```

Квадратные скобки обозначают необязательные аргументы. В табл. 10.2 приведено более подробное описание, чем в данном образце.

Таблица 10.2. Аргументы функции InStr

Аргумент	Обязательность	Тип данных	Описание
Начальная точка	Необязателен	Численное выражение	Устанавливает начальную точку для поиска. Если не указана, поиск ведется с первого символа
Строка, в которой искать	Обязателен	Строковое выражение	Может быть строкой, полем или выражением (которое может включать в себя другую функцию). Поиск производится в пределах выражения, введенного здесь
Строка, которую искать	»	То же	Строка или выражение, которые нужно найти
Сравнение	Необязателен	Константа или значение	Необязателен. Определяет способ сравнения строк

В табл. 10.3 описаны другие полезные строковые функции.

Таблица 10.3. Другие полезные строковые функции с примерами

Функция	Аргумент 1	Аргумент 2	Результат	Пример	Дает
Lease	Строка или выражение		Переводит в нижний регистр	Lcase("KID")	kid

Функция	Аргумент 1	Аргумент 2	Результат	Пример	Дает
Left	»	Количество символов слева, которое нужно включить в результат	Возвращает x символов слева из строки	Left ("Bobby",3)	Bob
Len	»		Возвращает количество символов в строке	Len ("Karen")	5
Ltrim	»		Удаляет пробелы слева в строке	Ltrim (" Street")	Street
Right	»	Количество символов справа, которое нужно включить в результат	Возвращает x символов справа из строки	Right ("myway",3)	way
Rtrim	»		Удаляет пробелы справа в строке	Rtrim ("Street ")	Street
Str	Численное значение или выражение		Переводит численное значение в строку	Str(224)	"224"
Trim	Строка или выражение		Удаляет пробелы с обеих сторон в строке	Trim (" Street ")	Street
Ucase	То же		Переводит в верхний регистр	Ucase ("cia")	"CIA"
Val	»		Переводит строковое значение в численное	Val("1618")	1618

Функциииподпрограммы

Вы уже узнали некоторые отличия между функциями и подпрограммами. Функции возвращают значения; процедуры ничего не возвращают. Функции могут быть вызваны практически из любого места в Access, а подпрограммы, поскольку они не используются в выражениях, вы обычно вызываете при событиях (формы или отчета), в функции или в другой процедуре. Например, по-

скольку подпрограммы не возвращают значения, они не могут быть использованы в правой части выражения так, как вы используете любую встроенную функцию, например Left или Len. По этой же причине вы не можете вызвать процедуру, приравняв переменной ее значение. Кроме того, если вы попытаетесь использовать подпрограмму в качестве аргумента в том месте, где вы обычно используете функцию, например в запросе, вы получите сообщение «Неизвестная функция (название_процедуры) в выражении» (в английском варианте - «Undefined function (название_процедуры) in expression». – *Пер.*). С другой стороны, функции можно вызывать из запросов, элементов управления, макросов, форм, отчетов или из других процедур, потому что вы можете использовать выражения во всех этих объектах. Если явно не оговорено обратное, как подпрограммы (Sub), так и функции (Function) являются глобальными (общедоступными) по умолчанию.

Чтобы больше узнать о глобальных процедурах, см. «Что такое область действия (scope)?» дальше в этой главе.

Чтобы больше узнать о подпрограммах, см. раздел «How to Use DAO and ADO to Manipulate Data» в гл. 12.

Чтобы узнать больше о функциях, см. раздел «Using Functions with Queries, Revisited» в гл. 16.

Как подпрограммы, так и функции могут быть вызваны внутри других подпрограмм и функций. Это делает их *модульными*, что означает, что они могут быть использованы снова и снова, делая программный код проще для отладки, уменьшая общий объем написанного кода. Вместо того чтобы переписывать одну и ту же процедуру снова и снова каждый раз, когда она вам понадобится, вы просто вызываете ее.

Комбинирование функций и запросов может оказаться чрезвычайно производительным. Предположим, вы создали функцию, которая сэкономила ваше время, автоматически сделав первую букву в строчке заглавной. А теперь представьте, сколько времени вы могли бы сэкономить, если бы применили эту функцию к таблице, состоящей из тысяч записей! Например, функция Proper, представленная в этой главе, делает заглавным первый символ для каждого слова в поле. Запрос, использующий эту функцию, сэкономит вам время на каждой таблице в вашей базе данных.

Подпрограммы могут прекрасно справляться с манипуляцией данными и повседневными задачами в формах и отчетах. По нажатии одной кнопки сложные задачи могут быть выполнены моментально. Если этого недостаточно, в подпрограмме можно вызывать как функции, так и макросы. Поскольку к исходным таблицам и другим подобным объектам для форм и отчетов можно легко получить доступ из подпрограмм, привязанных к их (форм и отчетов. - *Пер.)* событиям, преимущества таких подпрограмм становятся очевидными.

Что такое область действия (scope)?

Областью действия (scope) называют диапазон, область, степень или сферу доступности переменной, константы или процедуры для других процедур. Переменная, константа или процедура могут быть объявлены как *открытая* (public, глобальная, публичная, общественная) или *частная* (private). Если они объявлены как открытые, то они являются доступными для всех процедур во всех модулях во всех приложениях, если только не задействована опция «Частная для модуля» (Private Module); в этом случае они доступны только внутри проекта, в котором они находятся. Все, что объявлено как «частное», доступно только в пределах модуля, в котором они объявлены.

ЗАМЕЧАНИЕ. Всякая переменная, объявленная как частная в разделе объявлений модуля, доступна любой процедуре данного модуля. Если переменной нет в разделе объявлений (другими словами, если она объявлена в процедуре), она доступна только в самой процедуре. Раздел объявлений расположен сверху в стандартном модуле, перед всеми процедурами. Для пущей уверенности выберите declarations (объявления) из ниспадающего списка в правом верхнем углу окна модуля, чтобы попасть туда. Выражение Dim («В данном месте» - *Пер.*) функционально не отличается от выражения Private, но ключевое слово Private делает запись более очевидной.

Помните, что по умолчанию все процедуры и функции объявляются как глобальные, кроме процедур обработки событий. Ключевое слово Private автоматически вставляется перед объявлением процедуры, когда VBA создает процедуру обработки событий. В любой другой ситуации, если вы явно не укажете, что ваша процедура должна быть частной, она становится глобальной. Например, если вы хотите, чтобы ваша функция была доступна в ваших запросах, не объявляйте ее как частную.

Если переменная объявлена как статическая, она находится в памяти все время, пока запущено приложение. Напротив, динамические переменные заново инициализируются каждый раз при запуске процедуры. Как вам это может пригодиться? Если вы хотите узнать, сколько раз использовался объект, такой, как список с полем или командная кнопка, вы можете объявить переменную-счетчик, которая будет увеличиваться каждый раз, когда запускается программа, например:

```
Counter = Counter + 1
```

Единственная проблема состоит в том, что каждый раз при обработке объекта эта переменная будет сбрасываться и инициализироваться как единица. Если же она была объявлена как статическая, она будет «накручиваться» все время, пока открыта база данных, таким образом вы получите правильное число.

Создание процедур

вы уже создавали процедуры, но давайте поближе рассмотрим этот процесс. Откройте любую базу данных Access. В разделе **Объекты** выберите **Модули** и нажмите **Создать.** Выберите меню **Вставка, Процедура.** Вы должны увидеть окно **Добавление процедуры,** в которое вы введете имя процедуры (рис. 10.1). Вы также можете выбрать ее тип и область действия или назначить все локальные переменные статическими. Вы можете не выбирать опцию вставки процедуры, а также просто создать вашу функцию или подпрограмму вручную, этот способ сэкономит вам немножко времени.

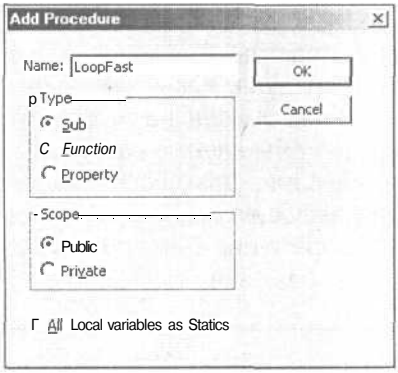


Рис. 10.1. Вы можете использовать диалоговое окно Add Procedure для установки настроек ваших процедур

Для типа процедуры вам предлагают на выбор **Подпрограмму** (Sub), **Функцию** (Function) или **Процедуру-свойство** (Property). Вы должны быть уже знакомы с подпрограммами и функциями на данный момент, но как быть с процедурами-свойствами? Процедура-свойство позволяет вам добавлять и манипулировать определенными свойствами. После создания она становится свойством модуля, содержащего эту процедуру.

Visual Basic for Applications (VBA), являющийся частью Visual Basic, поставляется вам вместе с программой Access. Когда вы нажимаете **Создать,** вы попадаете в окно Visual Basic, которое открывается при использовании Visual Basic Editor (Редактор Visual Basic), программы, позволяющей создавать и редактировать код VBA (рис. 10.2).

Если вы создаете функцию, вы используете стандартный модуль, как это сделали только что, но, если вы хотите создать подпрограмму, прикрепленную к событию в форме или отчете, вы должны использовать модуль классов (class module). На самом деле все процедуры обработки событий являются подпрограммами, собранными в модулях классов. В окне **Программа** для формы отчета список объектов в левом верхнем углу в списке отображается элемент управления, который вы используете в данный момент; список процедур в правом верхнем углу отображает имя процедуры обработки событий, над которой вы в данный момент работаете.

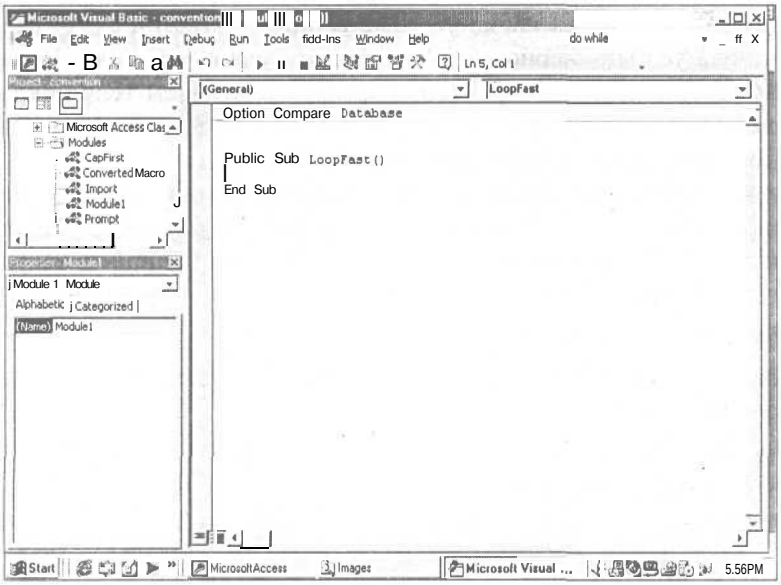


Рис. 10.2. Редактор Visual Basic позволяет вам создавать программный код в окне стандартного модуля

В окне **Проект** слева вы можете выбрать любую процедуру базы данных, дважды щелкнув на ней мышью. Вначале показываются все модули, прикрепленные к формам или отчетам. Далее идут стандартные модули. Это означает, что вам не нужно выходить из процедуры в стандартном модуле, чтобы изменить процедуру в модуле класса, и наоборот. Хотя списки одинаковы, вне зависимости от того, находитесь ли вы в модуле классов или в стандартном модуле, в модуле классов обычно находится намного больше пунктов.

Несколько способов создания цикла

Цикл - это повторяющаяся программная конструкция, позволяющая выражению или группе выражений обрабатываться снова и снова, пока выполнено условие. Следовательно, циклы предоставляют замечательный способ делать круги по программе. Если вы изучаете строку, вы можете делать циклы по одной букве или по одному слову за раз. Если вы сделали счетчик, в каждом цикле к нему можно добавлять различные значения. Если вы ищете объект в коллекции объектов, цикл - прекрасный механизм для перебора объектов из списка.

Цикл Do..Loop

выражение Do. . Loop (дословно «делать..цикл». – *Пер.*)выполняет программный код, находящийся внутри Do. . Loop либо пока условие остается истинным, либо пока условие не станет истинным (т. е. пока оно ложно. - *Пер.*), в зависимости от того, как построено выражение. Существует два способа использова-

ния ключевых слов While (дословно «пока условие верно». - *Пер.)* и Until (дословно «до тех пор, пока условие верно». - *Пер.)* для проверки условия в выражении Do. . Loop. Вы можете проверять условие перед выполнением тела цикла, либо вы можете это сделать после того, как цикл уже пройден. В последнем случае цикл всегда выполняется хотя бы один раз.

Давайте немного поиграем. Представьте себе, что ваш шеф дал вам поручение написать программу для Access, которая считает от 1 до 10. Звучит довольно просто, но как вы это сделаете? выполнив следующие шаги, вы напишите такую программу, причем тремя различными способами.

1. Откройте новый модуль в Access.
2. Выберите **Вставка процедуры** и назовите ее Count Loop. Оставьте все параметры, в том числе Sub и Public, как есть.
3. Вставьте следующий код между строк Public Sub и End Sub (рис. 10.3) и нажмите Ctrl+G для открытия окна проверки.

```
Dim iCounter As Integer

Do While iCounter < 10
  iCounter = iCounter + 1
  Debug.Print iCounter
Loop
```

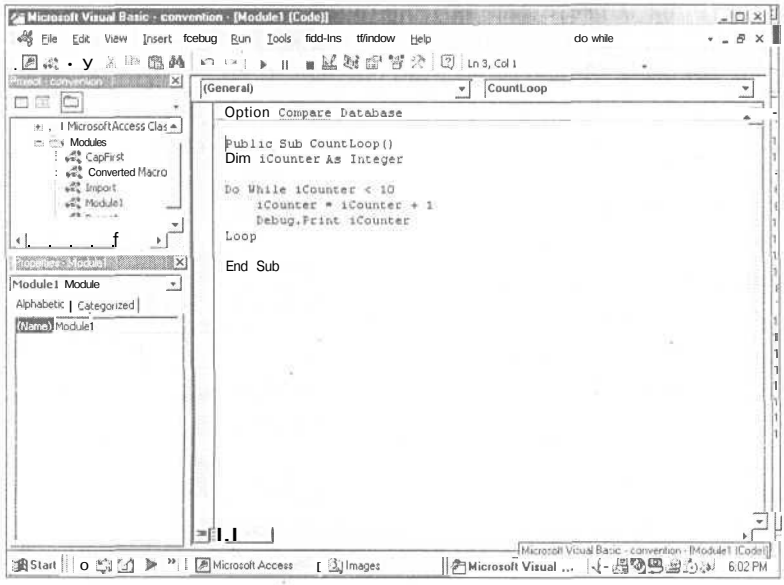


Рис. 10.3. Подпрограмма CountLoop может быть запущена из окна **Модуль**

4. Убедитесь, что курсор находится на строке Public Sub, и нажмите кнопку Run (Запуск). Вы также можете нажать F5 для запуска процедуры.
   Вы должны увидеть числа от 1 до 10 в окне проверки (рис. 10.4). Нажмите Ctrl+G, если вы забыли открыть это окно в п. 3. Вы можете щелкнуть мышью

на верхней части окна проверки и растянуть его, если вам необходимо больше места.

Кнопка Run (Выполнение) напоминает кнопку воспроизведение на CD-плейере

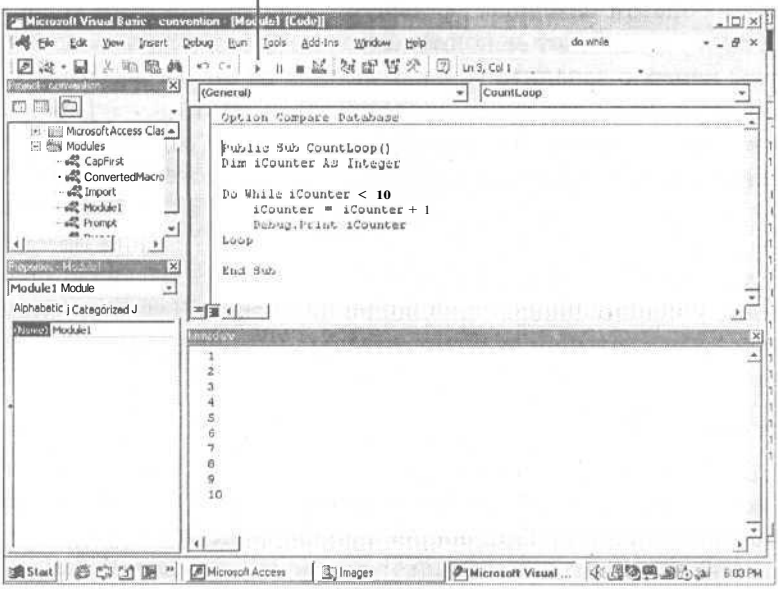


Рис. 10.4. Вы можете увидеть результаты работы процедуры CountLoop в окне проверки

5. Смените While на Until, а < на =. Строка теперь должна выглядеть так:
   Do Until iCounter = 10
6. Снова нажмите кнопку запуска.
   Результат должен быть тем же.
7. Поместите курсор перед буквой «U» слова Until. Выделите слова Until iCounter = 10, нажмите кнопку **Вырезать,** либо в меню выберите **Правка, Вырезать.**
8. Поместите курсор после выражения Loop и нажмите **Вставить** (Ctrl+V). Теперь процедура должна выглядеть следующим образом:

```
Dim iCounter As Integer

Do
iCounter = iCounter + 1
Debug.Print iCounter
Loop Until iCounter = 10
```

9. Снова запустите программу. Она должна выдать тот же результат. Не закрывайте окно для следующего примера.

Вы, наверное, думаете: «Ну и в чем же смысл? Почему бы просто не выбрать один метод и не пользоваться им?» Используя не один, а несколько способов, вы

действуете более гибко. Хотя порядок проверки условия не так важен в маленькой программе, как в этой, он становится немаловажен в более объемных и сложных программах. В той части программы, где используется Until, циклическая конструкция прерывает свою работу, когда условие становится истинным, таким образом завершая процедуру. До тех пор пока iCounter не будет равна 10, программа будет выполняться.

Если вы неверно зададите условие выхода из цикла, вы можете создать бесконечный цикл. Если это произошло, просто нажмите Ctrl+Break для прерывания программы. В результате вы вернетесь в режим редактирования, где вы можете сделать все необходимые исправления. Вы также можете выйти из цикла, вставив условно выполняющееся выражение End Do в цикл (возможно, опечатка в оригинале; нужно вставлять просто End, например If iCounter > 9 Then End - *Пер.*). Другая интересная замена, позволяющая более наглядно проиллюстрировать циклы, - вместо Debug.Print написать MsgBox. Когда вы запустите такую программу, результаты цикла будут выдаваться в окнах сообщений.

Цикл For..Next

Вы можете использовать цикл For. . Next, чтобы выполнить код, находящийся внутри цикла (так называемое «тело цикла». - *Пер.)* определенное чисто раз. Оператор Step вместе со следующим за ним численным аргументом позволяет вам наращивать переменную iCounter на число, большее или меньшее чем единица.

Предположим, ваш шеф передумал и решил, что вы должны заставить программу считать начиная от 20 до 100, прибавляя по K). Хотя вы можете сделать это при помощи цикла Do, конструкция For-Next идеально приспособлена для такой задачи.

1. В окне Visual Basic, которое вы оставили открытым с прошлого примера, сотрите все начиная с Do до 10 в старой программе и замените следующим кодом:

```
For iCounter = 20 To 100 Step 10
 MsgBox iCounter
Next iCounter
```

2. Запустите программу и наблюдайте за результатами в окнах сообщений. Убедитесь, что окно проверки открыто.
3. Закройте окно.

Заметьте, что какой-то строки не хватает. Нет строки iCounter = iCounter +1. Вам не нужно наращивать переменную в теле цикла. Кроме того, код уменьшился в объеме. Если вы уберете выражение Step 10, программа будет считать от 20 до 100 добавляя по единице. Существует другой вариант: вам не обязательно использовать оператор Step (буквально «шаг». - *Пер.*), вы можете просто добавлять дополнительное число для получения большего (или меньшего) прироста, чем по умолчанию, т. е. чем 1.

Если вам точно известно число необходимых циклов (итераций), это идеальная конструкция для ваших целей. Вам не нужно заботиться об установке каких-либо условий для окончания цикла. Если вы хотите отсчитывать в обратную сторону, просто поставьте шаг -1. Например, следующая программа ведет обратный отсчет от 100 до 90:

```
Dim iCounter As Integer

For iCounter = 100 To 90 Step -1
Debug.Print iCounter
Next iCounter
```

Цикл For..Each

Циклические конструкции, которые вы только что изучили, хороши для обычных переменных, но как быть с объектами и объектными переменными? Хотя вы можете использовать циклы Do и For для объектов, в VBA имеется циклическая конструкция, специально предназначенная для объектов.

Цикл For-Each создан специально для работы с коллекциями объектов, таких, как элементы управления. Как и цикл For-Next, For-Each пробегает последовательность значений. Разница в том, что значения эти в действительности - указатели на объекты.

Предположим, вы хотите перечислить каждый элемент управления в вашей форме. Следующая программа делает именно это. Вам нужно будет создать переменную типа Control для использования в этой программе.

1. Откройте фазу данных Convention в папке AccessByExample.
2. В разделе **Объекты** выберите **Формы** и выделите форму Client. Нажмите **Конструктор** на панели инструментов.
3. Скопируйте и вставьте кнопку **Открытие таблицы Flight.** Выбрав только что созданную кнопку, нажмите F4, чтобы открыть список свойств, как показано на рис. 10.5.

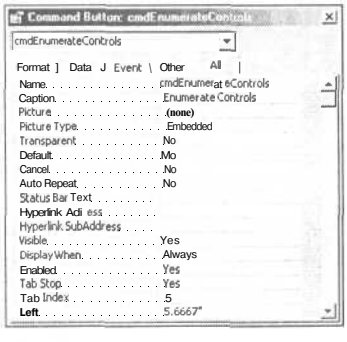


Рис. 10.5. Вы можете использовать цикл For..Each, чтобы перечислить элементы управления на вашей форме

Когда элемент управления выбран, вы также можете выбрать в меню **Правка, Дублировать,** чтобы создать копию элемента управления.

4. Выберите вкладку Все. В свойстве **Имя** наберите cmdEnumerateControls, а в Подписи - Enumerate Controls.
5. Переключитесь на вкладку **События** и щелкните на ниспадающем списке свойства **Нажатие кнопки.** Выберите **Процедура обработки событий,** а затем щелкните кнопку **Создать** для создания процедуры, которая будет использовать цикл For. . Each, как показано на рис. 10.6.

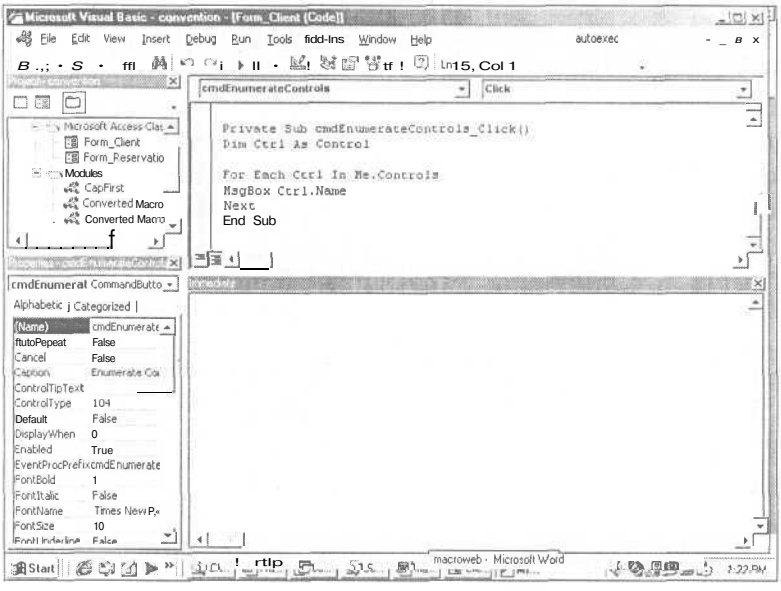


Рис. 10.6. Простая программа, присоединенная к командной кнопке для перечисления элементов управления на форме

6. После строчки Private Sub наберите следующий код:

```
Dim Ctrl As Control

For Each Ctrl In Me.Controls
MsgBox Ctrl.Name
Next
```

7. Закройте окно модуля и список свойств.
8. Нажмите кнопку Вид, чтобы переключиться в **Режим формы,** а затем нажмите кнопку Enumerate Controls. Понаблюдайте, как цикл перебирает элементы управления (рис. 10.7).
9. Закройте форму.

Рис. 10.7. Когда вы видите, как программа перебирает элементы управления, вы начинаете осознавать ту мощь, которая вам подвластна

Рассмотрим внимательнее возможности, находящиеся в вашем распоряжении. Если вы можете перебирать элементы управления, значит, вы можете менять свойства для каждого элемента управления. Если присоединить следующую программу к кнопке, вы измените цвет шрифта каждого поля для ввода и поля со списком (text box и combo box) на красный:

```
Dim Ctrl As Control
For Each Ctrl In Me.Controls
 If TypeOf Ctrl Is TextBox Then
   Ctrl.ForeColor = 255
 ElseIf TypeOf Ctrl Is ComboBox Then
   Ctrl.ForeColor = 255
 End If
Next
```

Этот код может быть слегка подправлен и прикреплен к событию формы On Current (Получение фокуса), чтобы изменить цвет каждого text box и combo box в каждой записи, имеющей поле, помеченное как Нет (в логическом поле Yes/No), на красный (или любой другой цвет по вашему желанию). Когда вы пометите все записи в вашей форме, вы увидите, что цвет шрифта изменился в каждой записи, помеченной Нет в поле Yes/No.

Несколько способов обработки условий

Программы должны уметь принимать решения в зависимости от условий. Довольно часто встречаются задачи, включающие так называемую *условную обработку,* т. е. способность определять, какое направление выберет программа в зависимости от значения переменной или совокупности других условий. Если значение X истинно, то делать Y. Если ложно, то делать Z. Если у Ральфа средний балл 4 или выше, то назначить ему стипендию. В противном случае не назначать ему стипендию. Это реальные жизненные ситуации, с которыми вы можете столкнуться. Короче говоря, вам нужно иметь возможность контролировать свою программу. Вы уже встречались с нижеследующими структурами, контролирующими ход программы, теперь у вас есть возможность оценить их на примере.

Конструкция If-Then-Else

Конструкция If-Then-Else позволяет вам проверить условие и в зависимости от того, истинно ли оно, произвести действие или серию действий. Это условие может быть одним значением, диапазоном значений или совокупностью значений, совмещенных вместе операторами, такими, как AND и OR. Табл. 10.4 иллюстрирует несколько примеров.

Таблица 10.4. Типы выражений, задающих условие

Условие	Пример
Одно значение	X = 2
Диапазон значений	X>=100 and X <200
Совокупность значений	X = 220 and Y = 480 or Z = 370

Если вы правильно установили диапазон значений в выражении после If, то все, что попадает в этот диапазон, становится истинным. Если ваш диапазон от 100 до 200, то 150 или любое другое значение из этого диапазона также является истинным. Когда установлена совокупность значений, только определенные значения истинны. Однако если оператор AND стоит между двумя значениями, *оба* значения должны присутствовать для того, чтобы условие стало истинным. С другой стороны, при операторе OR, если *хотя бы одно* значение присутствует, условие будет истинным.

Запомните, что оператор Else не обязателен. Вы можете ничего не предпринимать, если условие не является истинным. Необходимость этого оператора определяется в каждой конкретной ситуации. Следующий код демонстрирует это.

```
Sub done()
Dim i
For i = 1 To 10
  If i > 8 Then
    MsgBox "Almost done!"
  End If
Next i
End Sub
```

Нет нужды использовать выражение Else, поскольку в сообщении высвечивается только переменная больше восьми. Если условие не выполнено, программа идет дальше, просто продолжая считать.

ЗАМЕЧАНИЕ. Если вы объявляете переменную без указания типа данных, ей приписывается тип данных variant (дословно «различный» - *Пер.*). Тип данных variant применяет больше ресурсов, поэтому будьте разборчивы при его использовании.

Когда вы попадете в более сложные ситуации, вы поймете, что оператор Else может быть существенно полезным. Рассмотрим отрывок из программы, обещанной выше в этой главе.

```
For i = 1 to Len(StrInput)
  Char = Mid(StrInput, i, 1)
  If Char = ' ' then
    Mark = i
  ElseIf i = Mark + 1 Then
    Mid(StrInput, i, 1) = Ucase(Mid(StrInput, i,1))
  Else
    Mid(StrInput, i, 1) = Lcase(Mid(StrInput, i,1))
  End If
Next I
```

В этой процедуре вам нужно выполнить специальную функцию в случае, если условие не является истиной. Если рассматриваемый вами символ на одну позицию правее пробела, то сделать его заглавным. Если нет, сделать его строчным. Оператор ElseIf обеспечивает возможность проверки еще одного условия не переходя к выражению Select-Case.

Чтобы узнать больше о пользовательской процедуре, переводящей первые буквы каждого слова в верхний регистр, см. «Использование функций для настройки вашего приложения» далее в этой главе.

Конструкция Select-Case

Конструкция Select-Case заполняет брешь, оставленную конструкцией If-Then-Else. Если вы просто проверяете одну переменную на несколько условий, то Select-Case - ваш выбор. Выражение If также может проверять несколько значений, но когда вы вставите все многочисленные условия при помощи Elself, ваша программа будет выглядеть умопомрачительно. Здесь-то и приходит на помощь конструкция Select-Case. Она разработана специально для многочисленных условий.

Начнем с простого примера. Предположим, вы хотите передавать различные сообщения на каждом шаге цикла. Следующий код иллюстрирует такую ситуацию.

```
Dim i
For i = 1 to 3
  Select Case i
    Case 1
    MsgBox "This is the first pass."
    Case 2
    MsgBox "This is the second pass."
    Case 3
    MsgBox "This is the third pass."
  End Select
Next I
```

Заметьте, что здесь нужно проверять только одну переменную на многочисленные условия. Вы также можете использовать диапазоны и совокупности значений для каждого оператора case, как показано ниже:

```
Select Case i
  Case 2, 4, 10 to 20, 31, 43
  MsgBox "Number found"
  Case Is = NewNum
  MsgBox "Number found"
  Case Else
  MsgBox "Sorry, no match"
End Select
```

В этом примере, если значение переменной, указанной в операторе Select, совпадает с указанными числами, либо является одним из диапазона чисел (от 10 до 20), либо равно переменной NewNum, появляется окно сообщений, говоря-

щее, что Number found (Число найдено). Иначе сообщение говорит Sorry, no match (Извините, нет совпадения).

Представляйте себе эту конструкцию как фильтр. Переменная проверяется на соответствие каждому значению, пока не найдется совпадение; в этом случае выполняется код, следующий за оператором Case. После того как условие совпало, все дальнейшие условия пропускаются. Если не найдено ни одного совпадения, выполняется код, следующий за Case Else.

Еще одним преимуществом использования конструкции Select-Case - удобство чтения. Гораздо проще читать список свойств в Select-Case, чем в If-Then-Else с этими бесконечными ElseIf. Оператор Case Else обычно рекомендуют использовать только в случае, если все условия не являются истинными.

Использование функций для исполнения вашего списка пожеланий и предложений

К теперешнему моменту вы начинаете осознавать ту власть, которая доступна вам посредством программирования. У вас, как и у многих пользователей, наверняка есть список пожеланий и предложений с опциями, которые вы хотите видеть в базе данных. Если вам не удалось найти эти опции в окнах подсказки и в руководствах, время попытаться создать их самим. Если вы оформите вашу опцию как функцию, она будет доступна практически в любом месте базы данных.

В начале этой главы вам обещали показать функцию, которая делает прописной первую букву в каждом слове строки. На рассмотрение представлены два варианта. В первом варианте происходит «прыганье» по словам, используя функцию InStr для отыскания пробелов в строке. Во втором варианте анализируется вся строка символ за символом. У вас есть возможность получить полезный опыт от рассмотрения этого варианта. Взгляните но эту программу:

```
Function Proper(strWord As String)
Dim GetWord As String, pos As Integer
Dim Build As String, NewString As String, Char As String

NewString = strWord & " "
Do Until NewString = ""
  pos = InStr(NewString, " ") 'находим положение пробела
  GetWord = Left(NewString, pos) 'получаем слово
  'вычитаем слово для следующей итерации
  NewString = Right(NewString, Len(NewString) - pos)
 For i = 1 To Len(GetWord)
  Char = Mid(GetWord, i, 1) 'анализируем каждый символ
  If i - 1 Then
   Char = UCase(Char) 'переводим в верхний регистр, если это первая буква
  Else
   Char = LCase(Char)
  End If
 Build = Build & Char "строка переделывается по одной букве за шаг
 Next i
Loop
Proper = RTrim(Build) 'удаляем пробел в конце
End Function
```

вам не обязательно понимать все, что происходит в этой функции, но здесь присутствуют многие вещи из тех, с которыми вы познакомились в этой главе. Программа берет по одному слову из строки и анализирует каждое слово по одному символу. Она ищет первый символ каждого слова, чтобы сделать его заглавным. Читайте комментарии, чтобы проще было воспринимать сложные строки. Здесь активно используются встроенные строковые функции. Также обратите внимание на циклы и условные конструкции.

Использование функций для настройки вашего приложения

Теперь давайте посмотрим на функцию, производящую точно такую же операцию, но делающую это при гораздо меньшем количестве строк. Эта функция прописана в базе данных Convention, которую вы скачали. В этой функции не добавляется и не убирается никаких пробелов; она просто проверяет каждый символ как есть и изменяет его. Давайте посмотрим, как она работает.

```
Function CapFirstLetter(StrInput As String)
Dim Mark As Integer, Char As String

For i = 1 To Len(StrInput)
 Char = Mid(StrInput, i, 1)
 If Char = " " Then
  Mark = i
 ElseIf i = Mark + 1 Then
  Mid(StrInput, i, 1) = UCase(Mid(StrInput, i, 1))
 Else
  MicKStrInput, i, 1) = LCase(Mid(StrInput, i, 1))
 End If
Next i
CapFirstLetter = StrInput

End Function
```

Вся эта процедура использует только три переменные. Обратите внимание, насколько она короче. Позиция, которую занимает пробел в строке, записывается в переменную Mark. Когда позиция, занимаемая в строке, совпадает с Mark + 1 (т. е. первый символ после пробела), символ делается заглавным. Все другие символы делаются строчными.

Использование окна проверки для тестирования ваших функций

Обратите внимание на то, как использована функция mid в CapFirstLetter: - она обеспечивает идеальный способ анализирования строки по одному символу за шаг. В данном случае вы не только анализируете каждый символ, вы также изменяете символы, отвечающие соответствующим условиям.

Окно проверки предоставляет вам простой и быстрый способ проверки ваших программ, используя возможности его командной строки. Следуя данным инструкциям, вы протестируете функцию CapFirstLetter в окне проверки.

1. Откройте базу данных Convention, в разделе **Объекты** выберите **Модули,** выделите Proper и нажмите **Конструктор,** чтобы открыть модуль в конструкторе.
2. Нажмите Ctrl+G, чтобы открыть окно проверки.
3. Поместив курсор в окно проверки, наберите ? CapFirstLetter("pAR< pLaCe") и нажмите ввод. Программа должна вернуть Park Place, как показано на рис. 10.8.

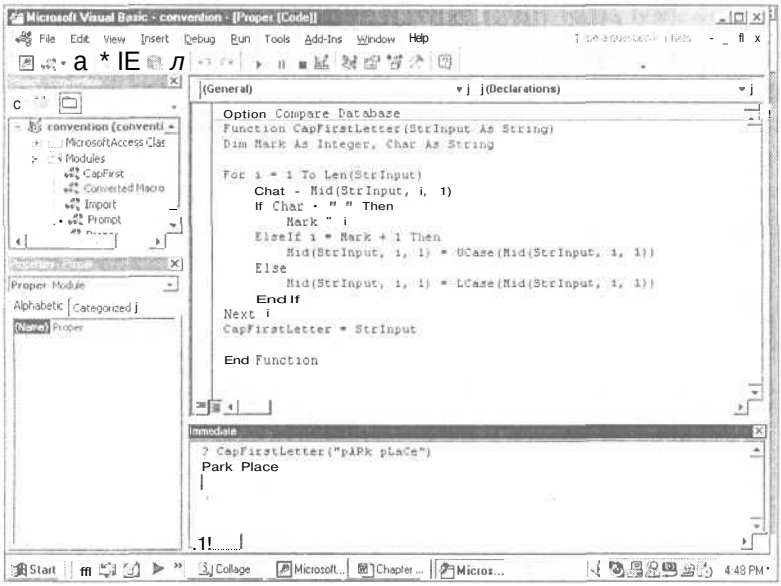


Рис. 10.8. Окно проверки показывает значение, возвращаемое функцией, убеждая вас, что ваша программа работает правильно

4. Закройте все окна, относящиеся к модулю.
5. В разделе **Объекты** выберите **Таблицы.**
6. Дважды щелкните мышью по таблице Speakers.
7. Обратите внимание, что там все набрано в верхнем регистре. Можно предположить, что эти данные получены от постороннего источника. Закройте таблицу.
8. Щелкните правой кнопкой по таблице Speakers и выберите **Копировать.** Теперь нажмите **Вставить** и назовите ее tblSpeakers.
9. Выделите таблицу tblSpeakers.
10. Из панели инструментов выберите **Новый объект, Запрос** - откроется диалоговое окно **Новый запрос.**
11. Нажмите OK, чтобы подтвердить выбор **Конструктор** в диалоговом окне **Новый запрос.**

12. Дважды щелкните мышью по заголовку окна таблицы tblSpeakers и затем перетащите мышью поля в сетку (щелкните левой кнопкой, зажмите и тяните).
13. Щелкните на ниспадающем списке кнопки **Тип запроса** и выберите **Обновление.**
14. В строке **Обновление** для поля Speaker наберите `CapFirstLetter([Speaker])`.
15. В строке **Обновление** для поля Topic наберите `CapFirstLetter([Topic])`, как показано на рис. 10.9.

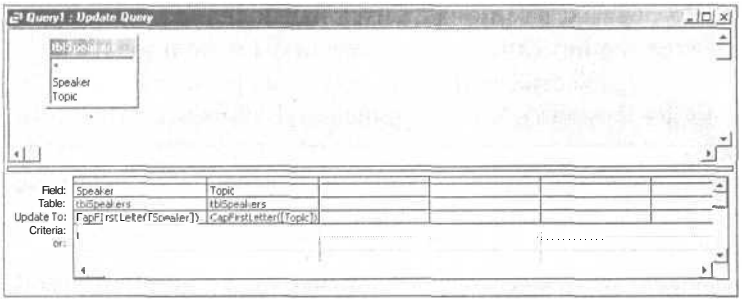


Рис. 10.9. Вы можете совмещать пользовательские функции и запросы, значительно увеличивая диапазон возможностей последних

16. Запустите запрос. Закройте запрос без сохранения и откройте таблицу tblSpeakers.
Как видите, все изменилось (рис. 10.10) - все слова начинаются с заглавной буквы. Если же все прошло не так, как надо, у вас есть изначальная таблица Speakers, к которой можно вернуться. Если все прошло удачно, то налюбовавшись, таблицу tblSpeakers можно удалить.

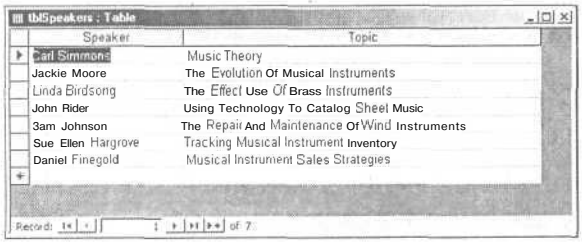


Рис. 10.10. Функция CapFirstLetter перевела все данные в таблице tblSpeakers из верхнего регистра в комбинацию заглавных и строчных символов

А теперь представьте, если бы в таблице tblSpeakers было несколько тысяч записей и всех их вам нужно было бы исправлять по заданию шефа! Подумайте, насколько непосильной задачей это могло бы оказаться, если б не использование функций.

Использование функций вместо шаблонов в запросах

Создание пользовательских функций не только поможет вам выполнить ваш список пожеланий и предложений. Функциями можно восполнить тот недостаток опций, которые забыли включить в Access. Иногда в процессе решения задачи вам приходится искать возможности, которые не включены в стандартный пакет Access.

Предположим, у вас есть таблица, полученная от постороннего источника, в которой имеется список ФИО, которые вы хотите разбить на отдельные поля: имя, отчество и фамилия. Во-первых, вам понадобится найти записи, в которых указаны только имя и фамилия, но нет отчества. Вы могли бы использовать подстановочные знаки Like "* *" в качестве условия отбора в запросе, по которому будут выбираться только имя + фамилия. Единственная загвоздка состоит в том, что под такой шаблон будут также попадать имя + отчество + фамилия. Если бы вам понадобилось выделить тройные имена из списка, содержащего арабские имена, Like "* * *" с тем же успехом выделил бы и имя известного математика аль-Хорезми Абу Абдалла Мухаммед бен Муса аль-Маджуси.

Но есть и хорошие новости: используя простую функцию Convention, закачанную вами вместе с базой данных, вы можете считать слова, что позволяет решать такого рода проблемы. Такое решение обеспечивает большую точность при проверке на совпадение по шаблону. Посмотрите следующую программу:

```
Function CountWord(FullString As String)
Dim AddWord As String, Build As String
Dim Iteration As Integer

Build = FullString & " "   'пробел является маркером конца строки

 Do Until Left(Build, InStr(Build, " ")) = " "  'устанавливает предел
  AddWord = Left(Build, InStr(Build, " ")) 'получаем слово
   If AddWord <> " " Then
    Iteration = Iteration + 1 'подсчитываем только полноценные слова
   End If
  Build = Right(Build, Len(Build) - Len(AddWord)) 'вычитаем слово
 Loop

CountWord = Iteration

End Function
```

Вы можете использовать эту функцию как критерий в запросе для подсчета слов, таким образом вы можете селективно подходить к разделению слов в записях, имеющих ровно то количество слов, которое вы искали. Функция считает слова путем вычисления количества итераций, необходимых для перебора всех пробелов в строке. В заключение скажем, что процедуры, сделанные пользователем, обеспечивают вашему приложению гораздо большую гибкость, так как они позволяют делать то, что вам заблагорассудится, вне зависимости от того, существует ли в Access встроенная функция или какая-либо другая опция такого рода или нет.

Что дальше?

вы видели, как VBA открывает целый диапазон новых возможностей, позволяющих вам осуществить свои желания. Вы можете обходить существующие ограничения и заполнять оставленные бреши в предоставляемых Access опциях, создавая свои собственные процедуры. В следующей главе объясняется, что делать, если вы сталкиваетесь с ошибкой. Для вас, разработчика, существует несколько возможностей для управления ошибками. Процедуры обработки ошибок помогают сделать ваше приложение более профессиональным и более удобным для пользователя. Другими словами, эффективное управление ошибками покажет вас и ваше приложение в лучшем свете.

11
Обработка ошибочного кода в Access

В предыдущей главе вы научились переделывать вашу базу данных, используя функции, доступные в VBA. Содержание этой главы сосредоточено на том, что делать, если у вас возникла ошибка. Добавление обработки ошибок в ваши процедуры делает выполнение кода более ровным. В частности, в этой главе вы узнаете:

- какие опции вам доступны, когда у вас имеется ошибка;
- как определить, какие ошибки возникли;
- как написать процедуры, которые дают информацию о том, какие ошибки возникли;
- как исправить ошибки после их обнаружения;
- как использовать оператор Resume.

Что делать, когда возникают ошибки

В идеале ошибка никогда не должна окончательно испортить вашу программу. Любой, кто четко представляет себе реальное программирование, должен признать, что программы, как бы аккуратно они ни были написаны, время от времени будут иметь ошибки. Если уж это не подлежит сомнению, то не лучше ли вам добавить операции обработки ошибок в ваши процедуры?

Давайте взглянем на некоторые из опций, которые открываются вам, когда возникает ошибка.

- Игнорировать ее, какой бы она не была, в вашей программе, не добавляя никакой обработки ошибок.
- Игнорировать ее в вашей программе с помощью обработки ошибок, которая продолжает работу программы, обходя ошибку.
- Обработать ошибку в вашей программе, зафиксировав ее, остановив программу и выдав пользователю сообщение об ошибке с вариантом продолжения работы или поиском дополнительной информации.
- Обработать ошибку, зафиксировав ее, после чего вызвать подпрограмму, которая устраняет проблему.
- Обработать ошибку, занеся запись о ней в таблицы, предоставив пользователю возможность продолжить.
- Обработать ошибку, подправив код.

Третий вариант используется чаще всего. Четвертый вариант не всегда возможен. Например, конкретная таблица может быть недоступна потому, что он" была случайно удалена. Хотя таблицу можно создать «на лету» в подпрограмме обработки ошибок, это нужно предвидеть заранее. Вы можете скомбинир

четвертый и пятый варианты, выдавая пользователю сообщение с опцией исправить ошибку, создав таблицу. Последний вариант является хорошим способом протестировать ваше приложения начиная с его запуска. Если бы у вас была таблица, которая использовалась бы в качестве журнала ежедневно возникающих ошибок, то было бы несложно отладить вашу программу, одновременно определяя, куда следует добавить дополнительные программы обработки ошибок.

Последний вариант является отличным способом обработки ошибок, хотя он иногда очень сложен. Например, если вы проходите по всем элементам управления на форме, изменяя цвет(forecolor) каждого элемента управления, вы создаете ошибку, ибо не все элементы управления имеют свойство цвета. Вы можете полностью обойти эту ошибку, используя конструкцию If..Then.. Else для проверки каждого элемента управления. Так как программа выделяет только элементы TextBoxe и ComboBoxe, то элементы без свойства цвета успешно исключаются. Следующая часть кода, взятая из предыдущей главы, выглядит так:

```
If TypeOf Ctrl Is TextBox Then
  Ctrl.ForeColor = 255
ElseIf TypeOf Ctrl Is ComboBox Then
  Ctrl.ForeColor - 255
End If
```

Не нужно писать никакой программы обработки ошибок, по крайней мере для этой конкретной задачи. Это означает, что пользователь никогда не столкнется с ошибкой. Это также означает, что вы экономите время, которое потребовалось бы на написание дополнительного кода для обнаружения ошибки. К сожалению, это не всегда возможно. Но когда можно исправить ошибку на уровне кода, обязательно воспользуйтесь этой возможностью.

Определение типа возникшей ошибки

Естественно возникает вопрос: «А откуда мне знать какая ошибка возникла?» И хотя Access предоставляет коды ошибок и сообщения об ошибках, для пользователей зачастую трудно интерпретировать их. Если этого недостаточно, то они обычно не могут понять, что делать с ошибкой. Обработка ошибок делает ваше приложение более удобным для пользователя, тем временем предоставляя вам лучший контроль за тем, что может встретить пользователь.

Предположим, что у вас имеется таблица с кодами ошибок и соответствующие им сообщения. Следующая программа DAO создает такую таблицу.

```
Sub GenerateErrorTable()
Dim db As DAO.Database, rs As DAO.Recordset
Dim i As Integer, Msg$
Set db = CurrentDb()
Set rs = db.OpenRecordset("ErrorCodes")
For i = 1 To 32766
 Msg$ = Error$(i)
 If Msg$ <> "Application-defined or object-defined error" _
 And Msg$ <> "Reserved Error" Then
  rs.AddNew
  rs![ErrorNumber] = i
```

```
    rs![ErrorMsg] = Msg$
    rs.Update
  End If
Next i
rs.Close
db.Close
Set rs = Nothing
Set db = Nothing
End Sub
```

Вам не следует беспокоиться о создании этой таблицы самим. Это уже сделано за вас. Она находится в скаченной вами базе данных ErrorHandle.mdb вместе с кодом, лежащим в ее основе. Обратите внимание на выражение If в программе. Оно устраняет всякое сообщение, которое относится к Ошибкам, определяемым приложением или объектом (Application-defined or object-defined error). Единственная проблема состоит в том, что большая часть записей содержит это сообщение. Соответственно только 87, а не несколько тысяч записей остается в таблице.

Откройте таблицу ErrorCodes из базы данных ErrorHandle, показанной на рис. 11.1. Это только небольшая часть сообщений, которые генерирует Access. Остается вопрос: «А как мне найти оставшиеся сообщения?»

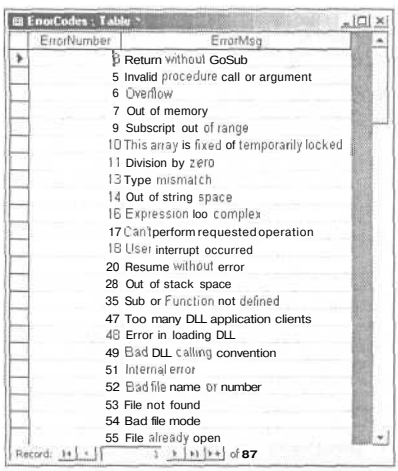


Рис. 11.1. Ошибки из-за отсутствия данных, приведенные в таблице ErrorCodes

Перед тем как ответить на этот вопрос, нужно задать еще один вопрос и ответить на него. Что такое Microsoft Jet? Это есть механизм управления базой данных, который создает таблицы и запускает запросы. Хотя у вас есть выбор в механизмах управления базой данных, таких, как Access 2000, все же Jet Database Engine более подходит для не слишком больших приложений. Что же в нем такого для нахождения сообщений об ошибках?

Как объектные модели DAO, так и объектные модели ADO могут давать ошибки при работе с Jet. По мнению Microsoft, DAO гораздо более эффективно

в использовании механизма управления базами данных Microsoft Jet, чем ADO, и многие операции быстрее DAO, иногда в 5-10 раз. С другой стороны, ADO является превосходной объектной моделью для решений по дизайну интерфейсной части приложений типа клиент-сервер, в который обычно привлекаются большие базы данных. Так как обе эти объектные модели имеют свои преимущества и недостатки и обе они дают ошибки, то вам предоставлена возможность посмотреть примеры обработки ошибок в обеих объектных моделях.

Как DAO, так и ADO имеют коллекции ошибок; однако ошибки DAO являются частью объекта DBEngine, тогда как ошибки ADO являются частью объекта Connection.

Следующее упражнение даст вам лучшее представление о коллекциях ошибок обеих объектных моделей.

1. Еще раз откройте базу данных ErrorHandle, если она еще не открыта.
2. В меню **Объекты** (Objects) выберите **Формы** (Forms), а затем щелкните дважды по форме JetErrorCodes.
3. Выделите 1 в индикаторе записи внизу формы (обратитесь к рис. 11.1). Наберите 51 и нажмите Enter, чтобы перейти к записи 51.
   На экране появляется код ошибки 3024, который выдает Could not find file <name>, как показано на рис. 11.2.

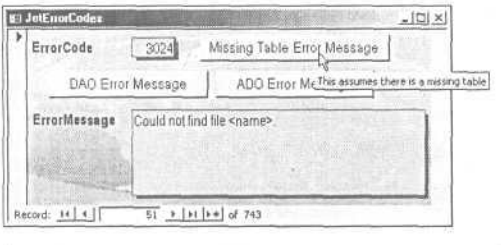


Рис. 11.2. Запись 51 показывает код ошибки 3024, которая является ошибкой из-за отсутствия данных таблицы, а также сообщение об ошибке Could not find file <name> (Не найден файл <имя >)

4. Поместите курсор на командную кнопку Missing Table Error Message (Сообщение об ошибке из-за отсутствия данных в таблице) и не нажимайте по крайней мере 3 с. Вы должны увидеть всплывающую подсказку This assumes there is *a* missing table (Здесь находится таблица с пропущенными данными). Фактически эта кнопка работает так, как будто бы в ней находится таблица с пропущенными данными.
5. Нажмите кнопку Missing Table Error Message, чтобы открыть всплывающую форму, показанную на рис. 11.3.

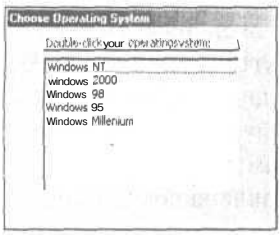


Рис. 11.3. Всплывающая форма позволяет вам выбирать операционную систему, таким образом выдавая правильный путь к системе подсказок

На рис. 11.3 показана всплывающая форма, которая, появляясь, просит вас выбрать операционную систему. Для свойств PopUp и Modal этой формы выставлено значение Yes. Для свойства BorderStyle выставлено значение property Dialog, а для свойства ControlBox выставлено значение No. При использовании оператора OpenForm объекта DoCmd для открытия формы вы должны выставить значение Dialog для Window Mode (Оконный режим) (см. процедуру на шаге 13), таким образом откладывая процедуру до тех пор, пока форма не будет закрыта. В противном случае в верхней части формы откроется диалоговое окно.

6. После выбора вашей операционной системы нажмите кнопку Help E появившемся диалоговом окне, как показано на рис. 11.4.

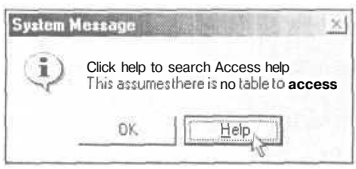


Рис. 11.4. Диалоговое окно, которое появляется, когда нажата кнопка, содержит кнопку Help

По нажатии кнопки помощи появляется окно с двумя панелями. Если у вас отображается только одна панель, то нажмите кнопку Show (Показать) в верхней части окна. На панели слева отображается ссылка на сообщение об ошибке Microsoft Jet, а на панели справа появляется сообщение, на которую была сделана ссылка. Если вы нажмете плюс (+) слева от ссылки на сообщение об ошибке Microsoft Jet, а затем нажмете Error Code Index (Индекс кодов ошибок), то вы увидите весь список кодов ошибок и соответствующих сообщений, как показано на рис. 11.5. Они показаны в формате HTML. Эти ошибки есть то, что вы видите в исходной таблице формы, в который вы на данный момент работаете.

7. Закройте окно помощи Microsoft Access, а затем нажмите OK в диалоговом окне, которое должно быть до сих пор открыто с шага б (см. рис. 11.4).

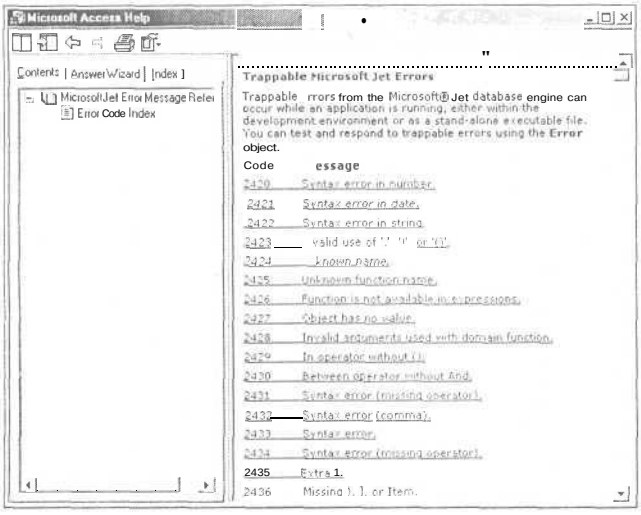


Рис. 11.5. Окно ссылок на сообщения об ошибках Microsoft Jet содержит индекс фиксируемых ошибок Microsoft Jet

8. Поместите курсор на кнопке, на которой написано DAO error message (Сообщение об ошибке DAO), по крайней мере на 3 с. Обратите внимание на то, что подсказка для этой кнопки говорит вам «Укажите ссылку на последнюю библиотеку DAO» (Set reference to latest DAO library). Для базы данных это уже было выставлено.
9. Нажмите кнопку сообщения об ошибке DAO, чтобы открыть диалоговое окно, как показано на рис. 11.6.

Рис. 11.6. Это диалоговое окно для ошибки DAO выдает вам релевантную информацию, такую, как источник ошибки

Номер ошибки такой же, какой и номер текущей записи (если она по-прежнему запись под номером 51). Заметьте также, что она сообщает Source: DAO workspace. (Источник: Рабочая среда DAO). Это означает, что DAO допустила ошибку (сейчас она и вправду ее допустила). И наконец, заметьте, что она ссылается на раздел в файле jeterr40.chm. Все это очень полезная информация; гораздо более полезная, чем та, которую вы обычно получаете. Давайте теперь сделаем быстрый взгляд «за кулисы», чтобы лучше понять все это.

10. Будучи по-прежнему в форме, нажмите OK в ответ на сообщение, а затем нажмите кнопку View, чтобы перейти в окно конструктора.
11. Нажмите кнопку Missing Table Error Message (Сообщение об ошибке в таблице с пропущенными данными), а затем нажмите F4, чтобы открыть список свойств кнопки.
12. Щелкните по свойству Event(Событие), а затем выберите событие On Click.
13. Просмотрите следующий код:

```
Private Sub cmdMissingTable_Click()
Dim Button As Integer, Title as String
Dim HelpFile As String
DoCmd.OpenForm "PopUp",,,,,acDialog
  Button - 64
  Title = "System Message"
  HelpFile = "Jeterr40.chm"
  MsgBox "Click help to search Access help" & _
  vbCrLf & "This assumes there is no table to access", _
  Button + vbMsgBoxHelpButton, Title, Path & HelpFile, 5003024
End Sub
```

Обратите внимание на переменную пути. Это открытая переменная, определенная в стандартном модуле Global. Может потребоваться ее изменить, так как ваша операционная система может переместить файл в другую папку. Однако Access все равно должен найти файл Jeterr40.chm. К тому же ваша операционная система должна быть в списке в всплывающей форме. Однако если файл помощи не находится, то поищите его на вашем жестком диске и измените путь к папке, которую Windows находит в поле характеристики события On Dbl Click формы, называемой Popup.

Так как об аргументах MsgBox уже было рассказано, то остается рассмотреть лишь немногое. Обратите внимание на то, что HelpFile и HelpContextID выставлены в аргументах окна сообщения. Вместо кода ошибки 3024 HelpContextID считывает 5003024.

ЗАМЕЧАНИЕ. Константа vbCrLf, встроенная в VBA, очень удобна. Часть CrLf константы символизирует перенос строки с возвратом *в* исходное положение (carriage return line feed). Вместо того чтобы затруднять себя написанием строчки типа "& Chr(13) & Chr(10)", чтобы добавить новую строку, разбивая таким образом длинную строчку, вы можете просто воспользоваться этой константой. Вы также можете видеть укороченную форму vbCr этой константы, которая очень подходит для текста на экране.

---

14. Пока вы все еще в модуле, перемещайте курсор вниз, пока не увидите код программы обработки ошибок DAO. Заметьте, что следующий код слегка отличается от кода из Access Help:

```
Private Sub Command5_Click()
'Из файла помощи Microsoft
'Нужно выставить последнюю библиотеку DAO в Tools Reference
 Dim dbsTest As Database
```

```
    On Error GoTo ErrorHandler
    ' Умышленно переключает ошибку.
    Set dbsTest = OpenDatabase("NoDatabase")
    Exit Sub
ErrorHandler:
    Dim strError As String
    Dim errLoop As Error
    ' Нумирует коллекцию ошибок и отображает свойства каждого объекта с
    'ошибкой.
    For Each errLoop In Errors
     With errLoop
      strError = "Error #" &.Number & vbCr
      strError = strError & _
       " " &.Description & vbCr
      strError = strError & _
       " (Source: " &.Source & ")" & vbCr
      strError = strError & _
       "Press F1 to see topic " &.HelpContext & vbCr
      strError = strError & _
       " in the file " &.HelpFile & "."
     EndWith
     MsgBox strError
    Next
    Resume Next
End Sub
```

Подпрограммы проверки ошибок

Следует обратить ваше внимание на некоторые детали этой программы. Первое, на что надо обратить внимание, - это оператор On Error Goto [Label]. Он работает только тогда, когда есть ошибка, но в этом случае намеренно делается так, чтобы ошибка сработала, что отмечено в комментарии. Заметьте, что оператор отправляет программу на метку ErrorHandler, где ошибка обрабатывается путем нахождения свойств ошибки для MsgBox. Второе, на что надо обратить внимание - это конструкция For. . Each, о которой уже было рассказано и с помощью которой, чтобы найти ошибку, делается проход через коллекцию ошибок, которая в этом случае содержит только одну ошибку. Оператор цикла With, вложенный в цикл For. . Each, есть превосходный способ просмотреть, сосчитать или изменить свойства объекта. Следующий пример иллюстрирует работу оператора «точка» (.), работающего совместно с оператором цикла With, получая доступ к свойствам любого объекта, н торый была сделана ссылка.

```
With MyControl
.FontSize = 12
.ForeColor = 255
End With
```

Исправление ошибок после их обнаружения

В предыдущем примере для обработки ошибки вызывалось окно MsgBox с дополнительной информацией. Это один из вариантов, но в конкретных случаях возможно по-настоящему исправить ошибку, которая является причиной проблемы, - в особенности если эту проблему невозможно исправить путем исправления кода. Следующая программа обрабатывает ошибку, создавая недостающую таблицу, которая была причиной ошибки.

```
Private Sub Command18_Click()
Dim td As TableDef, fld As Field, test As String
Dim db As Database, rs As Recordset, ws As Workspace
DoCmd.SetWarnings (False)
DoCmd.RunSQL "Delete NonCurrent.[Table Name] " & _
"FROM NonCurrent;"
DoCmd.SetWarnings (True)
' Эта программа создает таблицу с таблицами из вашей базе данных
 Set ws = CreateWorkspace("JetWorkspace", "admin", _
  "", dbUseJet)
 Set db = ws.OpenDatabase("l:\office\Archive.mdb")
 On Error GoTo FixIt
 test = CurrentDb.TableDefs("NonCurrent").NAME ' посмотрите, существует
ли таблица
 Set rs = CurrentDb.OpenRecordset("NonCurrent") 'если существует, то от-
кройте ее
 For Each td In db.TableDefs
  If Left(td.NAME, 2) <> "MS" Then
  With rs
    .AddNew
    ![Table Name] = td.NAME
    .Update
    .Bookmark =.LastModified
  End With
  End If
 Next
Set db = Nothing
Set td = Nothing
Me.Refresh
Exit Sub
Fixlt:
 Set td = CurrentDb.CreateTableDef("NonCurrent")
 Set fld = td.CreateField("Table Name", dbText, 55)
 td.Fields.Append fld
 td.Fields.Refresh
 db.TableDefs.Append td
 db.TableDefs.Refresh
 Resume Next
End Sub
```

Хотя эта DAO-программа большая и, может быть, несколько «продвинутая» для вас на настоящий момент, все же есть пара деталей, на которые вам следует обратить внимание. Эта программа использует пустую таблицу (NonCurrent), заполняя ее названиями таблиц из коллекции TableDefs. Если таблица

NonCurrent не существует, то код, следующий после FixIt, создает нужную программе таблицу. Оператор On Error GoTo ведет себя таким образом, как будто ничего не произошло, так как он незаметен для пользователя. Не требуется вывода сообщения об ошибке, так как все уже исправлено.

Операторы типа Resume

В списке вариантов в начале главы есть вариант, который нужно учесть в случае, когда встречающуюся ошибку следует игнорировать в вашей программе с помощью подпрограммы обработки ошибки, которая продолжает программу, обходя ошибку. Оператор On Error Resume Next позволяет программе работать дальше, избавляя пользователя от просмотра ошибки, или, что еще хуже, просмотра кода программы. К сожалению, такой вариант не дает никакой информации об ошибке и как исправить ее. Фактически пользователь даже и не подозревает, что возникла ошибка. Это может быть опасно.

Хотя существуют ситуации, подходящие для этого оператора, все же его использование следует минимизировать. Один из примеров должного использования оператора On Error Resume Next находится в *гл.* 18 в разделе «Использование метода файлового диалога». Если вы хотите отложить прочтение сообщения об ошибке или ее исправление, когда выскакивает окно сообщения об ошибке, то используйте оператор On Error Resume Next.

Неплохо было бы вернуться в форму JetErrorCodes и нажать ту последнюю кнопку, которая создает сообщение от ADO. В базе данных ErrorHandle имеются наборы ссылок, как DAO, так и ADO, так что кнопка должна сработать правильно. Если она не сработает, то попробуйте отменить выбор ссылки DAO и снова нажмите кнопку.

Метод Err.Raise

Есть вероятность, что время от времени могут возникать две ситуации, в которых требуется нарочно создать ошибку или симулировать ее. Для начинающего программиста кажется нелепым создание или симуляция ошибки. Неужто новичку-программисту и без этого мало ошибок? В каких ситуациях возникает такая необходимость?

Предположим, что вы захотели послать вашему пользователю сообщение в случае, когда значение превышает конкретный номер, или вы захотели протестировать конкретную ошибку, чтобы определить лучший способ ее обработки. В последнем случае вам нужно создать ошибку, а в первом - симулировать ее.

Следующая программа создает сообщение об ошибке, когда номер превосходит определенное значение. Вы вводите большой номер ошибки такой, что вы не повторите код ошибки, который использует система, но не превосходящий числа 65535, являющегося наибольшим значением, которое примет свойство Number. Обратите внимание на то, как создается ошибка.

```
Function CheckNumber(lngNumber As Long)
On Error GoTo HandleRaisedErr
  If lngNumber > 100000 Then
      Err.Raise 65530, Application.CurrentObjectName, _
```

```
        "You have exceeded 100,000. Try a lower number"
    End If
Exit Function
HandleRaisedErr:
MsgBox "Error Number: " & Err.Number & vbCrLf & "Error Source: " _
& Err.Source & vbCrLf & "Error Description: " & Err.Description
End Function
```

Вы можете проверить этот код в базе данных ErrorHandle откуда угодно, нажав Ctrl+G и набрав ? CheckNumber(100100). Появится окно сообщения, показанное на рис. 11.7.

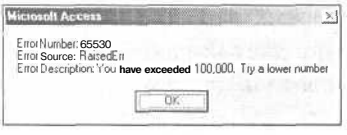


Рис. 11.7. Окно сообщения err.raise показывает выбранную ошибку, нужную вам

В следующем случае вы симулируете ошибку в тестовых целях. Наиболее часто встречающееся сообщение об ошибке - это Type mismatch. (Несоответствие типов). Если вы хотите написать подпрограмму обработки этой ошибки, вы можете симулировать ее, набрав

```
Err.Raise 13
```

Вот и весь код, который вам нужно вставить в процедуре между строками Sub и End Sub, чтобы сгенерировать ошибку. Конечно, если вы хотите «отловить» ошибку, вам нужно вставить строчку с On Error Goto до оператора Err.Raise, чтобы процедура выглядела так:

```
SubTest()
On Error GoTo ErrHandle
Err.Raise 13
Exit Sub
ErrHandle:
Msgbox Err.Number & vbCrLf & Err.Description
End Sub
```

Если вы ожидаете возникновения ошибки пользователя, для которой нет системной ошибки, то применяйте первый метод. Если вы ожидаете ошибку, для которой уже есть сообщение, такое, как сообщение о том, что объект не найден, тогда используйте второй метод. Если вы форсируете системную ошибку, вы делаете так, чтобы ваша процедура вела себя так, как будто ошибка и вправду возникла.

Что дальше?

Так как это книга непродвинутого уровня, то эта глава не является подробной, но она дает вам достаточно, чтобы понять и обрабатывать ошибки, если таковые возникнут. Следующая глава дает вам понятие о DAO и ADO. Вы познакомитесь с программированием обеих объектных моделей.

12
Основные инструменты, замечания и методы Visual Basic для Access

В гл. 11 вы научились фиксировать и обрабатывать ошибки в Access. В этой главе вы научитесь программировать, используя объектные модели DAO и ADO. Дано сравнение этих двух объектных моделей с различных точек зрения. В частности, вы узнаете:

- как использовать объектные модели для запуска запросов;
- как определить лучший способ управления наборами записей;
- как оценить критерий производительности для определения наиболее подходящей для использования объектной модели;
- как использовать коллекции DAO и ADO и познакомитесь с примерами использования SQL совместно с VBA;
- как использовать переменные совместно с SQL.

Сравнение DAO и ADO

Хотя DAO и ADO обсуждались в предыдущей главе, в этой главе вы узнаете об особенностях программирования с использованием этих двух объектных моделей. По ходу вы овладеете рядом дополнительных инструментов и методов и получите некоторые советы. Мы будем основываться на материале гл. 7, в которой рассказано об объектах. Так как количество доступных вам вариантов велико, то рассмотрение некоторых проблем производительности будет проходить по порядку.

Учитывая то, что компьютеры становятся мощнее и быстрее с каждым днем, можно опрометчиво предположить, что такие вопросы, как дисковое пространство и скорость больше не являются проблемой. Однако даже в более мощных и быстрых системах работа с объемными наборами записей требует лучшей производительности программного обеспечения. Экономия нескольких секунд могла бы показаться не главным преимуществом. Но все эти секунды накапливаются на протяжении долгого пути, в то время как большие базы данных все увеличивают потребность в более быстром поиске и более эффективных запросах.

Имеют место также вопросы дискового пространства и быстродействия. Предположим, например, что вы решили сделать любое числовое поле полем с двойной точностью, вне зависимости от того, где эта двойная точность действительно требуется. Единственной проблемой будет являться пустая трата дискового пространства. Даже если у вас имеется огромное количество места на диске, то все равно это будет не по-умному. Как сказано в гл. 3, «даже если у вас есть избыток лишнего места, нерациональное расходование ресурсов - плохая привычка. Когда ваши базы данных чересчур разрастутся, то выполнение вычислений, запросы, а также их компоновка будут занимать больше времени. Даже если ограничения по объему вас не заботят, то должна заботить скорость».

Основным моментом всего этого является то, что всегда следует быть экономным, вне зависимости от того, за какой системой вы работаете. Это также относится и к программированию, ибо программирование позволяет вам создавать таблицы, изменять данные внутри их и получать доступ к наборам записей. Поэтому, вы всегда должны помнить о максимальной производительности, когда вы пишете процедуры.

Использование VBA для запуска запросов

вы можете запускать запросы из-под VBA множеством способов. Также вы можете исполнять строки и запросы SQL, которые уже хранятся в вашей базе данных, разными способами. В табл. 12.1 показаны основные методы запуска запросов, доступные вам.

Таблица 12.1. Способы запуска запросов из-под VBA

Макрокоманда, метод или объект	Объект запускается из	Источник	Пример
Макрокоманда unSQL	DoCmd	Строка SQL	docmd.runSQL strSQL
Макрокоманда OpenQuery	DoCmd	Заготовленный запрос	docmd. OpenQuery "qryTotals"
Метод Open Recordset (DAO)	Включение или база данных	Строчка SQL или заготовленный запрос	db. OpenRecordset strSQL
Метод Open (DAO)	Включение или набор записей	Строка SQL, заготовленный запрос или процедура	rs.Open"SELECT * FROM Customers, cn
Метод Execute_(DAO)	Включение или база данных	Строка SQL	db.execute strSQL
Метод Execute_(ADO)	Включение или команда	Строка SQL	cnn. execute strSQL
QueryDef (заготовленный запрос DAO)	Включение или база данных	Заготовленный запрос	db.QueryDefs ("qryTotals")

Управление наборами записей с использованием коллекций в сравнении с управлением наборами записей с использованием запросов

Обычными способами для извлечения наборов записей и управления ими с использованием Access VBA есть запросы и циклы по наборам записей. Хотя существует множество описаний обоих методов, все же для начинающего разработчика может быть затруднительным найти подробные инструкции по тому, когда и зачем использовать именно такой метод, а не какой-нибудь другой. Хотя в некоторых источниках и объясняется, что циклы через набор записей следует использовать для изменения свойств набора записей, а запросы нужно применять для изменения данных в таких наборах, это все же порождает больше вопросов, чем ответов. Например, вы можете менять свойства набора записей, используя запросы DDL (язык определения данных), и вы можете изменять данные, используя циклы.

Одним словом: имеет место частичное совпадение этих двух методов. Если этого все еще недостаточно, чтобы ваша голова закружилась, то у вас есть две различные объектные модели (ADO и DAO), которые можно использовать, чтобы управлять наборами данных. Итак, какой же метод следует использовать в конкретной ситуации?

Критерий производительности

Если вы уверены в том, что задача, которая поставлена уже до вас, может быть решена (как с помощью запросов, так и с помощью циклов) то производительность есть первый показатель, к которому следует прибегнуть. Вообще запросы быстрее, чем циклы. Это все потому, что оптимизатор запроса сохраняет лучший план исполнения запроса, после того как Jet в первый раз запускает этот запрос. Последующие запуски запроса используют этот сохраненный сценарий с целью оптимизации. Так как у циклов по наборам записей нет плана оптимизации, то их эффективный запуск есть запуск «в первый раз», каждый раз когда они исполняются. Следующие примеры, в которых выполняется то же самое вычисление над довольно большой таблицей, с полученными в них результатами доказывают это утверждение.

```
Sub ADOElapsed()
Dim Cnn As New ADODB.Connection
Dim rsADO As New ADODB.Recordset

BegTime = Time
Set Cnn = New ADODB.Connection
Set rsADO = New ADODB.Recordset
'Подставьте нужный диск и папку для файла источника данных (Data Source)
Cnn.Open "Provider= Microsoft.jet.oledb.4.0;" & _
 "Data Source =D:\Database\Access\Timing.mdb;"
rsADO.Open "tblAccount", Cnn, adOpenKeyset, adLockOptimistic
Do While Not rsADO.EOF
 rsADO! [Commission] = rsADO! [SellingPrice] * 0.07
 rsADO.Update
 rsADO.MoveNext
```

```
Loop
EndTime = Time
Interval = DateDiff("s", BegTime, EndTime)
MsgBox (Interval)
Set rsADO = Nothing
Set Cnn = Nothing
RsADO.Close
Cnn.Close
End Sub
```

Подключение ADO

Что значит Set Cnn = New ADODB.Connection? Так вы просто устанавливаете соединение с источником данных. Это демонстрирует преимущество ADO над DAO в плане гибкости. ADO является интерфейсом для OLE DB, новейшей технологии Microsoft для получения доступа к данным, который предоставляет доступ к гораздо более широкому диапазону источников данных. Это означает, что если вы хотите подключиться и получить доступ *к* данным *из* больших систем баз данных, таких, как SQL Server или Oracle, то ваш выбор - это ADO. Если вам нужен доступ к данным через Интернет/интранет, то и здесь ADO нужный инструмент. Теперь даже нереляционные, e-mail и CAD/CAM источники данных открыты для вас через ADO.

Вы также можете использовать оператор IN из SQL для подключения внешних источников. Больше об этом методе вы можете узнать в разделе «Импорт и экспорт данных» гл. 18.

Как было обещано, для сравнения вы можете просмотреть две подпрограммы, выполняющие одну и ту же задачу, использующие как ADO, так и DAO. Во всех трех примерах для синхронизации исполнения кода используется функция DateDiff. Аргумент "s" представляет собой секунды. Аргументы BegTime и EndTime были введены для определения интервала между двумя значениями времени так, как если бы мы использовали секундомер. А где эти переменные были декларированы? Функция EOF была использована для проверки на окончание таблицы, содержащей 68,586 записей. Она возвращает значение False, пока не дойдет до последней записи, когда она возвращает значение True.

Каков же был результат? Даже на довольно слабой машине процедура была выполнена за 119 с.

Следующий код написан с использованием DAO.

```
Sub DAOElapsed()
Dim db As Database
Dim rs As Recordset
BegTime = Time
Set db = CurrentDb
Set rs = db.OpenRecordset("tblAccount")
Do While Not rs.EOF
 rs.Edit
 rs![Commission] = rs![SellingPrice] * 0.07
 rs.Update
 rs.MoveNext
Loop
EndTime = Time
```

```
Interval = DateDiff("s", BegTime, EndTime)
MsgBox (Interval)
Set rs = Nothing
Set db = Nothing
End Sub
```

Этот код похож на предыдущий потому, что, как отмечено выше, объектная модель ADO основана на объектной модели DAO. И хотя циклы в обеих процедурах практически одинаковы, методы открытия набора записей совершенно разные. Эта процедура была завершена за 30 с. Она почти в четыре раза быстрее, чем подпрограмма ADO.

В следующем коде использован запрос для той задачи, решение которой мы пытались найти с помощью предыдущих процедур.

```
Option Compare Database
Dim BegTime As Date
Dim EndTime As Date
Dim Interval As Integer
Sub QryElapsed()
Dim db As Database
Set db = CurrentDb()
DoCmd.SetWarnings False
BegTime - Time
db.Execute "UPDATE tblAccount "
& "SET tblAccount.Commission = [SellingPrice]*0.07;"
EndTime = Time
Interval = DateDiff("s", BegTime, EndTime)
MsgBox (Interval)
DoCmd. SetWarnings True
End Sub
```

Заметьте, что в первых четырех строках до начала процедуры декларированы переменные. Так как эти переменные находятся в месте для декларации, то они доступны для других процедур этого модуля, который включает все процедуры, рассмотренные на настоящий момент. Существует ряд способов запуска запросов с использованием VBA. Например, макрокоманду DoCmd. RunSQL можно было бы использовать вместо оператора db. Execute. Если бы эта макрокоманда была использована, то оставшаяся часть процедуры осталась бы неизмененной и процедура была бы выполнена за 10 с; в три раза быстрее, чем процедура DAO. Однако приведенная процедура работает всего 7 с.

Макрокоманда SetWarnings

Для того чтобы правильно оценить запрос в нашем примере, нужно отключить предупреждения, так как на ответ на них потребуется несколько секунд. Как упомянуто выше, запросы действия предупреждают пользователя перед тем, как изменить данные в таблице. Не путайте запросы действия с макрокомандами в Access. К большинству макрокоманд доступ может быть получен через объект DoCmd. Если после DoCmd вы поставите оператор «точка», то, будучи в модуле, вы сможете увидеть список доступных операций. Строка DoCmd. SetWarning подключит предупреждения обратно.

Как видите, между запросами и циклами существует огромная разница в плане производительности. Но производительность является не единственным критерием, к которому следует прибегать. С помощью запросов вы не можете настраивать элементы управления. Существуют свойства конкретных объектов, которые невозможно изменить даже с помощью запроса DDL (язык определения данных).

Запрос DDL позволяет вам создавать таблицы или менять их структуру. И хотя эти запросы ограничены в возможностях, все же в случае надобности они могут быть удобны. К примеру, следующий DDL-код изменяет размер поля LastName с 50 до 55 символов:

```
Sub ModField()
Dim db As DAO.Database
Set db = CurrentDb()
db.Execute "ALTER TABLE tblCustomer " _
& "ALTER COLUMN LastName TEXT(55)"
End Sub
```

Больше информации о запросах DDL вы найдете в разделе «Управляющие запросы» гл. 17.

И хотя, используя этот метод, вы можете менять свойства таблицы, все же вы не можете менять свойства элементов управления на форме, используя этот или любой другой тип запроса. Как вы откроете для себя позже, некоторые задачи препятствуют применению запроса для управления наборами данных. Хотя всегда неплохо начинать с производительности, есть другие параметры определения соответствующего метода для решения конкретной задачи. В большинстве случаев если есть возможность использовать запрос, то используйте его. Скорее всего так будет быстрее.

Хотя DAO превзошел ADO в только что разобранных примерах, ADO превосходит DAO в случае баз данных, отличных от Jet. Все сводится к тому, что конкретно вы хотите сделать. Давайте поподробнее рассмотрим программирование с использованием этих двух моделей.

Коллекции и объекты DAO

Когда в 1992 году появилась DAO, то она работала исключительно с механизмом управления базами данных Jet. Начиная с Access 97, вдобавок к Jet, DAO (3.5) поддерживала ODBCDirect, открывая доступ к серверам предприятий, таким, как SQL Server. DAO организована в виде иерархии коллекций и объектов. Это означает, что все таблицы, поля, запросы и тому подобное представлены как объекты, собранные в коллекции. Например, коллекция отчетов объекта Application содержит все отчеты, которые открыты в базе данных Access на данный момент. Объект Application относится к активному приложению Access.

Представьте себе объекты как склады для коллекций, которые, в свою очередь, хранят другие объекты и так далее по иерархической лестнице. Например, в обеих объектных моделях вы можете получить доступ к объекту «поле» через коллекцию.

Объект DBEngine и коллекция «Рабочих сред» (Workspaces Collection)

Так как объект DBEngine содержит и управляет всеми объектами иерархии, то он представляет собой объект наивысшего уровня. В Access вам следует использовать функцию CurrentDb для получения доступа к объектам в текущей базе данных. Вместо того чтобы спускаться вниз по иерархической лестнице в поисках нужного вам объекта (для чего вам, скорее всего, придется «копаться» в папках нижележащих уровней), вы можете воспользоваться клавишной комбинацией для вызова этой функции и употреблять ее для обращения к базе данных, с которой вы работаете. При работе вне Access с использованием подключения ODBC (в этом случае вам следует учесть и ADO, так как она тоже работает с источниками данных ODBC) вам следует использовать объект DBEngine.

Коллекция рабочих сред объекта DBEngine содержит коллекции Databases, Users и Groups. Если бы вам потребовалось обращение к базам данных, не открытым в настоящий момент, то вы можете использовать эту коллекцию. Объект рабочей среды поддерживает объект подключения при работе с ODBCDirect. Коллекции Users и Groups используются в целях безопасности.

Объект базы данных

В DAO объект базы данных обращается к открытой на настоящий момент базе данных. Фактически вы можете иметь несколько в одно и то же время открытых баз данных, даже если они различны по своему типу. Несмотря на то что коллекция баз данных содержит все объекты базы данных для конкретной рабочей среды, объект базы данных содержит пять коллекций объектов в текущей базе данных. Эти коллекции описаны в табл. 12.2.

Таблица 12.2. Пять коллекций объекта Database

Коллекция	Описание
TableDefs	Обращается к таблицам в базе данных, но не содержит табличных данных
QueryDefs	Обращается к запросам в базе данных, но не содержит данных запроса
Containers	Коллекция объектов типа «контейнер», которая обращается к коллекции Documents, предназначенной для безопасности базы данных
Relations	Обращается к связям между таблицами в базе данных
Recordsets	Обращается к наборам записей из таблиц или запросов в базе данных. Эти наборы записей содержат структуру полей каждой таблицы или запроса. К данным каждого поля можно обращаться через его свойство Value (Значение).

Существует четыре способа обращения к объектам из коллекции, которые указаны в табл. 12.3.

Таблица 12.3. Способы обращения к объектам из коллекции

Синтаксис	Пример
Collection(O)	Fields(3)
Col 1 ection(выражение или переменная)	TableDefs(strTable)
Collection ("name")	QueryDefs("PaidCustomers")
Collection![name]	Fields![Last Name]

В первом методе обращение к объекту происходит через порядковый номер (также называемый индексом). Каждый объект во всех коллекциях DAO имеет встроенный номер, начинающийся с нуля. Это удобно для нумерации объектов в коллекции. Однако если вы знаете имя объекта, вы также можете сделать специальное обращение к объекту способом, показанным в третьем или четвертом примере. Используя второй способ, с синтаксисом *Collection(expression or variable),* вы можете обратится к объекту, применяя переменную:

```
Dim strName As String
strName = "Customers"
TableDefs(strName)
```

Вам не нужно использовать скобки (как в последнем способе) для обращения к объектам, если в имени объекта не встречается нестандартного символа, такого, как пробел. Если вы хотите получит доступ к свойству коллекции через номер, то вы можете сделать это так, как показано в следующем примере:

```
Debug.Print CurrentDb.TableDefs(0).Name
```

Каждый объект имеет как свойства, так и методы. DAO обеспечивает доступ к этим свойствам, которые определяют характеристики объекта (атрибуты), но обращение к ним должно быть сделано синтаксически правильно. Вы можете использовать объектные переменные для обозначения объектов в синтаксисе. Например, если вы хотите обратиться к свойству имени объекта querydef (обозначенного переменной qd), то вы можете прибегнуть к следующей синтаксической структуре:

```
strQueryName = qd.Name
```

Заметьте, что в предыдущем примере нет круглых скобок. Это все потому, что мы имеем дело с объектом, а не с коллекцией. В случае коллекций вы используете круглые скобки для того, чтобы обратиться к конкретному объекту из этой коллекции, за именем которого следует оператор «точка», если вы хотите обратиться к свойству этого объекта. Если это объект или объектная переменная, то просто используйте «точку» для обращения к свойству этого объекта.

Методы - это процедуры, которые производят действия над объектами. Эти процедуры отличаются от процедур, задаваемых пользователем, в плане того, что они привязаны к объектам и не могут быть вызваны независимо. Методы могут делать такие вещи, как закрытие или открытие таблицы или нахождение записи в таблице. Если вам угодно открыть таблицу в базе данных, вам следует использовать метод OpenRecordset, как показывают следующие строки кода:

```
Dim db as DAO.Database, rs as DAO.Recordset
Set db = CurrentDb()
Set rs = db.OpenRecordset("tblCustomers")
```

В отличие от других переменных объектные переменные в Access должны использовать ключевое слово set для обращения к реальному объекту или к новому объекту, образованному при помощи ключевого слова new. Как только это сделано, все свойства и методы объекта, к которым было сделано обращение, становятся доступными по переменной, как показывает следующий пример: `db.OpenRecordset("tblCustomers")`.

Если бы вы захотели перейти к следующей записи из набора записей, представленного переменной rs variable, то вам надо было бы использовать следующую строчку кода:

```
rs.MoveNext
```

Коллекции и объекты ADO

Объектная модель ADO структурно проще, чем объектные модели DAO или Visual Basic's RDO. Объект Connection находится на самом верху иерархической лестницы и олицетворяет подключение к источнику данных. Этот объект содержит объекты Command и Recordset, а также коллекцию Error (коллекция ошибок). В табл. 12.4 определены функции главных составляющих объектной модели ADO.

Таблица 12.4. Функции объектной модели ADO

Объект или коллекция	Функция	Ближайший эквивалент из DAO
Connection	Создает подключение к источнику данных	Database Object
Recordset	Создает набор данных, используя запрос	Recordset Object
Command	Исполняет строки SQL, запросы действия или заготовленные процедуры	QueryDefObject
Errors	В случае ошибки одна или более ошибок вставляются в эту коллекцию	Errors Collection

Объект Connection

Объект Connection в ADO представляет физическое подключение к хранилищу данных. Если вы хотите создать объект Connection, то вам просто следует указать имя хранилища данных ODBC или источника данных. Открытие объекта Connection есть попытка подключиться к желаемому хранилищу данных. Вы можете определить, была ли ваша попытка подключиться успешной, путем обращения к свойству State(Состояние) объекта Connection. В случае успешной попытки возвращается внутренняя (встроенная) константа adStateOpen, как показано в следующем примере, в котором для обозначения объекта подключения используется переменная с пп:

```
If cnn.State = adStateOpen Then
 MsgBox "You were successful"
Else
 MsgBox "Please try again"
End If
```

Объект Recordset

Объект Recordset из ADO представляет собой набор записей, извлеченных из запроса, с указателем на эти записи. Если вы хотите открыть объект Recordset без явного открытия объекта Connection, то вы можете указать путь подключения к методу Open объекта Recordset. Однако, создавая и открывая объект Connection, вы можете открыть несколько объектов Recordset, используя то же подключение. Хотя объект Recordset представляет собой весь набор записей, но в каждый момент времени объект обращается только к одной текущей записи из набора.

Так как объект Recordset имеет больше всего свойств и методов и так как ADO практически всегда использует объекты Recordset для управления данными, то можно сказать, что этот объект есть сердце и душа ADO. Каждый объект Recordset состоит из записей (строк) и полей (столбцов). Доступные методы и свойства Recordset зависят от источника данных.

Указатели и типы указателей

Указатель ADO помогает быстро просматривать набор данных, полученный в результате исполнения запроса. На самом деле указатель иногда называют набором записей. Набор записей можно прокручивать вперед-назад от текущего положения.

У указателей в ADO есть три назначения: 1) указатель определяет перемещение внутри набора данных и то, будут ли в наборе записей отражены изменения, внесенные пользователем; 2) указатель определяет место хранения набора записей, если указатель открыт; 3) запирающий тип указателя определяет, как хранилище данных ADO будет блокировать записи, когда они будут изменены.

В ADO представлено четыре типа указателей, которые описаны в табл. 12.5.

Таблица 12.5. Типы указателей в ADO

Тип указателя	Функция
Статический	Дает статическую копию набора записей, что делает возможным все типы перемещений в этом наборе данных. Не дает другим пользователям возможности просмотра добавлений, изменений и удалений
Динамический	Делает возможными все типы перемещений в наборе записей, которые не зависят от закладок, но все же разрешает закладки, если они поддерживаются источником. Делает возможным просмотр другими пользователями добавлений, изменений и удалений. Microsoft Jet OLE DB не поддерживает этот тип, но другие источники могут поддерживать его

Тип указателя	Функция
Символьный	Ведет себя как динамический указатель, за исключением того, что добавления и удаления не видны другим пользователям. Тем не менее изменения данных, сделанные другими пользователями, видимы
Однонаправленный	Этот тип установлен по умолчанию. С помощью его вы можете лишь передвигаться вперед по набору записей. Добавления, удаления и изменения будут невидимы для других пользователей. Это улучшает производительность в случае, если вам требуется лишь разовый проход по набору записей

Существует три способа открытия объекта Recordset используя ADO ADO:

- открывая набор записей, использовать методConnection.Execute();
- открывая набор записей, использовать методCommand.Execute();
- открывать объект набора записей без объектов Connection или Command, передавая верную строку Connect второму аргументу метода Recordset.Open().

Как вы можете видеть, процедура открытия набора записей довольно гибка. Посмотрите следующий пример, в котором при открытии формы происходит ее заполнение:

```
Sub Form_Open(Cancel As Integer)
 Dim cn As ADODB.Connection
 Dim rs As ADODB.Recordset
  ' Простой способ использования соединения OLEDB Microsoft Access с базой
  ' данных Jet
 Set cn = CurrentProject.Connection
 Set rs = New ADODB.Recordset
 With rs
.Source = "SELECT * FROM Customers"
.ActiveConnection = cn
.CursorType = adOpenKeyset
.LockType = adLockOptimistic
.Open
 End With
 Set Me.Recordset = rs
End Sub
```

Эта процедура уникальна, по крайней мере касательно одного аспекта. Обычно к концу процедуры вы закрываете набор записей операторами типа rs.Close и set rs = Nothing, но в этом случае событие закрытия формы берет на себя все эти профилактические процедуры, чтобы освободить память. Это все потому, что процедура делает набор записей формы набором записей ADO. Более того, если вы закрыли набор записей или подключение, то у формы не будет подключения, что нужно было бы для того, чтобы убедиться, что изменения в форме отображаются в исходной таблице.

Объект Command

Объект ADO Command может быть использован для выполнения запросов по базе данных и возвращения записей путем указания на запросы действия, строки кода SQL или заготовленные процедуры. Его целью является создание специальной команды, которую нужно применить к источнику данных. Объект DAO QueryDef похож на объект Command. Синтаксическая структура для возвращения записей будет следующая:

```
Set recordset = command.Execute(Записи, Параметры, Опции)
```

Как использовать DAO и ADO для управления данными

Хотя в этой главе приведено множество примеров программ, до сих пор не было предложено никаких практических советов и способов «на примере». Это было сделано нарочно, ибо вам необходимо понять основы перед тем, как применять их. С помощью следующего примера мы убьем сразу двух зайцев. Во-первых, он даст вам знание «из первых рук» того, как управлять данными используя обе объектные модели. Во-вторых, он познакомит вас с мощным методом, которой вы легко можете приспособить для ваших прямых нужд.

Необходимо подчеркнуть еще раз то, что нужно устанавливать ссылки на правильные библиотеки в окне модуля так, как объясняется в гл. 7. В этом случае (так как в следующем примере идет работа как с DAO, так и с ADO) должны быть установлены ссылки на обе библиотеки. Если обе библиотеки установлены, то убедитесь в том, что обращение в выражениях dim сделано к правильным объектам. Например, вы объявляете объект набора записей выражением:

```
Dim rs As DAO.Recordset
```

Допустим, что вы занимаетесь продажей фруктов и орехов. Вы делаете скидки при продаже более крупных партий продуктов. Вы хотите заполучить справочную таблицу типа Excel, в которой можно найти позицию записи, чтобы определить верную цену. Для начала вы подумываете о том, чтобы ваша таблица цен выглядела как табл. 12.6.

Таблица 12.6. Пример справочной таблицы

Продукт	Фунты	Цена
Земляные орехи	110	$1.95
Земляные орехи	11-50	$1.60
Земляные орехи	50+	$1.20
Орехи кешью	1-10	$4.30
Орехи кешью	11-50	$3.50
Орехи кешью	50+	$2.63

Но вдруг вы понимаете, что этот способ создает слишком много избыточности. Заметьте, что названия орехов повторяются при каждой расценке. Затем вам

приходит на ум устроить перекрестный запрос по этой таблице так, чтобы фунты были заголовками столбцов, и избыточность будет, таким образом, устранена. Однако теперь у вас появилась еще одна проблема. Как же работает процедура выбора нужного столбца для конкретной расценки? Способ с использованием поиска есть единственный выход.

1. Откройте базу данных FormsAndControls из папки AccessByExample.
2. Щелкните дважды по таблице LookNuts, чтобы открыть ее.
3. Обратите внимание на то, что эта таблица показана на рис. 12.1. Нажмите и, не отпуская, перетащите строку заголовка таблицы как можно левее; затем хорошенько выровняйте столбцы и уменьшите окно так, чтобы оно примерно подходило к левой половине экрана.

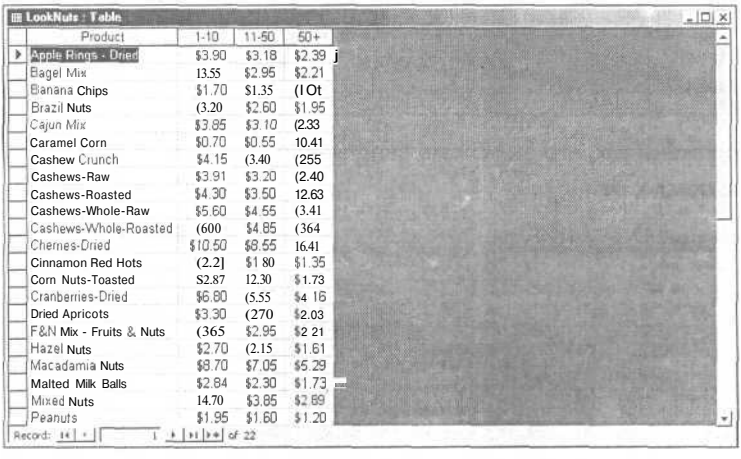


Рис. 12.1. Таблица LookNuts используется в процедурах поиска значений

4. Если есть возможность щелкнуть по окну базы данных, то щелкните. Если такой возможности нет, то нажмите F11, чтобы активизировать это окно.

ПОДСКАЗКА. Кнопка F11 прекрасно работает почти во всех местах Access, показывая окно базы данных. Исключением является окно модуля. В этом случае эту функцию выполняет комбинация клавиш Alt+F11. В отличие от F11, Alt+F11 работает как тумблер, поэтому вы можете переключиться обратно.

5. В меню Forms щелкните по форме FruitNutOrder. Вы должны увидеть три окна. На рис. 12.2 показано расположение этих окон.
6. Щелкните по окну таблицы, а затем щелкните по окну формы. Переместите окно формы направо, чтобы все выглядело так, как на рис. 12.2. Теперь вы готовы к проверке результатов.
7. Щелкните по селектору добавления записи (кнопка со звездочкой) внизу формы, как изображено на рис. 12.3, чтобы открыть новую запись.
8. Наберите 1119 в поле CUSTID. Наберите 2742 в поле OrderID.

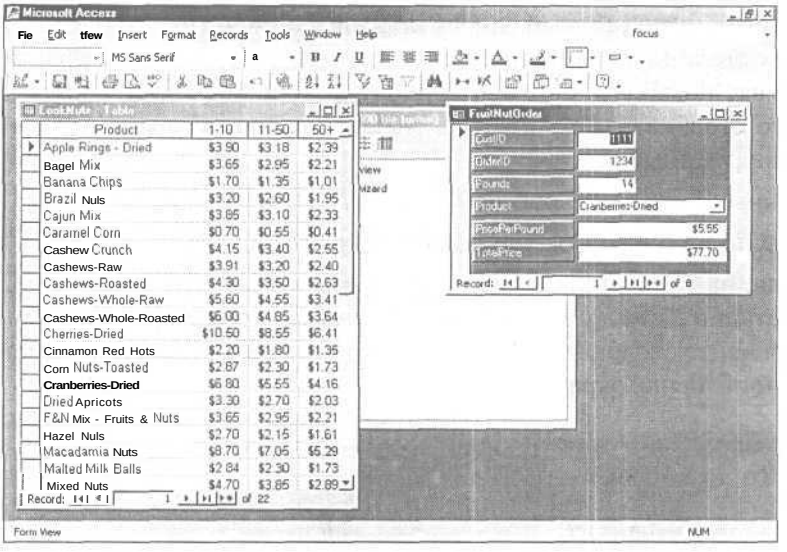


Рис. 12.2. Три окна расположены так, что вы можете видеть справочную таблицу в то время, когда форма обращается к ней

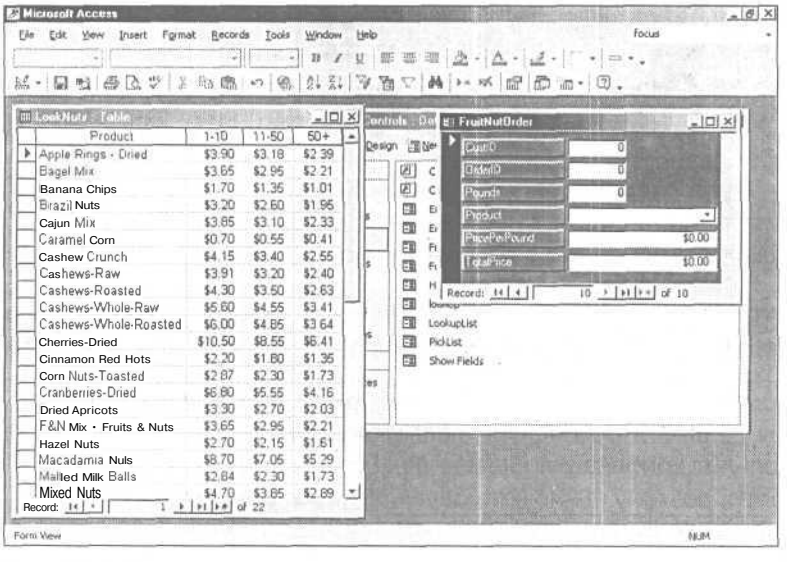


Рис. 12.3. Три окна расположены так, что вы можете видеть справочную таблицу в то время, когда форма обращается к ней

9, Наберите 50 в поле Pounds, а затем выберите Banana Chips (Банановые чипсы) из выпадающего списка Product(Продукт). Автоматически должно подставиться значение цены в 1.35 за фунт, как показано в таблице LookNuts, что слева. Заметьте, что в таблице расценка 1.35 верна для 50 фунтов. Также заметьте, что подсчитана итоговая цена - 67.50.

10. Попробуйте другие продукты по различным ценам и в разных количествах.
11. Останьтесь в базе данных.
12. Закройте таблицу LookNuts и форму FruitNutOrder

Вы, должно быть, заметили, что таблица устроена как результат перекрестного запроса. Этот метод DAO идеален для поиска. Где ваше практическое применение? Если вы страховой агент, то вы могли бы использовать этот метод для поиска ставок рабочих. Если вы банкир, то вы могли бы использовать этот метод для поиска ссудных процентов. Вот так.

Использование SQL внутри VBA

Если вы думаете, что такой метод поиска будет слишком трудно написать, то давайте посмотрим на тот код, который лежит в его основе. В следующем примере совмещено несколько программных операций, которые вы уже изучили, с довольно несложной процедурой.

1. В меню **Формы** (Forms) щелкните по FruitNutOrder, а затем щелкните по опции **Конструктор** (Design), чтобы открыть форму в конструкторе.
2. Щелкните по комбинированному окну Product и нажмите F4, чтобы открыть список свойств этого элемента управления.
3. Щелкните по ярлыку Event (Событие). Щелкните по окну After Update (После обновления), а затем нажмите кнопку Build (Создать), чтобы открыть окно модуля, содержащее процедуру, закрепленную за этим событием. Вы должны увидеть следующий код:

```
Dim db As DAO.Database
Dim rs As DAO.Recordset
Dim sQuote As String, dPPP As Double
Dim sFld As String, sCrit As String
sCrit = Me.Product 'Присваивает выбранное значение переменной
sQuote = Chr$(34)
Select Case Me.Pounds 'Находит нужное поле для поиска
 Case 1 To 10
   sFld = "[1-10]"
 Case 11 To 50
   sFld = "[11-50]"
 Case Is > 50
   sFld = "[50+]"
 Case Else
   MsgBox ("Please reenter pounds")
End Select
Set db = CurrentDb()
'Производит поиск, используя правильный вес в фунтах
Set rs = db.OpenRecordset("SELECT LookNuts.Product, " _
& "LookNuts." & sFld _
& "FROM LookNuts " _
.& "WHERE LookNuts.Product = " & sQuote & sCrit & sQuote & ";")
dPPP = rs.Fields(sFld)
Me.PricePerPound = dPPP
```

```
rs.Close
Set rs = Nothing
Set db = Nothing
End Sub
```

4. Останьтесь в базе данных и оставьте окно модуля с процедурой открытым, пока не прочитаете следующий подраздел. Закройте модуль, список свойств и форму.

Использование переменных с SQL

Процедура, которую вы только что просмотрели, имеет несколько интересных моментов. Например, заметьте, как используется метод OpenRecordset и строка кода SQL для запуска запроса. Так как имя поля может меняться, то в выражение SQL - с использованием оператором конкатенации (&) - подставляется переменная. Запрос в строке кода SQL использует переменную sFld, чтобы гарантировать, что для поиска выбрано верное имя поля. Выделенный столбец, в свою очередь, дает правильные расценки относительно числа фунтов, заказанных покупателем.

Также нужно обратить внимание на оператор Select. . Case, который является важной частью поиска, ибо он гарантирует, что переменной sFld присвоено верное имя поля. Переменная управляет условием отбора для поля Product, чтобы гарантировать, что запрос выбирает правильный продукт.

Запрос возвращает лишь одно значение, выбранное согласно правильно указанному для поиска продукту. Теперь все дело за выбором правильного поля (из двух) в записи, что отлично делается присваиванием dPPP = rs.Fields(sFld). В этом присваивании присутствуют три переменные. Первая - это dPPP (цена за фунт), которая присваивает значение полю, представленному переменной в переменной объектной ссылки rs (recordset - набор записей), т. е. в конце концов полю PricePerPound в форме с помощью выражения Me.PricePerPound = dPPP.

Короче говоря, процедура DAO осуществляет поиск по координатам, который отбирает координату(или «ячейку» для тех из вас, кто пользуется электронными таблицами), по которой пересекаются строка продуктов и столбец весов в фунтах. Помните, что вы почти также легко можете использовать переменную для представления имени таблицы в строке кода SQL. Это может быть мощным инструментом. Существует очень много способов применения SQL как в DAO, так и в ADO.

Результат тот же, а объектные модели разные

Следующий кусочек кода есть та же процедура, подогнанная под ADO. Главным различием является то, как получается доступ к набору записей. Программная логика, однако, остается той же.

```
Private Sub Product_AfterUpdate()
Dim cn As ADODB.Connection
Dim rs As ADODB.Recordset
Dim sQuote As String, dPPP As Double
```

```
Dim sFld As String, sCrit As String
sCrit = Me.Product 'Присвоить переменной выбранное значение
sQuote = Chr$(34)
Select Case Me.Pounds 'Найти правое поле для поиска
 Case 1 To 10
 sFld = "[1-10]"
 Case 11 To 50
 sFld = "[11-50]"
 Case Is > 50
 sFld = "[50+]"
 Case Else
 MsgBox ("Please reenter pounds")
End Select
Set cn = CurrentProject.Connection
Set rs = New ADODB.Recordset
 With rs
.Source = "SELECT LookNuts.Product, " _
 & "LookNuts." & sFld _
 & "FROM LookNuts " _
 & "WHERE LookNuts.Product = " & sQuote & sCrit & sQuote & ";"
.ActiveConnection = cn
.CursorType = adOpenKeyset
.LockType - adLockOptimistic
.Open
 End With
dPPP = rs.Fields(1)
Me.PricePerPound = dPPP
rs.Close
cn.close
Set rs = Nothing
Set cn = Nothing
End Sub
```

Заметьте, что оператор SQL использован совместно со свойством набора записей Source, чтобы результат получился тот же. Конструкция With использована для установления источника записей, открытия набора записей и установки других опций, таких, как CursorType. Вы можете испытать эту программу просто открыв форму ADO FruitNutOrder и используя такие же проверочные действия, которые применялись к форме FruitNutOrder.

Как использовать DAO и ADO для нумерации объектов

Могут возникнуть случаи, когда вам необходимо занумеровать объекты. Почему это важно? Предположим, что вы захотели показать пользователям динамический список отчетов, из которого можно было бы делать выборки для печати. Или предположим, что вы захотели, чтобы пользователь делал выборки из списка таблиц для импортирования или экспортирования. Вы можете скомбинировать код, представленный в этой главе, с методами «списка для выбора», которые вы изучили для разработки творческих приложений для пользователей. На самом деле в гл. 18 об этом и рассказано.

Существует множество способов нумерации объектов в Access. Один из методов и вовсе не требует программирования. Вы можете просто создать запрос. Фактически все уже сделано за вас. Запрос имеет смысл, если вы понимаете, к каким объектам обращается запрос. Вы можете посмотреть на системную таблицу, называемую MsysObjects, в окне базы данных, просто щелкнув по меню Tools, Options, View, которое находится в строке меню Access, и выбрав как скрытые объекты, так и системные объекты. На рис. 12.4 показано меню View.

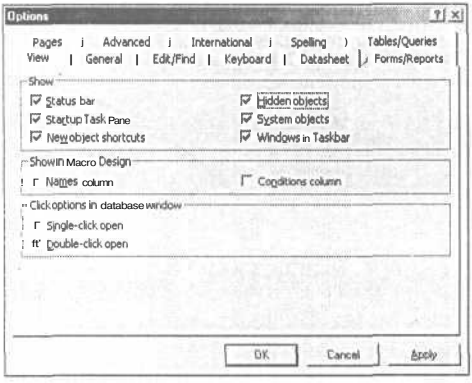


Рис. 12.4. Выберите Tools, Options, View, чтобы выбрать опции Hidden Objects (Скрытые объекты) и System Objects (Системные объекты), чтобы можно было видеть скрытые и системные объекты

Если обе эти опции выбраны, то вы сможете увидеть таблицы в окне базы данных, которые обычно не видны. Когда вы поставите галочки в этих полях, нажмите App1y(Применить) и потом OK. Дважды щелкните по таблице MSysObjects и делайте прокрутку вправо, пока не увидите поля Name и Type. Поле имени выдает список всех пользовательских объектов в окне базы данных. Поля типа относится к типу объекта в столбце Name.

На рис. 12.5 изображена сетка проекта QBE, которая обращается к таблице MSysObjects.

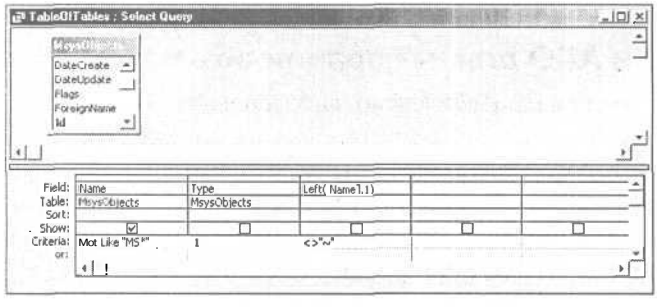


Рис. 12.5. Выставив значение поля Type, равное единице, в таблице MSysObjects, вы извлечете все таблицы базы данных

Создание списка объектов

К полям Name и Type делается обращение через запрос TableOfTables. Существуют ключи для установки параметров запроса, который обращается к объектам базы данных. Тип желаемого вами списка объектов определяется полем Type, а сам объект, конечно же, определяется полем Name. Поле Name также является тем местом, в котором исключаются системные объекты путем использования подстановочных знаков в условии отбора.

Сделайте обратные изменения в меню Tools, Options, View, а затем выполните следующие шаги, чтобы просмотреть и запустить запрос:

1. Откройте в конструкторе запрос TableOfTables.
2. Щелкните по выпадающему меню кнопки View button и переключитесь на конструктор. Заметьте, что в условии отбора столбца Name с помощью условия отбора Not Like MS* исключены системные таблицы. Заметьте также, что для таблицы тип объекта установлен в «1».
3. Находясь в представлении QBE, нажмите кнопку Run, чтобы извлечь таблицы базы данных, как показано на рис. 12.6.

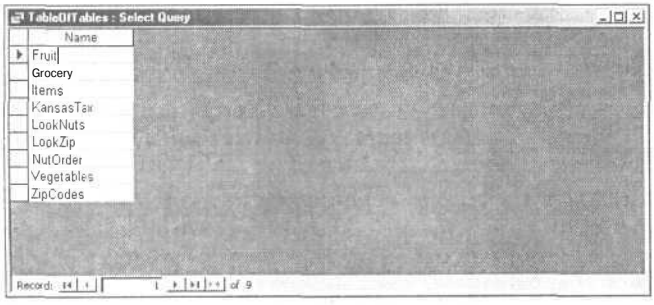


Рис. 12.6. Вы можете создать запрос, который обращается к системным объектам, таким, как таблицы и запросы, чтобы пронумеровать их

Заметьте, что все таблицы отображаются динамически, что делает запрос постоянно активным. Это означает, что если вы добавите таблицу и перезапустите запрос, то помимо других таблиц появится и новая.

4. Вернитесь в QBE и поменяйте значение 1 в строке условия отбора столбца Type на значение 5. Перезапустите запрос и заметьте, что все выводится список всех запросов базы данных.
5. Выполните п. 4, используя значения Perform - 32768 в столбце Type для форм и -32764 для отчетов, а затем поменяйте условие отбора столбца Type обратно на 1.
6. Выберите View, SQL, чтобы открыть представление в виде SQL. Выделите (если еще не выделен) весь код SQL и нажмите Ctrl+C, чтобы скопировать код в буфер.

7. Закройте запрос без сохранения. Теперь вы можете вставить скопированную строчку кода SQL в форму.
8. В меню **Формы** (Forms) дважды щелкните по форме EnumerationForm, чтобы открыть ее.
9. Нажмите на кнопку Enumerate Tables (Пронумеровать таблицы) и пробегитесь по таблицам в базе данных. Также нажмите кнопку Enumerate Queries (Пронумеровать запросы). Это один из способов пронумеровать объекты. Вскоре мы посмотрим на код, что лежит в основе действия этих кнопок.
10. Нажмите кнопку **Вид** (View), чтобы зайти в конструктор, а затем правой кнопкой мыши щелкните по верхнему списковому окну и выберите опцию Properties (Свойства).
11. Щелкните по ярлыку Data, который содержит свойство Row Source (Код для строки).
12. Щелкните по полю свойства Row Source (Код для строки) и нажмите Ctrl+V, чтобы вставить скопированный на шестом шаге код (рис. 12.7).

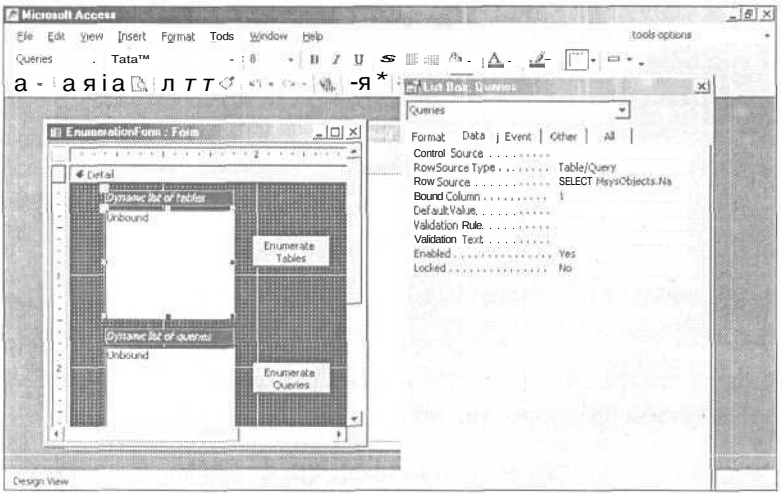


Рис. 12.7. Вставка строчки кода SQL в поле свойства Row Source (Код для строки) создает динамическую копию объектов вашей базы данных

Не беспокойтесь о возвратах каретки в строке кода SQL. В окне вы будете видеть только одну строку кода за шаг, но вы легко можете щелкнуть в любом месте окна и нажать кнопку «стрелка вниз», чтобы увидеть остаток строки; или щелкните правой кнопкой мыши по окну и выберите опцию увеличения (zoom), чтобы видеть всю строку.

13. Щелкните по второму списковому окну, помеченному как Dynamic List of Queries (Динамический список запросов).
14. Вставьте все тот же код, который вы вставляли на шаге 12 в окно Row Source, а затем нажмите кнопку Build.

15. В конструкторе QBE поменяйте значение условия отбора столбца Type с 1 на 5, чтобы заставить запрос извлекать объекты запроса. Нажмите Close (Закрыть) и нажмите Yes, чтобы сохранить изменения в выражении SQL. Закройте окно свойств.
16. Нажмите кнопку Вид (View) для сохранения и откройте форму.

Заметьте, что два запроса, которые являются кодами для строк в двух списковых окнах, идентичны за исключением условия отбора для поля Type. Как было упомянуто ранее, эти списковые окна всегда будут отражать текущее состояние базы данных. Следующие шаги показывают на примере, как работают эти списковые окна.

1. Посмотрите на оба этих списковых окна в форме и заметьте, что в них приведен список всех таблиц и запросов базы данных.
2. Снова щелкните по конструктору, чтобы открыть форму в нем. Правой кнопкой мыши щелкните по кнопке Enumerate Tables (Пронумеровать таблицы), а затем выберите опцию Build Event (Создать событие), чтобы открыть процедуру. Вы должны увидеть следующий код:

```
Option Compare Database
Dim db As Database
Private Sub Command4_Click()
Dim td As TableDef
Set db = CurrentDb
 For Each td In db.TableDefs
 If Left(td.Name, 2) <> "MS" And _
 Left(td.Name, 1) <> "~" Then 'Исключает системные объекты
 MsgBox td.Name
 End If
 Next
Set db = Nothing
Set td = Nothing
End Sub
Private Sub Command5_Click()
Dim qd As QueryDef
Set db = CurrentDb
 For Each qd In db.QueryDefs
 If Left(qd.Name, 2) <> "MS" And _
 Left(qd.Name, 1) <> "~" Then 'Исключает системные объекты
 MsgBox qd.Name
 End If
 Next
Set db = Nothing
Set qd = Nothing
End Sub
```

Так как обеим процедурам нужна переменная db Object, то она была декларирована в месте для деклараций модуля, содержащем эти процедуры. Заметьте, что во второй процедуре переменная db не декларирована. Только потому, что запрос устраняет все объекты, которые начинаются с "MS" или "~", обе процедуры делают то же самое, используя оператор If. Обе процедуры довольно

просты. Вы могли бы просто пронумеровать элементы в списковом окне, но тогда бы вы не знали, как сделать то, что делает запрос в DAO. Единственное отличие в этих двух процедурах есть использование либо объекта TableDef, либо объекта QueryDef.

1. Закройте первую форму нумерации. Дважды щелкните по форме EnumerationForm версии ADO, чтобы открыть ее в виде формы.
2. Вставьте код SQL из TableOfTables в оба поля Row Source (Источник строки) обоих списковых окон в соответствии с указаниями. Убедитесь в том, что после вставки кода вы поменяли значение поля Type во втором окне.
3. Нажмите кнопки Enumerate Tables и Enumerate Queries, чтобы упорядочить таблицы и запросы базы данных, используя объектную модель ADO.
4. Теперь сами (не подглядывая в инструкции) зайдите в окно с кодом, который лежит в основе работы этих кнопок. Вы должны увидеть следующий код:

```
Private Sub EnumQueries_Click()
Dim obj As AccessObject, dbs As Object
Set dbs = Application.CurrentData 'Измененная версия из подсказок от
Microsoft
  'Ищет запросы AccessObject во всех коллекциях AllQueries.
 For Each obj In dbs.AllQueries
  'Печатает имя объекта.
 MsgBox obj.Name
 Next obj
End Sub
Private Sub EnumTables_Click()
Dim cn As ADODB.Connection
Dim cat As New ADOX.Catalog 'Must set ADO Ext. 2.5 for DDL
Dim tbl As Table
Set cn = CurrentProject.Connection
Set cat.ActiveConnection = cn
For Each tbl In cat.Tables
 If Left(tbl.Name, 2) <> "MS" And _
 Left(tbl.Name, 1) <> "~" And _
 tbl.Type = "TABLE" Then 'Исключает просмотр объектов.
 MsgBox tbl.Name
 End If
 Next
Set cn = Nothing
Set cat = Nothing
End Sub
```

Первая программа из файла помощи от Access приведена потому, что она просто нумерует объекты запроса Access, используя объект AccessObject из коллекции AllQueries. Вторая программа, использующая объектную модель ADO, требует, чтобы ссылка на библиотеку (обратите внимание на комментарии) была указана так, чтобы обращаться к каталогу ADOX. Объект Catalog представляет всю базу данных, так как он содержит ссылки на объекты, такие, как таблицы, окна представления и заготовленные процедуры. Короче говоря, он содержит все элементы базы данных.

Создание списка полей

Списковые окна, с которыми вы только что имели дело, работали как списки для выбора. А что если вам требуется список полей для выбора для каждой таблицы из вашей базы данных? вы могли бы пронумеровать коллекцию полей практически таким же способом, каким вы нумеровали коллекцию таблиц. Однако есть гораздо более легкий заготовленный способ для нумерации полей любой часто просматриваемой таблицы. Вы можете использовать тип кода для строки FieldList спискового поля, чтобы вывести список всех полей из выбранных таблиц с любым кодом для строк. Вы можете воспользоваться этим вариантом, чтобы отображать в виде списка поля любой таблицы вашей базы данных одним лишь щелчком мыши. Просто распечатайте форму для твердой копии вашего списка. Просмотрите следующую процедуру, которая выдает список полей любой таблицы вашей базы данных, по которой вы щелкнули кнопкой мыши.

```
Private Sub lstTables_Click()
Dim iTableIndex As Integer
Dim sTableName As String
iTableIndex = Me.lstTables.ListIndex
sTableName = Me.lstTables.ItemData(iTableIndex)
Me.lstFieldList.RowSource = sTableName
End Sub
```

FieldList есть имя спискового окна, что справа. Tables есть имя спискового окна, что слева. Эти имена являются лишь индексами,нужными для аргумента ItemData. Далее вы просто приписываете таблицу к коду для строк спискового окна FieldList. Затем выставьте тип кода для FieldList. В этом примере это уже сделано за вас. Выполните следующие шаги, чтобы посмотреть все это в действии.

1. Откройте базу данных FormAndControls. В меню **Формы** (Forms) щелкните дважды по форме Show Fields (Показать поля), чтобы открыть ее.
2. Щелкните по нескольким таблицам и обратите внимание на то, что список полей, что справа, изменился таким образом, что теперь в нем присутствуют копии полей таблицы, что слева.
3. Закройте форму Show Fields; затем щелкните по ней правой кнопкой мыши и выберите опцию Copy (Копировать).
4. Откройте базу данных Northwind из папки AccessByExample, что находится на вашем жестком диске.
5. Нажмите Ctrl+V, чтобы вставить форму, а затем дайте форме имя Show Fields.
6. Дважды щелкните по форме Show Fields, чтобы открыть ее в базе данных Northwind, как показано на рис. 12.8. В этих таблицах присутствует больше полей, которыми может быть заполнено списковое окно. Вы можете скопировать форму в любую созданную вами в Access 2002 или Access 2000 базу данных, и она должна функционировать как обычно.

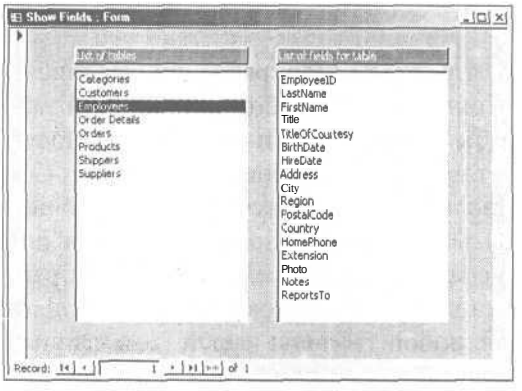


Рис. 12.8. Форма Show Fields пригодна для любой базы данных

Теперь вы знаете, как можно управлять объектами и нумеровать их как в ADO, так и в DAO. Вы получили несколько советов и познакомились с некоторыми методами использования встроенных функций и свойств Access. Вы также узнали о том, как оценить производительность с помощью таймера.

Что дальше?

вы пополняете свой набор инструментов и методов, который вы сможете использовать для улучшения ваших приложений. Эти знания также будут полезны при решении ежедневных проблем, с которыми встречаются разработчики. Следующая глава посвящена разработке и дизайну Web-приложений. Вы узнаете о нескольких форматах Web-страничек и о том, как преобразовывать объекты вWeb-странички.

Часть IV
Использование новейших возможностей Access

13

Опубликование базы данных в Access

Проще говоря, World Wide Web (WWW) есть не что иное, как сеть всех сетей. Всемирная сеть, которая была названа Интернетом, в корне преобразовала наш бизнес и доступ к информации. Внутренние сети (intranet) - это, по существу, внутренние сети компаний или частные сети. Базы данных - это гигантские хранилища электронной информации, именно поэтому базы данных и Интернет должны быть тесно связаны между собой.

Базы данных Access 2002 содержат много удобных пользователю возможностей с целью облегчения доступа к информации. Большинство усовершенствований были сделаны с расчетом на интеграцию с Web-технологиями.

В гл. 12 вы узнали, как программировать используя DAO (Data Access Objects) и ADO (ActiveX Data Objects). Эта глава посвятит вас в сферу интеграции Access и Интернета. В частности:

- вы узнаете, как конвертировать объекты в Web-страницы;
- как создавать страницы доступа к данным;
- познакомитесь с HTML, XML и другими Web-форматами;
- страницами доступа кданным(Data Access Pages);
- активными серверными страницами (Active Server Pages).

Конвертирование объектов в Web-страницы

Web-страница нужна для представления информации в Интернете и является средством обращения к информации. Она содержит набор команд, так называемые *теги,* с помощью которых текст и графика форматируются и располагаются должным образом. Гиперссылки - это теги, которые соединяют с другими Web-страницами, обеспечивая, таким образом, основной способ навигации в Интернете. Они работают как указатели на другой объект. Причем указываемый объект не обязательно является Web-страницей. Гиперссылка может указывать на графический файл, почтовый адрес(e-mail), файл (например, документ Microsoft Office) или программу. Например, гиперссылки можно применять для переключения между документами Office.

HTML и XML

Web-страницу можно создать используя Hypertext Markup Language, или сокращенно HTML. Впрочем, в Access 2002 есть средства, необходимые для прямого преобразования объектов в документ HTML таким образом, чтобы они были доступны в Web. Все, что вам нужно, - это Web-браузер, например Internet Explorer или Netscape Navigator.

Вследствие различия формата данных в приложениях нелегко добиться совместимости с Интернетом. Несмотря на то что HTML хорошо справляется с обеспечением визуальной текстовой и графической информации для Web-браузеров, его возможности ограничены при определении структур данных. Вам следует обратиться к языку XML: задать спецификацию, форму, стандарт, и эти действия всегда будут корректными. Это упрощение стандартного обобщенного языка разметки (SGML) было специально разработано для Web-документов.

XML - структура взаимного обмена данными, предназначенная для создания, доставки, интерпретации, обоснования и обработки данных в Интернете. Взаимный обмен данных означает, что разработчик может обмениваться данными с разнородными приложениями. Существует два класса XML-приложений: обмен данными и издательская система. В отличие от HTML, который описывает внешний вид Web-страницы, XML характеризует структуру данных Web-страницы. Если HTML - стандартный язык для создания и изображения Web-страниц, то XML - стандартный язык для описания и доставки данных в Интернете.

На первый взгляд XML похож на HTML, потому что они оба основаны на SGML. Однако синтаксис XML отличен от синтаксиса HTML. Он более строгий и не имеет предопределенных тегов. По мере роста HTML становился все более и более многосложным вследствие появления новых тегов. В отличие от HTML, XML позволяет пользователю создавать новые теги. Хотя HTML и пользуется успехом, XML был создан для указания на его недостатки, которые оцениваются некоторыми авторами как значительные.

Что касается вышесказанного, HTML не собирается исчезать в ближайшем будущем. Первый рабочий проект версии HTML с введенными XML-технологиями (также известный как XHTML 1.0), был выпущен в феврале 1999 года. XHTML был своеобразной попыткой объединить XML и HTML.

Access и XML

Access 2002 предоставляет набор инструментов для импорта и экспорта XML-данных; эта возможность недоступна в Access 2000. Это значит, что с помощью версии 2002 пользователи могут быстро публиковать формы, отчеты, запросы или таблицы для Web, используя XML/XSL. Привязанный XSL-файл (XSL - открытый язык стилей) используется для презентации объектов, позволяя пользователям просматривать формы и отчеты, созданные в Access с помощью любого браузера, поддерживающего HTML 4.0. В процессе экспорта файла доступна опция создания XSD-файла, который содержит схему информации, привязанной к XML-файлу базы данных. Табл. 13.1 послужит удобным справочником для интернетовских акронимов.

Таблица 13.1. Важные акронимы в Интернете

Акроним	Означает
CSS	Cascading Style Sheet (Каскадные таблицы стилей)
DTD	Document Type Definition (Описание шаблона документа)
HTML	Hypertext Markup Language (Язык гипертекстовой разметки)
HTTP	Hypertext Transfer Protocol (Протокол передачи гипертекстовых файлов)
ISO	International Standard Organization (Международная организация по стандартизации)
SGML	Standard Generalized Markup Language (Стандартный обобщенный язык разметки)
TCP/IP	Transmission Control Protocol/Internet Protocol (Протокол управления передачей/протокол Internet)
URL	Universal Resource Locator (Унифицированный указатель информационного ресурса)
W3C	World Wide Web Consortium (WWW-консорциум)
WWW	World Wide Web (Всемирная сеть, или Интернет)
XHTML	Extensible Hypertext Markup Language (Расширяемый язык гипертекстовой разметки)
XML	XML Extensible Markup Language (Расширяемый язык разметки)
XQL	XML Query Language (язык XML-запросов)
XSD	XML Schema Definition (Определение схемы XML)
XSL	XML Stylesheet Language (Открытый язык стилей)
XSLT	Extensible Stylesheet Language Transformation (Язык преобразования расширяемых стилей)

Рассмотрим более подробно новую опцию экспорта в XML на примере.

1. Откройте базу данных Music Store из папки AccessByExample и в меню **Таблицы** (Tables) щелкните по таблице Customers.
2. Выберите **Файл** (File), **Экспорт** (Export), из главного меню.
3. Выберите директорию, куда вы хотите сохранить ваши файлы; убедитесь, что вы указали именно нужную вам директорию. Щелкните по ниспадающему меню раздела Save As Type (Сохранить как), далее выберите тип XML как типсохраняемогофайла.
4. Щелкните Export для того, чтобы открыть диалоговое окно Export XML, как показано на рис. 13.1. Обратите внимание на три опции этого диалогового окна.
5. Отметьте все три опции и щелкните Advanced (Подробнее) для Export XML диалогового окна. Щелкните по закладке Schema (Схема). Обратите внимание, что на рис. 13.2 стоит отметка, что будет создана отдельная схема документа customers.xsd.

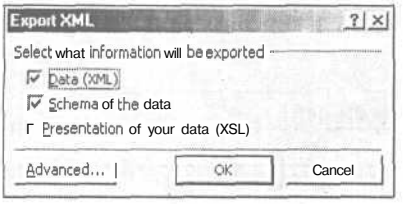


Рис. 13.1. Опция экспорта в диалоговом окне Export XML

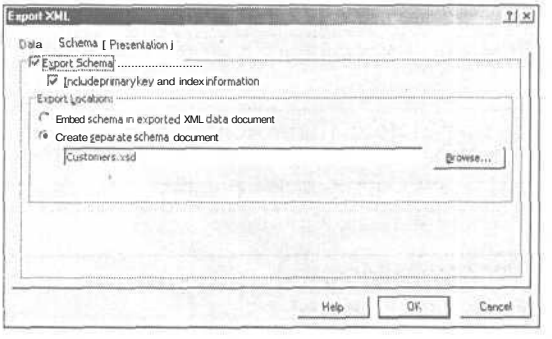


Рис. 13.2. Закладка Schema, расположенная в диалоговом окне Export XML, позволяет создавать отдельную схему документа

ЗАМЕЧАНИЕ. Когда вы отмечаете все три опции, вы создаете четыре отдельных файла, это можно изменить только путем предварительной установки опций диалогового окна Export XML. Два типа файлов, XML и XSL, появляются в закладке Schema. Тип файла HTML появляется в закладке Presentation (Презентация).

Для дополнительной информации о типе файла Schema (XSD) в закладке Schema щелкните по кнопке Help для открытия тематического раздела справки About XML Data и Access. Выберите подраздел What are XML Schemas. Там также есть информация о типе файла XSL в разделе Extensible Stylesheet Language Transformation.

6. Щелкните по закладке Presentation (Презентация). Обратите внимание на опцию, которая позволяет экспортировать презентации с клиента или сервера, как показано на рис. 13.3.

ЗАМЕЧАНИЕ. Если вы выберете опцию Run from: Server, вы создадите активную серверную страницу (ASP). Это популярная технология создания динамических Web-сайтов. Вы можете отметить, что это активная серверная страница, указав расширения файла.asp. Вы можете увидеть это расширение во время работы в Web-сайтах.

7. Щелкните OK для создания четырех файлов.

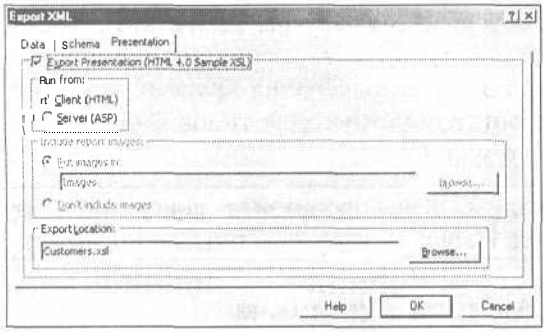


Рис. 13.3. У вас есть возможность экспортировать презентации клиента или сервера

Теперь, когда страница создана, просмотрим ее в браузере для проверки.

1. Откройте Windows Explorer, зайдите в директорию, в которую вы записали файлы базы данных Music Store, следуя предыдущим шагам.

ЗАМЕЧАНИЕ. Операционная система Windows должна быть настроена таким образом, чтобы можно было распознать расширения файлов. Когда вы открываете файл с расширением HTML или XML, файл автоматически откроется в вашем браузере.

2. Убедитесь в том, что опция просмотра типов файлов в окнах Windows активизирована. Дважды щелкните по файлу с расширением.html. Вы увидите изображение таблицы Customers. Обратите внимание, что файл открыт в Internet Explorer.
3. Попытайтесь редактировать данные в любом поле. Заметьте, что вам не удается что-либо изменить. Закройте HTML-файл.
4. Сделайте двойной щелчок по Customers file с расширением.xml. Вы должны увидеть следующий листинг (покажется только первоначальный код):

```
   <?xml version="1.0" encoding="UTF-8" ?>
- <dataroot xmlns:od="urn:schemas-microsoft-com:officedata"
xmlns:xsi="http://www.w3.org/2000/10/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="Customers.xsd">
- <Customers>
 <CustomerID>101</CustomerID>
 <SortName>Adams</SortName>
 <Customer>Adams High School</Customer>
 <Address>1443 S. Lewis</Address>
 <City>Tulsa</City>
 <State>OK</State>
 <Zip>74136</Zip>
 </Customers>
```

5. Обратите внимание, что листинг представлен в виде схемы файла Customers.xsd. Закройте окно браузера и Window Explorer, но оставьте открытой базу данных Music Store.

Это упраженение покажет вам некоторые опции, доступные в меню File, Export, предназначенные для создания XML-файла. Объяснять разницу между

файлами - это одно, а разобраться в их содержимом - другое. Теги в XML-файле, который вы только что рассмотрели, такие, как <State>, используются для разграничения данных, но в то же время для их истолкования другими приложениями, которые читают файл. Полное описание структуры типов файлов, описываемых в учебнике, не поместится в этой книге.

ЗАМЕЧАНИЕ. Для детального анализа XML прочитайте книгу «XML by Example «, автор Beniot Marchal (Бенуа Маршал), издательство Que Publishing.

Знакомство со страницами доступа к данным (Data Access Pages)

Data Access Page (Страница доступа к данным) - Web-страница, хранящаяся за пределами Access, которую пользователь может открыть при помощи web-браузера или обновить в Access, так как пользователь имеет доступ к базе данных. Для форм и отчетов этот файл сохраняется на языке, основанном на XML, так называемом ReportML, который снабжает данными представления так же, как и моделями, данных, для создания Data Access Page.

Web-страницы бывают статическими (страницы, которае созданы заранее и хранятся для последующей отправки клиентам) и динамическими (страницы, которые создаются только по запросу клиента для выдачи ему самой последней информации и ответов на запросы к базам данных). Статическая Web-страница отображает только первоначальное состояние той базы данных Access, из которой и была создана эта страница. Иначе говоря, никакие последующие изменения не отображаются на стр.. В отличие от статической страницы динамическая страница автоматически обновляется каждый раз при просмотре страницы.

Страницы доступа к данным - динамические. Вы можете просматривать, искать, изменять и добавлять данные в форме, даже если вы отключены от сети, используя Internet Explorer 5.0 или более поздние версии. Это дает целый ряд возможностей разработчику, вашим коллегам, пользователям. Однако имейте в виду, что вы можете не позволять абсолютно всем просматривать вашу информацию. Именно поэтому Data Access Page удобна для внутренней сети, в которой защита информации более важна, чем в Интернете.

Перед тем как вникать в строение Data Access Page, примите во внимание следующее.

- Одно лишь создание Data Access Page не означает, что она будет доступной в Интернете, но только до тех пор, пока не будут выполнены надлежащие инструкции для публикации страницы, что означает возможность просмотра и обращения с этой страницей в Internet Explorer.
- Вы можете работать с Data Access Page в Windows Explorer и использовать ее так же, как если бы она была формой Access. Это делает доступным вашу формулюдям внутри вашей организации, которые не имеют копии программы Access. Даже не обязательно открывать Access, чтобы получить доступ к странице и делать в ней изменения. Как только страница окажется доступной в сети, ее можно будет открыть с помощью Internet Explorer 5.0 (EE 5.0) или

более поздней версии. После внесения новых данных вы можете удостовериться в сделанном добавлении (или изменении), просмотрев страницу в том объекте Access, который был использован для создания ее.

- Создание Data Access Page не означает, что она автоматически станет доступной через Интернет, даже если у вас имеется Web-сервер, потому что страницы доступа к данным разработаны для IE 5 или более поздней версии. Вследствие того что эти браузеры содержат специфические Microsoft ActiveXэлементы управления, вы не можете просматривать Data Access Pages в других браузерах, таких, как Netscape navigator, без плагина (plug-in) - программы, которая расширяет возможности браузера. В частных сетях, таких, как интранет, вы можете удостовериться, что каждый работает в Ш 5 или в более поздней версии IE. В основном совместно используемые данные открыты для доступа так же, как и в Интернете.
- Вследствие того что страницы хранятся вне Access, нажимая на иконки на стр. в Access, вы только получаете доступ к ярлыкам файлов, находящихся на вашем жестком диске. Это означает, что любые изменения конфигурации страницы сохраняются на жестком диске, вне базы данных. Однако динамическое изменение данных отразится на базе данных. При попытке удалить страницу появится сообщение-вопрос: хотите ли вы удалить соединение в сети и файлы или только соединение? Если удалить HTML-файл с жесткого диска, в Access останется ярлык, ни на что не указывающий.
- Конвертировать объект базы данных предпочтительнее, чем создать страницу из памяти или из мастера. Даже если Data Access Page содержит в себе большинство выполняемых функций первоначального объекта, который был конвертирован в эту страницу, не следует ожидать, что VBA- код (Visual Basic для прикладных программ) будет работать в Explorer. Есть несколько различий между страницами и объектами базы данных, из которых создаются страницы, к которым будут обращаться.
- Так как страница использует динамический HTML, в целом целесообразен доступ к базе данных в среде клиент-сервер. (См. подробное обсуждение Active Server Page, посвященное проблемам клиента-сервера ниже в этой главе.)

Создание страницы базы данных Access

Всего лишь несколько лет назад, создание динамической, или «живой», Web-страницы, соединенной с данными в базе данных, даже в Microsoft Access было непростой задачей. Страницы базы данных Access весьма изменились в отношении оптимизации этого процесса. Простое помещение формы в Интернет - далеко не все, что может предложить Access сейчас. Появился абсолютно новый способ интерактивного взаимодействия с «живыми» данными не только из офиса, но также из любого места во всем мире.

Существует множество способов создания страницы базы данных Access. Так какой же из них является наилучшим? Это зависит от того, что вы хотите создать. Если вы хотите создать страницу, которая выглядит как только что создан-

ный вами объект (например, форма), просто выделите нажатием клавиши мыши объект и в меню выберите **Файл** (File), **Сохранить как** (Save As). Этот способ подойдет, если вы хотите опубликовать адресную форму.

Другие способы также следует рассмотреть. Например, вы можете использовать мастер для создания страницы базы данных Access. Другие способы включают в себя использование существующей Web-страницы из Интернета или использование Autopage. В конце концов страницу базы данных Access можно создать из памяти.

Сохранение объекта в виде Data Access Page

Это самый простой путь создать довольно-таки хорошее подобие вашей формы или отчета. Но это вовсе не означает, что вы не можете применить этот способ в таблицах или запросах. Однако есть некоторые ограничения, которые следует иметь в виду при конвертировании объектов в Data Access Pages. Эти ограничения указаны в табл. 13.2.

Таблица 13.2. Ограничения Data Access Page

Неподдерживаемые объекты	Неподдерживаемые свойства
Закрепленные границы объектов	Значение выводится как значение строки
Незакрепленные границы объектов	Множество столбцов в списковых окнах
Переключатели	Преобразование подформы и подотчета
Табуляторный элемент управления	Преобразование кода
Диагонали	выражения, которые обращаются к свойствам форм и подформ

Однако даже с этими ограничениями (этот список не полон) опция Save As - прекрасный способ эмулировать функциональные возможности вашего первоначального объекта. Помните также, что после преобразования вы можете редактировать страницы согласно вашим потребностям. Выполните следующие шаги для конвертирования вашего объекта в Data Access Page.

1. В окне базы данных Music Store в разделе **Формы** щелкните по форме Customers Example.
2. Выберите в меню **Файл** (File), **Сохранить как** (Save As). В окне с сообщением Save Form 'Customers Example' То поверх указания Copy (Копировать) объекта Customers Example наберите указание читать CustomersExamplePage.
3. В окне As (Как) щелкните по стрелке, указывающей вниз, и выберите Data Access Page, как показано на рис. 13.4.
4. Щелкните OK для открытия диалогового окна. Убедитесь, что вы указали нужную вам директорию, и затем снова щелкните OK. Закройте страницу, которая откроется.
5. В меню **Объекты** (Objects) щелкните **Страницы** (Pages). Затем сделайте двойной щелчок по CustomersExamplePage, как показано на рис. 13.5. (вы можете открыть эту страницу в Windows Explorer.)

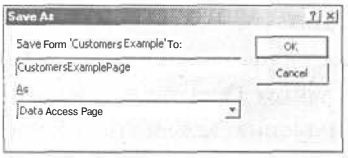


Рис. 13.4. Используйте опцию Сохранить как (Save As) для конвертирования существующего объекта Access в Data Access Page

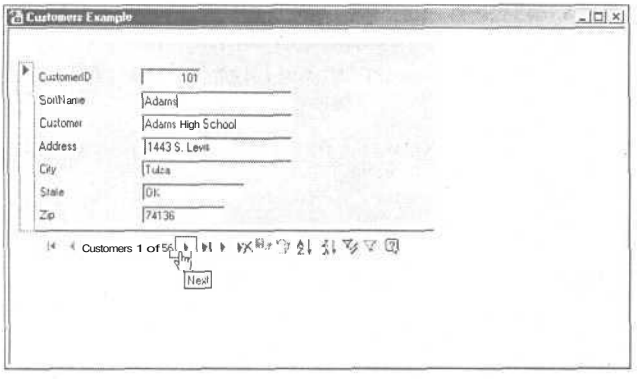


Рис. 13.5. CustomerExamplePage Data Access Page была создана из формы Access Customers Example

6. Щелкните по навигационной кнопке следующей записи внизу формы, как показано на рис. 13.5. Помните, что вы можете передвигаться по форме с помощью этой кнопки.
7. Щелкните по первой навигационной кнопке внизу слева, чтобы вернуться к первой записи.
8. Щелкните по полю City, в котором появится сообщение Tulsa.
9. Щелкните по иконке Apply Filter (Применить фильтр), которая выглядит как воронка с молнией. Заметьте, что индикатор внизу изменился с 1 из 56 на 1 из 15. Только записи Tulsa доступны для просмотра.
10. Чтобы выключить фильтр, щелкните по воронке без молнии справа от кнопки Apply Filter.
11. Щелкните по полю Zip, а затем по иконке Sort (Сортировка), на которой нарисована буква «А» и буква «Z» под «А». Просмотрите записи и убедитесь, что они отсортированы.
12. Щелкните по полю SortName и отсортируйте его. Закройте страницу, но не закрывайте базу данных.

К этому времени вы должны принять во внимание, что вы сохранили некоторые функциональные возможности исходной формы.

Создание Data Access Page с помощью мастера

Обратите внимание в следующем примере на то, что если вы выбираете таблицу, то в результате получаете форму. Это то, чем является Data Access Page - форма, которую можно открыть в браузере. Не имеет значения, какой способ вы использовали для создания страницы, в результате всегда получится форма. Механизм мастера сходен с механизмом мастера форм или мастера отчетов.

1. В базе данных Music Store, в меню **Объекты** (Objects), щелкните Pages, чтобы просмотреть список Data Access Pages. Дважды щелкните мышью по опции Create Data Access Page by Using Wizard (Создать страницу доступа данных с помощью мастера) для открытия первой страницы Page Wizard (Мастер страниц).
2. Выберите таблицу Customer из ниспадающего окна **Таблицы/запросы** (Tables/Queries), как показано на рис. 13.6.

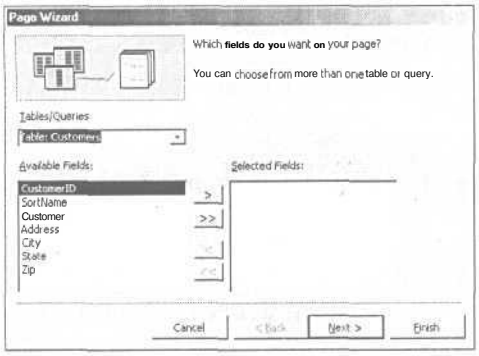


Рис. 13.6. Вы выбираете таблицы и поля в мастере Data Access Page так же, как и в других мастерах Access

3. Щелкните по >>, чтобы выделить все поля, а затем щелкните Next, чтобы перейти к следующей странице мастера. Двойным щелчком выделите поле State для уровня группировки, как показано на рис. 13.7.

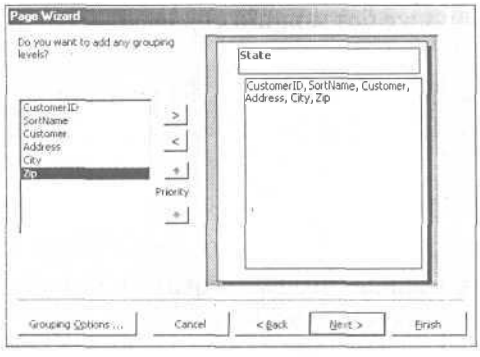


Рис. 13.7. Вы выбираете поле State для уровня группировки в мастере страниц

4. Щелкните Next (Дальше), чтобы перейти к следующей странице мастера создания страниц, в которой вы устанавливали порядок сортировки. Выберите SortName в ниспадающем меню для порядка сортировки на этой странице (см. рис. 13.8).

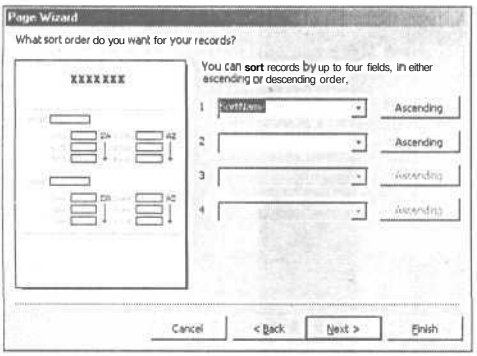


Рис. 13.8. Раздел Мастера Создания Страниц, где вы указываете порядок сортировки, позволяет вам выбрать условие сортировки

5. Щелкните Next, чтобы открыть последнюю страницу мастера создания страниц, как показано на рис. 13.9.

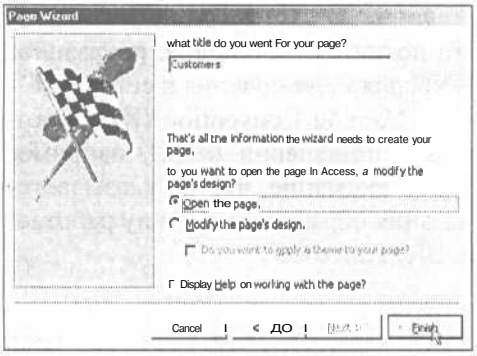


Рис. 13.9. На последней странице мастера создания страниц вы можете или немедленно открыть, или редактировать оформление страницы

6. Выберите Customers Group Page при вопросе What Title Do You Want For Your Page? (Как вы хотите назвать вашу страницу?). Щелкните по кнопке Open the Page (Открыть страницу), а затем по кнопке Finish, чтобы открыть страницу.
7. Щелкните Close (Закрыть), а затем Yes (Да), появится сообщение Do You Want to Save Changes to the Data Access Page? (вы хотите сохранить изменения Data Access Page?). Как показывает рис. 13.10, когда вы наводите курсор на поле File Name, появляется всплывающая подсказка по управлению, кото-

рая просит ввести имя файла или Web-адрес (http://). Тем не менее имя Customers Group Page должно автоматически появиться в этом поле.

Рис. 13.10. Введите имя файла в диалоговом окне Save As после запуска мастера

8. Щелкните Save (Сохранить). Заметьте, что появится сообщение, показанное на рис. 13.11, которое просит вас ввести UNC для подключения к сети. UNC - это путь в сети, соответствующий Universal Naming Convention (Имя, соответствующее соглашению об универсальном назначении имен), например `\\ServerName\FolderName\FileName`. Обратите внимание, что не указывается имя диска. Этот универсальный способ указания обращения к файлу работает в локальной сети или корпоративной сети. Щелкните OK.

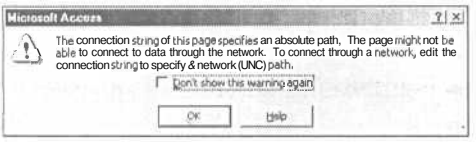


Рис. 13.11. Когда вы нажимаете Save для закрытия Page Wizard, появляется предупреждение, что вы должны выбрать путь UNC на тот случай, если вы хотите подключиться к сети

9. Двойной щелчок по странице Customer Group Page, которую вы создали. Заметьте, что вы видите только поле State (см. рис. 13.2).
10. Щелкните по кнопке переключения записей Next (Следующая), которая выглядит как направленная направо стрелка.

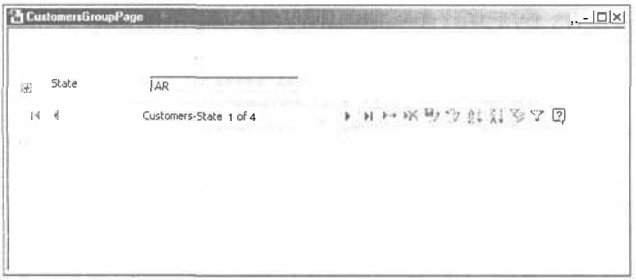


Рис. 13.12. Когда вы открываете страницу Grouped Data Access Page, вы видите только поле State перед тем, как продвинуться ниже

11. Щелкните по кнопке переключения записей First (Первая), а потом щелкните + возле метки State(Режим) для развертывания группы.
12. Щелкните Next в навигационной панели инструментов высокого уровня, как показано на рис. 13.13 (показаны шаги с 1-го по 6-й для Арканзаса).

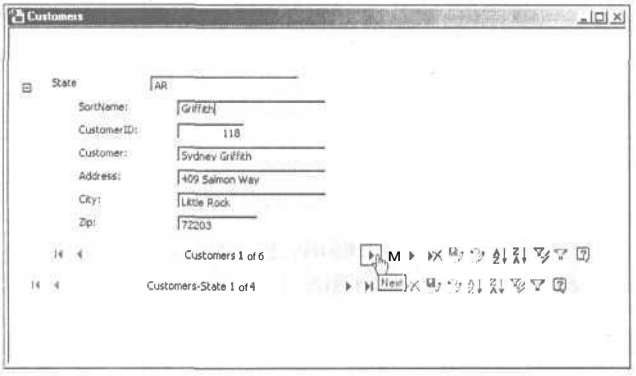


Рис. 13.13. После развертывания группы, вы увидите записи для каждого из режимов

13. Щелкните Next в навигационной панели инструментов низкого уровня. Разверните следущий режим, и выполните п. 12.
14. Щелкните по кнопке Delete (с символом, напоминающим X) на панели инструментов высокого уровня, но щелкните No(НеТ), чтобы избежать подтверждения удаления и отменить удаление. Закройте страницу. Вы также можете добавлять и редактировать записи, так как эта страница - динамическая.

ПОДСКАЗКА. Вы можете открыть вашу группированную страницу в режиме Design и изменить свойства группировки по умолчанию таким образом, что, когда открывается форма, поле State развернуто. Просто щелкните правой клавишей мыши в любом месте в верхнем разделе (Customers-State) и в появившемся меню выберите свойства уровня группировки. Затем измените значение Expanded ByDefault на True. Снова откройте страницу - и вы сразу заметите внесенные изменения.

Создание Data Access Page из существующей Web-страницы

У вас есть любимая Web-страница? вы можете использовать способ создания собственной Web-страницы, основанной на уже существующей Web-странице. При этом нужно иметь в виду то, что вы получите динамическую страницу только в том случае, если HTML-документ, который вы редактируете, содержит XML-код. Тем не менее вы можете сэкономить время и, возможно, с максимальной пользой применить этот метод.

Начинайте с простых страниц или (что было бы лучше) используйте собственные страницы, если вы уже создали какую-либо. Некоторые Web-страницы являются многосложными и поэтому выглядят устрашающе. Ищите страницы, которые содержат списки.

Следующие шаги проведут вас через процесс создания страницы с помощью использования существующей Web-страницы.

1. В окне Database (База данных), щелкните Pages (Страницы) в меню Objects (Объекты). Щелкните по кнопке New (Новая) на панели инструментов в окне базы данных.
2. В диалоговом окне New Data Access Page выберите Existing Web Page.
3. Щелкните OK для открытия диалогового окна Locate Web Page (Найти Web-страницу), затем найдите Web-страницу или HTML-файл, который вы хотите открыть.

Вы можете найти Web-страницу щелкнув по Search the Web (обратите внимание на положение курсора на рис. 13.14) в диалоговом окне Locate Web Page. У вас должна быть возможность получить динамический документ в зависимости от того, есть ли XML-код в пределах HTML-страницы.

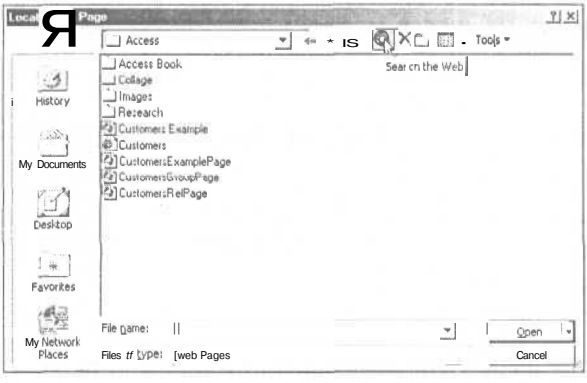


Рис. 13.14. Используйте кнопку Search the Web, чтобы найти существующую Web-страницу

ЗАМЕЧАНИЕ. Если вы не можете найти подходящую страницу, обратитесь к ссылке http://www.microsoft.com/office/access/using/default.htm. Вам, должно быть, придется сначала подключиться к Интернету, в зависимости от того, какой у вас тип подключения.

4. Сохраните копию страницы с помощью команды Save As в меню File в Microsoft Internet Explorer и выберите Edit Web page (Редактировать Web-страницу), которая уже есть в разделе Pages (Страницы).
5. Выберите папку, в которой вы сохранили Web-страницу, и дважды щелкните по файлу, который вы выбрали. После просмотра страницы, когда будете ее закрывать, выберите Yes(Да) для ее сохранения.
6. После закрытия страницы Microsoft Access создаст ярлык к HTML-файлу в окне Database (окно базы данных).

Создание Data Access Page с помощью AutoPage

Этот метод достаточно прост и не требует особых усилий. Просто выберите New в разделе Pages, а затем выберите AutoPage: Columnar (Columnar означает стиль Rolodex). Преимущество этого метода в его скорости. Недостаток в том, что вы должны в совершенстве знать, что именно вам нужно: все поля из отдельной таблицы или из запроса. Но вы можете воспользоваться этим методом для быстрого создания страницы, доступной для редактирования. Тогда вы уже сможете удалять все ненужные вам поля.

Создание Data Access Page в конструкторе с нуля

Хотя этот метод и самый сложный, он хорош для ознакомления с доступными инструментальными средствами для редактирования. В некоторых случаях метод является единственно возможным.

1. В меню Objects выберите Pages, чтобы просмотреть сохраненные страницы Data Access Pages.
2. Двойной щелчок по Create Data Access Page in Design View (Создание Data Access Page в режиме конструктора). Щелкните OK, когда появится сообщение, предупреждающее, что вы не можете открыть созданную ранее вами страницу в конструкторе Access 2000.
3. Щелкните по текстовому окну Click here and Type Title Text, затем в нем наберите `Customers` Relational Page.
4. Щелкните по области под этим заголовком; перетаскивайте нужные вам поля из списка полей в эту область страницы.
5. Щелкните по кнопке Field List слева от кнопки Toolbox (или выберите в главном меню View, Field List).
6. Если раздел Tables не раскрыт, щелкните + чтобы раскрыть его.
7. Раскройте Customers table, щелкнув +, чтобы просмотреть Field List, как показано на рис. 13.15. Обратите внимание на раздел Related Tables.
8. Щелкните по полю `CustomerID` затем, зажав клавишу Ctrl, щелкните по полям Customer и Address. Далее перетащите поля левой клавишей мыши в Drag Fields из Field List и поместите в этой секции страницы.

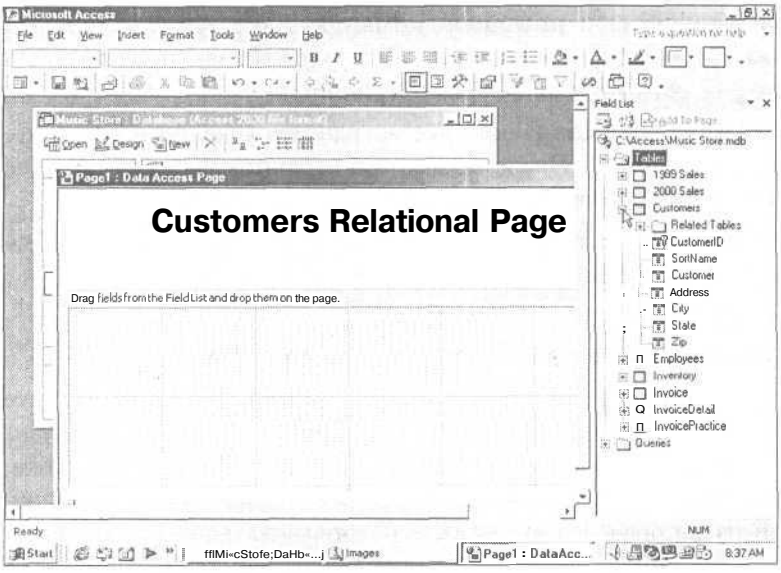


Рис. 13.15. В окне Field List в режиме Design View Data Access Page видны поля и таблицы

9. Пока поля выделены, переместите их как можно дальше в секции.

10. В окне Field List щелкните по полю City и, зажав клавишу Shift, щелкните по полю Zip. Перетащите поля вправо от полей в секции Customers.

11. Щелчок правой клавишей мыши по полю State, а затем выберите Promote, как показано на рис. 13.16. Заметьте, что поле State

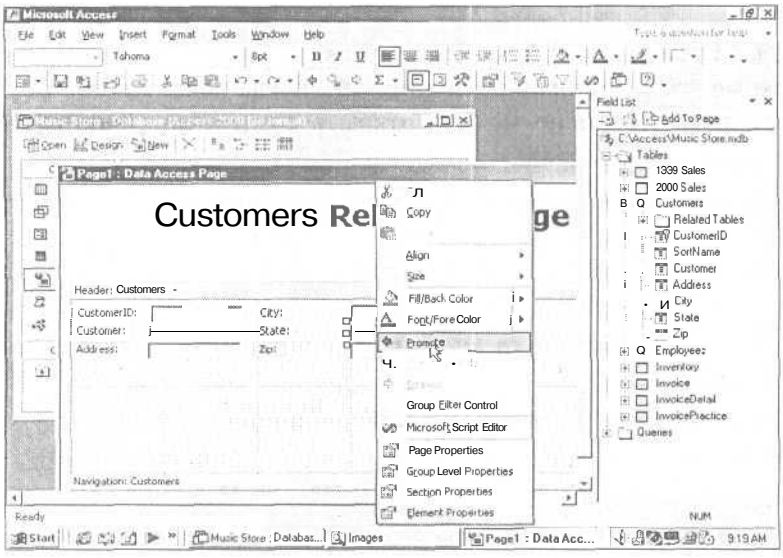


Рис. 13.16. Вы можете повысить приоритет любого поля, чтобы создать заголовок группы для вашей Data Access Page

12. Щелкните по кнопке Undo (Отменить) (для того, чтобы вы посмотрели результат).
13. Щелкните + в поле Related Tables для того, чтобы открыть его. Щелкните + в таблице Invoice для того, чтобы раскрыть ее. Щелкните по таблице (в отличие от Field List или плюс, или минус) и перетащите ее в секцию Customers.
14. Когда появится мастер планирования, выберите опцию Pivot Table (Сводная таблица). Ваша страница должна выглядеть так, как показано на рис. 13.17. Если Мастер планирования не появляется, отмените размещение, активизируйте указатель мастера в вашей панели инструментов и проделайте те же самые действия еще раз.

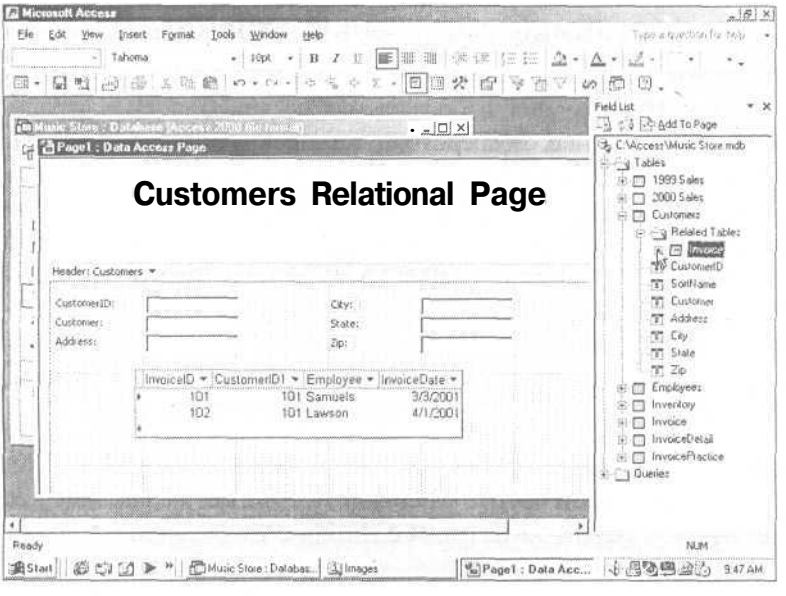


Рис. 13.17. Вы можете перетаскивать отнесенную (related) таблицу целиком по Data Access Page и использовать опцию Pivot Table

15. Сохраните и закройте форму с названием CustomersRelPage.
16. Откройте двойным щелчком мыши CustomersRelPage. Ваша страница должна выглядеть как показано на рис. 13.18.
17. Просмотрите страницу, обращая внимание на наличие реляционной функциональности. Закройте форму. Теперь вы можете открыть страницу при помощи Windows Explorer или Internet Explorer.

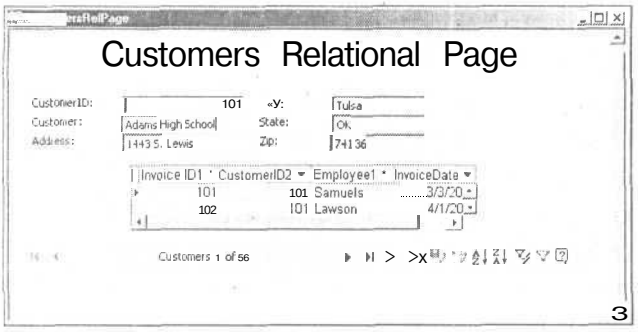


Рис. 13.18. Страница Customers Relational Page была создана с нуля в режиме конструктора

Active Server Pages (Активные серверные страницы)

ASP - технология Microsoft на основе сервера, предназначенная для создания динамических Web-страниц. Информация, отображаемая на динамических страницах, зачастую связана с базой данных посредством языка подготовки сценариев, таким, как VBScript. ASP широко применяется в интранет компании. Это, конечно, один из самых популярных методов создания Web-страниц, своевременно изменяющихся при изменении данных сервером. Например, Web-страница Microsoft является ASP-страницей. Пример использования ASP можно найти на сайте Microsoft no адресу www.microsoft.com/biztalk/using/tips/default.asp.

Ниже перечислено несколько причин популярности ASP.

- **Независимость от браузера.** Вследствие того, что ASP запускаются с Web-сервера вне браузера, ASP не зависят от браузера.
- **Доступ к базе данных.** Поскольку в основе ASP лежит технология ActiveX Data Object (ADO), что является важной частью Microsoft Access, так же как и другие базы данных, такие, как SQL Server, интеграция с Интернетом очевидна.
- **Легкость в эксплуатации.** ASP написаны с помощью скриптов, поэтому код этих страниц легко редактировать. В случае обнаружения дефектов в ASP, просто откройте скрипт с помощью любого текстового редактора, например Notepad, и подкорректируйте его, причем нет необходимости в рекомпиляции.

Серверные и клиентские скрипты могут быть совмещены при написании ASP-страниц; вы можете использовать преимущество этого метода, даже если у вас совершенно нет опыта написания скриптов. Но, прежде чем экспортировать, убедитесь, что у вас есть доступ к Web-серверу и установлено соответствующее программное обеспечение, такое, как Microsoft Information Services (IIS). Если вы не уверены, спросите администратора сети, установлены ли программные средства сервера. Если программное обеспечение установлено, выполните следующие пункты (или попросите администратора выполнить их), чтобы подготовиться к функциональным возможностям ASP.

1. Создайте папку на сервере в корневом каталоге, где ASP-файлы могут постоянно храниться. Корневой каталог по умолчанию: \Webshare\Wwwroot for

Personal Web Server и \Inetpub\Wwwroot for Microsoft Internet Information Server.

2. Администратор должен определить соответствующие права доступа к папке.
3. Скопируйте ASP-файлы в папку на сервере, а также все сопутствующие файлы (графические файлы, привязанные файлы и все папки, содержащие файлы подобного рода) в ту же папку или убедитесь, что все связанные файлы могут быть найдены программными средствами сервера.
4. Скопируйте файлы базы данных Microsoft Access в папку или определите их положение в сети.
5. Определите источник данных как Систему (System) DSN (Data Source Name - «Имя источника данных») на Web-сервере (о том, как это сделать, говорится в последующих пунктах). Убедитесь, что имя источника данных такое же, как и то имя, которое вы вводили в соответствующем диалоговом окне Output Options, когда вы экспортировали ASP-файлы.
6. Для проекта Microsoft Access в окнах Username (Имя пользователя) и Password (Пароль) впишите имя пользователя базы данных и пароль для того, чтобы разрешить доступ пользователям к базе данных Microsoft SQL Server с Web-страницы. Если не будет вписано имя пользователя или пароль, имя пользователя по умолчанию - Sa, без пароля.
7. Пароль и имя пользователя должны совпадать с паролем и именем пользователя, которые вы вводили в окнах User To Connect As (Подсоединить пользователя под именем) и Password for User (Пароль) в диалоговом окне Microsoft Active Server Pages Output Options, которые появляются, когда вы выкладываете ASP-файлы.

После выполнения этих шагов вы почти готовы к выполнению операции экспорта. Однако если вы не уверены в определении источника данных ODBC в п. 5, выполните следующие пункты для полной уверенности, что у вас есть активный DNS. Если снова появятся какие-либо сложности, обратитесь к администратору сети.

1. Откройте ODBC Data Source Administrator (Администррирование источника данных ODBC) на вашем Web-сервере. В Windows 98 щелкните Start (Пуск), Settings (Настройка), Control Panel (Панель управления), а затем откройте ODBC Data Sources (32 bit) (Источник данных ODBC). В Windows 2000 и Windows NT щелкните Start, Settings, Control Panel; щелкните Administrative Tools (Администрирование) и затем откройте ODBC Data Sources (Источники данных ODBC).
2. В диалоговом окне Data Sources выберите закладку System DSN (Системный DNS). Щелкните Add (Добавить), если вы не видите имя System DNS (Системный DNS), которое вы использовали как DNS в процессе экпорта. Щелкните Microsoft Access Driver (Драйвер), затем Finish.

ЗАМЕЧАНИЕ. Если не появился Microsoft Access-драйвер (Driver), его нужно инсталлировать на ваш Web-сервер.

3. Введите запрашиваемую информацию в диалоговом окне ODBC Microsoft Access Setup. Имя, которое вы ввели в окне Data Source Name, совпадает с именем, которое вы используете в окне Data Source Name в диалоговом окне Экспорт (Export).
4. Щелкните OK, чтобы закрыть диалоговое окно ODBC Microsoft Access Setup, а затем щелкните OK в диалоговом окне ODBC Data Source Administrator.
5. Если DSN в System DSN на вашем Web-сервере отлично от того DSN, который вы использовали при создании Web-страниц, вернитесь в процесс экспорта в Microsoft Access и создайте заново ваши Web-страницы, используя правильный DSN.

Если вы не имеете доступа к Web-серверу, нужно чтобы администратор принял соответствующие меры, выполнив указанные действия. Когда вы экспортируете объект Access, в окне Save As Type выберите Microsoft Active Server Pages и выберите папку, которую вы использовали в п. 1 в окне Server URL. Например, если у вас был путь \Inetpub\Wwwroot folder, наберите следующий URL: http://<server name>/wwwroot/<aspfilename.asp>.

После того как вы опубликуете ASP-файлы на Web-сервере, IIS (Информационный сервер Интернета) сможет запустить код VBScript, вызвать серверный(?) элемент управления ActiveX, открыть базу данных и отправить динамически созданный файл HTML в Web-браузер как Web-страницу.

Гиперссылки в объектах Access

Гиперссылки легко реализуются в Access. Хотя и существуют другие методы создания гиперссылок в Access, вы можете просто напечатать адрес Интернета или e-mail-адрес в поле гиперссылок. Тем не менее вы можете не ограничиваться только лишь Интернет-страницами и адресами e-mail. Через гиперссылку вы можете подключить объекты баз данных, такие, как таблицы, формы или отчеты, документы Word или даже документы Excel. При щелчке мыши по гиперссылке откроется подключенный к ней объект.

Вы можете использовать гипрессылки для навигации по базе данных. У них есть много преимуществ над макросами или процедурами, подключенными к кнопкам управления, которые открывают объекты Access. Прежде всего их легче создавать, легче эксплуатировать и они исправно адаптируются к создаваемым меню. Но для некоторых людей лучшим преимуществом является то, что им не придется быть экпертом по Access, чтобы создавать и эксплуатировать гиперссылки.

Проследуйте по следующим пунктам, чтобы бегло ознакомиться *с* мощью гиперссылок.

1. Откройте базу данных FormAndControls. В разделе **Формы** (Forms) откройте форму гиперссылок.
2. Щелкните по некоторым ссылкам для проверки. Формы должны открыться.
3. Щелкните по кнопке View, чтобы войти в режим Design View. Выберите Insert, Hyperlink (рис. 13.19).

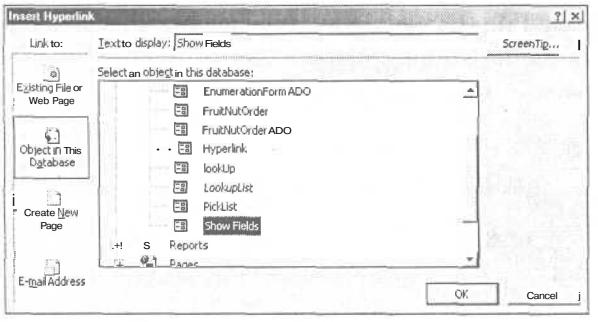


Рис. 13.19. Вы можете вставлять гиперссылки в формах, чтобы открывать объекты, обращаясь к ним как к пунктам меню

4. В Link To в левой части экрана выберите Objects In This Database для просмотра списка объектов баз данных с помощью навигационных кнопок.
5. Щелкните + в Forms для прокрутки вниз. Используйте вертикальную полосу прокрутки и следуйте вниз до тех пор, пока не увидите форму Show Fields.
6. Щелкните по форме Show Fields и обратите внимание, что кнопка OK активизировалась. Щелкните OK. Гиперссылка на форму Show Fields вставится в верхний левый угол формы.
7. Укажите курсором в центр гиперссылки Show Fields. Когда курсор видоизменится и будет выглядеть как рука, позиционируйте гиперссылку, напрямую перетащив ее в форму PickList, как показано на рис. 13.20.

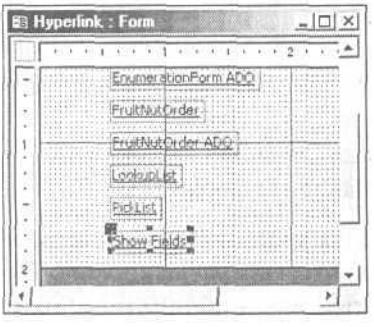


Рис. 13.20. Окно проектирования после вставки гиперссылки

8. Закройте и сохраните форму. Проверьте вашу гиперсылку, открыв форму Hyperlink и щелкнув по гиперссылке.
9. Откройте базу данных Northwind. В Forms откройте форму Suppliers, затем щелкните по полю Home Page (Домашняя страница). Щелкните Insert (Вставить), Hyperlink (Гиперссылка) для того, чтобы открыть диалоговое окно Insert Hyperlink (Вставить гиперссылку).

10.В панели Link To выберите Existing File (Существующий файл) или Web Page (Web-страница). Обратите внимание на кнопку Browse the Web, которая выглядит как глобус с увеличительным стеклом, как показано на рис. 13.21.

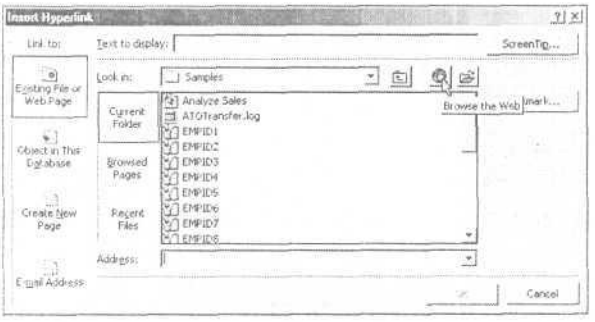


Рис. 13.21. Вы можете щелкнуть по кнопке Найти в Web (Browse the Web) для вставки гиперссылки или просто щелкните по любимой Web-странице, которую вы уже нашли

11.В Look In щелкните Browsed Pages (Найденные страницы), чтобы просмотреть недавно найденные Web-сайты и файлы. Эта опция предоставляет простой способ для вставки ссылок в последние недавно найденные страницы.

12.Щелкните по вашей любимой Web-странице в списке, затем щелкните OK. Заметьте, что ссылка автоматически вставляется в поле.

ЗАМЕЧАНИЕ, Убедитесь, что вы выбираете именно гиперссылку; в отличие от них файлы имеют префикс file:/// перед именем файла.

13.Пока ссылка подсвечена, напечатайте поверх ссылки e-mail-адрес вашего друга. Заметьте, что ссылка сразу же подчеркивается по мере того, как вы печатаете. Вы можете отправить письмо другу.

14.После того как вы наберете адрес, щелкните по нему правой клавишей мыши, выберите Hyperlink, а затем - Remove Hyperlink. Закройте форму.

Как уже ранее упоминалось, гиперссылки являются отличной альтернативой кнопкам управления. Теперь вы знаете на примерах, что гиперссылки выполняют много других функций, помимо просто открытия Web-Страниц; они легко доступны для наладки, эксплуатации и обслуживания.

Что дальше?

вы изучили Data Access Pages, Active Server Pages и гиперссылки в этой главе. Как вы уже убедились, Access 2000 имеет возможности для интеграции с Интернетом. В следующей главе мы рассмотрим интеграцию с SQL Server. Access хорошо подходит для работы с источниками данных вне Access. Эта гибкость только лишь увеличивает возможности для расширения вашей области действий при работе с этим мощным средством.

14
Интеграция с SQL-сервером

В предыдущей главе вы узнали об интеграции Access со Всемирной сетью - Интернетом. В этой главе рассматривается интеграция с SQL Server 2000 с использованием ядра Desktop Engine, которое поставляется вместе с Access 2002. Теперь у вас есть альтернатива Microsoft Jet. В частности, вы узнаете следующее:

- историю создания Microsoft Data Engine;
- чем MSDE 2000 отличается от SQL Server 2000;
- чем MSDE 2000 отличается от Jet;
- как установить MSDE 2000;
- как установить учебное приложение «Борей» («Northwind»);
- чем проект Access отличается от базы данных Access;
- как создавать сохраненные процедуры и триггеры.

История развития MSDE

Microsoft Data Engine (MSDE) был представлен с выпуском Access 2000. Он обеспечивал совместимость с Microsoft SQL Server 7.0 и был альтернативой ядру обработки данных Microsoft Jet. С появлением Access 2002 фирма Microsoft произвела улучшения и изменила название этого мощного ядра, продолжив его бесплатное распространение в качестве дополнительного бонуса (хотя он и не входит в стандартную установку Access). Теперь это ядро называется Microsoft SQL Server 2000 Desktop Engine (MSDE 2000) и полностью совместимо с SQL Server 2000. Это позволяет пользователю легко создавать и модифицировать SQL Server-совместимые базы данных, которые могут быть целиком портированы на SQL Server 2000 без каких-либо изменений.

Поскольку MSDE 2000 основан на том же ядре обработки данных, что и SQL Server 2000, большинство проектов Microsoft Access или клиент-серверных приложений смогут работать в обеих версиях. Однако это не означает, что новое ядро обладает абсолютно теми же средствами, что и SQL Server 2000. Следует рассматривать его как облегченную версию SQL Server 2000. В отличие от SQL Server 2000 ядро MSDE 2000 имеет ограничение на размер базы данных в 2 Гбайт (такое же, как в Access); но не поддерживает симметричную мультиобработку (Symmetrical Multiprocessing, SMP) для Windows 98 и старше и при использовании транзакционных репликаций не может быть публикатором репликаций (хотя может выступать в роли абонента репликаций).

Хотя Microsoft MSDE 2000 не такой гибкий, как SQL Server 2000, он все-таки может принести вас большую пользу. Как уже говорилось, вы можете использовать его вместо ядра Microsoft Jet. Преимуществом MSDE 2000 перед Jet является то, что вы можете использовать его для разработки и тестирования Access-проекта или клиент-серверного приложения на персональном компьютере (ра-

бочей станции), а затем изменить информацию о связях с данными в Access-проекте для соединения с базой данных SQL Server на удаленном сервере, чтобы произвести окончательное тестирование и выпуск продукта. Вам может недоставать некоторых инструментов управления из SQL Server, но у вас будет более чем достаточно возможностей для разработки при использовании MSDE 2000.

Используя проект данных Access в Access 2002, пользователи могут создавать и изменять простые сохраненные процедуры (stored procedures) SQL Server (совершенно новая возможность), используя конструктор сохраненных процедур (Stored Procedure Designer). Это дает возможность пользователям создавать сохраненные процедуры не изучая Transact SQL. (См. обсуждение Transact SQL в разделе «Сохраненные процедуры» далее в этой главе.) вы просто выбираете раздел **Запросы** из объектов вашей базы данных и дважды щелкаете мышью на пункте «Создание сохраненной процедуры в режиме конструктора» (Create Stored Procedure in Designer).

Что все это значит для вас, начинающего разработчика? Это значит, что у вас есть инструмент, который вы можете использовать в качестве сервера базы данных для маленькой рабочей группы. Например, если вы предвидите, что со временем ваш бизнес будет разрастаться, пока в конце концов ему не понадобится вся функциональность SQL Server 2000, работающего на большом сетевом сервере, обязательно разрабатывайте ваши приложения используя Access-проект, связанный с SQL Server 2000 Desktop Engine.

Это также значит, что у вас есть SQL-платформа для тестирования, изучения и даже разработки, позволяющая вам сделать первые шаги в среде SQL, не платя при этом за дорогие мощные системы. Это приведет только к улучшению вашего резюме и расширению горизонтов, связанных с поиском престижной работы, не говоря уже об удовольствии и удовлетворении, которые вы получите от этих шагов.

Сравнение MSSQL Server 2000 Desktop Engine и Jet

Если забыть об очевидном преимуществе совместимости Desktop Engine с SQL Server, возникает соблазн спросить: «А почему бы просто не отдать предпочтение Microsoft Access?» Ваш бизнес может расшириться в дальнейшем. Как Jet, так и MSDE 2000 разработаны для маленьких рабочих групп. У обеих технологий имеется ограничение на размер файла в 2 Гбайт. (Поэтому не следует исключать возможности перехода на SQL Server, раз уж она вам предоставляется бесплатно. – *Пер.*.)

Большая производительность

Даже для небольших рабочих групп MSDE 2000 показывает большую производительность, чем Jet, поскольку запросы обрабатываются на сервере, а не на рабочей станции. И только данные, полученные от обработки запроса, пересылаются на рабочую станцию с сервера. В добавление к этому при создании клиент-серверных систем сервер обычно устанавливается на более мощную машину. Поскольку сервер производит обработку запросов, загрузка снимается с клиентов.

Лучшая сохранность данных

В дополнение целостность и надежность данных выше при использовании desktop engine. Это объясняется тем, что MSDE 2000 имеет те же инструменты для поддержания целостности данных, что и SQL Server. Например, если перебой в электросети приводит к повреждению базы данных, SQL Server может произвести ее восстановление, используя журнал транзакций (log-файл): каждое изменение базы данных запротоколировано в нем. Он используется механизмом автоматического восстановления для возвращения базы данных к ее последнему работоспособному состоянию всего за несколько минут. Это означает, что критические ко времени приложения могут продолжить свою работу тотчас же. Напротив, если сбой системы происходит при использовании Jet, база данных может оказаться поврежденной, что может привести к тому, что вам придется возвращаться к последней резервной копии.

Надежнее безопасность

Подход «клиент-сервер» также обеспечивает более высокий уровень безопасности. По сравнению с ним уровень безопасности в Access ограничен. При пользовании базой данных SQL Server несанкционированный пользователь должен сначала получить доступ на сервер, тогда как в Access он открывает непосредственно файл базы данных. Более того, сохраненные процедуры могут играть роль дополнительных мер безопасности. Например, вы можете запускать встроенные сохраненные процедуры SQL Server, чтобы обеспечивать пользователю доступ к базам данных в зависимости от его регистрационного имени (login), которое он вводит при входе на SQL Server. Или, если захотите, вы можете давать ему доступ в зависимости от его регистрационного имени Windows в сети.

Поскольку SQL Server Enterprise Manager (программа управления SQL Server. - *Пер.*), являющаяся частью SQL Server 2000, не входит в комплект поставки MSDE 2000, инструменты управления безопасностью будут недоступны после установки MSDE 2000. Доступны, однако, другие возможности установки разрешений, включая следующие:

- Вы можете употребить встроенное регистрационное имя **sa** (system administrator, системный администратор) с паролем, чтобы пользователи могли заходить как администраторы SQL-сервера. Минусом такого подхода является то, что у них будут полные администраторские права в MSDE 2000.
- Если у вас установлена Windows NT или Windows 2000, вы можете применить встроенные настройки безопасности, чтобы дать пользователям локальные администраторские права Windows; но при этом они получат полный доступ к ресурсам компьютера.
- Вы можете купить и установить Microsoft XP Developer, в состав которого входит лицензионное соглашение конечного пользователя (end-user license agreement, EULA), распространяющееся только на разработчиков (developer-only), позволяющее установить SQL Server Personal (включая Enterprise Manager, обеспечивающий инструменты для создания учетных записей пользователям и установки им разрешений на применение баз данных).

- Если у вас есть доступ к копии SQL Server 2000, вы можете установить копию SQL Server Enterprise Manager с CD. Это дает вам объектный уровень безопасности и позволяет добавлять пользователей сетевого домена.
- Вы можете применять встроенные сохраненные процедуры SQL для обеспечения пользователям доступа к базе данных в зависимости либо от имени, вводимого при входе на SQL Server, либо от имени пользователя Windows в сети.

Если у вас не установлена Windows NT или 2000 и нет доступа к SQL Server Enterprise Manager, вы можете использовать последнюю из этих возможностей для обеспечения сетевой безопасности. Существует четыре встроенные (системные) процедуры, которые вы можете использовать для обеспечения пользователей правами доступа к базе данных. Первая сохраненная процедура (stored procedure), sp_grantlogin, позволяет пользователю получить доступ к MSDE 2000. Вторая сохраненная процедура, sp_default db, определяет базу данных, которая будет применяться по умолчанию новым пользователем. Третья сохраненная процедура, sp_grantdbaccess, обеспечивает пользователя правами доступа к базе данных. И последняя сохраненная процедура, sp_addrolemember, определяет, какого рода доступ дозволен.

Следующая сохраненная процедура применяет все четыре встроенные процедуры для запроса у пользователя имени <domain>\<user> (<домен>\<пользователь> - *Пер.*)и названия базы данных:

```
CREATE PROCEDURE GrantAccess
@NewUser    VarChar(30),
@db         VarChar(30)
AS
  EXEC sp_grantlogin @NewUser
  EXEC sp_defaultdb @NewUser, @db
  EXEC  sp_grantdbaccess  @NewUser
  EXEC sp_addrolemember "db_owner", NewUser
RETURN
```

По всем этим и другим не названным здесь причинам MSDE 2000 - хороший выбор по сравнению с Jet, особенно в сетевом окружении. С другой стороны, Jet использует меньше памяти (как дисковой, так и оперативной), чем engine. По этой причине Jet выигрывает в скорости у MSDE 2000 на старых машинах с малым количеством памяти. Создание приложений проще с использованием Access 2000 и Jet, поскольку требуется меньше административной работы.

Установка и запуск сервера

На одну только эту тему могла бы быть написана целая книга, но вместо чтения книг лучше изучить MSDE 2000 на примере. Первое, что вы должны сделать, - это инсталлировать ядро. Нет ничего проще. Просто вставьте компакт-диск Office XP в ваш CD-ROM, нажмите кнопку **Пуск** (Start) на панели задач и выберите **Выполнить** (Run). Когда появится окно **Запуск программы** (Open), нажмите **Обзор** (Browse). В зависимости от того, какая буква приписана вашему CD-ROM, подставьте соответствующий диск вместо D в следующем примере:

```
d:\msde2000\setup.exe
```

После того как вы вернетесь к окну «Запуск программы», нажмите OK. После того как установка программы подойдет к концу, вас спросят, хотите ли вы перезагрузить компьютер для завершения процесса установки. Ответьте «Да». После того как ваш компьютер перезагрузится, останется еще один важный шаг. В системном трэе (system tray), который обычно расположен в нижнем правом углу экрана, вы увидите новую иконку, на которой изображен компьютер, рядом с которым белый кружок. Внутри белого кружка - красная точка. Дважды щелкните по ней мышью и оставьте полученное окно развернутым. Вы должны видеть на экране примерно то же, что показано на рис. 14.1.

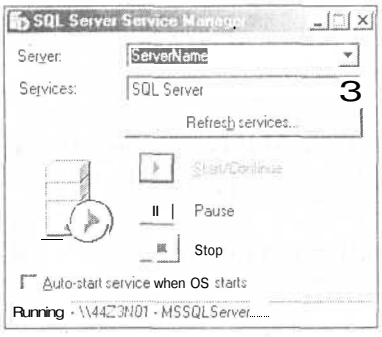


Рис. 14.1. Диалоговое окно MSSQL Server Service Manager

Если на вашем компьютере не установлено другого сервера (такого, как MSDE), программа установки должна была взять в качестве названия сервера название вашего компьютера. Чтобы увидеть, так ли это, щелкните правой кнопкой мыши на «Сетевом окружении» (Network Neighborhood) и выберите **Свойства;** далее найдите вкладку, где написана идентификационная информация, и проверьте имя компьютера (в Windows 2000 нужно в свойствах «Моего компьютера» переключиться на вкладку **Сетевая идентификация.** - *Пер.*).

Теперь, когда сервер установлен, его нужно запустить. Нажмите кнопку Start/Continue. Обратите внимание на опцию Auto-Start Service when OS Starts (Автоматически запускать службу при загрузке операционной системы). Если вы выберете эту опцию, помните, что сервер потребляет системные ресурсы, а бывают случаи, когда вам нужно освободить максимально возможный объем оперативной памяти. Если по вашему мнению вам не хватает памяти или вы не уверены, не выбирайте эту опцию. Если у вас много RAM, для вас это не должно быть проблемой, однако имейте этот факт в виду.

Установка учебного приложения «Northwind» («Борей»)

Теперь, когда вы произвели установку и запуск, вам нужен проект, с которым можно работать. Откройте базу данных «Борей» («Northwind»). Если у вас ее нет или вы в этом сомневаетесь, ее можно установить из меню **Справка** в Access. Просто выберите **Справка, Примеры баз данных** (Help, Sample Databases). Если программа не сможет найти либо базу данных «Борей», либо проект «Борей»,

она спросит вас, хотите ли вы их установить. Выберите «Да» для того и другого или для одного из них, в зависимости от того, что вам нужно.

Допустим, возникла проблема во время установки или вы не можете найти установочный CD для Office. Как уже упоминалось ранее, вы можете скачать базу данных «Борей» с сайта Microsoft. Но если вам нужен всего лишь проект, существует другая возможность, которую в любом случае неплохо иметь в виду. Вы можете преобразовать вашу базу данных в проект (сделать upsize). (У вас могут быть также и другие базы данных, которые необходимо преобразовать.) Если ваш учебный проект «Борей» установился без проблем, вы можете удалить проект, получившийся преобразованием, который нужен был только для тренировки. В противном случае используйте преобразованный проект в примерах этой главы. Следуйте этим инструкциям для преобразования.

1. Откройте базу данных «Борей» («Northwind»).
2. Из меню выберите **Сервис, Служебные программы, Мастер преобразования в формат SQL Server** (Tools, Database Utilities, Upsizing Wizard) для открытия мастера преобразования.
3. На первой странице мастера выберите **Создать базу данных** (Create New Database) (рис. 14.2) и нажмите **Далее.**

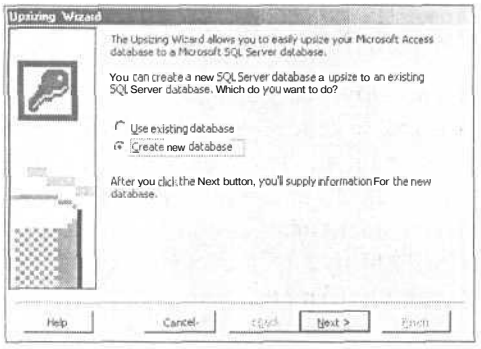


Рис. 14.2. На стр. 1 мастера преобразования вы можете выбрать создание нового проекта базы данных, основанного на существующей базе данных Access

4. На стр. 2 мастера имя вашего компьютера будет автоматически вставлено в поле, где должен быть указан сервер SQL Server для базы данных (What SQL Server Would You Like to Use for This Database?). Если это не так, удостоверьтесь, что запущен MSSQL Server. Будем надеяться, вам не придется переустанавливать ядро сервера заново.
5. В поле для имени пользователя («Код входа», Login ID) просто укажите sa (system administrator, системный администратор) без пароля. (Если у вас или вашего клиента установлена система NT Server, SQL Server может использовать имя и пароль NT для удостоверения ваших прав доступа на сервер, но в данной демонстрации это не используется.) Оставьте имя новой базы данных неизмененным, как показано на рис. 14.3.

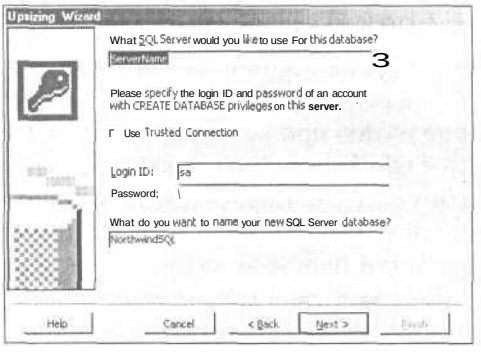


Рис. 14.3. На стр. 2 мастера преобразования просто введите sa без пароля

6. На стр. 3 нажмите >>, чтобы выбрать все имеющиеся таблицы, а затем нажмите **Далее.**
7. На стр. 4 оставьте все как есть (см. рис. 14.4) и нажмите **Далее.**

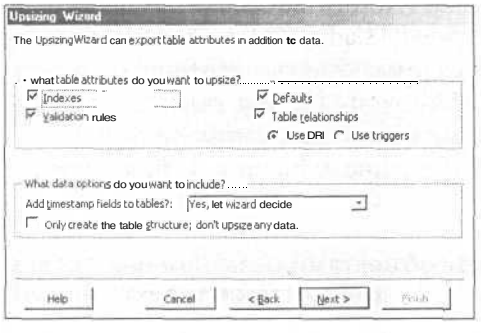


Рис. 14.4. Стр. 4 мастера позволяет вам выбрать атрибуты таблиц, которые будут преобразованы вместе с данными

Обратите внимание на различные варианты образования связей таблиц. В SQL Server существует два способа образования этих связей: DRI (Declarative Referential Integrity - декларативная целостность данных) и триггеры. SQL Server 2000 поддерживает каскадные DRI (аналогично каскадам в Access), которые позволяют вам создавать связи между главными и подчиненными таблицами (master table и dependent table). Такие связи приводят к тому, что удаление или изменение содержимого записи в главной таблице каскадом переносится на соответствующие подчиненные.

8. Если желаете, вы можете изменить каталог и имя создаваемого ADP-файла на следующем экране. Нажмите **Далее.**
9. Оставьте без изменения стр. 6 и закончите преобразование. (По умолчанию на последней странице установлено открытие ADP-файла после окончания преобразования.) Должен высветиться отчет, резюмирующий все выбранные настройки преобразования. Подтвердите его.

Изучение проекта «Northwind» («Борей»)

Как было замечено ранее, подробный анализ SQL Server мог бы занять целую книгу. Следовательно, в этой главе дан всего лишь обзор. Несмотря на это, вы почувствуете вкус того, что может стать вам доступно при условии, что вы уделите время изучению возможностей Microsoft SQL Server 2000 Desktop Engine. На самом деле ядро обладает неким базисным инструментарием, включающим резервное копирование, восстановление, удаление, копирование, перенос и опции безопасности SQL. Однако средства разработки проектов являются частью отнюдь не of MSDE 2000, а Microsoft Access. В заключение можно сказать, что, используя проекты Access (ADP), вы можете строить клиент-серверные приложения, включающие триггеры, сохраненные процедуры, представления, журналы протоколирования транзакций, откат и т. д., вне зависимости от того, используете ли вы MSDE 2000 или полную версию SQL Server 2000.

Попытайтесь мысленно отделить данные проекта от объектов, которые управляют этими данными. Данные содержатся в отдельных файлах с расширением.mdf. Например, файл данных для «Борей» («Northwind» в нерусифицированной версии. – *Пер.*)называется БорейCS.mdf (NorthwindCS.mdf). А файл проекта «Борей» называется БорейCS.adp (NorthwindCS.adp). Он содержит объекты пользовательского интерфейса, такие, как формы, отчеты, страницы доступа к данным, макросы и модули. Нет нужды беспокоиться. Вы можете работать с объектами данных так, как если бы они были записаны в том же файле, что и проект Access. В табл. 14.1 приведено сравнение объектов в базе данных Access и в проекте Access.

Таблица 14.1. Соответствие между объектами базы данных Access и объектами проекта Access

База данных Access	Проект Access
Таблица	Таблица
Запрос на выборку (Select Query)	Представление
Запрос на действие (Action Query)	Сохраненная процедура
Окно связей	Диаграмма базы данных
Форма	Форма
Отчет	Отчет
Страница доступа к данным	Страница доступа к данным
Макрос	Макрос
Модуль	Модуль

Таблицы

Когда вы два раза щелкните мышью на любой таблице в проекте Access, она будет выглядеть так же, как и в базе данных Access. Различие будет видно, когда вы нажмете кнопку **Конструктор** (design); различие будет в основном между типами данных. В табл. 14.2 показано соответствие между типами данных в базе данных Access и в SQL Server.

Таблица 14.2. Наиболее близко соответствующие типы данных в базе данных Access и в SQL Server

Access 2002 Database	Access 2002 Project
Текст	Varchar, Nvarchar, Char, Nchar
Memo	Text, Ntext
Счетчик	Колонка «Идентификация»
Currency	Money, Smallmoney
Date/time	Datetime, Smalldatetime
Yes/No	Bit
Объект OLE	Image
Гипертекстовая ссылка	Нет эквивалента
Long Integer	Int
Integer	Smallint
Double, Single	Float, Real

Чтобы просмотреть свойства таблицы, выполните следующее:

1. В проекте «Борей» («Northwind») (называющемся БорейCS или БорейSQL, в зависимости от того, какой метод вы использовали для создания проекта) в списке объектов, выберите **Таблицы.**
2. Выделите таблицу «Заказы» и нажмите **Конструктор,** чтобы открыть таблицу в виде конструктора. Обратите внимание, что для столбца «Код заказа» не отмечен галочкой пункт «Разрешить Null», поскольку поля, входящие в первичный ключ, должны иметь уникальные (неповторяющиеся) данные без пустых ячеек. Вид, который должен возникнуть на экране, показан на рис. 14.5.

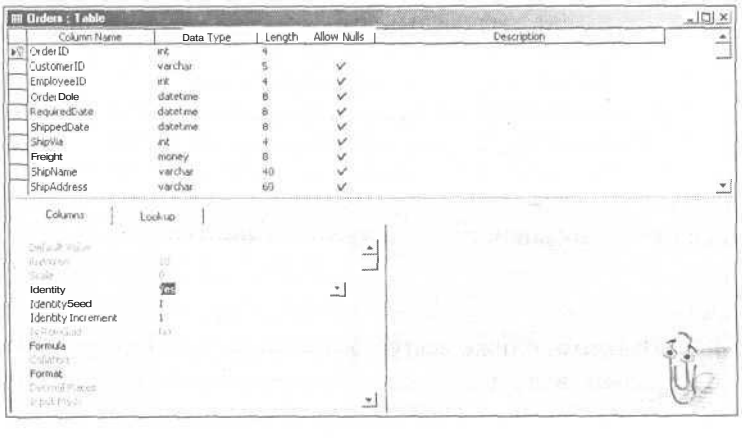


Рис. 14.5. Вы можете просматривать и изменять атрибуты таблицы в разделе Конструктор

* Информация противоречит справочной системе Access, см. раздел справки Access «Работа с проектами Microsoft Access»-«Общие сведения о проектах Microsoft Access»-«Различие типов данных в базе данных и в проекте Microsoft Access». - *Пер.*

3. Нажмите кнопку **Свойства.** Вы должны увидеть то, что показано на рис. 14.6.

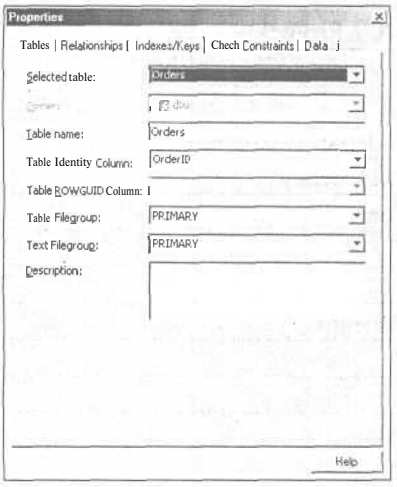


Рис. 14.6. Кнопка **Свойства** в конструкторе открывает страницу со свойствами, упорядоченными по вкладкам

4. Переключитесь на вкладку **Связи** - будут показаны связи первичного/внешнего ключа.
5. Переключитесь на вкладку **Индексы и ключи,** щелкните мышью по ниспадающему списку **Выделенный индекс** и выберите поле Дата размещения (ShippedDate). Отметьте галочкой опцию UNIQUE; обратите внимание, что затемненные элементы стали доступны. Уберите обратно галочку.
6. Переключитесь на вкладку **Проверить ограничения** и нажмите **Создать.** Вам предлагается ввести выражение. Нажмите **Удалить,** чтобы вернуть все как было.

В таблице «Сотрудники» (Employees) есть пример ограничения, которое является аналогом правила проверки соответствия (validation rule) в базе данных Access.

---

7. В завершение обзора свойств переключитесь на вкладку **Данные.**

Представления

Как видно из табл. 14.1, объектом, ближе всего соответствующим запросу, на выборку из Access, в SQL Server является представление. Оба они работают одинаково в том смысле, что они являются, по сути, выражениями, позволяющими выделять запрашиваемые данные из таблиц. Объединения (joins) - одна из многих схожих черт. В SQL Server можно создавать внутренние и внешние объединения, точно так же, как и в Access. Однако существует список операторов, таких, как =, >=, <=, <, которые вы можете использовать в объединениях для

SQL Server. Форматы страниц конструктора также очень схожи. Для того чтобы познакомиться с представлениями, вы начнете с изучения уже готовых представлений в «Northwind», а затем создадите свое представление с нуля.

1. Выделите раздел Запросы, из по-прежнему открытого проекта NorthwindCS. Щелкните правой кнопкой на Sales Total by Amount и выберите Конструктор, чтобы открыть представление. (Если вы получили проект «Northwind» преобразованием из базы данных, то у вас может отсутствовать указанное представление вследствие того, что для него не было прообраза в исходной базе данных. Это связано с недоработкой при создании Office XP. Для выполнения упражнений этой главы вам следует воспользоваться готовым проектом «Northwind». В варианте русификации, которым мне довелось пользоваться в проекте «Northwind», названия всех таблиц, форм и пр. приведены без перевода. Не без оснований полагая, что существует только один вариант русификации, здесь я также оставляю эти названия без перевода для вашего удобства. - *Пер.)*
   Обратите внимание, что условие отбора записано для двух полей. Также отметьте для себя формат написания дат в условии отбора (`BETWEEN '19970101' AND '19971231'`). В поле Subtotal таблицы Order Subtotals обратите внимание на наличие значка воронки, обозначающего применение условия отбора.
2. Щелкните правой кнопкой мыши на линии объединения *(join* line) и выберите Свойства - откроется страница свойств линии объединения, показанная на рис. 14.7.

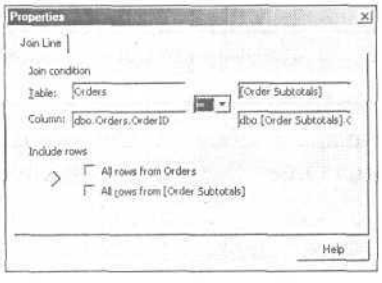


Рис. 14.7. Страница свойств линии объединения определяет настройки объединения таблиц

3. Щелкните мышью по стрелочке, расположенной после знака равенства. Обратите внимание на возможность вставки различных операторов сравнения. Если вы поставите галочку на «Все строки с Orders» (All Rows from Orders), вы создадите левое внешнее объединение. Если вы поставите галочку на «Все строки с [Order Subtotals]» (All Rows from [Order Subtotals]), вы создадите правое внешнее объединение. Если вы поставите обе галочки, то создадите полное внешнее объединение. Закройте окно объединения, а затем окно представления. Ответьте Нет на предложение о сохранении сделанных изменений.

4. Находясь в разделе **Запросы,** дважды щелкните мышью на **Создать представление в режиме конструктора** (Create View in Designer) - откроется диалоговое окно **Добавление таблицы,** показанное на рис. 14.8.

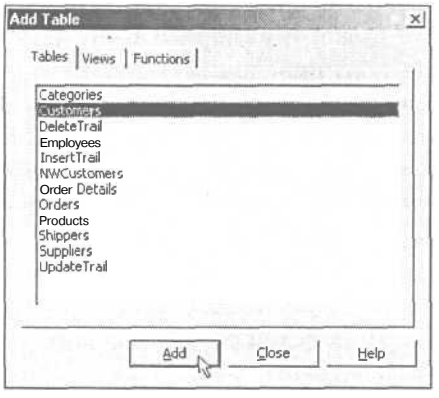


Рис. 14.8. Диалоговое окно «Добавление таблицы» в конструкторе запросов позволяет добавлять таблицы, представления или определенные пользователем функции в диаграмму вашего представления

5. Дважды щелкните мышью по таблицам Customers, Orders и Order Details, чтобы перенести их в конструктор запросов, а затем нажмите **Закрыть.** Обратите внимание, что линии объединения появляются автоматически, поскольку связи уже установлены.

Не путать конструктор запросов и конструктор базы данных, который похож на окно связей в обычной базе данных Access.

6. Выберите CompanyName и ContactName из таблицы Customers, OrderDate из таблицы Orders и ProductID и Quantity из таблицы Order Details, отмечая их слева галочкой.
7. В пустой строке сразу после Quantity напишите `[Order Details].Quantity * [Order Details].UnitPrice`. В столбце **Псевдоним** (Alias) напишите `ExtPrice`.
8. Запустите представление так же, как вы запускаете запрос в Access. Программа попросит сначала сохранить представление. Сохраните его под именем Sample View. Обратите внимание на поле, полученное в результате обсчета: ExtPrice.
9. Закройте представление.

Сохраненные процедуры

На первый взгляд сохраненные процедуры выглядят немного как гибрид Visual Basic и SQL. И опять-таки сохраненные процедуры имеют свои собственные уникальные характеристики. Transact-SQL (T-SQL) - это расширение языка

программирования баз данных SQL в SQL Server. *Сохраненная процедура* – это прекомпилированная программа на T-SQL, содержащая SQL-выражения и необязательные операторы контроля хода выполнения программы (такие, как выражения `if-then-else`). Вы можете использовать входные параметры, возвращать значения и даже вызывать другие сохраненные процедуры из вашего T-SQL-кода. Они схожи с запросами на действие в Access, но могут выполнять гораздо более широкий диапазон операций с базами данных.

Существует много причин, по которым следует применять сохраненные процедуры. Некоторые их преимущества приведены ниже.

- **Увеличение быстродействия.** Сохраненная процедура компилируется сервером при создании, поэтому она выполняется быстрее, чем отдельное SQL-выражение.
- **Простота использования.** вы можете запустить несколько SQL-выражений из одной сохраненной процедуры (разумеется, последовательно. - *Пер.*). Вы можете вызывать другие сохраненные процедуры из вашей, что упрощает написание сложных выражений. Вместо того чтобы копировать и вставлять выражения SQL в VBA-программу, вы можете сохранить эти выражения SQL в сохраненной процедуре и вызывать их так, как если бы они были функциями, записанными в модуле.
- **Больше опций безопасности.** Если нужно, вы можете отградить пользователей от возможности прямого доступа к таблицам. При этом можно позволить им обращаться к сохраненным процедурам, даже если у них нет прав на доступ к таблицам или представлениям, на которые эти процедуры ссылаются.

Вы можете создавать и записывать ваши собственные сохраненные процедуры в проекте Access используя параметры. Выполните следующие инструкции, чтобы запустить параметризованную сохраненную процедуру.

1. В разделе **Запросы** выделите сохраненную процедуру Sales by Year.
2. Нажмите **Конструктор.** Обратите внимание, что в заголовке написано «сохраненная процедура». Также обратите внимание на условие отбора для столбца ShippedDate.
3. Запустите сохраненную процедуру.
4. В качестве первого параметра напишите 3/1/98. Для второго - 3/1/99. Вы должны увидеть записи, соответствующие этим параметрам.
5. Закройте сохраненную процедуру.

Вы должны использовать сохраненные процедуры, если хотите произвести такие операции, как обновление, вставка, добавление или создание таблицы (Update, Insert, Append, Make-Table). На самом деле, если вы использовали преобразование этих типов запросов из базы данных Access, они трансформировались в сохраненные процедуры. Однако было бы разумным протестировать эти полученные преобразованием сохраненные процедуры, чтобы убедиться, что они изменены корректно.

Сохраненные процедуры для начинающих

Сохраненные процедуры, представленные здесь, разработаны для проекта «Northwind». Предположим, имеется выражение SQL, которое вы хотите преобразовать в сохраненную процедуру:

```
Select UnitPrice, Quantity From [Order Details]
```

Если вы нажмете **Создать** в разделе **Запросы** и выберете **Ввод сохраненной процедуры** (Create Text Stored Procedure), вы сможете записать новую процедуру, которая будет выглядеть примерно так:

```
CREATE PROCEDURE ProcSamp
AS
Select UnitPrice, Quantity From [Order Details]
```

Название процедуры может быть любым. Вы можете вызвать ее используя ее имя. Нет ничего проще и незатейливее, чем это. Даже без использования ключевого слова RETURN вам будет возвращено два столбца. Ключевое слово RETURN может либо возвращать значение, либо вызывать выход из программы, в зависимости от того, что вам нужно.

Теперь добавим немного новых возможностей, не слишком увлекаясь. Возьмем, допустим, переменную. Просто поставьте перед переменной знак @, точно так же, как вы делали это для параметра. Разница только в том, что вам нужно объявить ее. С переменной нужно что-то делать, поэтому добавим оператор WHERE, и все это будет выглядеть следующим образом:

```
ALTER PROCEDURE ProcSamp
AS
declare @i as int
select @i=40
Select UnitPrice, Quantity From [Order Details]
Where Unitprice >@i
Return
```

Обратите внимание на пару нюансов. Во-первых, вместо CREATE в CREATE PROCEDURE стоит ALTER. Таким образом, вместо того чтобы создавать процедуру заново, вы изменяете уже созданную (в предыдущем примере. - *Пер.*). Во-вторых, заметьте, что оператор WHERE использует переменную в чистом виде, без всяческих ритуальных приседаний с расстановкой кавычек и амперсандов, как это было, когда вы использовали выражения SQL в VBA. И наконец, обратите внимание на использование ключевого слова RETURN. В данном случае оно производит всего лишь выход из процедуры, но оно также легко могло бы быть использовано для возвращения значения.

Чтобы преобразовать это в программу на T-SQL, использующую параметр, нужно просто изменить несколько строк:

```
ALTER PROCEDURE SampProc
    (
   @i  int
    )

AS
Select UnitPrice, Quantity From [Order Details]
```

```
Where Unitprice >@i
Return
```

Обратите внимание, что ключевое слово AS находится по другую сторону параметра. Другими словами, переменная стоит *после* ключевого слова AS, в то время как параметр стоит перед ним. Также обратите внимание, что параметру не присвоено значения, поскольку это будет необходимо сделать только после запуска.

В случае, если вас интересует контролирование хода программы, рассмотрим последнюю сохраненную процедуру. Она немного сложнее, но будет полезной для понимания того, как происходит контроль хода программы для сохраненных процедур.

```
ALTER Procedure SelectOwners
@Titlechar(20)
As
--Если найдены записи, соответствующие введенному параметру, вывести их
--и установить возвращаемое значение (RETURN) равным 1
--When prompted, type "Owner"
IF (SELECT COUNT(*) FROM Customers WHERE ContactTitle=@Title)>0
    BEGIN
        SELECT ContactName, ContactTitle
        FROM Customers
        WHERE ContactTitle=@Title
        RETURN 1
    END
ELSE
--Иначе установить возвращаемое значение равным 0
        RETURN 0
```

Этот листинг демонстрирует использование ключевого слова RETURN для возвращения значения. Он также демонстрирует использование контроля над ходом программы в сочетании с выражениями SQL для определения того, какое из двух выражений SQL будет выполнено. Вот такого рода возможность теперь стала доступна вам.

Триггеры

Триггер - это специфический тип сохраненной процедуры SQL Server. Так же как программы на Visual Basic отвечают на события, триггеры отвечают на изменения в таблице. В SQL Server существует три типа триггеров:

- **триггер вставки** - срабатывает при добавлении новой записи в таблицу,
- **триггер удаления** - срабатывает при удалении записи из таблицы,
- **триггер обновления** - срабатывает при обновлении записи в таблице.

Чтобы добавить триггер в таблицу, просто щелкните на ней правой кнопкой мыши и выберите «Триггеры». Затем нажмите **Создать.** Вы должны увидеть то, что показано на рис. 14.9.

Написав и сохранив код программы, вы таким образом прикрепляете его к вашей таблице. Если в дальнейшем вам понадобится отредактировать или удалить триггер, вы можете щелкнуть правой кнопкой по той же самой таблице

и выбрать «Триггеры». Затем выбрать необходимый триггер из ниспадающего списка и нажать кнопку **Изменить** или Удалить.

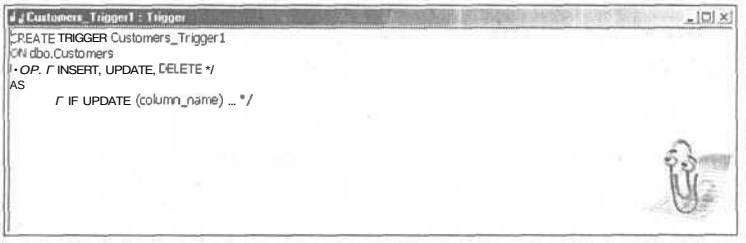


Рис. 14.9. Создан новый триггер в редакторе инструкций SQL и в нем по умолчанию записано для вас выражение SQL

Бывает полезно посылать сообщения пользователю, чтобы напомнить ему о последствиях изменений таблицы, но помните, что инструкция SQL Server PRINT не работает в проектах 2002. Предположим, например, что вы хотите послать пользователю сообщение из триггера. Выражение PRINT выглядело бы следующим образом:

```
PRINT 'The updated record has been copied into the UpdateTrail table'
```

Этот триггер сработает, но ничего не произойдет, поскольку выражение PRINT не может посылать сообщения. Если вы хотите отправить пользователю сообщение, когда сработает ваш триггер, вам нужно использовать RAISERROR, как показано в следующем примере:

```
CREATE TRIGGER NewCount
ON dbo.Customers AFTER DELETE
AS
    DECLARE @RecNum Int
    SELECT @RecNum = COUNT(*) FROM Customers

RAISERROR ('The new record count is %i', 16, 1,@RecNum)
```

Синтаксис RAISERROR может показаться слишком сложным для начинающего, не говоря уже о том, что это не входит в компетенцию данной книги. За дополнительной информацией обращайтесь к SQL Server *Books Online (BOL)* (электронная книга по SQL Server), которую можно скачать по ссылке www.microsoft.com/SQL/techinfo/productdoc/2000/books.asp. Будьте готовы выпить чашечку кофе: загрузка займет некоторое время, в зависимости от вашей системы.

Иногда вам может понадобиться запротоколировать все удаления, вставки или обновления полей, произведенные в данной таблице. Ниже приведены триггеры, использующие копию таблицы Customers проекта Northwind, называющуюся NWCustomers, позволяющие это сделать:

```
ALTER TRIGGER TrackDeletes
ON dbo.NWCustomers AFTER DELETE
AS
    INSERT INTO DeleteTrail
    Select CustomerID, CompanyName, ContactName, ContactTitle,
    Address, City, Region, PostalCode, Country, Phone, Fax
```

```
   FROM deleted
GO

ALTER TRIGGER TrackInserts
ON dbo.NWCustomers AFTER INSERT
AS
   INSERT INTO InsertTrail
   Select CustomerID, CompanyName, ContactName, ContactTitle,
   Address, City, Region, PostalCode, Country, Phone, Fax
   FROM inserted
GO

ALTER TRIGGER TrackUpdates
ON dbo.NWCustomers AFTER UPDATE
AS
   INSERT INTO UpdateTrail
   Select CustomerID, CompanyName, ContactName, ContactTitle,
   Address, City, Region, PostalCode, Country, Phone, Fax
   FROM deleted
GO
```

Эти программы используют временные таблицы *deleted* (дословно «удаленное») и *inserted* («вставленное»), которые имеют ту же структуру (поля и типы данных), что и первоначальная таблица. Таблица deleted добавляется к таблице DeleteTrail (дословно «История удаления»), а таблица inserted - к InsertTrail («История вставки»). Обратите внимание, что таблица deleted используется также в триггере TrackUpdates, который заполняет таблицу UpdateTrail («История обновления»). Хотя они и записаны вместе, это три отдельных триггера, каждый из которых можно найти в меню «Триггеры». За исключением нескольких операторов, эти триггеры идентичны. Для каждой записи, добавляемой к любой из трех таблиц-протоколов, можно легко дописать дату и время.

Попробуйте потестировать эти триггеры, выполнив следующие инструкции:

1. В проекте «Northwind» скопируйте таблицу Customer, щелкнув по ней правой кнопкой мыши и выбрав **Копировать.** На панели инструментов нажмите **Вставить** - откроется диалоговое окно **Вставка таблицы.**
2. Выберите **Только структура** и назовите эту таблицу DeleteTrail.
3. Выполните шаги 1 и 2, чтобы создать еще две таблицы, которые будут называться InsertTrail и UpdateTrail.
4. Щелкните правой кнопкой мыши на таблице Customers, выберите **Триггеры, Создать.**
5. Перепишите триггер DeleteTrail из приведенного примера. Закройте и сохраните его.
6. Повторите шаги 4 и 5 для триггеров InsertTrail и UpdateTrail, аккуратно и без ошибок переписывая программу.
7. Протестируйте заданную программу, вставив запись-пустышку (двух полей будет достаточно). Закройте таблицу.
8. Обновите запись, которую вы вставили, написав какое-нибудь слово в одно из его полей, а затем удалите запись, которую вы только что обновили.
9. Посмотрите, что получилось в таблицах DeleteTrail, InsertTrail и UpdateTrail.

Декларативная целостность данных

Способность устанавливать целостность ссылочных данных при помощи ограничений внешнего ключа (foreign key constraints), определяемых вами, называется *декларативной целостностью данных* (Declarative Referential Integrity, DRI). DRI становится фактически частью определения таблицы. Вы можете использовать другие методы, например триггеры, для установления связей. Это называется *процедурной целостностью данных*. Используя Access 2002, вы можете производить каскадное обновление и удаление в проектах Access. Следующие инструкции описывают, как можно увидеть DRI-связи.

1. В проекте NorthwindCS, в разделе **Схемы баз данных** окна **Проект,** выделите Relationships (дословно «Взаимосвязи». - *Пер.)* и нажмите **Конструктор,** чтобы увидеть связи DRI.
2. Щелкните правой кнопкой мыши на связи между Employees и Orders и выберите **Свойства** - откроется страница свойств.
3. Переключитесь на вкладку **Связи** («Relationships») и обратите внимание на последние две опции, называющиеся **Каскадное обновление связанных полей** (Cascade Update Related Fields) и **Каскадное удаление связанных полей** (Cascade Delete Related Fields), которые показаны на рис. 14.10.

Рис. 14.10. Вы можете использовать новые опции «Каскадное обновление» и «Каскадное удаление» для связей в проектах Access 2002

Как вы видите, проекты Access - мощное и практичное средство. Проекты Access расширяют функциональные способности Access, обеспечивая создание клиент-серверной архитектуры.

Что дальше?

Теперь, когда вы попробовали, что можно сделать при помощи интеграции с SQL Server, вы можете продолжить изучение и узнать об интеграции с другими типами программ. Следующая глава охватывает технологию OLE и некоторые другие новые возможности, добавленные в Access 2002.

15

Удобные для пользователя расширенные возможности

В гл. 14 вы изучили, как объединять Access с SQL Server. Эта глава обращает внимание на расширенные возможности, удобные для пользователя. Некоторые из этих возможностей упрощают интеграцию с другими приложениями, с такими, как, например, Microsoft Excel. В частности, вы изучите следующее:

- опцию сжатия в Access,
- почему сжатие такважно?,
- преобразование документов предыдущих версий Access,
- объединение с другими приложениями,
- как использовать Object Linking (Связывание объектов) и Embedding (Внедрение),
- как объединяться с другими приложениями используя код.

Парадокс баз данных

В течение нескольких лет Microsoft Access получил мириады заслуженных отличных отзывов от IT-профессионалов и пользователей. Однако критицизм повис, как альбатрос, на Access начиная с его года выпуска в 1992 году. Люди неохотно делали решительный шаг в сторону более устойчивых и расширенных функциональных возможностей, связанных с базами данных Access, отказываясь от ограниченных возможностей Excel, так как процесс обучения требовал именно решительного перехода от одного к другому. Можно лишь только размышлять, будет ли Access иметь такой же успех без внедрения многих удобных пользователю обновлений. Тем не менее, когда уже кажется, что люди проявляют все больший интерес к могуществу Access, некоторые еще до сих пор не убеждены в предоставляемых им удобствах.

У вас есть или мощность, или простота использования. Предполагается, что у вас не может быть одновременно и того и другого. Это и есть парадокс баз данных. (Разумное определение тех баз данных, которые претендуют на оба критерия.) Access 2002 обозначил этот парадокс некоторыми «удобными для пользователя» дополнительными возможностями.

Расширенные возможности сжатия баз данных

Поразителен тот факт, что такие важные особенности так часто упускаются из виду. Так что вам не придется полагаться на свою память, начиная работу с Access, пользователь может выбрать опцию Compact on Close (Сжатие при закрытии).

После типичного сеанса работы с Access, который обычно включает в себя такие процессы, как создание таблиц, удаление записей и т. д., база данных на-

чинает расти в размерах, занимая немало места на диске. Важно отметить, что место, занятое удаленными записями, таблицами или другими объектами, не освобождается автоматически для новых объектов. Сжатие не только освобождает место для новых объектов, но и уменьшает общий размер базы данных. Поэтому лучше проводить сжатие перед архивацией.

Насколько часто следует проводить сжатие? Это зависит от того, сколько работы вам приходится выполнять. Вероятно, для большинства пользователей, опция Compact on Close (Сжатие при закрытии) - неплохой вариант. Регулярное сжатие гарантирует обновленные статистические данные и оптимальную эксплуатацию базы данных.

Вы можете проверить, как сжатие влияет на размер файла базы данных, посмотрев mdb-файл (например, MyDatabase.mdb) в Windows Explorer до и после сжатия. Если база данных большая по размеру, то разница будет ощутимая. Еще одно преимущество: Access исправляет базу данных одновременно с операцией сжатия. Даже если Access не обнаружит, что база данных повреждена, странное поведение внутри Access может быть хорошим индикатором необходимости сжатия базы данных.

Если вы опасаетесь, что можете забыть проводить операцию сжатия периодически, вы можете выбрать опцию Compact on Close (Сжатие при закрытии), которая влияет только на ту базу данных, с которой вы работаете. Следуйте приведенным ниже указаниям для активизации опции Compact on Close (Сжатие при закрытии).

1. Откройте базу данных, которою хотите сжать.
2. Щелкните Tools в меню и затем щелкните Options для того, чтобы открыть диалоговое окно Options.
3. Щелкните по закладке General в диалоговом окне Options.
4. Выберите, щелкнув мышью, опцию Compact on Close (Сжатие при закрытии), как показано на рис. 15.1, и затем щелкните OK.

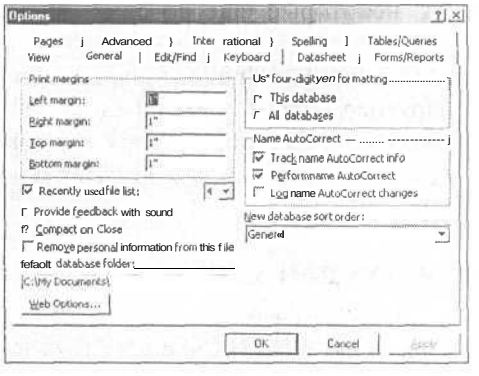


Рис. 15.1. Установите опцию Compact on Close (Сжатие при закрытии) для автоматического сжатия вашей базы данных каждый раз при ее закрытии

Корпорация Microsoft заявляет, что в Access 2002 добавлены большие функциональные возможности сжатия и исправления. Например, файлы с поврежденными формами и отчетами чаще исправляются при помощи последнего обновления. Чтобы активизировать эту опцию, просто выберите в меню Tools, Database Utilities, Compact and Repair Database. Теперь вы можете сжимать и исправлять используя VBA(Visual Basic for Applications - «Visual Basic для прикладных программ») посредством нового метода CompactRepair.

Конвертирование объектов, созданных в предыдущих версиях Access

Если происходит ошибка при конвертировании базы данных из других версий (таких, как Access 95, Access 97 или Access 2000) в базу данных Access 2002, создается таблица, которая протоколирует информацию о каждой ошибке. Это упрощает процесс обнаружения и решения проблем в конвертированных базах данных.

Access 2002 использует по умолчанию формат файла Access 2002, когда вы создаете новую базу данных. Это означает, что версии Access 2000 и 2002 могут использовать и модифицировать одну и ту же базу данных Access 2000, гарантируя таким образом совместимость с существующими приложениями Access 2000, когда организация решит перейти на Access 2002.

К тому же, если вы хотите конвертировать в новый опциональный файловый формат Access, вам будет предоставлен более быстрый доступ к данным и более быстрая обработка информации, особенно это касается больших баз данных. В итоге, пользователь может оставить файл в формате Access 2000 или модернизировать в формат файла Access 2002. Это очень необычно. Для преобразования нужно просто выбрать в меню Tools опцию Convert Database. Выберите эту опцию, а затем отметьте пункт To Access 2002 Format.

Совместное использование и интеграция с другими приложениями

Существует потребность конвертирования не только из предыдущих версий Access, но и из других документов Office, таких, как Excel. К тому же существует потребность совместного использования данных с приложениями, отличными от Office, и с большими базами данных, такими, как базы данных Oracle. Но программы Office используют совместно многие схожие опции и ресурсы, делая процесс конвертирования почти безвредным. Эта функциональная возможность делает процесс передачи данных чрезвычайно привлекательным.

Например, предположим, что вы знаете, что Access поддерживает адресные ярлыки(mailing labels) посредством мастера ярлыков (Label Wizard), в который можно зайти щелкнув по опции New в меню Reports (Отчеты). К тому же вам нравятся функциональные возможности опции автоматического составление стандартных писем в Microsoft Word. Итак, вы решили передать список адресатов в Microsoft Word, который также может оперировать адресными ярлыками. Когда вы щелкнете по таблицам (Tables) в разделе Объекты (Objects), кнопка Связи с Office (Office Links) активизируется с опцией Слияние с MS Word

(Merge It with Microsoft Word). Эта опция предоставляет простой способ передать список адресатов в Word. Кстати, вы можете создавать стандартное письмо с использованием фиксированных форм в отчете Access.

Использование опции Drag and Drop («вытяни и положи»)

Корпорация Microsoft максимально упростила процесс передачи таблицы Access в Word в целях составления стандартных писем. Просто откройте одновременно оба приложения. Разверните новое окно Microsoft Word до максимального размера и восстановите до меньших размеров развернутое во весь экран окно Access (средняя кнопка в верхнем правом углу окна). В окне базы данных (вам не нужно открывать таблицу) щелкните по таблице и, не отпуская кнопку мыши, перетащите таблицу в окно Word; отпустите кнопку мыши (рис. 15.2). Удалите заглавную строку, для выделите ее и выберите в меню Table, Delete, Row. Сохраните файл и используйте его в дальнейшем как источник данных для составления стандартных писем с использованием фиксированных форм. Это все, что нужно выполнить; во всяком случае со стороны Access.

Со стороны Word нужно запустить Mailing Label Wizard (Мастер писем) в разделе меню **Файл** (File), New и затем выбрать закладку Letters & Faxes (Письма&Факсы). Выберите Mailing Label Wizard среди доступных опций и просто выполните действия согласно подсказкам мастера. Если вы хотите создать форму письма, выберите Tools и Mail Merge в меню Word и затем следуйте подсказкам мастера.

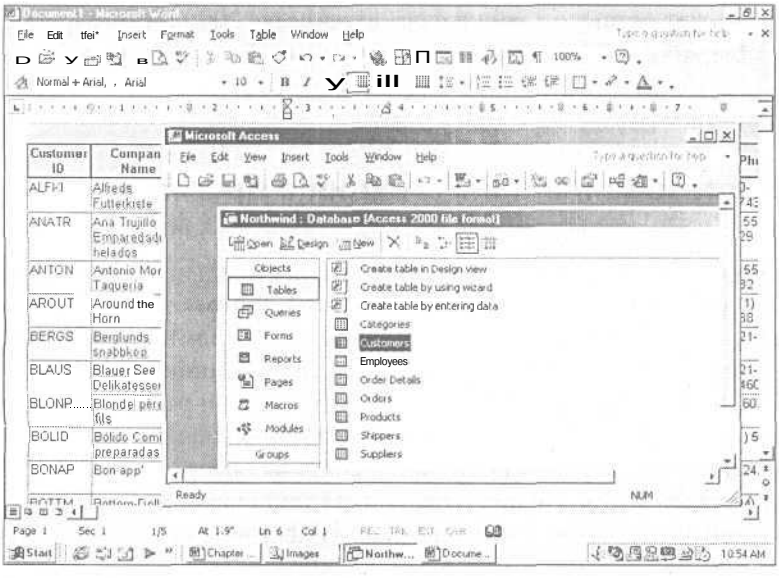


Рис. 15.2. Вы можете перетащить и поместить напрямую из Access в Word для создания источника данных для стандартных писем

ЗАМЕЧАНИЕ. Метод drag-and-drop с таким же успехом можно использовать и в Excel.

Копирование и вставка

Если вы хотите более простую альтернативу выбору File, Get External Data и прохождению через Import Wizard (Мастер импорта файлов) для импортирования данных Excel, вы должны иметь в виду возможность копирования и вставки области Excel (drag-and-drop здесь также будет работать). При использовании этой альтернативы обратите внимание на следующие моменты:

- создание заголовка столбца для каждого поля;
- убедитесь, что нет линий пробелов между заглавным рядом и первым рядом данных;
- нужно избегать пустых столбцов;
- включать только нужные вам данные в область, которую вы копируете;
- Ответьте Yes(Да) на следующий вопрос: «Содержит ли первый ряд ваших данных заголовки столбцов?»

Для того чтобы скопировать из Microsoft Excel и вставить в Microsoft Access, убедитесь, что оба приложения открыты. Выполните следующие пункты:

1. Активизируйте окно Excel и затем выделите нужную область ячеек, которую вы хотите перенести.
2. Нажмите Ctrl+C для копирования в буфер. Активизируйте окно базы данных Access и затем нажмите Ctrl+V для вставки.
3. Выберите Yes в появившемся диалоговом окне, как показано на рис. 15.3.

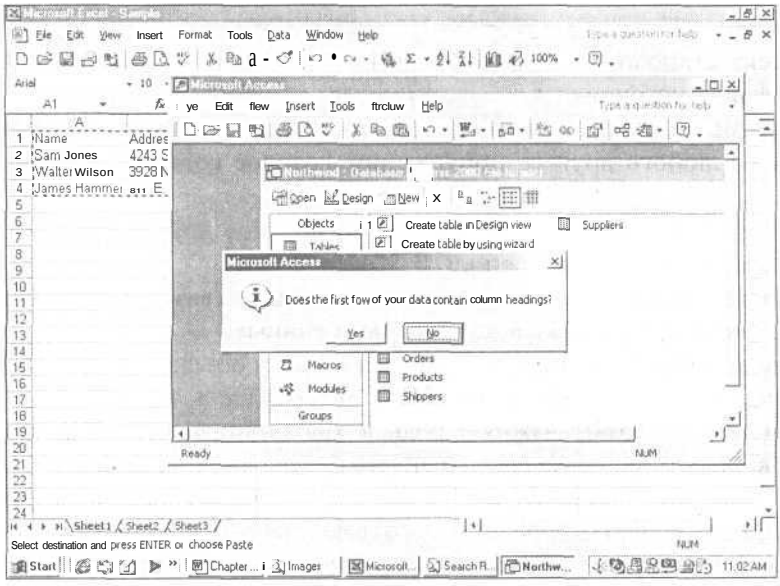


Рис. 15.3. Вы можете скопировать в буфер и вставить из буфера базу данных напрямую из Excel в Microsoft Access

Использование OLE (Связывание и внедрение объектов)

В Access 2002 Microsoft по большей части концентрирует свои усилия на улучшении интеграции с Web- и SQL-Server приложениями. Однако существуют и другие типы интеграции. Слово *интеграция* главным образом означает сборку воедино с целью образования единого целого и совместной работы приложений. Когда дело доходит до интеграции Access с другими приложениями, то в этом случае интеграция есть нечто большее, чем просто копирование таблицы в другое приложение.

ЗАМЕЧАНИЕ. Узнать больше об улучшении интеграции Web- и SQL-Server приложений вы можете в гл. 13 и 14.

Нужно понять два типа интеграции приложений. Когда созданный в одном приложении объект добавляется в файл, созданный в другом приложении, то этот процесс называется *связыванием* и *внедрением объектов* (OLE). Вы можете как *внедрить,* так и *связать* объект из внешнего источника. Главное различие между связанным и внедренным объектом заключается в том, где хранятся их данные. Могут быть полезны следующие моменты.

В случае связанного объекта поддерживается соединение между файлом-источником и файлом назначения так, что связанный объект *обновляется,* когда данные из источника меняются.

ЗАМЕЧАНИЕ. Файл-источник - это тот файл, в котором объект был создан. Файл назначения - это тот файл, в который вы помещаете объект из файла-источника.

Внедренный объект становится частью файла назначения и *не* обновляется в случае изменения файла-источника.

Когда объект связан с другим приложением, то текущие данные объекта можно просмотреть из любого другого приложения, которое содержит ссылку на эти данные.

Когда объект внедряется в другое приложение, то никакое другое приложение не имеет доступа к данным из внедренного объекта.

Когда вы связываете объект, то вы вставляете указатель на связанный объект в ваше приложение, вместо того чтобы вставлять сами данные.

Когда вы встраиваете объект, то все относящиеся к этому объекту данные *копируются* в контейнерный элемент управления OLE и *содержатся* в нем.

Последние два момента играют важную роль в хранении данных. Так как встроенные данные копируются, когда вы сохраняете содержимое не привязанного ни к чему элемента управления в mbd-файле Access, то сохраненный файл будет содержать имя приложения, которое создало объект, данные объекта и образ объекта в виде метафайла. Таким образом, внедренные объекты могут значительно увеличить размер файла. Связанные объекты, наоборот, не так сильно увеличивают размер файла. Чтобы доказать это утверждение, по мере изучения OLE вы можете попробовать вставить документ Microsoft Word в отчет:

1. Откройте любой mbd-файл в Access с помощью проводника, а затем откройте любой отчет в окне конструктора.
2. На инструментальной панели (индикатор мастера элементов управления должен быть выделен) выберите Unbound Object Frame (Рамка непривязанного объекта). Побольше растяните прямоугольник, чтобы создать рамку объекта.
3. Когда мастер рамки непривязанного объекта активизируется, выберите опцию Create (Создать) из меню File (Файл), как показано на рис. 15.4.

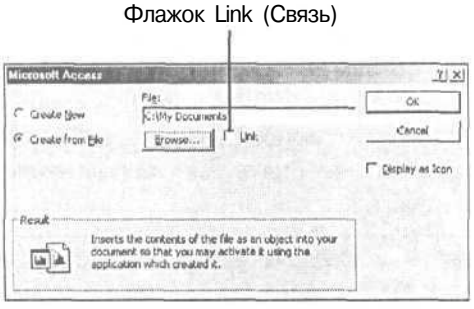


Рис. 15.4. Если вы не отметите галочкой опцию Link, когда вы вставляете объект в отчет, то объект будет автоматически внедрен

4. Нажмите кнопку Browse (Обзор), а затем выберите документ Word объемом примерно в одну страницу.
5. Выбрав файл, не отмечайте опцию Link; закройте и сохраните отчет. Сожмите базу данных.
6. Закройте базу данных, чтобы вернуться в проводник, и наберите размер файла. Если вы не видите размера файла, то выберите опцию Details (Детали) из ниспадающего меню Views (кнопки), чтобы появились следующие параметры: Name (Имя), Size(Размер), Type (Тип) и Modified (Изменения).
7. Откройте заново ту же базу данных и проделайте еще раз действия с 1-го по 6-е, но на этот раз удалите рамку предыдущего объекта перед тем, как вставить новую, и отметьте галочкой опцию Link.
8. После закрытия базы данных опять проверьте размер файла. Файл теперь должен быть меньше.

Так как вы не отмечали опцию Link в первый раз, то объект был внедрен. Вы заметили бы еще большую разницу, если бы объект был графическим файлом, который еще больше по размеру. Если вы скопировали и открыли mdb-файл из Access, содержащий внедренный графический объект, на другой компьютер без изначального объекта, то вы все равно сможете открыть отчет и просмотреть объект. Однако если бы вы привязали графический объект, то вам бы было предложено найти этот объект на диске, после попытки открыть отчет, ибо первый есть копия, а последний есть указатель.

По вышеуказанным причинам имеет смысл внедрять графические объекты и привязывать файлы-документы. В общем документы, созданные в таких программах, как Word или Excel, обычно требуют обновления чаще, чем графические файлы, что делает их более приспособленными к ссылкам (привязкам). Графическим файлам, с другой стороны, зачастую требуется быть частью файла, в который они инкапсулированы, что делает их более приспособленными к внедрению. Тем не менее если графические файлы огромны, то вам следует помнить, что размер вашей базы данных может значительно увеличиться.

Ситуации могут быть разными. Вы должны взвесить все «за» и «против» и решить, какой способ использовать. Если вы хотите, чтобы изменения, сделанные в программе-источнике, отражались в программе назначения, то используйте привязку. Если вы хотите, чтобы копия исходного объекта была в вашем Access-файле, то используйте внедрение. Если вы беспокоитесь о слишком большом увеличении размера вашего Access-файла, то опять-таки используйте привязку. Все зависит от того, что вы хотите сделать.

ЗАМЕЧАНИЕ. Привязываете ли вы объект или внедряете его, вы в любом случае можете открыть программу-источник просто дважды щелкнув мышью по объекту в Access. Это относится только к конструктору формы или отчета.

Перед тем как продолжить, следует упомянуть о последней детали касательно OLE. Иногда термины претерпевают обновление по мере изменений в технологии. Dynamic Data Exchange (Динамический обмен данными) (DDE) есть фундамент, на котором был выстроен OLE версии OLE 1.0 (1991). Однако даже Microsoft был вынужден признать, что DDE имел свои ограничения. Неожиданно появилась Component Object Model (COM, модель компонентных объектов), которая сыграла роль движущей силы и нового стандарта до выпуска новой версии OLE 2.0 в 1993 году. Это не только разрешило множество проблем, унаследованных еще от DDE, но и предопределило множество новых продвинутых и новаторских технологий, таких, как элементы управления. Но даже при таких условиях вы все еще можете найти ссылки на DDE в файле помощи Access.

Весь этот прогресс означает, что технология, которая стоит за всем тем, что называют «объектами составных документов», претерпела значительные улучшения за все эти годы. И это отличная новость для всех пользователей. Теперь у вас имеется даже больше возможностей для взаимной интеграции приложений. Но будьте осторожны, ибо в пособиях и руководствах эти термины иногда взаимозаменяют друг друга.

Использование кода

Было бы очень кстати обратиться к Excel лишь для того, чтобы дать вам представление о том, как интегрировать приложения при помощи VBA. Объект Application находиться на вершине иерархической лестницы объектов Excel. Через него вы можете обращаться к объектам Excel, таким, как Workbook(Рабочий журнал) и Worksheet(Планшет). Объект Workbook представляет файл Excel с расширением xls в проводнике, тогда как Worksheet представляет лист в рабочем

журнале. Объект ActiveCell важен для навигации по электронной таблице. Чтобы переместить три строки вниз и одну вправо, вы используете свойство Offset объекта ActiveCell:

```
ActiveCell.Offset(3, 1).Select
```

После выполнения выборки вы можете скопировать эту выборку, как это сделано с помощью кода в следующем примере:

```
Selection.Copy
```

Вы можете использовать свойство Range для копирования и объект ActiveSheet для вставки:

```
Sub SampleXL()
Range("A1:A4").Copy
ActiveCell.Offset(5, 0).Select
ActiveSheet.Paste
End Sub
```

Следующая процедура выбирает диапазон ячеек в Excel, а затем копирует этот диапазон ячейку за ячейкой в таблицу, называемую FromExcel в Access. Как поясняется в комментариях, эта программа должна быть запущена из-под модуля Excel.

```
Sub ExcelToAccess()
Dim cn As ADODB.Connection
Dim rs As ADODB.Recordset
Dim rg As Range
'Эта процедура должна быть запущена из-под модуля Excel

Set cn = New ADODB.Connection
Set rs = New ADODB.Recordset
cn.ConnectionString = "Provider=Microsoft.Jet.OLEDB.4.0;" _
& "Data Source=c:\AccessByExample\ExportImport.mdb;"
cn.Open
rs.Open "FromExcel", cn, adOpenKeyset, adLockOptimistic

  For Each rg In Range("a2:a5")
    With rs
      .AddNew
      .Fields("Name").Value = rg.Value
      .Update
    End With
  Next rg

  rs.Close
  cn.Close
  Set rs = Nothing
  Set cn = Nothing
  Set rg = Nothing
End Sub
```

Теперь, когда вы немного познакомились с Excel, как все же запустить код Excel из Access? Во-первых, вы должны установить библиотеку объектов Excel 10.0 в диалоговом окне References (Ссылки), к которой вы обращаетесь, выбрав

из меню Tools пункт References на строке меню, когда вы находитесь в модуле Access, как показано на рис. 15.5

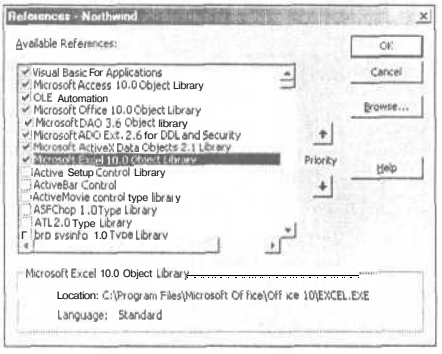


Рис. 15.5. Вы также можете сделать ссылки на другие приложения, такие, как Excel, выбрав в модуле пункт References из меню Tools

Далее декларируйте ваши переменные для объекта Excel Application. Следующий код исполняется из модуля Access, используя объект Excel Application. Так как диапазон устанавливается с помощью внутренней константы, если вы добавляете значения в этот диапазон в столбец A, то имя диапазона выставляется снова, подстраиваясь под новые значения. За исключением присваивания имени диапазону, в основном исполняется все та же операция, что и в предыдущей процедуре, но запускается она из-под Access. Процедура обращается к коллекции Excel Range, а затем добавляет каждую ячейку в таблицу Access.

```
Sub GetExcelData()
Dim ToAccess As String, rg As Range
Dim ExcelApp As Excel.Application
Dim cn As ADODB.Connection
Dim rs As ADODB.Recordset

Set cn = CurrentProject.Connection
Set rs = New ADODB.Recordset
rs.Open "FromExcel", cn, adOpenKeyset, adLockOptimistic
Set ExcelApp = New Excel.Application

ExcelApp.Workbooks.Open "c:\My Documents\Sample.xls"
ExcelApp.Visible = True
ExcelApp.Range("a2").Activate
ToAccess = ExcelApp.Range(Selection, Selection.End(xlDown)).Address
ActiveWorkbook.Names.Add Name:="ToAccess", RefersTo:="=" & ToAccess
Set rg = ExcelApp.Range("ToAccess") 'Uses the named range "ToAccess"
  For Each rg In rg
    With rs
      .AddNew
      .Fields("Name").Value = rg.Value
      .Update
    End With
  Next rg
```

```
rs.Close
cn.Close
Set rs = Nothing
Set cn = Nothing
Set rg = Nothing

End Sub
```

Что дальше?

Теперь вы открыли для себя методы интеграции других приложений с Access. В следующей главе вы узнаете о запросах еще больше и у вас будут способы преодолеть трудности при разработке.

Часть V
Обход ограничений Access

16

Преодоление ограничений запросов

В прошлой главе вы узнали об интеграции Access с другими приложениями. В этой главе вам предлагается методика, которую вы можете использовать для преодоления ограничений запросов. Так как запросы - очень важная составляющая Access, не будет лишним узнать о них как можно больше. В частности:

- вы узнаете о методике построения запросов, основанных на таблице примеров;
- как использовать подзапросы для преодоления ограничений запросов;
- познакомитесь с дополнительной информацией о том, как использовать функции с запросами.

Проблемы запросов

Ничто не раскрывает возможностей базы данных лучше, чем запрос. Запросы даже используются в Access для выполнения на заднем плане многих задач без вмешательства пользователя - пользователь может даже не знать об этом. Хотя существуют ситуации, которые могут затруднять разработчика и могут быть по меньшей мере проблемными. Обычно запросы - первое, к чему обращается разработчик за помощью. Однако не всегда лучший способ применения этого мощного инструмента очевиден.

Представьте, что вы работаете в страховой компании и ваш босс обращается к вам с тем, что они в страховом бизнесе называют «поуровневым отчетом». Он хочет, чтобы денежные суммы были сгруппированы по уровням начиная с самых малых. Для того чтобы определить частоту убытков, вы должны указать число исков в этих различных уровнях. Для того чтобы определить степень ущерба, вы должны указать денежные суммы в различных уровнях.

Происходит это примерно так: «Сколько появилось исков на суммы между 1 и 500 долл. и какова была итоговая сумма для этих величин? Сколько появилось исков на суммы между 501 и 1000 долл. и какова была итоговая сумма для этих величин?» Так продолжается до того момента, когда исковые суммы уже потенциально выражаются в миллионах. В результате у вас получается отчет, который выглядит примерно так, как в таблице 16.1

Таблица 16.1. Поуровневая таблица убытков

Уровень, долл.	Число исков	Ущерб, долл.
1-500	1,206	248,011
501-1,000	236	156,004
1,0015,000	269	671,794
5,001-10,000	116	852,609
10,001-25,000	129	2,094,949
25,001-50,000	65	2,272,332
Более 50,001	34	3,440,358

Использование запросов, основанных на таблице примеров, вместо сценариев If - Then

Одно из решений - использование строки вложенных IIF-функций в запросе на обновление. Выглядело бы это примерно так:

```
IIF([Incurred]>=1 and ([Incurred]<=500,[Level]=1,IIF([Incurred]>=501 and
[Incurred]<=1000,[Level]=501,IIF([Incurred]>=1001 and [Incurred]<=5000,
[ic:ccc][Level]=1001,IIF([Incurred]>=5001 and
[Incurred]<=10001,[Level]=5001,
[ic:ccc]IIF([Incurred]>=10001 and [Incurred]<=25000,[Level]=10001,
[ic:ccc]IIF(Incurred]>=25001 and [Incurred]<=50000,[Level]=25001,
[ic:ccc]IIF([Incurred]>50000,[Level]=50001
```

С этим решением связано несколько проблем. Во-первых, его очень утомительно записывать и сложно поддерживать в порядке. Предположим, вы хотите изменить уровни. Следующая проблема: что бы вы делали, если бы у вас было 40 уровней? Можете представить себе, какой длинной была бы строка IIF. Третья проблема состоит в том, что вы не выполняете никаких действий по обобщению (aggregate functions). Вы просто указываете, в каком уровне находится каждая запись. Вам пришлось бы возвращаться и «группировать» уровни, подводя итоги и учитывая финансовую информацию. Бывает, что функции не играют важной роли при работе с запросами, но здесь мы имеем другой случай.

К счастью, существует лучшее решение. Вы можете создать небольшую таблицу, выполняющую функции образца, на котором вы сможете построить ваш запрос. Рассмотрим этот простой метод на примере.

Прежде всего вам необходимо построить маленькую таблицу с двумя полями: Low_(Нижний) и High (Верхний). Low представляет минимальное число для каждого уровня, в то время как High представляет максимальное число. Это создает прекрасную основу для диапазона Between - And (Между ... и ...) в запросе. В результате у вас получается таблица, подобная табл. 16.2

Таблица 16.2. Таблица уровней

Low	High
1	501
501	1000
1001	5000
5001	10000
10001	25000
25001	50000
50001	100000
100001	250000
250001	500000
500001	1000000

Табл. 16.2 находится в базе данных ваших исков (Claims) вместе с таблицей, содержащей значения исков, называемой StratLosses. Следующие шаги раскрывают эту методику.

1. Откройте базу данных исков Claims из вашей папки AccessByExample.
2. **В Объектах** нажмите на **Запросы** для того, чтобы показать хранимые запросы.
3. Щелкните мышью два раза на **Создании запроса в режиме конструктора** - откроется диалоговое окно **Добавление таблицы.** Когда оно появится, щелкните два раза мышью на таблицах Strata и StratLosses и нажмите на **Закрыть.**
4. В конструкторе запроса щелкните два раза на поле Low в таблице Strata и тремя двойными щелчками на поле Incurred из таблицы StratLosses перенесите три раза это поле в сетку запроса (Query grid).
5. Нажмите кнопку **Групповые операции** (Totals, греческий символ «сигма») для выполнения групповых операций. Вы должны увидеть опцию **Группировка** (Group By) для каждого поля.
6. Замените **Группировку** (Group By) в первой строке итогов для Incurred на Условие (Where); замените **Группировку** во второй строке итогов для Incurred на Count; замените **Группировку** в третьей строке итогов для Incurred на Sum.
7. В колонке Incurred с оператором Условие (Where) введите Between [low] and [high] (дословно «между [low] и [high]»), как показано на рис. 16.1.
8. Нажмите кнопку **Запуск** для того, чтобы запустить запрос. Вы должны увидеть экран, подобный тому, что изображен на рис. 16.2. Заметьте, что числа Count и Sum находятся в «корзинках» (buckets) в соответствии с числами уровней. Например, с первого взгляда вы можете сказать, что убытки в размере 236 долл. находятся на уровне между 501 и 1000. Вы можете отформатировать выходные данные просто щелкнув правой кнопкой мыши на соответствующем поле в конструкторе и используя опцию форматирования.
9. Закройте и сохраните как Stratify ваш запрос/

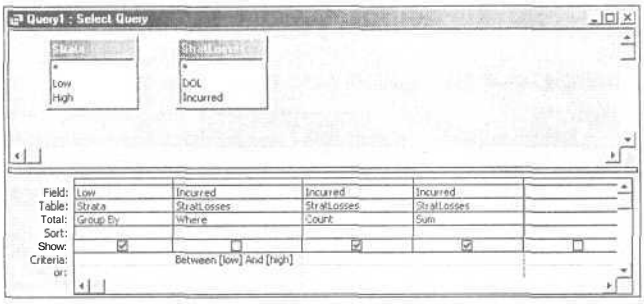


Рис. 16.1. При разработке запроса, основанного на таблице примеров, используются связи (relationships) без линий соединения

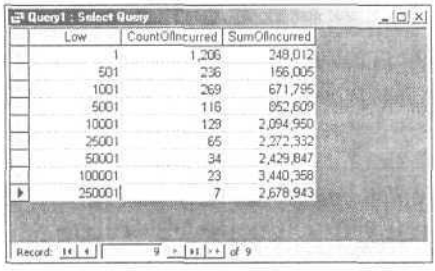


Рис. 16.2. Каждый уровень точно сгруппирован в запросе, основанном на таблице примеров

Этот запрос работает, потому что операторы Group By и Where работают совместно. Можно лучше это понять, проверив SQL-код, стоящий за этой конструкцией. Когда запрос проверяет запись, в которой представлен иск в 100 долл., он поместит его в первую группу, потому что оператор Where указывает на диапазон в том же уровне. Первая группа представляет диапазон от 1 до 500. Таким образом, выражение

```
WHERE StratLosses.Incurred Between [low] And [high]
GROUP BY Strata.Low
```

становится выражением

```
WHERE 100 Between 1 And 500
GROUP BY 1
```

для этой записи.

Итак, где же ограничения запросов? Предположим, вы выбираете первый метод и хотите установить запрос на обновление, который заполняет пустое поле Level (Уровень) правильным обозначением уровня. Проверим табл. 16.3, которая имитирует сетку запроса на обновление.

Таблица 16.3. Должен существовать лучший способ, чем этот запрос

Запрос с обновлением, поясняющий ограничения		
Поле:	Incurred (Ущерб)	Level (Уровень)
Таблица:	StratLosses (Уровни убытков)	StratLosses (Уровни убытков)
Обновленное поле: (Update To:)		1-500
Условие отбора:	Between 1 and 500 (Между 1 и 500)	
или:		

Этот запрос прекрасно работает для одного уровня, но что если вы хотите больше? вы не можете использовать оператор Or (Или), потому что не существует способа сопоставить «если» и «тогда» в добавленном условии. Представьте, что в строке «или:» записано «Between 501 and 1000» (между 501 и 1000). Куда вы поместите «тогда» для этого условия? вы не можете поместить его в строку «Обновленное поле:» для поля Level. Вы уже видели, почему не рекомендуется многократное использование функций IIF. Хорошая новость заключается в том, что ограничение может быть преодолено при помощи небольшого нововведения.

Как раз в тот момент, когда вы уже считали проблему решенной, ваш босс подкидывает вам другую «задачку». Он просит вас сгруппировать все иски по годам выдачи полисов. Из-за того что эти годы, в отличие от традиционных, начинаются 1 июня и заканчивается 31 мая, вы не можете использовать опцию группировки по годам в отчете для того, чтобы получить нужный ему отчет.

И снова на помощь может прийти запрос, основанный на таблице примеров. Все, что вам нужно делать, - это добавить два поля (один из другой таблицы) в ваш исходный запрос. Выполните следующие действия для того, чтобы добавить эти поля.

1. Из базы данных Claims откройте ваш запрос Stratify в конструкторе.
2. Нажмите кнопку **Отобразить таблицу** (желтый крестик) и два раза щелкните мышью на таблице Period. Нажмите **Закрыть,** чтобы закрыть диалоговое окно **Добавление таблицы.**
3. Щелкните два раза на поле Date1 из таблицы Period и на поле DOL (Date of Loss, Дата несения убытков) из таблицы StratLosses. (Сдвиньте оба поля в самое левое положение для правильной сортировки.)
4. В поле DOL поставьте оператор Where (Условие) и введите Between [Date1] and [Date2] в строке «Условие отбора» этого поля, как показано на рис. 16.3.
5. Запустите запрос. Ваш запрос должен выглядеть как показано на рис. 16.4.
6. Закройте и сохраните запрос.

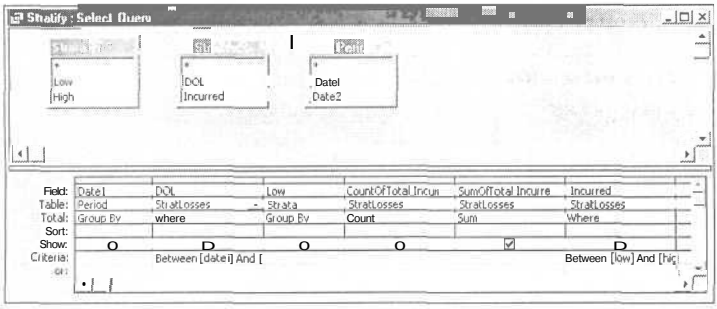


Рис. 16.3. Вы можете добавить группировку по дате в ваш запрос, основанный на таблице примеров, для того, чтобы сгруппировать по нестандартным временным интервалам, таким, как сроки выдачи полисов

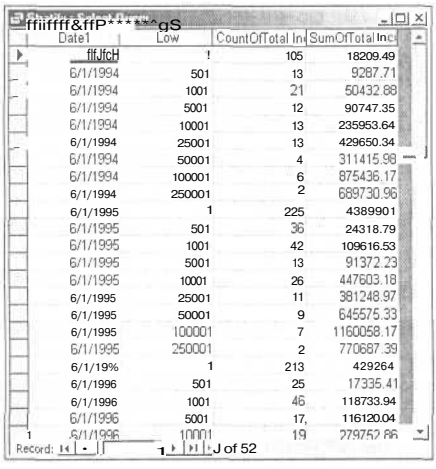


Рис. 16.4. Обратите внимание на группировку для каждого года выдачи полисов в запросе, основанном на таблице примеров

Использование подчиненных запросов для раскрытия возможностей запросов

Почему бы не расширить ваши возможности используя запросы? Существуют определенные сценарии, которые хорошо приспособлены к IN-использованию подчиненного запроса. Если вы хотите проверить значения на соответствие другой таблице, подчиненный запрос - ваше решение. Считайте его подстановочной таблицей (lookup table). Рассмотрим следующий пример SQL, который использует две таблицы для того, чтобы учесть все продукты, которые были проданы в количестве более 70 шт.

```
SELECT Products.ProductName
FROM Products
WHERE Products.ProductID
In (SELECT ProductID FROM [Order Details] WHERE Quantity >=70);
```

Если вы выберете вид конструктора (Design view) этого запроса, обратите внимание, что оператор IN становится условием отбора для поля ProductID в таблице продуктов (Products). По тому же принципу вы, возможно, хотите увидеть записи из двух связанных таблиц, которые не подпадают по некое условие. Например, сколько записей в таблице клиентов не имеют соответствующих записей в таблице счетов. Следующий подчиненный запрос использует оператор NOT для выполнения этого.

```
SELECT *
FROM Customers
WHERE CustomerID
NOT IN (SELECT CustomerlD
FROM Invoices);
```

Вам не нужно использовать две таблицы для того, чтобы создать практически ценный подчиненный запрос. Часто требуется разбить значения на процентные составляющие от целого. Следующий подзапрос, который использует базу данных Northwind (Борей), эффективно справляется с этой задачей.

```
SELECT [Order Details].UnitPrice,
(SELECT sum(UnitPrice) FROM [Order Details]) AS SumPrice,
[UnitPrice]/[SumPrice] AS [Percent]
FROM [Order Details];
```

Взгляните на этот подчиненный запрос внимательно. Обратите внимание, что он работает только *с* одним полем в базовой таблице - UnitPrice. Хотя создает два поля (SumPrice и Percent) и выводит три поля (UnitPrice, SumPrice и Percent - Цена единицы товара, Суммарная цена и Процентное отношение). Вложенный оператор Select употребляется как вычисляемое поле, которое затем используется в делении для вычисления процентного отношения. рис. 16.5 показывает вид конструктора этого подчиненного запроса, а рис. 16.6 показывает его в табличном виде (Datasheet view).

Вместо того чтобы использовать подчиненный запрос как будто он был бы полем, как показано в последнем примере, вы можете использовать подзапрос как элемент в операторе Where (немного в другом виде, чем эти операторы в первых двух примерах). Обратите внимание на следующий пример, который также использует только одну таблицу.

```
SELECT [Order Details].UnitPrice
FROM [Order Details]
WHERE UnitPrice>(SELECT Avg(UnitPrice) from [Order Details])
```

Этот запрос находит записи в таблице Order Details, которые имеют цену выше, чем средняя цена. Трудно поверить, что вы можете сделать так много с одной таблицей и одним полем.

Кроме того, существует еще одно препятствие, которое нужно преодолеть. Если вы хотите удалить записи в одной таблице, основываясь на условии отбора из другой связанной таблицы, вы столкнетесь с затруднениями, если попытаетесь объединить две таблицы вместе, выбрать критерии из обеих таблиц и произвести удаление. Использование подчиненных запросов снова может нам по-

мочь. Следующий запрос будет работать не выдавая сообщение об ошибке (так как ошибки не будет. - *Пер.*):

```
DELETE Connect.*, Connect.Company, Connect.ID
From Connect
WHERE Company = "Big Heart HMO" AND Connect.ID IN
(SELECT ID from Insureds where Connect.ID=Insureds.ID and Type = "Resident")
```

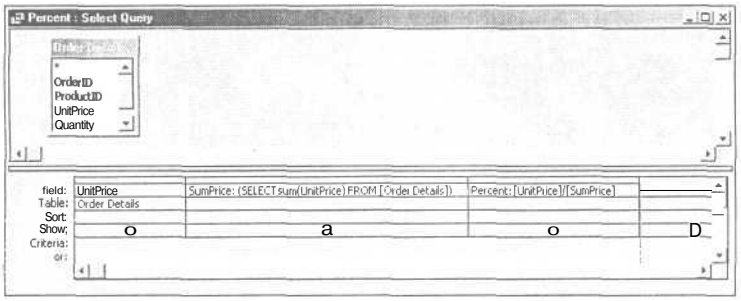


Рис. 16.5. Вы можете создать подчиненный запрос, который становится полем ваших возможных вычислений

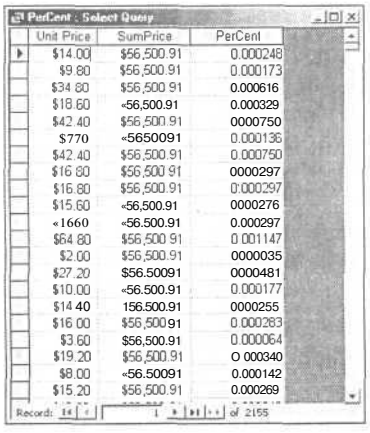


Рис. 16.6. Поле SumPrice - это сумма всех UnitPrice, которые используются для вычисления процентного отношения

Использование функций с запросами, добавления

Бывает, что вы сталкиваетесь с препятствиями, к которым вы не готовы. Представьте, что клиент послал вам список имен, которые были введены двумя секретарями. Первому секретарю нравилось все печатать прописными буквами. Второй же секретарь комбинировал строчные и прописные регистры. После сортировки и введения вашего личного первичного ключа записи были перетасованы, как колода карт. Ваш босс хочет, чтобы вы нашли все записи, написанные строчными буквами, и изменили их так, чтобы они начинались с заглавной буквы, а все остальные буквы были строчными. Следующая функция - логическая

(Boolean). Как вы уже выяснили, логический тип данных имеет два значения True либо False.

```
Function ChkUpper(strField As String) As Boolean
Dim i As Integer, chk As String

For i = 1 To Len(strField)
chk = Mid(strField, i, 1)
  If Asc(chk) >= 32 And Asc(chk) < 97 Then
    ChkUpper = True
  Else
    ChkUpper = False
    Exit For
  End If
Next i

End Function
```

Если эта функция находит ровно один незаглавный символ, она завершается и возвращает False. Обратите внимание, что выражение If использует встроенную функцию ASC для проверки регистра. Каждый символ ASCII представляется числом. Находясь в Access, нажмите комбинацию клавиш Ctrl+G для того, чтобы открыть окно проверки (immediate window). В окне напишите следующее:

```
? ASC("A")
```

Вы должны получить на выходе 65. Теперь введите следующее:

```
? ASC("a")
```

Вы должны получить 97. Здесь должен возникнуть очевидный вопрос: «Почему бы не установить диапазон значений от 65 до 97?» Ответ очень прост. Вам необходимо включить в рассмотрение пробелы, а номер пробела - 32. Следующий вопрос такой: «Но как вы собираетесь преобразовать записи прописными буквами в комбинацию строчных и прописных?» Помните функцию CapFirstLetter из гл. 10? В этой ситуации она работает идеально.

Наше решение - создать обновленный запрос, который использует обе функции. В базе данных исков Claims есть обе функции. Вы можете разрешить эту дилемму на примере, выполнив следующее:

1. Откройте базу данных Claims. В таблицах щелкните правой кнопкой мыши на Clients (Клиенты) и выберите **Копировать.**
2. Нажмите кнопку **Вставить.** Назовите ее Insureds (Застрахованные лица).
3. Щелкните два раза на таблице Insureds и обратите внимание на смесь записей заглавными буквами и незаглавными.
4. В объектах нажмите **Запросы.**
5. Щелкните два раза на **Создание запроса в виде конструктора** - откроется диалоговое окно **Добавление таблицы.**
6. Щелкните два раза на таблице Insureds и выберите **Закрыть** в окне **Добавление таблицы.**
7. Следуя рисунку 16.7, установите запрос на обновление, используя обе функции.

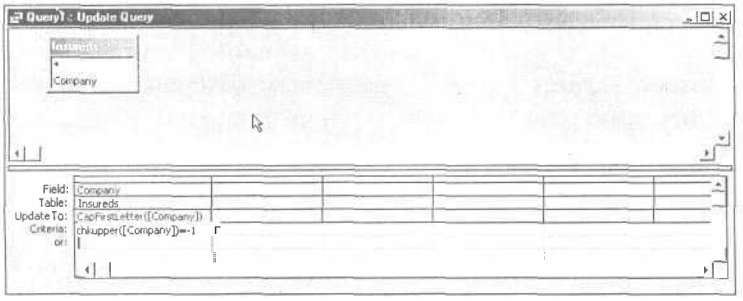


Рис. 16.7. Строки **Условие отбора** (Criteria) и **Обновленное поле** (Update To) используют функции для изменения только тех значений, которые подходят под критерий

8. Щелкните два раза Company для внесения ее в сетку.
9. В строке **Условие отбора** поля Company введите `chkupper([Company])=-1`. Когда вы проверяете логическое условие в запросе, вы должны использовать -1 для true (истина) и 0 для false (ложь).
10. Нажмите на кнопку **Тип запроса** и установите тип запроса **Обновление.**
11. В разделе **Обновленное** поле поля Company введите `CapFirstLetter([Company])`.
12. Запустите запрос и нажмите Да в предупреждающем сообщении, которое говорит вам, что вы не можете отменить действие. Вы всего лишь обновляете записи, которые набраны заглавными буквами.
13. Закройте запрос и откройте таблицу Insureds для проверки результатов, показанных на рис. 16.8.

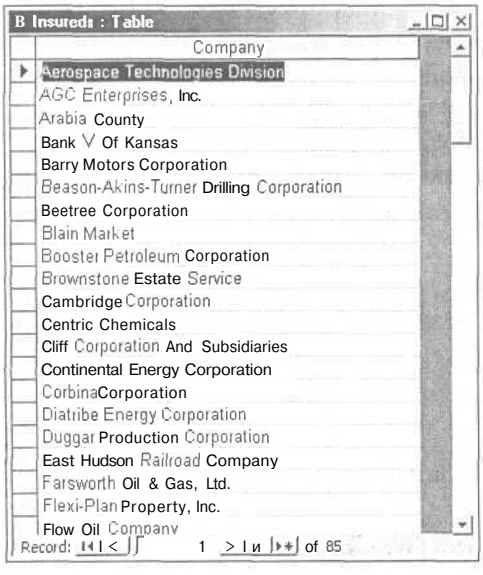


Рис. 16.8. Результаты запроса на обновление

Хотя результаты и не идеальны, этот метод намного проще, чем вручную проходить каждую запись. Очень сложно определить акронимы в программе. Как программа может сказать, является ли AON названием компании, которое получается из греческого варианта слова «возраст», или акронимом? Тем не менее эта функция может выполнять многое.

Что дальше?

В следующей главе вы продолжите рассматривать трудности, возникающие при программировании, такие, как низкая производительность, получая при этом еще лучшее понимание все большего количества тем, связанных с запросами; таким образом, вы уже интуитивно будете представлять, какой инструмент использовать в данной ситуации.

17

Наиболее эффективное использование запросов

В прошлой главе вы научились преодолевать ограничения запросов. В этой главе рассматриваются способы создания более быстрых и эффективных запросов. Вы также ознакомитесь с запросами SQL. В частности, вы узнаете следующее:

- как оптимизировать ваши запросы,
- как сжатие оптимизирует вашу базу данных,
- как индексирование ускоряет запросы,
- как избежать использования действий обобщения (aggregate functions),
- как включать только нужные поля,
- почему использование типов данных меньших размеров помогает оптимизированию,
- почему использование внешних соединений недостаточно помогает оптимизированию.

Сделаем вашу базу данных более эффективной

Прекрасный способ завоевать доверие клиента - это показать ему, как сделать их приложение быстрее и эффективнее. Как уже отмечалось, только из того, что у пользователя имеется быстрая система, вовсе не следует, что он получают оптимальное быстродействие этой системы. Медленное приложение может приводить в уныние пользователей, и они станут искать другие решения. Так как запросы обычно - самые распространенные процессы в базе данных, вы должны быть уверены, что они полностью оптимизированы.

Оптимизирование запросов

Ядро базы данных Microsoft Jet имеет встроенный оптимизатор запросов, который использует статистические данные для определения того, какую стратегию запросов применять. По этой причине вы должны открыть, перезаписать и запустить ваши запросы для того, чтобы перекомпилировать их, если вы добавляете значительное число записей в вашу базу данных, - особенно если записи добавлены в таблицу, на которой основан запрос. Вы также должны перекомпилировать их в случае смены индекса.

К сожалению, вы не можете ни посмотреть схемы оптимизации ядра базы данных, ни определить, как вы хотите оптимизировать запрос. Однако вы можете использовать «Архивариус базы данных» (Database Documenter) для выяснения того, присутствуют ли индексы и насколько уникален индекс. Просто выберите **Сервис, Анализ, Быстродействие** из меню Access. Так как анализ быстродействия запроса тесно связан с ядром базы данных Jet, Performance Analyzer (Анализатор быстродействия) предлагает добавление индексов только в тех случаях, когда они на самом деле будут использованы ядром базы данных Jet, чтобы оптимизировать запрос.

Сделайте вашу базу данных Access компактной

Частое уплотнение позволяет получить большой выигрыш в быстродействии. Ядро базы данных Microsoft Jet, которое использует основанный на затратах (cost-based) метод оптимизации, определяет самый дешевый (в смысле быстродействия. - *Пер.)* список задач, выполнение которых позволяет получить необходимые результаты. По мере роста вашей базы данных схема оптимизации может перестать быть эффективной, потому что статистические данные изменяются и требуют обновления. Уплотнение базы данных обновляет статистику базы данных и переоптимизирует все запросы.

Также по мере роста вашей базы данных она становится фрагментированной. При сжатии происходит запись всех данных из одной таблицы на соседние места на жестком диске, улучшая быстродействие при последовательном сканировании. Вы также можете дефрагментировать ваш жесткий диск, если это необходимо.

Индексируйте поля в Access

Индексированные поля на обеих сторонах запроса соединяют (join) или создают связь (relationship) между полями. Jet создает индексы на полях, которые вы связываете в окне «Схема данных» (relationships window), если их там еще нет. Это может ускорить выполнение запроса, позволяя оптимизатору запроса использовать более сложную внутреннюю стратегию связывания. Пытайтесь как можно больше использовать первичные ключи (primary keys) в противоположность уникальным индексам. Кроме того, если вы индексируете поля, которые необходимо часто сортировать, вы можете увеличить быстродействие запроса.

Избегайте использования статистических функций по подмножествам (Domain Aggregate Functions) в запросах

Не все выражения в условии отбора (criteria expressions) могут быть оптимизированы. Избегайте использования статистических функций по подмножествам (domain aggregate functions), - таких, как `DLookup` `Dsum`, и `Dcount`, для доступа к данным из таблицы, которая не находится в запросе. Статистические функций по подмножествам (domain aggregate functions) необычны для Microsoft Access; это значит, что ядро базы данных Microsoft Jet не сможет оптимизировать запросы, которые их используют. В качестве альтернативы создайте подзапрос или добавьте таблицу, к которой обращается запрос, в этот запрос.

Используйте только необходимые поля

При создании запроса выбирайте только те поля, которые вам необходимы. Если поле не должно присутствовать в запросе, не добавляйте его. Это гарантирует оптимальную скорость для ваших запросов, особенно когда запрос основан на большой таблице. Также для полей, которые вы используете только для проверки условия отбора, щелкните мышью на галочке «вывод на экран» (Show), чтобы снять ее, если вы не хотите отображать эти поля.

Используйте самые маленькие типы данных, необходимые для полей

При задании поля в таблице выберите наименьший тип данных, соответствующий данным, ожидаемым в поле. Например, если вам не требуется более 255 символов, не используйте тип Memo (текстовое поле большой длины). Если вам не требуются дроби и не ожидается, что числа будут астрономические, используйте тип long integer. Кроме того, выбирайте такой же или совместимый тип данных в полях, которые планируется использовать в связях.

Старайтесь по возможности реже использовать внешние объединения

Внешнее объединение требует полного сканирования левой стороны связи. Поэтому для оптимального быстродействия используйте их, только когда это чрезвычайно необходимо.

Преобразовывайте в файловый формат Access 2002

Используя новый (опциональный) файловый формат Access, вы можете наслаждаться более быстрым доступом и обработкой данных в больший базах данных. Среди прочих достоинств заготовлено место для незнакомых свойств и объектов, которые могут появиться в последующих версиях Access, способность сохранять файл Access как MDE или ADE в Access 2002 и улучшенный формат хранения.

Преобразуйте ваше приложение в файл MDE

Убедитесь в наличии резервной копии базы данных и взвесьте все «за» и «против» перед тем, как выбрать эту опцию. Также убедитесь сначала, что вы преобразовали файл в формат Access 2002. Это хороший способ улучшить уровень безопасности вашей базы данных.

Когда вы сохраняете вашу базу данных Microsoft Access как файл MDE, все модули компилируются, удаляется весь редактируемый исходный код и получаемая база данных уплотняется. Хотя ваши программы будут выполняться, вы не можете его просматривать или модифицировать. Во всем остальном ваша база данных нормально функционирует, что означает, что вы также можете обновлять данные и запускать отчеты. Учитывая уменьшенный из-за удаления кода размер базы данных, использование памяти оптимизировано - таким образом улучшая общее быстродействие базы данных.

Если вы хотите убедиться, что никто не просмотрит ваш исходный код, эта опция поможет вам и в этом. Просто знайте, что сохранение вашей базы данных как файла MDE предотвращает следующие действия:

- просмотр, модифицирование или создание форм, отчетов или модулей в конструкторе (Design view);
- добавление, удаление или изменение ссылок на объектные библиотеки или базы данных;

- изменение кода (файл MDE не содержит исходного кода);
- импорт и экспорт форм, отчетов или модулей (таблицы, запросы, страницы доступа к данным и макросы могут быть импортированы из базы данных или экспортированы в базы данных не MDE-формата).

Запросы против фильтров

Для преодоления препятствий, возникающих изо дня в день, вы должны иметь возможность знать, когда использовать нужный инструмент. *Фильтр* - это критерий, который применяется к текущим данным для получения подмножества записей. Можете рассматривать оператор WHERE запроса как фильтр. В Access фильтр - это инструмент, который использует критерии для получения временного (пока вы не сохраните объект, который открыли) подмножества записей, которые вы можете просматривать или редактировать. Вы применяете фильтр когда таблица или форма открыта. В запросе вы применяете фильтр к результатам запроса в режиме таблицы (Datasheet View).

Есть места в Access, где разница между фильтром и запросом расплывается. Однако обычно все дело в терминологии, а не в функциональности. Например, когда вы используете команду (action) OpenForm (ОткрытьФорму) в макросе, вы можете указать в аргументе **Имя фильтра** (Filter) запрос или фильтр, который сохранен как запрос. Хотя фильтры и запросы могут извлекать подмножество данных, основываясь на критериях (criteria), существует несколько четких различий. Вы используете фильтры в следующих ситуациях:

- вам нужен метод «одного щелчка мышью» **(Фильтр по выделенному,** Filter By Selection) просмотра подмножества записей.
- вам нужен метод «одного щелчка мышью» получения подмножества записей для редактирования.
- вам нужна альтернатива «одним щелчком мыши» для **Найти далее** (Find Next) **(Фильтр по выделенному,** Filter By Selection, может использовать часть поля, а **Фильтр по форме,** Filter By Form, может использовать подстановочные знаки).
- вам нужен быстрый способ для того, чтобы возвращать подмножество записей только из одной таблицы (для фильтра существует предел - одна таблица).
- вам нужен механизм для создания *временного* подмножества записей, которые вы можете быстро включать и выключать. (Если вы не сохраните таблицу, запрос или форму с примененным фильтром, это подмножество не будет доступно при следующем запуске запроса или открытии таблицы или формы.)
- вам не нужно исключать какие-либо поля.

Запросы вы используете в следующих ситуациях:

- вам нужно извлечь подмножество записей не открывая одну или более таблиц.
- вам нужно провести вычисления по данным из полей.

- вам нужно использовать функции для данных в полях.
- вам нужно объединить две таблицы, чтобы извлечь записи.
- вам нужно сортировать несмежные поля в различных направлениях.
- вам нужно быстро, одним действием изменить табличные данные. (С фильтром вы вручную изменяете данные *после* получения подмножества.)
- вам нужно выполнить какую-либо операцию, которую производит запрос на изменение (action query): append (добавление), update (обновление), delete (удаление), make-table (создание таблицы).
- вам нужно исключить поля.

Хотя фильтр по форме более функционален, чем фильтр по выделенному, разница в функциональности запроса все же велика. И хотя запросы намного мощнее фильтров, существуют ситуации, в которых имеет смысл применять фильтры. Кто-то может поспорить, что и запрос можно запускать «одним щелчком мыши», но это достигается только после проведенной вами работы по его созданию. В этом вся суть. Вы не обязаны создавать что-либо, имея под рукой кнопку **Фильтр по выделенному** (Filter By Selection). Если вы хотите все записи Colorado, просто нажмите кнопку **Фильтр по выделенному,** находясь на поле с этим значением, и фильтр применится.

Предположим, вы хотите посмотреть на всех клиентов из вашего списка, у кого есть слово «продажи» (sales, например sales agent - агент по сбыту. - *Пер.)* в названии его должности. Вы также, возможно, хотите посмотреть, кто является владельцем и управляющим (менеджером). Познакомиться ближе вам поможет следующий пример.

1. Откройте базу данных «Борей» (Northwind) и откройте таблицу «Клиенты» (Customers).
2. Перейдите с помощью клавиши Tab в поле **Должность** после **Обращаться к** (Contact Title) и затем найдите запись в этом поле, которая имеет слово с корнем «прод» (sales в англоязычной версии) в названии должности, например «Менеджер по продажам» (Sales menager).
3. Щелкните два раза на слове «Продажа» (Sales) для его выделения.
4. Нажмите **Фильтр по выделенному** (Filter By Selection). Обратите внимание на то, чтобы были выбраны все записи, которые начинаются со слова «Продажа» (Sales) в поле **Должность** (Contact Title) (рис. 17.1).
5. Нажмите кнопку **Удалить фильтр.**
6. Для нахождения владельцев и менеджеров в поле **Должность** (Contact Title) нажмите кнопку **Фильтр по форме** (Filter by form, в некоторых версиях переведено как **Изменить фильтр).**
7. Раскройте ниспадающий список поля **Должность** (Contact Title) и выберите **Совладелец** (Owner).
8. Щелкните на вкладке **Или** (Or) (слева внизу) и наберите *Manager (*Менеджер) в поле **Должность** (Contact Title).

9. В панели инструментов нажмите на кнопку **Применить фильтр** (Apply Filter), которая выглядит как воронка. Вы должны получить или владельцев или менеджеров.
10. Нажмите на кнопку **Удалить фильтр** (Remove Filter) и затем закройте таблицу без сохранения изменений макета таблицы.

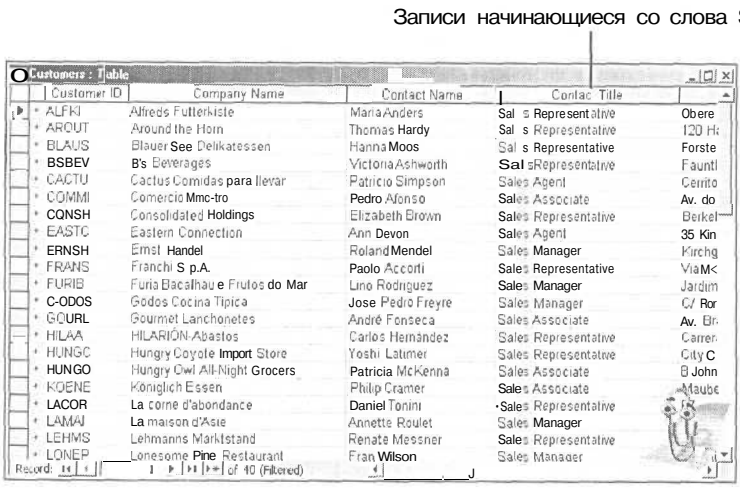


Рис. 17.1. Выбирая «Продажа» в поле **Должность** (Contact Title) с использованием **Фильтра по выделенному,** вы получаете записи, которые начинаются со слова **Продажи** (Sales)

Особые запросы, создаваемые при помощи SQL

Находясь в окне конструктора запроса, вы можете выбрать из меню Запрос (Query) строку **Запрос SQL** (SQL Specific). Вам будут представлены три варианта: **Объединение** (Union), **К серверу** (Pass-Through) и **Управление** (Data Definition).

Эти запросы не могут быть созданы в окне конструктора QBE (в обычной сетке запроса. - *Пер.*); также они не могут быть преобразованы в формат SQL Server (процесс, называемый upsize). В любом случае запрос типа «К серверу» выполняется на сервере. Вы можете создать эти запросы прямо выбирая «Создать запрос в режиме конструктора» (Create Query in Design view) и затем нажимать на **Закрыть,** когда появляется диалоговое окно «Добавление таблицы». Далее нажмите на **Запрос** в меню и выберите «Запрос SQL» (SQL Specific). Затем выберите для создания запроса тип вашего запроса.

Запросы на объединение

UNION - это не оператор и не выражение SQL, а оператор для объединения двух или большего числа таблиц или запросов с использованием двух или большего числа выражений SELECT. Этот тип запроса выдает набор записей только

для чтения (неизменяемые, non-updatable). Дублирующиеся записи удаляются по умолчанию, если только вы не используете ключевое слово ALL после оператора UNION.

Совместимость для двух выражений SELECT означает, что они должны иметь одинаковое число столбцов в одинаковом порядке. Хотя вы обычно будете использовать одинаковые названия полей и типы данных в обоих выражениях SELECT, вы можете использовать поля Number (Номер) и Text (Текст) в качестве соответствующих полей. Следующий запрос UNION (результаты см. на рис. 17.2) использует только два поля:

```
SELECT CompanyName, City, "Customers" as Relationship
FROM Suppliers
WHERE Country="Germany"
UNION SELECT CompanyName, City, "Suppliers"
FROM Customers
WHERE Country="Germany";
```

Рис. 17.2. Запрос UNION показывает результаты объединения данных их двух таблиц

Обратите внимание на то, что по ходу дела создано поле с именем Relationship. Вы также можете использовать все поля из обеих таблиц, при условии, что они имеют одинаковое число полей в одинаковом порядке. Если бы таблицы «Клиенты» (Customers) и «Поставщики» (Suppliers) в базе данных «Борей» (Northwind) имели одинаковое число полей (на самом деле это не так), вы могли бы записать запрос UNION следующим образом:

```
Select * from Customers
UNION
Select * from Suppliers;
```

Если вы опробуете этот запрос в базе данных Northwind, вы получите сообщение о том, что количество столбцов не совпадает. Но он прекрасно заработает, если вы удалите поле гиперссылок (последнее) в таблице поставщиков. (Можете попробовать, но используйте копию таблицы поставщиков.) Код клиента (Customer ID) в таблице покупателей является текстовым полем. Код поставщика (Supplier ID) в таблице поставщиков является полем autonumber (счетчик). Тем не менее они сочетаются в результирующем наборе записей. Если имена полей не совпадают, используется имя поля из первой таблицы. Таким образом, имя поля Код клиента применяется в результирующем наборе записей.

Как вам использовать запрос UNION? Вы можете употреблять его для вывода в отчет, если вам требуется увидеть две таблицы, объединенные вместе. Вы можете использовать его в качестве теста перед тем, как объединить обе таблицы. С ключевым словом ALL вы можете быстро узнать, существуют ли где-нибудь повторения. Конечно, для той же цели вы можете использовать мастер поиска повторов (Find Duplicates Wizard). Вы можете применить его для того, чтобы выяснить, сколько покупателей и поставщиков живут в одном и том же городе, как вы это делали с базой данных Northwind. Подставьте то, что вам нужно. Сколько клиентов и продавцов живут в одном городе? Ваше воображение поможет вам привести собственные примеры.

Запросы к серверу

Запрос к серверу (pass-through query) Access SQL в обход процессора запросов Microsoft Jet посылает выражения SQL прямо к источнику данных ODBS (Open Database Connectivity, «Открытый интерфейс доступа к базам данных»), такому, как сервер SQL Server. Вы должны использовать соответствующий синтаксис SQL, требуемый сервером, к которому вы получаете доступ. Вы можете производить любое действие, поддерживаемое тем источником данных, включая извлечение записей, изменение данных или даже выполнение хранимой процедуры или триггера на серверной стороне. Вы можете возвращать или не возвращать записи, в зависимости от того, какое действие вы хотите произвести. ADO всегда сохраняет запросы к серверу (pass-through query) SQL в коллекции Procedures (Процедуры) вне зависимости от того, возвращаются записи или нет.

Когда вы создаете запрос к серверу (pass-through query), вы обычно заполняете строку для подсоединения к источнику в свойстве ODBC Connect Str (Строка подключения ODBC) из списка свойств запроса. Щелкните правой кнопкой на заголовке окна запроса к серверу для того, чтобы попасть в список свойств. Если строка не заполнена, вам подскажут это, когда вы будете запускать запрос.

Создание запроса к серверу разбивается на две стадии. Первая стадия касается установки System DSN (Data Source Name). Вторая стадия касается создания пропускающего запроса, используя созданный вами DSN. Следующие шаги проведут вас через обе стадии создания запроса к серверу.

1. В меню **Пуск** (Start) Windows наведитесь на Settings (Настройки) и затем нажмите на **Панель управления** (Control Panel) для того, чтобы открыть **Панель управления.**
2. Щелкните два раза на значке 32bit ODBC (Windows 95) или на значке ODBC (Windows NT). В Windows 2000 или Windows XP откройте двойным нажатием Administrative Tools (Администрирование) и затем двойным же нажатием Data Sources (Источники данных) (ODBC). В Windows Millennium Edition щелкните два раза на ODBC Data Sources.
3. Нажмите на вкладку System DSN для того, чтобы посмотреть системные источники данных.
4. Нажмите кнопку Add (Добавить) для того, чтобы открыть первую страницу мастера создания нового источника данных (Create New Data Source Wizard) и выбрать драйвер.

5. Выберите драйвер, который вы хотите использовать. Например, выберите SQL Server. Нажмите **Готово** для того, чтобы открыть стр. 2, как показано на рис. 17.3.

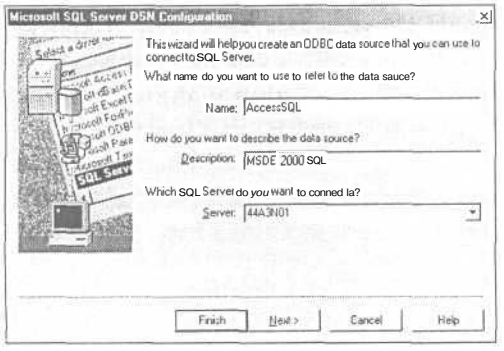


Рис. 17.3. Вы задаете имя, описание и сервер для конфигурирования DSN

6. В строке **Имя** (Name) источника данных введите имя для вашего DSN, достаточно длинное, чтобы по нему можно было понять, что это такое. Вы также можете ввести описание в строке Description для облегчения идентификации вашего DSN. В строке **Сервер** (Server) введите имя действующего сервера SQL Server.
7. Если на вашем локальном компьютере установлен MSDE 2000, вы можете ввести здесь имя своего компьютера. Нажмите Далее для того, чтобы открыть третью страницу мастера.
8. Выберите нужные настройки входа в систему. Хотя на этой странице вам и предлагается использовать личные данные, употребляемые для входа в Windows NT, сейчас выберите SQL Server authentication (что позволит вам самим вводить нужное имя пользователя и пароль при подсоединении к SQL Server. - *Пер.)* и нажмите **Далее.**
9. Выберите базу данных для подключения по умолчанию; в этом случае выберите Northwind. Нажмите **Далее** и затем нажмите **Готово.**
10. Проверьте ваше соединение нажатием на кнопку **Проверка источника данных** (Test Data Source) и, если этот тест пройден, нажмите OK.

Для создания пропускающего запроса проделайте следующее:

1. В окне базы данных «Борей» (Northwind), в разделе **Объекты** (Objects), щелкните мышью на **Запросах** (Queries) и выберите **Создать** (New) в панели инструментов окна базы данных для того, чтобы открыть диалоговое окно **Новый запрос** (New Query).
2. В диалоговом окне **Новый запрос** (New Query) выделите **Конструктор** (Design View) и нажмите на OK для того, чтобы открыть диалоговое окно **Добавление таблицы** (Show Table).
3. Не добавляя таблиц или запросов, нажмите **Закрыть** в диалоговом окне **Добавление таблицы** (Show Table).

4. Выбрав пункт меню **Запрос** (Query), наведитесь на **Запрос SQL** (SQL Specific) и нажмите на **К серверу** (Pass-Through).
5. В панели инструментов нажмите на **Свойства** для отображения списка свойств запроса.
6. В списке свойств запроса выберите свойство **Строка подключения ODBC** (ODBC Connect Str). Нажмите на кнопку Build (Построить) в конце строки для того, чтобы открыть диалоговое окно **Выбор источника данных** (Select Data Source).
7. Выберите вкладку Machine Data Source (Источник данных компьютера) и затем выберите источник данных, который вы создали на первой стадии. Нажмите **OK.** Строка подключения автоматически введется для вас, как показано на рис. 17.4.

Строка подключения

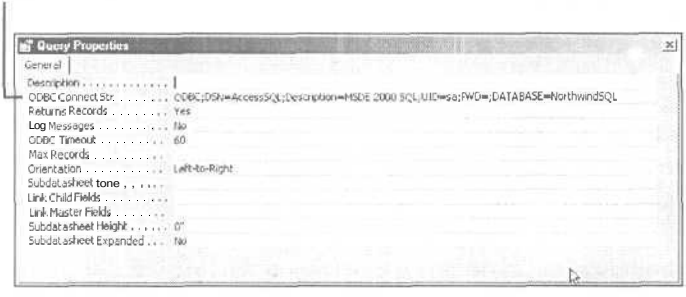


Рис. 17.4. Вы можете создать ODBC Connect Str вручную или использовать кнопку «Построить» для создания нужной строки

Вы также можете вручную установить свойство ODBC Connect Str для задания информации о базе данных, к которой вы хотите подключиться. Для получения деталей о синтаксисе вашего запроса смотрите документацию по серверу базы данных SQL, к которому вы отправляете запрос.

8. В окне SQL запроса к серверу введите ваш запрос к серверу. В нашем примере введите следующее:

```
"SELECT * FROM [Customers];"
```

ЗАМЕЧАНИЕ. Для получения информации по созданию более сложных запросов обратитесь к SQL Server Books Online (BOL), которые вы можете скачать cWeb-сайта Microsoftwww.micro5oft.com/SQLytechinfo/productdoc/2000/books.asp.

9. В меню **Запрос** выберите **Запуск** (Run). В результате должна появиться серия записей из таблицы **Клиенты** (Customers) базы данных «Борей» (Northwind).

Управляющие запросы

вы уже ознакомились с запросами, использующими DDL (Data Definition Language - язык описания данных). Хотя вы можете делать все, что они делают, вручную и даже больше, управляющие запросы предоставляют способ автома-

тизации, потому что они могут выполняться из VBA. Предположим, вы получаете таблицу от клиента каждый месяц, которая должна изменяться каждый раз, когда вы ее получаете. Будем считать, что одно из полей - текстовое поле, которое должно быть использовано как дата. Если поле содержит всего шесть символов, вмещая даты как, например, 112289, вы должны увеличить размер поля на два символа, чтобы включить разделяющие черточки. Это может быть сделано автоматически командой ALTER TABLE. Далее вы можете добавить черточки при помощи запроса и преобразовать текст в поле даты. Это может быть выполнено, даже если в таблице есть данные.

Ниже представлен пример команды ALTER TABLE.

```
ALTER TABLE Customers
ALTER COLUMN City TEXT(20)
```

Вы также можете создавать таблицу при помощи запроса на создание таблицы или в VBA с помощью метода CreateTableDef. Однако, когда вы создаете таблицу используя управляющий запрос, некоторые свойства не могут быть установлены, такие, как правила проверки корректности введенной информации (validation rules) и неуникальные индексы (nonunique indexes). Тем не менее он предоставляет четкий прямой путь создания таблицы. В отличие от использования запроса на создание таблицы вы в результате имеете табличную структуру без данных. Бывают случаи, когда это желательно. Следует только помнить, что вы должны согласовывать типы данных с типами данных в SQL, когда вы создаете запрос.

Ниже следует пример команды CREATE TABLE.

```
CREATE TABLE Customers
(CustomerID INTEGER CONSTRAINT PK_Customers, PRIMARY KEY,
LastName TEXT(50) NOT NULL,
FirstName TEXT(50) NOT NULL,
Phone TEXT(10))
```

Управляющие запросы поддерживают все нижеследующие выражения SQL:

- CREATE TABLE,
- ALTER TABLE,
- DROP TABLE,
- CREATE INDEX,
- DROP INDEX.

Что дальше?

вы узнали, как оптимизировать ваши запросы. Теперь вы должны лучше понимать разницу между запросами и фильтрами. Наконец, вы попробовали некоторые особые типы запросов, которые могут пригодиться в определенных ситуациях при разработке. В следующей главе вы узнаете, как объединять и разделять данные, изучив конкатенацию (concatenation) и разбор (parsing). Вы также узнаете, как импортировать и экспортировать используя программный код.

18
Работа с данными из внешних источников

В гл. 17 вы научились оптимизировать вашу базу данных. В этой главе элементы из других глав будут соединены вместе различными способами. Это означает, что вы будете строить на том фундаменте, который был заложен в предыдущих главах. Вещи, которые вы встретите, будут вам знакомы, поскольку они будут либо продолжением, либо взглядом с другой позиции на темы, уже изученные в предыдущих главах. Части информации, казавшиеся несвязанными, будут объединены, чтобы образовать одно гармоничное целое.

Мы обсудили разные типы интеграции и показали их на примерах. Эта глава немного глубже изучает то, как интеграция (посредством автоматического импорта и экспорта) может помочь преодолеть общие препятствия при разработке. В частности, вы изучите следующее:

- как автоматизировать импорт и экспорт;
- как обеспечить проверку ошибок в реальных условиях;
- как реализовать метод File Dialog;
- как управлять возможностями масштабирования для баз данных;
- преимущества связанных таблиц (linked tables);
- преимущества разделителей баз данных (database splitter);
- как проверять связи (links), когда таблица открывается с использованием макроса Autoexec.

Использование автоматизации для увеличения производительности

Представьте себе, что ваш босс пришел с претензией к вам, что ваша Access-база данных слишком разрослась (некоторые таблицы содержат более 30 тыс. записей). Ваша компания задумывается о покупке SQL-сервера, но это произойдет не раньше чем через год. Пользователи начинают жаловаться, что уплотнение занимает слишком много времени даже на их мощных компьютерах. Хуже того, они начинают замечать ухудшение работы при выполнении запросов.

Вы уже использовали все советы по повышению производительности, представленные в этой книге. Несмотря на то что эти приемы улучшили работу, они недостаточны в свете того факта, что база данных растет так быстро. Пользователи объяснили, что, несмотря на то что многие таблицы являются старыми, они, возможно, могут иногда понадобиться.

Мысль, которая приходит вам в голову, состоит в том, что единственное решение - это создать архивную базу данных, в которой могут храниться старые таблицы. Пользователям нравится идея, но они жалуются, что экспорт и импорт

таблиц вручную вызывает много трудностей. Кроме того, данные клиента чрезвычайно критичны ко времени.

вам становится ясно, что вы должны автоматизировать импорт таблиц в архивную базу данных и экспорт таблиц из нее. Так как в базе данных не много графических объектов, вы заключаете, что таблицы занимают в ней больше всего места. Но как вы можете сделать приложение, которое вы собираетесь создать, удобным для пользователя?

Внезапно вы вспоминаете метод запросов, описанный в гл. 12, который перечисляет все таблицы в базе данных. Это подходило для текущих баз данных, а как насчет архивных? Как можно запрашивать таблицы в другой базе данных из текущей?

Импорт и экспорт данных

Вот с такой задачей вам, возможно, придется столкнуться. Вначале вы можете приняться за последнюю проблему. Во-первых, все ваше решение можно воплотить в окне одной формы. Чтобы запрашивать другую базу данных, воспользуйтесь оператором IN, который вы можете выполнять как при помощи RunSQL, так и непосредственно в виде источника строк. Здесь приведена типичная SQL-конструкция:

```
SELECT * from Customers;
```

Чтобы запустить запрос из другой базы данных, вы просто добавляете оператор IN с указанием базы данных и ее пути в кавычках:

```
SELECT * from Customers IN "C:\AccessByExample\Northwind.mdb"
```

Следующая иллюстрация показывает другой синтаксис, который приводит к тому же результату:

```
SELECT * from [C:\AccessByExample\Northwind.mdb].Customers
```

Применительно к нашему случаю следующая SQL-строка представляет собой источник строк для ниспадающего списка для таблиц в другой базе данных:

```
SELECT MsysObjects.Name AS TableName
FROM [C:\AccessByExample\Archive].MsysObjects
WHERE (((MsysObjects.Name) Not Like "MS*") And ((MsysObjects.Type)=1) And
[ic: ccc]((Left([Name],1 ))<>"~"))
ORDER BY MsysObjects.Name;
```

Создание имен таблиц

Вспомним, что таблица MsysObjects - невидимая и содержит имена различных объектов в базе данных. Запросы к этой таблице могут обеспечить динамичное представление таблиц в базе данных. Если вы создаете таблицу имен таблиц, вы должны постоянно обновлять ее каждый раз, когда таблица удаляется, добавляется или переименовывается. Динамический список всегда содержит самую новую информацию, и при этом не нужно обращаться и обновлять статическую таблицу со списком имен таблиц. Вообще говоря, чем более динамичной будет ваша форма, тем лучше. Но это не всегда возможно.

Следующее, что вы должны сделать, - создать очередь имен таблиц, которые будут экспортироваться или импортироваться. Это осуществляется с помощью двух ниспадающих списков (list box): один для экспорта и один для импорта. Так как источник строк для этих таблиц создается динамически посредством программы, они только временные, что подходит для вашей задачи.

Автоматизирование процессов импорта/экспорта

Материал этой главы основывется на приемах нумерации, которые были изучены. Однако вместо обычной нумерации объектов вы в действительности будете экспортировать или импортировать их при помощи программного кода. Начните с команды TransferDatabase объекта DoCmd. Вы можете пользоваться методами объекта DoCmd, чтобы запускать команды (макрокоманды) Microsoft Access из VBA.

Если вы выбрали любую из команд переноса, таких, как TransferDatabase, TransferSpreadsheet, TransferText и OutputTo, вам настоятельно рекомендуется создать временный макрос, который переносил бы объекты, и затем преобразовывать его в модуль для проверки корректности синтаксиса. Не забудьте протестировать его, прежде чем вы преобразуете макрос. Синтаксис для метода TransferDatabase объекта DoCmd выглядит следующим образом:

```
DoCmd.TransferDatabase([transfer type], [database type], [database name],_
[object type],[source],[destination],[structure only],[store login]
```

Типичный пример метода TransferDatabase выглядит примерно так:

```
DoCmd.TransferDatabase acExport, "Microsoft Access",
[ic:ccc]"C:\AccessByExample\Archive.mdb", acTable, _
"Customers","Customers",False
```

Заметьте, что имя таблицы «Customer» является как источником, так и адресатом информации. Это означает, что таблице будет присвоено имя «Customer» в базе данных-получателе, которая является базой данных Microsoft Access (так как это определено в аргументе «database type») в другой папке. В качестве типа переноса («transfer type») вы можете выбирать только, acImport, acExport и acLink. Тип объекта в примере - acTable, а имя базы данных включает в себя папку с именем файла и расширением (mdb).

Создайте подкаталог с именем AccessByExample, в который вы можете скопировать архивную (Archive) базу данных (содержащую только две таблицы) из скачанных вами файлов. Есть два способа переместить таблицы при экспорте или импорте. Вы можете скопировать или переместить таблицы в место назначения. Так как перемещение приводит к удалению исходной таблицы, появляется сообщение о предупреждение, которое вы можете отменить.

Прежде чем вы перейдете к форме, познакомьтесь с новой (по крайней мере для вас) объектной переменной, называемой объектной переменной ListBox (дословно - тип переменной «Ниспадающий список». - *Пер.*). Предположим, что имя ниспадающего списка в форме - ExportList. Вы можете приписать значение переменной списку следующим образом:

```
Dim Exports as ListBox
Set Exports = Me.ExportList
With Exports
    RowSource = Table1;Table2;Table3
End With
```

Здесь предполагается, что свойство Тип источника строк (Row Source Type) было установлено как Список значений (Value List). Теперь вы готовы приступить к следующему примеру приложения, которое вы можете использовать и изменять.

1. Откройте базу данных ExportImport из папки AccessByExample. (Убедитесь в том, что архивная (Archive) таблица скопирована в папку AccessByExample.)
2. В разделе **Формы** (Forms) дважды щелкните на ExportImport, чтобы открыть форму, показанную на рис. 18.1.

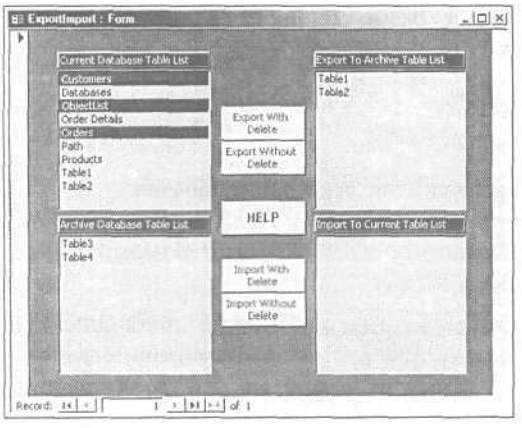


Рис. 18.1. Форма ExportImport показывает, как использовать выбираемый список (pick list) в качестве инструмент для экспорта и импорта таблиц

3. Обратите внимание на содержимое окна на рис. 18.1. Щелкните мышью на Customer и Order Details и затем дважды нажмите на Orders. Заметьте, что три таблицы перемещены в экспортный список справа, как показано на рис. 18.2.
4. Щелкните на Table1 и затем дважды на Table2.
   Заметьте, что предыдущие данные в правом текстовом окне смещены новыми. Поэтому нет необходимости в создании кнопки Clear (Очистка) для случаев, когда совершается ошибка.
5. Нажмите кнопку Export with Delete (Экспорт с удалением). Прежде чем нажать Yes, заметьте, что файл был скопирован в левый нижний список (Archive). Нажмите Yes в каждом сообщении (одно сообщение для каждой удаляемой таблицы), чтобы удалить две таблицы после того, как они были скопированы. Считайте, что кнопка Export with Delete как бы производит операцию перемещения.

ЗАМЕЧАНИЕ. Table1 - Table4 - пустые макетные таблицы, с которыми вы можете поиграть. Если вы получили сообщение об ошибке, возможно, ваша архивная база данных находится в неверном каталоге. Помните, что архивная база данных (Archive) должна находиться в подкаталоге AccessByExample на диске C.

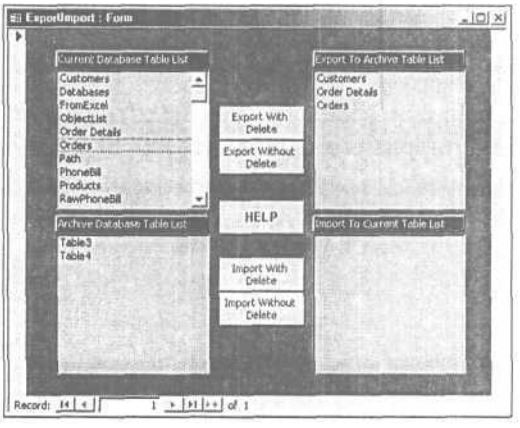


Рис. 18.2. Три таблицы перемещены в список справа после двойного щелчка

6. Теперь сделайте обратное действие, нажав Import with Delete (Импорт с удалением) и затем нажав Yes, чтобы импортировать таблицы.
7. Нажмите кнопку Help в центре окна и прочитайте сообщение, показанное на рис. 18.3, которое объясняет, как импортировать и экспортировать таблицы. Обратите внимание на то, как аккуратно текст размещен в окне. В каждой строке не более 70 символов.

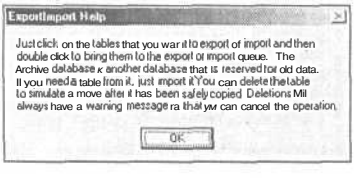


Рис. 18.3. Сообщение ExportImporf Help рассказывает пользователю, как экспортировать или импортировать таблицы

Следующая функция WrapString (дословно «сделать заворачивание текста в строке», т. е. расставить указатели перехода на новую строку, чтобы текст был так же аккуратно скомпонован, как в указанном выше сообщении. – *Пер.*)использует два аргумента. Первый - это строка, которую надо завернуть, и второй - максимальная длина каждой размещаемой строки. Когда используется комбинация с функцией MsgBox, как это сделано в кнопке Help, длинное сообщение (как, например, сообщение подсказки) заворачивается очень удачно. Она построена на приемах, использованных в функции CountWord из гл. 10.

```
Function WrapString(sFullStr As String, iWid As Integer)
Dim sAddWord As String, sNewVal As String
Dim sFrag As String, iNewStrLen As Integer
Dim sBuild As String, sRemnant As String

sRemnant = sFullStr 'только в первой итерации используется вся строка
целиком
Do Until Len(sRemnant) < iWid 'выход из цикла, когда остаток
                              'строки по длине
[ic:ccc]меньше, чем iWidth
  sBuild = Left(sRemnant, iWid) 'набираем каждую строку в соответствии
                                'с указанной iWidth
    Do Until Left(sBuild, InStr(sBuild, "")) = "" 'Составляем строку
      sAddWord = Left(sBuild, InStr(sBuild, " ")) 'получаем слово
      sBuild = Right(sBuild, Len(sBuild) - Len(sAddWord)) 'вычитаем слово
      sFrag = sFrag & sAddWord 'добавляем слово к новой строке
    Loop
  iNewStrLen = iNewStrLen + Len(sFrag) 'считаем длину новой строки
  sRemnant = Right(sFullStr, Len(sFullStr) - iNewStrLen) 'обновляем
                                                         'остаток строки
  sFrag = RTrim(sFrag) 'избавляемся от пробелов на конце строки
  sNewVal = sNewVal & sFrag & vbCrLf 'Построение заворачивающегося "
                                     'текста строка за строкой
  sFrag = "" 'обновляем переменную sFragment
             'для использования в следующем цикле
Loop
WrapString = sNewVal & sRemnant 'Добавляем последний кусок
                                'без возврата каретки
End Function
```

Следующий отрывок кода иллюстрирует программу, создающую кнопку Help.

```
Dim WrapText As String
WrapText = "Just click on the tables that you want to export "
& "or import and then double click to bring them to the export "
& "or import queue. The Archive database is another database "
& "that is reserved for old data. If you need a table from it, "
& "just import it. You can delete the table to simulate a move "
& "after it has been safely copied. Deletions will always have a "
& "warning message so that you can cancel the operation."
MsgBox WrapString(WrapText, 70),, "ExportImport Help"
```

Если бы это была реальная ситуация, вы бы немедленно захотели сжать базу данных после удаления файлов, которые были экспортированы, чтобы заполнить место. К примеру, если вы экспортируете с перемещением две таблицы по 30 тыс. записей каждая, вы увидите большую разницу в размере файла базы данных Access (mdb) после компоновки, когда посмотрите его в Проводнике Windows.

Взгляните на код, который стоит за всем этим и делает процесс перемещения возможным. Если вы щелкнете правой кнопкой на списковом окне, помеченном как Current Database Table List, выберете **Свойства** (Properties), а затем выберете событие Нажатие кнопки (On Dbl Click), то увидите следующий VBA код:

```
Dim frm As Form, CurrentList As ListBox, ExportList As ListBox
Dim strChoice As Variant, fillCtl As String

Set CurrentList = Me.Current
Set ExportList - Me.Export
  For Each strChoice In CurrentList.ItemsSelected 'пробегаем по списку
    fillCtl = fillCtl & CurrentList.ItemData(strChoice) & ";"
    CurrentList.Selected(strChoice) = False 'снимаем выделение
  Next strChoice
ExportList.RowSource = fillCtl
```

Сначала программа пробегает по выбранным элементам, используя свойство ItemSelected, которое является коллекцией. Заметьте, что strChoice должна иметь тип variant (дословно: разнообразный – *Пер.*).Переменная fillCtl является просто строкой, которая строится наращиванием. Точка с запятой используется как ограничитель (разделитель) между выбранными элементами для заполнения свойства Источник строк (Row Source) списка «Export» в верхнем правом углу.

Если вам нужен обновитель коллекций (refresher on collections), обратитесь к разделу «Объекты базы данных» в гл. 12.

После того, как таблицы были выбраны и перемещены в правый список, они будут либо импортированы, либо экспортированы и удалены, если выбрана опция удаления. Если вы выбираете экспорт с удалением для экспорта или импорта таблиц, которые выбраны, то для удаления каждой таблицы используется управляющий запрос (data definition query) DROP TABLE. Так как в этом запросе нет предупреждения об ошибке, вы должны обеспечить его программно. Вы можете использовать окно сообщения (MessageBox), чтобы предоставить пользователю выбор для подтверждения, следующим образом:

```
If MsgBox("The table has been sucessfully copied." & vbCr _
& "Do you want to delete " & strList & " in the Archive " _
& "database?", vbYesNo) = vbYes Then
cn.Execute "drop table " & strList
End If
```

Тем не менее, так как программа импорта имеет доступ к другим базам данных, вы должны иметь механизм, позволяющий выполнить запрос из базы данных ExportImport. Обнадеживает то, что метод RunSQL поддерживает запросы на языке описания данных (Data Definition Language, DDL). Но плохо то, что из-за отсутствия оператора IN для запроса DROP TABLE способ ссылаться на другую базу данных, используя этот подход, не подходит.

Так каково же решение? ADO - ваш ответ. Просто установите соединение с архивной базой данных и используйте метод Execute объекта connection (дословно «соединение». - *Пер.)* для запуска управляющего запроса (data definition query). Вам не нужно использовать набор записей, так как вы не открываете таблицы или запроса. Обратите внимание на следующий код, который привязан к командной кнопке Import with Delete:

```
Dim strList As String, i As Integer
Dim LC As Integer, ImportList As ListBox
Dim cn As ADODB.Connection
'Содержание списка «Export» - временное
Set cn = New ADODB.Connection

cn.ConnectionString = "Provider=Microsoft.Jet.OLEDB.4.0;" _
& "Data Source=C:\AccessByExample\Archive.mdb;"
cn.Open
Set ImportList = Me.Import
LC = ImportList.ListCount - 1
  For i = 0 To LC 'переменная i отвечает за данные в рядах
    strList = ImportList.Column(0, i)
    DoCmd.TransferDatabase acImport, "Microsoft Access", _
    "c:\AccessByExample\Archive.mdb", acTable, strList, strList, False
  If MsgBox("The table has been sucessfully copied." & vbCr _
  & "Do you want to delete " & strList & " in the Archive " _
  & "database?", vbYesNo) = vbYes Then
  cn.Execute "drop table " & strList
  End If
  Next i Me.Refresh
```

Свойство ListCount содержит количество элементов в списке значений для элемента управления «Список» (ListBox). Так как он отсчитывается от нуля, вы должны перебирать значения от 0 до ListCount - 1. Чтобы получить строку (элемент) из свойства column (столбец), вы должны вначале указать столбец, как это сделано в ImportList. Column(0, i). Представляйте себе это как электронную таблицу, где 0 = A (первый столбец) и переменная i = последовательные строки (1, 2, 3 и т. д.). Объект DoCmd используется для запуска макроса TransferDatabase из процедуры. Справка Access говорит: «Метод TransferDatabase вызывает макрос TransferDatabase в Visual Basic».

Обратите внимание на использование управляющего запроса (data definition query) DROP TABLE для удаления таблицы. Конкатенация (concatenation), которая подробно описана в следующей главе, делает возможным использование переменной в запросе. Так как объект cn указывает на архивную базу данных (Archive), управляющий запрос (data definition query) исполняется на соответствующей таблице в этой базе данных, которая находится вне текущей базы данных.

Для более подробной информации о конкатенации обратитесь к разделу «Искусство конкатенации» в гл. 19.

Дилемма экспорта

Ваша форма для импорта и экспорта - большой успех, но пользователи по-прежнему жалуются: они еще хотят иметь возможность экспортировать в файлы иных форматов, отличных от файлов Access. Хотя вы можете импортировать таблицы, запросы, формы, отчеты и другие объекты Access используя меню **Файл** (File), **Импорт** (Import), вы не можете экспортировать сразу несколько объектов используя **Файл, Экспорт.**

Единственное решение - сделать это с применением VBA. Вам нужна только одна форма с четырьмя списками (ListBox). В верхнем левом списке будут содержаться объекты для экспорта. В верхнем правом списке будут содержаться варианты файловых форматов, таких, как Excel или HTML. Третье текстовое поле (TextBox) будет содержать путь, по которому будут находиться экспортированные объекты. Четвертое поле будет видимым, только если вы выберете Microsoft Access в качестве файлового формата. Оно содержит имя базы данных и путь для экспорта. Когда вы щелкнете на нужном типе объекта, таком, как таблица, запрос, форма или отчет, соответствующие файловые форматы автоматически появятся (см. рис, 18.4) в списке Export Types.

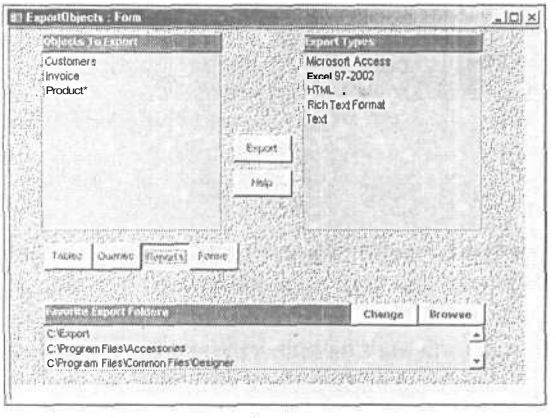


Рис. 18.4. Форма ExportObjects показывает, как вы можете использовать выбираемый список (PickList) для экспорта различных типов объектов Access

Сейчас вы, возможно, удивляетесь, как эти формы работают. После того как вы попробуете запустить форму, у вас будет возможность посмотреть программу, скрывающуюся за ней. Во-первых, создайте папку на диске С с именем C:\AccessByExample\Export. Затем выполните:

1. В разделе Формы дважды щелкните на ExportObjects, чтобы открыть форму. Нажмите кнопку Help, чтобы получить некоторые инструкции о том, как использовать форму.
2. Нажмите Tables в группе кнопок, которые показаны рядом с текстом подсказки ControlTip на рис. 18.5. Обратите внимание, что оба списка, Objects To Export (Объекты для экспорта) и Export Types (Типы экспорта), заполнены.
3. Под Objects To Export щелкните Customers, чтобы добавить его для экспорта.
4. Под Export Types щелкните на Excel 97-2002, чтобы выбрать этот файловый тип для экспорта таблицы Customers.
5. Нажмите Browse за Favorite Export Folders, чтобы открыть диалоговое окно Обзор (Browse).
6. Выберите C:\AccessByExample\Export, как показано на рис. 18.6, и нажмите ОК. (Она уже должна быть создана.)

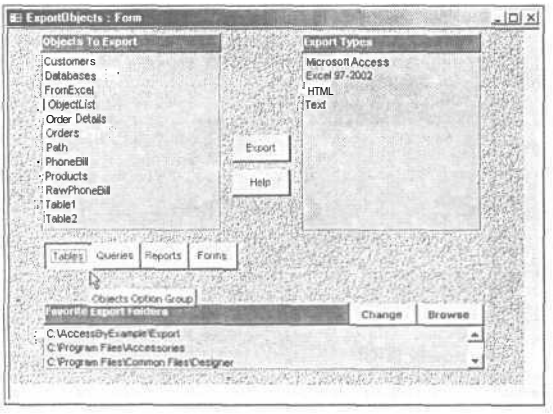


Рис. 18.5. Текст подсказки ControlTip показывает группу кнопок, которые позволяют вам выбирать типы объектов для экспорта

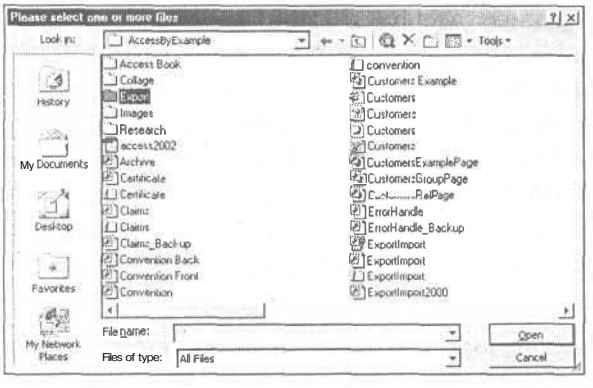


Рис. 18.6. Свойство FileDialog объекта Application применяется для воспроизведения диалогового окна **Обзор** (Browse), в котором пользователь может выбирать папки

7. После того как вы выбрали папку C:\AccessByExample\Export и убедились в том, что экран выглядит как показано на рис. 18.7, на котором показаны опции, которые необходимо выбрать, нажмите Export.
   Должны быть выбраны Customers, Excel 97-2002; C:\AccessByExample\Export.
8. Откройте Проводник Windows, откройте в нем папку C:\AccessByExample\Export и дважды щелкните на файле Customers.xls, чтобы открыть его. (Если в проводнике Windows детализация не активирована, ищите значок Excel.) Убедитесь, что файл был экспортирован, и закройте файл Excel. Сверните проводник Windows, чтобы снова видеть форму.
9. Нажмите на Table1, удерживайте клавишу Ctrl и нажмите Table2.
10. Под Export Types нажмите Microsoft Access. Заметьте, что окно Databases активировалось.

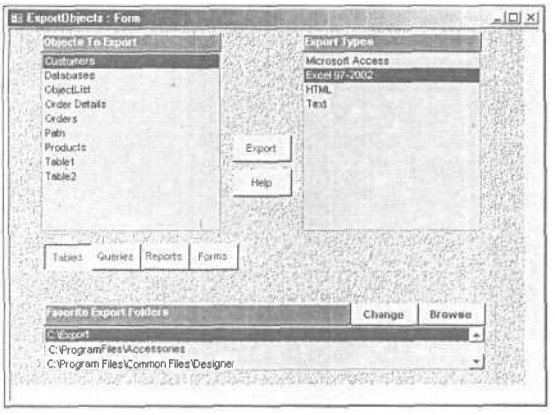


Рис. 18.7. Вы выбираете объект для экспорта, объектный тип и папку, в которую хотите поместить экспортируемый объект

11. Нажмите Browse, выберите базы данных Archive и Claims и нажмите OK. Вы можете выбрать более одного файла, когда просматриваете, так как свойство AllowMultiSelect установлено как True.
    В отличие от формы ExportImport при помощи данной формы вы можете экспортировать не только таблицы. Формы, отчеты и запросы могут также экспортироваться в другие базы данных Access.
12. Попробуйте снова в окне **Обзор** выбрать базу данных Archive. Вы получите сообщение об ошибке, которое скажет вам, что вы пытаетесь создать дубликат записи.

Теперь у вас есть средство, с помощью которого ваши пользователи могут осуществлять экспорт различных типов объектов. Можно не только применять разные типы объектов, но и экспортировать объекты, используя различные типы файлов, такие, как Excel, HTML или Rich Text Format (RTF). К примеру, если клиент хочет, чтобы вы создали для него копию отчета на диске, вы можете экспортировать его в RTF и поместить в Microsoft Word.

Добавьте проверку ошибок в ваше приложение

Чтобы попытаться сделать приложение как можно более устойчивым к ошибкам, вы пытаетесь предвидеть возможные ошибки, которые могут произойти. Что если пользователь решит переименовать или переместить каталог «Избранное»? В этом случае пользователю может быть предложено войти в другой каталог. А что если таблицы, содержащие объекты или пути доступа к файлам, удалены? Вы можете начать с проверки, чтобы посмотреть, существуют ли таблицы. Если их нет, они могут быть созданы «на лету».

Чтобы начать, вам надо изучить, как переключать типы объектов нажатием кнопки. Несмотря на то что вы уже познакомились с элементом управления «группа кнопок» (option group), вы еще не научились программировать их. Есть

четыре кнопки в группе кнопок WhichObject. Следующий код привязан к событию After Update этого элемента управления.

```
Dim ObjType As Long, TypeName As String
Select Case Me.WhichObject
  Case 1
  ObjType = 1
  TypeName = "Table"
  Case 2
  ObjType = 5
  TypeName = "Query"
  Case 3
  ObjType = -32764
  TypeName = "Report"
  Case 4
  ObjType = -32768
  TypeName = "Form"
End Select

SQLString = "SELECT MsysObjects.Name " _
& "FROM MsysObjects " _
& "WHERE (((MsysObjects.Name) " _
& "Not Like ""MS*"") And ((MsysObjects.Type)=" & ObjType & ") " _
& "And ((Left([Name],1))<>""~""));"

Me.Objects.RowSource = SQLString
Me.Types.RowSource = "SELECT FileType, Object " _
& "FROM ObjectList " _
& "WHERE ObjectList.Object =" & Chr$(34) _
& TypeName & Chr$(34) & ";"
Me.Refresh
```

ЗАМЕЧАНИЕ. Группа кнопок может быть создана с использованием мастера. Только убедитесь, что выделена «волшебная палочка», прежде чем нажать на инструмент «группа кнопок». Когда вы выбираете одну из кнопок в группе, например переключатель (toggle button), номер выбранной кнопки присваивается самой группе.

Конструкция Select - Case использует свойство значений группы свойств, чтобы определить, какая из кнопок нажата. Если нажата первая кнопка, выбираются таблицы. Если вторая - запросы и т. д. Положительные и отрицательные значения, которые устанавливаются для различных объектных типов, получены из таблицы MsysObjects. Они используются в динамическом запросе, встраиваясь в строку SQL. После того как переменной SQLString присвоено значение строки, она записывается в свойство RowSource списка Objects следующим образом:

```
Me.Objects.RowSource = SQLString
```

После того как свойство RowSource списка Objects установлено, обратите внимание, как запрашивается таблица ObjectList. Вы можете скрывать и показывать эту таблицу, нажав правую кнопку мыши на ней и выбрав Properties. Сначала выберите меню **Сервис** (Tools), **Параметры** (Options), **Вид** (View); затем поставьте галочки напротив Скрытые и Системные объекты (Hidden и System). На-

жмите правой кнопкой на таблице ObjectList и заметьте, что атрибут Скрытый (Hidden) установлен. Если вы не хотите, чтобы пользователь видел таблицу, вы можете скрыть ее для защиты. Эта таблица имеет поля FileType и Object. Эти поля действуют совместно в запросе. Например, RTF появляется только тогда, когда вы нажимате Reports.

После того как кнопка определена, определяются две переменные: одна для имени объекта (TypeName) и одна для типа объекта (ObjType). Источники строк двух списков затем устанавливаются динамически *с* помощью двух запросов; один для каждого списка. Так как используется событие After Update, значения по умолчанию для группы свойств не установлено. Например, если для какого-либо элемента значение по умолчанию было установлено, нажатие на него не вызовет никакого действия, так как это не обновление (update). Но из-за того что значение по умолчанию не установлено, нажатие первой кнопки вызовет немедленное срабатывание события.

Списки для директорий и баз данных не динамические, но и не установлены как «избранные» в Windows. Пользователи должны не забывать возвращаться и менять путь к файлу, если они переименовали или переместили его в другую папку. Если вы попытаетесь добавить элемент, который окажется дубликатом, вы получите сообщение. Следующий отрывок кода, который привязан к событию On Click кнопки Browse (Обзор) около списка для директорий, предотвращает возможные попытки добавить дубликат.

```
Dim ItemPicked As Variant
Dim cn As ADODB.Connection
Dim rs As Recordset

'Вы должны установить ссылку на библиотеку Microsoft Office 10.0 Object
Library
Set cn = CurrentProject.Connection
Set rs = New ADODB.Recordset
rs.Open "Path", cn, adOpenKeyset, adLockOptimistic
With Application.FileDialog(msoFileDialogFolderPicker)
  If.Show Then
    On Error Resume Next
    For Each ItemPicked In.SelectedItems
      With rs
        .AddNew
        rs("PathName") - ItemPicked
        rs.Update
      End With
    If Err.Number = -2147217887 Then
    MsgBox "Error " & Str(Err.Number) & " was generated." _
    & vbCr & "You have attempted to create a duplicate " _
    & "record. Try another item." & vbCr _
    & "Source: " & Err.Source
    End If
    Next ItemPicked
  End If
End With
Me.Refresh
```

Использование метода File Dialog (диалогового окна открытия файла)

В ранних версиях Microsoft Access вы не могли выводить файловое диалоговое окно без использования элемента управления ActiveX или обращений к Windows API. Microsoft Access 2002 дает вам возможность выводить файловое диалоговое окно, применяемое Microsoft Access. Употребляя метод FileDialog, вы можете определять, какие файлы были выбраны пользователем. Коллекция Selectedltems объекта FileDialog содержит пути к файлам, выбранным пользователем. Применяя цикл For - Each, вы можете перебрать эту коллекцию и вывести на дисплей каждый файл. Табл. 18.1 показывает четыре свойства DialogType, которые вы можете использовать с этим объектом, с описанием того, что делает каждый диалоговый тип.

Таблица 18.1. Свойства DialogType

Диалоговый тип	Функция типа	Использование типа
MsoFileDialogOpen	Диалоговое окно **Открыть**	Позволяет пользователю выбирать один и более файлов (в зависимости от того, установлен ли AllowMultiSelect как True), которые затем могут быть открыты в ведущей прикладной программе посредством использования метода Execute
MsoFileDialogSaveAs	Диалоговое окно «Сохранить как»	Позволяет пользователю выбирать одиночный файл, который затем может быть сохранен с названием («saved as») посредством использования метода Execute
MsoFileDialogFilePicker	Диалоговое окно выборки файлов	Позволяет пользователю выбирать один и более файлов (в зависимости от того, установлен ли AllowMultiSelect как True). Пути к файлам, которые выбрал пользователь, собраны в коллекции FileDialog SelectedItems
MsoFileDialogFolderPicker	Диалоговое окно выборки папок	Позволяет пользователю выбирать директории. (AllowMultiSelect не работает с этим типом.) Директория, которую выбирает пользователь, вводится в коллекцию FileDialog Selectedltems

Не забудьте установить ссылку на библиотеку объектов Microsoft Office 10.0 Object Library, если хотите использовать эти новые возможности. В окне модуля выберите **Сервис** (Tools), **Ссылки** (References) и прокрутите список, пока не найдете библиотеку. Метод Show объекта FileDialog возвращает True, если пользователь выбирает один и более файлов. Так как процедура выбирает папки, множественный выбор недоступен. Вследствие этого коллекция SelectedItems возвращает только один элемент.

Процедура для кнопки Browse (Обзор) около списка Databases немного сложнее, но не пугайтесь. Она показывает, как применять свойство AllowMultiSelect, чтобы позволить пользователю выбирать более одного файла одновременно. Следующий листинг употребляет ADO, чтобы добавить выбранные элементы в таблицу Databases так, чтобы базы данных могли быть использованы снова и снова, как если бы они были в папке «Избранное» (Favorites).

```
Dim ItemPicked As Variant
Dim cn As ADODB.Connection
Dim rs As Recordset
Dim dlg As Office.FileDialog

Set dlg = Application.FileDialog(msoFileDialogFilePicker)
Set cn = CurrentProject.Connection
Set rs = New ADODB.Recordset
rs.Open "Databases", cn, adOpenKeyset, adLockOptimistic
With dlg
   .AllowMultiSelect = True
   .Show
   .Title = "Please select one or more files"
  On Error Resume Next
    For Each ItemPicked In.SelectedItems
      With rs
         .AddNew
        rs("DbName") = ItemPicked
        rs.Update
      End With
    If Err.Number = -2147217887 Then
      ErrorHandle Err.Number, "You have attempted to " •_
      & "create a duplicate record.", Err.Source, _
       "Try another database."
    End If
    Next ItemPicked
End With
Me.Refresh
```

Как утверждалось раньше, использование On Error Resume Next следует свести к минимуму. Тем не менее, когда вы просто хотите задержать сообщение об ошибке вместо того, чтобы отменить его, вы можете воспользоваться оператором Resume Next. Если применять его правильно, сообщение об ошибке не сработает, пока вы не захотите, чтобы оно сработало. Иногда активируется сразу несколько сообщений об ошибках. Когда вы тестируете вашу процедуру, вам следует закомментировать конструкцию On Error Resume Next, поместив апост-

роф (') перед всей строкой. После того как вы прочитали сообщения, которые были сгенерированы, вы определяете, какие из них существенны. Таким образом, вы можете выбрать сообщение, которое будет видеть пользователь. После помещения вашей подпрограммы обработки ошибок в процедуру просто удалите апостроф перед строкой On Error Resume Next и повторно протестируйте процедуру.

Когда происходит ошибка, в свойства объекта Err заполняется информация о том, какая ошибка произошла и что с ней делать. Подпроцедура ErrorHandle просто использует информацию, которая туда вводится, чтобы послать сообщение пользователю, не возвращая значения, как показано в следующий листинге.

```
Sub ErrorHandle(ErrNum As Double, Desc As String, _
Source As String, Solution As String)

MsgBox "Error Number: " & ErrNum & vbCr _
& "Description: " & Desc & vbCr _
& "Source: " & Source & vbCr _
& "Solution: " & Solution

End Sub
```

Масштабируемость

Ваш босс приходит к вам через месяц с новой проблемой. Сетевой трафик растет, так как были наняты новые работники, чтобы обслужить всех клиентов. Они тоже должны иметь доступ к базе данных. Производительность снова уменьшается, но на этот раз по другой причине. Вы провели поиск творческих идей (метод «мозговой атаки») с другими компьютерными профессионалами и решили разбить базу данных на две: одну для таблиц и одну для форм, отчетов, макросов, модулей и других объектов.

Если бы это произошло в реальных условиях, у вас было бы несколько возможных вариантов, а именно:

- разбиение базы данных на две, как советовалось выше;
- использование проекта Access;
- превращение вашей базы данных в приложение intranet посредством использования Web-сервера, к которому служащие могут получать доступ, вводя регистрационное имя и пароль.

Если в компьютерном мире говорится, что что-то является *масштабируемым,* это означает, что это что-то может приспосабливаться к увеличивающимся размерам. Например, если размер шрифта можно увеличить, говорят, что он масштабируем. Аналогично, если сеть или система баз данных может приспосабливаться к увеличивающимся размерам файлов или большому числу пользователей, говорят, что она масштабируема. Таким образом, задача, представленная в этом сценарии, касается масштабируемости.

Связанные таблицы

Так как второй и третий варианты уже обсуждались, пойдем дальше по сценарию и предположим, что вы решили отдать предпочтение первому варианту. Первое, на что надо обратить внимание, заключается в следующем: как обеспечить вашим пользователям возможность доступа к таблицам из другой базы данных, как если бы они находятся в локальной базе данных? Приняв во внимание то, что таблицы находились на файловом сервере, вы можете привязать (link) таблицы к каждому локальному пользовательскому ПК.

Вот некоторые преимущества связанных таблиц:

- вы не увеличиваете размер вашей базы данных, когда добавляете новые таблицы, так как данные лежат в другом месте;
- вы можете работать с таблицами из других приложений, таких, как Excel, в вашей базе данных Access, не импортируя их;

А вот некоторые недостатки этого метода:

- если внешние данные перемещаются в другое место, вы должны ипользовать Linked Table Manager для обновления связей;
- вы лишаетесь возможности устанавливать целостность ссылочных данных для связей (referential integrity on relationships), если связанные таблицы не находятся в базе данных Access;
- связанные таблицы работают немного медленнее, чем «родные» таблицы Access.

Разделение баз данных

Разделенная база данных хорошо работает в сетевом (многопользовательском) пространстве. Она состоит из двух файлов базы данных. Одна из этих баз данных, называемая *back-end* (прикладная часть - часть клиент-серверного приложения, выполняющаяся на сервере; антоним: front-end. - *Пер.*), содержит только таблицы и соотношения (relationships). Эта база данных находится на сетевом файловом сервере. Другая база данных, называемыя*front-end* (интерфейсная часть), содержит все остальные объекты базы данных, такие, как запросы, формы, отчеты, макросы и модули. Эта база данных копируется на каждый локальный пользовательский компьютер.

Идея разделенной базы данных имеет следующие преимущества:

- Улучшается работа, так как только используемая часть базы данных находится на локальных пользовательских дисках. Таблицы находятся на файловом сервере.
- Уменьшается сетевой трафик, так как только данные передаются по сети. Остальные объекты в базе данных локальные.
- вам не надо прерывать работу, чтобы обновить формы, отчеты, макросы или модули.
- пользователи могут создавать свои собственные индивидуальные объекты, такие, как запросы, формы или отчеты, не оказывая влияния на других пользователей.

Мастер разделения баз данных

После того как база данных разделена на две, разделенные части работают как одна посредством связанных таблиц. Практически связанные таблицы работают так, как если бы они находились в клиентской базе данных, хотя они немного медленнее. Каждый клиентский компьютер имеет свою персональную копию клиентской базы данных, которая взаимодействует с одиночной копией серверной базы данных, расположенной на файловом сервере.

Мастер разделения баз данных автоматизирует процесс разделения, создавая серверную базу данных и перемещая туда все таблицы. Связи таблиц, позволяющие сообщаться двум базам данных, создаются автоматически. Не волнуйтесь, если у вас только один компьютер: вы все равно сможете почувствовать, как происходит этот процесс.

Выполните следующие шаги, чтобы разделить вашу базу данных, используя мастер.

1. Выберите **Файл** (File), **Открыть** (Open) из строки меню и выберите папку AccessByExample. Сделайте копию базы данных Convention и назовите ее Convention Front.

   Вы также можете щелкнуть правой кнопкой на базе данных Convention, чтобы открыть меню быстрого вызова и выбрать **Копировать** (Copy), как показано на рис. 18.8. Щелкните правой кнопкой на большой белой области справа от Convention и выберите **Вставить** (Paste). Будет создан файл «Копия Convention» (Copy of Convention). Переименуйте копию в Convention Front.

Рис. 18.8. Вы можете копировать, вставлять, удалять и переименовывать базы данных из диалогового окна открытия файла

ЗАМЕЧАНИЕ. Вы можете сделать папку AccessByExample папкой по умолчанию, выбрав **Сервис** (Tools), **Параметры** (Options), **Общие** (General) и написав в поле **Рабочий каталог** (Default Database folder) строку C:\AccessByExample. Здесь вы также можете увеличить число «запоминаемых» недавно использованных файлов.

2. Дважды щелкните на базе данных Convention Front, чтобы открыть ее.
3. Выберите **Сервис** (Tools), **Служебные программы** (Database Utilities), **Разделение базы данных** (Database Splitter) из строки меню, чтобы открыть мастер **Разделение базы данных** (Database Splitter), показанный на рис. 18.9.

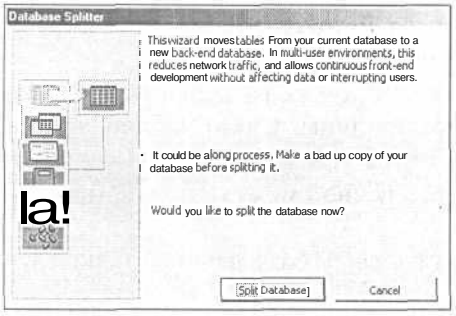


Рис. 18.9. Мастер Разделение базы данных (Database Splitter) создает серверную базу данных, содержащую только таблицы, которые будут связаны с клиентской базой данных

4. Нажмите кнопку **Разделить** (Split Database), чтобы открыть диалоговое окно **Создание базы данных с таблицами** (Create Back-end Database), как показано на рис. 18.10.

Рис. 18.10. Диалоговое окно Создание базы данных с таблицами (Create Back-end Database) позволяет вам называть серверную базу данных и выбирать папку для нее

5. Нажмите кнопку **Разделение** (Split), чтобы создать базу данных Convention Front_BE (сокращение BE означает back-end - серверная база данных) Появляется следующее сообщение: Database successfully split appears (База данных успешно разделена).
6. Обратите внимание на необычный вид таблиц, показанных на рис. 18.11, после разделения базы данных. Большая черная стрелка перед именем таблицы обозначает связь.

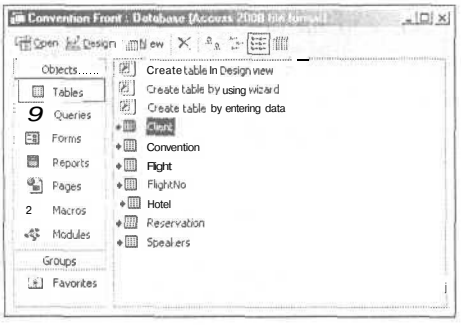


Рис. 18.11. Связанные таблицы по-разному изображаются в интерфейсной части базы данных, указывая на то, что они связаны

7. Откройте таблицу Client. Даже если она находится в другой базе данных, она откроется, как если бы она находилась в базе данных Convention Front. Закройте таблицу, но останьтесь в базе данных.

Диспетчерсвязанныхтаблиц

Лучше не перемещать и не переименовывать таблицы или базы данных, имеющие отношение к связанным таблицам, без острой необходимости. Если же вы все-таки переместили или переименовали таблицу или базу данных, вы можете использовать Linked Table Manager (Диспетчер связанных таблиц), чтобы восстановить связи.

Следующий пример знакомит вас с работой диспетчера связанных таблиц (Linked Table Manager).

1. В базе данных, открытой в последнем упражнении, выберите **Файл** (File), **Открыть** (Open), чтобы высветилось диалоговое окно открытия файла.
2. Щелкните правой кнопкой на Convention Front_BE database и затем выберите **Переименовать** (Rename). Измените имя на Convention Back. Закройте диалоговое окно открытия файла.
3. Откройте таблицу Client, как делали до этого. Появится сообщение, показанное на рис. 18.12. Нажмите OK.

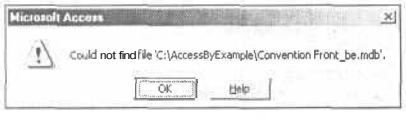


Рис. 18.12. Если база данных перемещена или переименована, появится сообщение о возникшей проблеме

4. Нажмите **Сервис** (Tools), **Служебные программы** (Database Utilities), **Диспетчер связанных таблиц** (Linked Table Manager), чтобы открыть диспетчер связанных таблиц.

5. Нажмите **Выделить все** (Select All), как показано на рис. 18.13, и нажмите **ОК**, чтобы открыть диалоговое окно **Выбор нового расположения: Client** (Select New Location of Client).

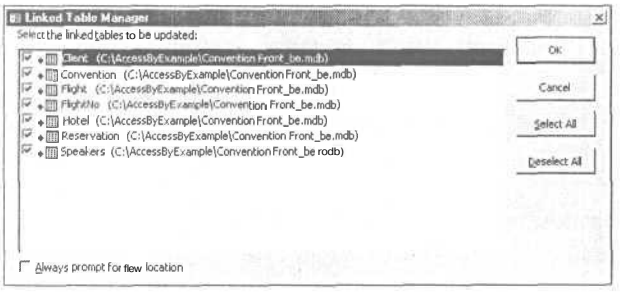


Рис. 18.13. Вы можете выбрать все связанные таблицы в диспетчере связанных таблиц и обновить связи

6. Когда появится диалоговое окно **Выбор нового расположения: Client** (Select New Location of Client), выделите Convention Back и затем **Открыть** (Open). Сообщение говорит вам, что все связанные таблицы успешно обновлены. Нажмите **Закрыть** (Close).
7. Снова откройте таблицу Client и заметьте, что связи обновлены. Закройте таблицу.

Проверка связей

вы можете написать программу для автоматической проверки связей, когда открывается интерфейсный файл. VBA-процедура проверки связей может быть прикреплена к макросу Autoexec так, что связи будут проверяться каждый раз при открытии базы данных. Следующий код, содержащийся в вашей базе данных Scale Front, проверяет связи и выдает пользователю сообщение в зависимости от результата проверки.

```
Function ChkLink()
Dim cn As ADODB.Connection
Dim cat As New ADOX.Catalog 'Должна быть установлена ссылка на ADO Ext.
2.5 for DDL
Dim tbl As Table, varTable As Variant, varItem As Variant
Dim sFolderFile As String, sMsg As String
Dim dlg As Office.FileDialog 'Должна быть установлена ссылка на Office
10.0

On Error GoTo ReLink_err
Set cn = CurrentProject.Connection
Set cat.ActiveConnection = cn
  For Each tbl In cat.Tables
    If tbl.Type = "LINK" Then 'Исключаем любой тип, кроме ссылок (link)
      varTable = tbl.Columns(0) 'Просто проверка ссылки
    End If
```

```
  Next
cn.Close
Set cn = Nothing
Set cat = Nothing
Exit Function

ReLink_err:
sMsg = "There is a problem with the linkage of the" _
& " table " & tbl.Name & ". Probable cause: The back end" _
& " database may have been moved or renamed. Please enter" _
& " the name of the source path and database in the next" _
& " screen and an attempt will be made to relink. Otherwise" _
& " choose Cancel and click Tools, Database Utilities, Link" _
& " Table Manager."
If MsgBox(WrapString(sMsg, 52), vbOKCancel) = 1 Then
  With Application.FileDialog(msoFileDialogFilePicker)
    If.Show Then
      For Each varItem In.SelectedItems 'выбираем базу данных
        sFolderFile = varItem
      Next varItem
    End If
  End With
Else
  Exit Function
End If
Resume TestAgain:

TestAgain:
sMsg = tbl.Name & " does not exist in back end database." _
    & " Choose OK to open the Linked Table Manager or" _
    & " Cancel to cancel the procedure and find the table."
On Error Resume Next
cn.Execute "SELECT * FROM " & sFolderFile & "." & tbl.Name
If Err.Number = -2147217865 Then
  Select Case MsgBox(sMsg, vbOKCancel)
    Case 1 ' OK
      DoCmd.RunCommand acCmdLinkedTableManager
    Case 2 ' Cancel (Отмена)
      Exit Function
    Case Else
      Exit Function
  End Select
Else
  DoCmd.DeleteObject acTable, tbl.Name
  DoCmd.TransferDatabase acLink, "Microsoft Access", _
  sFolderFile, acTable, tbl.Name, tbl.Name, False
End If

End Function
```

Вы, должно быть, уже хорошо знакомы к этому моменту с большинством программных решений, примененных в функции ChkLink. Выражение varTable = tbl.Columns(0) проверяет связи, пытаясь обратиться к первому полю в каж-

дой таблице. Если таблица или база данных была перемещена или переименована, связь нарушается, что вызывает ошибку. Фактически процедура проверяет связи дважды. Первый раз она проверяет, чтобы убедиться, что файл базы данных не перемещен и не переименован (что приводит к неверности ссылок). Второй раз она проверяет достоверность ссылок используя запрос. Наконец, у вас есть возможность запускать «Диспетчер связанных таблиц» (Linked Table Manager) командой `DoCmd.RunCommand acCmdLinkedTableManager`. Таким способом вы можете исполнять любую команды из строки меню.

Добавление кода в макрос Autoexec

вы можете использовать действие `RunCode` из макроса, чтобы запускать VBA-процедуру. Если вы хотите, чтобы процедура запускалась, когда база данных открывается, вы должны указать ее в макросе `AutoExec`. Следующие шаги проведут вас через процесс присоединения функции `ChkLink` к макросу.

1. Откройте базу данных Scale Front. В разделе **Макросы** нажмите **Создать** (New), чтобы открыть чистый лист макроса.
2. Раскройте ниспадающий список в первой строке столбца **Макрокоманда** (Action) и выберите **Запуск программы** (RunCode).
3. Нажмите F6, чтобы перейти к строке **Имя функции** (Function Name). Наберите `ChkLink()` в этом окне.
4. Нажмите **Сохранить** (Save) и назовите макрос `Autoexec`. Закройте макрос.
5. Откройте модуль Check Link, чтобы просмотреть программу, присоединенную к макросу. Если вы хотите- протестировать процедуру, переименуйте `Table3` в Table2 и заново откройте Scale Front (сначала закройте, а потом откройте). Закройте модуль.

Что дальше?

Эта глава рассказала об изобретательном преодолении препятствий при разработке с помощью автоматизации. Следующая глава будет о решении проблем разработки в реальных условиях с помощью объединения и разделения данных. Вы научитесь, как справляться с препятствиями с помощью приемов программирования.

19
Связывание и разделение данных

В гл. 18 вы изучили, как работать с данными из внешнего источника. Данная глава расширяет эту тему, объясняя, что надо делать после того, как вы получили данные.

Очень вероятно, что при работе с Microsoft Access вам понадобится получать данные из внешних источников. Возможно, ваш друг имеет файл Excel и хочет, чтобы вы преобразовали его в файл Access. Или, возможно, ваша компания хочет, чтобы вы преобразовали файл ASCII (текстовый), полученный в Access из компьютера клиента.

Какая бы ситуация ни была, вам необходимо понять, как связать (concatenate) данные или разделить (parse) их. Причина этого в том, что данные, полученные из внешнего источника, не всегда сохранены в нужном формате. Чтобы быть более конкретными, перечислим ниже то, что вы изучите:

- как связывать данные;
- как разделять данные;
- как действовать в реальных условиях, когда вы используете связывание (конкатенацию) и разделение для решения задач.

Искусство конкатенации (связывания)

Например, необычно импортировать таблицу с датой «111197». Предполагая, что строка уже была разбита на «месяц» «день» и «год», как вы свяжете строку в дату с добавлением символов «\»?

Так как все это текстовая информация, вы можете использовать оператор «+» (который используется почти во всех языках программирования) для связывания следующим образом:

```
"11" + "/" + "11" + "/" + "97"
```

Если каждая запись в поле, содержащем это число, имеет «\», вы можете преобразовать это поле в поле данных Access. Оператор «+» ведет себя по-разному в зависимости от того, с каким типом данных он работает. Табл. 19.1 показывает различные результаты, получаемые при работе с разными переменными в выражениях с оператором «+».

Таблица 19.1. Различные результаты при использовании оператора «+»

Когда	Что происходит
Обе переменные числового типа (Byte, Boolean, Integer, Long, Single, Double, Date, Currency или Decimal)	Сложение
Обе переменные строкового типа	Конкатенация (сложение строк)

Когда	Что происходит
Одна переменная числового типа, а остальные (представляющие числа) произвольного типа, кроме null	Сложение
Одна переменная числового типа, а остальные (представляющие строки) произвольного типа, типов кроме null	Ошибка несоответствия
Одна переменная - строка, а остальные (представляющие строки) произвольного типа, кроме null	Конкатенация
Любая переменная пустого типа (универсальный тип данных в некоторых языках программирования и в OLE)	Возврат в виде результата неизмененного выражения
Одна переменная числового типа, а остальные - строки	Ошибка несоответствия типов
Любая переменная - null	Результат null

Хотя и полезно знать, что вы можете применять оператор «+» более широко, чем просто для сложения чисел, есть и недостатки в его использовании. Представьте, что в предыдущем примере вы забыли заключить последнее число в кавычки.

Если вы нажмете Ctrl+G для вызова окна прямого ввода, наберете ? "11" + "/" + "11" + "/" + 97 и нажмете Enter, то получите сообщение об ошибке несоответствия типов. Причина этой ошибки в том, что 97 не имеет кавычек. Другими словами, вы не можете объединять числа с текстом используя оператор «+». Вы можете использовать встроенную функцию Access для объединения типов данных, как в примере на рис.:

```
? "11" + "/" + "11" + "/" + cstr(97)
```

Функция Cstr преобразовывает числовой тип в строку. Если вы наберете ? "Today's date " + date, то получите еще одно сообщение об ошибке несоответствия типов. Но если вы наберете ? "Today's date: " + cstr(date), будет получена сегодняшняя дата: 06/01/01 (замените ее текущей).

Есть более простой метод объединения данных разных типов. Он использовался на протяжении всей книги. Оператор ampersand_(&) при конкатенации преобразовывает данные нестроковых типов в строки. Вы можете комбинировать практически любые типы данных используя этот оператор.

Объединение полей

Конкатенация часто используется для вычисляемых полей, как демонстрирует следующий оператор SELECT:

```
SELECT FirstName & " " & LastName AS [CustomerName] from Customers;
```

Этот пример показывает, как надо вставлять место между двумя объединяемыми полями.

Оператор AS используется для быстрого переименования поля. Оператор & очень удобен, когда вы хотите использовать переменные со строками SQL. Даже имена таблиц и полей могут быть представлены переменными, если использовать этот метод. Это означает, что, если вы хотите выполнить один и тот же запрос над разными таблицами с одинаковыми именами полей, вы можете сделать это, используя переменную для имени таблицы. Если вы хотите выполнять один и тот же запрос для разных таблиц с разными именами полей, вы также используете переменные для имен полей.

Объединение переменных

Рассмотрим следующий листинг, который использует переменную с амперсандом, чтобы получить доступ к текущему сертификационному номеру.

```
Dim dCert As Double
dCert = Me.CertNo
DoCmd.RunSql "INSERT INTO Amend SELECT [Clients].* " _
& "FROM [Clients] WHERE [Clients].[CertNo] =" & dCert & ";"
```

Этот код сначала помещает текущее значение сертификационного номера из базовой таблицы в переменную dCert. Затем он использует переменную как критерий для добавляемого запроса. Так как поле CertNo уникально, программа выбирает одну запись для добавления в таблицу Amend (изменений), таким образом сохраняя историю добавленных сертификатов, когда они модифицируются. Обратите внимание на то, что амперсанды используются для того, чтобы вставить элементы строки вместе. Не важно, если переменная dCert имеет числовой тип (double).

Используя переменные, вы даже можете ссылаться на множественные вхождения полей в строке, применяя конкатенацию. Например, если у вас есть строка SQL, которая ссылается три раза на одно и то же поле (в count, sum и average), вы можете найти и удалить каждое вхождение поля так: `" & FieldVar & "` (в предположении, что вы объявили переменную FieldVar). Так вы легко могли бы выполнять такой же запрос для другого поля. Ссылаться на таблицу так же легко, как ссылаться на имя поля в строке SQL с переменной. Есть несколько несоответствий при использовании кавычек с переменными в строке SQL, которые будут рассмотрены в следующей главе.

Искусство разбора

Давайте представим, что ваш босс пришел к вам с, как ему кажется, прекрасной идеей. Телефонная компания прислала ему диск, содержащий информацию о телефонных звонках за последний месяц. Они могут предоставить ее в течение месяца. Он хочет, чтобы вы переместили эту информацию в базу данных Access так, чтобы он мог сказать, кто проговорил дольше всех. Когда вы, наконец, увидели телефонный список, он был похож на таблицу на рис. 19.2 (если убрать из него несколько тысяч записей).

Таблица 19.2. Текстовый файл данных для таблицы телефонного списка

2790017	9185436525	20001215 351NGTBixby	13538547	24	168
2790017	9184681269	20001129 824DAYBixby	10538547	18	180
2790017	9184681181	20001227 101NGTBixby	3538547	4	28
2790017	9185459191	20001227 102NGTBixby	2538547	2	14
2790017	9185448823	20001227 301NGTBixby	3538547	4	28
2790017	9185452929	20001230 550NGTBixby	8538730	14	98
2790017	9187274948	20001209 1522DAYBixby	15538547	26	260
2790017	9185455118	20001226 2020EVEBixby	2538547	3	24
2790017	9185442288	20001230 56NGTBixby	31538547	55	385

Вы ничего не говорите своему боссу, но думаете про себя: «Как же мне привести это в порядок?» С помощью звонка в телефонную компанию вы получаете документ, показывающий структуру полей с начальными и конечными характерными номерами каждого поля. Когда ваш босс приходит снова, он говорит, что некоторые поля не нужны.

С этого момента вы начинаете обдумывать, какие инструменты есть в вашем распоряжении. Хотя вы и можете использовать возможность Excel **Текст по колонкам** (Text to Columns), было бы лучше иметь инструмент в Access, который можно сохранить и использовать снова и снова. Ваш босс хочет, чтобы все было автоматизировано от начала до конца так, чтобы можно было этим пользоваться каждый месяц. Вот сценарий, с которым вы столкнетесь. Ну и что вы собираетесь делать?

Этап 1

Вы можете начать с написания программы для импорта текстовых файлов. Лучший способ сделать это - импортировать сначала вручную так, чтобы вы могли создавать «детальное» имя для импорта. Затем примените детализацию к вашей программе. Детализация - это табличная структура, которая позволяет Access узнавать, какие имена полей, типы данных и размеры полей должна иметь импортная таблица.

Во время вашего первого процесса ручного импорта в базе данных создаются две системные таблицы Access для записи атрибутов полей в таблице. Эти таблицы содержат вашу детализацию. Все последующие детализации также будут регистрироваться в этих таблицах. Таблица MSysIMEXColumns создается для записи таких атрибутов, как имя поля, его ширина и начальная позиция каждого поля. Вторая таблица, называемая MSysIMEXSpecs, содержит среди прочего имя детализации и тип файла.

Если вы хотите импортировать эти детализации в другую базу данных, просто проверьте «Скрытые» и «Системные объекты» (Hidden and System objects) в меню Сервис (Tools), Параметры (Options), Вид (View) перед импортом, чтобы эти две системные таблицы можно было видеть, выбрать и импортировать. Если у вас есть три детализации в двух таблицах, а нужны только две импорти-

рованные, вы легко можете удалить ненужную детализацию после импортирования. Следующие шаги проведут вас через процесс ручного импортирования.

1. Откройте базу данных ExportImport.
2. Выберете **Файл** (File), «Внешние данные» (Get External Data), «Импорт» (Import), чтобы открыть диалоговое окно «Импорт» (Import), показанное на рис. 19.1.

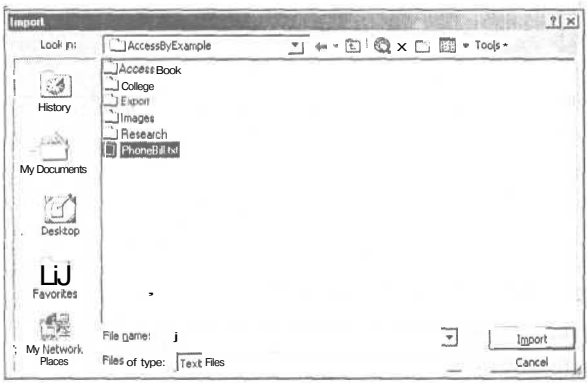


Рис. 19.1. Выберите Текстовые файлы (Text Files) в выпадающем меню Тип файлов (Files of type) диалогового окна Импорт (Import)

3. Выберете «Текстовые файлы» (Text Files) в выпадающем меню «Тип файлов» (Files of type), чтобы установить файловый тип для импорта (рис. 19.1).
4. В папке AccessByExample нажмите на PhoneBill.txt и затем нажмите «Импорт» (Import), чтобы открыть окно мастера «Импорт текста» (Import Text), показанное на рис. 19.2.

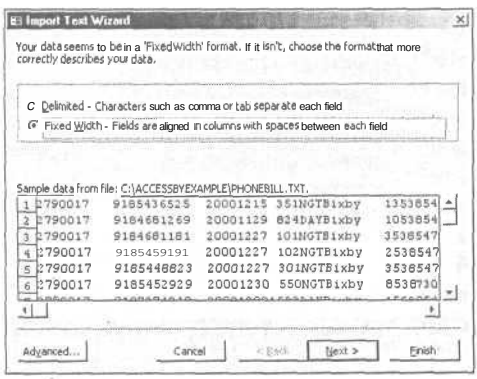


Рис. 19.2. Окно мастера Импорт текаа (Import Text) показывает выборочные данные из файла, который вы импортируете

5. На первом изображении должно быть выбрано **Фиксированная ширина** (Fixed Width). Если нет, выберите и нажмите **Далее** (Next), чтобы открыть следующее окно мастера, показанное на рис. 19.3.

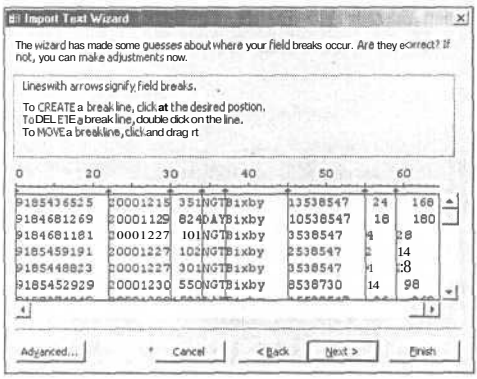


Рис. 19.3. Мастер Импорт текста (Import Text) позволяет вам создавать шаблоны, которые будут использоваться снова и снова

6. На линейке разметки щелкните на 30, 34 и 37, чтобы разделить разные поля (рис. 19.3), затем нажмите «Дополнительно» (Advanced).
7. Поместите девять имен полей и типов, показанных в табл. 19.3 и на рис. 19.4, в раздел информации о поле.

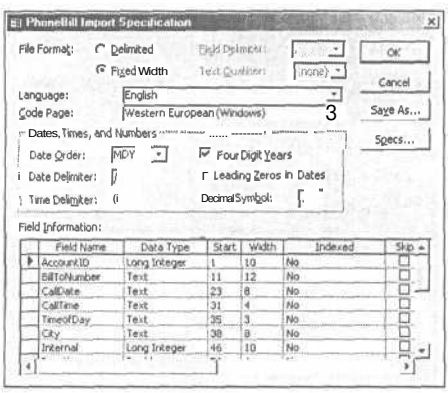


Рис. 19.4. Детализация импорта регистрирует атрибуты поля для создаваемой таблицы

Таблица 19.3. Детализации для таблицы телефонного списка

Поле	Тип данных	Начало	Ширина	Индекс	Пропуск
AccountID	Long Integer	1	10	No	0
BillToNumber	Text	11	12	No	0
CallDate	Text	23	8	No	0
CallTime	Text	31	4	No	0

Поле	Тип данных	Начало	Ширина	Индекс	Пропуск
TimeofDay	Text	35	3	No	0
City	Text	38	8	No	0
Internal	Long Integer	46	10	No	0
Duration	Double	56	4	No	0
Cost	Double	60	5	No	0

8. Нажмите **Сохранить как** (Save As), как показано на рис. 19.5.

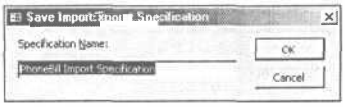


Рис. 19.5. Диалоговое окно Сохранение спецификации импорта и экспорта (Save Import/Export Specification) позволяет вам сохранять детализацию полей, которые могут быть использованы позже

9. Продолжайте нажимать Далее (Next) без выбора первичных ключей. Когда вы доберетесь до последнего окна, нажмите **Дополнительно** (Advanced), еще раз сохраните детализацию без первичного ключа и нажмите **ОК.** Убедитесь, что имя таблицы PhoneBill, и нажмите **Готово** (Finish), чтобы импортировать файл. Вы получите сообщение о том, что импортирование прошло успешно.

Теперь у вас есть таблица и детализация импорта, которая может быть использована в других базах данных. Вы можете автоматизировать процесс импорта, используя метод TransferText, работающий в качестве аргумента, так что случайностей произойти не может.

Фактически цель ручного импортирования - это создать детализацию, которую можно будет снова использовать, когда понадобится. На втором этапе вы научитесь, как автоматизировать процесс импортирования текстовых файлов.

Этап 2

Хотя вы уже произвели некоторое разбиение, надо сделать еще больше, чем вы займетесь на третьем этапе. На этом этапе вы сконцентрируетесь на том, что сделать в процедуре. Работа над предыдущими примерами дала две вещи. Во-первых, она познакомила вас с ручным импортированием текстовых файлов. Во-вторых, она научила вас, как устанавливать формат детализации для файла, который к этому моменту должен быть у вас в базе данных. Теперь вы можете сконцентрироваться на создании макроса переноса, выполняя следующие шаги:

1. В разделе **Макросы** (Macros) нажмите **Создать** (New), чтобы открыть окно макроса.
2. В строке **Макрокоманда** (Action) выберите TransferText (Преобразовать текст) из выпадающего окна, чтобы активизировать раздел **Аргументы макрокоманды** (Action Arguments) для этого действия.

3. Выберите свойства для TransferText (Преобразовать текст), как показано в табл. 19.4, которая совпадает с рис. 19.6.

Таблица 19.4. Аргументы для макрокоманды TransferText (Преобразовать текст)

Transfer Type:	Import Fixed Width
Specification Name:	PhoneBill Import Specification
Table Name:	PhoneBill
File Name:	c:\AccessByExample\phonebill.txt
Has Field Names:	No

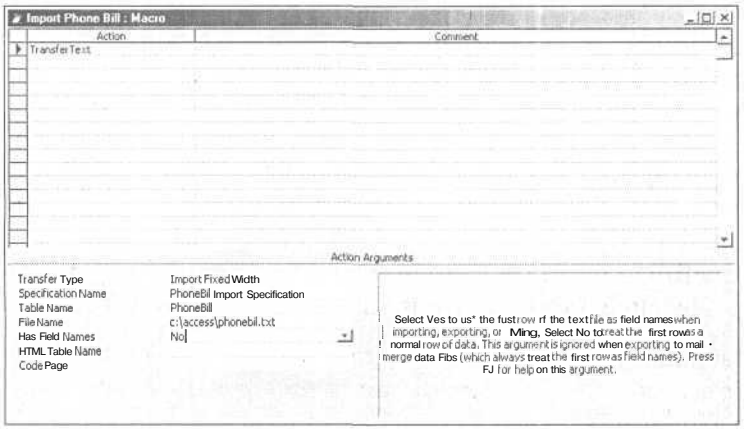


Рис. 19.6. Обратите внимание на PhoneBill Import Specification во второй строчке макроса импорта

4. Сохраните макрос как Import Phone Bill и закройте его.

Теперь у вас есть макрос, который может быть преобразован в процедуру, которая послужит основой для дальнейшего построения.

Этап 3

Этот этап работы над нашей задачей интенсивно использует запросы как для измененных табличных структур, так и для управления данными. До создания запросов вы должны преобразовать макрос Import Phone Bill в модуль, совершив следующие шаги:

1. В разделе **Макросы** (Macros) щелкните правой кнопкой на Import Phone Bill и выберите **Сохранить как** (Save As) из меню быстрого вызова, чтобы открыть диалоговое окно **Сохранение** (Save As).
2. Нажмите на ниспадающее окно Как (As) и затем выберите **Модуль** (Module). Нажмите OK, чтобы сохранить макрос как Copy of Import Phone Bill, и откройте диалоговое окно **Преобразование макроса** (Convert macro).

3. Нажмите кнопку **Преобразовать** (Convert), когда появится диалоговое окно **Преобразование макроса** (Convert macro). Оставьте обе галочки установленными на проверку ошибок и комментариев макроса.
4. После нажатия OK на сообщении о завершении преобразования у вас появляется доступ к окну **Модуль** (Module).

ПРЕДУПРЕЖДЕНИЕ. Всякий раз, когда вы создаете модуль из макроса, в списке модулей открывается первый модуль. Например, если вы создаете модуль из временной памяти и присваиваете ему имя, начинающееся с «а» (ставящее его первым в списке), откроется этот модуль вместо того, который вы создаете из макроса. Таким образом, вы должны открыть только что созданный модуль из окна Module. Если вы не видите окна Project-Export слева от окна модуля, нажмите Ctrl+R. После этого вы можете видеть модуль в окне Project-Export и открывать его.

5. В окне Проект-Export (Project-Export) дважды щелкните на **Преобразованном макросе** - Import Phone Bill (Converted Macro Import Phone Bill), чтобы развернуть окно **Модуль** (Module), показанное на рис. 19.7.

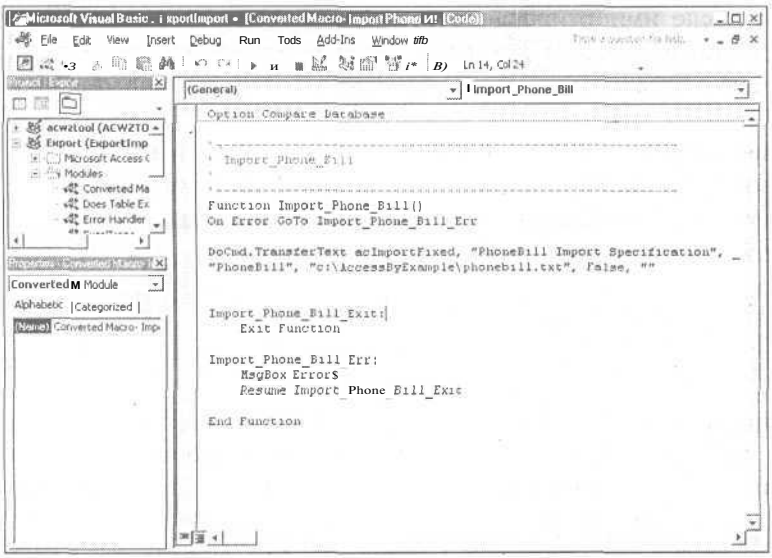


Рис. 19.7. Преобразованный модуль содержит правильные аргументы, вставленные для вас в метод TransferText

6. Вставьте подчеркнутый символ вслед за элементом PhoneBill Import Specification,.
Убедитесь, что поставлена запятая. После подчеркнутого символа нажмите Enter, чтобы получить изображение как на рис. 19.7.

ЗАМЕЧАНИЕ. Вставка подчеркнутого символа просто делает ваш код более удобочитаемым. Кроме того, убедитесь в том, что подчеркнутый символ - последний символ в строке. Компилятор воспринимает это как одну длинную строку, как если бы она была размещена на одном уровне (без переноса). Тем не менее теперь у вас есть строка на двух строках для лучшей читаемости и вам не надо перемещать ее, чтобы прочитать всю строку.

7. Нажмите кнопку Save, чтобы сохранить модуль, и оставьте модуль открытым для выполнения следующего упражнения.

Хотя все, кажется, идет хорошо, есть проблема с двумя полями, которые должны быть адресованы. Если вы вернетесь и проверите табл. 19.3, вы можете заметить, что поля CallDate и CallTime слишком маленькие. CallDate имеет два символа, а CallTime - четыре. Чтобы преобразовать эти поля в тип данных **Дата/время** (date/time), им нужно место, чтобы, вмещать дополнительные символы. Например, 12/12/2001 имеет 10 символов, включая косую черту. Аналогично 12: 30 имеет пять символов, включая двоеточие. Вы не можете добавлять дополнительные символы в файл спецификации, так как добавление места в одно поле перекрывает следующее поле. Единственная возможность - это поменять размер поля таблицы после импортирования без уничтожения данных. Таким образом, у вас есть достаточно места для разделения полей и перестраивания строк с использованием конкатенации для добавления таких дополнительных символов, как косая черта.

Запрос определения данных - это просто инструмент, нужный вам для изменения табличной структуры для двух полей. К сожалению, вы не можете изменить оба поля в одном запросе. Следующие два запроса определения данных добавляют дополнительные символы в соответствующие поля:

```
ALTER TABLE PhoneBill
ALTER COLUMN CallDate TEXT(10)
ALTER TABLE PhoneBill
ALTER COLUMN CallTime TEXT(5)
```

Чтобы автоматизировать эти запросы, вам нужен механизм, чтобы запускать их из VBA. Как вы до этого узнали, вы можете запускать рабочие запросы и специальнае запросы SQL используя метод RunSQL объекта DoCmd. Сделайте следующие шаги, чтобы добавить код:

1. В окне модуля **Преобразованный макрос - Import Phone Bill** (Converted Macro - Import Phone Bill), оставленном открытым в последнем упражнении, вставьте следующий код после двух строк DoCmd.TransferText:

```
DoCmd.RunSQL "ALTER TABLE PhoneBill " _
& "ALTER COLUMN CallDate TEXT(10)" 'Расширить второе поле
DoCmd.RunSQL "ALTER TABLE PhoneBill " _
& "ALTER COLUMN CallTime TEXT(5)" 'Расширить первое поле
```

2. Сохраните и закройте модуль.

Следующий шаг включает получение полей, готовых к преобразованию в тип **Дата/время** (Date/Time). Следующие функции, которые доступны в вашей базе данных, ExportImport делают это:

```
Function MakeDate(strVar As String)
Dim sParse As String
'Эта функция как разбирает данные по кусочкам, так и собирает их в одно целое

  sParse = Mid(strVar, 5, 2) & "/" & Right(strVar, 2) _
  & "/" & Left(strVar, 4)

MakeDate = sParse
End Function

Function MakeTime(strVar As String)
'Эта программа объединяет и производит структурный анализ (parse)
Dim sParse As String
strVar = LTrim(strVar)
  Select Case Len(strVar)
    Case 2
    sParse = "00:" & Right(strVar, 2)
    Case 3
    sParse = "0" & Left(strVar, 1) & ":" & Right(strVar, 2)
    Case 4
    sParse = Left(strVar, 2) & ":" & Right(strVar, 2)
  End Select
MakeTime = sParse
End Function
```

Первая процедура разделяет строку и перестраивает ее, используя конкатенацию. Например, возьмем дату 20001227. Первые четыре символа (2000) представляют год. Таким образом, вы начинаете с 12 (с пятой позиции представляющей Декабрь) после года, используя строковую функцию Mid. Затем вы берете последние два символа для дня. После этого выбираете первые четыре символа для года. Между каждыми из этих частей, данных вам, надо добавить косые черты. Заканчиваем следующим отрывком кода:

```
sParse = Mid(strVar, 5, 2) & "/" & Right(strVar, 2) _
& "/" & Left(strVar, 4)
```

Вторая процедура немного сложнее. Аргументы для строковых функций Left и Right зависят от длины строки. Снова проверьте табл. 19,2. Значения времени основаны на формате времени "military time" (20:20 представляется в виде 8:20 p.m.). Таким образом, 2020 (с двоеточием, разделяющим числа 20) представляет собой 20:20 (8:20 p.m.). Access позаботится о преобразовании из формата «military time» в стандартный, когда поля преобразовываются в «Дату/время». Тем не менее вы должны добавить двоеточие. Если этого не сделать, преобразование не осуществится.

Если размер строки - два символа, вы должны добавить два головных нуля за двоеточием. Если размер строки - три символа, вы должны добавить один го-

...ий нуль за первым символом, за двоеточием, и два последних символа и т. д. Завершим оператором Select - Case, как показано далее:

```
Select Case Len(strVar)
    Case 2
    sParse = "00:" & Right(strVar, 2)
    Case 3
    sParse = "0" & Left(strVar, 1) & ":" & Right(strVar, 2)
    Case 4
    sParse - Left(strVar, 2) & ":" & Right(strVar, 2)
  End Select
```

Вы можете комбинировать функции с одним обновленным запросом, чтобы переделывать строки в таблице. В базе данных ExportImport есть таблица *с* именем RawPhoneBill. Необходимость приставки raw (дословно «необработанный». - *Пер.)* объясняется тем, что данные-даты из импорта не преобразованы. Дважды щелкните на таблице, показанной на рис. 19.8.

RawPhoneBill : Table

AccountID	BillToNumber	CallDate	CallTime	TimeofDay	City	Internal	Duration	Cost
2790017	9185436525	20001215	351	NGT	Bixby	13538547	24	168
2790017	9184681269	20001129	824	DAV	Bixby	10538547	18	180
2790017	9184681181	20001227	101	NGT	Bixby	3538547	4	28
2790017	9185459191	20001227	102	NGT	Bixby	2538547	2	14
2790017	9185448823	20001227	301	NGT	Bixby	3538547	4	28
2790017	9185452929	20001230	550	NGT	Bixby	8538730	14	98
2790017	9187274948	20001209	1522	DAY	Bixby	15538547	26	260
2790017	9185455118	20001226	2020	EVE	Bixby	2538547	3	24
2790017	9185442288	20001230	1 №	NGT	Bixby	3153s547	55	385

Record: 1 of 9

Рис. 19.8. Преобразованный текстовый файл PhoneBill (теперь называемый RawPhoneBill) после импорта, но до преобразования полей с датами

Чтобы понять эту часть процесса преобразования, проверьте запрос на тестовой таблице до включения его в код программы. Следующие шаги проведут вас через процесс создания обновленного запроса.

1. В разделе **Запросы** (Queries) нажмите на **Создание запроса в режиме конструктора** (Create Query in Design view), чтобы открыть диалоговое окно **Добавление таблицы** (Show Table).
2. Из диалогового окна **Добавление таблицы** (Show Table) дважды щелкните по RawPhoneBill, чтобы добавить таблицу в сетку запроса (Query design), и затем нажмите **Закрыть** (Close).
3. Из таблицы RawPhoneBill дважды щелкните на полях CallDate, CallTime и Cost.
4. Нажмите кнопку на панели инструментов **Тип запроса** (Query Type) и из выпадающего меню выберите **Обновление** (Update Query).
5. В строке **Обновление:** (UpdateTo:) для поля CallDate наберите MakeDate([CallDate]).
6. В строке **Обновление:** (UpdateTo:) для поля CallTime наберите MakeTime([CallTime]).

7. В строке **Обновление:** (UpdateTo:) для поля Cost наберите `[Cost]*0.01`. Вы умножаете время на `.01`, чтобы преобразовать поле в доллары и центы. Ваш запрос должен выглядеть как на рис. 19.9.

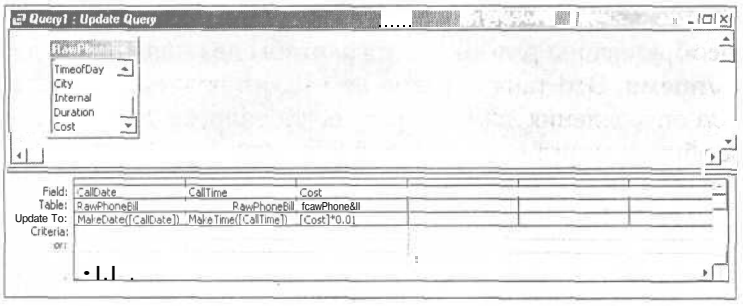


Рис. 19.9. Этот обновленный запрос использует функции, чтобы переделать данные в текстовых полях для приспосабливания к типу данных - датам

8. Нажмите кнопку **Запуск** (Run), чтобы запустить запрос. Нажмите Да (Yes), когда придет время подтвердить обновление этих данных. Закройте запрос, не сохраняя его.
9. В разделе **Таблицы** (Tables) дважды щелкните на таблице RawPhoneBill, чтобы проверить результаты запроса, показанные на рис. 19.10. Поля дат будут должным образом преобразованы. Поле Cost будет в долларах и центах.

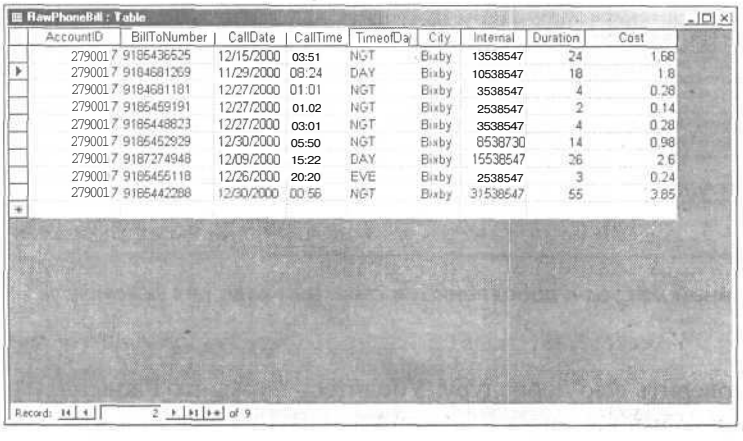


Рис. 19.10. Результаты обновленного запроса показывают правильно преобразованные данные

10. В разделе **Модули** (Modules) нажмите на **Преобразованный макрос - Import Phone Bill** (Converted Macro Import Phone Bill) и затем нажмите **Конструктор** (Design).

11. Добавьте следующий код после последней строки ALTER COLUMN, которая является SQL-эквивалентом созданного вами запроса.

```
DoCmd.RunSQL "UPDATE PhoneBill SET PhoneBill.CallDate = " _
& "MakeDate([CallDate]), PhoneBill.CallTime = " _
& "MakeTime([CallTime]), PhoneBill.Cost = [Cost]*0.01;"
```

12. Нет смысла делать преобразование только для того, чтобы два поля привести вручную к виду **Дата/время**. Это также можно автоматизировать. Добавьте следующие два запроса определения данных сразу после запроса Update, чтобы позаботиться о преобразовании текста в дату двух полей:

```
DoCmd.RunSQL "ALTER TABLE PhoneBill " _
& "ALTER COLUMN CallDate DATE" 'Изменяет тип данных
DoCmd.RunSQL "ALTER TABLE PhoneBill " _
& "ALTER COLUMN CallTime DATE" 'Изменяет тип данных
```

После введения кода окно **Модуль** (Module) должно выглядеть как на рис. 19.11.

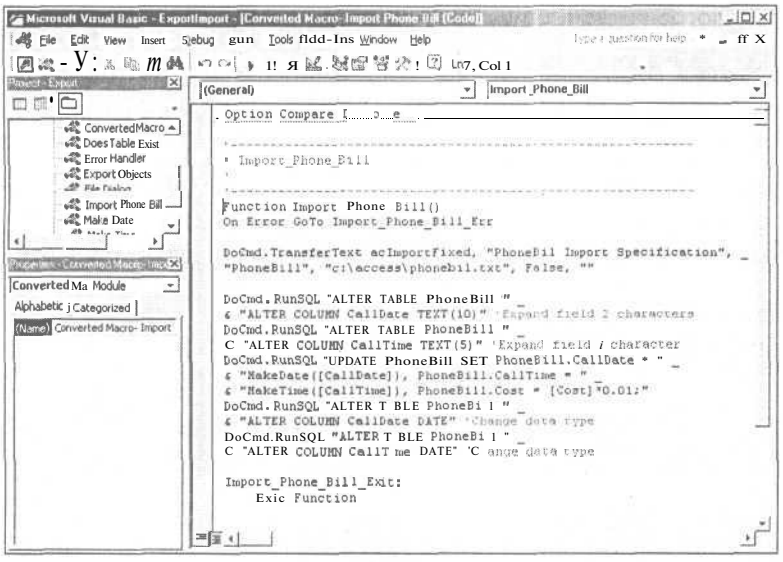


Рис. 19.11. Преобразованный макрос с добавленными строками кода для реконструкции строк

Теперь вы готовы проверить всю процедуру. Убедитесь в том, что Phonebill.txt, который вы загрузили, находится в директории c:\AccessByExample или что программа указывает на ту директорию, в которой этот файл находится. Если все работает правильно, программа берет файл из необработанного текстового файла в должным образом форматированную таблицу Access.

ЗАМЕЧАНИЕ. Прелесть этой процедуры в том, что ее можно использовать постоянно или когда ваш босс попросит (согласно воображаемому сценарию.)

Теперь, когда процедура создана, проделайте следующие шаги, чтобы проверить ваш код.

1. Выберите Отладку (Debug) - Компилировать (Compile) из меню, чтобы проверить код.
2. Поместите ваш курсор где-нибудь на строке, где объявлена функция, и нажмите Запустить процедуру (Run Sub/User Form) или F5.
3. Сохраните и закройте модуль. Откройте таблицу PhoneBill, чтобы проверить поля дат.

Сценарий, который вы только что выполнили, очень похож на реальный проект. Фактически он был взят из реального проекта. Это подтверждает то, что вы можете автоматизировать почти любой проект, который включает в себя пошаговые процедуры, повторяющиеся из месяца в месяц.

Что дальше?

В этом проекте применялось многое из того, что вы уже изучили в предыдущих главах. Это подтвердило, что лучше всего учиться на примерах. В следующей главе разбирается, как работать с кавычками, неопределенными значениями (nulls) и датами. Вы увидите, как справляться с большими проблемами, спрятанными в маленьких вещах.

20
Борьба с большими проблемами, возникающими из маленьких неприятностей

В гл. 19 вы изучили разделение и связывание данных. Эта глава учит справляться с другими подводными камнями, которые может встретить программист и которые начинаются с, казалось бы, маленьких неприятностей. Когда вы заранее знаете, с чем встретитесь и как с этим справляться, единственное, что вам еще может помочь, - улучшение программистских навыков. Вот причина, по которой так много было уделено преградам, возникающим при разработке. Когда начинаются проблемы, вы уже инстинктивно будете чувствовать, какой инструмент нужно задействовать, чтобы с ними справиться. Конкретно вы узнаете:

- как освоить кавычки и знак &,
- как использовать функцию BuildCriteria,
- как проверять кавычки и &,
- что надо сделать для проверки неопределенных значений (nulls),
- в чем разница между неопределенными значениями и пустыми строками в поле,
- как работать с неопределенными значениями и пустыми строками в поле,
- в чем разница между функциями Nz и IsNull,
- как задавать формат даты,
- как считать промежутки времени между датами.

Использование кавычек и амперсанда

Использование переменных с кавычками и & с первого взгляда выглядит простым. Удивительно, как много неприятностей и срывов в работе эти кажущиеся простыми операции доставили начинающим и продвинутым разработчикам Access. Здесь разобраны несколько способов, как справляться с трудностями, которые могут появиться.

Кавычки

Пародоксально, что маленькие вещи в Access могут быть наиболее сложными для разработчика. Кавычки - лучший тому пример. В некоторых ситуациях операции с кавычками не выглядят такими сложными. Например, если вы пытаетесь создать фильтр, используя текстовое поле, вы просто употребляете такое выражение:

```
[State] = "Oklahoma"
```

В этом месте нет ничего сложного. Появляются неприятности тогда, когда вы пытаетесь вставить строковую переменную внутрь другой строки. Например, представьте, что у вас есть переменная с именем strState, которую вы хотите использовать как часть выражения для фильтра в отчете. Если вы пытаетесь вывести переменную strState в окне проверки (immediate), вы получите Oklahoma (без кавычек), но не "Oklahoma"(с кавычками). Чтобы иметь возможность получить переменную для вывода строки, заключенной в кавычки, вы должны определить строку переменной таким образом:

```
strState = """Oklahoma"""
```

Вы должны поставить пару кавычек вокруг кавычек, окружающих Oklahoma, чтобы на выходе получить строку, заключенную в кавычки. Другими словами, вам нужно по три кавычки с каждой стороны строки. Это начинает становиться интересным. Но будет еще интереснее.

Вы хотите, чтобы другая переменная с именем strWhere содержала все выражение, которое использовано для фильтра. Представим, что strState содержит строку "Oklahoma", вы пытаетесь поместить все операторное выражение where в переменную, как показано ниже:

```
strWhere = "[State] = " strState
```

Это генерирует ошибку компиляции из-за неправильного синтаксиса. Итак, вы решаете поместить вторую кавычку в конце строки, как показано ниже:

```
strWhere = "[State] = strState"
```

С первого взгляда кажется, что этот подход сработает. Единственная проблема этого метода - это то, что вы достигаете следующего, присваивая переменной strWhere:

```
[State]= strState
```

Вам не нужно буквенное значение strState в вашем выражении, так как неясно, есть ли у вас такое выражение в вашей базе данных. Итак, вы становитесь действительно творческим работником и пытаетесь связывать строки, как показано ниже:

```
strWhere = "[State]=" & strState
```

В этом месте вы думаете, что все получилось, но, на самом деле это обман. Если вы попытаетесь использовать это выражение в процедуре, то получите следующее:

```
[State] = Oklahoma
```

Это выглядит правильно, но, если вы попытаетесь вставить это выражение в отчет как фильтр кода, вы не добьетесь ничего, кроме разочарования, потому что отчет требует, чтобы строковые значения были заключены в кавычки. То, чем вы хотите закончить, показано ниже:

```
[State] = "Oklahoma"
```

Как же вы получите это? Начните с помещения всего выражения, которое вам нужно, в кавычки, как показано ниже:

```
"[State]=Oklahoma"
```

Теперь окружите встроенную строку (строка внутри строки) тремя кавычками (как делали до этого), как показано ниже:

```
"[State]="""Oklahoma"""
```

Помните, что первая и последняя кавычки предназначены для всего выражения. Далее свяжите переменную, содержащую "Oklahoma", с амперсандом, как показано ниже:

```
"[State]="""& strState &"""
```

Наконец-то у вас получилось! Теперь присвойте всю строку переменной, как показано ниже:

```
strWhere = "[State]="""& strState &"""
```

Вы также можете использовать одинарную кавычку для фильтров, так что следующее выражение тоже будет работать:

```
strWhere = "[State] = '" & strState & "'"
```

Чтобы запомнить технику добавления вложенных кавычек, выполните следующие шаги:

1. Начните с заключения выражения, которое вы хотите, в кавычки (например, «[State] = Oklahoma»).
2. Вставьте набор из трех кавычек (двойных) по обеим сторонам вложенной строки (например, "[State]="""Oklahoma""").
3. Переместите вложенную строку с переменной и поместите амперсанды по обеим сторонам переменной, разделенной пробелами (например, "[State]="""& strState &""").
4. Свяжите все выражение с переменной (например, strWhere = "[State]="" ' & strState &"").

Другой способ, заслуживающий внимания, использует chr$(34), чтобы вставить кавычку. Из гл. 16 вы, возможно, помните, что символы клавиатуры соответствуют кодам символов, представленным в числами. Вы можете установить, какой код соответствует определенному символу, используя функцию asc. Например, чтобы определить код двойной кавычки («), просто наберите ? asc("") в окне непосредственного ввода и оно возвратит 34. Вы должны использовать четыре кавычки, чтобы определить строку в одну кавычку. Теперь вы установили, что 34 - это номер, который вы используете с функцией chr$. Если вы напишете то же выражение, используя этот метод, то закончите следующим:

```
"[State]=" & chr$(34) & strState & chr$(34)
```

Также вы можете присвоить chr$(34) переменной для лучшей читаемости. Если вы присвоите chr$(34) переменной sQuote, выражение будет выглядеть так:

```
"[State]=" & sQuote & strState & sQuote
```

Еще одна альтернатива - это использовать константу для представления кавычки. Вы присваиваете константе четыре кавычки, как показано ниже:

```
Const strOneQuote = """"
```

Используя этот метод, вы заканчиваете так:

```
"[State]=" & strOneQuote & strState & strOneQuote
```

Функция BuildCriteria

Если вас удивляет, почему нет собственной функции Access, которая может управлять вложенными кавычками и которая основана на критерии построения, то обрадую вас, что функция BuildCriteria идеальна для объединения элементов из фильтра или аргумента критерия. Предполагая, что значением strState является Oklahoma, вы используете функцию таким образом:

```
strWhere = BuildCriteria("[State]", 8, strState)
```

Синтаксис для функции с аргументами выглядит так:

```
BuildCriteria(FieldName, FieldType, Expression)
```

Тип поля также может быть описан с использованием внутренней константы. Например, вместо использования 8 для аргумента FieldType, представляющего тип текстового поля, вы можете использовать константу adBSTR. Табл. 20.1 показывает константы, которые работают с функцией BuildCriteria, и их эквивалентные типы и значения табличных полей.

Таблица 20.1. Константы типов данных ADO

Эквивалент Access	Константа	Значение
Текст	AdBSTR	8
Валюта	adCurrency	6
Дата	adDate	7
Число	adDecimal	14
Число	adDouble	5
Число	adInteger	3
Число	adSingle	4
Число	adSmallInt	2
Число	adTinyInt	16
Любой тип	adVariant	12

Константа adBSTR может преобразовывать даты и числовые значения. Имейте в виду, что функция BuildCriteria всего лишь генерирует критерий. Так как вы теперь знаете приемы, которые срабатывают, не обращая внимания на то, есть

ли у вас критерий или нет, то у вас есть различные способы работы с вложенными строками. Так, если у вас есть имя, например Charles "Sparky" Mahan, которое вы хотите присвоить переменной, вы знаете, что делать, чтобы вставить прозвище. В предположении, что переменная strNickName содержит "Sparky", выражение будет выглядеть так:

```
strFullName = "Charles " " & strNickName & " " Mahan"
```

Использованный прием не должен быть сюрпризом для вас, так как вы знаете принципы, по которым работает это выражение. После того как присваивание выполнено, переменная strFullName содержит Charles "Sparky" Mahan, как и полагается.

Символ фунта &

Как вы уже знаете, символы & автоматически вставляются в сетку запроса, когда вы набираете дату. А как насчет того, когда вы не в сетке запроса? Несмотря на то что в выражениях VBA с символами & легче работать, чем с кавычками, они все же могут оказаться коварными. Предположим, что у вас есть строковая переменная с именем strDate, и вам нужно разграничить (разделить) переменную символом &, как показано ниже:

```
"[AccidentDate] = # " & strDate & " #"
```

Это выражение выглядит довольно просто. Оно станет немного более сложным, когда вы работаете с диапазонами дат в выражении, как показано ниже:

```
"[AccidentDate] Between #" & strBDate & "# And #" & strEDate & "#"
```

Вы можете взять эту строку и использовать в фильтре, прикрепив ее к свойству события OnOpen отчета с помощью такой программы:

```
Private Sub Report_Open(Cancel As Integer)
Dim strBDate As String, StrEDate As String
Dim strFilter As String

strBDate = "1/1/97"
StrEDate = "1/1/98"
  strFilter = "[AccidentDate] Between #" & strBDate _
  & "# And #" & StrEDate & "#"
Me.Filter = strFilter
Me.FilterOn = True
End Sub
```

Тем не менее, так как амперсанд управляет преобразованием типа данных, вы можете переписать фильтр таким образом:

```
Private Sub Report_Open(Cancel As Integer)
Dim dtmBDate As Date, dtmEDate As Date
Dim strFilter As String

strBDate = #1/1/97#
StrEDate = #1/1/98#
  strFilter = "[AccidentDate] Between " & strBDate _
```

```
  & " And " & StrEDate
Me.Filter = strFilter
Me.FilterOn = True
End Sub
```

Даже если бы обе процедуры работали, вы можете сократить себе неприятности, задействовав функцию BuildCriteria, как вы делали до этого. Используя константу adBSTR, вы можете не волноваться о символах &, просто наберите даты, как если бы они были строкой, таким способом:

```
strCrit = BuildCriteria("[AccidentDate]", adBSTR, "Between 1/1/1996
[ic:ccc]and 1/1/98")
```

Значение переменной strCrit станет таким:

```
[AccidentDate] Between #1/1/1996# And #1/1/1998#
```

Если вы хотите использовать этот метод с переменными в процедуре, чтобы применить фильтр к отчету, то закончите так:

```
Private Sub Report_Open(Cancel As Integer)
Dim strBDate As String, StrEDate As String
Dim strFilter As String

strBDate = "1/1/96"
StrEDate = "1/1/98"
  strFilter = BuildCriteria("[AccidentDate]",_
  adBSTR, "between " & strBDate & " and " & StrEDate)
Me.Filter = strFilter
Me.FilterOn = True
End Sub
```

Проверка кавычек и амперсанда

Настал момент истины, когда вы примените то, что изучили. Вы можете проверять переменные и выражения в стандартном модуле, используя окно непосредственного ввода, перед тем, как вставлять их в модуль класса. Если вы хотите проверить переменные в процедурах события модулей класса, то установите точки прерывания так, чтобы, когда вы запускаете форму или отчет, вы могли проверить ваш код процедуры там, где установлены точки прерывания. В отличие от этого в стандартном модуле вы можете проверить переменные как используя оператор Print.Debug для просмотра выражений в окне непосредственного ввода, так и устанавливая контрольные точки и дежурные переменные (watches) для этого. В любом случае вы можете просматривать выражения, не открывая форму или отчет, в которые вы в итоге копируете выражения.

Следующие шаги проведут вас через процесс проверки.

1. Откройте базу данных Claims из папки AccessByExample.
2. Откройте отчет Loss Report в конструкторе.
3. Щелкните правой кнопкой мыши на строке заголовка отчета и выберите **Свойства** (Properties), чтобы открыть список свойств (Property sheet) формы.

4. Выберите вкладку **Событие** (Event); затем раскройте ниспадающий список события ОпОреп (Открытие) и выберите [Процедура обработки события] ([Event Procedure]). Нажмите кнопку **Создать** (Build).
5. Оказавшись в окне модуля Visual Basic Editor, выберите в меню **Вставка** (Insert), **Модуль** (Module). Если окно отладки (immediate) не открыто, нажмите Ctrl+G.
6. Наберите следующий код в окне Module:

```
Sub Test()
Dim strState As String, strFilter As String

strState = "FL"
strFilter = "[AccidentState]=" & strState & """"
Debug.Print strFilter

End Sub
```

7. Выберите в меню **Отладка** (Debug), **Компилировать Claims** (Compile Claims), чтобы проверить опечатки.
8. Нажмите кнопку **Запуск процедуры** (Run Sub/User Form). рис. 20.1 показывает следующий результат, который появляется в окне отладки (immediate):

```
[AccidentState]="FL"
```

Рис. 20.1. Выражения можно проверить в окне отладки (immediate) до того, как перемещать их в другой модуль

9. Поменяйте строку, которая начинается с strFilter, на следующую:

```
strFilter = "[AccidentState]=" & Chr$(34) & strState & Chr$(34)
```

10. Снова нажмите кнопку **Запуск** (Run Sub/User Form). Результаты должны быть теми же.

11. Мышью выделите все между Sub Test() и Debug.Print, как показано на рис. 20.2. Скопируйте строки нажатием Ctrl+C.

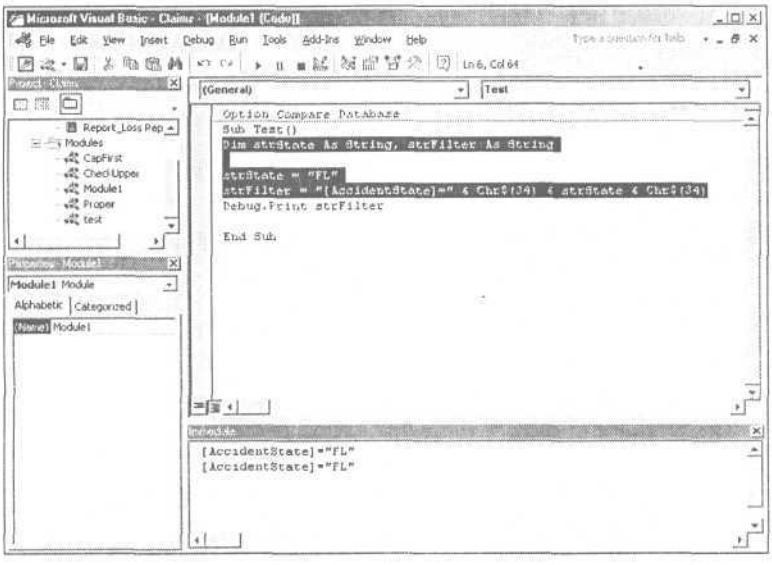


Рис. 20.2. Скопируйте и вставьте код из стандартного модуля в модуль классов

12. В окне **Проект** (Project) слева нажмите знак плюс (+) рядом с **Объектами класса Microsoft** Access (Microsoft Access Class Objects). Дважды щелкните Report_Loss Report, чтобы открыть процедуру, привязанную к событию Открытие (ОпОреп), начатую в п. 4.

13. Вставьте скопированные строки в процедуру сразу за Private Sub Report_Open(Cancel As Integer), нажав Ctrl+V.

14. Наберите Me.Filter = strFilter в строке сразу за вставленными строками. Затем наберите Me.FilterOn = True в следующей строке, чтобы все это выглядело так:

```
Private Sub Report_Open(Cancel As Integer)
Dim strState As String, strFilter As String

strState = "FL"
strFilter = "[AccidentState]=" & Chr$(34) & strState & Chr$(34)
Me.Filter = strFilter
Me.FilterOn = True
End Sub
```

15. Нажмите **Отладка** (Debug), **Компилировать Claims** (Compile Claims), чтобы проверить ошибки, затем нажмите Alt+F11, чтобы вернуться в конструктор.

16. Нажмите кнопку **Предварительный просмотр** (Print Preview), чтобы просмотреть отчет. Появится только запись Florida.

ПРЕДУПРЕЖДЕНИЕ. Если появилось окно параметров для поля состояний, то ошибка была сделана в процедуре. Например, если вы для поля AccidentState набрали AcidentState, появится окно параметров с неправильно набранным выражением.

17. Закройте окно предварительного просмотра (Print Preview). Нажмите Alt+F11, чтобы вернуться к процедуре, прикрепленной к событию Открытие (ОпОреп) в окне **Модуль** (Module).

18. Удалите код между строками Sub и End Sub и вставьте следующий код:

```
Dim strBDate As String, strEDate As String
Dim strFilter As String

strBDate = "1/1/96"
strEDate = "1/1/98"
  strFilter = BuildCriteria("[AccidentDate]", _
  adBSTR, "between " & strBDate & " and " & strEDate)
Me.Filter = strFilter
Me.FilterOn = True
```

19. Нажмите Alt+F11, чтобы вернуться в конструктор. Нажмите **Предварительный просмотр** (Print Preview), чтобы просмотреть записи между 1/1/96 и 1/1/98.

20. В окне предварительного просмотра раскройте ниспадающий список **Масштаб** (Zoom) и выберите 75 %, чтобы видеть окно **Предварительный просмотр** (Preview) в масштабе 75 % , как показано на рис. 20.3.

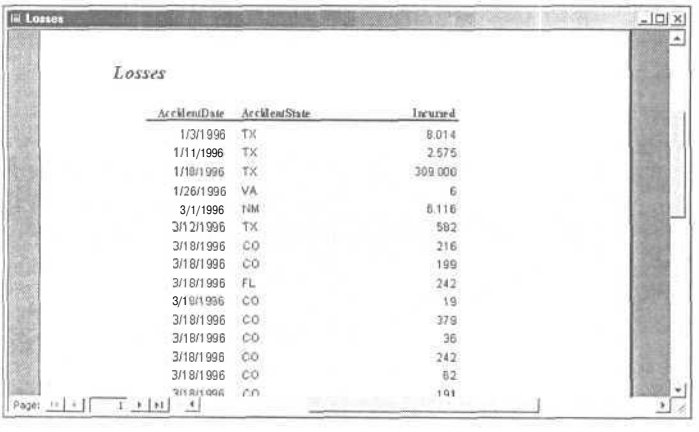


Рис. 20,3. Вы можете просмотреть отфильтрованные записи в окне предварительного просмотра (Print Preview), чтобы убедиться, что фильтр сработал

21. Закройте окно предварительного просмотра, затем закройте и сохраните отчет, но не сохраняйте стандартный модуль (Standard), который был предназначен только для проверки и практики.

Проверка значений Null

Работа с неопределенными значениями - общая проблема всех начинающих программистов Access. Когда значение поля или переменной Null (неопределенное), это значит, что в них нет достоверных данных. В отличие от пустых строк это значит, что значение, содержащееся в строке, имеет нулевую длину, или Empty (пустой - для переменных), что значит, что любая переменная не инициализирована (не присвоено значение вначале). Трудно различить все три значения, так как все они представляют отсутствие значения или то, что значение пустое. Чтобы попытаться различить их, запомните следующее:

- Variant (произвольный) - это единственный тип данных, который может содержать значения Empty (пустое), Null (неопределенное) и Nothing special;
- значения Null, Empty и пустая строка не имеют символов, которые выводятся на экран или принтер;
- значение Null распространяется на выражения, что означает, что если любой компонент выражения содержит значение Null, то все выражение преобразуется в Null;
- пустые строки могут вводиться в поля Text, Memo или Hyperlink базы данных Access, если свойство AllowZeroLength поля установлено на Yes;
- вы можете присвоить значения Empty, Null и строковые значения, как, например, Empty string (пустая строка), произвольному типу данных Variant.

Следующая процедура проливает свет на эту дилемму:

```
Sub NullVal()
Dim varItem As Variant

varItem = Null

Debug.Print varItem

End Sub
```

Есть несколько интересных мест в этой процедуре. Во-первых, обратите внимание, что тип переменной varItem - Variant. Если бы varItem была объявлена строковой, varItem = Null сгенерирует ошибку "Invalid use of null" (неправильное использование null). Это из-за того, что variant - единственный тип данных, который может содержать значения Null или Empty. Во-вторых, оператор Debug.Print показывает, что значение varItem - Null, а не пустая строка. Следующая процедура добавляет одну строку:

```
Sub NullValO
Dim varItem As Variant

varItem = Null
varItem = Nz(varItem)

Debug.Print varItem

End Sub
```

Функция Nz имеет два аргумента, но второй аргумент необязательный. Когда вы запускаете эту процедуру с добавленной строкой, она (строка) преобразовывается в “ ” или пустую строку. Вместо того чтобы видеть в окне прямого ввода Null, как в первой подпрограмме, вы не видите ничего. Вы можете убедиться, что это пустая строка, с помощью следующего условного теста:

```
If varItem = ' ' Then
Debug.Print "You got zero-length."
End If
```

Для большей информации о функции см подраздел «Nz против IsNull» далее в этой главе.

Какая разница в использовании Null или пустой строки? Во-первых, вы проверяете два значения по-разному. Это варьирование вашей исходной процедуры проверяет значения Null и пустую строку:

```
Sub NullorNot()
Dim varNullTest as Variant, varZeroTest as Variant

varNullTest = Null
    If IsNull(varNullTest) Then
       Debug.Print "This variable is Null"
    End If

varZeroTest = ""
    If varZeroTest = '' ' Then
       Debug.Print "This variable is zero-length
    End If
End Sub
```

Запомните, что varZeroTest проверяется отдельно от varNullTest. Если вы установите varNullTest как "", IsNull(varNullTest) преобразуется в False. Аналогично, если вы установите varZeroTest как Null, varZeroTest = '' преобразуется в False. Следующий вариантный тест для пустой строки использует функцию Nz:

```
Sub NullorNot()
Dim varNullTest as Variant, varZeroTest as Variant

VarNullTest = Null
    If Nz(varNullTest) = '' ' Then
       Debug.Print "First pass is zero-length"
    End If

VarZeroTest = ""
    If Nz(varZeroTest) = ' ' Then
       Debug.Print "Second pass is zero-length"
    End If '
End Sub
```

Эта процедура очень интересна, так как демонстрирует, что функция Nz преобразовывает в "" независимо от того, имеет ли проверенная переменная значение Null или пустую строку. Это можно будет удачно использовать, как вы дальше увидите.

Null против пустой строки в полях

Когда вы работаете со значениями полей, в голову приходит другое важное различие между Null и пустыми строками. Значения Null и пустые строки могут быть допустимы или недопустимы в типах Text, Memo и Hyperlink полей. В зависимости от ваших настроек значения Null и пустая строка ведут себя абсолютно по-разному. Вы можете использовать это различие для вашей пользы.

Свойства AllowZeroLength и Required работают независимо друг от друга. Свойство Required только устанавливает, допустимо ли значение Null в поле. Если свойство AllowZeroLength установлено как Yes, пустая строка будет допустимой в качестве входных данных независимо от установок свойства Required. Табл 20.2 показывает результаты различных комбинаций установок свойств AllowZeroLength и Required.

Таблица 20.2. Комбинации свойств AllowZeroLength и Required

Обязательно	Допускает нулевую длину	Действие пользователя	Содержащееся значение
Нет	Нет	Нажат Enter Нажат пробел Введена пустая строка	Null Null не допустимо
Нет	Да	Нажат Enter Нажат пробел Введена пустая строка	Null Null Пустая строка
Да	Нет	Нажат Enter Нажат пробел Введена пустая строка	Недопустимо Недопустимо Недопустимо
Да	Да	Нажат Enter Нажат пробел Введена пустая строка	Недопустимо Пустая строка Пустая строка

Вы можете использовать свойство события OnLostFocus, чтобы различать представление значений Null и пустой строки. Используя это свойство, можно получить "Unknown", когда введена пустая строка, с помощью следующей процедуры:

```
Private Sub FieldName_LostFocus()
If Me.FieldName = "" Then
  Me.FieldName = "Unknown"
End If
End Sub
```

Конечно, эта процедура подразумевает, что свойство AllowZeroLength установлено на Yes. Ничего не произойдет, если вы введете значение Null в поле FieldName, пока свойство Required установлено как No. Когда вы вводите группу записей, процедура запроса может легко различить значения Null и пустую строку, была ли эта процедура применена или нет.

Почему это важно? Представьте, что человек вводит список адресов сотрудников. Вы хотите, чтобы у него был выбор в поле Fax, так как сотрудник может иметь факс, а может и не иметь. По этой причине вы устанавливаете свойство AllowZeroLength как Yes и свойство Required как No. Пользователь может пропустить эту ячейку поля, чтобы показать, что сотрудник не имеет факса. Иначе пользователь может ввести пустую строку, чтобы показать, что сотрудник имеет факс, но его номер неизвестен на момент ввода данных. Таким образом, пользователь может заполнить неизвестные номера позднее.

Когда он дойдет до поля Social Security Number, вы хотите убедиться, что значение введено. Чтобы сделать это, вы устанавливаете оба свойства AllowZeroLength и Required как Yes.

Пользователь не сможет покинуть эту ячейку, но, если номер Social Security неизвестен, можно ввести пустую строку, так что его можно будет ввести позже, когда он будет известен. Используя любой из этих методов, вы можете не волноваться о напечатанных в отчете сообщениях "unknown" или "none", но вы все еще можете определить, какое значение введено.

Nz против IsNull

Функция IsNull возвращает значение Boolean. Это или True, или False, в зависимости от того, содержит ли выражение-аргумент правильные данные или нет (Null). Синтаксис IsNull прост:

```
IsNull(expression)
```

Если вы хотите быстро проверить на значение Null, функция Nz - это то, что вам нужно. Как упоминалось ранее, функция Nz имеет два аргумента. Следующий код показывает правильный синтаксис для Nz:

```
Nz(Value, ValuelfNull)
```

Если вы используете функцию Nz в выражении в запросе без использования аргумента ValuelfNull, результатом будет пустая строка в поле или переменная, содержащая значение Null. С аргументом ValuelfNull функция возвращает этот аргумент, если только аргумент Value имеет значение Null. Следовательно, функция Nz может работать как конструкция If - Then - Else. В отличие от нее функция IsNull только возвращает True, если выражение-аргумент - Null, что означает, что вы должны использовать ее в условных выражениях, а не отдельно. Здесь пример IsNull для сравнения:

```
varResult = IIf(IsNull(varShipping), "No Shipping Charge", varShipping)
```

Если значение переменной varShipping - Null, возвращается "No Shipping Charge". Иначе возвращается значение varShipping. Следующий пример показывает более короткий способ использования Nz:

```
varResult = Nz(varShipping, "No Shipping Charge")
```

Этот пример выполняет в точности ту же операцию, что и предыдущий. Если значение varShipping не Null, возвращается varShipping. Если оно Null, возвращается "No Shipping Charge". В действительности вы получаете операцию If • Then Else. Если вы не используете второй аргумент и значением является Null, возвращается пустая строка.

Бывают ситуации, в которых имеет смысл использовать IsNull. Проверьте следующую процедуру:

```
Function MayAddCR(Optional strChkString) As String
'Вставляет два жестких конца строки, если строка непустая

If IsNull(strChkString) Then
  MayAddCR = ""
Else
  MayAddCR = strChkString & vbCrLf & vbCrLf
End If

End Function
```

Если вы попытаетесь использовать Nz, чтобы выполнить ту же самую операцию в этом примере, то столкнетесь с препятствием. Так как аргумент Value является как входным, так и выходным значением, какое бы входное значение ни было, оно тоже возвращается, когда значение не Null. Следующий пример показывает, почему это не сработает:

```
MayAddCR = Nz(strChkString & vbCrLf & vbCrLf, "")
```

Если вы пытаетесь проверить аргумент Value, вы уже потерпели неудачу, так как вы проверяете не Null-значение, когда связываете собственную константу vbCrLf. Даже если вы обнаружите способ сделать это с помощью Nz, велика вероятность, что применение IsNull будет все же более понятным и легким для чтения и использования.

При использовании функции IsNull необходим только один аргумент. Всегда употребляйте функцию IsNull для определения того, содержит ли выражение значение Null (в отличие от одиночного значения). Выражения, которые, как вы думаете, преобразуются в True, такие, как If Var = Null и If Var <> Null, всегда являются False. Это потому, что любое выражение, содержащее Null, само по себе Null, а потому является False.

IsNull в действии

Представьте, что вы хотите подтверждать сохранение в созданной вами базе данных нажатием кнопки на записи, которую хотите сохранить. Когда кнопка нажата, в поле ResNotes (отметки сохранений) вводится "Confirmed". Хорошо, если поле Null. Но что будет, если поле не Null? Если в поле ResNotes уже есть

информация, "Confirmed" может легко затеряться в тексте, если вы присоединяете ее в конце того, что уже там, и заместит текст в противном случае. Вы хотите, чтобы она выделялась помещением в отдельную строку. Это производится вводом двух жестких знаков конца строки до прикрепления "Confirmed" к полю.

Следующий пример показывает, как создавать программу, прикрепленную к кнопке Confirm, которая уже находится на форме. Перед созданием кнопки проверьте функцию, которая производит указанные действия, выполнив следующие шаги:

1. Откройте базу данных Convention и затем откройте форму Reservation в конструкторе.
2. Щелкните мышью на кнопке Confirm и затем F4, чтобы открыть список свойств.
3. Раскройте ниспадающий список напротив события On Click (Нажатие кнопки) и выберите [Процедура обработки события] ([Event Procedure]). Нажмите кнопку Создать (Build).
4. В окне Проект (Project) дважды щелкните на модуле ChkNull, как показано на рис. 20.4.

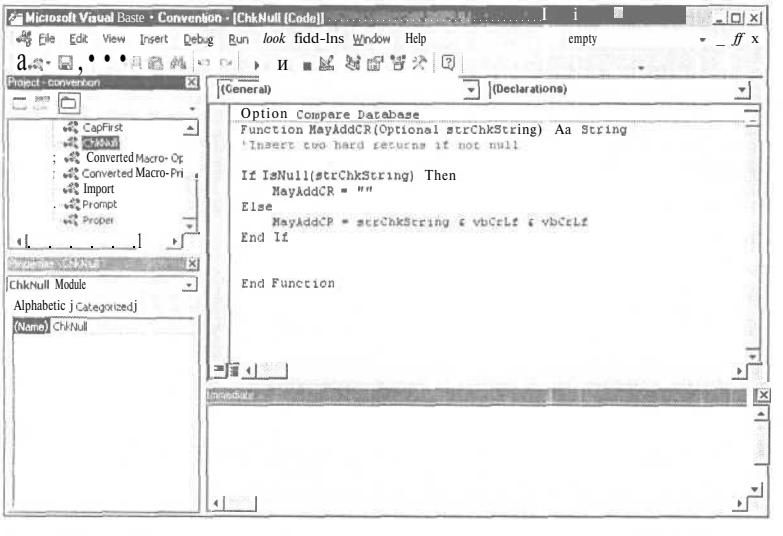


Рис. 20.4. Дважды щелкните на стандартном модуле ChkNull, чтобы проверить его

5. Нажмите Ctrl+G, чтобы открыть окно отладки (immediate). Наберите в нем ? MayAddCR(Null) и нажмите Enter. Вы ничего не видите. Выделите все в окне отладки и удалите.
6. На этот раз наберите ? MayAddCR("Confirmed") и нажмите Enter, чтобы увидеть результат, показанный на рис. 20.5.
7. В окне проекта дважды щелкните по Form_Reservation, чтобы открыть модуль, прикрепленный к кнопке Confirm.

Рис. 20.5. Вы можете проверить функцию, используя пример, который очень похож на реальный

8. Поместите следующую строку между Sub и End Sub в модуле:

```
Me.ResNotes = MayAddCR(Me.ResNotes) & "Confirmed"
```

9. Закройте модуль и список свойств. Нажмите кнопку Вид (View), чтобы запустить форму.
10. Нажмите кнопку Confirm в первой записи. Заметьте что "Confirmed" добавлена без предшествующего ей возврата каретки.
11. Нажмите кнопку Next Record, чтобы перейти к ReservationID 2.
12. Нажмите кнопку Confirm снова. Заметьте что "Confirmed" добавлена на отдельную строку, отделенную от остального текста двумя жесткими символами конца строки, как показано на рис. 20.6.

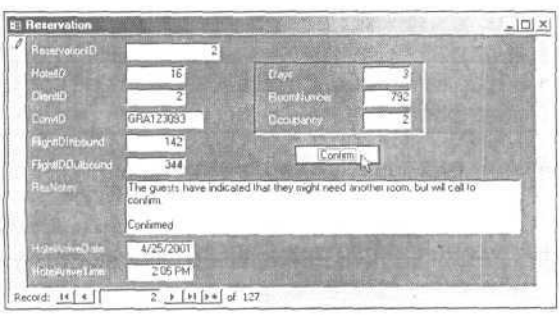


Рис. 20.6. Используя функцию MayAddCR, вы можете добавлять возврат каретки перед "Confirmed", если текст в поле уже существует

13. Закройте форму и сохраните изменения.

«Завоевание» дат

Еще один барьер в Access - это то, что многие программисты спотыкаются при расчетах с датами. Почему работа с датами должна быть важной для вас как программиста Access? А что, если вы захотите узнать число дней между двумя датами или посчитать дату через шесть месяцев? Если вы пишете бухгалтерское приложение, вам может понадобиться узнать, сколько времени прошло с момента, когда некоторый продукт был заказан. Если вы работаете со страхованием, вам может понадобиться узнать, сколько времени прошло после сообщения о претензии. В торговле вам может понадобиться отследить число дней, которое прошло с момента, когда вы последний раз звонили покупателю. Работа с датами - это неотъемлемая часть большинства приложений.

Вы можете начать с простых собственных функций. Табл. 20.3 показывает стандартные функции Access для работы с датами.

Таблица 20.3. Стандартные функции Access для работы с датами

Функция	Возвращаемое значение	Пример
Date()	Текущая дата	7/1/1999
Now()	Текущая дата и время	10/25/2001 10:01:39 p.m.
Time()	Текущее время	08:02:45 p.m.

Функции форматирования дат

Представьте, что вы хотите задать формат функции способом, отличным от показанных в табл. 20.3. Вы можете внедрить (вставить) функцию date внутрь функции format, чтобы достичь желаемого результата. Если сегодня 10/25/2001, следующая функция возвратит 10:

```
Format(date(),"mm")
```

Следующая возвратит 25:

```
Format(date(),"dd")
```

Если вы хотите просмотреть обычные форматы, обратитесь к разделу «Тестирование и применение пользовательских форматов» в гл. 3.

Если вы хотите получить просто полный год, введите следующее:

```
Format(date(),"yyyy")
```

Тем не менее, если вы хотите полную укороченную дату, введите следующее:

```
Format(date(),"mm/dd/yyyy")
```

Следующее:

```
Format(date(),"mmmm dd, yyyy")
```

возвращает

```
October 25, 2001
```

для полной длинной даты; следующее:

```
format(date(),"long date")
```

возвращает

```
Thursday, October 25, 2001
```

для дня недели плюс длинная дата.

Как Access хранит даты

Microsoft Access хранит тип данных Date/Time (Дата/время) как double - точно, число с плавающей запятой размером до 15 десятичных разрядов. Целочисленный компонент (до десятичного) типа double - точное число представляет дату; десятичный компонент представляет время.

Верными данными являются значения из диапазона от -647434 (1 января, 100 от сего числа) до 2958465 (31 декабря, 9999 от сего числа). Верными данными времени являются значения из диапазона от 0.0 (00:00:00) до 0.99999 (23:59:59). Десятичное значение представляет часть одного дня. Вы можете преобразовать десятичное значение в часы, минуты и секунды, умножив его на 24.

Если вы хотите для интереса посмотреть, как хранятся даты, используйте функцию CDbl в окне прямого ввода, чтобы преобразовать дату в число двойной точности. Табл. 20.4 иллюстрирует это.

Таблица 20.4. Преобразование даты в число

Данные в окне прямого ввода	Возвращаемое значение
`? CDbl(#10/15/2001 14:00#)`	37179.5833333333
`? CDbl(#12/15/1889 17:32#)`	-3667.73055555556

Если вы хотите преобразовать, наоборот, из числа двойной точности в «Дату/время», используйте функцию CVDate, как показано в табл. 20.5.

Таблица 20.5. Преобразование числа в дату

Данные в окне прямого ввода	Возвращаемое значение
`? CVDate(1.5000)`	12/31/1899 12:00:00 PM
`? CVDate(35000.7812)`	10/28/1995 6:44:56 PM

Если вы умножаете число лет (100) XX века на число дней в году (примерно 365.25), результат будет 36525. Если вы преобразуете это число в дату, то получите 31 декабря 1999 года, что является последним днем XX века. Это не так просто, так как в году на самом деле 362.2422 дня, но идея понятна. Microsoft, очевидно, урегулировала это несущественное расхождение, делая отсчет от последнего дня 1899 года вместо первого дня 1900-го. Табл. 20.6 показывает, как. преобразуются компоненты чисел.

Таблица 20.6. Number-to-Date (Число в дату) преобразование компонентов

Хранимое число	Компонент даты	Представленная дата	Компонент времени	Представленное время
2.50	2	1/1/1900	0.50	12:00:00 PM
36526.75	36526	1/1/2000	0.75	6:00:00 PM

Вычисление промежутков времени

Несмотря на то что вы преобразуете даты в числа и используете их для вычислений, трудно получить согласующиеся результаты с десятичными дробями с плавающей точкой. Например, если сегодняшняя дата 25 октября 2001 года и вы набираете ? `Now()=DateValue("10/25/2001")` в окне прямого ввода, возвращается результат False. Причина этого в том, что Now возвращает число с двойной точностью, тогда как DateValue возвращает целое значение, представляющее только дату. Если это несогласование не является достаточной причиной, то, например, не всегда удобно справляться с такими потенциальными проблемами, как, например, високосный год. Лучше для вас будет использовать для выполнения этой работы внутреннюю функцию Access.

Функция DateAdd

Используйте функцию DateAdd, чтобы добавлять временные промежутки, как, например, часы, дни или годы, в значения Date/Time (Дата/время). Следующий синтаксис для функции DateAdd показывает все три необходимых аргумента:

`DateAdd(interval, number, date)`

Табл. 20.7 показывает детализацию аргументов для функции DateAdd.

Таблица 20.7. Аргументы функции DateAdd

Аргумент	Описание
Interval	Обязательный. Строковое выражение, которое является интервалом времени, который вы хотите добавить
Number	Обязательный. Числовое выражение, которое является числом промежутков, которые вы хотите добавить. Оно может быть положительным (чтобы получить дату в будущем) или отрицательным (чтобы получить дату в прошлом)
Date	Обязательный. Произвольное выражение (Дата) или буквенное, представляющее дату, в которую добавляется промежуток

Табл. 20.8 показывает варианты аргумента Interval и что они представляют.

Таблица 20.8. Варианты для аргумента Interval

Значение аргумента	Описание
Yyyy	Год
q	Квартал
m	Месяц
y	День года
d	День
w	День недели
ww	Неделя
h	Час
n	Минута
s	Секунда

ЗАМЕЧАНИЕ. При работе с неделями w представляет рабочие дни минус суббота и воскресенье, в отличие от этого ww представляет недели.

В следующем листинге показана процедура, которая использует функцию DateAdd, чтобы добавить 20 лет в дату:

```
Sub TestDate()
Dim dBeforeIntv As Date
Dim dAfterIntv As Date

dBeforeIntv = #1/1/1980#
dAfterIntv = DateAdd("yyyy", 20, dBeforeIntv)
Debug.Print dAfterIntv
End Sub
```

Функция возвращает 1/1/2000 в окне прямого ввода. Так как аргумент интервала (промежутка) был установлен как year, функция DateAdd добавила 20 лет в 1/1/1980. Переменная dBeforeIntv - это дата, в которую был добавлен интервал в 20 лет. Чтобы добавить 20 дней в ту же дату, просто поменяйте аргумент интервала "yyyy" на "d" для дней, как показано в следующем листинге:

```
Sub TestDate()
Dim dBeforeIntv As Date
Dim dAfterIntv As Date

dBeforeIntv = #1/1/1980#
dAfterIntv = DateAdd("d", 20, dBeforeIntv)
Debug.Print dAfterIntv

End Sub
```

Измененная функция теперь возвратит 1/21/1980, так как она добавляет 20 дней вместо 20 лет. Следующая процедура добавляет временное значение в дату перед изменением интервала, как показано в следующем листинге:

```
Sub TestDate()
Dim dBeforeIntv As Date
Dim dAfterIntv As Date

dBeforeIntv = #1/1/1980 6:00:00 PM#
dAfterIntv = DateAdd("n", 20, dBeforeIntv)
Debug.Print dAfterIntv

End Sub
```

Заметьте, что переменная dBeforeIntv преобразуется с временным значением (6:00 p.m.), добавленным к дате. Также заметьте, что интервал был заменен на "n" для минут. Эта функция возвращает 1/1/1980 6:20:00 p.m. в окне прямого ввода.

Функция DateDiff

То, что вы можете складывать интервалы с датами, хорошо, но что если вы захотите посчитать промежуток времени между двумя датами? Функция DateDiff - это то, что надо в этом случае использовать. Только первые три из пяти ее аргументов, показанных в следующем синтаксисе DateDiff, необходимы. Необязательные аргументы показаны в квадратных скобках.

```
DateDiff(interval, date1, date2, [firstdayofweek], [firstweekofyear])
```

Табл. 20.9 показывает детализацию аргументов для функции DateDiff.

Таблица 20.9. Аргументы функции DateDiff

Аргумент	Описание
Interval	Обязательный. Строковое выражение, являющееся промежутком времени, который вы используете для вычисления разницы между date1 и date2
date1, date2	Обязательный. Произвольное выражение (Дата). Две даты, которые вы хотите использовать в вычислениях. Если date1 ссылается на более поздний момент времени, чем date2, возвращается отрицательное число
Firstdayofweek	Необязательный. Константа, которая устанавливает первый день недели. Если не установлено, по умолчанию берется воскресенье
firstweekofyear	Необязательный. Константа, которая устанавливает первую неделю года. Если не установлено, первой неделей по умолчанию берется неделя, которая содержит 1 января

Несмотря на то что аргумент Interval тот же самый для функции DateDiff, что и для функции DateAdd, он здесь повторен для удобства. Следующая процедура показывает две переменные-даты, как до этого, но в этот раз обе переменные используются в функции и присваиваются третьей переменной:

```
Sub TestDate()
Dim dFirstDate As Date
Dim dSecondDate As Date
Dim lDiff As Long

dFirstDate = #1/1/1980#
dSecondDate = #8/25/1980#
lDiff = DateDiff("d", dFirstDate, dSecondDate)
Debug.Print lDiff

End Sub
```

Эта функция возвращает 237 дней, которые являются разницей в днях между двумя датами.

Что дальше?

В этой главе вы изучили, как справляться с общими препятствиями, на которых спотыкается программист. В следующей главе вы научитесь решать проблемы, связанные с формами, но это не все, что вы изучите. По ходу вы узнаете о том, как улучшить ваши формы, чтобы они выглядели профессиональными, были удобными для пользователя и эффективными.

21
Осуществление контроля над формами

В гл. 20 вы изучили, как справляться с большими проблемами в малых задачах. Эта глава работает с более значимыми вещами, показывая, почему формы являются таким мощным инструментом.

К этому моменту вы уже достаточно знаете об основах форм. Вы можете выполнять программно те операции, которые вам приходилось делать вручную до этого. Вы можете изменять свойства форм, искать значения, импортировать и экспортировать объекты и выполнять общие операции, которые вы рассчитываете осуществлять с помощью форм. Но надо еще многое изучить о многофункциональных объектах форм. В действительности вы, возможно, подумаете, что лучшее было усвоено до этого момента (ну почти до этого). В этой главе вы изучите следующее:

- как параметры запроса используются с формами;
- все о всплывающих формах;
- как использовать PivotTables;
- как управлять редактированием, сохранением и блокированием с помощью форм;
- как преодолевать трудности с большими таблицами, используя формы;
- как применять табуляторные элементы управления (элемент управления, используемый для постраничного представления многостраничной информации пользователю).

Использование параметризованных запросов как источников записей форм

Запрос, который предлагает пользователю ввести информацию, называется *запросом с параметрами* (параметризованный запрос). Информация, которая вводится, становится в запросе критерием. Вы вводите приглашение (пользователя к действиям), заключенное в кавычки (как если бы это было поле), в поле или поля, в которые вы хотите добавить критерий. После того как запрос готов, вы можете прикрепить его к форме как источник записей. Представим, что ваш коллега хочет добавить критерий в запрос так, чтобы пользователь мог открывать форму, отображающую требование для любого выбранного состояния. Затем вы бы могли добавить другой параметр, чтобы запрос мог выбирать диапазон дат. После выбора параметров пользователь может просматривать записи выбранных данных. Все это можно сделать, приложив небольшие усилия.

Проделайте следующие шаги, чтобы создать запрос и затем прикрепить его к форме. В конце просмотрите форму, чтобы сверить данные.

1. Откройте базу данных Claims из папки AccessByExample.

2. Откройте новый запрос, основанный на таблице Losses.
3. Дважды щелкните по строке заголовка и затем перетащите все поля до сетки запроса.
4. В строке условия отбора (criteria) для поля AccidentState наберите `[Enter a State]`, как показано на рис. 21.1.

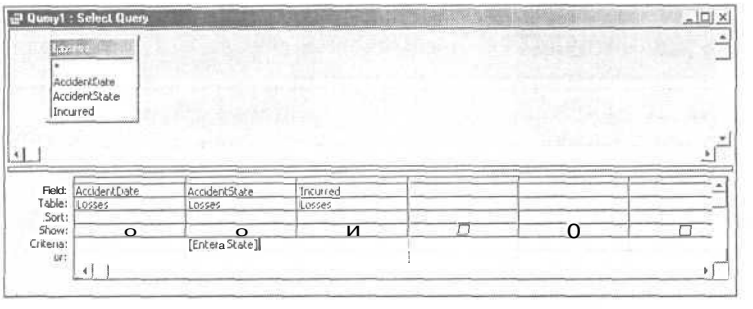


Рис. 21.1. Когда вы вводите параметр в сетку запроса, вы можете обращаться к пользователю во врем 1 рабочего цикла

5. Нажмите кнопку Запуск (Run) для проверки обращения и просмотрите результаты запроса.
6. Когда предлагается ввести штат, наберите FL для Florida. Запрос найдет записи со штатом Florida.
7. Нажмите кнопку Вид (View), чтобы переключиться в конструктор. В строке условия отбора (criteria) для поля `AccidentDate` наберите `Between [Date1] And [Date2]`, как показано на рис. 21.2.

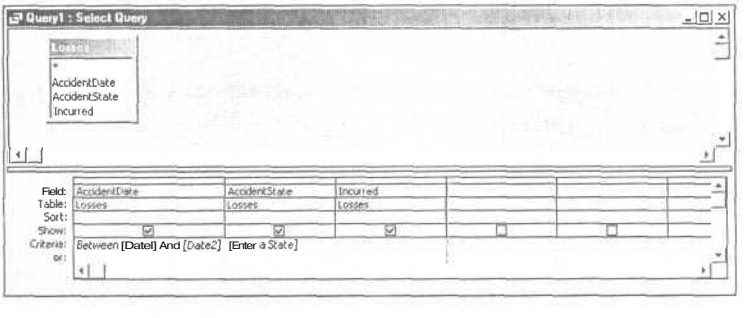


Рис. 21.2. Вы можете вставлять составные параметры в сетку запроса, чтобы обращаться к пользователю

8. Запустите запрос, чтобы проверить обращение. Когда вас спросят первую дату, наберите 1/1/97. Затем при вводе второй даты наберите 1/1/98. Когда запрашивается штат, снова наберите FL. Запрос найдет шесть записей, как показано на рис. 21.3.

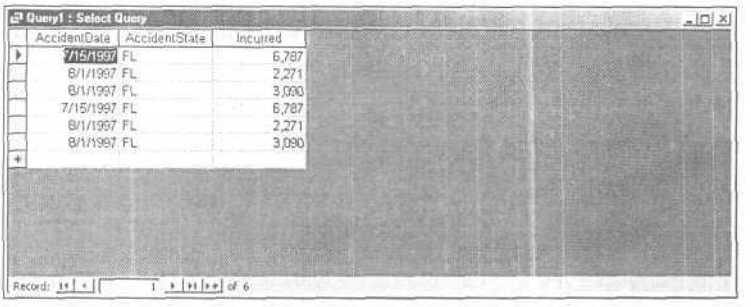


Рис. 21.3. Запрос с параметрами просто возвращает записи, которые соответствуют условию отбора, введенному с использованием параметров

9. Нажмите кнопку **Вид** (View), чтобы вернуться в конструктор, затем закройте и сохраните запрос как qryParameter.
10. Откройте форму LossesParameter в конструкторе для того, чтобы изменить его свойства.
11. Щелкните на селекторе формы - черном квадратике в левом верхнем углу (form selector) и нажмите F4, чтобы открыть список свойств формы.
12. Переключитесь на вкладку **Событие** (Event), затем раскройте ниспадающий список свойства «Источник записей» (RecordSource) и выберите qryParameter.
13. Закройте список свойств. Нажмите кнопку **Вид** (View), чтобы переключиться в режим формы.
14. Введите 1/1/96 и 1/1/99 в указанном порядке для двух запросов дат. Наберите FL, когда предлагается ввести штат.
15. Просмотрите несколько записей, закройте и сохраните форму. Оставьте базу данных открытой.

Конечно, параметры также можно использовать и для отчетов. Сравните технику параметров со следующей техникой.

Всплывающиеформыпродвинулисьдальшеоконсообщений

Запросы с параметрами работают как источник записей как для отчетов, так и для форм. Тем не менее бывают случаи, когда ваши нужды выходят за пределы возможностей параметров и окон сообщений. Конечно, вы можете вводить значения, используя параметры и окна ввода, но что если вам нужно 10 элементов в одном поле для отчета?

Чем всплывающая форма отличается от регулярной формы Access? Всплывающие формы остаются поверх всех других форм и объектов, активны они или нет. Другое отличие состоит в том, что кнопки передвижения записей (навигации между объектами в базах данных) не так важны на всплывающих формах,

так как они (формы) часто работают как меню и диалоговые окна, которые запрашивают информацию у пользователя.

Два типа всплывающих форм - это modeless (немодальные) и modal (модальные). Отличие между ними состоит в том, что немодальные формы дают возможность доступа к другим объектам, тогда как модальные не дают такой возможности. Это означает, что вы должны среагировать на модальную всплывающую форму перед тем, как Access освободит ее, чтобы вы могли продолжить.

Вы создаете немодальную всплывающую форму, устанавливая свойство PopUp как Yes и свойство Modal как No. Для создания модальной всплывающей формы установите свойства PopUp и Modal как Yes. Если вы открываете модальную всплывающую форму, используя метод OpenForm, вы должны определить внутреннюю константу acDialog в аргументе WindowMode.

Всплывающие формы дают вам функции для простого поиска составных значений, которые могут быть использованы как критерий. Они могут быть достаточно малы, чтобы не попасться на пути, но и достаточно мощны, чтобы обеспечить вам немалую гибкость. Вы можете использовать всплывающие формы как календари, калькуляторы, диалоговые окна, списки для выбора, окна поиска и т. д. Например, если вы хотите разработать справочную систему, но считаете окно сообщения слишком ограничивающим, всплывающие формы - это наиболее вероятный вариант для вас.

Загрузка всплывающих форм

Перед тем как говорить о механизме создания всплывающих форм, давайте обратимся к тому, как их загружать. Если ваша форма была названа PopUp, то вы просто вставляете такую строку в ваш код:

```
DoCmd.OpenForm "PopUp"
```

Пока она просто как любая другая форма. Вы начинаете со значений по умолчанию. Как показывалось в предыдущем разделе, если вы хотите приостановить вашу программу, ожидая закрытия формы или ее скрытия, вы должны использовать аргумент acDialog метода OpenForm. Это предотвращает открытие окна сообщения или чего-то еще, пока закрыто окно. При использовании этого подхода код будет выглядеть так:

```
DoCmd.OpenForm "PopUp". . . . acDialog
```

ПРЕДУПРЕЖДЕНИЕ. Если вы не используете константу acDialog для аргумента WindowMode метода OpenForm, то для acWindowNormal применяется значение по умолчанию, которое означает, что, даже если у вас свойство Modal установлено как Yes, оно аннулируется значением по умолчанию. Другими словами, вы думаете, что получаете Modal, но на самом деле вы получаете нормальные установки Windows. Это нормально, только если вам не требуется Modal, так что намотайте это себе на ус.

Если вы хотите передать строку (или даже составные параметры, как в нашем случае) через аргумент OpenArgs метода OpenForm всплывающей или любой дру-

гой форме, вы это можете легко сделать. Например, если вы хотите передать имя штата, по которому вы фильтруете форму с именем LossesArg, используйте аргумент OpenArgs таким образом:

```
DoCmd.OpenForm "LossesArg". . . . ., strArg
```

Заметьте, что здесь шесть запятых вместо пяти. Переменная strArg используется для аргумента OpenArgs. Воспринимайте этот аргумент как «лидера», который передает строку. Получатель - это свойство OpenArgs получающей формы. Следующая процедура, которую можно найти в форме LossesArg, демонстрирует, как вы можете прикрепить код к событию формы ОпОреп, чтобы получать переданную переменную:

```
Dim strArg As String

strArg = Nz(Me.OpenArgs)
Me.Filter = "[AccidentState]=" & Chr$(34) & strArg & Chr$(34)
Me.FilterOn = True
```

Процедура во всплывающей форме PassArg, которая вызывает форму LossesArg и передает строку, выглядит так:

```
Dim strArg As String
strArg = "CO"
DoCmd.OpenForm "LossesArg",,,,,, strArg
DoCmd.Close acForm, "PassArg"
```

Имя переменной не должно быть одинаковым в обеих процедурах для формы, которая назначена для получения переданной переменной. Заметьте, что форма PassArg закрывает сама себя.

Проделайте следующие шаги, чтобы посмотреть на эту технику в действии.

1. Откройте форму PassArg.
2. Нажмите кнопку «Open Form» (Открыть форму), чтобы открыть и отфильтровать форму LossesArg. Заметьте, что отфильтрованы записи со штатом Colorado, как показано на рис. 21.4. Закройте форму LossesArg и оставайтесь в базе данных.

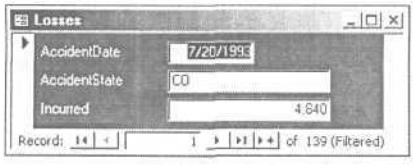


Рис. 21.4. Аргумент OpenArgs используется для фильтрации в форме LossesArg

Создание всплывающих форм

Когда вы создаете всплывающую форму, помните, что вам не нужны полосы прокрутки, выборщики записей или кнопки навигации. Вам просто нужно окно со строкой заголовка, так, чтобы вы его могли перемещать. Вам даже не обязательно иметь всплывающую форму, связанную с таблицей или запросом.

Если вы поменяете только первые три свойства в табл. 21.1 на предложенные настройки, вы закончите с внешним видом формы. Тем не менее установка свойства PopUp на Yes гарантирует вам, что форма всегда будет поверх всех окон. Установка свойства Modal на Yes приводит к приостановке действия формы, пока она не будет закрыта. Табл. 21.1 иллюстрирует типичные установки для всплывающих форм. Эти установки предназначены просто для ориентира. Они не только оказывают влияние на всплывающие формы, но и очень важны. Вы можете поэкспериментировать, пока не найдете подходящую для вас настройку.

Таблица 21.1. Типичные настройки всплывающих форм

Свойство	Значение	Описание
Scroll Bars	No	Наличие у формы полосы прокрутки
Record Selectors	No	Наличие у формы выборщика записей
Navigation Buttons	Neither	Наличие у формы кнопок навигации
Pop Up	Yes	Открытие формы как всплывающей, что значит, что форма всегда будет поверх всех окон
Modal	Yes	Открытие формы как немодального объекта (вы можете переключаться в другие окна) или модального (форма остается активной до тех пор, пока не будет закрыта)
Border Style	Dialog	Тип границы и граничных элементов (строка заголовка, кнопка «Закрыть», меню управления, кнопки «Развернуть» и «Свернуть»), используемых в форме. Также определяет, будет ли форма масштабируемой
Control Box	Yes	Наличие у формы меню управления в режиме формы

Свойство	Значение	Описание
Min Max Buttons	Both Enabled	Наличие у формы кнопок «Развернуть» и «Свернуть» в режиме формы. Если вы устанавливаете свойство BorderStyle как Dialog, Microsoft Access автоматически удаляет кнопки «Развернуть» и «Свернуть»
Close Button	Yes	Отсутствие кнопки «Закрыть»
AutoCenter	Yes	Автоматическое центрирование формы при открытии в окне приложения.

Следующие шаги помогут вам в создании всплывающей формы.

1. В разделе **Формы** (Forms) нажмите «Создание формы в режиме конструктора» (Create Form in Design View), чтобы открыть пустую форму в конструкторе.
2. Щелкните на селекторе формы и затем F4, чтобы открыть список свойств формы.
3. Переключитесь на вкладку **Все** (АН) и приведите 10 настроек из таблицы в соответствие с указанными. Установите свойство «Ширина» (Width) равным 2.5. Вы также можете перетащить правую сторону формы на 2.5 дюйма по масштабной линейке, чтобы установить это свойство.
4. Закройте форму, сохраните ее как Form1 и снова откройте, как показано на рис. 21.5. Обратите внимание на появившиеся настройки. Удалите Form1. Оставайтесь в базе данных.

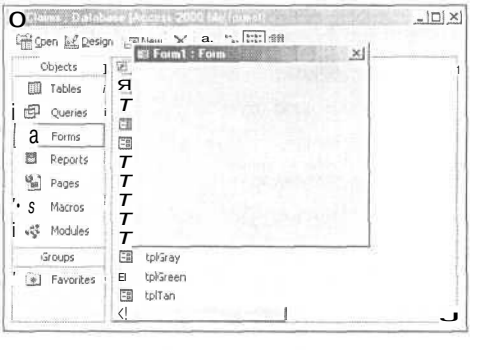


Рис. 21.5. Всплывающие формы обычно не имеют полос прокрутки или кнопок навигации

Пустая всплывающая форма - это не слишком захватывающе для вас, пока вы с ней что-нибудь не сделаете. Используйте воображение. Вы можете собирать информацию у пользователей, применяя всплывающие формы в качестве диалоговых окон. Всплывающая форма может работать как календарь или «администратор» встреч. Вы можете установить критерий отчета, используя всплывающую форму. В следующих подразделах будут рассмотрены некоторые примеры этих применений.

Использование всплывающих форм в качестве диалоговых окон

Одно из наиболее интересных применений всплывающих форм - это диалоговые окна. Если вы заглянете в меню **Сервис** (Tools) - **Параметры** (Options) в Access, вы увидите пример диалоговых окон. Например, если вы переключитесь на вкладку **Вид** (View), вы увидите кнопки-флажки и переключатели.

Возможно, вы удивитесь, почему нет способа использовать шаблон для формы. Вместо постоянного изменения множественных настроек вы сохраняете шаблон со всеми нужными вам настройками. Затем вы можете просто указать шаблон при создании новой формы - и дело сделано.

Если вы сохраняете форму под именем «Обычная» (Normal), она автоматически станет служить шаблоном для любой другой формы, которая будет создана далее. Если хотите, вы также можете изменить шаблон формы в меню **Сервис** (Tools), **Параметры** (Options), **Формы и отчеты** (Forms/Reports) с «Обычного» (Normal) на имя формы, которую вы создали. Но это не очень удобно. Возможно, вам понадобится несколько разных шаблонов.

В базе данных Claims есть форма с именем CreatePopup. Она применяет шаблон для некоторых настроек, которые будут использованы в создаваемой форме, и дает вам возможность определять другие настройки «на лету». За секунды вы получаете пустую всплывающую форму с нужными вам настройками.

Эта всплывающая форма не имеет источника записей. Поэтому ни один из контроллеров на ней не связан. Эта форма демонстрирует некоторые общие контроллеры, используемые для всплывающих форм с функцией диалогового окна, такие, как группы кнопок выбора (опций) и флаговые кнопки. Она является поучительной в некоторых ракурсах.

Во-первых, она показывает, как программировать флаговые кнопки. Похоже, что они - это часть группы кнопок выбора, но в действительности они только внешне похожи на нее, так как являются отдельными контроллерами. Во-вторых, она показывает, как создавать форму программно, используя шаблоны. В-третьих, она показывает, как работать с твипами.

ЗАМЕЧАНИЕ. Твип (1/20 точки) - это мера, используемая при определении объектов, которые надо вывести на экране компьютера или принтера. Он равен 1/1440 дюйма, или 1/567 см. Он равен 1/20 точки, которая является стандартной мерой в печати. Точка - это, примерно, 1/72 дюйма. Скажем, вы хотите, чтобы объект в Access имел размер 7 дюймов. Вы умножаете 1440 на 7, чтобы получить полное число твипов.

Проделайте следующие шаги, чтобы проверить форму.

1. Откройте форму CreatePopUp в конструкторе, как показано на рис. 21.6.

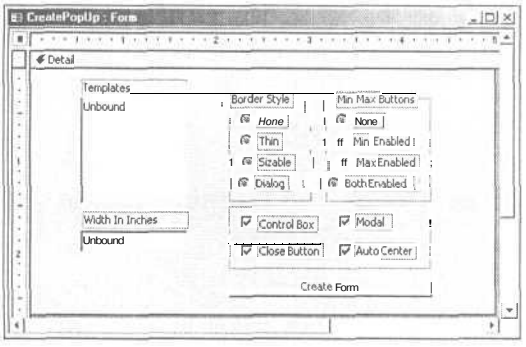


Рис. 21.6. Форма CreatePopUp (показанная в окне конструктора) использует такие элементы управления, как кнопки-флажки и группы переключателей, для настройки вида диалогового окна

2. Щелкните правой кнопкой мыши на кнопке Create Form и нажмите **Создать** напротив события, чтобы открыть процедуру, прикрепленную к событию Нажатие кнопки (OnClick). Появится следующий код. Оставайтесь в модуле, пока не закончите чтение комментария, следующего за процедурой. Затем закройте модуль, список свойств и форму. Выберите **Нет,** если Access предложит вам сохранить изменения.

```
Private Sub cmdCreateForm_Click()
Dim iBorder As Integer, iMinMax As Integer
Dim frm As Form, sglWidth As Single
Dim iTplIndex As Integer
Dim sTplName As String

Const TPI = 1440 'TPI = Twips Per Inch, твип на дюйм
iTplIndex = Me.lstTemplates.ListIndex 'получаем индекс выборки
sTplName = Me.lstTemplates.ItemData(iTplIndex)
iBorder = Me.optBorder 'Устанавливаем стиль обрамления (BorderStyle)
'Предыдущая строка, устанавливающая стиль обрамления, может быть переписана так:
'Select Case Me.optBorder ' Устанавливаем стиль обрамления
(BorderStyle)
    'Case 0
    '   iBorder = 0
    'Case 1
    '   iBorder = 1
    'Case 2
    '   iBorder = 2
    'Case 3
    '   iBorder = 3
'End Select

iMinMax = Me.optMinMax 'Устанавливаем MinMaxButtons
```

```
    sglWidth = IIf(IsNull(Me.txtWidth), 3, Me.txtWidth)
    Set frm = CreateForm(, sTplName)
    With frm 'Настройки всплывающей (PopUp) формы устанавливаются как для
    диалогового окна
       .BorderStyle = iBorder
       .MinMaxButtons = iMinMax
       .Width = sglWidth * TPI
       .AutoCenter = Me.chkCenter
       .Modal = Me.chkModal
       .CloseButton = Me.ChkClose
       .ControlBox = Me.chkControl
       .PopUp = True
    End With

    DoCmd.Close acForm, "CreatePopUp"
    DoCmd.Restore
    End Sub
```

Конструкция Select-Case делает так, что текст появляется зеленым при проверке в форме. Она показана только для того, чтобы продемонстрировать общий метод управления потоками данных процедуры посредством программного реагирования на кнопку, которая была выбрана в Option group (группа кнопок выбора). В этом отдельном примере здравый смысл говорит вам, что каждый "then" - это то же, что и каждый "if" или case. Потому что значения, представляющие каждую опцию настройки свойства, с отсчетом от нуля.

Когда вы создаете группу кнопок выбора, используя Option Group Wizard, каждая опция последовательно пронумерована по умолчанию. Например, если у вас четыре опции, значения, присвоенные каждой опции, будут от 1 до 4 соответственно. Тем не менее вы можете изменить значения опций на диапазон от О до 3, чтобы привести их в соответствие со схемой нумерации с отсчетом от нуля. Когда была создана optBorder Option group, она была установлена на последовательную нумерацию начиная с нуля так, что значения Select-Case могли быть синхронизированы. Таким образом, код Select-Case не нужен. Вы можете просто использовать iBorder = Me.optBorder, чтобы контролировать выбор пользователя.

Табл. 21.1 показывает значения Visual Basic, которые взаимодействуют с настройками Border Style.

Таблица 21.2. Установки свойства BorderStyle

Параметр настройки (установка)	Visual Basic
None	0
Thin	1
Sizable	2
Dialog	3

Это просто рационально. Оба свойства, BorderStyle и MinMaxButtons, настроены одинаково. Они оба имеют четыре значения VBA с отсчетом от нуля,

расположенные последовательно и представляющие четыре установки. Таким образом, пока вы вносите настройки в том же порядке, в каком они перечислены на форме, вы можете обойтесь без такой строки:

```
iMinMax = Me.optMinMax 'Конструкция Select..Case не нужна
```

Заметьте, что настройки собраны со всей формы. Так как оба свойства, BorderStyleи MinMaxbuttons, имеют по четыре свойства, они идеально подходят для группы кнопок выбора. Может быть выбрано только одно свойство на группу. Четыре флаговые кнопки используются для свойств, которые имеют значения «истина» или «ложь». Любая или все из этих кнопок могут быть включены или выключены. Все они имеют значение по умолчанию -1, которое VBA интерпретирует как «истина». Значение 0 интерпретируется как «ложь». Если вы отключаете кнопку, ее значение становится 0. Это значение передается таким образом:

```
.CloseButton = Me.ChkClose
```

Если флаговая кнопка chkClose на форме CreatePopUp отключена, свойство CloseButton новой формы устанавливается на False. Если текстовое окно левое пустое, значение по умолчанию 3 выбирается с помощью такой строки:

```
sglWidth = IIf(IsNull(Me.txtWidth), 3, Me.txtWidth)
```

Если вы хотите, чтобы ваша форма имела размер 3.5 дюйма, введите 3.5 в строку, помеченную как Width In Inches. Возможно, наиболее интересная строка во всей программе - эта:

```
Set frm = CreateForm(, sTplName)
```

Эта строка фактически создает форму, используя любой шаблон, который вы предложите, и устанавливает указатель на переменную frm. Вы пропускаете первый аргумент, который является именем базы данных, в методе CreateForm, и добавляете шаблон, на котором хотите построить свою форму, в качестве второго аргумента. Переменная sTplName получается из спискового окна посредством таких строк:

```
iTplIndex = Me.lstTemplates.ListIndex 'Получаем индекс выборки
sTplName = Me.lstTemplates.ItemData(iTplIndex)
```

Так как вам не нужны множественные выборки, то вам не надо использовать настройки Simple и Extended свойства MultiSelect спискового окна. Отсавьте значение по умолчанию None для этого свойства.

Конструкция With используется с переменной frm для установки свойств только что созданной формы таким образом:

```
With frm 'Настройки всплывающей (PopUp) формы устанавливаются как для
диалогового окна
   .BorderStyle = iBorder
   .MinMaxButtons = iMinMax
   .Width = sglWidth * TPI
   .AutoCenter = Me.chkCenter
   .Modal = Me.chkModal
   .CloseButton = Me.ChkClose
```

```
    .ControlBox = Me.chkControl
    .PopUp = True
End With
```

Заметьте, что свойство Width формы установлено то, которое было введено в дюймах в списковом окне Width In Inches, содержащемся в переменной sglWidth, умноженной на число твипов в дюйме (1440). Свойство PopUp установлено на True, потому что как-никак, а это всплывающая форма.

Теперь, когда у вас есть идея, как работает процедура, стоящая за формой, давайте посмотрим на нее в действии. Проделайте для этого следующие шаги:

1. Дважды щелкните на форме CreatePopUp, чтобы открыть ее в режиме формы.
2. Из списка Templates (Шаблоны) выберите tplGreen. Три шаблона, перечисленные в окне Templates, имеют свойство ScrollBars, установленное на Neither, и свойства RecordSelectors и NavigationButtons, установленные на No. Новая форма будет основываться на шаблоне tplGreen.
3. В текстовом поле Width In Inches (Ширина в дюймах) напишите 4.5. Отключите флажок Control Box, как показано на рис. 21.7.

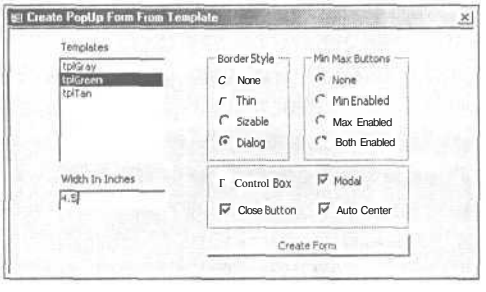


Рис. 21.7. Установки, которые вы выбрали при помощи элементов управления формы CreatePopUp, перенесены на новую форму

4. Нажмите кнопку Create Form (Создать форму). Заметьте, что форма появилась в конструкторе. Если строки DoCmd.Restore там нет, то форма останется минимизированной.
5. Нажмите F4, чтобы открыть список свойств формы, или щелкните правой кнопкой на строке заголовка и выберите **Свойства** (Properties). Выберите вкладку **Все** (АИ). Убедитесь, что ползунок вертикальной полосы прокрутки в самом верху. Также убедитесь, что список свойств растянут вертикально так, чтобы можно было видеть максимальное количество свойств, как показано на рис 21. 8.
   Заметьте, что свойства ScrollBars, RecordSelectors и NavigationButtons отключены согласно шаблону. Также заметьте, что свойство ControlBox установлено на No, как вы указали.
6. Закройте список свойств. Обратите внимание на масштабную линейку наверху формы, на которой установлена ширина 4.5, как вы указали.

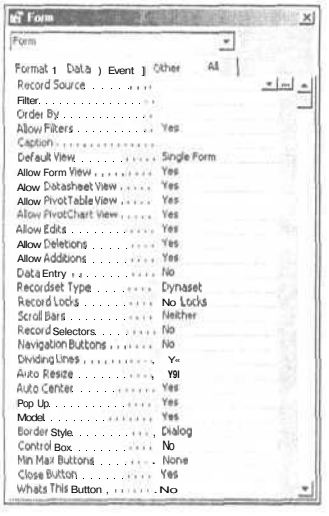


Рис. 21.8, Список свойств созданной формы показывает, что форма CreatePopUp изменила ее (новой формы) свойства

7. Закройте форму, сохраните ее как PopUp и снова откройте. Заметьте, что она отцентрована и не имеет кнопок управления окном. Так как вы не можете вручную закрыть форму, просто щелкните по ней правой кнопкой, выберите **Конструктор** и закройте его. На этой форме нет созданных пользователем элементов управления. Тем не менее это прекрасный способ начать создавать ваши всплывающие формы. Такие контроллеры, как кнопки-флажки или переключатели, могут быть легко добавлены.
8. Удалите форму и оставайтесь в базе данных.

Использование контроллеров всплывающих форм для выбора критерия отчета

Использование контроллеров всплывающих форм для выбора критерия отчета одинаково интересно и весело. Вы можете передавать строковый аргумент в открытый отчет, как описывалось ранее в этой главе, или вы можете добавить строку SQL в аргумент FilterName метода OpenReport. Строка может быть построена по частям с помощью строк, полученных из опций, которые пользователь выбрал на форме.

Следующий пример строит строку SQL, содержащую диапазон дат и выбор штата для критерия. Вы можете либо ввести дату вручную, либо автоматически сгенерировать диапазон дат из выбранных на форме элементов.

Чтобы попробовать на примере, проделайте следующие шаги:

1. Откройте форму CreateCriteria, чтобы просмотреть доступные опции.
2. Во-первых, выбирете FL (Флорида) из спискового окна Choose a State (Выберите штат).

3. Из ниспадающего окна Choose Start Year (Выберите начальный год) выберите 1996. Заметьте, что списковое окно Between (между) изменяется, чтобы отобразить год. Из ниспадающего окна Choose End Year (Выберите конечный год) выберете 1998.
4. Из группы переключателей Choose Month (Выберите месяц) выберите Mar и заметьте, что месяц отображается в обоих окнах, Between и And, как показано на рис. 21.9.

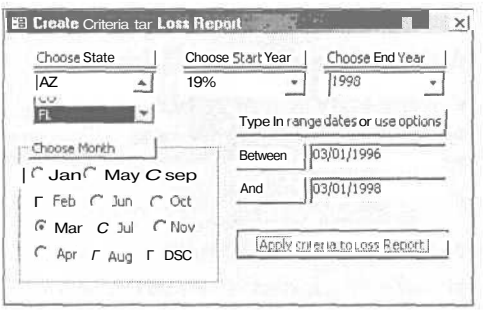


Рис. 21.9. Элементы, которые вы выбрали, используя всплывающую форму CreateCriteria, употребляются для критерия отчета

5. Нажмите на кнопку Apply Criteria to Loss Report (Применить условие отбора для отчета Loss), чтобы открыть Loss Report в окне предварительного просмотра, как показано на рис. 21.10. С помощью курсора-увеличителя щелкните по направлению к середине отчета, чтобы приблизиться достаточно для того, чтобы можно было проверить, что диапазон дат и штат именно те, что вы выбрали. Закройте отчет.

Рис. 21.10. Предварительный просмотр проверяет, что отчет точно отображает то, что вы выбрали для критерия

6. Выберите штат CO (Колорадо) из списка Choose State. Выберите годы 1994 и 1995 соответственно из ниспадающих окон. Нажмите May в группе переключателей.
7. Нажмите кнопку Apply Criteria to Loss Report, чтобы просмотреть результаты изменений критерия. Закройте отчет.
8. Оставьте один месяц, измените штат на AZ и измените оба года на 1993.
9. Нажмите кнопку Apply Criteria to Loss Report, чтобы запустить отчет. На этот раз вы получите сообщение, так как нет записей. Нажмите OK.
10. Закройте отчет и форму, но оставайтесь в базе данных.

До этого момента вы видели, что всплывающие формы могут быть очень полезными. Возможно, вы удивитесь, как работает форма CreateCriteria. Давайте взглянем «за кулисы» и просмотрим код, который делает это.

1. Откройте форму CreateCriteria в конструкторе, чтобы просмотреть код.
2. Выделите мышью поле со списком сразу под подписью Choose Start Year и нажмите F4. Переключитесь на вкладку **Событие** (Event) и нажмите кнопку **Создать** (Build) напротив события **После обновления** (After Update), чтобы просмотреть следующий код:

```
Dim sStart As String, iIndex As Integer
Dim sName As String

iIndex = Me.cboStartYear.ListIndex 'Получаем индекс выборки
sName = Me.cboStartYear.ItemData(iIndex)

sStart = Me.txtStartDate
Me.txtStartDate = Left(sStart, 6) & sName
End Sub
```

   Здесь нет ничего необычного. Год, выбранный здесь, просто присоединен к концу даты в текстовом окне txtStartDate. Окно txtEndDate установлено так же.
3. Нажмите **Правка** (Edit), **Найти** (Find) из строки меню Visual Basic Editor. Наберите optMonth в ниспадающем списке **Найти** (Find What) и затем нажмите **Найти далее** (Find Next), чтобы найти первое встречающееся слово optMonth в модуле. Этот код прикреплен к группе переключателей optMonth. Закройте диалоговое окно **Найти** (Find) и прокрутите вниз, чтобы просмотреть следующий код:

```
Dim ctlMonth As Control, i As Integer
Dim sSDate As String, sEDate As String

Set ctlMonth = Me.optMonth
sSDate = Me.txtStartDate
sEDate = Me.txtEndDate
For i = 1 To 12
  If ctlMonth = i And i < 10 Then
    Me.txtStartDate = "0" & i & Right(sSDate, 8)
    Me.txtEndDate = "0" & i & Right(sEDate, 8)
  ElseIf ctlMonth = i And i >= 10 Then
```

```
    Me.txtStartDate = i & Right(sSDate, 8)
    Me.txtEndDate = i & Right(sEDate, 8)
  End If
Next
```

Переменная ctlMonth представляет Option Group. Option Group имеет 12 значений, пронумерованных последовательно от 1 от 12 в цикле For-Next. Значение ctlMonth будет любым, которое выберет пользователь и которое оказывается месяцем года. Например, если пользователь выбрал И, это значение представляет ноябрь.

Ничего не произойдет, пока ctlMonth = *i,* переменная, представляющая номер текущего цикла. Если номер цикла меньше 10, номер будет иметь длину только 1, что означает, что надо присоединить нуль к номеру в начале. В противном случае оператор ElseIf присоединяет без нуля. Чтобы держать дату в текстовых окнах txtStartDate и txtEndDate, показывающих 10 полных символов, надо добавлять нуль перед месяцем, который является только одним символом. Другие восемь неизмененных символов прикрепляются к выбранному месяцу.

4. Выберите **Правка** (Edit), **Найти** (Find) из строки меню Visual Basic Editor. Наберите cmdViewReport в строке **Найти** (Find What) и нажмите «Искать далее» (Find Next), чтобы найти первое вхождение вашего выбранного элемента. Закройте диалоговое окно. Код, который вы просматриваете, прикреплен к командной кнопке cmdViewReport. Прокрутите всю процедуру и оставайтесь в модуле, пока не прочитаете комментарии после листинга.

```
Dim strSQL As String, strWhere As String
Dim iIndex As Integer, sName As String
Dim sStart As String, sEnd As String

iIndex = Me.lstState.ListIndex 'Получаем индекс выборки
sName = Me.lstState.ItemData(iIndex) 'Получаем компонент

sStart = Me.txtStartDate 'Получаем начальную дату из поля для ввода
sEnd = Me.txtEndDate ' Получаем конечную дату из поля для ввода
strSQL = "Select * from Losses Where " 'Составляем строку
strWhere = "[AccidentState] = "
strWhere = strWhere & Chr$(34) & sName & Chr$(34) & " And "
strWhere = strWhere & "[AccidentDate] " & "between #" _
& sStart & "# And #" & sEnd & "#;"
strSQL = strSQL & strWhere
On Error GoTo GetOut 'Если данные отсутствуют, игнорировать ошибку отчета
DoCmd.OpenReport "Loss Report", acViewPreview, strSQL
'DoCmd.Close acForm, "CreateCriteria"
GetOut:
Exit Sub
```

Большинство из этих процедур включает построение SQL-строки из данных, полученных из контроллеров. Вначале процедура захватывает элемент штата. Затем начальную и конечную даты диапазона. Затем она строит SQL-строку.

Затем выводит отчет на экран с помощью метода OpenReport. Заметьте, что третий аргумент в методе - это строка SQL. Если отчеты не возвращают записей, подпрограмма ошибки останавливает процедуру.

Следующие шаги помогут вам понять, как работает свойство PopUp.

1. Закройте модуль. Щелкните на селекторе формы (Form selector) и нажмите F4, чтобы открыть список свойств. Измените свойство PopUp на Yes. Нажмите кнопку **Вид** (View), чтобы запустить форму. Выберите Colorado (CO) и затем годы 1993 и 1999 в указанном порядке. Нажмите кнопку Apply Criteria to Loss Report, чтобы просмотреть отчет.

   Обратите внимание на две вещи. Независимо от того, что вы делаете, форма остается поверх всех окон. Также заметьте, что вы не можете нажать на отчет под формой. Закройте форму, не сохраняя ее. Теперь можете нажать на отчет. Закройте отчет. Теперь вы на примере увидели, что делают свойства PopUp и Modal. Свойство PopUp обеспечивает расположение формы поверх всех окон. Свойство Modal не дает вам возможности нажимать на другие окна.

2. Снова откройте форму CreateCriteria в конструкторе. Щелкните правой кнопкой мыши на кнопке Apply Criteria to Loss Report и выберите **Обработка события** (Build Event). Удалите апостроф (') перед следующей строкой:

   ```
   'DoCmd.Close acForm, "CreateCriteria"
   ```

3. Закройте окно Module. Нажмите кнопку View, чтобы запустить форму. Выберите Colorado (CO) и затем годы 1993 и 1999 в указанном порядке. Нажмите кнопку Apply Criteria to Loss Report, чтобы просмотреть отчет. Заметьте, что вам предлагается сохранить форму. Это из-за того, что вы удалили апостроф, делающий метод Close активным. Выберите No, чтобы закрыть форму. Оставайтесь в базе данных.

Это упражнение научило вас тому, что форма может закрываться сама, но что более важно - это то, что вы научились управлять взаимодействием объектов. Вы получили сообщение, так как вы запустили форму до сохранения изменений в ней. На этот раз вы хотите, чтобы форма CreateCriteria оставалась открытой, чтобы вы могли сделать другую выборку критерия после закрытия отчета. В других случаях вы, возможно, захотите закрыть форму, которая вызывает отчет, сразу после открытия отчета. Отмените сделанные изменения, поставив апостроф, который вы удалили в процедуре, и сохраните изменения.

Как сказать, что форма открыта

Зачем вам надо знать, открыта ли форма? Если вы хотите найти некоторые данные в форме, вы можете использовать такое выражение:

```
strState = Forms!LossesArg!AccidentState
```

Строка срабатывает, когда открывается форма. Она не исполняется, если форма закрыта, независимо от того, всплывающая это форма или нет. Поэтому бывают случаи, когда вам нужно определить, загружена ли форма. Следующая процедура определяет, загружена ли указанная форма в базе данных:

```
Function IsFormOpen(frm As String) As Boolean
  Dim objAccess As AccessObject, dbObject As Object
  Set dbObject = Application.CurrentProject
  'Просмотреть коллекцию AllForms для загруженной формы
  For Each objAccess In dbObject.AllForms
    If objAccess.IsLoaded = True And _
    objAccess.Name = frm Then
      IsFormOpen = True
      Exit Function
    Else
      IsFormOpen = False
    End If
  Next objAccess
End Function
```

Теперь у вас есть функция, которая может быть вызвана из процедуры, чтобы определить, открыта ли конкретная форма так, чтобы в ней можно было найти значения таким образом:

```
If IsFormOpen("LossesArg")
  strState = Forms!LossesArg!AccidentState
End if
```

Станьте многомерным с PivotTables

Сводная таблица (PivotTable) - это инструмент анализа, который обеспечивает большую гибкость при просмотре сводных данных под разными ракурсами. Вы можете думать о PivotTable как о расширяющейся сводке, так как она аккуратно сохраняет информацию, предназначенную для просмотра расширением вашей выборки. Например, если вы просматриваете сводку по годам, вы можете указать любой год, чтобы просмотреть его детализацию. Способность «надевать на стержень» ракурсы вашей таблицы, трансформируя строки и столбцы, дала PivotTable ее название.

PivotTables по-новому определяет понятие «интерактивный», которое относится к возможности пользователей вводить данные или команды в программы, в противоположность только что просмотренным данных. Таблицы и формы в некоторой степени интерактивны. Вы можете сортировать, фильтровать и изменять данные с таблицами и формами. PivotTables позволяет пользователям быстро суммировать или перекрестно табулировать большие объемы данных. Если вы хотите чередовать строки и столбцы интерактивно («на лету»), используйте PivotTables.

Превращение из PivotTable в PivotChart требует нажатия всего одной кнопки, делающей PivotCharts полностью интерактивными. PivotChart - это диаграмма, привязанная к PivotTable и полученная из нее.

PivotTables по функциональности лучше запросов Crosstab (перекрестные табличные данные), когда дело доходит до просмотра данных с различных точек зрения. Хотя PivotTables недоступны в Access 2000, в Access 2002 они являются долгожданным дополнением. Вы можете взять любую существующую форму и быстро создать из нее PivotTable несколькими щелчками мыши, или вы може-

те использовать мастер PivotTable Wizard под формами в окне Database. Но вы не ограничены только формами. Вы можете просмотреть любую таблицу или запрос Access, любую проектную таблицу Access, вид, хранимую процедуру или форму либо в виде PivotTable, либо в виде PivotChart. Если этого недостаточно, вы можете сохранить их PivotTable и PivotChart виды как Data Access Pages, которые могут быть просмотрены и использованы другими с помощью их браузеров (программы просмотра).

Если вы дважды щелкните по таблице или запросу и затем щелкните правой кнопкой по строке заголовка, то сможете выбрать вид PivotTable. После того как вы поместили поля, которые вам надо, в нужное место, сохраните вид с таблицей или запросом.

Чтобы показать вам, насколько это просто, давайте создадим для примера PivotTable:

1. В разделе **Формы** (Forms) нажмите **Создать** (New), чтобы открыть диалоговое окно **Новая форма** (New Form).
2. Нажмите **Автоформа: сводная таблица** (AutoForm:PivotTable) и выберите Losses из ниспадающего списка таблиц и запросов. Нажмите OK, чтобы открыть пустую сводную таблицу (PivotTable), показанную на рис. 21.11.

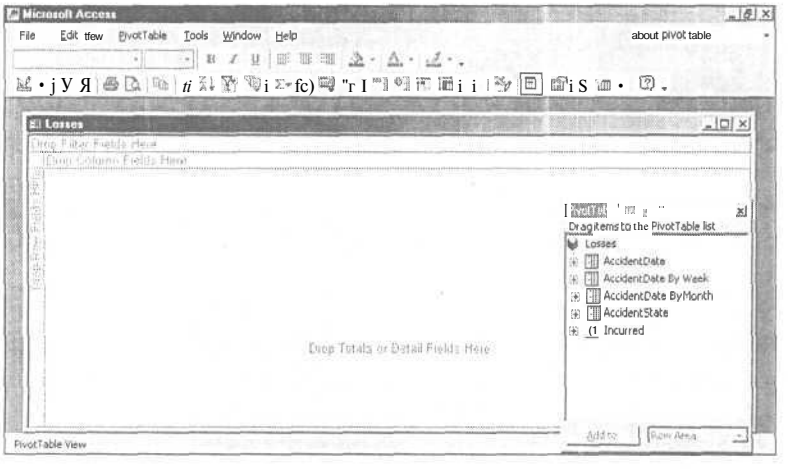


Рис. 21.11. Вы можете брать и перетаскивать поля в пустую сводную таблицу

3. В списке полей сводной таблицы нажмите на знак «+» около AccidentDate By Month, чтобы раскрыть его. Нажмите на Years и перетащите его в раздел под названием «Перетащите сюда поля строк» (Drop Row Fields Here), пока он не выделится голубым, как показано на рис. 21.12, и отпустите его.
4. Щелкните правой кнопкой по заголовку строкового раздела Years, который вы только что перетащили, и выберите **Свернуть** (Collapse). Перетащите свернутое AccidentDate из списка полей на правую часть Years, пока не увидите синюю вертикальную линию, как показано на рис. 21.13, и отпустите его.

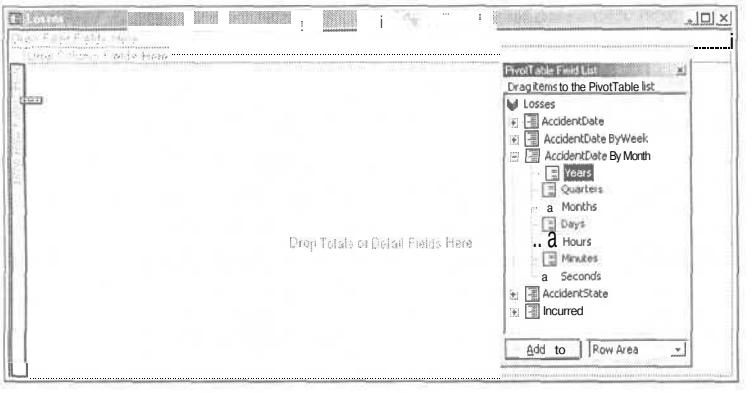


Рис. 21.12. Когда вы перетаскиваете и отпускаете поле года в раздел, помеченный как «Перетащите сюда поля строк» (Drop Row Fields Here), он растягивается и помещает в себя строкой

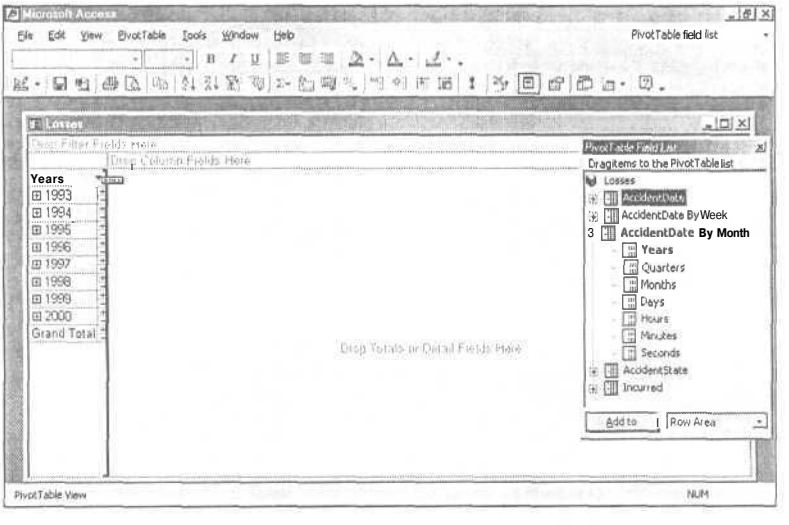


Рис. 21.13. Когда вы увидите синюю вертикальную линию около раздела Years, отпустите поле, которое вы хотите увидеть, когда расширится Years

5. Из списка полей PivotTable перетащите свернутое AccidentState в раздел, помеченный как **Перетащите сюда поля столбцов** (Drop Column Fields Here).
6. Щелкните мышью (не перетаскивайте) на свернутом поле Incurred.
7. В конце списка полей PivotTable выберите **Данные** (Data Area) из ниспадающего списка около кнопки **Добавить в** (Add to).
8. Нажмите на кнопку **Добавить в** (Add to), чтобы добавить поле с итоговыми результатами в сводную таблицу (PivotTable), как показано на рис. 21.14.

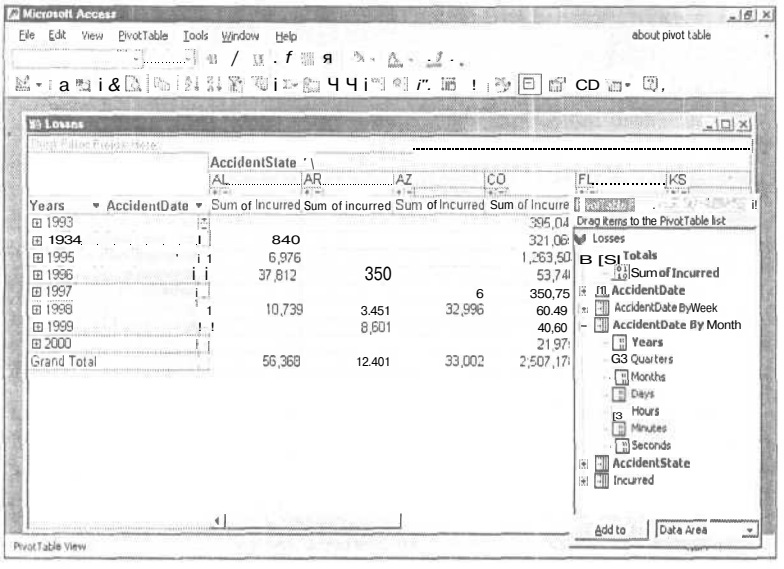


Рис. 21.14. Когда вы используете кнопку Add, чтобы добавить финансовое поле в область данных PivotTable, то добавляются итоги

9. Щелкните правой кнопкой на заголовке **Сумма поля Incurred** в PivotTable и выберите **Свойства** (Properties), чтобы открыть диалоговое окно свойств. Переключитесь на вкладку **Заголовки** (Captions), как показано на рис. 21.15.

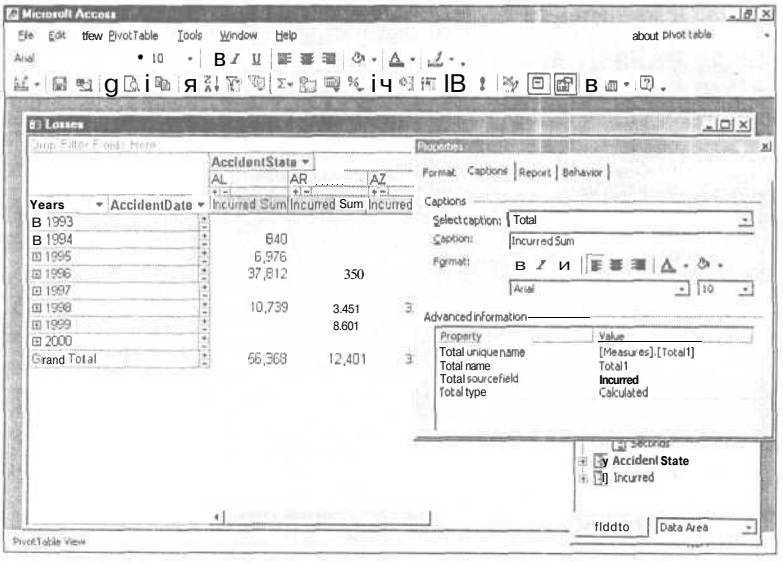


Рис. 21.15. Вкладка «Заголовки» (Captions) в диалоговом окне **Свойства** (Properties) позволяет вам изменять заголовок столбца

10. Измените «Сумма поля Incurred» на «Incurred Sum» и заметьте, что заголовки поменялись у всех полей.
11. Закройте диалоговое окно **Свойства** (Properties). Раскройте год 2000 нажатием на знак «+» около блока. Заметьте, что Colorado - единственный штат, у которого все убытки пока относятся к этому году. Также заметьте, что вы можете сразу точно сказать, когда они имели место.
12. Закройте PivotTable и сохраните ее как Loss Years.
13. В разделе **Таблицы** (Tables) откройте таблицу Sales просто для того, чтобы освежить в памяти, как она выглядит. Обратите особое внимание на избыточность. Это таблица, которую вы используете для следующей сводной таблицы, которая аннулирует избыточность.
14. В разделе **Формы** (Forms) нажмите **Создать** (New), чтобы открыть диалоговое окно **Новая форма** (New Form), и выберите **Автоформа: сводная таблица** (AutoForm: PivotTable). Выберите Sales из ниспадающего списка и нажмите OK, чтобы открыть пустую сводную таблицу.
15. Из списка полей сводной таблицы перетащите Regions в область **Перетащите сюда поля строк** (Drop Row Fields Here). Перетащите Years в область **Перетащите сюда поля столбцов** (Drop Column Fields Here).
16. Выделите Sales и затем выберите **Данные** (Data Area) из ниспадающего списка около кнопки **Добавить в** (Add to). Нажмите кнопку Add to, чтобы добавить поле в сводную таблицу, как показано на рис. 21.16.

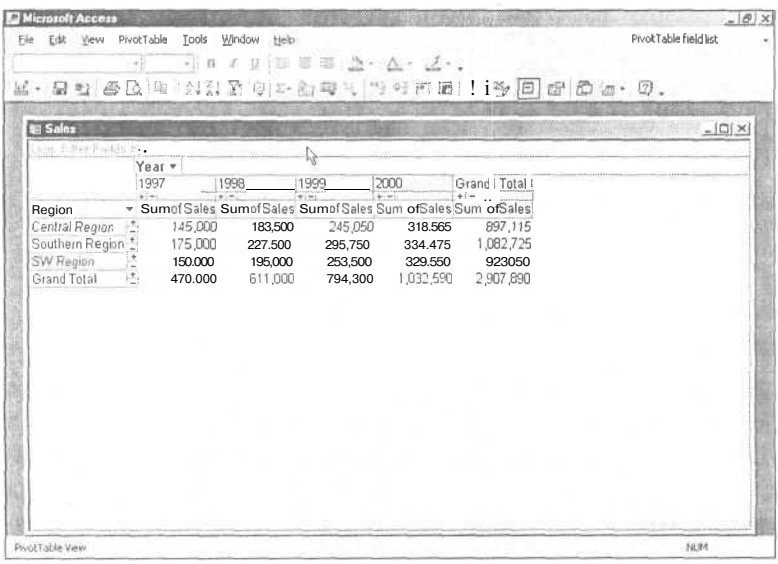


Рис. 21.16. Новая сводная таблица, основанная на таблице Sales, аннулирует избыточность

17. Щелкните правой кнопкой по заголовку Region в разделе строк сводной таблицы и выберите **Удалить.**
18. Нажмите на заголовок Year в разделе столбцов и перетащите его в раздел строк, помеченный как **Перетащите сюда поля строк** (Drop Row Fields Here).
19. Из списка полей сводной таблицы перетащите поле Region в раздел столбцов сводной таблицы, помеченный как **Перетащите сюда поля столбцов** (Drop Column Fields Here), как показано на рис. 21.17.
20. Закройте сводную таблицу, сохранив ее как YearsRegions, но останьтесь в базе данных.

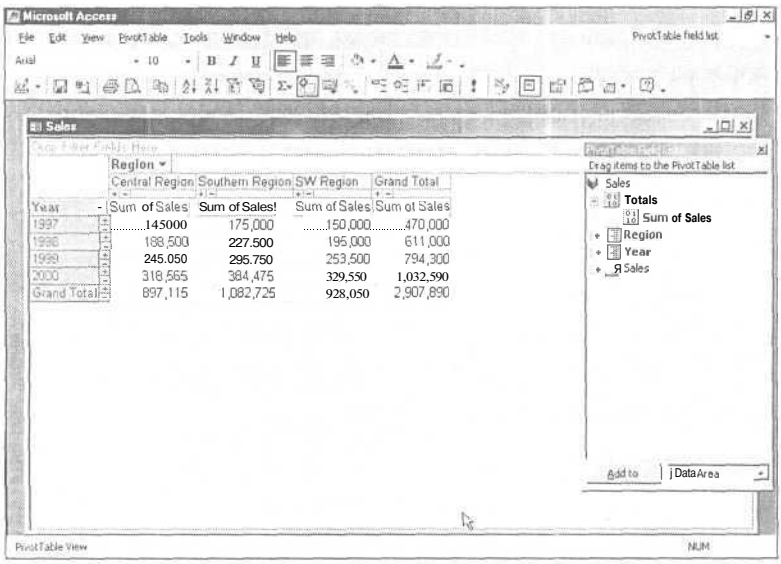


Рис. 21.17. Используя сводную таблицу, легко просмотреть ваши данные с разных точек зрения

Использование форм для управления редактированием и сохранения записей

В однопользовательской среде можно подумать, что Access в состоянии управлять такими задачами, как сохранение записей в формах, особенно когда вы полагаете, что Access автоматически сохраняет записи, которые вы добавляете или редактируете, сразу, как только вы перемещаете точку ввода (курсор) на другую запись или закрываете текущую форму. Все это не так в многопользовательской среде. Вы хотите, чтобы два или больше пользователя имели доступ к одной и той же записи одновременно или могли сохранять ее изменения? Это зависит от того, что вы хотите сделать. Табл. 21.3 объясняет различие трех установок свойства RecordLocks на форме.

Таблица 21.3. Установки свойства RecordLocks

Свойство	Значение	Комментарий
No Locks	0	Это свойство часто называется «оптимистическим блокированием». Это настройка по умолчанию. В формах два или более пользователя могут редактировать одну и ту же запись одновременно. Если два пользователя попытаются сохранить изменения для одной и той же записи, Microsoft Access выводит сообщение второму пользователю, который пытается сохранить запись. Этот пользователь затем может оставить запись, скопировать ее в буфер или удалить изменения, сделанные другим пользователем. Эта установка обычно применяется в однопользовательских базах данных или в многопользовательских, в которых одному или более пользователям нужно иметь возможность изменять одну и ту же запись одновременно
All Records	1	Все записи в лежащей в основе таблице или запросе заблокированы, пока форма открыта в окне формы или виде Datasheet (изображение данных в виде таблицы). Хотя пользователи могут читать записи, никто не сможет редактировать их, добавлять или удалять любую запись, пока форма не будет закрыта
Edited Record	2	Это свойство часто называется «пессимистическим блокированием». Страница записей блокируется, как только какой-нибудь пользователь начнет редактирование любого поля в записи, и остается заблокированной, пока пользователь не перейдет к другой записи. Таким образом, запись может редактироваться в данный момент только одним пользователем.

Хотя No Locks является установкой по умолчанию, когда вы только устновили Access, вы можете изменить настройку по умолчанию для блокирования записей в разделе Advenced, выбрав Tools, Options из строки меню. Помните, что значение по умолчанию для блокировки записей игнорируется, когда вы открываете набор записей посредством кода. Также помните, что вы можете усилить оптимистическое блокирование, установив свойство AllowEdits на False.

Оптимистическое блокирование против пессимистического

В чем разница между методиками оптимистического и пессимистического блокирования и какую из них следует применять? Оптимистическое блокирование ссылается на схему, используемую Jet, ожидающим запрос на блокирование

страницы записей, пока пользователь не сохранил изменения в записях. Как показано в табл. 21.3, Jet использует оптимистическое блокирование записей по умолчанию. При использовании пессимистического блокирования страница, содержащая редактируемую запись, немедленно блокируется, становясь недоступной для других пользователей, пока пользователь не отключит блокировку, сохранив или отменив изменения в записи. Рассмотрение преимуществ и недостатков обеих методик будет полезным для вас.

Преимущества пессимистического блокирования следующие:

- пользователи не получают сообщений, которые могут запутать их;
- пользователи не имеют контроля над тем, чего не понимают;
- пользователи не могут перезаписывать изменения других пользователей;

Недостатки пессимистического блокирования следующие:

- страница записей обычно заблокирована;
- блокировки удерживаются длительный промежуток времени;

Преимущества оптимистического блокирования следующие:

- пользователи имеют одновременный индивидуальный доступ к записям;
- блокировки удерживаются на короткий промежуток времени;
- находчивые пользователи имеет выбор при редактировании возникающих конфликтов.

Недостатки оптимистического блокирования следующие:

- пользователи могут получать сообщения, которые их собьют с толку;
- пользователи могут перезаписывать изменения других пользователей.

Оптимистическое блокирование в основном более предпочтительно, так как обычно нехорошо на большой период времени препятствовать пользователям иметь возможность изменять данные. Если условия вынуждают, вы всегда можете усилить пессимистическое блокирование.

Обновление против повторного запроса

В среде, в которой данные в сети открыты для общего применения, другие пользователи могут изменять данные в то же время, когда вы их просматриваете в форме. Access обновляет данные, которые вы видите, через постоянные промежутки времени. Тем не менее вы можете ускорить немедленное отображение большей части текущих данных путем обновления записей. Помните, что обновление записей только обновляет записи, которые уже существуют в вашей форме. Оно не выводит новые записи и не удаляет удаленные. Для выполнения этих операций сделайте повторный запрос.

Вы можете изменить интервал обновления в Advanced, выбрав в строке меню Tools, Options. Некоторые приложения могут поддерживать уменьшение

60-секундного интервала по умолчанию. Это предоставляет более быструю обратную связь с пользователем. С другой стороны, очень маленький интервал может создать слишком много сетевого трафика, особенно в больших сетях.

Если вы хотите достичь большего понимания методик блокирования и ускоренного обновления/повторного запрашивания на примере, выполните следующие шаги:

1. В разделе **Формы** (Forms) откройте SaveEdit в режиме формы, как показано на рис. 21.18.

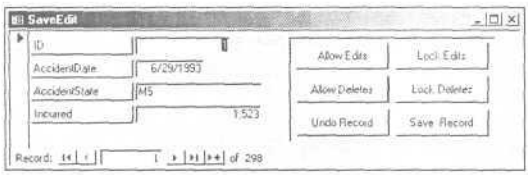


Рис. 21.18. Форма SaveEdit дает вам лучшее понимание блокирования записей

2. Измените поле AccidentDate с 6/29/1993 на 6/29/1994.
3. Измените поле AccidentState с MS на MZ.
4. Из строки меню выберите **Записи** (Records), **Сохранить запись** (Save Record) (или нажмите Shift+Enter). Из панели инструментов нажмите Undo (Отмена). Заметьте, что изменения отменены.
5. Выполните пп. 2 и 3. Нажмите Page Down, чтобы перейти к следующей записи. Нажмите Page Up, чтобы вернуться к предыдущей записи. Нажмите Undo из панели инструментов. Заметьте, что изменения не отменены.
6. Измените поле AccidentDate с 6/29/1994 на 6/29/1995.
7. Измените поле AccidentState с MZ на MY.
8. Нажмите кнопку Save Record (Сохранить запись) на форме и затем нажмите Undo Record (Отменить запись). Заметьте, что запись не отменена как в п. 5. Код, скрывающийся за кнопкой Save record, выглядит так:

```
If Me.Dirty Then
DoCmd.RunCommand acCmdSaveRecord
Me.Refresh
Me.Requery
End If
```

   Если вы удалите строку `Me.Requery`, кнопка выдаст результат как в п. 4. Изменение может быть отменено. Если Me. Dirty - «истина» (что значит, что данные изменены) выполняется операция acCmdSaveRecord, которая эквивалентна выбору **Записи** (Records), **Сохранить запись** (Save Records) из строки меню.
9. Измените поле Accident Date на 6/29/1993 и поле Accident State на MS.
10. Нажмите кнопку Lock Deletes (Запретить удаление), чтобы предотвратить любое удаление данных. Заметьте, что кнопка остается отжатой, так как это

переключатель (элемент управления интерфейсом, имеющий два фиксированных состояния), в отличие от командной кнопки, как показано на рис. 21.19. Также заметьте, что на панели инструментов отсутствует кнопка «Удалить запись» (Delete Record). Даже если вы нажмете на селектор записей слева на форме и нажмете Delete, ничего не произойдет.

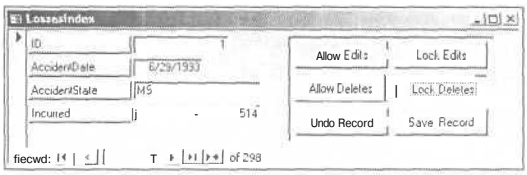


Рис. 21.19. Кнопка-переключатель Lock Deletes остается отжатой, пока вы не нажмете кнопку Allow Deletes (Разрешить удаление)

ЗАМЕЧАНИЕ. Хотя расположенные рядом кнопки-переключатели и командные кнопки выглядят одинаково, вы видите различие при нажатии на кнопку-переключатель. Она остается отжатой, хотя командная кнопка не нажата вовсе. Пока у вас не ограничено свойство TripleState, кнопка-переключатель будет иметь два состояния: включенное и выключенное. Кнопку-переключатель можно заменить на флаговую кнопку или кнопку выбора вариантов.

11. Нажмите кнопку Allow Deletes. Заметьте, что кнопка **Удалить запись** (Delete Record) активирована. Также заметьте, что кнопка Lock Deletes больше не отжата.
12. Нажмите кнопку **Удалить запись** (Delete Record). Появится сообщение о том, что одна запись удалится. Нажмите No.
13. Нажмите кнопку Lock Edits (Запретить изменение) на форме. Попробуйте изменить любое поле. У вас ничего не получится. Попробуйте нажать Allow Edits (Разрешить изменение). Вы не сможете нажать на нее или любую другую кнопку на форме. Вы можете обойти это, изменив процедуру события OnClick для кнопки Lock Edits так, чтобы свойство RecordsetType формы было установлено на Snapshot с помощью такой строки:

    ```
    Me.RecordsetType = 2
    ```

    Когда вы используете тип данных Snapshot, вы можете нажимать любую кнопку на форме, так как можно воздействовать только на данные. Если вы изменили кнопку Allow Edit так, что свойство RecordsetType формы установлено на Dynaset (0), соответственно две кнопки будут отжаты. Dynaset позволяет редактировать, но так как Snapshot только для чтения, он этого не позволяет.
14. Закройте форму, но оставайтесь в базе данных.

Чтобы еще немного лучше понять, как работают кнопки на форме SaveEdit, давайте посмотрим на код за этой формой:

```
Private Sub cmdSave_Click()
If Me.Dirty Then
DoCmd.RunCommand acCmdSaveRecord
Me.Refresh
Me.Requery
End If
End Sub

Private Sub cmdUndo_Click()
If Me.Dirty Then
DoCmd.RunCommand acCmdUndo
Me.Refresh
End If
End Sub

Private Sub TogAllowDeletes_Click()
Me.AllowDeletions = True
Me.Refresh
Me.togLockDeletes = False
End Sub

Private Sub togLockDeletes_Click()
Me. AllowDeletions = False
Me.Refresh
Me.TogAllowDeletes = False
End Sub

Private Sub togAllowEdits_Click()
Me.AllowEdits = True
Me.Refresh
Me.togLockEdits = False
End Sub

Private Sub togLockEdits_Click()
Me.AllowEdits = False
Me.Refresh
Me.togAllowEdits = False
End Sub
```

Строка `Me.TogAllowDeletes = False` изменяет отжатое состояние кнопки Allow Deletes из кнопки Lock Deletes. Вы не хотите позволить обеим кнопкам быть отжатыми одновременно, так как отжатое состояние дает знать, можете ли вы удалить записи. Таким образом, если одна кнопка включена, другая выключена и наоборот. Опции `DoCmd. RunCommand` дают вам удобный способ запуска таких команд строки меню, как Undo. Так как кнопка Allow Edits становится бесполезной, когда кнопка Lock Edits отжата, вы можете изменить кнопки, чтобы установить типы наборов записей и проверить их.

Использование форм для оптимизации больших таблиц

Что вы делаете, когда ваша таблица становится слишком большой настолько, что работает по-черепашьи медленно? Когда таблицы становятся большими, пользователь начинают сравнивать. Не важно, насколько креативны ваши приложения - если они медленные, они не будут очень впечатляющими. Если вы использовали все оптимизационные методы в этой книге и вам все еще досаждают проблемы скорости, лучшим выходом для вас будет ограничить размер таблицы. Вы можете разбить таблицы, которые имеют слишком много полей, применив принципы нормализации (упорядочения) из гл. 1. Но, принимая во внимание, что это было сделано, покажем другие альтернативы.

Привязывание устаревших записей к другой таблице

Одна из альтернатив состоит в том, что вы можете удалять старые записи. Единственная проблема с этим в том, что вы никогда не знаете, когда вам может понадобиться доступ к этим записям с разрешенными целями. Вы можете создать архивную таблицу для старых записей, которая может быть прикреплена из вашей формы с помощью такого кода:

```
Dim Cert As Double
Cert = Me.CertNo
DoCmd.RunSql "INSERT INTO Amend SELECT [Clients].* " _
& "FROM [Clients] WHERE [Clients].[CertNo] =" & Cert & ";"
```

Архивной таблице не надо быть в той же базе данных. Вы можете включить дату прикрепления вместе с датой создания записи. Вы не увеличите быстро размер таблицы, используя этот метод.

Активирование и деактивирование записей

Но ваша задача может отличаться. Представим, что вы администратор в высшем медицинском учебном заведении. Записи в таблице, возможно, надо будет активировать или деактивировать. Если вы деактивируете их, добавляя в архивную таблицу, это может закончиться большими неприятностями, так как студенты связаны со страховыми полисами по модели реляционной базы данных. Более подходящая альтернатива - это установить поле Activated с вариантами Yes/No.

Если оно отключено, то оно активировано. Любые активированные записи фильтруются так, что пользователь никогда не увидит деактивированные записи. Таким образом, пользователь работает с меньшим числом записей одновременно. Деактивированные записи имеют другой цвет шрифта, чтобы их можно было отличить, когда нажата кнопка АИ.

1. В разделе **Формы** (Forms) откройте форму Insureds. Нажмите кнопку All, как показано на рис. 21.20.

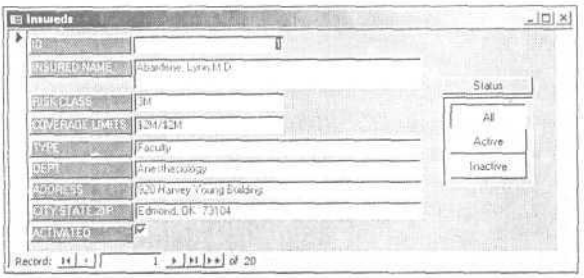


Рис. 21.20. Кнопка АН выключает свойство FilterOn, так что можно просмотреть все записи в форме

2. Нажмите кнопку **Следующая запись** (Next Record) три раза. Заметьте, что тестовые поля в третьей записи имеют другой цвет шрифта. Нажмите **Следующая запись** (Next Record) несколько раз, наблюдая за полем Activated. Заметьте, что каждый раз, когда галочка снята, записи показываются другим цветом шрифта.
3. Нажмите кнопку Inactive. Отфильтровано шесть записей. Даже если это маленькая таблица, метод можно легко применить к большой таблице. Если вы просматриваете эти записи, каждая запись будет иметь один и тот же цвет шрифта.
4. Нажмите кнопку Active и заметьте, что применен фильтр и записи снова изменили цвет.
5. Закройте форму.

Чтобы понять, как работает форма, проверьте следующий код, который прикреплен к методу OnCurrent формы:

```
Dim Ctrl As Control
If Me.Activated = False Then
  For Each Ctrl In Me.Controls
    If TypeOf Ctrl Is TextBox Then
     Ctrl.ForeColor = 9474120
    End If
  Next
Else
  For Each Ctrl In Me.Controls
    If TypeOf Ctrl Is TextBox Then
     Ctrl.ForeColor = 10040115
    End If
  Next
End If
```

Эта процедура демонстрирует, как вы можете вложить конструкцию I" в цикл For-Each, который вложен в другую конструкцию If. Если поле Activated, у которого два состояния Yes/No, является False, установите все текстовые окна на форме на другой цвет. При открытии формы установится фильтр посредством свойства события ОпОреп со следующим кодом:

```
Me.Filter = "[Activated] = True"
Me.FilterOn = True
```

Довольно прямольнейно, а? Программирование для опции управления группами под названием optStatus, которая включает состояние поля Activated, так же просто, как показано в следующем коде:

```
Select Case Me.optStatus
  Case 1
  Me.FilterOn = False
  Case 2
  Me.Filter = "[Activated] = True"
  Me.FilterOn = True
  Case 3
  Me.Filter = "[Activated] = False"
  Me.FilterOn = True
End Select
```

Просмотр последовательных порций данных против просмотра целой таблицы

Другая альтернатива, когда рассматривается методика работы с большими таблицами, дает возможность пользователю видеть сразу только маленькие порции данных. Приводимая ниже процедура, связанная с кнопкой, по нажатии которой появляется следующая страница, дает вам основу идеи, которую можно развить:

```
Dim sSQL As String
Dim iMin As Integer, iMax As Integer

iMin = Me.ID + 20
iMax = iMin + 20
sSQL = "Select * FROM Losses"
sSQL = sSQL & "Where [ID] Between " & iMin & " And " & iMax
Me.RecordSource = sSQL
```

Переменная iMax всегда на 20 записей впереди iMin. Когда вы нажимаете командную кнопку, вы имитируете переход к следующей странице запуском запроса, который захватывает следующие 20 записей из текущей позиции, определенной полем ID. Обратите внимание на конкатенацию переменных iMin и iMax в строке SQL. Также заметьте, что источник записей изменяется каждый раз при нажатии кнопки.

Выигрыш от табуляторных элементов управления (элемент управления, используемый для постраничного представления многостраничной информации пользователю)

Когда вам надо справиться с большим объемом данных на форме, пространство на которой ограничено, табуляторные элементы управления - это, возможно, ваш ответ. Они позволяют вам иметь много страниц с данными в маленькой

области путем группировки схожих категорий данных в каждом элементе. Вы перемещаетесь между этими областями просто нажатием на нужную область. Воспринимайте табуляторные элементы управления как папки электронных файлов в вашей визуальной картотеке.

Табуляторные элементы управления идеальны для меню, диалоговых окон, подформ, изображений, стандартных или ActiveX элементов управления. Хотя вы можете создать иллюзию вложенных ТЭУ, установив программно видимое свойство, если вы поместите ТЭУ на любую страницу другого ТЭУ и затем просмотрите форму в окне формы, вставленный ТЭУ будет видимым из любой страницы главного ТЭУ. Таким образом, ни для каких целей вы не можете вкладывать ТЭУ. рис. 21.21 демонстрирует различные способы, которыми может быть форматирован ТЭУ.

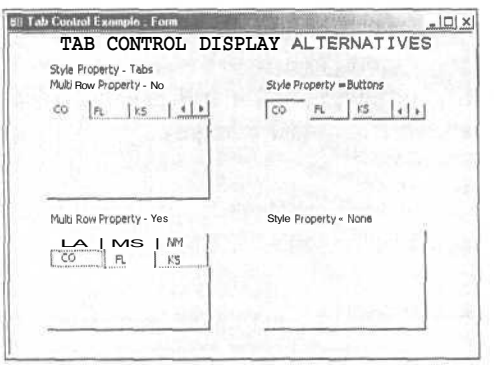


Рис. 21.21. Вы можете показать одиночные или составные строки ярлыков на ТЭУ Access

Использование ТЭУ в качестве замены для многостраничных форм

До того как ТЭУ появились в Access 97, если у вас было слишком много данных для одной страницы, вам приходилось прибегать к созданию другой страницы формы, которая могла быть временами неудобной и сбивающей с толку. ТЭУ предоставляют способ аккуратно и систематично выводить вашу информацию с помощью техники, которая имеет смысл.

Есть два метода вывода информации на ТЭУ. Первый метод - «поддерживать на поверхности» информацию. Он означает, что вы будете видеть тот же элемент, например подформу, или группу элементов, например текстовые окна, при переходе от страницы к странице. Второй метод - вставлять информацию. При использовании этого метода каждая страница выводит разную информацию, когда вы переходите от страницы к странице.

В окне проекта формы вы можете либо поместить элементы управления прямо на ТЭУ из панели инструментов, либо вырезать и вставить элементы управления из места вне ТЭУ на форме. Если вы поместите их прямо из панели инструментов, убедитесь, что вы вывели элемент управления прямо на страницу, ко-

торая стала выделенной и в которую вы хотите вставить элементы управления. Помещение элемента управления таким способом гарантирует, что он будет помещен на страницу. Если вы хотите, чтобы элемент управления «плавал на поверхности», вывидете его вне ТЭУ перед тем, как перетаскивать его наверх любой одиночной страницы.

Если вы вырезаете и вставляете ЭУ (или группу ЭУ) извне ЭУ или формы, убедитесь, что у вас есть одиночная страница, в которую вы хотите вставить ЭУ, выбранный до приклеивания объекта, иначе объект не вставится на страницу. Например если у вас есть раздел детализации, выбранный до приклеивания ЭУ, объект будет «плавать на поверхности», когда вы запустите форму. Используя эту технику, вы можете решать, когда вы будете вставлять, а когда «плавать на поверхности». Вам не надо запускать форму, чтобы определить, вставлены ли ЭУ или они «плавают»: вы можете определить это исходя из окна проекта.

Если у вас есть родительская таблица на одной странице и подформа с дочерней таблицей на другой, вы, очевидно, захотите вставить ЭУ. Тем не менее, если вы хотите использовать ТЭУ для фильтрования одной и той же информации по значению, представленному выбранным ТЭУ, вам следует сделать информацию «плавающей».

Использование ТЭУ для организации подформ

Представьте, что у вас есть большая главная таблица с 20 полями, которая связана с другой таблицей в окне связи. После того как вы создали форму, на стр. больше нет места для подформы. Вы можете поместить ТЭУ на форму, используя одну страницу для родительской таблицы, а другую - для подформы с дочерней таблицей как ее источником записей.

Давайте посмотрим на пример использования ТЭУ для подформ.

1. Откройте базу данных Convention из вашей папки AccessByExample.
2. В разделе **Формы** (Forms) откройте One-To-Many Tab Control Example *в* режиме формы. Нажмите на ТЭУ Reservation Information, как показано на рис. 21.22. Заметьте, что у вас есть две группы кнопок навигации.
3. Нажмите на кнопку «Следующая запись» (Next Record) в нижней группе кнопок. Заметьте, что число показанных записей зависит от связи с родительской таблицей. Также заметьте, что благодаря ТЭУ большое количество информации содержится в маленькой области.
4. Закройте форму.

Изображения также могут занимать большой объем экранного пространства. Вы можете поместить большой рис. на одну страницу в ТЭУ, не доминируя над формой, так как следующая страница может содержать дополнительную информацию. Отличный пример использования изображений с ТЭУ можно найти в базе данных Northwind формы Employee.

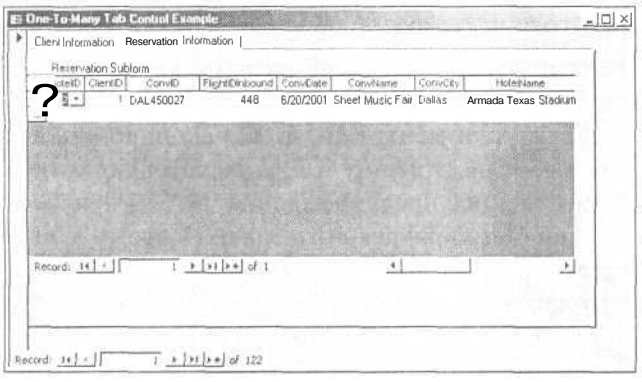


Рис. 21.22. ТЭУ - это прекрасный способ организовать таблицы, имеющие связи «один ко многим»

Использование ТЭУ для организации меню

Если вы хотите, чтобы ваше приложение было управляемым в режиме меню, ТЭУ могут обеспечить альтернативу кнопочной форме. Вы можете установить меню для навигации по вашей базе данных с помощью гиперссылок или кнопок. ТЭУ позволяют вам организовать пользовательские опции в категории для простого доступа. Например, один ТЭУ мог бы содержать отчеты для печати, другой - отчеты для просмотра. Или вы могли бы организовать ваши отчеты о продажах на одном ТЭУ, а отчеты - о расходах на другом. Гиперссылки в комбинации с ТЭУ - это также прекрасный способ организовать ваш любимый Web-сайт.

В качестве примера меню ТЭУ рассмотрим следующие шаги.

1. Откройте базу данных Claims из вашей папки AccessByExample.
2. В разделе **Формы** (Forms) откройте TabControlMenu в режиме формы, как показано на рис. 21.23.

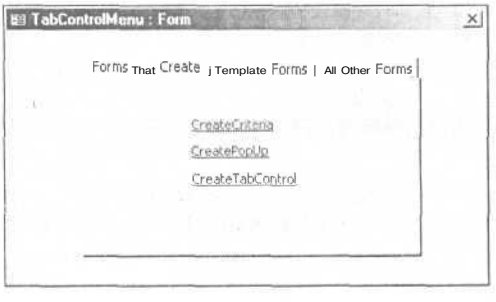


Рис. 21.23. ТЭУ - это прекрасный способ организовать меню для ваших приложений

3. Нажмите на ТЭУ и заметьте, что опции гиперссылок появились в категориях.
4. Нажмите на любую гиперссылку, чтобы открыть форму.
5. Закройте форму, но оставайтесь в базе данных.

Использование ТЭУ в диалоговых окнах

Если у вас есть какие-то сделанные диалоговые окна, которые вы сохранили как формы, вы можете скомбинировать их на одной форме используя ТЭУ. Используйте опцию Select All из опции строки меню Edit, чтобы скопировать все ЭУ из формы. Убедитесь, что вы выбрали страницу ТЭУ, на которую хотите вставить ЭУ, до приклеивания. Единственное предупреждение: любой прикрепленный к исходной форме код должен быть приклеен отдельно. Поэтому лучше всего создать диалоговые окна после создания ТЭУ.

Чтобы посмотреть на пример созданных диалоговых окон с ТЭУ, проделайте следующие шаги:

1. Откройте базу данных Claims из вашей папки AccessByExample.
2. В разделе **Формы** (Forms) откройтеTabControlDialogBox в режиме формы, как показано на рис. 21.24.

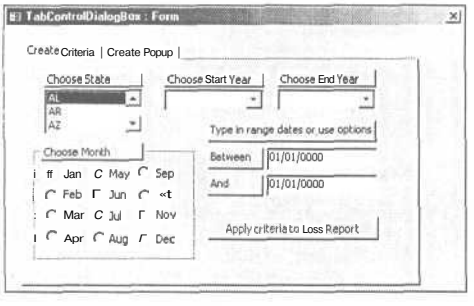


Рис. 21.24. ТЭУ - это прекрасный способ организовать диалоговые окна для ваших приложений

3. Диалоговые окна, показанные на первой странице, должны выглядеть одинаково. Это холостые диалоговые окна просто для показательного примера.
4. Нажмите на ЭУ Create PopUp и закройте форму. Оставайтесь в базе данных.

ПрограммноесозданиеТЭУ

Теперь действительно становится интересно. Вы фактически можете создать ваш ТЭУ программно, чтобы иногда экономить силы. Это особенно полезно, если вам надо вставить большое количество ТЭУ. Если этого недостаточно, это даже лучше. ТЭУ могут отображать информацию о том, что вы организуете. Например, если ТЭУ должен выводить информацию о таблице потерь, вы можете сделать процедуру, создающую отдельный ТЭУ для каждого состояния в таблице, работающий как фильтр.

После того как ТЭУ создан, вам нужно знать, какой ТЭУ выбран. Свойство имени ТЭУ становится частью строки SQL, которая присвоена свойству RecordSource подформы. Подформа «плавает» сверху ТЭУ, так что она появля-

ется на каждой странице. Каждый раз, когда вы нажимаете на ТЭУ, критерий в строке SQL изменяется, отражаясь, в свою очередь, в подформе.

Следующие шаги проведут вас через процесс создания формы и ТЭУ. После того как будет создан ТЭУ, вы научитесь программировать ТЭУ. Затем вы изучите, как писать процедуру, создающую ТЭУ.

1. В разделе **Формы** (Forms) откройте CreateTabControl в режиме формы.
2. Нажмите кнопку Create Tab Control, чтобы создать форму и ТЭУ, показанные на рис. 21.25.

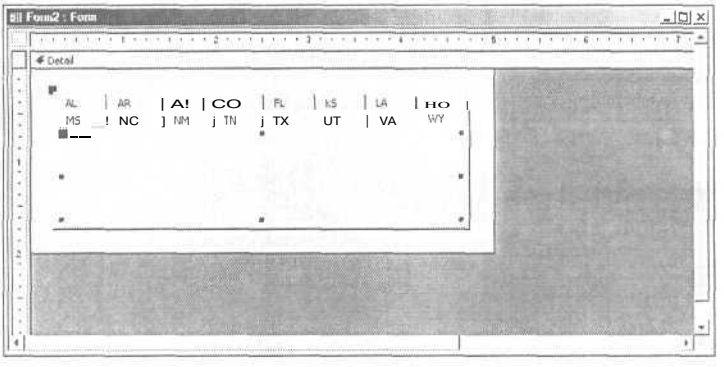


Рис. 21.25. ТЭУ, отображающий каждое состояние таблицы потерь, был создан при нажатии кнопки Create Tab Control

3. Закройте форму без сохранения, чтобы показать форму CreateTabControl снова.
4. На этот раз нажмите кнопку Pick Tab Control, чтобы выбрать штаты, которые вам нужны для состояния ТЭУ.
5. В форме PickTab выделите AZ, CO, FL, TN и TX и затем дважды щелкните на UT, чтобы открыть форму, показанную на рис. 21.26.

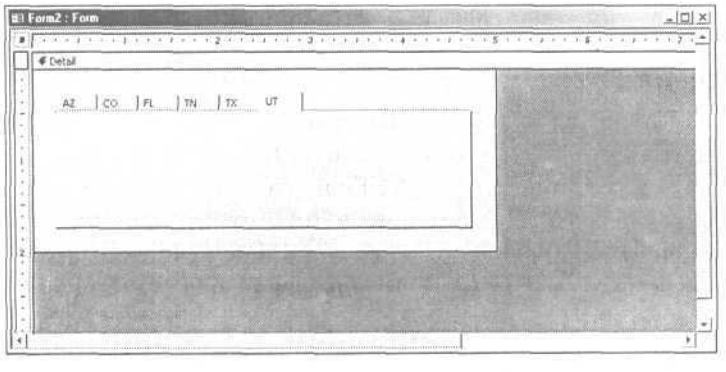


Рис. 21.26. ТЭУ, отражающий штаты из таблицы Losses, которые вы выбрали, был создан при нажатии кнопки Pick Tab Control в форме CreateTabControl

6. Нажмите кнопку **Панель элементов** (Toolbox), чтобы открыть ее. Выбрав «волшебную палочку», нажмите на инструмент **Подчиненная форма** (Subform) и нарисуйте прямоугольник, немного меньший, чем ТЭУ, ниже ТЭУ, как показано на рис. 21.27. Откроется мастер подчиненных форм (Subform Wizard).

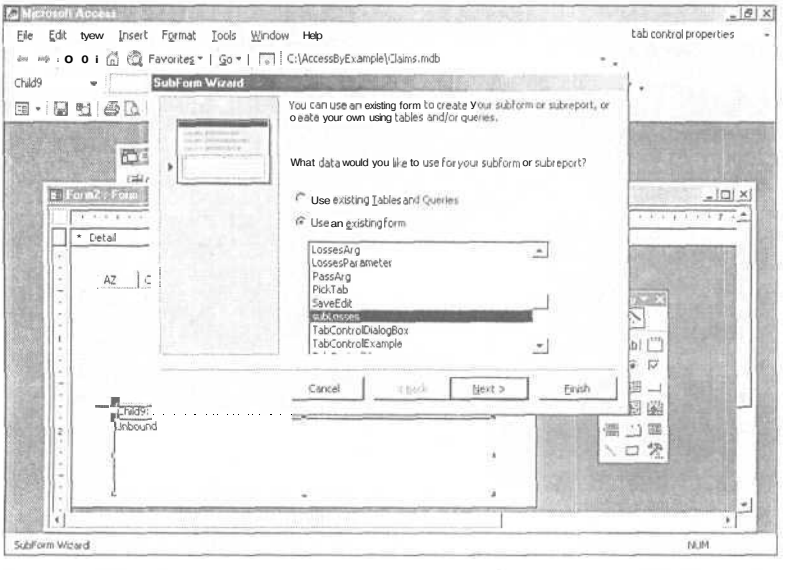


Рис. 21.27. Нарисуйте прямоугольник подформы ниже ТЭУ, чтобы создать «плавающую» подчиненную форму

ЗАМЕЧАНИЕ. Здесь приведено в действие то, что вы изучили в разделе «Использование ТЭУ в качестве замены для многостраничных форм» ранее в этой главе. Вы можете вставить (соединить) подформу на стр. ТЭУ, нарисовав ее прямо на стр., или можете нарисовать подформу в области вне ТЭУ и перетащить обратно на ТЭУ, чтобы она «плавала».

7. На первой странице мастера выберите подчиненную форму subLosses из списка и нажмите **Далее** (Next). Нажмите **Готово** (Finish) на следующей странице мастера, чтобы поместить подчиненную форму на форму.
8. Перетащите подчиненную форму, чтобы поместить ее на ТЭУ. Щелкните на подписи subLosses на верхнем левом крае подчиненной формы и нажмите Delete, чтобы удалить его. Нажмите на некоторые ТЭУ, чтобы проверить, что подчиненная форма «плавает». Перетащите нижний край формы вверх до нижнего края ТЭУ.
9. Нажмите на ТЭУ напротив одной из страниц. Вы можете определить, выбран ли ТЭУ, по тому, что метки-манипуляторы находятся на самом краю ЭУ, как показано на рис. 21.28.

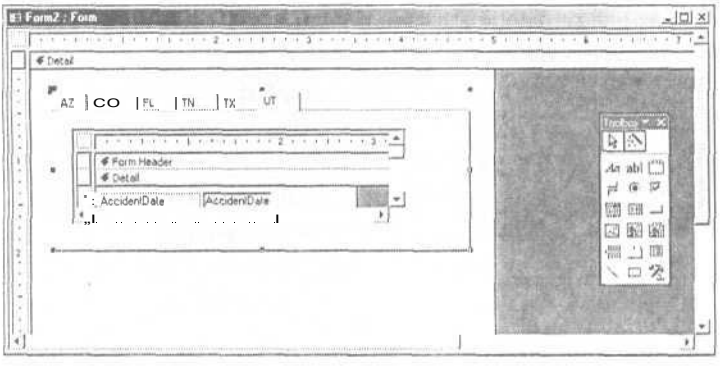


Рис. 21.28. Нажмите на ЭУ так, чтобы вы могли открыть его список свойств

10. Нажмите F4, чтобы открыть список свойств формы. Переключитесь на вкладку **Событие** (Event). Наберите следующий код в окне модуля события: OnChange:

```
Dim Pg As String, sSQL As String

With Me.tabOnFly
   Pg =. Pages(.Value).Name
   sSQL = "Select * from Losses where [AccidentState]=" _
   & Chr$(34) & Pg & Chr$(34)
   Me. subLosses.Form.RecordSource = sSQL
   Me.subLosses.Form.Refresh
End With
```

Имя ТЭУ - tabOnFly. Оператор With имеет доступ к свойствам ТЭУ. Каждый раз, когда вы нажимаете на ТЭУ, срабатывает событие OnChange, которое запускает его процедуру события. Имя ТЭУ доступно с помощью строки Pg =.Pages(.Value). Name. Последняя строка имеет доступ к подчиненной форме прямо через ее управляющее имя, которым является subLosses. Строка SQL затем вставляется в источник управления *подформы*.

11. Закройте модуль, лист свойств и форму, сохранив форму как ControlTabFilter.

12. Снова откройте форму и нажмите на ТЭУ, когда заметите число записей внизу формы.

13. Прокрутите подформу, чтобы проверить, что записи на месте.

14. Закройте форму.

Давайте проанализируем процедуру, которая создала ТЭУ, начав с процедуры, создающей ТЭУ со всеми состояниями:

```
Dim iCount As Integer
Dim frm As Form, iLeft As Integer, iTop As Integer
Dim iWidth As Integer, iHeight As Integer
Dim cn As ADODB.Connection, fld As String
Dim rs As ADODB.Recordset

iCount = 0
```

```
Const TPI = 1440 'TPI = Twips Per Inch, твип на дюйм
  Set cn = CurrentProject.Connection
  Set rs = New ADODB.Recordset
  With rs
   .Source = "States"
   .ActiveConnection = cn
   .CursorType = adOpenKeyset
   .LockType = adLockOptimistic
   .Open
  End With
'Положение ТЭУ
iLeft = 0.25 * TPI
iTop = 0.25 * TPI
'Размеры ТЭУ
iWidth = 4.5 * TPI
iHeight = 1.5 * TPI
Set frm = CreateForm 'Создаем форму для ТЭУ
With CreateControl(frm.Name, acTabCtl, , _
"", "", iLeft, iTop, iWidth, iHeight)
   .Name = "tabOnFly"
   .MultiRow = True
   Do
     fld = rs.Fields("Code").Value
     rs.MoveNext 'Продвигаемся по множеству записей запроса
       If iCount > 1 Then 'Пропускаем первые две страницы
         .Pages.Add
       End If
     .Pages(iCount).Name = fld 'Вкладка для каждого состояния
     iCount = iCount + 1
   Loop Until rs.EOF
End With
DoCmd.Restore
```

Оператор With устанавливает свойства для ТЭУ. Ширина и высота формы в совокупности с ее положением на форме устанавливаются с помощью функции CreateForm. Заметьте, что свойство MultiRow установлено на True, что делает его включенным. Аббревиатуры штатов получаются из таблицы States в поле Code. Коллекция страниц ТЭУ имеет такой доступ, что могут добавляться и именоваться новые страницы. Метод Restore гарантирует, что окно восстановлено так, чтобы вы могли немедленно просмотреть его.

Следующая процедура, дающая вам возможность выбирать то состояние, которое вы хотите, очень похожа. Она прикрепляется к событию OnDblClick формы PickTab.

```
Dim iLeft As Integer, iTop As Integer
Dim iWidth As Integer, iHeight As Integer
Dim frm As Form, iCount As Integer
Dim lstSelect As ListBox, vItem As Variant

iCount = 0
Const TPI = 1440
'Положение ТЭУ
iLeft = 0.25 * TPI
```

```
iTop = 0.25 * TPI
'Размеры ТЭУ
iWidth = 4.5 * TPI
iHeight = 1.5 * TPI
Set frm = CreateForm 'Создаем форму для ТЭУ
With CreateControl(frm.Name, acTabCtl, , _
"", "", iLeft, iTop, iWidth, iHeight)
  .Name = "tabOnFly"
  .MultiRow = True
  Set lstSelect = Me.lstPick
  For Each vItem In lstSelect.ItemsSelected
    If iCount > 1 Then
      .Pages.Add
    End If
  .Pages(iCount).Name = lstSelect.ItemData(vItem)
  iCount = iCount + 1
  Next
DoCmd.Restore
DoCmd.Close acForm, "PickTab"
```

End WithНа этот раз ТЭУ поименованы из раздела окна списка. iCount = iCount + 1 инкрементирует счетчик страниц. Имя спискового окна, из которого выбираются состояния - lstPick. Переменная IstSelect указывает на lstPick и доступна через ее свойство ItemsSelected для определения выбранных в списковом окне пунктов.

ПРЕДУПРЕЖДЕНИЕ. Не используйте настройку Dialog свойства BorderStyle формы, которое создает форму и ТЭУ, так как она генерирует ошибку.

Что дальше?

Эта глава в основном концентрировала внимание на контроле над формами посредством наилучшего использования имеющихся инструментов. Следующая глава посвящена тому же, только для отчетов. Отчеты просматривают ваши данные под разным углом зрения. Она глубже посвящает вас в некоторые принципы, которые вы изучили в гл. 6.

22

Осуществление контроля над отчетами

Гл. 21 - «Осуществление контроля над формами» - закрепляла технику, которая поможет вам контролировать формы. Эта глава посвящена тому же самому, только для отчетов.

Что касается писанины, безбумажный офис еще не появился. Пока торжествует бумага, отчеты будут необходимы. Учитывая это, вы хотите уметь получать максимум от ваших отчетов. Это требует выхода далеко за пределы основ. В частности, эта глава научит вас следующему:

- как программно управлять группировкой и сортировкой;
- как использовать связи, чтобы контролировать, отображены ли данные в отчете;
- как удалять надоедливые пустые строки из отчета;
- как работать с отчетами со сложной структурой столбцов;
- как печатать дополнительные копии записи в отчет.

Управлениегруппировкойисортировкойпрограммно

В гл. 6 - «Анализ отчетов» - вы изучили, как вручную группировать и сортировать записи в отчете, Не было бы здорово уметь это делать «на лету»? Наличие в распоряжении такой функциональности дает возможность просматривать, как выглядит отчет до того, как вы взялись вручную устанавливать постоянные уровни группировки и сортировки в отчете. Или вы можете использовать ее для предварительного просмотра отчета перед его печатью, чтобы посмотреть, удовлетворительны ли уровни группировки и сортировки. Просто нажмите кнопку - и вы немедленно увидите результаты.

Перед тем как вы попытаетесь программно создать групповой уровень, с помощью которого будете группировать или сортировать в отчете, вам нужно определить, какие уровни уже существуют. Иначе ваш код продублирует уровень, который там уже есть. Проблема в том, что Access не предоставляет прямого способа определить, существует ли конкретный групповой уровень в отчете или нет. То же самое и для разделов (секций). Вы можете получить доступ к свойствам уровня с помощью такого кода:

```
Reports!FlexibleSort.GroupLevel(0).ControlSource = strLevel1
```

Тем не менее, если вы попытаетесь добраться до него самостоятельно с помощью следующего присваивания:

```
varGrpLevel = Reports!FlexibleSort.GroupLevel(0)
```

вы сгенерируете ошибку.

Решение состоит в том, чтобы дать ошибке определить, существует ли групповой уровень. Другими словами, если ошибка сгенерирована, уровень не суще-

ствует. Почему бы не дать ошибкам работать на вас для получения желаемого результата? Следующая процедура делает это:

```
Function LevelExists(objReportName As Object, iLevel As Integer) _
As Boolean

'Не нужно беспокоится об источнике данных (ControlSource) кроме как для
тестирования
On Error GoTo Sorry
  If objReportName.GroupLevel(iLevel) _
  .ControlSource <> "" Then
    LevelExists = True
  End If
Exit Function

Sorry:
LevelExists = False

End Function
```

Так как вы объявляете objReportName как объектную переменную, вы используете эту функцию:

```
If LevelExists(Reports!FlexibleSorts, 0 ) Then
  Do something.....
End If
```

Отчет должен быть открыт и уровень должен быть в наличии, чтобы функция возвратила True. Отметьте, что вы не можете использовать только имя отчета в аргументе не ссылаясь на его коллекцию. Второй аргумент - это групповой уровень с отсчетом от нуля. Табл. 22.1 показывает, на что ссылается каждый групповой уровень.

Таблица 22.1. Ссылки групповых уровней

Групповой уровень	На что ссылается
GroupLevel(0)	Первое поле или выражение, по которому вы сортируете или группируете
GroupLevel(1)	Второе поле или выражение, по которому вы сортируете или группируете
GroupLevel(2)	Третье поле или выражение, по которому вы сортируете или группируете

После того как вы определили группы, которые надо создать, вы используете функцию CreateGroupLevel для их создания. Синтаксис функции следующий:

```
CreateGroupLevel(ReportName, Expression, Header, Footer)
```

Она применяется таким образом:

```
varGroupLevel = CreateGroupLevel("Loss Report", "AccidentDate", True, True)
```

Функции надо имя отчета, поле или выражение, по которому группировать, и, если хотите, заголовок или нижний колонтитул. Если вы введете True или False (в таком порядке) для последних двух аргументов, вы укажете заголовок группы и не укажете нижний колонтитул.

Вы можете прикрепить панель инструментов к вашему отчету, который предоставит вам возможность группировать ваш отчет согласно вашим спецификациям. Процедуры, которые вы создаете, становятся опцией в вашей панели инструментов. Следующий пример использует процедуры панели инструментов, чтобы сгруппировать отчет FlexibleGroup.

Перед тем как идти дальше, давайте посмотрим на примере, что вы можете делать с групповыми уровнями:

1. Откройте базу данных Claims из папки AccessByExample. В разделе **Отчеты** (Reports) откройте отчет FlexibleGroup. Обратите внимание на панель инструментов, которая прикреплена к отчету.
2. Щелкните правой кнопкой по строке заголовка отчета в окне предварительного просмотра и затем раскройте ниспадающий список **Масштаб** (Zoom), как показано на рис. 22.1. Выберите 75 % .

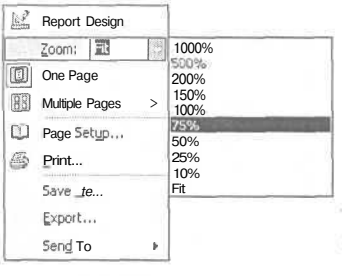


Рис. 22.1. Когда вы щелкаете правой кнопкой по окну отчета в окне предварительного просмотра, вы можете изменять масштаб изображения

3. Расширьте правую сторону окна отчета. Щелкните на OneLevel (Один уровень) из панели инструментов, чтобы сгруппировать отчет, как показано на рис. 22.2.
4. Нажмите TwoLevel (Два уровня), чтобы добавить другой уровень в отчет. Ваш первый групповой уровень - AccidentState, а второй - AccidentDate, но их довольно сложно определить без обозначения по крайней мере одного группового заголовка.
5. Щелкните правой кнопкой по строке заголовка отчета и выберите конструктор отчета (Report Design), чтобы открыть отчет в режиме конструктора. Переместите поле AccidentState из области данных (Detail Section) в раздел «Заголовок группы AccidentState» (AccidentState Header) под подпись AccidentDate. Переместите поле AccidentDate под подпись AccidentState в область данных (Detail Section), как показано на рис. 22.3.

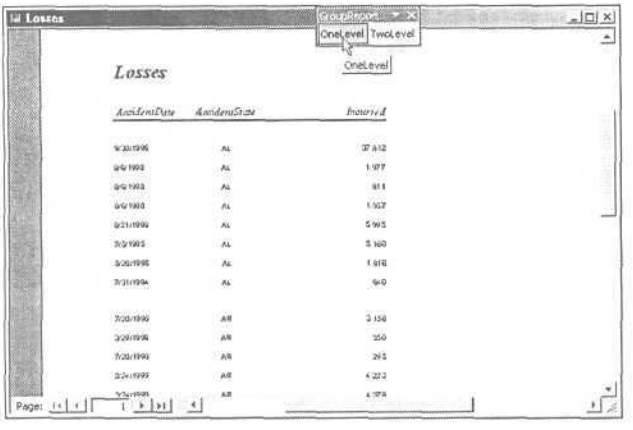


Рис. 22.2. После масштабирования вашего отчета вы можете запустить одну из процедур панели инструментов

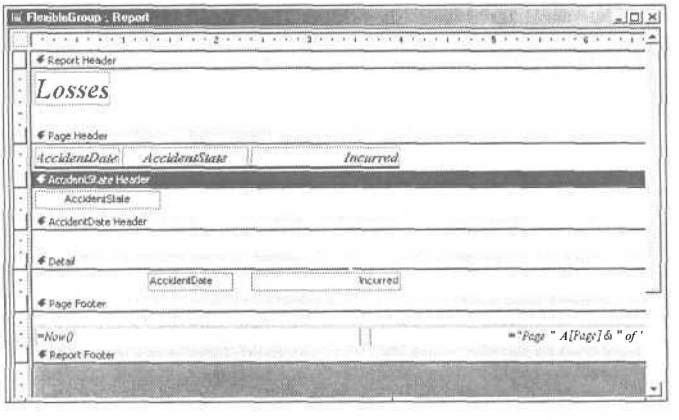


Рис. 22.3. Вы регулируете позиции полей отчета, чтобы лучше понять, как работают группы

6. Переставьте подписи AccidentDate и AccidentState так, чтобы метка AccidentState была первой слева. Нажмите **Предварительный просмотр** (Print Preview) из панели инструментов и затем прокрутите окно до тех пор, пока не увидите по крайней мере два штата, как показано на рис. 22.4. Обратите внимание на то, как в поле AccidentDate сгруппированы похожие значения. Если значение-дата не имеет схожего значения, с которым группироваться, запись группируется отдельно, что объясняет появление дополнительных пустых строк.
7. Закройте отчет, не сохраняя его. Оставайтесь в базе данных.

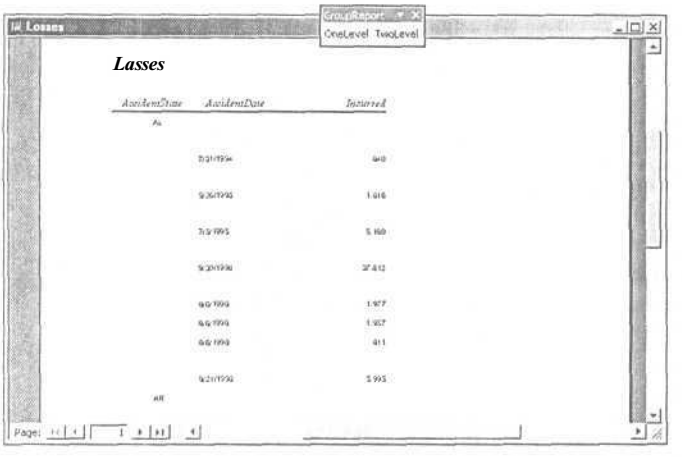


Рис. 22.4. Окно Print Preview показывает группирование поля AccidentState и подгруппирование поля AccidentDate

Процедура, которая создает группы и подгруппы, выглядит следующим образом:

```
Public Function GroupLevel(strGroup1 As String, _
Optional strGroup2 As String)
Dim vMakeGroup As Variant

On Error GoTo ListError
DoCmd.Echo False
DoCmd.OpenReport "FlexibleGroup", acViewDesign

If LevelExists(Reports!FlexibleGroup, 0) = False Then
  'Создаем групповой уровень на strGroup1
  vMakeGroup = CreateGroupLevel("FlexibleGroup", _
  strGroup1, True, False)
  'Устанавливаем высоту верхнего колонтитула strGroup1
  Reports!FlexibleGroup.Section(acGroupLevel0Header) _
  .Height = 400
End If
If strGroup2 <> '" And _
  LevelExists(Reports!FlexibleGroup, 1) = False Then
  vMakeGroup = CreateGroupLevel("FlexibleGroup", _
  strGroup2, True, False)
  Reports!FlexibleGroup.Section(acGroupLevel1Header) _
  .Height = 400
End If
DoCmd.OpenReport "FlexibleGroup", acViewPreview
DoCmd.Echo True
Exit Function

ListError:
DoCmd.Echo True
MsgBox "Error Code: " & Err.number & vbCr _
& "Error Description: " & Err.Description

End Function
```

Отчет FlexibleGroup не имеет групповых уровней. Причина, по которой процедура LevelExists проверяет любой случай, состоит в том, что пользователь может нечаянно нажать на одно из свойств панели инструментов дважды. Наличие процедуры, проверяющей групповые уровни, дает вам также возможность использовать процедуру с другими отчетами. Чтобы изменить эту процедуру для работы с другими отчетами, просто используйте переменную для представления отчета вместо того, чтобы специально ссылаться на отчет FlexibleGroup. Функция Input хорошо работает, когда вы хотите спросить пользователя, какой отчет надо сгруппировать. Входные данные пользователя присваиваются переменной, которую метод Open report может употребить, вместо того, чтобы конкретно указывать отчет. Вместо вызова функции из панели инструментов вы можете вызывать ее из процедуры, прикрепленной к кнопке, если вам нравится.

Давайте рассмотрим код более подробно. Когда вы открываете отчет, он уже находится в режиме Print Preview. Вы переключаетесь на окно проекта, так как функция CreateGroupLevel не запускается в виде Preview (в окне предварительного просмотра). Итак, вы используете объект DoCmd, который показан ниже:

```
DoCmd.OpenReport "FlexibleGroup", acViewDesign
```

чтобы переключиться в окно проекта и добавить групповой уровень или уровни, какой бы случай ни был.

Затем та же строка, за исключением второго аргумента, запускается снова, чтобы переключиться обратно в режим Print Preview:

```
DoCmd.OpenReport "FlexibleGroup", acViewPreview
```

«Сердце процедуры» находится между двумя этими строчками. Переменная первого уровня - это strGroup1, которая получает значение одного из аргументов функции. Эта переменная определяет, на каком поле будет основываться первый групповой уровень. Обратите внимание на то, что strGroup2 определяется как необязательный аргумент. Это значит, что вам не надо определять второй аргумент, пока вам не понадобится второй групповой уровень. Как упоминалось ранее, функция CreateGroupLevel даже не запускается, пока определено, что нет групповых уровней, которые обращаются ко второму аргументу функции LevelExists. Для первого уровня вы заканчиваете так:

```
If LevelExists(Reports!FlexibleGroup, 0) = False Then
  'Создаем групповой уровень на strGroup1
  vMakeGroup = CreateGroupLevel("FlexibleGroup", _
  strGroup1, True, False)
  'Устанавливаем высоту верхнего колонтитула strGroup1
  Reports!FlexibleGroup.Section(acGroupLevel0Header) _
  .Height = 400
End If
```

Так как два последних аргумента функции CreateGroupLevel - True и False, вы соответственно создаете только групповой заголовок. За исключением некоторых мелочей, вторая конструкция If-Then очень похожа на первую. Вы создаете группу на другом поле и определяете второй уровень во втором аргументе функции LevelExists. Вы также определяете второй уровень, когда устанавли-

ваете высоту раздела в свойствах Section с помощью собственной константы acGroupLevel1Header.

Пример отчета FlexibleGroup контролируется панелью инструментов. Она дает вам механизм просмотра результатов изменений дизайна отчета «на лету». Следующий пример демонстрирует, как создавать сортирующую процедуру и прикреплять ее к панели инструментов.

1. В разделе **Модули** (Modules) нажмите **Создать** (New), чтобы открыть новый модуль.
2. Наберите следующий код после Option Compare Database:

```
Public Function SortLevel(strLevel1 As String, _
strLevel2 As String)

On Error GoTo ListError
DoCmd.Echo False
DoCmd.OpenReport "FlexibleSort", acViewDesign

If LevelExists(Reports!FlexibleSort, 0) Then
  Reports!FlexibleSort.GroupLevel(0) _
  .ControlSource = strLevel1
  Reports!FlexibleSort.GroupLevel(1) _
  .ControlSource = strLevel2
Else
  MsgBox "You need to insert groups into " _
  & "your report", vbOKOnly
End If

DoCmd.OpenReport "FlexibleSort", acViewPreview
DoCmd.Echo True
Exit Function

ListError:
DoCmd.Echo True
MsgBox "Error Code: " & Err.number & vbCr _
& "Error Description: " & Err.Description

End Function
```

3. Закройте модуль и сохраните его как SortReport, оставайтесь в базе данных.

Есть некоторые сходства между этой процедурой и последней. На этот раз тем не менее вы присваиваете значения групповым уровням, вместо того чтобы создавать их с помощью такого кода:

```
Reports!FlexibleSort.GroupLevel(0).ControlSource - strLevel1
```

И опять это выражение никогда не сработает до тех пор, пока отчет открыт в окне проекта. Тем не менее, так как вы прикрепили этот код к отчету, используя панель инструментов, вы гарантируете, что отчет будет открыт, когда будет запущен код. Обе переменные, strLevel1 и strLevel2, являются обязательными аргументами. Они содержат поля, по которым вы хотите сортировать. Порядок, в котором вы поместили их в функции, определяет порядок сортирующих уровней.

Например, если вы набираете ? `SortLevel("AccidentState","AccidentDate")` в окне прямого ввода и с открытым отчетом, функция будет сортировать сначала поAccidentState,азатемпоAccidentDate.

ЗАМЕЧАНИЕ. Убедитесь, что вы окружили имена полей пробелами в скобках без кавычек. Выключение Echo не дает экрану обновляться, пока запущена программа.

Так как почти все в процедуре было объяснено, давайте прикрепим этот код к панели инструментов.

Следующий пример проведет вас через процесс создания панели инструментов с помощью прикрепления кода к ней и прикрепления этой панели инструментов к отчету.

1. Выберите **Сервис** (Tools), Настройка (Customize) из строки меню, чтобы открыть диалоговое окно **Настройка** (Customize), показанное на рис. 22.5.

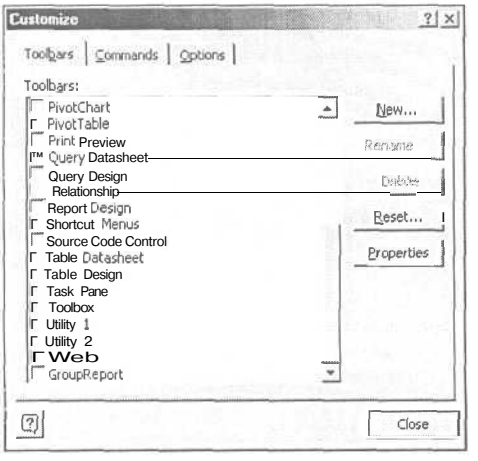


Рис. 22.5. Вы можете добавлять ваши собственные процедуры в панель инструментов, которые могут как запускаться сами по себе, так и быть прикреплены к форме или отчету Access

2. Переключитесь на вкладку **Панели инструментов** (Toolbars), чтобы просмотреть доступные панели инструментов. Нажмите **Создать** (New), чтобы открыть диалоговое окно **Создание панели инструментов** (New Toolbar). Наберите `SortReport` в поле с подписью **Панель инструментов** (Toolbar Name). Нажмите OK и обратите внимание на панель инструментов, которая появляется за диалоговым окном **Настройка** (Customize), как показано на рис. 22.6.
3. Переключитесь на вкладку **Команды** (Commands), чтобы просмотреть список категорий команд. Выделите **Файл** (File) в списке **Категории** (Categories) и **Специальная** (Custom) в списке **Команды** (Commands).

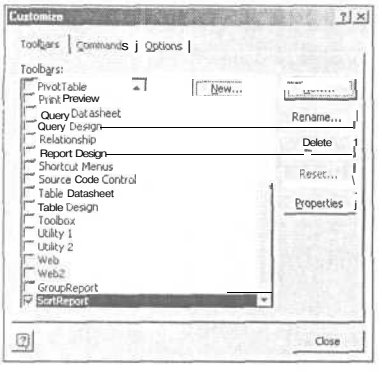


Рис. 22.6. Процедуры могут быть добавлены в новую панель инструментов, которая появляется, когда вы нажимаете OK

4. Перетащите кнопку **Специальная** (Custom) в только что созданную панель инструментов, как показано на рис. 22.7.

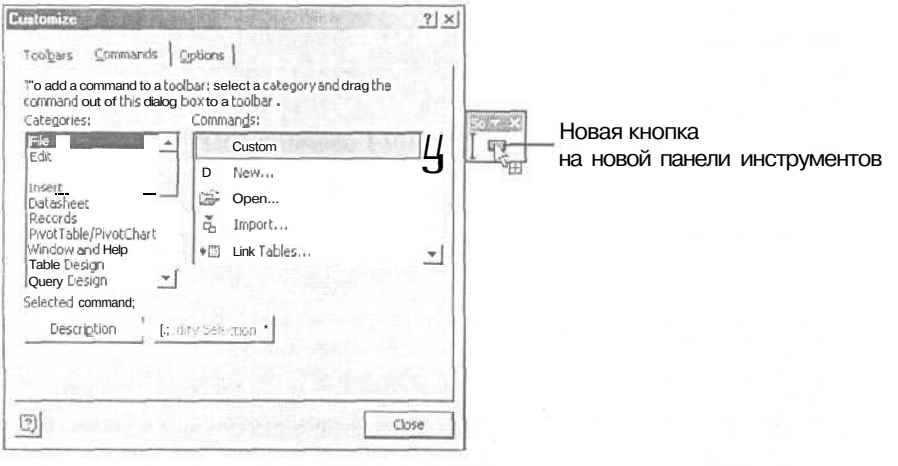


Рис. 22.7. Перетаскивание новой кнопки Специальная (Custom) в панель инструментов дает вам пустую кнопку, к которой вы можете прикрепить код и ярлыки панели инструментов

5. Перетащите еще одну такую же кнопку в панель инструментов. Выделите первую кнопку.
6. Нажмите на кнопку **Изменить выделенный объект** (Modify Selection) и затем выберите **Свойства** (Properties), чтобы просмотреть диалоговое окно **Свойства элемента SortReport** (SortReport Properties), показанное на рис. 22.8.
7. **В Подписи** (Caption) наберите State/Date. Для **Всплывающего сообщения** (ScreenTip) наберите Sort first by AccidentState and then by AccidentDate. **В Действии** (OnAction) наберите `=SortLevel("AccidentState","AccidentDate")`. Не забудьте знак равенства (=). Нажмите **Закрыть** (Close), чтобы вернуться в диалоговое окно **Настройка** (Customize).

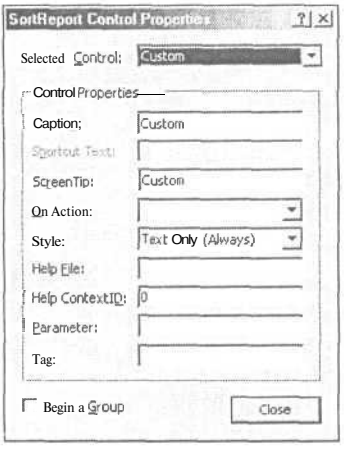


Рис. 22.8. Среди прочих опций диалоговое окно Свойства элемента SortReport (SortReport Properties) дает вам возможность определять процедуру, которая будет запускаться, когда нажата кнопка

8. Выделите вторую кнопку и нажмите кнопку **Изменить выделенный объект** (Modify Selection), а затем выберите **Свойства** (Properties). В **Подписи** (Caption) наберите Date/State. Для **Всплывающего сообщения** (ScreenTip) наберите **Sort first by AccidentDate and then by AccidentState. В Действие** (OnAction) наберите `SortLevel("AccidentDate", "AccidentState")`. Нажмите **Закрыть** (Close), чтобы вернуться в диалоговое окно **Настройка** (Customize). Нажмите **Закрыть** (Close) снова, чтобы закрыть диалоговое окно **Настройка** (Customize).
9. В разделе **Отчеты** (Reports) откройте FlexibleSort в окне конструктора для того, чтобы вы могли прикрепить панель инструментов к отчету.
10. Щелкните правой кнопкой на селекторе отчета и выберите **Свойства** (Properties), чтобы открыть список свойств. Переключитесь на вкладку **Другие** (Other) и затем раскройте ниспадающий список свойства **Панель инструментов** (ToolBar), как показано на рис. 22.9. Выберите SortReport из двух доступных вариантов. Закройте список свойств. Закройте и сохраните отчет.
11. Снова откройте отчет FlexibleSort в режиме отчета. Перетащите панель инструментов сразу за строку меню отчета. Закройте и снова откройте отчет. Заметьте, что панель инструментов осталась на месте.
12. Щелкните правой кнопкой на строке заголовка отчета и выберите масштаб 75 % снова, чтобы получить лучший вид. Нажмите кнопку State/Date, чтобы отсортировать отчет, как показано на рис. 22.10.
13. Нажмите кнопку Date/State и посмотрите на изменения отчета снова.
14. Закройте отчет, не сохраняя его.

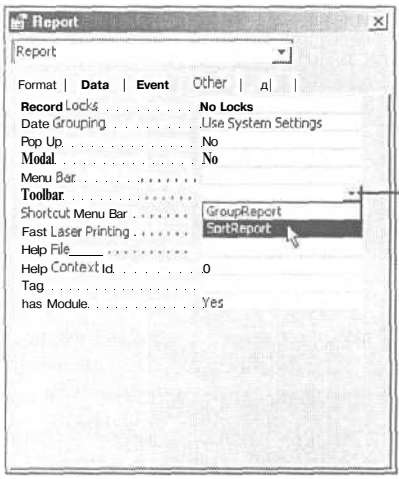


Рис. 22.9. Выбрав вкладку Другие списка свойств, выберите панель инструментов из ниспадающего списка свойства Панель инструментов (Toolbar) так, чтобы в открытом отчете появилась панель инструментов

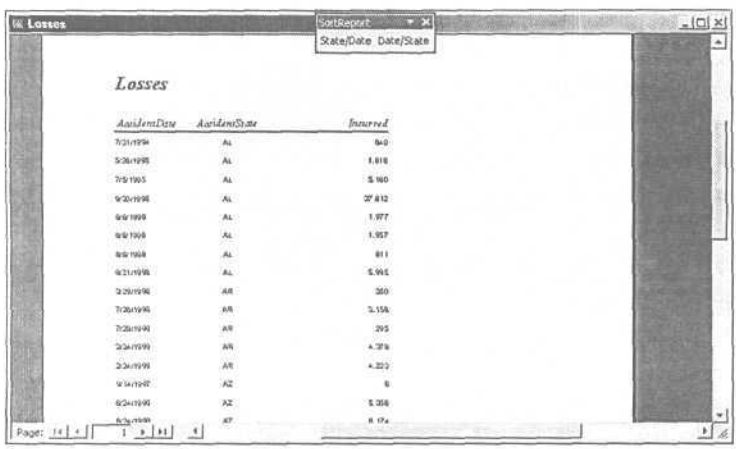


Рис. 22.10. Кнопка State/Date сортирует сначала по AccidentState, а затем по AccidentDate

Использование связей таблиц для включения или исключения целых разделов данных в отчете

Вы уже увидели, как связи могут обеспечивать эффективный способ просмотра данных в форме. Если у вас есть данные, которые должны быть введены в отчет с условиями в зависимости от значений полей, связи таблиц могут стать бесценным инструментом, который поможет вам достичь цели. Если вы думаете об одной стороне связи как о родительской таблице, а о многих других - как о дочерних, техника, которую вы увидите, переворачивает все с ног на голову. Родительская таблица функционирует как многосторонняя связь, а дочерние - как

односторонняя. Тем не менее цель одна и та же: минимизировать избыточные данные. Условная техника - это дополнительная польза, которая часто незаметна.

Перед вниканием в механизм этой техники покажем другую функцию, которая часто незаметна. Несмотря на то что некоторые могут поспорить с такой позицией, базы данных имеют определенное преимущество перед крупноформатными таблицами и даже текстовыми процессорами, когда дело доходит до предмашинных форматов (форма для ввода данных в ЭВМ). Ясно, что независимо от того, каким бизнесом вы занимаетесь, вы должны заполнять разные виды форм. Возможно, вы пишете заявление на кредитную карту или долгосрочную аренду апартаментов. Если вы хотите иметь возможность отслеживать эту информацию для сортировки, обработки и восстановительных целей, ваш ответ - это базы данных. Отчет не ограничен для вывода в предмашинный формат, так как он может напечатать всю форму, если необходимо.

Следующий пример - это недействительное страховое свидетельство. Он предназначен только для демонстрации. Тем не менее он показывает, как использовать базу данных для печати целой формы (не путайте с формой Access), включая строки, блоки и рисунки. Он также демонстрирует ранее упоминавшуюся технику связей.

Проделайте следующие шаги, чтобы узнать, как условные элементы управления могут работать с отчетом без какого-либо специального кода.

1. Откройте базу данных Certificate из папки AccessByExample.
2. В разделе **Формы** (Forms) откройте Liberty в режиме формы.
3. Нажмите кнопку View Cert, чтобы открыть сертификат в режиме предварительного просмотра (Print Preview), как показано на рис. 22.11.

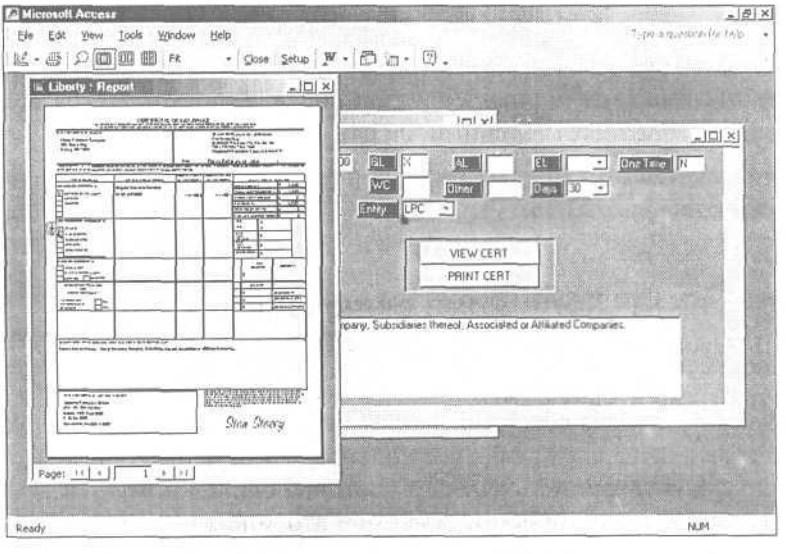


Рис. 22.11. Свидетельство выводит разделы согласно тому, что было введено в соответствующих элементах управления в форме Liberty

4. Разделы страхования этого отчета не должны быть связаны с текущими разделами отчета Access. Нажмите на раздел отчета Automobile Liability, в котором вы увидите курсор - увеличительное стекло на рис. 22.11, как показано на рис. 22.12.

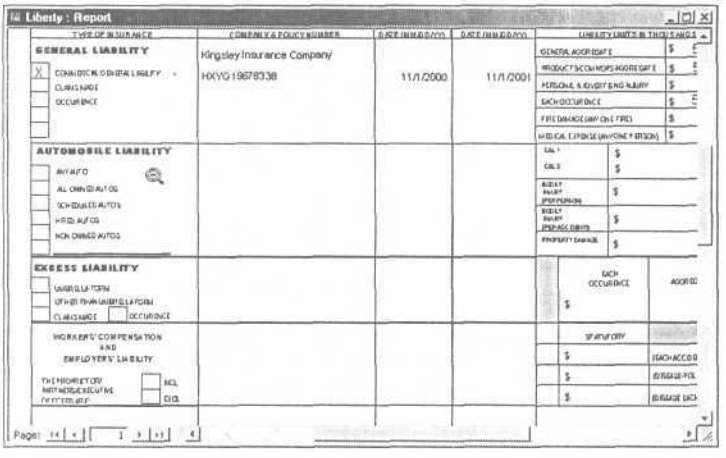


Рис. 22.12. Когда вы меняете масштаб, вы можете проверить, что раздел Automobile Liability пустой

Заметьте, что целый раздел Automobile Liability (AL) пустой. Все элементы управления в этом разделе отчета управляются элементом формы AL. Если в нем находится X, реляционный запрос (relational query) сопоставляет поле в таблице AL и собирает всю подходящую информацию для раздела Automobile Liability.

5. Закройте отчет.
6. Наберите X в поле AL. Нажмите кнопку View Cert снова. Заметьте, что раздел AL появляется со всеми соответствующими блоками.
7. Курсором-увеличительным стеклом нажмите на раздел AL, чтобы увеличить масштаб, как показано на рис. 22.13.
8. Закройте отчет.

Процедура кнопки View Cert очень проста. Вы отыскиваете номер текущего сертификата в поле [Cert No] и присваиваете его переменной. Затем присоединяете к аргументу WhereCondition метода OpenReport. Следующий код прикреплен к событию OnClick кнопки View Cert:

```
Dim cn As String
Cn = Me.Cert_No
Me.Requery
DoCmd.OpenReport "Liberty", acPreview,, "[Cert No]=" & cn
'Me.Requery
```

Рис. 22.13. Раздел AL вставлен в отчет, так как помечено поле AL формы Liberty

Обратите внимание на строки Requery в процедуре. Если вы удалите апостроф перед строкой Me.Requery после метода OpenReport и поместите апостроф перед строкой Me.Requery до метода OpenReport, то получите разные результаты. Отчет не обновляется для отображения изменений формы, по крайней мере не сразу. Этот пример демонстрирует важность порядка в процедуре.

Понимание запроса за отчетом

Теперь, когда вы ознакомились с техникой в действии, у вас есть возможность увидеть, как она работает. Если вы посмотрите на связи между таблицами в окне Relationships, вы обнаружите, что маленькие таблицы, как, например, GL и AL, находятся в связи на стороне многих, тогда как более большие таблицы-сертификаты находятся в связи на стороне одного. Маленькие таблицы объединены слева с таблицей-сертификатом, что является единственным способом работы этого способа.

Так как все связи - левые, то, если элемент управления - левый пустой, пустые записи все еще будут отыскиваться в запросе наряду с непустыми. Это означает, что одна запись сертификата может быть пустой или непустой с информацией из маленьких таблиц, которые связаны с ней, в зависимости от того, сопоставлены ли связанные поля или нет.

Все это взаимодействует с отчетом, который отображает пустые или непустые записи маленьких таблиц как пустые или непустые разделы в отчете.

Заметьте, что нет никаких изменений в дизайне отчета. Прелесть этого метода в том, что все делается интерактивно. Следующий пример продемонстрирует эту технику в работе.

1. Откройте запрос LinkReport в режиме конструктора (рис. 22.14).

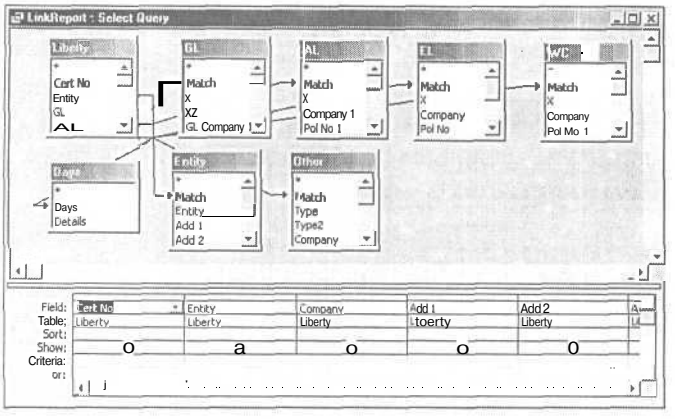


Рис. 22.14. Меньшие таблицы связаны левыми внешними связями, так что записи сопоставляются на соответствие

2. Наберите 1 в строке **Условие отбора** (Criteria) поля Cert No. Используя горизонтальную линейку прокрутки, найдите справа поле X таблицы GL. Наберите X в строке условия отбора поля X и нажмите кнопку **Запуск** (Run), чтобы запустить запрос.
3. Прокрутите вправо лист данных, используя курсор или линейку прокрутки, пока не увидите поле GL как показано на рис. 22.15. Заметьте, что информация GL заполнена.

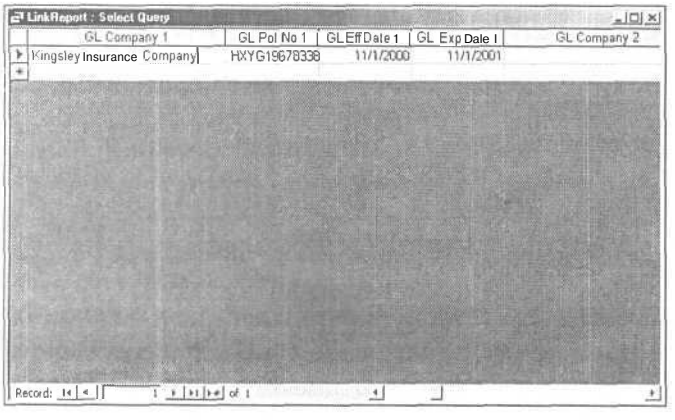


Рис. 22.15. Информация GL введена, так как X был помещен в поле X таблицы Liberty

Продолжайте прокручивать вправо, пока не увидите информацию AL. Заметьте, что она пустая. Если вы поместите X в соответствующее поле AL, эта информация будет введена, и т. д. В таблице AL есть поле с именем Match, которое содержит в себе X. Когда X помещается в поле AL таблицы Liberty, он отыскивает поле-пару в таблице AL.

ПРИМЕЧАНИЕ. На вопрос, почему не были использованы кнопки-флажки, ответ таков: кнопки-флажки могут быть либо включены, либо выключены (если вы не установили свойство Triple State как Yes). Некоторые ситуации требуют большего числа парных полей. Например, поле Entity может содержать любое подразделение компании. Если бы число подразделений было велико, программа могла бы представить каждое из них.

4. Нажмите **Вид** (View), чтобы переключиться в режим конструктора. Щелкните правой кнопкой по линии между полем GL в таблице Liberty и полем Match в таблице GL и выберите «Параметры объединения» (Join Properties), как показано на рис. 22.16. Откроется диалоговое окно «Параметры объединения» (Join Properties). Заметьте, что выбрана вторая опция. Эта техника работает только с левым или правым объединением, так как равные объединения сопоставляются только с записями, которые ищутся.

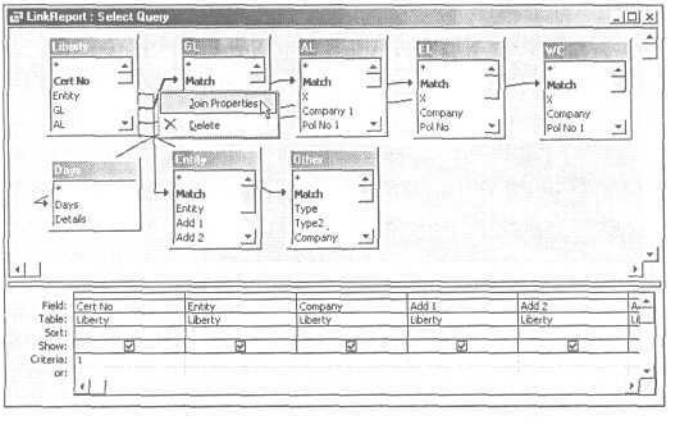


Рис. 22.16. Когда вы проверяете параметры объединения (Join) между таблицами Liberty и GL, вы обнаруживаете, что выбрана вторая опция, что делает ее левым объединением

5. Закройте диалоговое окно **Параметры объединения** (Join Properties) и запрос, не сохраняя его. Оставайтесь в базе данных.

Если вы работаете в отделе трудовых ресурсов большой фирмы, вы, возможно, думаете: «Чем эта техника полезна для меня? Я не связан со страхованием». Если вы хотите создать базу данных заявлений, вы можете включить раздел «Образование» в печатный текст, только если соответствующий образовательный уровень проверен. Если вы работаете в розничной торговле, вы можете включить только товарно-материальные ценности, которые находятся в резерве товарно-материального отчета. Используйте воображение, чтобы применить эту технику в вашем бизнесе или в домашних делах.

Что делать с надоедливыми пустыми строками

Допустим, что у вас есть три строки адресов, а нужны только две, но вы не хотите, чтобы в отчете, где должна быть адресная строка, появилась некрасивая дыра. Хотя Access и предусмотрел эту общую проблему, она оборачивается классической ситуацией хорошие новости/плохие новости. Хорошая новость заключается в том, что Access имеет свойство, которое управляет пустыми строками в отчете. Свойство CanShrink может передвигать элементы управления под пустые элементы управления, так что пустые линии будут «затерты». Плохая новость состоит в том, что это свойство не работает, если прикреплена метка или рис. за элементом управления.

Решение CanShrink для пустых строк

Тем не менее, если у вас нет меток или рисунков в той части отчета, в которой надо удалить пустые строки, используйте свойство CanShrink. Когда вы устанавливаете свойство CanShrink элемента управления как Yes, строка удаляется (подавляется), когда элемент пустой. Используя свойство CanShrink, помните следующее:

- Установки свойств не влияют на горизонтальные интервалы между элементами управления. Они влияют только на вертикальные интервалы, занимаемые элементами.
- Совмещение элементов управления не приводит к сжиманию.
- Высота большого элемента управления может сдержать «сжимание» контроллеров за ним. Например, если у вас есть четыре коротких текстовых блока со свойством CanShrink, установленным на Yes, слева от раздела детализации, и один высокий справа от него (который почти равен совмещенным четырем маленьким блокам), текстовые блоки слева не сожмутся.

Другое решение для пустых линий

Представьте, что у вас есть рис., например файл .pcx, справа от контроллеров (элементов управления), имеющих пустые строки, которые надо удалить (подавить). Этот рис. предохраняет контроллеры слева от удаления линий, так как он делает недействительным (отменяет) свойство CanShrink контроллера. Следующее выражение, введенное в свойство Control Source контроллера, выполняет операцию «подавления» строк:

```
=IIf(Not IsNull([Liberty.add 1]) And IsNull([Liberty.add 2]),[City]
[ic:ccc]& ", " & [State] & " " & [Zip],[Liberty.add 2])
```

Если контроллер над текущим контроллером в этом выражении не Null, а текущий ([Liberty.add_2]) - Null, выражение передвигает City, State и Zip до текущего контроллера.

Если [Liberty.add 2] не Null, используйте это.

Чтобы посмотреть на эту технику в действии, проделайте следующие шаги:

1. В разделе **Отчеты** (Reports) откройте Liberty в режиме конструктора, чтобы просмотреть макет отчета, который печатает сертификат.
2. Увеличьте окно отчета и прокрутите вниз до конца отчета, как показано на рис. 22.17.

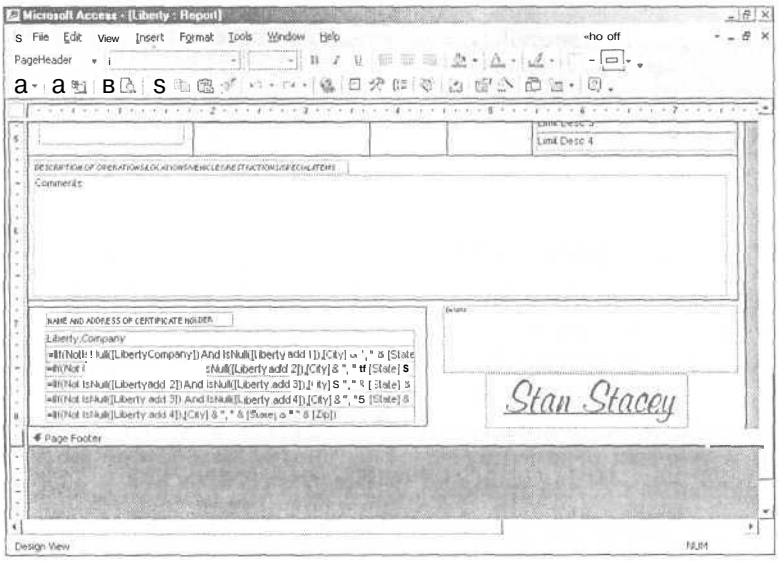


Рис. 22.17. Контроллеры слева все еще могут «подавлять» пустые строки выражениями, прикрепленными к источнику управления каждого контроллера

Обратите внимание на сигнатуру, являющуюся .pcx-файлом, справа от адресного контроллера. Помещение этого рисунка отменяет любую из операций CanShrink адресного контроллера слева. Вы, возможно, также оставили значения по умолчанию No свойства CanShrink каждого контроллера.

Также обратите внимание на то, как осуществляется ссылка на некоторые контроллеры в адресном контроллере. Например, на поле [add 1] ссылаются не как на [add 1], а как на [Liberty.add 1]. Вот почему в отчете больше одного поля с таким именем. Имя таблицы перед именем поля отличает его от любого другого поля с таким же именем. Вы можете избежать этого, присвоив это имя полю в памяти, но иногда повторяющихся имен полей сложно избежать, особенно в базах данных с большим числом таблиц.

3. Нажмите на кнопку **Предварительный просмотр** (Print Preview) панели инструментов. Курсором-увеличительным стеклом нажмите на участок отчета, содержащий строки адресов, чтобы увеличить эту часть отчета, как показано на рис. 22.18.

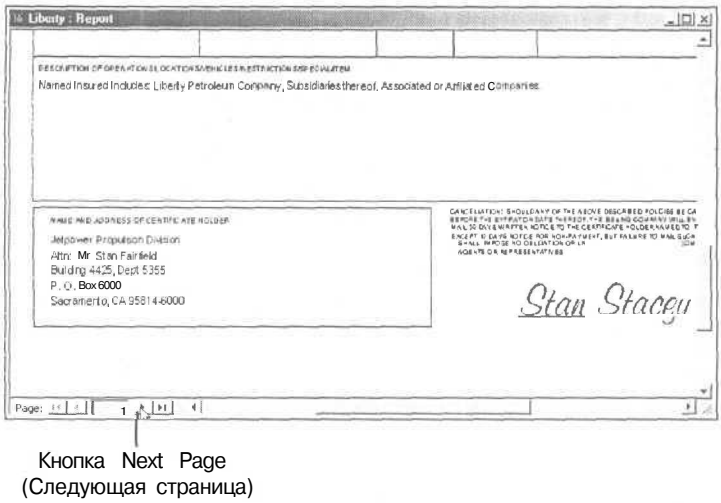


Рис. 22.18. Нажимайте кнопку Следующая страница (Next Page) в самом низу отчета, чтобы пролистать отчет

4. Нажмите кнопку **Следующая страница** (Next Page) несколько раз и обратите внимание, как «подавлены» пустые строки.
5. Закройте отчет, но оставайтесь в базе данных.

Многостолбцовые отчеты

Обычное применение многостолбцовых отчетов - это почтовые наклейки. Хотя процедура для установки наклеек довольно простая, ее реализация может оказаться довольно сложной. В следующем примере настройки вашего принтера могут немного отличаться, так как драйвер принтера, который вы используете, скорее всего отличается от драйвера в этом примере. По этой причине для эксперимента советуем получить подходящие настройки вашего принтера. Тем не менее вы можете использовать пример как указание к действию.

Проделайте следующие шаги, чтобы создать почтовые наклейки.

1. В разделе **Отчеты** (Reports) нажмите **Создать** (New), чтобы открыть диалоговое окно **Новый отчет** (New Report). Выберите мастер **Почтовые наклейки** (Label Wizard) и таблицу Liberty из ниспадающего окна. Нажмите OK, чтобы открыть мастер почтовых наклеек (Label Wizard), как показано на рис. 22.19.
2. Выберите производителя Avery, а код товара (Product number) - 8196, как показано на рис. 22.19, и нажмите **Далее** (Next), чтобы открыть следующую страницу мастера.
3. На второй странице мастера выберите шрифт и цвет (рис. 22.20).
4. Нажмите **Далее** (Next), чтобы выбрать значения по умолчанию и открыть страницу, показанную на рис. 22.21.

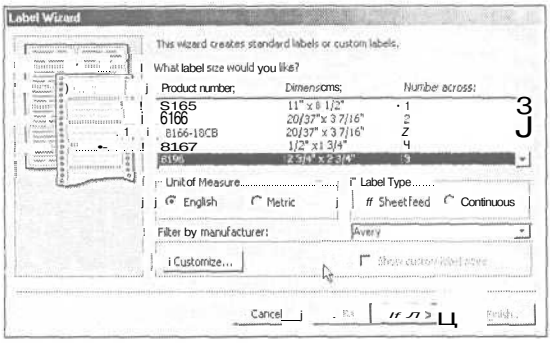


Рис. 22.19. Мастер наклеек (Label Wizard) дает вам возможность устанавливать спецификации для подбора различных производителей наклеек

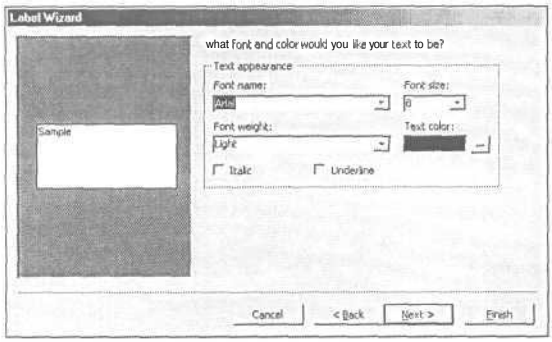


Рис. 22.20. Вы выбираете название шрифта, его размер, начертание и цвет на второй странице Label Wizard

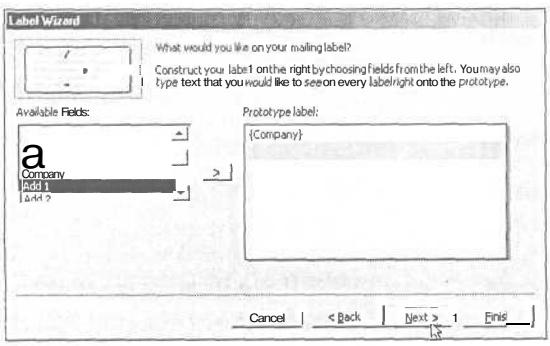


Рис. 22.21. Выберите нужное поле и нажмите Enter, чтобы убедиться, что каждое поле находится на отдельной строке

5. Выберите поле Company двойным щелчком и нажмите Enter так, чтобы следующее поле добавилось на следующую строку. Выберите Add1 и нажмите Enter. Проделайте ту же процедуру, что и для Add1, для Add 2 - Add 4.

6. Выберите поле City и поставьте запятую и пробел. Выберите поле State и нажмите пробел дважды. Выберите поле Zip, как показано на рис. 22.22, и нажмите Next, чтобы открыть следующую страницу мастера.

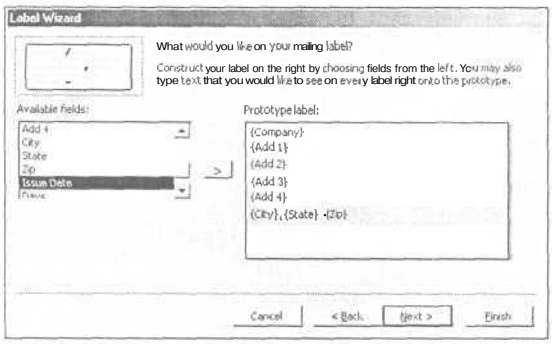


Рис. 22.22. Поля City, State и Zip должны быть на одной и той же строке, отделенные разделителями полей

7. Дважды щелкните на Company, чтобы выбрать его как сортирующее поле, и нажмите **Далее** (Next), чтобы открыть следующую страницу мастера, показанную на рис. 23.23.

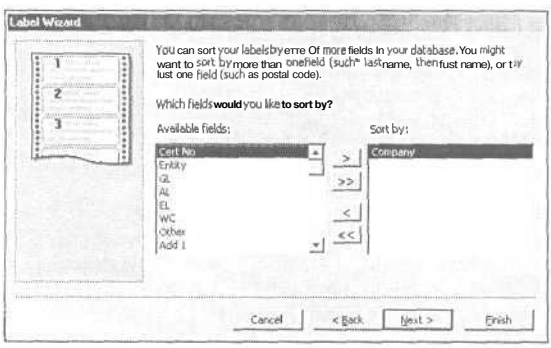


Рис. 22.23. Наклейки сортируются по полю, которое вы выбираете на четвертой странице мастера

8. Нажмите **Готово** (Finish), чтобы предварительно просмотреть отчет (рис. 22.24).
9. В зависимости от драйвера вашего принтера вы можете получить сообщение о том, что ширина страницы недостаточна. Нажмите OK. Выйдите из отчета и оставайтесь в базе данных.

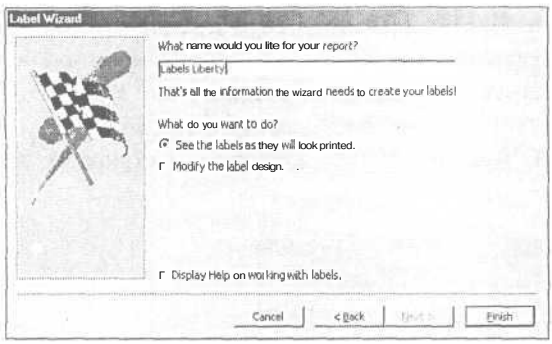


Рис. 22.24. На последней странице мастера наклеек (Label Wizard) присвойте имя отчету и определитете, в каком окне открыть отчет: Print Preview (Предварительный просмотр) или Design (Конструктор)

Многостолбцовые отчеты

Почтовые марки - это не единственное применение отчетов, использующих составные столбцы. Например, вы можете напечатать модернизированный отчет телефонной книги, используя более одного столбца. Место для создания отчета со сложными столбцами не там, где вы, возможно, думаете. Следующие шаги проведут вас через наладку отчета с более чем одним столбцом.

1. Оставив отчет в режиме предварительного просмотра, выберите **Файл** (File), **Параметры страницы** (Page Setup) из строки меню.
2. Переключитесь на вкладку **Столбцы** (Columns), показанную на рис. 22.25. Заметьте, что здесь вы можете изменить число столбцов поперек страницы.

Рис. 22.25. Диалоговое окно Параметры страницы (Page Setup) дает возможность настраивать установки для отчетов с составными столбцами

3. Измените **Размер столбца** (Column Size) с 2.75 до 2.25. Переключитесь на вкладку **Поля** (Margins). Измените установку левого на 0.75. Нажмите OK, чтобы подтвердить настройки. Заметьте, что разбивка стала лучше (в зависимости от вашего принтера) и вы больше не получаете предупреждающего сообщения. Ваш отчет должен быть похож на тот, который изображен на рис. 22.26.

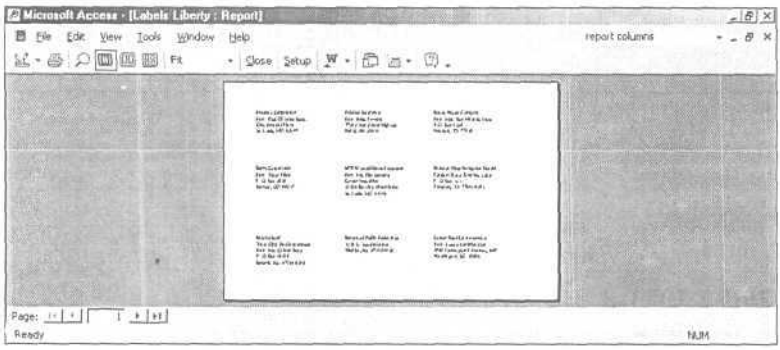


Рис. 22.26. Отчет с наклейками выглядит лучше после проделанных *в* Параметрах страницы (Раде Setup) регулировок

4. Нажмите на кнопку **Вид** (View), чтобы переключиться в режим конструктора. Нажав на верхний край раздела **Нижний колонтитул** (Page Footer), перетащите его вверх для уменьшения размера области данных (Detail), как показано на рис. 22.27.

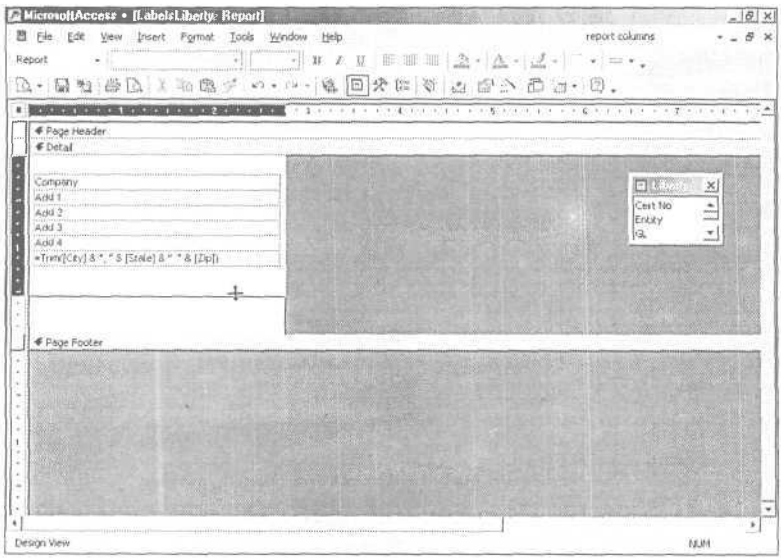


Рис. 22.27. Отчет с наклейками выглядит лучше после проделанных в Параметрах страницы (Раде Setup) регулировок

5. Нажмите на кнопку **Предварительный просмотр** (Print Preview) и заметьте, что на границе помещается больше ярлыков. Вам, возможно, придется произвести некоторые из тех настроек, чтобы сделать строку ярлыков подходящей для вашего конкретного принтера, но даже если не придется, вы знаете, как это сделать.

Печать дополнительных копий

Могут быть случаи, когда вам понадобится напечатать копии конкретных записей в вашей форме. Следующий код, прикрепленный к событию OnClick командной кнопки на форме, выполняет эту задачу:

```
RunCommand acCmdSelectRecord
DoCmd.PrintOut acSelection,,,, 3
```

Эта функция использует метод RunCommand с собственной константой acCmdSelectRecord для выполнения команды Select Record из меню Edit строки меню, которая выбирает текущую запись. Затем метод PrintOut объекта DoCmd с аргументом PrintRange_установленным на acSelection, печатает выбранную запись. Пятый аргумент метода Printout указывает принтеру напечатать три копии. Вы также можете использовать метод Printout для печати группы страниц. Так как вы печатаете из формы вместо отчета, поместите обрыв страницы (принудительный) на форме так, чтобы запись печаталась как страница.

Что дальше?

Эта глава сосредоточила ваше внимание на способах улучшения отчета. Следующая глава обсуждает некоторые продвинутые вопросы, например репликацию (средства копирования). Вы познакомитесь с новым акронимумом - GUID (Globally Unique Identifier, глобальный уникальный идентификатор).

23
Использование репликаций

В прошлой главе вы узнали, как эффективнее использовать отчеты. В этой главе речь идет о репликациях. Как вы уже обнаружили, базы данных часто имеют дело с перенаправлением данных из множества источников в одно центральное место. Сети, в том числе Интернет, разрешили многие спорные вопросы касательно интеграции данных из различных источников. Тем не менее пользователи не всегда подключены к сети, когда они вводят информацию. Репликации зачастую могут предоставить решение дилемм распределенной обработки данных. В частности, в этой главе вы узнаете следующее:

- почему используются репликации баз данных,
- когда не следует применять репликации баз данных,
- как работают репликации,
- как получить список пользователей в текущей базе данных.

Репликация базы данных

Репликация базы данных - это процесс копирования базы данных особым образом так, что две или более копии могут обмениваться обновленными данными. Этот обмен между базами данных называется *синхронизацией.* Эта операция называется обменом, потому что синхронизация обычно происходит в обоих направлениях, хотя вы можете выбрать синхронизацию в одном направлении, если вы так желаете. Каждая копия базы данных называется *репликой,* которая содержит обычный набор объектов базы данных.

База данных, с которой вы начинаете, преобразуется в мастер-реплику (основную реплику, Design Master). Каждая реплика является частью *набора реплик,* который содержит мастер-реплику и другие реплики. *Мастер-реплика* - единственная реплика, которая позволяет вам производить изменения в структуре базы данных. Все реплики, которые принадлежат к одному и тому же набору реплик, могут синхронизироваться друг с другом.

Почему используют репликации баз данных?

Представьте, что у вашей компании три торговых представителя (с именами Роджер, Элис и Майк), курирующих три различные территории. Все трое торговых представителя имеют переносные компьютеры, которые они берут с собой в дорогу с копией главной базы данных из главного офиса. Так как Роджер получил клиентуру и заказы, он добавил их в свою базу данных. Элис потеряла несколько клиентов, поэтому она удалила их из своей базы данных. Некоторые клиенты Майка сменили адреса, которые он сразу обновил в своей базе данных.

Тина отвечает за обновление базы данных. Можете себе представить разочарование Тины при виде стоящей перед ней задачи, когда торговые представите-

ли возвращаются в центральный офис? Во-первых, ей нужно добавить новых клиентов Роджера. Даже если она копирует из базы данных Роджера, его копия содержит его добавления, однако не содержит удаления, произведенные Элис, и обновления, произведенные Майком. Поэтому, ей придется затем искать и удалять клиентов Элис. Чтобы подлить масла в огонь, положим, что Тина сделала сама некоторые обновления. Ее обновления, наряду с изменениями, внесенными остальными, таюке должны быть включены во все копии базы данных.

Тина находится на встрече с боссом для того, чтобы обсудить возможные решения. Торговые представители могут заняться сетью главного офиса со службой удаленного доступа. Это вызовет проведение новых телефонных линий, неучитываемое бюджетом. Президент компании не склонен к установке базы данных Интернета по причинам безопасности, и внутренняя сеть может никогда не возникнуть.

Наконец, входит сетевой администратор и объясняет преимущества репликаций базы данных. Теперь Тине вовсе не обязательно вручную что-то править. После того как базы данных скопированы на сервер, она просто нажимает нужные кнопки - и все добавления, удаления и обновления немедленно появляются в главной базе данных и во всех ее репликах.

Другие причины для использования репликаций

Только что представленный сценарий не единственная причина для использования репликаций. Другие причины состоят в следующем.

- **Распределение обновлений.** Разработчики могут добавлять новые формы и отчеты в мастер-реплику и распространять изменения автоматически через репликации. Это означает, что вы можете дублировать больше, чем просто данные. Вы не можете корректировать формы или отчеты в реплике, как это можно делать с мастер-репликой. Это также предоставляет разработчикам встроенную защиту их приложений.
- **Синхронизация офиса и дома.** Пользователь, который переносит базу данных офиса между офисом и домом, может легко синхронизировать изменения, произведенные дома, с базой данных в офисе.
- **Создание резервной копии базы данных.** Вместо того чтобы создавать резервную копию всей базы данных, эффективнее использовать репликации для этой цели, потому что она копирует только произведенные изменения.
- **Как избежать влияния сетевого трафика.** Если связь по сети с одной из баз данных плохая из-за повышенного трафика, репликации предоставляют другую альтернативу для расщепленного доступа к базе данных. Пользователи могут делать обновления данных на их локальных компьютерах, а не на сервере и в конце дня синхронизировать их с мастер-репликой.

Когда не следует использовать репликации

Хотя репликация базы данных может быть решением многих проблем, с которыми вы сталкиваетесь при обработке распределенной базы данных, вы должны уметь распознать ситуации, которые не подходят для применения репликаций. Вы могли бы обдумать заново использование репликаций, если возникают следующие ситуации:

- **Многочисленные реплики и большие записи.** Приложения, которые делают необходимым частое обновление большого числа существующих записей в многочисленных репликах, имеют больше конфликтов записей, чем приложения, которые просто вставляют добавленные записи в таблицы в базе данных. То, что конфликты должны быть разрешены вручную, влечет за собой больше административного времени. Следовательно, скорее всего это не самая лучшая идея - использовать репликации в этой ситуации.
- **Приложения, требовательные ко времени.** Приложения, которые основываются на такой требующей непрерывной точности информации, как бронирование туристических путевок, движение средств между банками, слежение за грузоперевозками, - обычно не самые лучшие кандидаты для использования репликаций. Существование времени задержки между временем изменения данных и временем синхронизации данных предполагает, что дублированные базы данных более подходят для приложений, которые не требуют, чтобы данные были постоянно новыми. Если возможно, что два пользователя будут корректировать данные в одной и той же записи в течение одного промежутка времени, неизбежные конфликты должны быть разрешены, что говорит о том, что, вероятно, использование репликаций не самое лучшее намерение. С другой стороны, если каждый пользователь несет полную ответственность за обновление определенного набора записей, шансы получить конфликты уменьшаются, особенно в отключенных от сети базах данных. В этом случае репликации могут быть обоснованными.

Как работают репликации

С первого взгляда репликация выглядит просто. Вы просто используете строку меню Access и по истечении нескольких секунд у вас есть мастер-реплика с несколькими репликами. Однако первые впечатления могут быть обманчивыми. Можно легко написать целую книгу хотя бы об одном из аспектов репликаций. Это не значит, что для того, чтобы было возможно эффективное использование репликаций, требуется знать о них все.

Для того чтобы вести учет сложностям контроля изменений в базе данных, много действий происходит за пределами видимости. Когда вы создаете реплики, Access создает несколько новых скрытых таблиц в базе данных. Кроме добавления таблиц, Access добавляет поля в существующие таблицы для управления синхронизацией и разрешения конфликтов. Табл. 23.1 показывает поля, которые могут быть добавлены для того, чтобы сделать таблицу способной к репликации. Хотя добавлены всего четыре поля, вы не можете просмотреть их

без проверки System Objects (Объектов системы) в Tools (Инструменты), Options (Опции), View (Вид).

Таблица 23.1. Поля репликации

Поле	Описание
s_Generation	Определяет, были ли произведены изменения со времени последней синхронизации
s_GUID	GUID означает глобальный уникальный идентефикатор. Он однозначно определяет каждую запись. Из-за того что поля AutoNumber могут быть ненадежными для приведения в соответствие записей при синхронизации, Access использует GUID вместо AutoNumber
s_Lineage	Прослеживает историю изменений в записи и решает конфликты при изменении отдельной записи в составных репликах
s_ColLineage	Определяет победителя в конфликте между столбцами
Gen_FieldName	Добавлен в каждое большое объектное поле (OLE или Memo) в таблице, где FieldName представляет имя большого объектного поля. Это поле всего лишь прослеживает изменения в соответствующем большом объектном поле

Конфликты полей AutoNumber

Поля AutoNumber разработаны для увеличения на единицу каждый раз, когда вы добавляете запись. Этого не происходит в среде репликации. Если бы это происходило, каждая из набора реплик добавляла бы записи независимо друг от друга с одинаковыми значениями первичных ключей, приводя к конфликтам при каждой синхронизации. Поэтому, когда вы создаете реплику из базы данных, ей навязывается произвольное увеличение (между -2,000,000,000 и 2,000,000,000), что значительно сокращает вероятность присваивания одного и того же значения первичного ключа двумя разными репликами. Если вы используете AutoNumber в целях как уникальности, так и последовательной нумерации, вам, возможно, потребуется рассмотреть другие альтернативы перед дублированием вашей базы данных. Хотя вы и можете использовать поле GUID как поле первичного ключа, вообще это не рекомендуется.

Сжатие перед дублированием

Перед репликацией или синхронизацией дважды сожмите мастер-реплику для достижения лучших результатов. Первое сжатие помечает дублированные объекты, которые требуется удалить, хотя на самом деле оно не удаляет их. Второе сжатие удаляет помеченные объекты и освобождает место, соответствовавшее удаленным объектам. Вообще это существенно только в мастер-реплике. Обычные реплики должны быть сжаты только один раз.

Если вы сжимаете испорченную реплику, она потеряет свой статус реплики. Это значит, что, если это мастер-реплика, она перестанет быть мастер-репликой. Нормальные испорченные реплики получают нормальный, нереплицированный статус базы данных. Однако все скрытые системные таблицы и поля все же присутствуют. Необычное поведение в базе данных показывает, что база данных испорчена.

Преобразование базы данных в мастер-реплику

Следующий пример показывает, как легко дублировать базу данных. Вы создаете мастер-реплику вместе с двумя репликами. После произведения изменений вы синхронизируете базы данных и разрешаете конфликты.

Следующие шаги показывают вам процесс репликации.

1. Откройте базу данных Master Sales в папке AccessByExample.
2. Выберите **Сервис** (Tools), **Служебные программы** (Database Utilities), **Сжать и восстановить базу данных** (Compact and Repair Database) для сжатия и восстановления базы данных; затем повторите эти действия для того, чтобы второй раз сжать и восстановить базу данных.
3. Выберите **Сервис** (Tools), **Репликация** (Replication), **Создать реплику** (Create Replica) в строке меню для того, чтобы начать процесс репликации. На запрос закрыть базу данных и создать реплику ответьте Да (Yes). На запрос сохранения базы данных ответьте Да для того, чтобы открыть диалоговое окно **Размещение новой реплики** (Location of the New Replica), показанное на рис. 23.1.

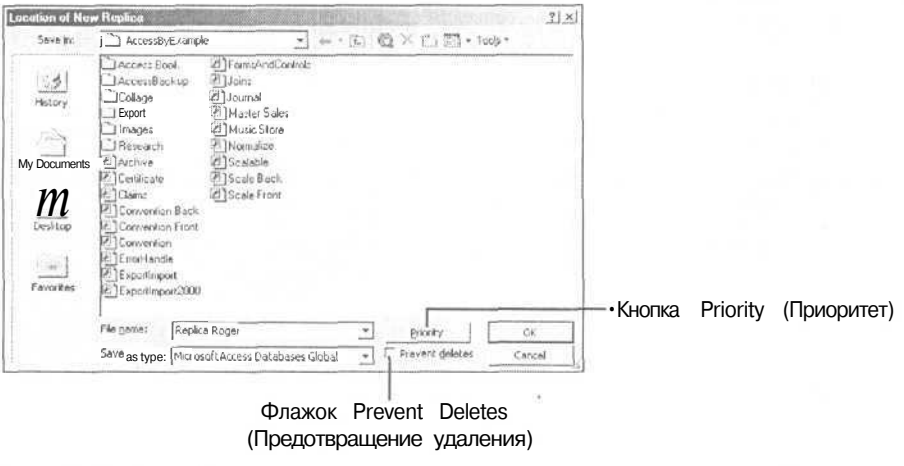


Рис. 23.1. Вы вводите местоположение реплики в диалоговом окне **Размещение новой реплики** (Location of the New Replica)

Обратите внимание на окно **Запретить удаление** (Prevent Deletes) в диалоговом окне **Размещение новой реплики** (Location of the New Replica). Если вас

беспокоит, что пользователь может удалить запись, которая важна для другого пользователя, поставьте там галочку. Если в такой проверке нет нужды, оставьте окно пустым.

Также обратите внимание на кнопку **Приоритет** (Priority) в диалоговом окне **Размещение новой реплики** (Location of the New Replica). Репликам при их создании присваиваются значения приоритетов, выраженные числами между О и 100. Например, когда база данных создается с возможностью для ее репликации, приоритет мастера-реплики устанавливается равным 90. В случае конфликта синхронизации побеждает реплика с большим значением приоритета.

4. Убедитесь, что выбрана папка AccessByExample и назовите только что созданную базу данных Replica Roger, как показано на рис. 23.1, и нажмите ОК для того, чтобы создать новую реплику. Когда в следующем сообщении вам скажут, что структура базы данных может быть сделана только в мастер-реплике, нажмите OK.
5. Повторите шаги 3 и 4, но на 4-м шаге назовите базу данных Replica_Alice. Нажмите Да для того, чтобы закрыть и открыть заново базу данных. Обратите внимание на новый вид вашей базы данных Master Sales. В частности, каждый объект в окне базы данных имеет картинку репликации рядом с иконкой сразу после имени объекта.
6. Откройте базу данных Replica Roger из папки AccessByExample. Нажмите **Формы** (Forms) под «Объектами» (Objects). Обратите внимание на то, что кнопка **Создать** (New) отключена. Также обратите внимание на то, что как кнопка «Создать форму при помощи конструктора» (Create Form in Design View), так и кнопка «Создать форму с использованием мастера» (Create Form By Using Wizard) отсутствуют, как показано на рис. 23.2.

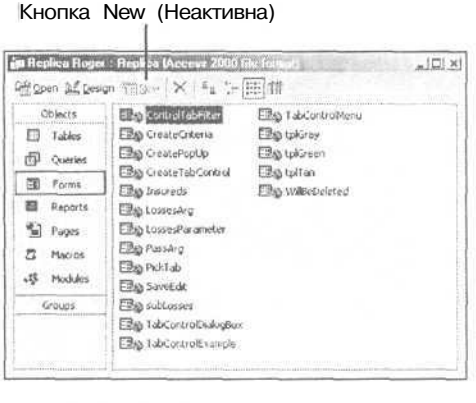


Рис. 23.2. База данных реплик не позволяет вам создавать новую форму

ЗАМЕЧАНИЕ. Отметьте также, что вы не можете изменять дизайн запросов, таблиц, форм и сообщений. Ни макросы, ни модули не могут быть созданы в репликах.

7. Выберите **Запросы** (Queries) ниже **Объектов** (Objects). Щелкните два раза на запрос AddToReplica. Нажмите Да, когда сообщение предупредит вас, что данные будут изменены в вашей таблице. Снова нажмите Да, когда получите сообщение о том, что вы собираетесь добавить записи. Запрос добавляет две записи в таблицу LossesIndex, так что у вас есть над чем проводить репликацию.

ПОДСКАЗКА. Добавляя таблицу с полем autonumber (счетчик), не включайте это поле в запрос. Автонумерация берет на себя заботу о нумерации вместо вас.

8. Выберите **Таблицы** (Tables) под **Объектами** (Objects) и откройте таблицу LossesIndex. Две записи были добавлены. Если вы не видите добавленные записи перед Ш 1 в таблице LossesIndex, как показано на рис. 23.3, нажмите Ctrl+End для того, чтобы перейти к последней записи. Если это положительные числа, то они добавляются в конец таблицы. В противном случае вы можете увидеть новые записи, когда вы открываете таблицу. Не беспокойтесь о том, какие числа отображает поле Autonumber (счетчик) - положительные или отрицательные.

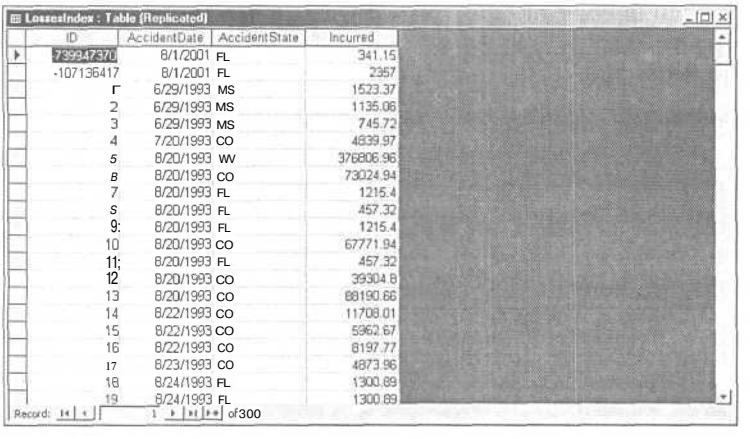


Рис. 23.3. Поле-счетчик (Autonumber) в реплике меняется на добавлением произвольного числа после репликации

9. Выберите в меню **Сервис** (Tools), **Служебные программы** (Database Utilities), **Сжать и восстановить базу данных** (Compact and Repair Database) для того, чтобы сжать базу данных. Выберите **Сервис** (Tools), **Репликация** (Replication), **Синхронизация** (Synchronize Now) для того, чтобы открыть диалоговое окно **Синхронизация базы данных** (рис. 23.4).

ЗАМЕЧАНИЕ. Когда лучше всего производить репликацию? После того, как были сделаны изменения в любой из реплицированных баз данных. В действительности из-за логических рассуждений вы, скорее всего, не всегда сможете делать репликацию сразу же после того, как были произведены изменения.

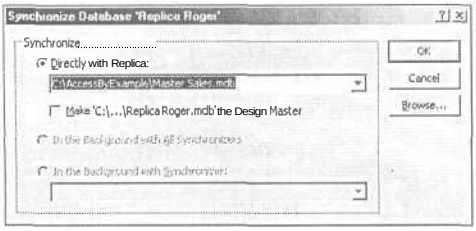


Рис. 23.4. Ниспадающее окно в диалоговом окне «Синхронизация базы данных» показывает вам базы данных, которые можно реплицировать

10. Выберите реплику Replica Alice.mdb в ниспадающем окне и затем нажмите (Ж для синхронизации баз данных. Если реплика Replica_Alice не появляется в ниспадающем окне, нажмите кнопку **Обзор** и затем выберите реплику Alice в папке AccessByExample. Нажмите Да, когда появится сообщение о том, что база данных должна быть закрыта для синхронизации с репликой. Вас проинформировали об успешной синхронизации.

Следующим шагом после того, как вы синхронизировали базы данных, является получение подтверждения, что произошла удачная репликация. Хотя вас проинформировали, что репликация успешно завершилась, вам важно видеть результаты синхронизации. Проводится большее число синхронизации, обновлений и проверок с целью представления вам вида того, что в действительности происходит, когда базы данных синхронизируются. Вы также обнаруживаете, что происходит, когда выявляются конфликты. Процесс проверки разбивается на следующие этапы:

1. Откройте реплику Replica Alice и таблицу LossesIndex для подтверждения того, что записи были добавлены. Измените параметр MS на WY в поле AccidentState из ID 1. Закройте таблицу и выберите **Сервис** (Tools), **Служебные программы** (Database Utilities), **Сжать и восстановить базу данных** (Compact and Repair Database) из строки меню для сжатия базы данных.
2. Откройте реплику Replica Roger в проводнике Windows в папке AccessByExample. У вас теперь должно быть открыто две базы данных Access, если вы посмотрите на вашу панель задач. В противном случае откройте заново реплику Replica Alice. Откройте таблицу LossesIndex и измените поле AccidentState из первой записи с MS на CO. Закройте таблицу и произведите сжатие базы данных.
3. В базе данных Replica Roger выберите **Сервис** (Tools), **Репликация** (Replication), **Синхронизация** (Synchronize Now) в строке меню для того, чтобы открыть диалоговое окно **Синхронизации базы данных.**
4. Убедитесь, что выбрана база данных Replica Alice.mdb и затем нажмите OK для запуска синхронизации. Нажмите Да, когда увидите сообщение, уведомляющее вас о том, что база данных должна быть закрыта для проведения синхронизации с репликой. Сообщение означает, что синхронизация прошла

успешно. После сообщения об успешной синхронизации появляется другое сообщение, как показано на рис. 23.5.

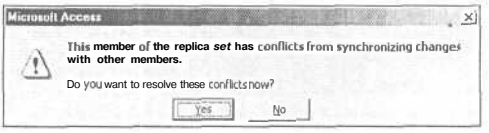


Рис. 23.5. Выберите Да при запросе «Хотите ли вы разрешить конфликты сейчас?»

5. Нажмите Да для того, чтобы открыть диалоговое окно **Просмотр конфликтов** (Conflict Veiwer), показанное на рис. 23.6.

ЗАМЕЧАНИЕ. Если сообщение появляется, но мастер просмотра конфликтов не открывается, у вас, возможно, отсутствует на жестком диске файл мастера wzcnflct.exe. Это, вероятно, означает, что вам нужно переустановить Access 2002.

Рис. 23.6. Просмотрщик конфликтов автоматически загружается, когда дан положительный ответ на вопрос «Хотите ли вы разрешить конфликты сейчас?»

6. Нажмите **Просмотр** (View) для того, чтобы открыть следующую страницу мастера просмотра конфликтов, как показано на рис. 23.7.

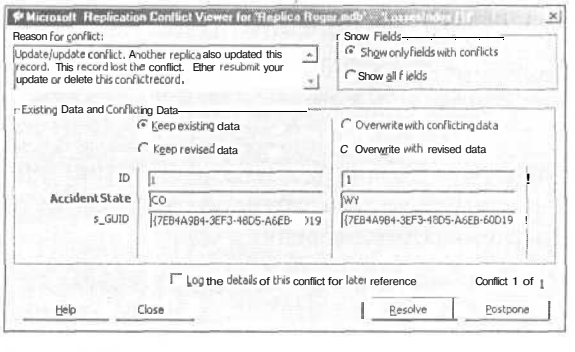


Рис. 23.7. Следующая страница мастера просмотра конфликтов позволяет вам определить лучший метод для разрешения конфликтов синхронизации

7. Выберите кнопку **Переписать с исправленными данными** (Overwrite with Revised Data) и затем нажмите **Разрешить** (Resolve) для того, чтобы разрешить конфликты в базе данных Replica Roger. Вы возвращаетесь на первую страницу мастера просмотра конфликтов, получая сообщение <нет конфликтных таблицX Табл. 23.2 показывает доступные просмотрщику конфликтов (Conflict Veiwer) опции.

Таблица 23.2. Опции просмотрщика конфликтов(Conflict Veiwer)

Опция	Описание
Причина конфликта	Показывает характерную причину возникновения конфликта и способ, каким был разрешен этот конфликт
Показать только поля с конфликтами	Показывает колонку GUID и только те поля, в которых есть конфликты
Показать все поля	Показывает все поля в записи с конфликтами
Принять существующие данные	Принимает текущее разрешение конфликта
Принять данные исправлений	Принимает текущее разрешение конфликта, но включая исправления, сделанные в окнах редактирования
Принять с удалением данных	Принимает текущее разрешение конфликта, данные будут удалены
Игнорировать конфликт	Отмечает конфликт как разрешенный без каких-либо исправлений. Из-за того, что результаты в конфликте не разрешены, должны быть как можно раньше приняты другие меры
Переписать с конфликтными данными	Отвергает способ разрешения конфликта и оставить видимыми конфликтующие данные
Переписать с текущими данными	Отвергает способ разрешения конфликта и оставляет видимыми существующие данные
Переписать с данными исправлений	Отвергает способ разрешения конфликта с исправлениями, сделанными в окнах редактирования, и оставляет видимыми данные исправлений
Удалить данные	Удаляет данные, показанные в полях
Сохранить детали конфликта для последующего обращения к ним	Делает копию конфликта и сохраняет ее в отдельном файле-журнале (лог-файл) для дальнейшего использования. Файл сохраняется в папке *Windows\profiles\user*Application Data\Microsoft\Database Replication\UnresolvedConflicts.log

8. Закройте и откройте заново реплику Replica Alice. Обратите внимание на то, что конфликтное сообщение автоматически снова возникает. Повторите действия, описанные в пп. 5-7 для разрешения конфликтов в реплике Replica Alice.
9. Откройте базу данных Sales Master (мастер-реплика) для синхронизации всех трех баз данных после внесения изменений в базу данных. Откройте таблицу LossesIndex. Обратите внимание на то, что две записи не были добавлены.

Измените значение Incurred в первой записи с 1523.37 на 1524.37. Закройте таблицу LossesIndex и дважды произведите сжатие базы данных.

10. Синхронизируйте основную базу данных с репликами Replica Roger и Replica Alice, выбрав **Сервис** (Tools), «Репликация» (Replication), «Синхронизация» (Synchronize Now) и используя ниспадающее окно для выбора соответствующей таблицы. Откройте таблицу Losseslndex и обратите внимание на то, что были добавлены недостающие две записи. Откройте реплики Replica Roger и Replica Alice и заметьте, что значение в поле Incurred в записи ГО 1 было изменено с 1523.37 на 1524.37. Это доказывает, что синхронизация прошла в обоих направлениях.
11. Закройте таблицу и останьтесь в базе данных.

Теперь вы имеете лучшее представление о том, что происходит во время репликации. Вы увидели, что репликация действительно осуществляется в обоих направлениях. Если вы изменяете одно и то же поле в одной и той же записи, но различными способами для разных реплик, вы тем самым создаете конфликт. Однако вы также узнали, как разрешать конфликты между реплицируемыми таблицами. Некоторые объекты в репликах не могут быть созданы или изменены. До настоящего времени вы работали с целыми таблицами в пределах баз данных, но что если вы хотите работать с частичной информацией в одной или более таблицах? В следующем подразделе описываются ситуации, возникающие при частичной репликации.

Частичные реплики

Возможно, у вас есть служащий, или отделение, или целая секция в вашей компании, которым нужно видеть только часть базы данных. В этом случае можно создать частичную реплику, которая предоставляет лишь подмножество информации от одной или большего числа реплицируемых таблиц. Другими словами, удерживается только та информация, которая существенна для отдельного пользователя или группы пользователей.

Например, продавцы с ограниченной емкостью их жестких дисков могут беречь свое дисковое пространство, получая только частичные реплики, относящиеся к их конкретной области. Этот тип реплики позволяет достичь максимального уровня безопасности. Это значит, что важная информация может быть защищена путем ограничения доступа к ней для всех, кроме отдельных индивидуумов или отделов.

Частичную реплику создают путем использования Partial Replica Wizard (Мастер частичных реплик), который позволяет вам выбирать только ту таблицу (или таблицы, если установлены отношения), к которой вам нужно иметь доступ, вместе с фильтром для того, чтобы определять, какие записи включать в подмножество. После создания новой частичной реплики не удаляйте полный вариант реплики, потому что она выполняет роль резервной копии (хотя и частичной) с момента до начала создания частичной реплики. Кроме того, частичные реплики могут быть синхронизированы только с полными репликами.

Если вас интересует, что случилось с Майком, то он сохраняет только частичную реплику, потому что ему нужна лишь часть, относящаяся к искам по Флориде для его страховых продаж. Для флоридских записей должен быть создан фильтр. Следующий пример знакомит вас с процессом создания частичной реплики, но заканчивается раньше, чем обычное создание реплики, потому что вы учитесь, как создавать реплики программно.

1. В базе данных Sales Master - мастер-реплика выберите **Сервис** (Tools), **Репликация** (Replication), **Мастер частичной репликации** (Partial Replica Wizard) для того, чтобы открыть первую страницу мастера (рис. 23.8) и начать процесс создания.

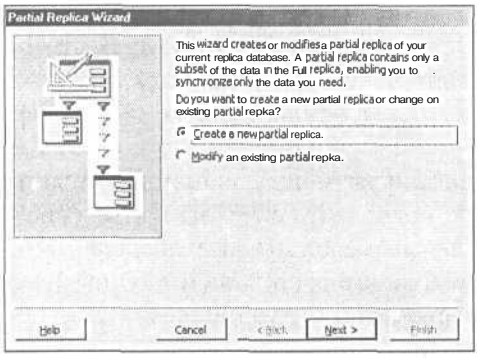


Рис. 23.8. Выберите **Создать новую частичную реплику** (Create a New Partial Replica), которая содержит только подмножество данных полной реплики, на первой странице мастера

2. Нажмите Далее для того, чтобы выбрать опцию **Создать новую частичную реплику** (Create a New Partial Replica) и открыть следующую страницу мастера частичной репликации, изображенную на рис. 23.9. Вы можете выбрать глобальную, локальную или анонимную реплику на этой странице.

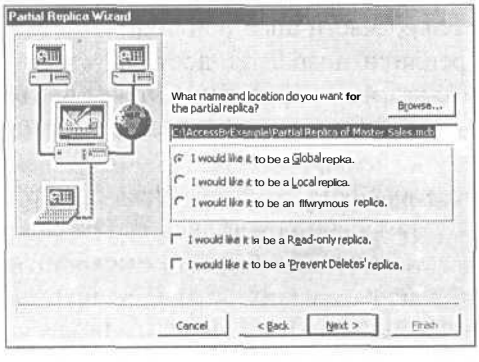


Рис. 23.9. Вы можете сделать вашу частичную реплику глобальной, локальной или анонимной на второй странице мастера

Табл. 23.3 описывает выборы глобальной, локальной или анонимной реплики.

Таблица 23.3. Видимость реплик

Видимость	Объяснение
Глобальная	Может синхронизироваться со всеми другими глобальными репликами и репликами, созданными из нее самой, с некоторыми исключениями. Из глобальной реплики вы можете создать глобальную, локальную или анонимную реплику
Локальная	Может синхронизироваться только со своей родительской, глобальной репликой. Локальные реплики имеют приоритет, равный нулю, который нельзя изменить
Анонимная	Может синхронизироваться только со своей родительской, глобальной репликой. Подобна локальным репликам, но ее информация реплики не всегда хранится в системной таблице MSysReplicas. Анонимные реплики имеют приоритет, равный нулю, который нельзя изменить

3. Нажмите **Далее** для того, чтобы выбрать глобальную видимость и открыть следующую страницу мастера (рис. 23.10).

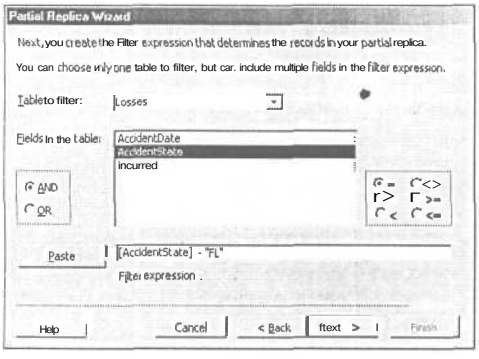


Рис. 23.10. Вы можете создать выражение для фильтра на стр. фильтра мастера частичных реплик

4. Выберите AccidentState в ниспадающем окне, следующем за **Таблицей для фильтрации** (Table to Filter). Оставьте выбранным знак «=» в группе опций, и нажмите **Вставить** (Paste) для того, чтобы вставить выражение [AccidentState]=[Expression] в строке «выражение для фильтра» (Filter expression). Смените [Expression] на «FL» (чтобы выглядело, как изображено на рис. 23.10) и нажмите **Далее** (Next) для того, чтобы открыть следующую страницу мастера, изображенную на рис. 23.11.

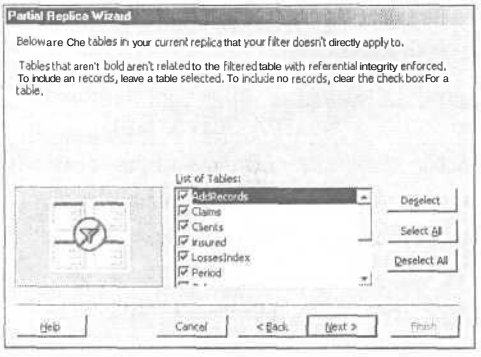


Рис. 23.11. На странице, показывающей таблицы, которые не используют фильтр, нажмите Далее, чтобы выбрать установки по умолчанию

5. Нажмите **Дальше** для того, чтобы открыть последнюю страницу мастера частичной репликации, изображенную на рис. 23.12.

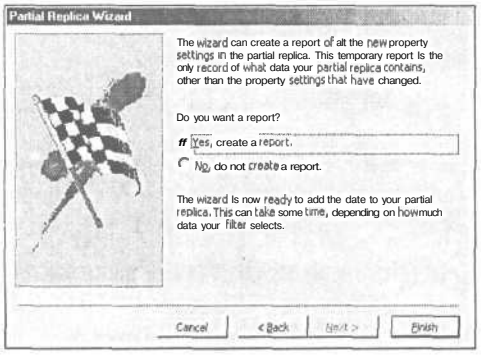


Рис. 23.12. Выберите Отмена (Cancel) на последней странице мастера частичной репликации

6. На последней странице мастера вы можете выбрать создание отчета, но из-за того, что вы создадите мастер программно, нажмите **Отмена** для того, чтобы отменить процесс создания реплики.

Программное создание частичной реплики

Теперь, после того, как вы узнали, как вручную создавать частичную реплику, вы можете создать ее программно. Следующий листинг создает частичную репликуизмастера-реплики:

```
Public Sub PartialReplica()
Dim rplMaster As New JRO.Replica
Dim rplPartial As New JRO.Replica

'Должны быть установлены ссылки на Microsoft Jet и
'Replication Objects 2.6 (или последнюю версию)
```

```
'Должна быть запущена из иных баз данных, нежели основной (Master)
'или реплики (Replica), к которым эта процедура обращается

'Должно быть установлено соединение с мастером дизайна (Design Master)
rplMaster.ActiveConnection = "C:\AccessByExample\Master Sales.mdb"

'Создаем частичную реплику из мастера дизайна (Design Master)
rplMaster.CreateReplica "C:\AccessByExample\" & _
  "Partial Mike.mdb", _
  "Partial Replica For Mike", jrRepTypePartial

'Должно быть создано эксклюзивное соединение для PopulatePartial
rplPartial.ActiveConnection = _
  "Provider=Microsoft.Jet.OLEDB.4.0;" & _
  "Data Source=C:\AccessByExample\" & _
  "Partial Mike.mdb;Mode=Share Exclusive"

'Создаем фильтр
rplPartial.Filters.Append "Losses", jrFilterTypeTable, _
  "AccidentState = 'FL'"

'База данных, из которой вы получаете записи для заполнения
rplPartial.PopulatePartial "C:\AccessByExample\" & _
  "Master Sales.mdb"

Set rplMaster = Nothing
Set rplPartial = Nothing

End Sub
```

Аббревиатура JRO означает Microsoft Jet and Replication Objects. Как показывают комментарии, ссылка должна быть установлена на библиотеку JRO 2.6 в том модуле, в котором вы вводите этот код. Вам будет выгоднее набирать этот код в модуле в пустой базе данных или любой базе данных, кроме подверженных процедуре, потому что, набирая код, вы можете просматривать доступные опции, что очень полезно.

Процедура начинается с установления соединения с мастером-репликой. Далее из мастера-реплики создается пустая частичная реплика (по крайней мере насколько связаны табличные данные). Дальше устанавливается единственное соединение так, чтобы таблицы могли заполняться, но не до тех пор, когда в таблице создается фильтр для таблицы, которая должны быть частично заполнена. Наконец, частичная реплика заполнена записями. Запускайте процедуру и проверяйте таблицу «Убытки» в частичной базе данных Майка для подтверждения того, что флоридские записи были скопированы.

Что дальше?

Дальше - наслаждайтесь тем, что вы узнали, путем использования идей на практике. Теперь вы знаете, как применять силу Access. В сети Интернет вы также можете найти другие источники.

- Адрес Web-сайта издательства Que: www.quepublishing.com. Здесь вы найдете информацию о новых выпусках и бесплатно скачиваемых кодах.
- Внутри Microsoft Access - полезный журнал с Web-сайтом, полным советов и методик. Вы должны быть подписчиком для того, чтобы пользоваться всеми услугами этой службы. Адрес Web-сайта: www.elementkjournals.com.
- Web-форум разработчика Microsoft Office также полезен и может быть найден на www.msdn.microsoft.com/office. Там вы можете включить «Базу знаний» в ваши расширенные поиски.
- Еще один полезный сайт Microsoft называется «Используем Access». Здесь вы найдете советы и приемы, руководства, обучение и сертификации, конференции и другие интересующие вас вещи.
- Еще один сайт с подпиской - «Советник Access VB SQL» вы можете найти на www.advisor.com/www/AccessVBSQLAdvisor. Щелкните на Microsoft Access Advisor Zone для чтения статей, связанных с Access.

Учтите, что Web-сайты могут менять адрес, но вы всегда можете найти желаемый сайт с помощью поиска.

Предметный указатель

И

К

Л

М

Н

О

У

Ф

Ц

Ч

Э

Я

Содержание

ИЗДАТЕЛЬСТВО
«ОЦ КУДИЦ-ОБРАЗ»

Тел.: (095) 333-82-11; ok@kudits.ru, http://www.kudits.ru/publish

ПРИОБРЕТАЙТЕ КНИГИ У НАШИХ ПАРТНЕРОВ

Барнаул
Социалистический пр-т, 117а,
(3852) 38-18-72

Великий Новгород
ул. Б. Санкт-Петербургская, 44
тел. (81622)73-188доб. 34

Екатеринбург
"Книжный мир",
ул. 8 Марта, 8г, (3432) 71-18-87

Краснодар
"Мир книги", ул. Буденного, 147

Москва
"ОПТИМА+"
Розничная торговля
м. Тульская, Варшавское ш., 9,
книжная ярмарка "Центральная",
зеленая линия, павильон 412-57
Оптовая торговля
(095) 333-65-67

Нижний Новгород
"Нижегородский Дом книги",
ул. Советская, 14,
тел. (8312) 44-22-73

Новосибирск
"Книжный пассаж",
ул. Ленина, 10а, (3832) 29-50-30
"Сибирский Дом Книги",
Красный пр-т, 153, (3832) 26-62-39
"Книжный мир", пр-т К.Маркса, 51

Пермь
000 "Образование",
ул. Солдатова, 37,
тел. (3422) 45-96-55
тел./факс. (3422) 42-81-16

Омск
"Книжный Мир",
ул. Ленина, 17/19, (3812) 24-32-54

Ростов-на-Дону
"Мир книги", Ворошиловский пр-т, 33,
(8632) 62-54-61

Санкт-Петербург
"Санкт-Петербургский Дом книги"
Невский проспект, 28, тел. (812) 312-01-84
издательство "Наука и Техника"
пр. Обуховской Обороны, 107,
тел. (812) 567-70-25, 567-70-26

Саратов
"Книжный Мир",
пр-т Кирова, 32, (8452) 32-98-14

Ставрополь
"Книжный Мир", ул. Мира, 337, (8652) 35-47-90

Томск
"Книжный Мир", ул. Ленина, 141, (3822) 51-07-16

Уфа
000 ПКП "Азия", тел./факс: (3472) 50-39-00
Оптовая торговля
Ул. Зенцова, 70
Розничная торговля
Магазин "Оазис", ул. Чернышевского, 88
Магазин "Книжник", пр. Октября, 106

Ханты-Мансийск
Магазин "Книги", ул. Ленина, 39

Челябинск
"Книжный Мир", ул. Кирова, 90, (3512) 33-19-58

Ярославль
Магазин "Наука",
ул. Володарского, 63, (0852) 25-95-04

РОЗНИЧНАЯ ТОРГОВЛЯ ПО ИЗДАТЕЛЬСКИМ ЦЕНАМ В МОСКВЕ

**Варшавское ш., 9, м. Тульская, трамвай 3, 38, 47 до ост. "Стройдвор"
книжная ярмарка "Центральная", зеленая линия, павильон 515-57**

Программирование Access 2002

В ПРИМЕРАХ

Если вы, как и подавляющее большинство людей, не знакомы с предметом, вы узнаете его наилучшим образом, следуя ясным, лаконичным примерам. Книги серии "В примерах" придерживаются логического последовательного подхода в обучении, проводя читателя через небольшие последовательные этапы, позволяющие увидеть всю большую картину того, что вы изучаете.

Эта книга даст вам возможность быстро начать писать программы с помощью интегрированной среды разработки MS Access 2002. Она научит вас фундаментальным основам, таким, как нормализация баз данных, создание запросов, работа с объектами и оптимизация, а также создание пользовательских форм и отчетов с помощью программных средств. Эти умения позволят вам добиться большей эффективности, производительности и профессионального владения средствами Access VBA. Главы книги включают не только полезные советы, примечания, предупреждения и ссылки, но также содержат резюме, которые дают место каждой главы в общей перспективе. Каждая глава предлагает наглядные ситуации из реальной жизни, помогающие вам **учиться...** *в Примерах.*

Боб Вилльариал является автором и постоянным экспертом Web-журнала Microsoft Access, а также инструктором по Microsoft Access в колледже Tulsa Community Colledge. Наряду с этим он занимается вольным (freelance) программированием и преподаванием.
Боб разрабатывал базы данных *в* очень крупной страховой компании на протяжении более 15 лет.
У него более 7 лет опыта программирования на VBA как в Access, так и в Excel, им написано множество мониторинговых и управленческих приложений.

www.quepublishing.com

- Изучение принципов качественного проектирования баз данных
- Понимание связей и каскадных отношений, ссылочной целостности, реляционных форм и отчетов
- Создание мощных и эффективных запросов
- Применение страниц доступа к данным для интеграции вашей базы данных с Интернет
- Интеграция вашей базы данных данных с SQL Server
- Настройка вашей базы под пользователя с помошью своих VBA-программ
- Упрощение ваших баз данных, используя советы, методы и способы сокращения трудозатрат
- Раскрытие преимуществ программирования с использованием объектных моделей DAO и ADO
- Подстраховка вашей базы данных с помошью процедур обработки ошибок
- Интеграция вашей базы данных с другими приложениями