Использование Microsoft Query

5

В ЭТОЙ ГЛАВЕ

Что такое запрос

В предыдущей главе сравнивалось использование мастера запросов и компонента Microsoft Query[1]. Первый из них — это удобное, дружественное, но менее мощное средство, а второй — инструмент создания запросов, требующий от пользователя наличия определенных навыков, но вместе с тем обладающий куда более развитой функциональностью.

Аналогичная ситуация возникает при выборе между средством Microsoft Query и диспетчером базы данных. Microsoft Query предлагает несколько способов структурирования запроса (к примеру, выбор записей и полей, а также порядка сортировки), однако не обеспечивает всех тех возможностей, которыми обладает пользователь при непосредственном создании запроса с помощью полноценной СУБД.

Несмотря на это, Microsoft Query остается единственным средством, позволяющим импортировать данные в рабочую книгу Excel автоматически и без применения программирования. А один из наиболее удачных способов извлечения данных из БД в рабочую книгу Excel состоит в том, чтобы использовать средства базы

[1] *Слово "query" переводится как "запрос".* — Прим. ред.

данных для структурирования данных запроса, а затем применять Microsoft Query для перемещения этих данных на рабочий лист.

Едва ли не самая большая сложность в изучении запросов связана с самим словом *запрос*. Этим термином привыкли обозначать практически все — от данных, возвращаемых запросом, до набора инструкций, определяющих, как следует обрабатывать набор данных, и даже до приложения, предназначенного для создания запросов. Ситуацию еще более усложняет тот факт, что запросы делают нечто большее, нежели просто извлекают данные из источника данных. Запросы могут добавлять или удалять содержимое таблиц, изменять данные и даже создавать новые таблицы.

В этой книге под словом "запрос" подразумевается набор инструкций. В качестве примера рассмотрим простой запрос.

```
SELECT Плитка.КодПлитки, Плитка.КодПомещения, Плитка.Этаж,
Плитка.МестоДляКурения FROM Плитка;
```

Этот запрос написан на языке SQL (Structured Query Language — язык структурированных запросов). Он указывает на необходимость извлечь поля "КодПлитки", "КодПомещения", "Этаж" и "МестоДляКурения" из некоторого источника по имени `Плитка`, каковым может быть таблица или другой запрос. (Довольно часто запросы создаются на основе других запросов.) Большинство запросов, рассматриваемых в этой книге, являются *запросами на выборку*. Это запросы, которые извлекают данные из базы данных и передают их другому приложению. В нашем случае таким приложением является Microsoft Excel, однако принципы извлечения данных, изложенные в этой главе, могут быть применены и к любому другому приложению.

Если запрос на выборку каким-либо образом изменяет данные (например, "Если значение поля "Пол" равно 1, отобразить значение "Мужской"; если значение поля "Пол" равно 2, отобразить значение "Женский""), это выполняется *после* их извлечения. Таким образом, если тип запроса, рассматриваемого в книге, не указан явно, его можно считать запросом на выборку. О запросах на удаление (удаляет записи из таблицы), добавление (добавляет записи к таблице) или обновление (изменяет записи таблицы) будет сказано дополнительно.

> **Примечание**
>
> SQL – это *стандартный* язык запросов. Инструкция SQL, созданная в одной СУБД, скорее всего, будет работать и в другой СУБД. Присутствие в предыдущем предложении расплывчатой фразы *скорее всего* вызвано наличием у языка SQL нескольких разновидностей. Существует, к примеру, язык Transact-SQL, который во многом отличается от классического SQL. Тем не менее, базовый синтаксис запросов у них практически одинаков.

В действительности писать запросы непосредственно на языке SQL приходится крайне редко. В большинстве популярных приложений, использующих SQL, создавать запросы можно в графическом режиме. Затем приложение ин-

терпретирует графическую информацию, создавая на ее основе код SQL. Пример создания запроса с помощью графического интерфейса в приложении Microsoft Query показан на рис. 5.1. В области таблиц находится таблица со списком полей, в области данных — записи и значения полей, а в диалоговом окне Запрос SQL (SQL) показан текст соответствующего запроса на языке SQL.

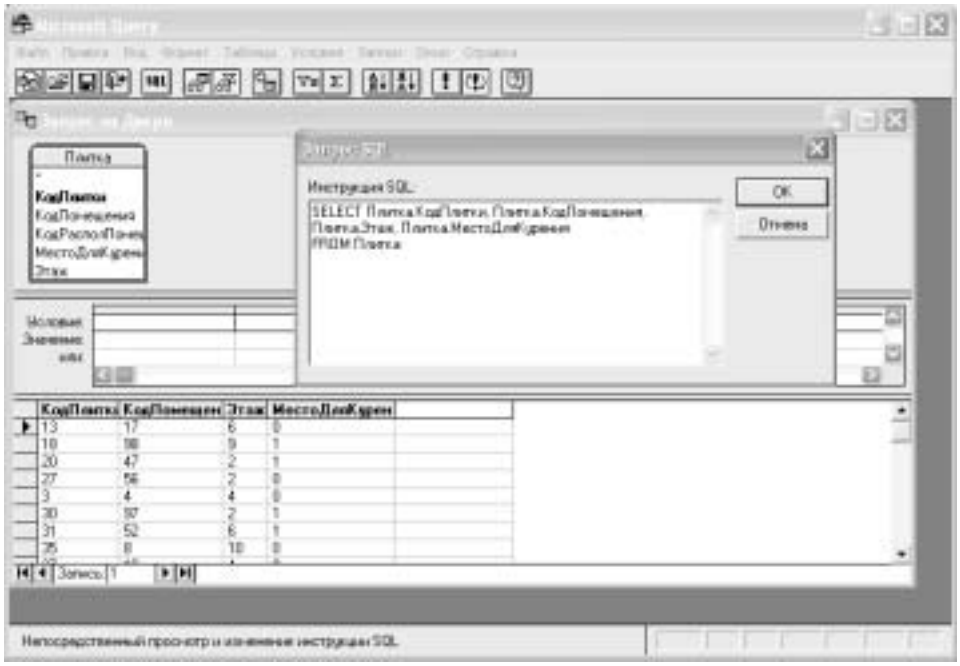


Рис. 5.1. Визуальное представление запроса на языке SQL в приложении Microsoft Query

Чтобы увидеть, как выглядит текст создаваемого запроса на языке SQL, щелкните в панели инструментов окна Microsoft Query на кнопке Режим SQL (SQL).

Совет

Лишь некоторые приложения, облегчающие создание запросов SQL, обладают средством поиска и замены. Предположим, что понадобилось заменить в запросе, показанном выше, имя таблицы `Плитка` на `Плитка2005`. Для этого в область таблиц можно добавить таблицу `Плитка2005`, затем открыть текст запроса и вручную изменить все ссылки на таблицу с `Плитка` на `Плитка2005`. Зачастую текст запроса удобнее скопировать в другое приложение (например, Блокнот), заменить слово `Плитка` на `Плитка2005` и затем скопировать обратно в окно запроса.

Система управления базами данных, будь то Access, Oracle, SQL Server или что-нибудь еще, получает запрос SQL, интерпретирует его и возвращает нужные данные.

Создание запроса к нескольким таблицам

Фундаментальное предназначение запросов состоит в том, чтобы объединять несколько таблиц. Необходимость их объединения может быть обусловлена различными причинами, две из которых, вскользь упоминаемых (правда, исключительно в контексте Excel) в главе 1, "Неправильное использование Excel в качестве средства управления базами данных", будут рассмотрены в следующих разделах.

ПРАКТИКУМ

Вы руководите отделом инвентаризации в компании "Бизнес-недвижимость". Ваша компания сдает нежилую площадь в нескольких принадлежащих ей зданиях в аренду другим компаниям, которым нужны офисные помещения. Персонал вашего отдела создал несколько баз данных, предназначенных для хранения информации о самих помещениях, об их оборудовании (например, кондиционерах и мини-АТС), а также о компонентах помещений (окнах, дверях и т.п.).

В одной из баз данных содержатся записи, которые описывают двери в зданиях компании "Бизнес-недвижимость": двери в офисы, в складские помещения, двери, отделяющие холлы и коридоры, двери на улицу и т.п. По некоторым причинам вам требуется знать имя техника по обслуживанию, который последним проверял состояние каждой двери.

Один из способов решения этой задачи — создать в таблице `Двери` специальное поле, например, "ИмяТехника". Затем, когда техник по обслуживанию проведет осмотр двери, его имя и фамилия будут внесены в таблицу `Двери`.

С другой стороны, ***нельзя*** допустить, чтобы имя и фамилию техника по обслуживанию приходилось каждый раз вводить вручную. В противном случае простая опечатка оператора приведет к "появлению" нового техника. К примеру, если вместо фамилии "Иванов" набрать "Ивашов", любая статистика, основанная на фамилиях техников по обслуживанию, станет некорректной. Она покажет, что дверь была осмотрена не Ивановым, а несуществующим Ивашовым.

До вашего прихода в компанию "Бизнес-недвижимость" персонал отдела инвентаризации управлял данными только с помощью Microsoft Excel. Сотрудники отдела решили проблему опечаток, настроив проверку данных с помощью команды **Данные⇨Проверка** (Data⇨Validation) и создав список проверки данных (рис. 5.2).

Вы, как новый руководитель, настаиваете на том, чтобы вся информация об инвентаре хранилась в полноценной базе данных. В связи с этим защита от опечаток с помощью списка проверки данных Excel становится неактуальной. Чтобы сохранить целостность данных в базе данных, вы создаете таблицу с именами и фамилиями техников по обслуживанию, которые работают в компании "Бизнес-недвижимость", и организуете таблицу `Двери` таким образом, чтобы имя техника выбиралось из раскрывающегося списка. Это позволит избежать опечаток при вводе с клавиатуры.

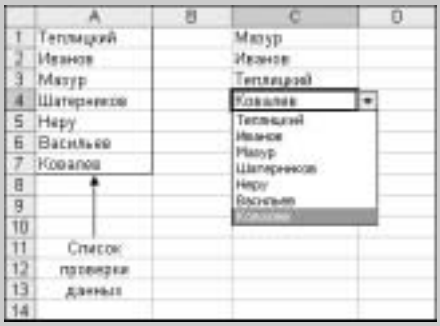


Рис. 5.2. Список проверки данных — еще один удачный пример использования динамического имени диапазона

Новая таблица получает имя `Техники`, а их имена и фамилии помещаются в поле "ИмяТехника". Помимо этого, в таблице `Техники` удобно создать поле, содержащее уникальные идентификаторы записей (например, "КодТехника"). Поле с таким же именем можно поместить и в таблицу `Двери`, заменив им поле "ИмяТехника". Пример подобной организации данных показан на рис. 5.3.

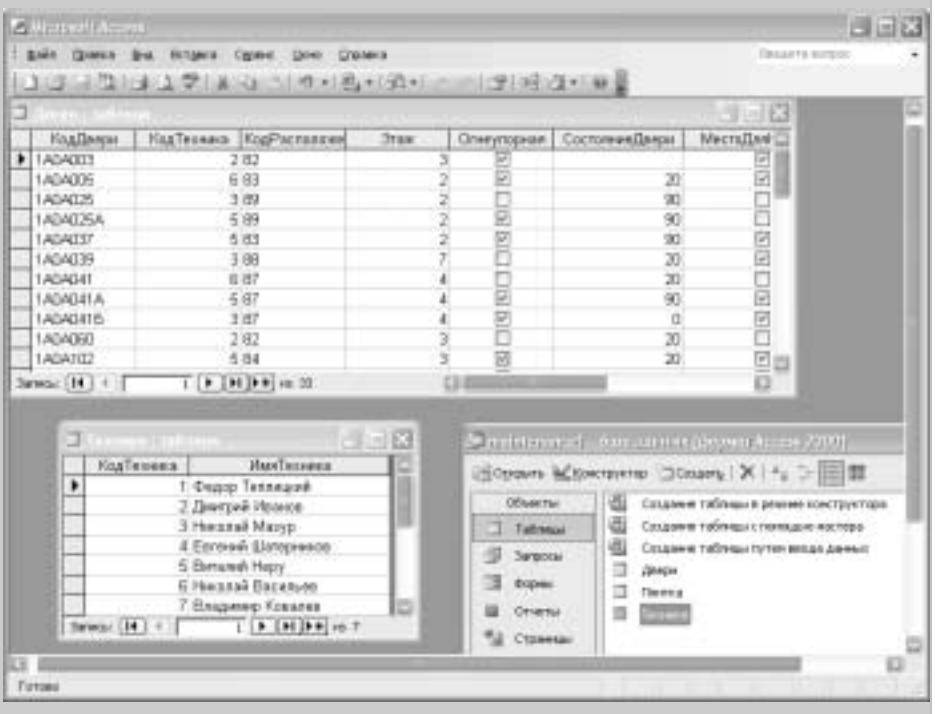


Рис. 5.3. В каждой из двух таблиц есть поле "КодТехника", что позволяет связать эти таблицы

Пусть в таблице Техники имеется запись, значение поля "КодТехника" которой равно 1, а значение поля "ИмяТехника" равно "Федор Теплицкий". Если Федор Теплицкий был последним техником по обслуживанию, который осматривал некоторую дверь, в записи, соответствующей этой двери (таблица Двери), значение поля "КодТехника" должно быть равным 1.

Организовав данные указанным способом, для извлечения записей в рабочую книгу Excel с помощью Microsoft Query можно предпринять следующие действия:

1. Откройте рабочую книгу Excel и выберите в меню команду **Данные⇨Импорт внешних данных⇨Создать запрос** (Data⇨Import External Data⇨New Database Query).
2. Создайте новый источник данных или выберите существующий. На экране появится окно средства Microsoft Query, а также диалоговое окно **Добавление таблицы** (Add Tables).

➔ О том, как создать новый источник данных, идет речь в главе 4, "Импорт данных: обзор", раздел "Выбор расположения и формата данных".

3. Щелкните в диалоговом окне **Добавление таблицы** на имени таблицы Двери, а затем на кнопке **Добавить** (Add). Таблица Двери появится в области таблиц.
4. Повторите шаг 3 для таблицы Техники.
5. Щелкните на кнопке **Закрыть** (Close), чтобы закрыть диалоговое окно **Добавление таблицы**.

Область таблиц станет выглядеть так, как показано на рис. 5.4.

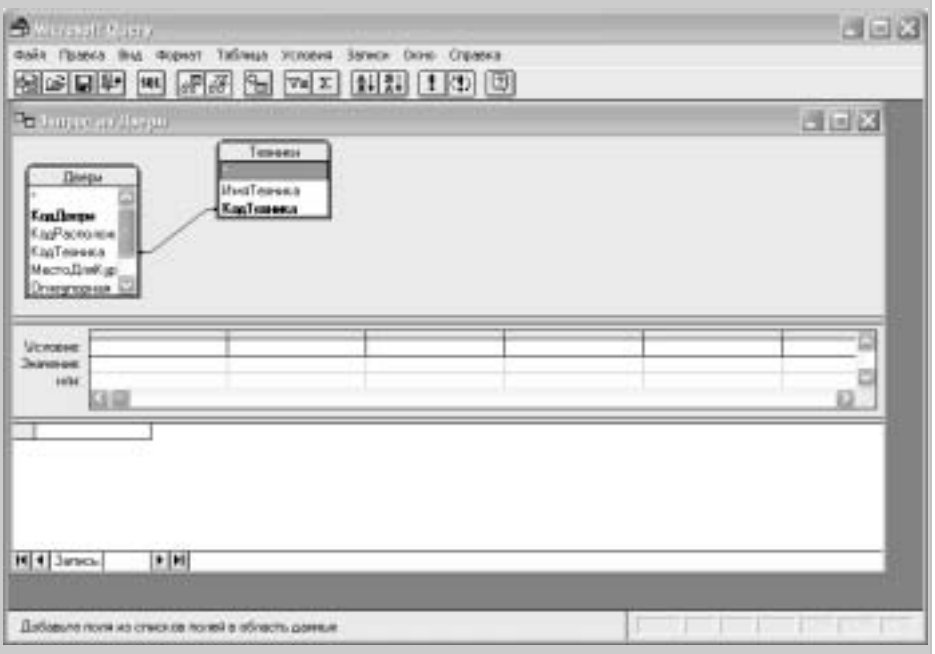


Рис. 5.4. Чтобы запрос возвращал все поля таблицы, дважды щелкните на "звездочке" в верхней части списка полей или же перетащите ее в область данных

Как видно на рис. 5.4, между двумя таблицами появится *линия объединения*, соединяющая поле "КодТехника" таблицы Двери с одноименным полем таблицы Техники.

> **Примечание**
>
> Если линия объединения не появилась автоматически, щелкните на имени поля "КодТехника" одной таблицы, перетащите его на имя поля "КодТехника" другой таблицы и отпустите кнопку мыши.

Чтобы запрос возвращал код двери и имя техника по обслуживанию, который осматривал эту дверь последним, перетащите поле "КодДвери" из таблицы Двери в область данных. Затем проделайте то же самое с полем "ИмяТехника" из таблицы Техники. Полученный результат показан на рис. 5.5.

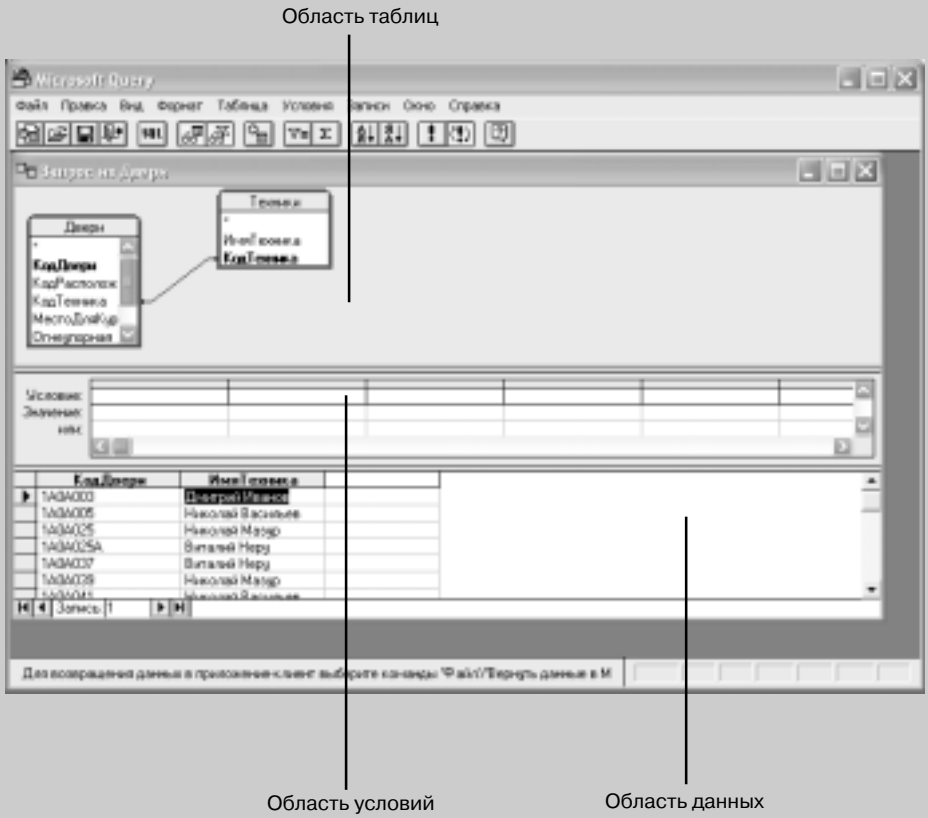
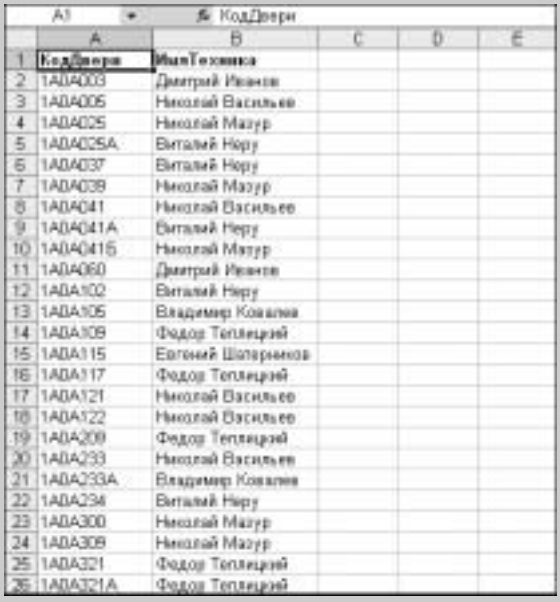


Рис. 5.5. Чтобы возвратить поле "ИмяТехника", в область данных не нужно перетаскивать поле "КодТехника"

Если выбрать в меню команду Файл⇨Вернуть данные в Microsoft Office Excel (File⇨Return Data to Microsoft Excel), на экране появится диалоговое окно Импорт данных (Import Data). Укажите, куда следует поместить извлеченные данные, и щелкните на кнопке OK. Результат выполнения запроса показан на рис. 5.6. Теперь можно анализировать состояние дверей в зданиях, принадлежащих компании "Бизнес-недвижимость", с помощью привычных средств анализа Microsoft Excel, например, сводных таблиц или диаграмм.



	A	B	C	D	E
1	КодДвери	ИмяТехника			
2	1ADA003	Дмитрий Иванов			
3	1ADA005	Николай Васильев			
4	1ADA025	Николай Мазур			
5	1ADA025A	Виталий Нору			
6	1ADA037	Виталий Нору			
7	1ADA039	Николай Мазур			
8	1ADA041	Николай Васильев			
9	1ADA041A	Виталий Нору			
10	1ADA041B	Николай Мазур			
11	1ADA060	Дмитрий Иванов			
12	1ADA102	Виталий Нору			
13	1ADA105	Владимир Коваленк			
14	1ADA109	Федор Теплицкий			
15	1ADA115	Евгений Шапранов			
16	1ADA117	Федор Теплицкий			
17	1ADA121	Николай Васильев			
18	1ADA122	Николай Васильев			
19	1ADA209	Федор Теплицкий			
20	1ADA233	Николай Васильев			
21	1ADA233A	Владимир Коваленк			
22	1ADA234	Виталий Нору			
23	1ADA300	Николай Мазур			
24	1ADA309	Николай Мазур			
25	1ADA321	Федор Теплицкий			
26	1ADA321A	Федор Теплицкий			

Рис. 5.6. Результат выполнения запроса: записи из разных таблиц, объединенные скрытым общим полем

Объединение родительских и дочерних записей

Предположим, что здания, о которых шла речь в предыдущем разделе, должны соответствовать определенным требованиям муниципальных властей. В число этих требований входит регулярная проверка дверей и устранение всех обнаруженных неполадок. Двери, ведущие на улицу, должны функционировать и в случае чрезвычайных ситуаций. Двери в холлах принято делать из огнеупорного материала, чтобы замедлять распространение огня. Двери в складские помещения должны обладать повышенной степенью защиты от взлома, в особенности, если на складах могут храниться опасные материалы.

Все это, в свою очередь, означает необходимость регулярного осмотра каждой двери и фиксирования в базе данных ее состояния, включая действия, предпринятые в целях ремонта. Более того, если съемщик офисного помещения жалуется, скажем, на то, что дверь в туалетную комнату не закрывается,

жалоба должна быть занесена в базу данных. Разумеется, жалобы от клиентов поступают в совершенно непредсказуемые моменты, вне зависимости от графика планового обслуживания двери.

ПРАКТИКУМ

Выбор родительской записи

Вышеописанная ситуация встречается на практике отнюдь не так редко, как кажется на первый взгляд. Как организовать хранение информации о дверях в базе данных компании "Бизнес-недвижимость"? Нужно ли выделить каждой двери отдельную запись? И как увязать это с тем, что на протяжении календарного года проверка состояния и техническое обслуживание каждой двери производится не известное заранее количество раз?

Сколько полей таблицы следует выделить для фиксации сведений по обслуживанию? Не забывайте, что по каждому факту осмотра двери и проведения ремонтных работ в базу данных следует внести информацию о технике, проводившем обслуживание, дате проведения работ, типе работ, распространении гарантии на указанный тип работ и т.п. Как видим, запись о двери должна включать в себя довольно много полей. На рис. 5.7 показано, как можно было бы организовать хранение информации о дверях в "плоском файле".

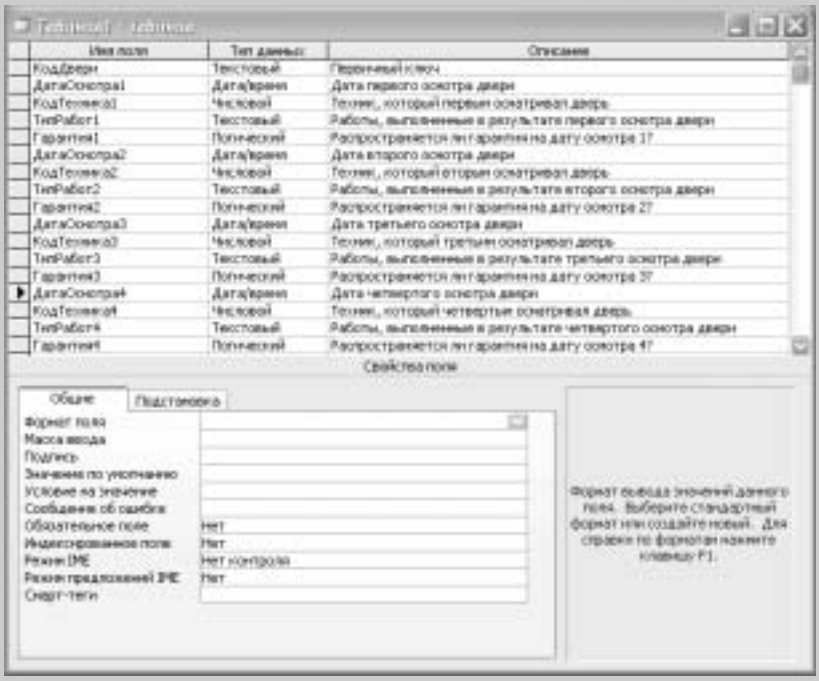


Рис. 5.7. Такая организация данных могла бы придти в голову человеку, не знакомому с реляционными структурами

Примечание

Плоский файл — это структура данных, не являющаяся реляционной. В качестве примера плоского файла можно привести список Excel: его записи находятся в строках, а поля — в столбцах. Это двумерная, или *плоская* структура данных.

Иногда организация данных, показанная на рис. 5.7, оказывается вполне удобной. Если вы уверены в том, что обслуживание двери за время ее существования будет проводиться не более четырех раз, использование плоского файла значительно упростит управление данными. Многие схемы организации данных, которые в теории выглядят ужасными, на практике проявляют себя просто замечательно.

Однако зачастую использование подобной организации данных влечет за собой множество проблем. В конце концов, каждый набор данных об обслуживании в действительности является *отдельной записью*, и представление этих данных в виде полей искажает ситуацию. Кроме того, для проведения анализа данных по обслуживанию их все равно придется преобразовать в отдельные записи.

Предположим, что нужно подсчитать, сколько раз двери ремонтировались, и сколько раз проводилась замена дверей. Если каждый набор данных о проведении обслуживания хранится в виде отдельной записи, выполнение таких подсчетов не составляет никакого труда: импортируйте поле "ТипРабот" в Excel, а затем воспользуйтесь сводной таблицей (или формулой массива) для подсчета количества записей со словом "Ремонт" и количества записей со словом "Замена".

Если же данные о проведении обслуживания хранятся не в виде записей, а в виде полей, выполнение анализа данных окажется куда более сложным. Для каждой двери придется импортировать данные о первом, втором, третьем,..., N-том проведении работ, извлекая содержимое полей "ТипРабот1", "ТипРабот2", "ТипРабот3",..., "ТипРаботN". Затем полученные данные понадобится преобразовать в список, и только потом к ним можно будет применить сводную таблицу.

Из всего вышесказанного следует: если нельзя предугадать, сколько раз будет проводиться обслуживание каждой двери, нужно искать другое решение. К примеру, каждый набор данных о проведении обслуживания может располагаться в отдельной записи таблицы `Двери`. В результате окажется неопределенное число *записей*, а не полей.

К сожалению, в каждую запись о проведении обслуживания двери придется включать все сведения о самой двери. Ведь необходимо знать, над какой дверью проводилась работа, а это довольно большой объем данных. Нужно поле для хранения номера, уникальным образом идентифицирующего дверь, поле для даты установки двери, поле для названия ее производителя, поле для даты окончания гарантии, поля, в которых будет указано, является ли дверь огнеупорной, закрывается ли она на ключ и т.п. Но зачем же повторять огромные фрагменты постоянной (!) информации о двери в записях, посвященных ее периодическому обслуживанию?

Использование объединений для создания реляционных структур

Для решения этой проблемы, как, впрочем, и большинства аналогичных проблем, структуру базы данных нужно организовать следующим образом: создать одну таблицу, в которой будет содержаться постоянная информация, и еще одну таблицу, содержащую меняющуюся информацию. В данном примере понадобится одна таблица, содержащая постоянную информацию о *самой двери* (код двери, дата установки и т.п.), и еще одна таблица, содержащая информацию об *обслуживании двери* (дата проведения работ, сотрудник, выполнивший работу, и т.п.).

В таблице `Двери` будут храниться параметры двери, которые уже не изменятся (или, по крайней мере, не должны измениться) с течением времени: уникальный код двери, ее производитель, ее расположение в здании и т.п. Такие записи называются *родительскими*.

В таблице `ОбслуживаниеДверей` будет содержаться информация о двери, меняющаяся с появлением каждой новой записи: вид проведенных работ (к примеру, осмотр или ремонт), имя техника, проводившего работы, дата проведения работ, результат и т.п. Такие записи называются *дочерними*. Каждая дочерняя запись принадлежит некоторой родительской записи.

Чтобы установить связь между родительскими и дочерними записями, в каждой из таблиц должно содержаться поле, позволяющее связать, или *объединить* таблицы таким образом, чтобы при извлечении конкретной родительской записи автоматически получать только те дочерние записи, которые принадлежат указанной родительской записи.

В рассматриваемом примере, если запрашивать сведения о входной двери, ведущей с лестницы на второй этаж в северной части здания, нужно гарантировать, что запрос извлечет из таблицы `ОбслуживаниеДверей` только те записи об осмотре и ремонте, которые касаются именно этой двери.

В Microsoft Query или СУБД наподобие Microsoft Access наличие связи между родительскими и дочерними записями показано в области таблиц с помощью так называемой линии объединения (рис. 5.8).

Как видно из рис. 5.8, в таблице `Двери` хранится сравнительно постоянная информация о двери: код двери, этаж, на котором она расположена, является ли дверь огнеупорной и т.п. В таблице `ОбслуживаниеДверей` хранится информация, которая меняется с течением времени: дата осмотра двери, дата, когда дверь была отремонтирована, в чем заключалась неполадка и т.п.

Рассматривая рис. 5.8, обратите внимание: и в таблице `Двери`, и в таблице `ОбслуживаниеДверей` присутствует поле "КодДвери". Именно это общее поле (оно называется *связующим*) позволяет установить связь между таблицами и превращает базу данных в *реляционную* структуру. Она больше не является простым плоским файлом.

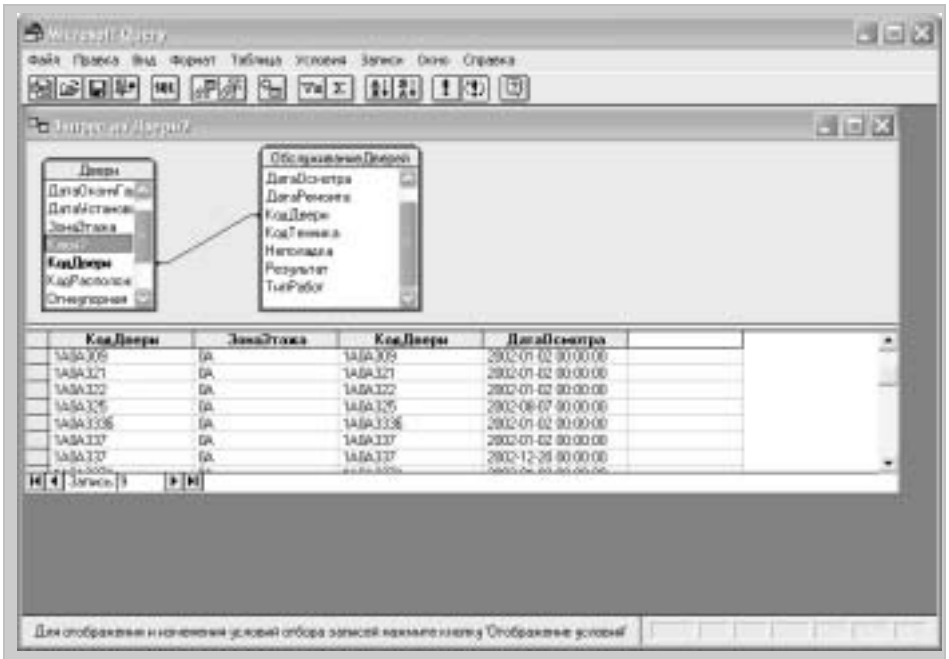


Рис. 5.8. Линия, связывающая таблицы, называется линией объединения

Примечание

Желательно, чтобы имена связующих полей в обоих таблицах были одинаковыми, однако это необязательно. К примеру, одно и то же поле в таблице `Двери` может называться "КодДвери", а в таблице `ОбслуживаниеДверей` — "НомерДвери".

Внутреннее объединение

Если создать объединение двух или более таблиц (см. рис. 5.8), запрос может возвращать записи из одной таблицы и связанные с ними записи из другой таблицы. Две записи из разных таблиц называются *связанными*, если значения связующих полей в этих записях равны. На рис. 5.9 продемонстрирован результат извлечения данных из Microsoft Access в Microsoft Excel с помощью запроса, показанного на рис. 5.8.

Обратите внимание, что значение поля "КодДвери" из родительской записи (записи о двери) всегда равно значению поля "КодДвери" из дочерней записи (записи о проведении обслуживания двери).

A1 КодДвери

	A	B	C	D	E
1	КодДвери	ЗонаЭтажа	КодДвери	ДатаОсмотра	
2	1A0A003	0A	1A0A003	05.12.2002 0:00	
3	1A0A005	0A	1A0A005	20.12.2002 0:00	
4	1A0A003	0A	1A0A003	22.08.2002 0:00	
5	1A0A003	0A	1A0A003	26.02.2003 0:00	
6	1A0A003	0A	1A0A003	07.08.2002 0:00	
7	1A0A003	0A	1A0A003	02.02.2003 0:00	
8	1A0A003	0A	1A0A003	26.03.2003 0:00	
9	1A0A005	0A	1A0A005	02.01.2002 0:00	
10	1A0A025	0A	1A0A025	20.12.2002 0:00	
11	1A0A025	0A	1A0A025	02.01.2002 0:00	
12	1A0A025A	0A	1A0A025A	20.12.2002 0:00	
13	1A0A025A	0A	1A0A025A	02.01.2002 0:00	
14	1A0A037	0A	1A0A037	26.04.2003 0:00	
15	1A0A037	0A	1A0A037	07.08.2002 0:00	
16	1A0A037	0A	1A0A037	02.01.2002 0:00	
17	1A0A037	0A	1A0A037	20.12.2002 0:00	
18	1A0A037	0A	1A0A037	26.02.2003 0:00	
19	1A0A039	0A	1A0A039	02.01.2002 0:00	
20	1A0A039	0A	1A0A039	20.12.2002 0:00	
21	1A0A041	0A	1A0A041	02.01.2002 0:00	
22	1A0A041	0A	1A0A041	20.12.2002 0:00	
23	1A0A041A	0A	1A0A041A	07.08.2002 0:00	
24	1A0A041A	0A	1A0A041A	22.08.2002 0:00	
25	1A0A041Б	0A	1A0A041Б	07.01.2003 0:00	
26	1A0A060	0A	1A0A060	04.05.2004 0:00	

Рис. 5.9. Включать в запрос связующие поля необязательно. Они показаны здесь лишь для наглядности

Следует также отметить, что в списке на рис. 5.9 код двери 1A0A003 встречается шесть раз. Данные, находящиеся в столбцах A и B, извлечены из таблицы `Двери`, в которой указанному экземпляру двери соответствует только одна запись. Откуда же взялось шесть упоминаний о двери в наборе данных, возвращенных запросом? Именно столько раз дверь 1A0A003 фигурирует в таблице `ОбслуживаниеДверей`: по разу на каждую проверку состояния двери, произведенную техником по обслуживанию. Связывая таблицы, запрос может отобразить постоянную информацию о двери наподобие этажа и зоны этажа вместе с меняющейся информацией, такой как дата проверки состояния двери.

Использование реляционной структуры данных позволяет избежать обеих проблем, описанных в предыдущем разделе: создания огромного количества полей, которые могут никогда не быть использованы, или же ненужного повторения постоянной информации из записи в запись.

На рис. 5.8 к полям "КодДвери" в таблицах `Двери` и `ОбслуживаниеДверей` применено так называемое внутреннее объединение (именно оно применяется по умолчанию). Операция *внутреннего объединения* возвращает объединение только тех записей каждой из таблиц, для которых существуют записи с таким же значением связующего поля в другой таблице. К примеру, как показано на рис. 5.8, и в таблице `Двери`, и в таблице `ОбслуживаниеДверей` есть запись о двери с кодом 1A0A321. Тогда запрос возвратит и ту и другую запись, "склеив" их в одну. При внутреннем объединении запрос *не возвращает* записи таблицы, для которых не существует записей с таким же значением связующего поля в другой таблице. Другими словами, запрос не выполнит объединение двух записей, если значения связующего поля в каждой из этих записей не равны.

Поля, по которым проводится объединение таблиц, обладают следующими свойствами.

- Связующие поля должны иметь одинаковый тип данных. Невозможно связать поля, если одно из них имеет числовой, а другое, к примеру, текстовый тип данных.
- Если типы данных и имена полей совпадают, и одно из них является первичным ключом соответствующей таблицы, объединение таблиц по указанным полям в Microsoft Query и Microsoft Access происходит автоматически.
- Если к таблицам применено внутреннее объединение, запрос возвращает объединение только тех записей каждой из таблиц, для которых существуют записи с таким же значением связующего поля в другой таблице.

Примечание

Первичным ключом называется поле таблицы, которое однозначно идентифицирует каждую ее запись. К примеру, в качестве первичного ключа таблицы сотрудников можно использовать номер паспорта, потому что он соответствует в точности одному человеку. В нашем примере первичным ключом таблицы `Двери` является поле "КодДвери", потому что оно однозначно идентифицирует дверь. С другой стороны, поле "КодДвери" не может быть первичным ключом таблицы `ОбслуживаниеДверей`, потому что каждая дверь (и, соответственно, ее код) может упоминаться в таблице по несколько раз, а значения первичного ключа должны быть уникальными. Первичные ключи имеют широкое применение в теории баз данных.

➔ Более подробно о первичных ключах и ключах других типов рассказано в главе 9, "Управление объектами баз данных", раздел "Выбор ключа".

Внешнее объединение

Существует еще два типа объединения: *левое внешнее объединение* и *правое внешнее объединение*. Область применения этих типов объединений не так широка, и в книгу они включены лишь с целью общего ознакомления.

Вернемся к рис. 5.8. Предположим, что в таблице `Двери` есть запись о двери с кодом 1A0A321A, а в таблице `ОбслуживаниеДверей` записей о двери с таким кодом нет (например, потому что обслуживание этой двери в текущем году еще не проводилось). Запрос не возвратит запись о двери с кодом 1A0A321A из таблицы `ОбслуживаниеДверей`, потому что таковая там отсутствует. Не будет возвращена запись о двери с таким кодом и из таблицы `Двери`, хотя в ней такая запись *существует*. Внутреннее объединение выполняется только тогда, когда для записи с некоторым значением связующего поля одной из таблиц существует запись с таким же значением связующего поля в другой таблице.

На рис. 5.10 показано, что произойдет, если в окне средства Microsoft Query выбрать команду Таблица⇨Объединения (Table⇨Joins) или дважды щелкнуть на линии объединения.

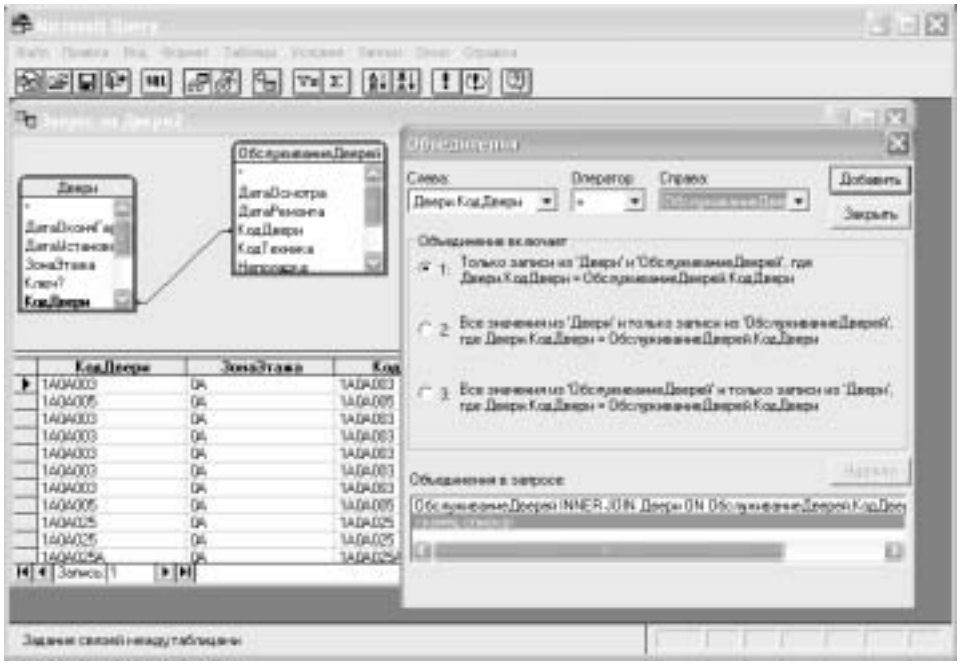


Рис. 5.10. Первый переключатель описывает внутреннее объединение, которое применяется по умолчанию. Второй и третий переключатели описывают внешние объединения

Установите второй переключатель, Все значения из 'Двери' и только записи из 'ОбслуживаниеДверей', где Двери.КодДвери = ОбслуживаниеДверей.КодДвери (All Values from 'Двери' and Only Records from 'ОбслуживаниеДверей' where Двери.КодДвери = ОбслуживаниеДверей.КодДвери), щелкните на кнопке Добавить (Add), а затем на кнопке Закрыть (Close). Результат выполнения полученного запроса показан на рис. 5.11.

Между ситуациями, показанными на рис. 5.8 и 5.11, существует два основных отличия. Во-первых, в области данных на рис. 5.11 появились записи, которых не было на рис. 5.8. У этих записей есть значения в поле "КодДвери" из таблицы `Двери`, но нет значений в поле "КодДвери" из таблицы `ОбслуживаниеДверей`. Именно это возвращает выбранный тип объединения (оно называется *левым внешним объединением*), то есть все записи из таблицы `Двери`, а также только те записи из таблицы `ОбслуживаниеДверей`, для которых существуют записи с таким же значением поля "КодДвери" в таблице `Двери`. Там, где значения полей "КодДвери" и "ДатаОсмотра" отсутствуют, запрос возвращает значения NULL (т.е. пустые значения).

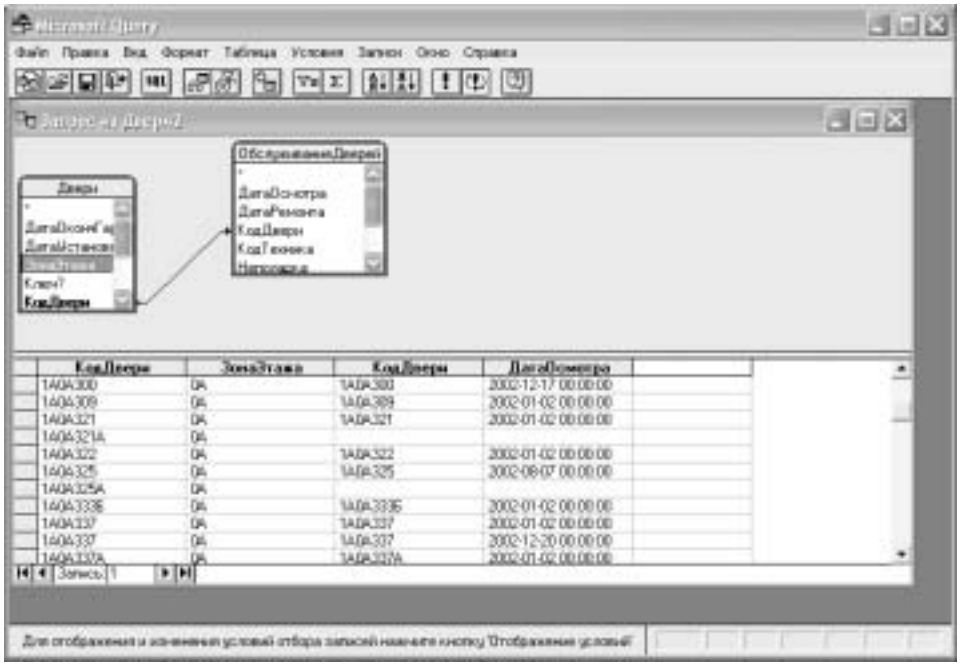


Рис. 5.11. Обратите внимание на стрелку в конце линии объединения. Она указывает на таблицу, в которой может не оказаться записей с соответствующим значением связующего поля

Сравните этот результат с тем, что показан на рис. 5.8. Там, к примеру, нет записи о двери с кодом 1A0A321A, потому что в таблице `ОбслуживаниеДверей` записи о двери с таким кодом найдено не было. На рис. 5.11, однако, запрос извлекает *все* записи из таблицы `Двери` вне зависимости от того, существуют ли для них записи с аналогичным значением связующего поля в таблице `ОбслуживаниеДверей`.

> **Совет**
>
> Данный тип объединения удобен для поиска записей в одной таблице, которым не соответствует ни одна из записей в другой таблице. В запросе на рис. 5.11 для поля "КодДвери" из таблицы `ОбслуживаниеДверей` можно было бы задать критерий отбора `Имеется Null` (Is Null). Этот запрос возвратил бы те и *только те* записи из таблицы `Двери`, для которых не существует записей с аналогичным значением связующего поля в таблице `ОбслуживаниеДверей`.

Второе отличие рис. 5.8 от рис. 5.11 заключается вот в чем: на рис. 5.11 линия объединения заканчивается стрелкой. В Microsoft Query и Microsoft Access наличие стрелки на конце линии объединения указывает на применение внешнего объединения. Стрелка указывает на таблицу, в которой не обязательно должны присутствовать записи с соответствующим значением связующего поля. Вместо недостающих записей будут возвращены значения NULL.

Разница между левым внешним объединением (оператор LEFT OUTER JOIN) и правым внешним объединением (оператор RIGHT OUTER JOIN) весьма тривиальна. Она определяется лишь тем, какая таблица указана до оператора объединения, а какая после. Следующие операции объединения эквивалентны.

```
FROM Двери LEFT OUTER JOIN ОбслуживаниеДверей ON Двери.КодДвери =
ОбслуживаниеДверей.КодДвери;
FROM ОбслуживаниеДверей RIGHT OUTER JOIN Двери ON
ОбслуживаниеДверей.КодДвери = Двери.КодДвери;
```

При выполнении левого внешнего объединения запрос возвращает все записи из таблицы, указанной слева от оператора JOIN. Соответственно, при выполнении правого внешнего объединения запрос возвращает все записи из таблицы, указанной справа от оператора JOIN. В языке SQL слово OUTER ("внешнее") можно опускать.

Использование запросов баз данных

Создание внешних объединений в Microsoft Query имеет по крайней мере одно ограничение: нельзя создать внешнее объединение, если в области таблиц окна Microsoft Query находится больше двух таблиц (рис. 5.12).

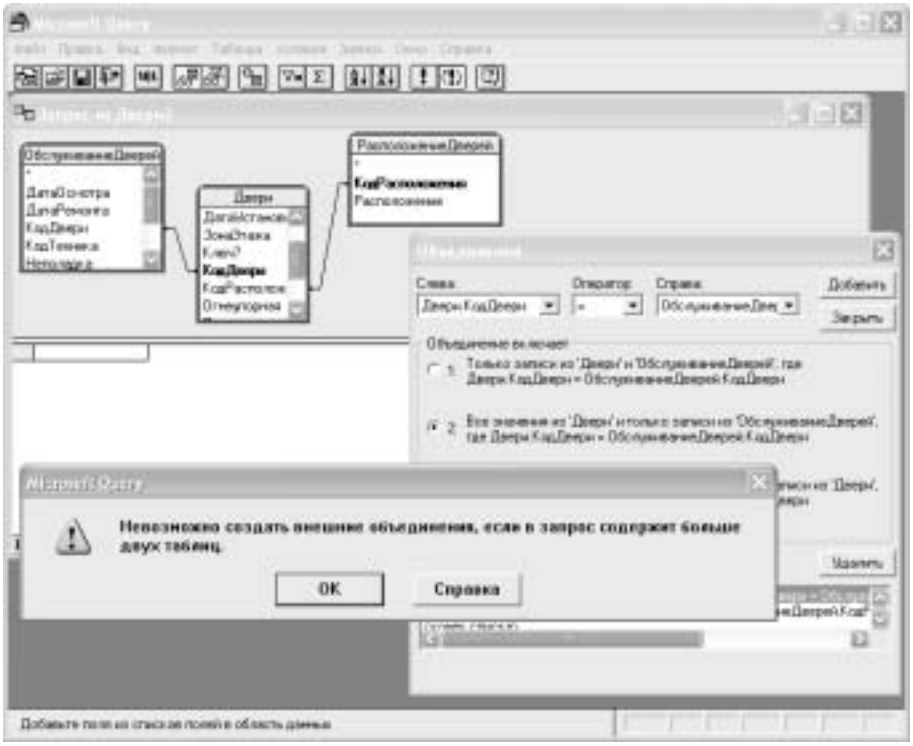


Рис. 5.12. Чтобы создать внешнее объединение на основе трех или более таблиц, необходимо воспользоваться более изощренным диспетчером запросов

Таким образом, даже небольшое усложнение запроса приводит к необходимости создания последнего с помощью средств самой СУБД (например, Microsoft Access, SQL Server или Oracle). В этих приложениях на создание запросов подобных ограничений не накладывается.

Создание запроса в Microsoft Access

Поскольку приложение Access входит в состав пакета Microsoft Office, оно есть у большинства пользователей Excel. Вот почему на его примере удобно рассматривать управление запросами с помощью диспетчера баз данных. Тем не менее, все принципы создания запросов, представленные в этом разделе, применимы не только к Access, но и ко всем остальным системам управления реляционными базами данных.

> **Примечание**
>
> В следующих главах книги будет рассказано, как использовать Excel для построения запросов, которые будут выполняться системой управления базами данных. В конце концов, эта книга описывает управление данными с помощью Microsoft Excel, а не Microsoft Access. Тем не менее, пользователю не мешает научиться создавать запросы и в самой СУБД, чтобы затем иметь возможность импортировать их результаты с помощью запроса Microsoft Query.

Предположим, что данные компании "Бизнес-недвижимость", о которых уже говорилось в этой главе, хранятся в базе данных Microsoft Access. Чтобы построить запрос, создание которого выходит за рамки возможностей Microsoft Query, следует воспользоваться средствами самого Access. Для последующего извлечения полученных данных в диапазон внешних данных или сводную таблицу Microsoft Excel можно использовать все то же средство Microsoft Query, рассматривая запрос Access как обычную таблицу.

Вначале базу данных нужно открыть в приложении Access. После ее открытия на экране появится главное окно Access, показанное на рис. 5.13.

Помимо всего прочего в окне базы данных перечислены три таблицы: `Двери`, `ОбслуживаниеДверей` и `РасположениеДверей`. Чтобы на основе этих таблиц создать запрос, щелкните в окне базы данных на кнопке Запросы (Queries) (или, в Access 97, на аналогичной вкладке), а затем на кнопке Создать (New). На экране появится диалоговое окно Новый запрос (New Query), показанное на рис. 5.14.

В появившемся списке выберите параметр Конструктор (Design View) и щелкните на кнопке OK. На экране появится диалоговое окно Добавление таблицы (Show Table), полностью аналогичное одноименному окну Microsoft Query, которое было показано на рис. 4.12. Диалоговое окно Добавление таблицы останется на экране до тех пор, пока вы не выберете все необходимые таблицы и запросы и не щелкнете на кнопке Закрыть (Close).

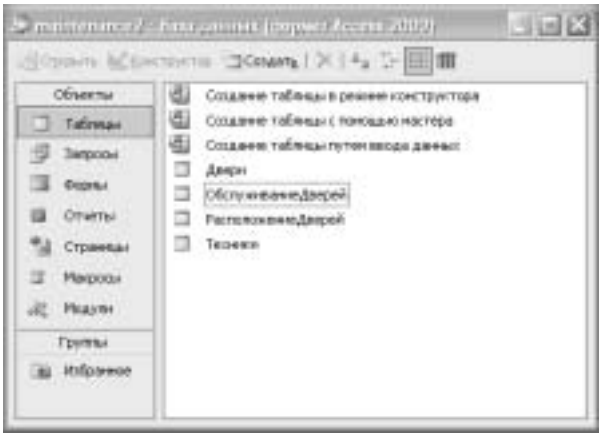


Рис. 5.13. Расположение объектов (таблиц, запросов, форм и т.п.) в окне базы данных зависит от того, какая версия Access используется

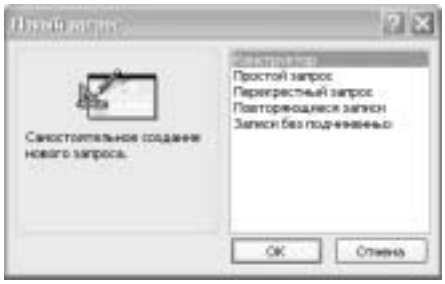


Рис. 5.14. Иногда для построения запроса можно воспользоваться мастером, однако обычно это делается в режиме конструктора

Совет

Если база данных включает в себя лишь несколько таблиц и запросов, перейдите во вкладку **Таблицы и запросы** (Both), чтобы отобразить все объекты за один раз. Если же запросов и таблиц слишком много, и для поиска нужного объекта приходится подолгу прокручивать их список, удобнее вначале отобразить перечень таблиц с помощью вкладки **Таблицы** (Tables), а затем воспользоваться вкладкой **Запросы** (Queries).

Выбрав таблицы и запросы, необходимые для создания нового запроса, щелкните на кнопке Закрыть. На экране появится окно запроса, показанное на рис. 5.15.

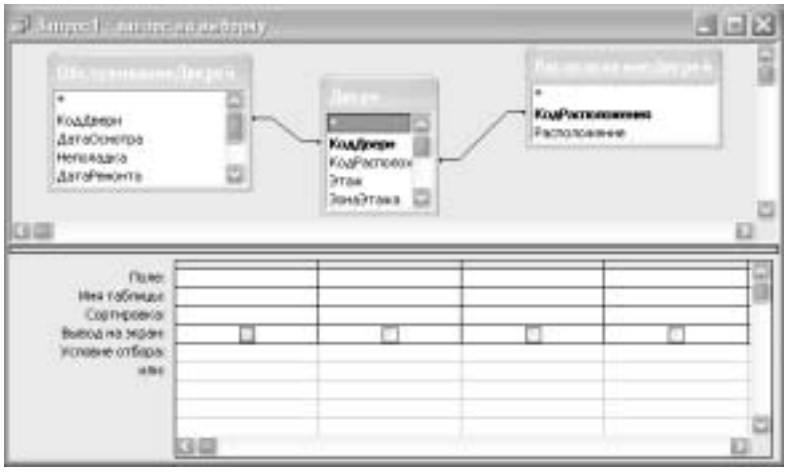


Рис. 5.15. В окне конструктора таблицы расположены в том порядке, в котором они выбирались в диалоговом окне Добавление таблицы

Между окном конструктора Microsoft Access и окном приложения Microsoft Query (см., к примеру, рис. 5.11) существует несколько важных отличий.

- В окне конструктора Microsoft Access нет области данных. Чтобы увидеть данные, возвращаемые запросом, необходимо щелкнуть в панели инструментов на кнопке Запуск (Run) или щелкнуть на кнопке Вид (View) и выбрать в появившемся списке параметр Режим таблицы (Datasheet View). При выполнении запросов на выборку для отображения результирующего набора данных можно использовать любую из указанных кнопок, а вот при выполнении запросов других типов (например, на удаление, обновление или добавление) рекомендуется отдавать предпочтение кнопке Вид. Щелкнув на кнопке Вид, вы узнаете, как будут выглядеть данные после внесения изменений, однако сами изменения еще не будут произведены. В отличие от этого, щелчок на кнопке Запуск приводит к фактическому внесению изменений.
- В бланк запроса (окно конструктора Microsoft Access) можно добавить поля и указать, чтобы они *не* отображались в результирующем наборе данных. Предположим, что мы хотим отсортировать извлеченные записи по некоторому полю, но не хотим, чтобы запрос возвращал само это поле. В окне конструктора решить эту проблему совсем не сложно. Перетащите требуемое поле в бланк запроса и выберите в строке

Сортировка (Sort) параметр По возрастанию (Ascending) или По убыванию (Descending). Затем сбросьте флажок в строке Вывод на экран (Show). Указанное поле не будет отображаться в наборе данных, извлеченных запросом.

- Создавая запрос на основе внешнего объединения в Microsoft Access, не обязательно ограничиваться двумя таблицами. На рис. 5.16 показано, как построить запрос с использованием таблиц `Двери`, `ОбслуживаниеДверей` и `РасположениеДверей`. Данный запрос возвращает все записи из таблицы `Двери`, для которых существуют записи с аналогичным значением связующего поля в таблице `РасположениеДверей`, но не обязательно существуют записи в таблице `ОбслуживаниеДверей`. При этом для отображения расположения двери используются данные таблицы `РасположениеДверей`. Создать такой запрос в Microsoft Query было бы невозможно.

Рис. 5.16. В бланке запроса указано, из какой таблицы взято то или иное поле

- В Microsoft Access запрос может подсчитывать итоговые данные. Предположим, нас интересует, сколько раз проводился осмотр каждой двери. Перетащите в бланк запроса поле "КодДвери" из таблицы `Двери` и одноименное поле из таблицы `ОбслуживаниеДверей`. После этого щелкните на кнопке Групповые операции (Totals). В бланке запросов появится строка Групповая операция (Total). Щелкните в этой строке под именем поля "КодДвери" из таблицы `Двери` и выберите из раскрывающегося списка параметр Группировка (Group By). Затем щелкните под именем поля "КодДвери" из таблицы `ОбслуживаниеДверей`

и выберите из раскрывающегося списка параметр Count. Окончательный вид запроса показан на рис. 5.17. Щелкните на кнопке Запуск, и на экране появится результат выполнения запроса (рис. 5.18).

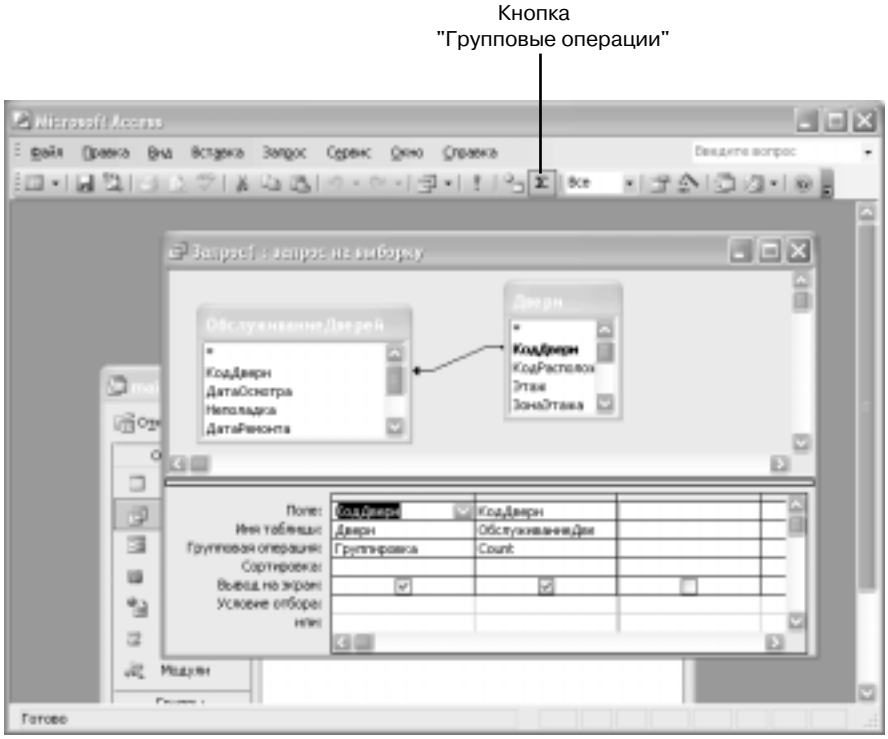


Рис. 5.17. В число других итоговых величин, которые могут возвращаться запросом, входят среднее, сумма и стандартное отклонение (параметр StDev)

Список возможностей конструктора запросов Access, приведенный выше, нельзя и близко назвать исчерпывающим. С помощью конструктора можно создать еще целый ряд запросов, построение которых было бы невозможным в Microsoft Query.

Создав запрос, сохраните его с помощью команды Файл⇨Сохранить (File⇨Save) или просто закройте окно запроса. Вам будет предложено сохранить изменения и присвоить новому запросу имя, которое по умолчанию выглядит не слишком содержательным — `Запрос1`, `Запрос2` и т.п.

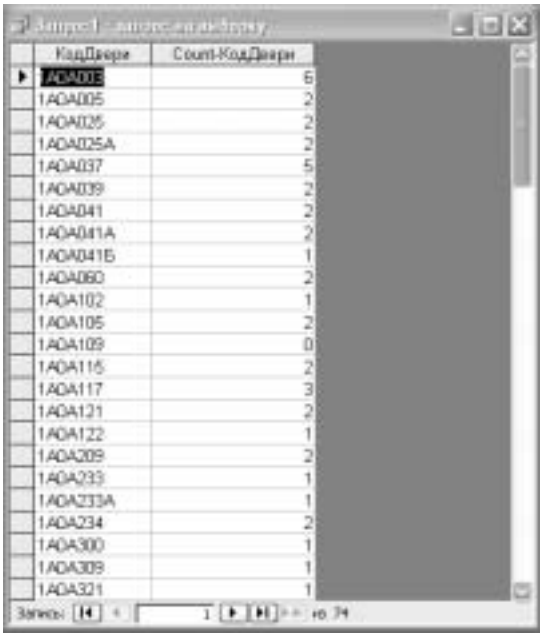


Рис. 5.18. Двери с кодом 1A0A109 в таблице `ОбслуживаниеДверей` не соответствовало ни одной записи. Вот почему в запросе на рис. 5.17 было использовано внешнее объединение

Импортирование результатов запроса базы данных с помощью Microsoft Query

После сохранения запроса базы данных его результат можно импортировать в диапазон внешних данных или сводную таблицу Excel. Делается это обычным способом, с помощью средства Microsoft Query, однако на сей раз запрос Microsoft Query применяется не к таблицам, а к запросу.

Предположим, что создан запрос, показанный на рис. 5.16, и ему присвоено имя `ЗапросДвери`. Для импортирования результатов выполнения запроса в Excel необходимо сделать следующее:

1. Откройте рабочую книгу Excel и выберите в меню команду Данные⇨Импорт внешних данных⇨Создать запрос (Data⇨Import External Data⇨New Database Query). В окне Выбор источника данных (Choose Data Source) выберите существующий источник данных или создайте новый. (Добавление к базе данных нового запроса никак не отражается на файле DSN, поэтому при наличии существующего источника данных создавать новый не нужно.)
2. В окне Добавление таблицы (Add Tables) выберите запрос `ЗапросДвери` и щелкните на кнопке Добавить (Add). Если в списке таблиц не указан запрос `ЗапросДвери`, щелкните на кнопке Параметры (Options). В поя-

вившемся диалоговом окне установите флажок Динамические наборы (Views) и, при желании, сбросьте флажок Таблицы (Tables), после чего щелкните на кнопке OK (рис. 5.19).

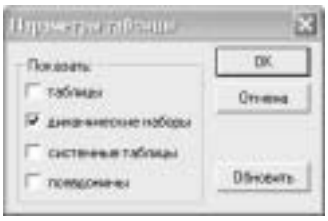


Рис. 5.19. Динамический набор — это фактически то же самое, что и запрос на выборку. В Microsoft Query эти понятия являются синонимами

3. Перетащите в область данных поля, которые следует импортировать из запроса `ЗапросДвери`[2]. Окно Microsoft Query приобретет вид, показанный на рис. 5.20.

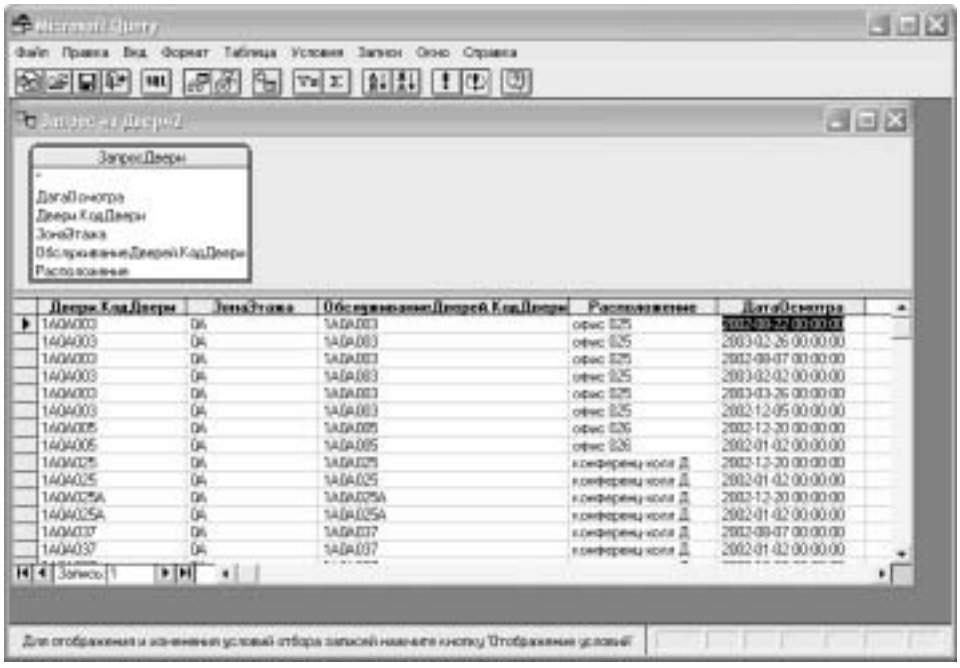


Рис. 5.20. Если в запросе встречаются одноименные поля из разных таблиц, к именам полей добавляются имена таблиц (например, `Двери.КодДвери`)

[2] *Если при попытке перетащить поле в область данных на экране появится сообщение "Слишком длинное имя поля", щелкните в верхней свободной ячейке области данных и выберите нужное поле из раскрывающегося списка.* — Прим. ред.

4. Щелкните на кнопке Вернуть данные (Return Data) или выберите в меню команду Файл⇨Вернуть данные в Microsoft Office Excel (File⇨Return Data to Microsoft Excel).

> **Примечание**
>
> Как уже отмечалось, *динамические наборы* аналогичны запросам на выборку. *Системные таблицы* применяются СУБД для отслеживания информации о таблицах, запросах и других объектах, созданных пользователем в базе данных. И наконец, *синонимы*, скорее всего, понадобятся только тем пользователям, которые ставят запросы к источникам данных Oracle.

На рис. 5.21 показаны данные, импортированные из запроса `ЗапросДвери` в рабочую книгу Excel.

	A	B	C	D	E	F	G
1	КодДвери	ЗонаЭтажа	КодДвери	Расположение	ДатаОсмотра		
59	1A5A043A	5A	1A5A043A	терраса юж. часть	01.01.2003 0:00		
60	1A5A043A	5A	1A5A043A	терраса юж. часть	20.03.2003 0:00		
61	1A3B127	3B	1A3B127	офис 301	28.02.2003 0:00		
62	1A3E256	3E	1A3E256	кафетерий 31	07.01.2003 0:00		
63	1A7E264	7E	1A7E264	кафетерий 71	17.03.2003 0:00		
64	1A6B264	6B		СУ центр			
65	1A3B321	3B	1A3B321	лестница зап. часть	07.01.2003 0:00		
66	1A7B453	7B	1A7B453	конференц-холл Т	10.01.2003 0:00		
67	1A4B453	4B	1A4B453	конференц-холл О	27.03.2003 0:00		
68	1A4B453	4B	1A4B453	конференц-холл О	08.01.2004 0:00		
69	1A7E217	7E	1A7E217	конференц-холл У	10.01.2003 0:00		
70	1A7E217	7E	1A7E217	конференц-холл У	13.03.2003 0:00		
71	1A2B706	2B	1A2B706	СУ сев. часть	06.01.2003 0:00		
72	1A3B713	3B	1A3B713	СУ вост. часть	27.02.2003 0:00		
73	1A2F305	2F	1A2F305	конференц-холл К	06.01.2003 0:00		
74	1A2F319	2F		конференц-холл К			
75	1A2F321	2F	1A2F321	конференц-холл К	07.01.2003 0:00		
76	1A2F333	2F		конференц-холл К			
77	1A2F335	2F	1A2F335	конференц-холл К	06.01.2003 0:00		
78	1A2F217	2F	1A2F217	конференц-холл К	06.01.2003 0:00		
79	1A3B153	3B	1A3B153	офис 301	28.02.2003 0:00		
80	1A3B145	3B	1A3B145	офис 301	28.02.2003 0:00		
81	1A4A230	4A	1A4A230	офис 415	26.03.2003 0:00		
82	1A4B456	4B	1A4B456	конференц-холл О	20.03.2003 0:00		
83	1A2B301	2B	1A2B301	лестница сев. часть	27.02.2003 0:00		
84	1A4A2425	4A	1A4A2425	СУ юж. часть	26.03.2003 0:00		
85	1A7E236	7E	1A7E236	конференц-холл У	17.03.2003 0:00		
86	1A7E421	7B	1A7E421	складск. помещение 71	17.03.2003 0:00		

Рис. 5.21. Обратите внимание на пустые ячейки в некоторых записях, извлеченных из таблицы `ОбслуживаниеДверей`. Появление таких записей стало возможным благодаря применению внешнего объединения

Работа с диапазоном данных

Получив диапазон внешних данных, можно настроить его свойства. Щелкните правой кнопкой мыши на любой ячейке диапазона внешних данных и выберите в появившемся контекстном меню команду Свойства диапазона

данных (Data Range Properties). На экране появится диалоговое окно Свойства внешнего диапазона (External Data Range Properties), показанное на рис. 5.22.

> **Примечание**
>
> В приложениях Microsoft Office для описания характеристик объектов применяется термин *свойства*. К примеру, у диапазона ячеек есть свойство Граница (Borders), которое определяет, какие границы (верхняя, нижняя, левая, правая, диагональная или нет границы) имеются у диапазона. Для большей наглядности свойство можно представить себе в качестве параметра.

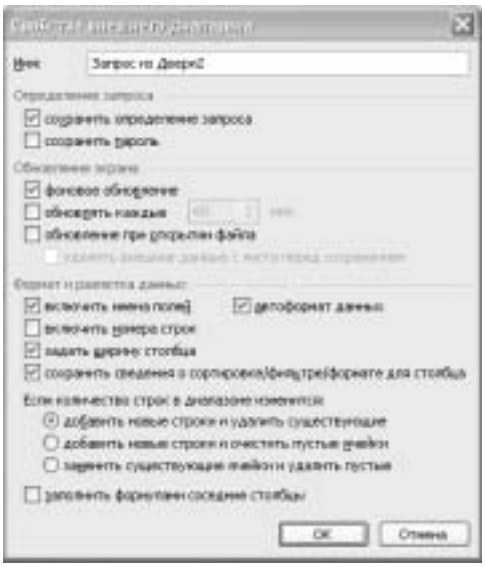


Рис. 5.22. Если в контекстном меню нет команды Свойства диапазона данных, значит, вы щелкнули за пределами диапазона внешних данных

Среди свойств, показанных на рис. 5.22, наибольшую ценность для управления данными представляют следующие.

- **Сохранить определение запроса (Save Query Definition).** Сбрасывайте этот флажок только в том случае, если точно знаете, что делаете. Сбросив флажок, вы больше не сможете обновлять данные запроса или даже редактировать запрос. Если сбросить флажок Сохранить определение запроса и сохранить рабочую книгу, запрос будет потерян навсегда.
- **Сохранить пароль (Save Password).** Как правило, этот флажок устанавливают при работе в сетевом окружении. (Действительно, от кого защищать данные пользователю, который работает на изолированном компьютере? Неужели от себя самого?) Как отмечалось ранее, пароль можно сохранить в файле DSN, который определяет источник данных, — его тип,

расположение и т.п. К сожалению, файл DSN является простым текстовым файлом, поэтому извлечь из него пароль не составляет никакого труда. В отличие от этого, определение запроса сохраняется в рабочей книге под скрытым именем, взломать которое намного труднее.

- **Фоновое обновление (Enable Background Refresh).** Если установить этот флажок, обновление диапазона внешних данных будет выполняться незаметно, не прерывая обычной работы пользователя в Excel, например, создание формул или сводных таблиц. Если же флажок будет сброшен, пользователю придется ждать, пока не закончится обновление запроса. Как правило, это свойство применяется лишь тогда, когда сам источник данных обновляется довольно часто, а выполнение запросов занимает слишком много времени.
- **Обновление при открытии файла (Refresh Data on File Open).** При установленном флажке Обновление при открытии файла Excel будет повторно выполнять запрос при каждом открытии рабочей книги, чтобы заполнить ее самыми свежими данными. Единственное, что может омрачить применение данного флажка, — наличие обработчика событий VBA, запускающегося при открытии рабочей книги. В подобных случаях следует помнить, что обработчик события `Open` выполняется *перед* автоматическим обновлением при открытии файла.

➔ Более подробно об обновлении данных рассказано в главе 4, "Импорт данных: обзор", раздел "Импорт данных в сводные таблицы".

Прежде чем переходить к рассмотрению следующих четырех свойств, показанных на рис. 5.22, предположим, что диапазон внешних данных занимает на рабочем листе ячейки `A1:B5` (рис. 5.23).

	A	B	C
1	**ДатаОсмотра**	**Расположение**	Строка 1
2	02.01.2002	Цитовая	Строка 2
3	02.01.2002	Цитовая	Строка 3
4	02.01.2002	Цитовая	Строка 4
5	02.01.2002	Цитовая	Строка 5
6	Столбец А	Столбец В	Строка 6
7			Строка 7
8			
9			
10			

Рис. 5.23. Исходный диапазон внешних данных выделен темным цветом

На рис. 5.23 границы исходного диапазона внешних данных отмечены фразами "Строка 1", "Строка 2" и т.п. в столбце `C`, а также "Столбец А" и "Столбец В" в строке 6. При обновлении указанного диапазона количество записей, возвращаемых запросом, может увеличиться или уменьшиться. На следующих шести рисунках показано, какой эффект на ячейки рабочего листа будет оказывать то или иное свойство при обновлении исходного диапазона.

- **Добавить новые строки и удалить существующие (Insert Cells for New Data, Delete Unused Cells).** Если в диапазоне внешних данных появятся две новые записи, ячейки столбцов A и B, начиная со строки 6, будут сдвинуты на две строки вниз. Ячейки столбцов C и далее до конца затронуты не будут. Другими словами, ячейки со значениями "Столбец A" и "Столбец B" из строки 6 переместятся в строку 8, чтобы освободить место для двух новых записей, а все ячейки столбца C останутся на своем месте (рис. 5.24). Если же в результате обновления в исходном диапазоне данных станет на две записи меньше, ячейки, находящиеся ниже строки 5, "подтянутся" вверх на место исчезнувших записей. Опять-таки, данная операция затронет только столбцы A и B. Ячейки со значениями "Столбец A" и "Столбец B" из строки 6 переместятся в строку 4. Ячейки столбца C, как и раньше, затронуты не будут (рис. 5.25).

	A	B	C
1	ДатаОсмотра	Расположение	Строка 1
2	02.01.2002	Щитовая	Строка 2
3	02.01.2002	Щитовая	Строка 3
4	02.01.2002	Щитовая	Строка 4
5	02.01.2002	Щитовая	Строка 5
6	02.01.2002	НОВАЯ	Строка 6
7	02.01.2002	НОВАЯ	Строка 7
8	Столбец A	Столбец B	
9			
10			

Рис. 5.24. Помимо всего прочего для исходного диапазона данных был установлен флажок Автоформат данных (Preserve Cell Formatting)

	A	B	C
1	ДатаОсмотра	Расположение	Строка 1
2	02.01.2002	НОВАЯ	Строка 2
3	02.01.2002	НОВАЯ	Строка 3
4	Столбец A	Столбец B	Строка 4
5			Строка 5
6			Строка 6
7			Строка 7
8			
9			
10			

Рис. 5.25. В результате повторного выполнения запроса из диапазона данных исчезли четыре старые записи и добавились две новые

- **Добавить новые строки и очистить пустые ячейки (Insert Entire Rows for New Data, Clear Unused Cells).** При появлении в диапазоне двух новых записей все строки рабочего листа, начиная со строки 5, будут сдвинуты на две строки вниз. Как следствие этого, ячейки C5:IV6 окажутся пустыми. На рис. 5.26 показано, что ячейки столбца C, содержащие значения "Строка 5", "Строка 6" и "Строка 7", а также ячейки со значениями "Столбец A" и "Столбец B" сместились вниз из-за вставки двух новых

строк (хотя эти строки были вставлены совсем не в том месте, где можно было ожидать). Если же в результате выполнения запроса вместо четырех первоначальных записей будут возвращены только две, ячейки A4:B5 будут очищены. Как видно из рис. 5.27, ячейки со значениями "Столбец A" и "Столбец B" останутся на своем месте, потому что неиспользованные ячейки не будут удалены. Следует также отметить, что Excel очистил только содержимое ячеек A4:B5, но не их формат. Поскольку ни ячейки, ни строки не были удалены, все значения, находящиеся за пределами диапазона внешних данных, останутся на своих местах.

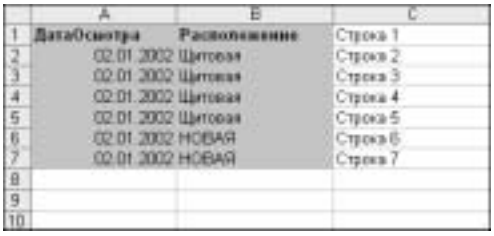
	A	B	C
1	**ДатаОсмотра**	**Расположение**	Строка 1
2	02.01.2002	Цитовая	Строка 2
3	02.01.2002	Цитовая	Строка 3
4	02.01.2002	Цитовая	Строка 4
5	02.01.2002	Цитовая	
6	02.01.2002	НОВАЯ	
7	02.01.2002	НОВАЯ	Строка 5
8	Столбец A	Столбец B	Строка 6
9			Строка 7
10			

Рис. 5.26. Обратите внимание на следующее несоответствие: новые строки в столбце C были вставлены между строками 4 и 5, а новые записи в столбцах A и B — между строками 5 и 6

	A	B	C
1	**ДатаОсмотра**	**Расположение**	Строка 1
2	02.01.2002	НОВАЯ	Строка 2
3	02.01.2002	НОВАЯ	Строка 3
4			Строка 4
5			Строка 5
6	Столбец A	Столбец B	Строка 6
7			Строка 7
8			
9			
10			

Рис. 5.27. Неиспользованные ячейки диапазона внешних данных не удаляются, а очищаются, однако определение диапазона изменяется на A1:B3

- **Заменить существующие ячейки и удалить пустые (Overwrite Existing Cells with New Data, Clear Unused Cells).** Для размещения новых записей не будут вставляться дополнительные строки или ячейки. Если ниже диапазона внешних данных находятся другие данные (что обычно является следствием неправильной организации рабочего листа), то они будут перезаписаны новым содержимым диапазона, И ВАС ОБ ЭТОМ НЕ ПРЕДУПРЕДЯТ. На рис. 5.28 показано, что, в результате появления в диапазоне двух новых записей, с рабочего листа исчезли значения "Столбец A" и "Столбец B". С содержимым столбца C ничего не случилось, потому что в столбце C не появилось новых данных. Если же запрос возвратит меньше записей, чем раньше, неиспользованные ячейки будут очищены, а остальные данные не изменятся.



	A	B	C
1	ДатаОсмотра	Расположение	Строка 1
2	02.01.2002	Щитовая	Строка 2
3	02.01.2002	Щитовая	Строка 3
4	02.01.2002	Щитовая	Строка 4
5	02.01.2002	Щитовая	Строка 5
6	02.01.2002	НОВАЯ	Строка 6
7	02.01.2002	НОВАЯ	Строка 7
8			
9			
10			

Рис. 5.28. Будьте осторожны, размещая информацию под диапазоном внешних данных; она может быть уничтожена

- **Заполнить формулами соседние столбцы (Fill Down Formulas in Columns Adjacent to Data).** Используя синтаксис формул и встроенные функции диспетчера баз данных, в запрос можно вставлять вычисляемые поля, но это не так уж и просто. Наличие флажка Заполнить формулами соседние столбцы позволяет обойтись без вставки вычисляемых полей. Просто создайте формулу на рабочем листе и поместите ее в столбец, который слева или справа прилегает к диапазону внешних данных. Затем установите указанный флажок. Если при обновлении диапазона внешних данных в нем появятся новые записи, Excel автоматически скопирует формулы, находящиеся в смежном столбце, в новые ячейки столбца вплоть до последней строки, занятой диапазоном внешних данных.

Управление полями типа "Логический" и флажками

Одна из проблем, связанных с импортированием содержимого базы данных на рабочий лист, касается полей типа Логический (Boolean). Поля этого типа еще называют полями ИСТИНА/ЛОЖЬ (True/False), а в Microsoft Access они зачастую фигурируют как поля Да/Нет (Yes/No).

Поля типа Логический могут принимать одно из двух значений: ИСТИНА или ЛОЖЬ. При извлечении логических значений из базы данных в Excel начинаются трудности, потому что в базах данных содержимое полей типа Логический не всегда хранится в виде значений ИСТИНА и ЛОЖЬ. Это следует учитывать либо при построении базы данных, либо при импортировании данных в Excel. Более того, в базах данных значения полей типа Логический часто представляют флажками, у которых помимо состояний ИСТИНА (флажок установлен) и ЛОЖЬ (флажок сброшен) может быть еще и третье состояние, NULL (флажок затенен).

Извлечение логических значений на рабочий лист

Допустим, в базе данных есть таблица `ПрименениеУспокоительных` с полем типа Логический (Boolean) под названием "МедицинскиеПоказания". На рис. 5.29 показано, как выглядит это поле в Microsoft Access при выборе вида Режим таблицы (Datasheet View).

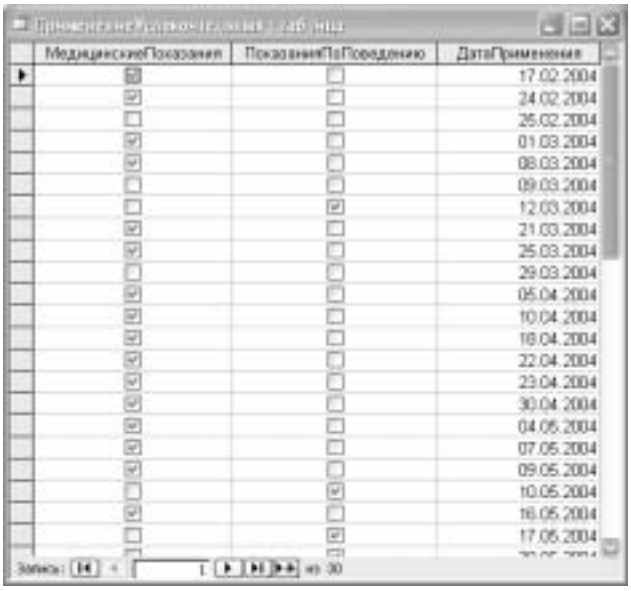


Рис. 5.29. В режиме таблицы значения полей типа Логический отображаются в виде флажков

На рис. 5.30 показано, как выглядит поле "МедицинскиеПоказания" на рабочем листе Excel в зависимости от того, каким методом оно было извлечено из базы данных.

	A	B	C	D
1	МедицинскиеПоказания		МедицинскиеПоказания	
2	ИСТИНА		1	
3	ИСТИНА		1	
4	ЛОЖЬ		0	
5	ИСТИНА		1	
6	ИСТИНА		1	
7	ЛОЖЬ		0	
8	ЛОЖЬ		0	
9	ИСТИНА		1	
10	ИСТИНА		1	
11	ЛОЖЬ		0	
12	ИСТИНА		1	
13	ИСТИНА		1	
14	ИСТИНА		1	
15	ИСТИНА		1	
16	ИСТИНА		1	
17	ИСТИНА		1	
18	ИСТИНА		1	
19	ИСТИНА		1	
20	ИСТИНА		1	
21	ЛОЖЬ		0	
22	ИСТИНА		1	
23	ЛОЖЬ		0	
24	ЛОЖЬ		0	

Рис. 5.30. Если вставить поле Access на рабочий лист с помощью команды Вставка (Paste), имя поля будет рассматриваться как название столбца, а к ячейке будет применена заливка серым цветом

- Если выделить поле в таблице Access, скопировать его, открыть рабочую книгу Excel и вставить поле на рабочий лист, значения поля будут представлены как ИСТИНА и ЛОЖЬ (столбец A на рис. 5.30).
- Если извлечь поле с помощью команды Данные⇨Импорт внешних данных⇨Создать запрос (Data⇨Import External Data⇨New Database Query), значения ИСТИНА будут заменены единицами, а значения ЛОЖЬ — нулями (столбец C на рис. 5.30).

Если нет желания работать с единицами и нулями, попробуйте выполнить обновление диапазона внешних данных с помощью процедуры VBA, — скажем, следующего обработчика события Open[3].

```
Private Sub Workbook_Open()
Application.Goto Reference:="Запрос_из_Медицинские_показания"
Selection.QueryTable.Refresh BackgroundQuery:=False
Application.Goto Reference:="Запрос_из_Медицинские_показания"
With Selection
    .Replace What:="0", Replacement:="ЛОЖЬ", LookAt:=xlWhole
    .Replace What:="1", Replacement:="ИСТИНА", LookAt:=xlWhole
End With
End Sub
```

Данная процедура выделяет существующий диапазон данных и обновляет его содержимое. Затем она еще раз выделяет диапазон внешних данных на случай, если в результате обновления его размер изменился. И наконец, она заменяет нули на значение ЛОЖЬ, а единицы на значение ИСТИНА. Замена выполняется только тогда, когда 0 или 1 являются полным значением ячейки. Это позволяет избежать, скажем, замены значения "МедицинскиеПоказания1" на "МедицинскиеПоказанияИСТИНА".

Управление состояниями флажков

Если флажки в формах данных Access соответствуют полям типа Логический (Boolean), импортирование данных не вызывает особых трудностей. Но иногда требуется, чтобы поля принимали не только значения ЛОЖЬ или ИСТИНА, но и значение NULL. Последнее может означать что-то вроде "Не применимо".

Представим себе, что поле "МедицинскиеПоказания" в таблице ПрименениеУспокоительных должно принимать одно из трех значений: ИСТИНА, если пациенту дали успокоительное по медицинским показаниям, ЛОЖЬ, если успокоительное дали по каким-либо другим показаниям, и NULL, если успокоительное не было использовано вообще. Если тип данных поля "МедицинскиеПоказания" определить как Логический, Access не сможет отличить значение ЛОЖЬ от NULL; и в том и в другом случае флажок будет сброшен.

---

[3] *Предполагается, что для импортирования данных в столбец C был создан источник данных под названием Медицинские показания.* — Прим. ред.

Если же определить тип данных поля “МедицинскиеПоказания” как Числовой (Integer), оно сможет принимать гораздо больше значений. Такое поле можно связать с флажком формы данных, для которого установлено свойство Тройное состояние (Triple State). Флажок с подобным свойством может иметь три состояния: установлен (ИСТИНА), сброшен (ЛОЖЬ) или затенен (какое-нибудь другое значение, обычно “Не применимо”). Пример описанной формы данных показан на рис. 5.31.

Рис. 5.31. Затененный флажок соответствует значению NULL

Если изменить тип данных поля Логический на Числовой, значение ИСТИНА превратится в -1 (а не в 1), а значение ЛОЖЬ — в 0. Если же включить свойство Тройное состояние для флажка, связанного с полем “МедицинскиеПоказания”, при затененном флажке в поле будет сохраняться значение NULL.

На рис. 5.32 показано, как выглядит диапазон внешних данных, если тип данных поля определен как Числовой, а значения поля определяются состоянием флажка формы данных. В ячейках A1:A11 содержится 10 значений. Пять единиц означают, что для соответствующих записей флажок формы данных был установлен. Пять нулей означают, что флажок формы данных был сброшен.

	A	B	C	D	E
1	МедицинскиеПоказания			МедицинскиеПоказания	
2	1		0	-1	
3	1		1	-1	
4	1		2	-1	
5	1		3	0	
6	1		4	0	
7	0		5	0	
8	0		6		
9	0		7		
10	0		8		
11	0		9		
12					
13					

Рис. 5.32. Чтобы отличить значения NULL от пустых ячеек, которые не входят в диапазон внешних данных, откройте окно свойств диапазона и установите флажок Включить номера строк (Include Row Numbers)

В ячейках D1:D11 также содержится 10 значений. Число -1 означает, что флажок был установлен, число 0 — что флажок был сброшен. И наконец, пустые ячейки в строках с 8 по 11 означают, что флажок был затенен.

Если применить обработчик события Open, который при обновлении диапазона внешних данных заменяет 0 на ЛОЖЬ, а -1 на ИСТИНА, нужно следить за тем, чтобы обработчик по ошибке не заменил номер строки 0, показанный на рис. 5.32 в столбце C. Для этого текст процедуры, приведенный ранее, рекомендуется переписать следующим образом:

```
Private Sub Workbook_Open()
Dim ExtRange As Range
Dim NCols as Integer, NRows as Integer

Set ExtRange = _
Worksheets("Лист1").Names("Запрос_из_Медицинские_показания2") _
.RefersToRange
NCols = ExtRange.Columns.Count-1
NRows = ExtRange.Rows.Count
Application.Goto Reference:="Запрос_из_Медицинские_показания2"
Selection.QueryTable.Refresh BackgroundQuery:=False
ExtRange.Offset(0, 1).Resize(NRows, NCols).Select
With Selection
    .Replace What:="0", Replacement:="ЛОЖЬ", LookAt:=xlWhole
    .Replace What:="-1", Replacement:="ИСТИНА", LookAt:=xlWhole
    .Replace What:="", Replacement:="#Н/Д", LookAt:=xlWhole
End With
End Sub
```

Процедура, приведенная выше, подсчитывает число строк и столбцов диапазона внешних данных, после чего выделяет диапазон ячеек, смещенный относительно диапазона внешних данных на один столбец вправо. При этом новый диапазон содержит столько же строк, сколько и диапазон внешних данных, но на один столбец меньше. В выделенном диапазоне значения -1 заменяются на ИСТИНА, значения — 0 на ЛОЖЬ, а пустые значения — на значения-ошибки #Н/Д (#N/A).

Что читать дальше

В этой главе обсуждалось использование средства Microsoft Query для импортирования внешних данных в рабочую книгу Excel. Основное внимание в этой, как и в предыдущей главе, было сосредоточено на применении команды Данные⇨Импорт внешних данных⇨Создать запрос (Data⇨Import External Data⇨New Database Query).

Между тем, в меню Данные (Data) есть и другие команды, позволяющие управлять данными с помощью сводных таблиц, группировать их по полям различных типов и создавать Web-запросы. Обо всем этом речь пойдет в главе 6, “Импорт данных: еще несколько соображений”.