ГЕННАДИЙ ГУРВИЦ

bhv®

MICROSOFT® ACCESS 2010

РАЗРАБОТКА ПРИЛОЖЕНИЙ НА РЕАЛЬНОМ ПРИМЕРЕ

PRO
ПРОФЕССИОНАЛЬНОЕ ПРОГРАММИРОВАНИЕ

- ПРОСТОЕ ACCESS-ПРИЛОЖЕНИЕ
- ПРОФЕССИОНАЛЬНОЕ ACCESS-ПРИЛОЖЕНИЕ
- ПЕРЕВОД ПРОГРАММНОГО КОМПЛЕКСА В АРХИТЕКТУРУ «КЛИЕНТ-СЕРВЕР»
- ОБЕСПЕЧЕНИЕ ИНФОРМАЦИОННОЙ БЕЗОПАСНОСТИ ПРИЛОЖЕНИЯ И БАЗЫ ДАННЫХ
- ПРИМЕНЕНИЕ В РАЗРАБОТКАХ ГОТОВЫХ РЕЦЕПТОВ
- ВЫХОД ЗА РАМКИ СТАНДАРТНОГО ИСПОЛЬЗОВАНИЯ ПРОДУКТА

+CD

Геннадий Гурвиц

MICROSOFT® ACCESS 2010

РАЗРАБОТКА ПРИЛОЖЕНИЙ НА РЕАЛЬНОМ ПРИМЕРЕ

Санкт-Петербург
«БХВ-Петербург»
2010

УДК 681.3.06
ББК 32.973.26-018.2
Г95

Гурвиц Г. А.

Г95 Microsoft® Access 2010. Разработка приложений на реальном примере. — СПб.: БХВ-Петербург, 2010. — 496 с.: ил. + CD-ROM — (Профессиональное программирование)

ISBN 978-5-9775-0579-6

Рассматриваются этапы создания приложений баз данных в файл-серверной и клиент-серверной архитектурах. Описывается работа с Microsoft Access 2010 (клиент) и Microsoft SQL Server 2008 (сервер). На примере небольшой, но реальной базы данных показан процесс создания простого Access-приложения и выполнена его модификация, придавшая приложению основные черты профессиональной разработки. Применен предложенный ранее автором метод оформления интерфейса приложения — метод пересекающихся каскадов. Даны практические приемы перевода созданного программного комплекса в архитектуру "клиент-сервер".

На прилагаемом компакт-диске содержится реальное приложение в двух вариантах: локальном и в архитектуре "клиент-сервер", а также 50 вариантов заданий для курсового проекта на разработку прикладного программного обеспечения.

Для студентов, преподавателей, программистов и разработчиков баз данных

УДК 681.3.06
ББК 32.973.26-018.2

Группа подготовки издания:

Главный редактор	*Екатерина Кондукова*
Зам. главного редактора	*Евгений Рыбаков*
Зав. редакцией	*Григорий Добин*
Редактор	*Анна Кузьмина*
Компьютерная верстка	*Ольги Сергиенко*
Корректор	*Зинаида Дмитриева*
Дизайн серии	*Инны Тачиной*
Оформление обложки	*Елены Беляевой*
Зав. производством	*Николай Тверских*

Лицензия ИД № 02429 от 24.07.00. Подписано в печать 30.04.10.
Формат 70×100 1/16. Печать офсетная. Усл. печ. л. 39,99.
Тираж 2000 экз. Заказ №
"БХВ-Петербург", 190005, Санкт-Петербург, Измайловский пр., 29.

Санитарно-эпидемиологическое заключение на продукцию
№ 77.99.60.953.Д.005770.05.09 от 26.05.2009 г. выдано Федеральной службой
по надзору в сфере защиты прав потребителей и благополучия человека.

Отпечатано с готовых диапозитивов
в ГУП "Типография "Наука"
199034, Санкт-Петербург, 9 линия, 12

ISBN 978-5-9775-0579-6

Оглавление

Введение

Microsoft Office Access 2010 — последняя версия продукта корпорации Microsoft, который представляет на сегодняшний день самую распространенную в мире систему управления реляционными базами данных для персональных компьютеров.

Microsoft Office, в состав которого входит Access, совершенствуется уже около двадцати лет. Первая версия пакета под номером 3.0 вышла еще в 1992 году. В набор программ тогда входили текстовый процессор Word 2.0, табличный процессор Excel 4.0, презентационное приложение PowerPoint 3.0 и почтовая программа Mail.

Дальнейшая история развития пакета выглядит следующим образом:

- 1993 год — Microsoft Office 4.0 (Word 6.0, Excel 4.0, PowerPoint 3.0, Access 1.0);
- 1994 год — Microsoft Office 4.3 (Word 6.0, Excel 5.0, PowerPoint 4.0, Access 2.0);
- 1995 год — Microsoft Office 95 (Word 95, Excel 95, Access 95 и др.), кодовое название — Office 7, в продолжение линейки Word;
- 1997 год — Microsoft Office 97 (Word 97, Excel 97, Access 97 и др.), кодовое название — Office 8;
- 1999 год — Microsoft Office 2000 (Word 2000, Excel 2000, Access 2000 и др.), кодовое название — Office 9;
- 2001 год — Microsoft Office XP (Word 2002, Excel 2002, Access 2002 и др.), кодовое название — Office 10;
- 2003 год — Microsoft Office 2003 (Word 2003, Excel 2003, Access 2003 и др.), кодовое название — Office 11;
- 2007 год — Microsoft Office 2007 (Word 2007, Excel 2007, Access 2007 и др.), кодовое название — Office 12;
- 2010 год — Microsoft Office 2010 (Word 2010, Excel 2010, Access 2010 и др.); кодовое название — Office 14.

Называть следующий офисный пакет Office 13 в Microsoft не решились из-за негативных ассоциаций, связанных с числом 13. Однако по старой традиции пакет Office 2010 получил следующий, 14 порядковый номер. Все это, разумеется, субъективные мелочи, не имеющие никакого отношения к наполнению пакета.

Известно, что каждые два года мощность персональных компьютеров удваивается, а каждые три-четыре года меняется сама концепция создания прикладного программного обеспечения, работающего с базами данных. Рассмотрим ее изменение на примере MS Access.

- MS Access 97/2000. Повсеместное использование встроенных средств продукта: стандартных форм, линейки прокрутки, конструктора запросов, отчетов и т. д. Отношение к нему, в большей степени, как к удобной игрушке, позволяющей автоматизировать работу маленького предприятия.
- MS Access 2002/2003. Отсутствие кода VBA — уже не просто плохой тон. Access-приложение без него просто не мыслится. Заказчик быстро понял все преимущества персональной разработки.
- MS Access 2007 оставлен в составе пакета MS Office, но значительно усилен сервером баз данных MS SQL Server 2005 Express Edition. С применением этой связки продуктов отныне можно без труда создавать решения масштаба предприятия.

С выходом в свет Microsoft Office Access 2010 в арсенал разработчиков к комбинации мощных современных технологий и развитых средств для создания прикладных программ нового поколения добавились новые возможности. Однако следует отметить, что каких-либо революционных изменений последняя версия продукта не содержит. Именно поэтому в среде разработчиков, применяющих в своей работе Microsoft Office Access 2010, распространилось его шутливое название "старый знакомый в новом костюме".

Для изучения проблем, с которыми не раз доводилось сталкиваться в процессе работы с новым программным продуктом, лучше всего подходит конкретное, полностью законченное приложение. Именно на этом и сделан особый акцент в данной книге.

От получения задания до сдачи законченной разработки базы данных заказчику описаны все моменты работы. Надеюсь, что вы найдете в ней простые приемы создания приложений баз данных, позволяющие избежать тупиковых решений и ненужных усилий, обычно ведущих к напрасной потере времени.

Перед вами руководство по быстрому освоению базовых возможностей Microsoft Access 2010. Поэтому не ищите в нем подробных экскурсов в теорию программирования. Сначала мы просто создадим довольно несложное приложение, и в процессе работы вы получите первый опыт использования возможностей рассматриваемой СУБД (системы управления базами данных).

Затем мы выполним модификацию нашего первого приложения, придав ему основные черты профессиональной разработки. В нем появится меню в стиле Stage,

а параллельно и знаменитая лента пользователя, многостраничные формы, объекты, контролирующие права доступа к расчетам, которые выполняет программный комплекс, а также отчеты, обеспечивающие передачу данных в другие продукты Microsoft Office 2010, система контекстуально-зависимой помощи, развитый механизм поиска данных и другие необходимые процедуры, делающие работу конечного пользователя более комфортной.

На третьем этапе мы рассмотрим процесс перевода нашего приложения на платформу "клиент-сервер". Общепризнано, что рассматриваемый в книге Microsoft SQL Server является самой выгодной серверной СУБД среди существующих в мире, исходя из соотношения цены и качества. Информационные системы уровня предприятия, построенные с использованием Microsoft SQL Server, выгодно отличаются невысокой суммарной стоимостью владения, а богатые возможности этой СУБД являются одним из самых важных критериев при выборе продукта, который будет использоваться на предприятии при построении баз данных.

"С появлением Windows 2000 и SQL Server 7.0 компания Microsoft сделала первый и очень важный шаг в реализации своей мечты о таком программном обеспечении, которое можно было бы горизонтально масштабировать от Web-уровня до уровня приложения и уровня базы данных, — сказал Билл Гейтс, главный архитектор программного обеспечения Microsoft. — Феноменальные результаты эталонных тестов доказывают правильность нашей стратегии, и эти новшества принципиально изменят игровое поле для платформ приложений масштаба предприятия"[1].

В последние годы быстрыми темпами растет число приложений и баз данных, созданных в архитектуре "клиент-сервер", которая стала применяться в середине 80-х годов в мини-компьютерах и мэйнфреймах. Начиная с 90-х она активно вторглась и в мир персональных компьютеров. Разработка баз данных в этой архитектуре — очень сложный и длительный процесс.

В книге вашему вниманию будет предложен самый легкий способ создания базы данных Microsoft SQL Server — конвертация созданной и успешно работающей базы Microsoft Access в Microsoft SQL Server средствами Access. Процесс конвертации данных займет несколько минут. Перевод же файл-серверного приложения на платформу "клиент-сервер" — куда более трудоемкий процесс, осложняющийся еще и тем, что в существующей литературе описаны проблемы, уже решенные другими разработчиками, а в реальности вы, скорее всего, столкнетесь с такими задачами, которые еще никто никогда не решал. Рекомендации этой книги помогут начать работу, а достичь высшей ступени мастерства вы сможете только на основании личного опыта. Не секрет, что для превращения новичка в профессионала необходимо три-четыре года напряженной работы и несколько приложений, понравившихся пользователю.

---

[1] См. **http://www.microsoft.com/presspass/press/2000/Feb00/SQLSvr2000PR.mspx**.

В последнее время появились и обнадеживающие вести. Работая над этой книгой, автор нашел в официальном блоге Microsoft Access Team Blog сообщение с заголовком "Access 15 and SQL Server". В самой статье сообщалось: "Привет всем! Так как Office 2010 близок к финальному релизу, мы начинаем работу над Office 15. Одна из сфер, которая нуждается в улучшении, — это поддержка продукта SQL Server. Основываясь на информации, которую удалось собрать от пользователей, надо сказать, что эта сфера нуждается в значительной переработке"[2]. Не исключено, что с выходом в свет Office 15 трудоемкий процесс перевода файл-серверного приложения на платформу "клиент-сервер" станет и вовсе обыденной вещью, не требующей вмешательства разработчика!

Занимаясь созданием прикладного программного обеспечения деятельности небольших предприятий на протяжении многих лет, могу отметить, что почти всегда заказчик сам не знает, чего хочет, и постановку задачи приходится воспринимать на слух, в процессе работы неоднократно уточняя те или иные моменты создаваемой программы. Более того, при очередной встрече с заказчиком, связанной с демонстрацией уже выполненных этапов, очень часто открываешь для себя все новые и новые горизонты предстоящей работы, требующей существенного изменения как структуры данных, так и интерфейса будущего приложения. Но это не самый худший вариант. Иногда уже через день после сдачи разработки заказчик переосмысливает свои цели, после чего задача меняется коренным образом, и следующий визит заставляет начать всю работу заново. Именно по этой причине я рекомендую вам, внимательно выслушав заказчика, попросить его описать задачу в письменном виде, на основании чего самостоятельно сформулировать постановку задачи и еще раз обсудить ее. Уверяю вас, если результат окажется положительным, то это будет признанием того, что ваш работодатель действительно нуждается в заказанном программном обеспечении, а самое главное — знает, чего хочет.

Существует известная поговорка: "Лучше один раз увидеть, чем сто раз услышать". Подавляющее большинство людей быстрее учатся, воспринимая материал визуально. Конечно, у вас будет достаточное количество изображений диалоговых окон, чтобы помочь воспринять визуально применяемые методы, но также вам будут предложены все аспекты разработки реального приложения.

Книги других авторов тоже претендуют на то, что в них приведены настоящие примеры. Но в этой вы найдете полное руководство разработчика от постановки задачи до создания файла-справки к приложению. Главный акцент в книге сделан на поэтапное — шаг за шагом — освоение нового, встречающегося на пути конкретной реальной разработки, и ни одного, даже самого маленького, шага в сторону.

---

[2] См. **http://blogs.msdn.com/access/archive/2009/12/03/access-15-and-sql-server.aspx**.

Кому адресована эта книга

При написании книги преследовались две цели. Она должна быть максимально полезной для всех категорий пользователей и при этом не выходить за рамки небольшого объема. Пользователю не придется просматривать сотни страниц дополнительной литературы, отыскивая ту информацию, которая нужна в данный момент. Все, что необходимо для разработки реального приложения, находится под рукой.

Часть I книги, которую вы держите в руках, не предполагает у читателя какого-либо опыта разработки приложений для работы с базами данных и предназначена для начинающих. *Часть II*, благодаря подробному описанию процесса создания реального программного обеспечения деятельности небольшого предприятия, поможет широкому кругу разработчиков, имеющих некоторый опыт программирования, быстрее освоить новую для них среду и начать зарабатывать своим умом. *Часть III* — для искушенных пользователей MS Access, столкнувшихся с проблемой создания многопользовательского приложения и не имеющих времени на изучение Visual Studio 2010 или не желающих "мигрировать" на другую платформу.

Наличие большого числа вариантов заданий, описывающих деятельность небольших организаций, делает книгу полезной для студентов и преподавателей высших учебных заведений.

Структура книги

Данная книга состоит из трех частей и приложения.

- **Часть I. Ваше первое приложение на MS Access 2010.**

  В которой вы научитесь делать то, что можно, и так, как нужно.

  - В *главе 1 "Введение в базы данных"* описываются основные понятия баз данных и систем управления базами данных. Дана характеристика организации информационной системы для архитектуры "клиент-сервер".
  - В *главе 2 "Разработка базы данных"* описан процесс создания реляционной базы данных Microsoft Access от постановки задачи до окончательного проекта. В ней вы узнаете, как создать хороший проект базы данных без особых усилий, как внести изменения в базу, а также о возможных последствиях этого, если в таблицы уже занесены некоторые данные. Самым подробным образом описан процесс создания таблиц, ключей и индексов. Рассмотрена процедура обеспечения ссылочной целостности базы данных.
  - В *главе 3 "Создание форм и отчетов MS Access 2010"* вы получите представление о том, что такое форма и для чего она нужна, а также как создать различные формы Microsoft Access и заставить их работать совместно

с данными. Рассматривается процесс создания простой формы и сложной, содержащей подчиненную форму. При помощи форм и элементов управления разработчик может создавать окна различной конфигурации, максимально приспособленные для решения конкретных задач. Вы научитесь использовать отчеты для отображения информации из базы данных в виде печатного документа. Отчет является конечным продуктом большинства приложений баз данных и представляет собой специальный тип непрерывных форм. В MS Access можно создавать отчеты в одну колонку, ленточные, многоколоночные, групповые отчеты и почтовые наклейки.

- В *главе 4 "Дополнительные возможности Microsoft Access"* приведены сведения, делающие работу с СУБД Microsoft Access еще более эффективной и комфортной.

♦ **Часть II. Дальнейшее развитие вашего приложения.**

В которой вы научитесь делать то, что нужно, и так, как можно.

- В *главе 5 "Основные сведения о Visual Basic for Applications"* рассматриваются основы этого популярного языка программирования. Visual Basic for Applications (VBA) — это мощный инструмент разработки приложений. Как и другие средства, например, MS Visual C++, MS Visual C#, MS VB, MS Visual FoxPro, Borland Delphi, он предоставляет разработчику возможность создать полностью законченные программные продукты. VBA встроен во все приложения Microsoft Office, AutoCAD, CorelDRAW и множество других. Изучив VBA, вы сможете создавать свои программные комплексы не только в Microsoft Access. Необходимо лишь дополнительно освоить иерархию объектов выбранного приложения. В этой главе вы получите сведения о типах переменных, константах, стандартных функциях и выражениях. Научитесь работать с массивами, процедурами и функциями. Значительное место уделено описанию операторов VBA.
- В *главе 6 "Использование SQL Access"* рассматриваются основы языка структурированных запросов к базе данных. Вы познакомитесь с основными операторами, научитесь создавать программным путем запросы на выборку, модификацию, добавление и удаление данных. Здесь же рассматривается процесс создания формы, генерирующей запрос с параметрами. Она фактически представляет собой своеобразный мастер, дающий возможность совершенно неподготовленному пользователю решать сложные производственные задачи.
- В *главе 7 "Разработка интерфейса приложения"* рассмотрен, предложенный ранее автором, метод создания рабочего интерфейса — метод пересекающихся каскадов. Приведен программный способ создания головного меню программного комплекса. В меню объединяют последовательности и группы команд, одну из которых может выбрать пользователь для совершения очередного действия. В одном меню команды объединяют на осно-

ве одного из двух принципов: либо это различные действия над одним объектом, либо однотипные действия над различными объектами. Согласно существующим формальным и фактическим стандартам проектирования интерфейса, работа прикладной программы непременно должна начинаться с активизации головного меню, которое находится в верхней части окна приложения.

- В *главе 8 "Обеспечение информационной безопасности приложения"* детально описан процесс создания системы защиты информации современного предприятия. Предложена методика реализации как мандатного, так и дискреционного принципов управления доступом.
- В *главе 9 "Создание основных форм приложения"* вы узнаете много интересного об обеспечении совместной работы нескольких пользователей с базой данных, о блокировках и достоинствах применения объекта "Набор вкладок". Вы непременно оцените удобный интерфейс этого объекта, предназначенный для просмотра сведений о записях, попавших в запрос.
- В *главе 10 "Взаимодействие с другими приложениями"* рассматривается технология передачи данных в Microsoft Office 2010. В большинстве случаев непрактично преобразовывать таблицу MS Excel или документ MS Word в таблицы базы данных MS Access, а сохранить потрясающую эффективность работы этих приложений с электронной таблицей или документом просто необходимо. Достаточные сведения по созданию внешних отчетов вы найдете в этой главе. Также подробным образом описан процесс создания контекстуально-зависимой справочной системы приложения в HTML Help Workshop. Находясь в любом месте программного комплекса, пользователь может получить помощь, нажав клавишу <F1>.
- В *главе 11 "Создание пользовательской ленты"* подробно рассказано о назначении нововведения от Microsoft, свойствах объекта, методах работы с ним и описаны некоторые авторские находки.

♦ **Часть III. Перевод приложения на платформу "клиент-сервер".**

В которой вы научитесь делать то, что нужно, и так, как нужно.

- В *главе 12 "Преобразование базы данных MS Access 2010 в базу MS SQL Server 2008"* рассмотрены все этапы и варианты конвертации accdb-файла в базу данных MS SQL Server. Подробно описана работа по созданию приложения в варианте "проект MS Access — база данных MS SQL Server".
- В *главе 13 "Основные сведения об MS SQL Server 2008"* рассмотрена работа с объектами MS SQL Server с применением основного рабочего инструмента SQL Server Management Studio. Описаны основы языка программирования Transact-SQL.
- В *главе 14 "Внесение изменений в проект Microsoft Access"* подробно описаны приемы, позволяющие решить те проблемы преобразования, которые

оказались не под силу мастеру MS Access 2010. Пример приложения, рассматриваемый в книге, подобран таким образом, чтобы выявить абсолютно все слабые стороны мастера преобразования MS Access 2010. При работе с реальным приложением доработок проекта у вас будет гораздо меньше.

- **Приложение. Описание компакт-диска.**

  Подробно описан процесс восстановления учебной базы Real Estate из dat-файла SQL Server 2008 на компьютере читателя.

Как создавалась эта книга

Несколько лет назад на кафедру университета, где работает автор, обратились представители администрации города с предложением разработать программный комплекс по учету недвижимости. Решением краевых властей бюро технической инвентаризации тогда перешло под эгиду мэрии. Предложение заслуживало внимания, если бы не тот факт, что программный комплекс должен быть завершен через три месяца, причем все объекты (здания, домовладения, квартиры, проживающие и т. д.) объемом 6 000 000 записей к тому времени уже должны быть в базе. Для занесения информации выделялся специальный штат работников в 20 человек с работой в две смены и 10 компьютеров.

К тому времени у автора уже был солидный опыт разработки, но имеющиеся в его арсенале Visual FoxPro 6 и Visual Basic 6 в связке с SQL Server 2000 не позволяли решить эту задачу к концу лета, как настаивал директор департамента муниципальной собственности.

Пришлось пойти на беспрецедентный, как тогда казалось, шаг — начать работу с применением самого быстрого средства разработки приложений — MS Access. Через неделю все наборщики сели за компьютеры. А еще через месяц база заработала в архитектуре "клиент-сервер".

Параллельно была создана учебная база Real Estate и собирался материал для книги "Microsoft® Access 2007. Разработка приложений на реальном примере"[3], которая после выхода в свет нашла свое место на полках учебных библиотек многих вузов России. В течение трех лет материал был "обкатан" в учебном процессе при обучении студентов как общетехнических специальностей (*часть I* книги), так и профильных: "Информационные системы и технологии", "Прикладная математика и информатика", "Комплексное обеспечение информационной безопасности автоматизированных систем" и "Системы автоматизированного проектирования", а автор получил большое количество писем с вопросами и предложениями по улучшению книги. Многие из них и были учтены при написании этого издания.

---

[3] Microsoft® Access 2007. Разработка приложений на реальном примере. — СПб.: БХВ-Петербург, 2007. — 672 с.: ил. + CD-ROM (Профессиональное программирование).

Специальные элементы книги

Книга содержит множество особых вставок, выделенных специальным образом. В них содержится дополнительная информация, облегчающая чтение.

Примечание

Дополнительная заметка, объяснение к тексту.

Совет

В советах рассказывается о некоторых хитростях, которые следует знать разработчику, чтобы наиболее эффективно использовать возможности рассматриваемых программных продуктов.

Предупреждение

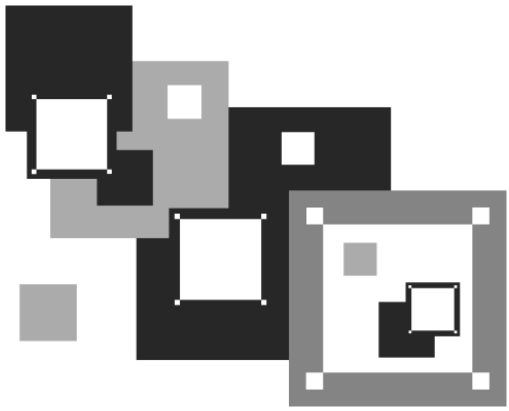
Предупреждения помогут вам избежать проблем из-за конкретных действий. В них сказано, чего следует опасаться, а также что нужно делать, чтобы избежать ошибок.

Об авторе

Геннадий Александрович Гурвиц — кандидат технических наук, доцент Дальневосточного государственного университета путей сообщения. Область интересов: технологии программирования, операционные системы, базы данных. Написал более 20 программных комплексов, обеспечивающих деятельность некоторых департаментов и отделов администрации своего города, учреждения юстиции, краевого бюро технической инвентаризации, налоговой полиции, а также небольших и средних предприятий. Является автором книг "Разработка реального приложения с использованием Microsoft Access 2000", "Разработка реального приложения в среде клиент-сервер", "Разработка реального приложения с использованием Microsoft Visual FoxPro 9" и "Microsoft Access 2007. Разработка приложений на реальном примере".

Автор благодарит вас за выбор этой книги и надеется, что она сможет стать вашим надежным гидом в увлекательной работе над первым реальным приложением MS Access 2010 — MS SQL Server 2008.

Буду очень рад выслушать ваши замечания по методике изложения материала и этапам создания реального приложения в клиент-серверной архитектуре. Не сомневаюсь, особо искушенные читатели скажут: "В книге есть, что “попилить”", и в вашем арсенале имеются более тонкие наработки. Но не забывайте, что эта книга, как и описываемая в ней база Real Estate, предназначена, по большому счету, для уровня разработчика ниже вашего, хотя и профессионал может найти в ней немало новых для себя моментов!



ЧАСТЬ I

ВАШЕ ПЕРВОЕ ПРИЛОЖЕНИЕ В MS ACCESS 2010

В которой вы научитесь делать то, что можно, и так, как нужно

ГЛАВА 1

Введение в базы данных

Все деловые приложения хранят значительные объемы данных, являющиеся ядром информационной системы. Такая система в первую очередь призвана облегчить труд человека, а для этого она должна как можно лучше соответствовать сложной модели реального мира. В главе рассматриваются основы построения баз данных и информационных систем. Даны важные понятия баз данных и систем управления базами данных. Рассмотрена наиболее распространенная реляционная модель представления данных. Описаны структуры файл-серверных и клиент-серверных информационных систем.

1.1. Понятие информационной системы

В широком понимании под определение информационной системы попадает любая система обработки информации. В составе информационной системы всегда можно выделить две составляющих:

- компьютерную инфраструктуру предприятия или организации, которая представляет собой совокупность сетевой, телекоммуникационной и организационной инфраструктур;
- функциональные подсистемы, обеспечивающие решение задач предприятия или организации и достижение их целей.

Первая составляющая носит название *корпоративной сети*. Требования к корпоративной сети едины и хорошо стандартизованы. Методы ее построения хорошо известны, многократно проверены на практике и выходят за рамки этой книги.

Вторая составляющая информационной системы целиком относится к прикладной области и полностью зависит от задач, решаемых предприятием или организацией. Требования к функциональным подсистемам, как правило, противоречивы, поскольку выдвигаются специалистами разных прикладных областей. Несомненно, это более важная составляющая информационной системы и именно для нее и строится компьютерная инфраструктура.

Взаимосвязи между составляющими информационной системы в действительности очень сложны. На первый взгляд, как это ни парадоксально, они кажутся независимыми. В самом деле, построение компьютерной сети и протоколы обмена данными абсолютно не зависят от методов и программ, применяемых для автоматизации работы предприятия. При более детальном рассмотрении легко выясняется, что функциональные подсистемы не могут существовать независимо от корпоративной сети. Невозможно эксплуатировать распределенную информационную систему без соответствующей ей сетевой инфраструктуры.

Предупреждение

Нельзя строить корпоративную сеть, не принимая во внимание прикладную функциональность информационной системы.

Типовая информационная система в самых общих чертах включает в себя следующие составляющие:

- серверы, рабочие станции с их периферийными устройствами;
- коммуникационное оборудование (маршрутизаторы, шлюзы, коммутаторы, мосты и т. д.);
- системное программное обеспечение (операционные системы, серверы баз данных и т. д.);
- базы данных;
- системы управления базами данных;
- прикладное программное обеспечение, разработанное прикладными программистами (приложение);
- обслуживающий персонал.

В компьютерной литературе иногда встречается понятие банка данных. *Банк данных* — это та же информационная система, в которой представлены функции централизованного накопления информации, хранящейся в нескольких базах данных.

Кратко рассмотрим компоненты информационной системы, относящиеся к ее второй составляющей — функциональной подсистеме.

База данных — совместно используемый набор логически связанных данных.

Система управления базами данных (СУБД) — программное обеспечение, с помощью которого пользователи могут создавать, модифицировать базу данных и осуществлять к ней контролируемый доступ.

Прикладное программное обеспечение (приложение) — комплекс программ, обеспечивающих автоматизацию обработки информации.

Приложения могут создаваться как в среде системы управления базами данных, так и вне ее. Работать с базой данных, входящей в состав простой информацион-

ной системы, очень часто можно и без написания дополнительного кода, что и продемонстрировано в *части I*.

Администратор сети — лицо, отвечающее за правильное функционирование аппаратно-программных средств сети, ее реконфигурацию и восстановление системного программного обеспечения после сбоев и отказов оборудования. В его обязанности входят также мероприятия по разграничению доступа к ресурсам сети и профилактические мероприятия.

Администратор базы данных — лицо, отвечающее за проектирование базы данных, защиту ее от несанкционированного доступа. В его обязанности также входит контроль за сохранностью, достоверностью хранимой в базе данных информации и обеспечение ее непротиворечивости и сохранности.

Разработчик программного комплекса (прикладной программист) — лицо, отвечающее за разработку приложения, созданного как в среде системы управления базами данных (например, MS Access), так и внешнего (например, MS Visual Studio).

Обслуживающий персонал — группа лиц, для улучшения работы которых и создана информационная система (работники предприятия или учреждения).

1.2. Архитектура "файл-сервер"

Исторически первыми появились информационные системы с использованием файл-сервера. Файл-сервер только извлекает данные из файла (файлов) базы данных и передает их клиенту для дальнейшей обработки (рис. 1.1).

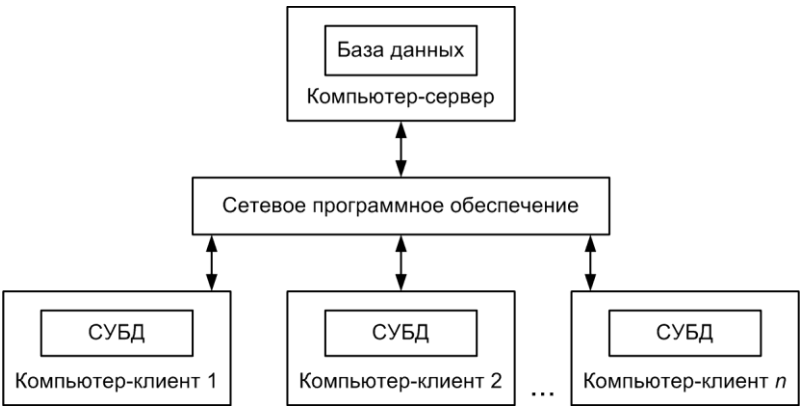


Рис. 1.1. Структура информационной системы с файл-сервером

В процессе работы из базы данных клиенту передаются большие объемы информации. Значительный сетевой трафик иногда особенно сильно сказывается при одновременной работе даже уже нескольких клиентов. В файл-серверной архи-

тектуре всегда передаются избыточные данные. Неважно, сколько записей из базы данных нужны клиенту — файлы базы данных передаются в самом общем случае целиком. Что касается MS Access, то нагрузку на сеть добавляют еще и объекты приложения, такие как формы, отчеты и т. д. Они вместе с данными хранятся в одном файле на компьютере-сервере.

Примечание

В MS Access 2010 у разработчика имеется возможность разделить данные и приложение, работающее с этими данными. В этом случае приложение тиражируется на компьютерах-клиентах, а база данных остается на компьютере-сервере.

Применение архитектуры "файл-сервер" привлекает своей простотой, удобством использования и доступностью. Она представляет интерес для малых рабочих групп, а нередко до сих пор используется и в информационных системах масштаба небольшого предприятия.

1.3. Архитектура "клиент-сервер"

Информационные системы с клиент-серверной архитектурой позволяют избежать проблем файл-серверных приложений. При такой архитектуре сервер базы данных, расположенный на компьютере-сервере, обеспечивает выполнение основного объема обработки данных. Клиентское приложение формирует запросы к серверу базы данных, как правило, в виде инструкций языка SQL. Сервер извлекает из базы запрошенные данные и передает на компьютер клиента. Главное достоинство такого подхода — значительно меньший объем передаваемых данных.

Большинство конфигураций информационных систем типа "клиент-сервер" использует двухуровневую модель, в которой клиент обращается к серверу (рис. 1.2).

Обеспечение безопасности данных — очень важная функция для успешной работы информационной системы. Если у базы данных слабая система безопасности, любой достаточно подготовленный пользователь может нанести серьезный ущерб работе предприятия. Следует отметить, что защита данных в файл-серверной информационной системе изначально не может быть обеспечена на должном уровне.

Безопасность же современных серверов баз данных, организованная в нескольких направлениях: с помощью самой операционной системы; с использованием схем, имен входов, ролей, шифрования базы данных и т. д.; путем ограничения доступа пользователей через представления, заслуживает похвалы.

В настоящее время архитектура "клиент-сервер" широко признана и находит применение для организации работы приложений как для рабочих групп, так и для информационных систем масштаба предприятия.

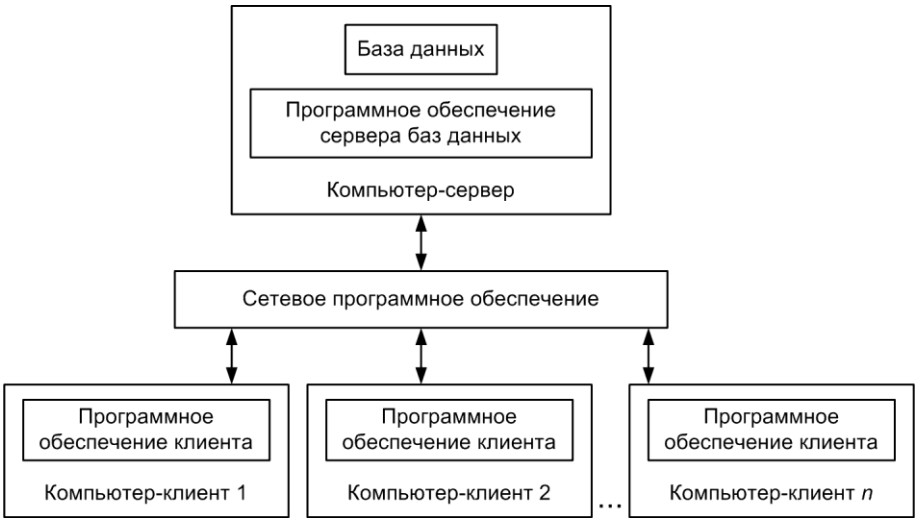


Рис. 1.2. Структура информационной системы с сервером базы данных

1.4. Реляционные базы данных

На сегодняшний день доминирующими среди баз данных являются реляционные. Такое широкое распространение связано с применением в них реляционной модели данных, обеспечивающей простоту, гибкость структуры, удобство реализации, а самое главное — наличие теоретического описания. В основе реляционной модели лежит один из разделов математики — реляционная алгебра. Реляционная модель данных пришла на смену сетевой модели, век которой быстро закончился именно из-за отсутствия теоретического описания. Многие разработчики старшего поколения до сих пор уверены в своих заблуждениях, что в основу сетевой модели данных положена теория графов!

Правда, наличие теоретического описания несет с собой и некоторые трудности. Терминология, используемая в реляционной модели, может привести к путанице, поскольку существуют два набора терминов. Один родом из математики, а второй — из теории реляционных баз данных. Но это еще полбеды. Есть также третий вариант терминологии, основанный на том, что физически в реляционной базе данных каждое отношение может храниться в отдельном файле. Тогда отношение — это файл, кортеж — запись, а атрибут — поле.

Не будем вдаваться в подробности отличия математического отношения от отношения (связи) в базе данных, просто посмотрите на табл. 1.1, и в дальнейшем будем применять только термины реляционной модели данных.

Таблица — регулярная структура, состоящая из конечного набора однотипных записей. Постараемся избегать термина "отношение".

Таблица 1.1. *Элементы реляционной модели данных*

Элемент реляционной модели	Форма представления
Отношение	Таблица
Схема отношения	Заголовок таблицы
Кортеж	Строка таблицы
Сущность	Свойства объекта
Атрибут	Заголовок столбца таблицы
Значение атрибута	Значение поля
Первичный ключ	Одно или несколько полей таблицы
Тип данных	Тип значений столбца таблицы

1.4.1. Таблицы базы данных

Каждая таблица реляционной базы данных состоит из строк и столбцов и предназначена для хранения данных об объектах информационной системы (рис. 1.3.). В дальнейшем будем придерживаться терминов "поле" и "запись".

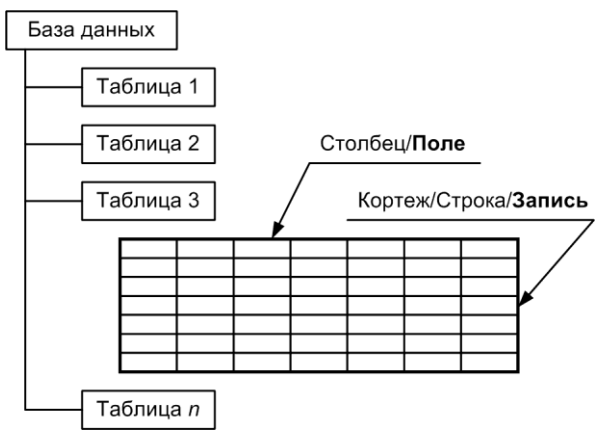


Рис. 1.3. Схема таблицы реляционной базы данных

Поле может содержать данные только одного из допустимых типов (тип данных) для конкретно используемой базы данных, например MS Access, MS SQL Server, Oracle и т. д.

Значения некоторых терминов, применяемых в реляционной модели данных и приведенных выше, легко понять, изучив содержимое конкретной таблицы (рис. 1.4).

Рис. 1.4. Таблица **Здания**

Сущность — объект любого происхождения, данные о котором хранятся в базе данных.

Атрибут — свойство, характеризующее сущность. В таблице представляет собой заголовок столбца.

Схема отношения — список имен атрибутов.

1.4.2. Ключи и индексы

Ключ представляет собой поле или группу полей, причем таких, которые однозначно определяют любую запись в таблице реляционной базы данных. Ключ предназначен для:

- идентификации записей в таблице;
- установления связи между таблицами базы данных;
- создания ограничений ссылочной целостности.

В реляционных базах данных ключ реализуется с помощью индекса.

Индекс — указатель на данные, размещенные в реляционной таблице. Он предоставляет информацию о точном физическом их расположении.

Одним из основных требований, предъявляемых к СУБД, является возможность быстрого поиска требуемых записей. В реляционных базах данных для реализации этого требования как раз и служат индексы. Индекс очень похож на алфавитный указатель в книге.

Например, у вас в руках книга по Microsoft Office Access 2010 и вы хотите узнать о том, что написано в ней об индексах. Загляните в конец книги и найдите в предметном указателе слово "индекс". Так как указатель отсортирован по алфавиту, вы без труда найдете нужное слово и ссылки на страницы, где оно встреча-

ется в книге. Индекс работает с таблицей по такому же принципу. Он содержит отсортированные значения указанного поля таблицы и ссылки на номера записей таблицы, где эти значения находятся. При поиске записи система управления базами данных сначала просматривает индекс, что занимает совсем немного времени, т. к. для этого используется специальный алгоритм, находит ссылку на номер записи и по ней — нужную строчку в таблице. Таким образом, отпадает необходимость последовательного просмотра всех записей в таблице.

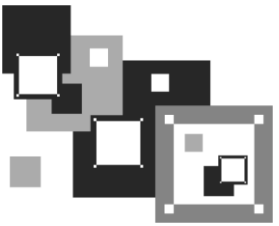
При создании индекса в нем располагается информация о местонахождении записей, относящихся к индексированному полю. При добавлении новых записей в таблицу или удалении имеющихся индекс модифицируется в реальном времени.

Рассмотрим работу индекса на конкретном примере. На рис. 1.5 показан фрагмент таблицы `Телефоны` и индекса, построенного по полю `Номер`. Требуется найти владельца телефонного номера 4983217. При поиске СУБД в первую очередь находит номер телефона, расположенный ровно в центре столбца (номера в индексе отсортированы по возрастанию). Это номер 4987312, который больше требуемого. Вывод: рассматривать нижнюю половину индексного файла для дальнейшего поиска не имеет смысла. Для десятимиллионного телефонизированного города это пять миллионов записей. Второй шаг — обращение точно в центр верхней половины. Там расположен абонент 2791519. Его номер меньше искомого. Долой еще два с половиной миллиона записей на этот раз из верхней четверти столбца. Третий шаг исключит из рассмотрения восьмую часть абонентов. Четвертый — шестнадцатую и т. д. Очень скоро работа алгоритма будет закончена. Нужный нам владелец телефонного номера будет найден. Его фамилия — Степанов.

Номер	Запись
2317415	9
2338218	3
2791519	8
3427511	2
4983217	6
4987312	10
5711107	5
5929214	11
5933970	7
6032773	1
6329145	4

Запись	Номер	Фамилия	...	...	...
1	6032773	Аксенов			
2	3427511	Попова			
3	2338218	Рыкова			
4	6329145	Иванов			
5	5711107	Андреев			
6	4983217	Степанов			
7	5933970	Акатьева			
8	2791519	Сергеев			
9	2317415	Смирнов			
10	4987312	Яковлев			
11	5929214	Мухина			

Рис. 1.5. Поиск владельца телефона по его номеру при помощи индекса



ГЛАВА 2

Разработка базы данных

Как и множество других задач, построение базы данных начинается с этапа проектирования. Вполне очевидно, что никто не возьмется за строительство здания без чертежей. Точно так же, грамотно спланированная база данных — основной шаг в успешной реализации проекта. Проектирование реляционной базы данных включает следующие этапы:

- моделирование приложения;
- определение данных, с которыми будет работать приложение;
- распределение данных по таблицам;
- организация связей между таблицами;
- создание необходимых индексов;
- создание механизмов проверки данных;
- создание необходимых запросов к базе данных.

2.1. Постановка задачи

Предстоит вам работа в информационно-аналитическом отделе небольшого предприятия или вы еще учитесь в университете и только готовитесь к самостоятельной деятельности — не имеет значения.

Пусть ваше первое задание — разработка прикладного программного обеспечения деятельности только что созданного на этом предприятии отдела по учету недвижимости, находящейся на балансе предприятия. В связи с реорганизацией городского хозяйства объектов в ведении отдела теперь около полусотни. Квартир порядка трех тысяч, в них проживает около десяти тысяч человек. Учет недвижимости, а также отслеживание квартплаты отныне в ведении этого отдела, но это уже второй этап работы.

В первую очередь на вас возложена задача компьютерного учета недвижимого имущества. Объем работы сравнительно небольшой. Не радуйтесь! Ваш началь-

ник требует, чтобы эксплуатация программного комплекса, заказанного им сегодня, началась еще вчера. Вы провели в отделе по учету недвижимости значительное время, но все, что вам удалось выяснить из разговора с персоналом, — это набор данных, которые будут храниться в электронном виде, их тип и максимальное количество в базе (табл. 2.1).

Они сведены вами в таблицу. Надеюсь, что вы предупредили работающих о том, что если какой-либо параметр отсутствует в базе данных, то извлечь его и выполнить какие-либо расчеты с его участием будет в дальнейшем невозможно.

Учтите, что приступать к созданию базы данных еще рано. На этом этапе вы должны определить задачи, которые будет решать разрабатываемое приложение. Другими словами, надо составить функциональную спецификацию. На первый взгляд кажется совершенно очевидным, что должно делать приложение. Однако в подавляющем большинстве случаев выясняется, что заказчик сам плохо представляет то, что предстоит сделать в этом направлении.

Совет

Задавайте наводящие вопросы до тех пор, пока вы окончательно не поймете, какие цели преследуют будущие пользователи этого программного комплекса. Не собираются ли они заменить разрабатываемую систему. Есть ли у них отработанные виды отчетов и т. д.

Разумно, если вы выбрали в качестве инструмента Microsoft Access 2010 — рекордсмена среди существующих СУБД по времени разработки приложения и запланировали перевод базы данных на платформу SQL Server 2008, т. к. при отслеживании платежей по квартплате число записей в таблице лицевых счетов достигнет 200—500 тыс., а это уже несколько превышает возможности MS Office Access 2010.

Таблица 2.1. *Набор данных "Недвижимость"*

№	Поле	Тип	Размер	Описание
1	Address	Текстовый	50	Адрес здания
2	District	Текстовый	15	Район города, где оно расположено
3	Land	Числовой	10	Площадь земельного участка
4	Year	Числовой	4	Год постройки здания
5	Material	Текстовый	15	Материал стен здания
6	Comment	Поле MEMO	Авто	Примечания
7	Wear	Числовой	2	Износ в процентах
8	Cost	Денежный	15	Стоимость здания в рублях
9	Line	Числовой	5	Расстояние от центра города
10	Square	Числовой	10	Площадь нежилых помещений

Таблица 2.1 *(окончание)*

№	Поле	Тип	Размер	Описание
11	Picture	Поле OLE	Авто	Фото здания
12	Kind	Числовой	1	Вид собственности
13	Elevator	Логический	1	Наличие лифта
14	Flat	Числовой	4	Номер квартиры
15	Storey	Числовой	2	Номер этажа
16	Rooms	Числовой	1	Количество комнат
17	SquareFlat	Числовой	Авто	Общая площадь квартиры
18	Dwell	Числовой	Авто	Жилая площадь квартиры
19	Branch	Числовой	Авто	Вспомогательная площадь квартиры
20	Balcony	Числовой	Авто	Площадь балкона
21	Height	Числовой	Авто	Высота квартиры
22	Account	Числовой	5	Номер лицевого счета
23	FioHost	Текстовый	60	Ф. И. О. квартиросъемщика
24	Pasport	Поле MEMO	Авто	Данные его паспорта
25	Fio	Текстовый	60	Ф. И. О. проживающего в квартире
26	Born	Числовой	4	Год рождения проживающего
27	Status	Текстовый	20	Льготы и статус проживающего

Отныне вам предстоит иметь дело с информационной системой, предназначенной для сбора, хранения и обработки информации. Такая система непременно должна ориентироваться на конечного пользователя, не обладающего высокой квалификацией. Поэтому программный комплекс должен обладать удобным, простым и легко осваиваемым интерфейсом, который предоставляет работнику все необходимые функции и в то же время не дает совершать ему лишних действий. Нам предстоит решить две задачи:

- разработать базу данных для хранения информации;
- разработать графический интерфейс и само пользовательское приложение, работающее с этой базой данных.

База данных — совместно используемый набор логически связанных данных для удовлетворения информационных потребностей организации. Это корпоративный ресурс, не принадлежащий какому-либо единственному отделу. База данных — неотъемлемая часть любой информационной системы.

СУБД (система управления базами данных) — программное обеспечение, с помощью которого пользователи могут создавать, модифицировать базу данных и

осуществлять к ней контролируемый доступ. СУБД непременно взаимодействует с прикладными программами пользователя и самой базой данных.

Для работы СУБД и прикладных программ необходимо аппаратное обеспечение, которое также является частью информационной системы и может варьироваться в очень широких пределах: от единственной персоналки до сети из многих компьютеров. Приложение, которое мы создадим в начале нашей работы, будет функционировать на одном компьютере или, в лучшем случае, на уровне рабочей группы (до 10 машин в сети Windows технологии NT) в режиме файлового сервера. В этом варианте папка с базой данных и прикладными программами размещается на самом мощном компьютере одноранговой сети, и к ней организуется совместный доступ работников. Нагрузка на локальную вычислительную сеть — максимальная. Информационная безопасность — на самом низком уровне.

Дальнейшее развитие нашего приложения и перевод его в архитектуру "клиент-сервер", описанное в *части III*, кардинальным образом исправит положение дел. Корпоративная сеть будет избавлена от излишнего трафика, а применение сервера баз данных поднимет безопасность информационных ресурсов корпорации на должную высоту.

В настоящее время существует больше сотни различных СУБД, от персональных компьютеров до мэйнфреймов. Подавляющее большинство из них работает с базой данных, в основе которой лежит реляционная модель. На сегодняшний день известны три модели данных: иерархическая, сетевая и реляционная. Microsoft Office Access 2010 и Microsoft SQL Server 2008 — это реляционные СУБД.

Современная реляционная база данных хранит не только сами данные, но и их описания. Такой подход позволяет отделить данные от приложения. Следовательно, добавление поля в таблицу или таблицы в базу данных никак не повлияет на работу приложения.

Предупреждение

Удаление поля из таблицы, используемой приложением, повлияет на его работу. Приложение придется модифицировать.

2.2. Нормализация данных

Теперь займемся проектированием эффективной структуры данных. Теория реляционной базы данных была разработана в начале 70-х годов прошлого века Коддом (E. F. Codd) на основе математической теории отношений. В реляционной базе данных все данные хранятся в виде таблиц, при этом все операции над базой данных сводятся к манипуляциям таблицами. Основными понятиями в этой теории являются: таблица, запись, поле, индекс, первичный и внешний ключи, связи. Таблица состоит из строк и столбцов и имеет уникальное имя в базе

данных. База данных содержит множество таблиц, связь между которыми устанавливается с помощью совпадающих полей. В каждой из таблиц содержится информация о каких-либо объектах одного типа.

Приступая к созданию нового приложения, главное — самым тщательным образом спроектировать структуру его таблиц. Если не уделить структуре должного внимания, то в лучшем случае это может проявиться в неэффективной работе приложения, а в худшем — в невозможности реализации некоторых требований к системе в целом. И наоборот, при хорошей организации набора таблиц будут решены не только текущие проблемы, но и потенциальные, которые в данный момент вы не могли предвидеть. В общем, структура данных является определяющим фактором успеха или провала всего приложения.

Э. Ф. Кодд доказал, что, следуя при создании таблиц и связей между ними только немногим формализованным правилам, можно обеспечить простоту манипулирования данными. Его методика получила наименование *нормализации данных*. Теория реляционных баз данных основана на концепции использования ключевых полей для определения отношений между таблицами. Чем больше таблиц, тем больше отношений требуется определить, чтобы связать их между собой. Из теории Кодда отнюдь не следует, что каждая таблица должна быть напрямую связана с любой другой таблицей. Но, поскольку каждая таблица связана хотя бы с одной таблицей в базе данных, можно утверждать, что все таблицы в базе имеют прямые или косвенные отношения друг с другом.

Мы установили, какие поля будут включены в базу данных. Следующий этап состоит в разделении их на таблицы. Конечно же, можно было бы работать с приведенной выше единственной таблицей "Недвижимость", но даже не знающим правил нормализации ясно, что для каждого проживающего в квартире не имеет смысла повторять всю информацию о здании, квартире, ответственном квартиросъемщике и лицевом счете, а при переименовании улицы вносить исправления в тысячи записей, содержащих сведения о технических характеристиках квартиры.

Наличие повторяющейся информации приведет к неоправданному увеличению размера базы данных. В результате снизится скорость выполнения запросов. При многократном вводе повторяющихся данных возрастет вероятность ошибки.

Представьте себе ситуацию, связанную с вводом данных о проживающих на Восточном шоссе. Это пять тысяч человек. Вот несколько вариантов адреса: Шоссе Восточное, Восточное шоссе, ш. Восточное, ш-се Восточное. А сколько еще вариантов может появиться у пользователя, работающего с вашей программой. О грамматических ошибках и вариантах с номером дома, запятыми и точками в адресе позволю себе умолчать. Какую информационно-поисковую систему мы получим в результате? Скорее всего, искать можно — найти нельзя! Вашему вниманию — несколько советов по включению полей в таблицы.

- Включайте поля, относящиеся только к предметной области таблицы. Поле, представляющее факт из другой предметной области, должно принадлежать

другой таблице. Позже при установлении отношений между таблицами вы увидите, как можно объединить данные нескольких полей из разных таблиц. А на этом этапе убедитесь, что каждое поле таблицы описывает только одну предметную область. Если вы обнаружите, что в разных таблицах встречается однотипная информация, значит, какие-то таблицы содержат лишние поля.

- Не включайте производные или вычисляемые данные. В большинстве случаев вам нет необходимости хранить результаты вычислений в таблице. С помощью Microsoft Access вы всегда сможете выполнить данные вычисления в нужный момент. Не имеет смысла хранить итоговые поля в таблице.

- Включите всю необходимую информацию. Довольно легко упустить важные данные. Вернитесь к собранным на первой стадии проектирования материалам. Внимательно рассмотрите бланки и отчеты и убедитесь в том, что вся интересующая вас информация может быть извлечена или вычислена из таблиц. Подумайте о вопросах, которые вы будете формулировать к базе данных. Сможете ли вы получить на них ответ, используя данные из ваших таблиц? Включили ли вы поля, в которых будут храниться уникальные данные типа кода клиента? Какие таблицы содержат информацию, обязательную для создания отчета или формы?

- Разделите информацию на наименьшие логические единицы. Вам может показаться, что удобно иметь одно поле для хранения полного имени клиента или одно поле для названия и описания продукта. Это ошибка. Если вы объединяете более одной категории информации в одном поле, вам будет потом очень не просто выделить из него отдельные факты. Постарайтесь разбить информацию на наименьшие логические части.

Воспользуемся практическими рекомендациями теории нормализации для разработки на основании таблицы "Недвижимость" многотабличной базы данных Real Estate. На рис. 2.1 вы видите то, что у вас должно получиться после всех манипуляций, кратко изложенных выше и предусмотренных теорией нормализации. Практический же путь к этому результату см. на следующих полутора десятках страниц.

Примечание

В данной части все названия таблиц базы данных назначены в соответствии со смыслом размещенной в них информации. Файл приложения, размещенный на компакт-диске, для этой части называется Real Estate Часть I. В *части II* к названиям таблиц добавлена приставка "tbl". Это сделано для того, чтобы отличить их от запросов, получивших там же приставку "qwr". Файл приложения для нее получил название Real Estate Часть II. Его вы найдете также на компакт-диске.

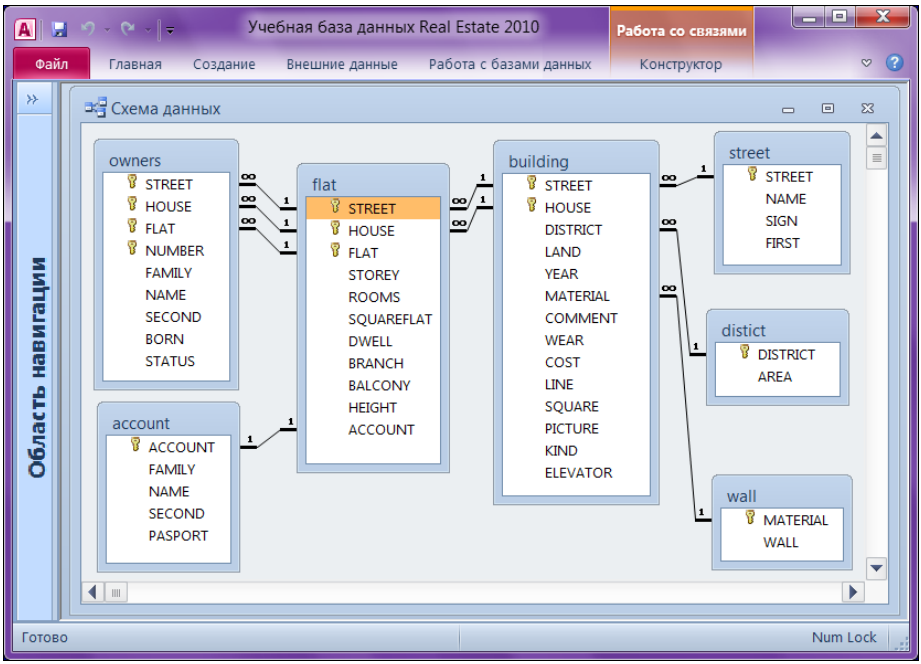


Рис. 2.1. Окончательный вид базы данных Real Estate

2.2.1. Первая нормальная форма

Таблица находится в первой нормальной форме, если значения всех ее полей атомарные и в ней отсутствуют повторяющиеся группы полей.

На "заре" существования реляционных баз данных на количество полей в записи накладывались определенные ограничения. Как следствие, разработчики объединяли несколько предполагаемых полей в одно, чтобы все нужные данные поместить в одну запись. Известно, что если поле содержит несколько значений, то существенно усложняется формирование отношений между полями, считывание данных и выполнение других операций, а необходимость выполнения поиска подстрок и синтаксического анализа полей в значительной степени замедляет работу приложения. К счастью, сейчас все ограничения на количество полей в записи сняты.

Приведем наши данные к первой нормальной форме. Выделим самостоятельные группы полей и поместим их в отдельные таблицы. На первый взгляд их четыре. Это информация о здании, адресе, квартире и собственниках. Добьемся атомарности всех полей. Поле `FioHost`, в которое записывается информация о фамилии, имени и отчестве ответственного квартиросъемщика, заменим тремя полями: `Family`, `Name`, `Second`. Также поступим и с проживающими в квартире. Поле `Address` разобьем на три: название, признак и порядок их следования в официальных документах. Получится следующая картина (табл. 2.2 и 2.3).

Таблица 2.2. *Информация об адресе (таблица* `street`*)*

№	Поле	Тип	Размер	Описание
1	STREET	Числовой	4	Номер улицы
2	NAME	Текстовый	30	Название улицы
3	SIGN	Текстовый	10	Признак адреса
4	FIRST	Логический	1	Порядок следования в документах

Таблица 2.3. *Пример данных об адресе*

Street	Name	Sign	First
173	Воронежская	Улица	Ложь
174	Воронежский	проезд	Истина
175	Воронежское	шоссе	Истина
176	Ворошилова	Улица	Ложь
564	Ленина	Площадь	Ложь

Примечание

Если значением поля `First` является `Ложь`, то при формировании адреса здания в официальных документах на первое место будет поставлен признак: *Улица Ворошилова*, а если `Истина` — название: *Воронежское шоссе* или *Воронежский проезд*.

Если значением поля `First` является `Ложь`, то значение признака адреса пишется с большой буквы: *Улица Ворошилова*, *Площадь Ленина*, а если `Истина` — с маленькой: *Воронежское шоссе*.

Обратите внимание также и на то, как легко будет сейчас решаться проблема переименования улицы. Допустим, что Воронежское шоссе, стоящее под номером 175 в таблице `Street`, переименовано, например, в улицу Муравьева-Амурского. Вносим исправления только в таблицу `Street`. Оставляем этот номер, меняем название, признак и значение поля `First` с `Истина` на `Ложь`. Проблема решена. Так как во всех остальных таблицах Воронежское шоссе (улица Муравьева-Амурского) фигурирует под номером 175, то никакие изменения не требуются.

Нам нужны и другие данные (табл. 2.4—2.6).

Таблица 2.4. *Информация о здании (таблица* `building`*)*

№	Поле	Тип	Размер	Описание
1	STREET	Числовой	4	Ссылка на номер улицы
2	HOUSE	Текстовый	10	Номер дома

Таблица 2.4 *(окончание)*

№	Поле	Тип	Размер	Описание
3	DISTRICT	Текстовый	15	Район города
4	LAND	Числовой	10	Площадь земельного участка
5	YEAR	Числовой	4	Год постройки здания
6	MATERIAL	Текстовый	15	Материал стен здания
7	COMMENT	Поле MEMO	Авто	Примечания
8	WEAR	Числовой	2	Износ в процентах
9	COST	Денежный	15	Стоимость здания в рублях
10	LINE	Числовой	5	Расстояние от центра города
11	SQUARE	Числовой	10	Площадь нежилых помещений
12	PICTURE	Поле OLE	Авто	Фото здания
13	KIND	Числовой	1	Вид собственности
14	ELEVATOR	Логический	1	Наличие лифта

Таблица 2.5. *Информация о квартире (таблица* `flat`*)*

№	Поле	Тип	Размер	Описание
1	STREET	Числовой	4	Ссылка на номер улицы
2	HOUSE	Текстовый	10	Номер дома
3	FLAT	Числовой	4	Номер квартиры
4	STOREY	Числовой	2	Номер этажа
5	ROOMS	Числовой	1	Количество комнат
6	SQUAREFLAT	Числовой	Авто	Общая площадь квартиры
7	DWELL	Числовой	Авто	Жилая площадь квартиры
8	BRANCH	Числовой	Авто	Вспомогательная площадь квартиры
9	BALCONY	Числовой	Авто	Площадь балкона
10	HEIGHT	Числовой	Авто	Высота квартиры
11	ACCOUNT	Числовой	5	Номер лицевого счета
12	FAMILY	Текстовый	20	Фамилия квартиросъемщика
13	NAME	Текстовый	20	Имя квартиросъемщика
14	SECOND	Текстовый	20	Отчество квартиросъемщика
15	PASPORT	Поле MEMO	Авто	Данные его паспорта

Таблица 2.6. *Информация о проживающих в квартире (таблица* `owners`*)*

№	Поле	Тип	Размер	Описание
1	STREET	Числовой	4	Ссылка на номер улицы
2	HOUSE	Текстовый	10	Номер дома
3	FLAT	Числовой	4	Номер квартиры
4	NUMBER	Числовой	2	Порядковый номер проживающего
5	FAMILY	Текстовый	20	Фамилия проживающего
6	NAME	Текстовый	20	Имя проживающего
7	SECOND	Текстовый	20	Отчество проживающего
8	BORN	Числовой	4	Год рождения проживающего
9	STATUS	Текстовый	20	Льготы и статус проживающего

Удовлетворение требованиям первой нормальной формы называется *структурной* или *синтаксической нормализацией*.

Итак, данные разделены на четыре родственные группы (табл. 2.2, 2.4—2.6): улицы, здания, квартиры и проживающие. Эти таблицы представлены в виде схемы (рис. 2.2). Значения всех полей таблиц — атомарные. Все таблицы находятся в первой нормальной форме. Однако останавливаться на этом не следует. С такими данными все еще возможно возникновение проблем. Прежде всего, в базе данных много повторений значений — не внутри одной записи, а в пределах одной таблицы. А там, где есть повторяющиеся значения, возможны противоречия. Посмотрите на поля `MATERIAL` и `DISTRICT` таблицы `building`. Та же картина, которая имела место чуть раньше с названиями улиц. Варианты названий материала стен: шлакобетон, шлакобетонные, шлб, шл. бет. Уберем название материала стен и названия районов в отдельные таблицы — справочники (`wall` и `district`), оставив в основной таблице `building` ссылки на эти справочники. База данных примет более правильный вид (рис. 2.3). В схеме появились еще две таблицы: `district` и `wall` (табл. 2.7 и 2.8).

Таблица 2.7. *Информация о районах города (таблица* `district`*)*

№	Поле	Тип	Размер	Описание
1	DISTRICT	Числовой	1	Номер района
2	AREA	Текстовый	15	Название района

Таблица 2.8. *Информация о материале стен здания (таблица* `wall`*)*

№	Поле	Тип	Размер	Описание
1	MATERIAL	Числовой	1	Номер материала
2	WALL	Текстовый	15	Название материала

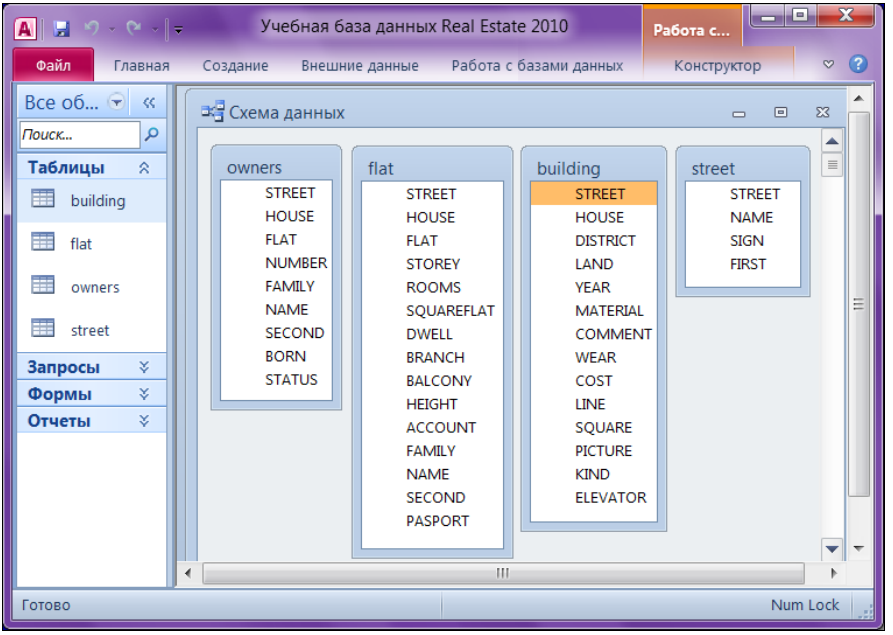


Рис. 2.2. Таблицы базы данных

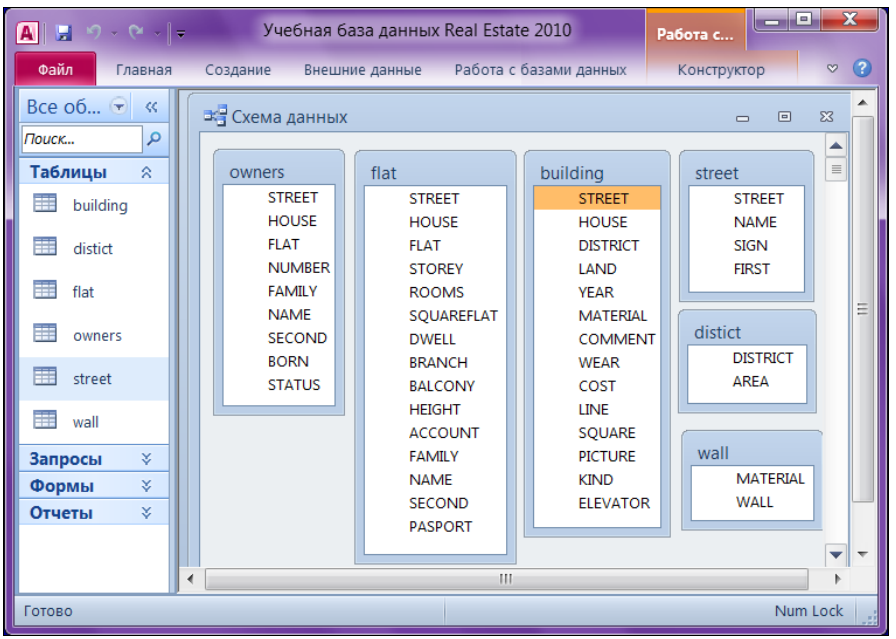


Рис. 2.3. Таблицы базы данных в первой нормальной форме

Структура таблицы `building` несколько изменилась. Вместо описаний района и материала стен появились ссылки на соответствующие таблицы `wall` и `district` (табл. 2.9).

Таблица 2.9. *Окончательная структура таблицы* `building`

№	Поле	Тип	Размер	Описание
1	STREET	Числовой	4	Ссылка на номер улицы
2	HOUSE	Текстовый	10	Номер дома
3	DISTRICT	Числовой	1	Ссылка на район города
4	LAND	Числовой	10	Площадь земельного участка
5	YEAR	Числовой	4	Год постройки здания
6	MATERIAL	Числовой	1	Ссылка на материал стен здания
7	COMMENT	Поле MEMO	Авто	Примечания
8	WEAR	Числовой	2	Износ в процентах
9	COST	Денежный	15	Стоимость здания в рублях
10	LINE	Числовой	5	Расстояние от центра города
11	SQUARE	Числовой	10	Площадь нежилых помещений
12	PICTURE	Поле OLE	Авто	Фото здания
13	KIND	Числовой	1	Вид собственности
14	ELEVATOR	Логический	1	Наличие лифта

2.2.2. Вторая нормальная форма

Таблица находится во *второй нормальной форме*, если она удовлетворяет условиям первой нормальной формы и любое неключевое поле однозначно идентифицируется полным набором ключевых полей.

Настало время поговорить о ключевых полях. Мощь реляционных баз данных, таких как Microsoft Access, опирается на их способность быстро найти и связать данные из разных таблиц при помощи запросов, форм и отчетов. Для этого каждая таблица должна содержать одно или несколько полей, однозначно определяющих каждую запись в таблице. Такие поля называют *первичным ключом таблицы*. Если для таблицы определен первичный ключ, то Microsoft Access предотвращает дублирование значений полей или ввод значений `Null` в эти поля. В Microsoft Access можно выделить три типа ключевых полей: простой ключ, составной ключ и счетчик. Если поле содержит уникальные значения, то его можно определить как *ключевое* или *простой ключ*. Примеры из нашей реальной жизни: идентификационный номер налогоплательщика, однозначно определяю-

щий каждого жителя нашей страны, номер свидетельства пенсионного фонда, кадастровый номер земельного участка, реестровый номер строения, номер автомобиля — все это уникальные номера в пределах страны. Поле STREET (номер улицы) в таблице street также можно определить как простой ключ. Этим же требованиям отвечают поля DISTRICT (номер района) и MATERIAL (номер материала) таблиц district и wall. Можно смело гарантировать их уникальность в пределах нашего программного комплекса. С таблицей building, содержащей информацию о зданиях, при определении первичного ключа нужно поступить таким образом. К нашим услугам *составной ключ*. Связка полей — номер улицы плюс номер дома — однозначно определит положение записи, относящейся к одному зданию в этой таблице. С однозначным определением квартиры в таблице flat (квартиры) дело состоит чуть сложнее. Составной первичный ключ выглядит так: номер улицы плюс номер дома плюс номер квартиры.

В очень редких случаях с определением первичного ключа для таблицы может сложиться тупиковая ситуация. Не отчаивайтесь, добавьте в таблицу поле и определите его тип как счетчик. Все остальное Access сделает самостоятельно. В это поле будет автоматически вноситься уникальное число даже при работе с вашей базой в сетевом варианте (с нескольких компьютеров одновременно). Если до сохранения созданной таблицы ключевые поля не были определены, Microsoft Access 2010 предложит создать ключевое поле автоматически. При нажатии кнопки **Да** будет создано ключевое поле счетчика.

2.2.3. Третья нормальная форма

Таблица находится в *третьей нормальной форме*, если она удовлетворяет условиям второй нормальной формы и ни одно из неключевых полей таблицы не идентифицируется с помощью другого неключевого поля.

Посмотрите внимательно на таблицу flat (квартиры) (см. рис. 2.3). Она содержит неключевое поле ACCOUNT (номер лицевого счета), которое однозначно определяет ответственного квартиросъемщика (поля: FAMILY, NAME, SECOND и PASPORT) в этой таблице. Уберем все эти поля в еще одну таблицу account и назначим в ней в качестве простого первичного ключа поле ACCOUNT (табл. 2.10).

Таблица 2.10. *Информация об ответственном квартиросъемщике (таблица* account*)*

№	Поле	Тип	Размер	Описание
1	ACCOUNT	Числовой	5	Номер лицевого счета
2	FAMILY	Текстовый	20	Фамилия квартиросъемщика
3	NAME	Текстовый	20	Имя квартиросъемщика
4	SECOND	Текстовый	20	Отчество квартиросъемщика
5	PASPORT	Поле MEMO	Авто	Данные его паспорта

2.2.4. Связи между таблицами

Осталось установить связи между таблицами, и база данных будет готова к работе. Microsoft Access поддерживает четыре типа связей: "один-к-одному", "один-ко-многим", "много-к-одному" и "много-ко-многим".

- **Связь "один-к-одному"** означает, что каждой записи одной таблицы соответствует только одна запись другой таблицы и наоборот. В качестве примера рассмотрим связь между таблицами `flat` и `account` (рис. 2.4). Одна квартира — один ответственный квартиросъемщик. Связь между ними поддерживается при помощи совпадающих полей `ACCOUNT`. Обратите внимание! У полей, используемых для связи, одинаковое наименование (`ACCOUNT`) и тип (числовой с 5 разрядами). Всегда придерживайтесь этого правила при определении полей для связи любого типа между таблицами. Хотя, если быть более точным, связь между таблицами устанавливается на основании значений совпадающих полей, а не их наименований.

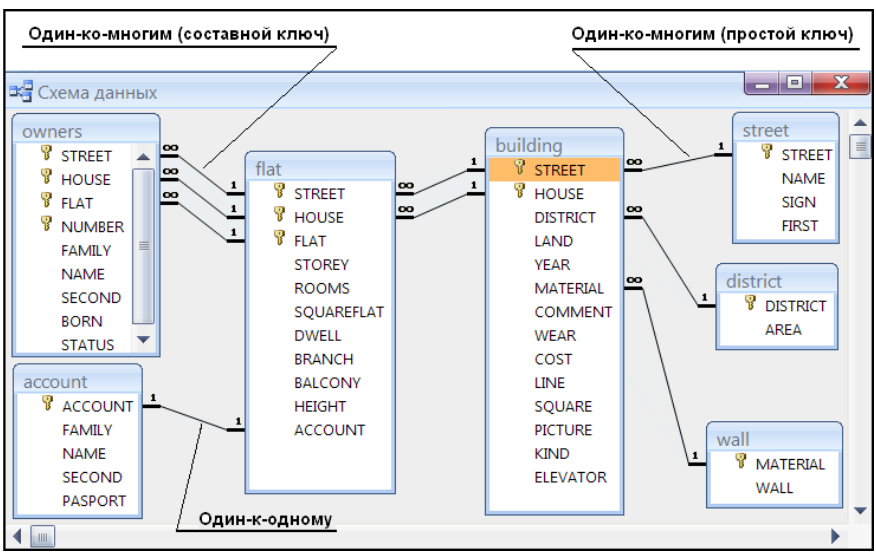


Рис. 2.4. Схема связей между таблицами

- **Связь "один-ко-многим".** В качестве иллюстрации данного типа связи обратимся к таблицам `street` и `building`. Одной улице в таблице улиц `street` соответствует несколько зданий из таблицы зданий `building`. Связь между ними осуществляется на основании значений совпадающих полей `STREET`. Используется простой первичный ключ таблицы `street`. В качестве других примеров могут быть рассмотрены таблицы `building` и `flat`, `flat` и `owners`. Одному зданию соответствуют несколько квартир, а одной квартире — несколько собственников. Для связи этих таблиц используются составные первичные ключи.

- **Связь "многие-к-одному"** аналогична ранее рассмотренному типу "один-ко-многим". Тип связи между объектами полностью зависит от вашей точки зрения. Например, если вы будете рассматривать связь между собственниками и квартирой, то получите "много-к-одному". Несколько собственников проживают в одной квартире.
- **Связь "многие-ко-многим"** возникает между двумя таблицами в тех случаях, когда одна запись из первой таблицы может быть связана более чем с одной записью из второй таблицы, а одна запись из второй таблицы может быть связана более чем с одной записью из первой таблицы. Таких связей следует избегать, т. к. реляционная модель не позволяет непосредственно работать с ними. Microsoft Access или любая другая реляционная СУБД в этом случае бесполезны. Всегда можно ввести в базу данных еще одну-две промежуточные таблицы и тем самым избежать возможных неприятностей при разработке интерфейса вашего приложения, используя понятные и безотказно работающие связи "один-ко-многим". Некоторые варианты заданий из этой книги могут привести к связи "многие-ко-многим" между таблицами базы данных. Обратившись к материалам *разд. 2.10*, вы увидите авторское видение решения этой проблемы.

2.2.5. Что за третьей нормальной формой?

Если вы довели уровень нормализации таблиц вашей базы данных до третьей нормальной формы и ваша задача — разработка системы масштаба предприятия, то смело можете переходить к разработке интерфейса. Однако если вы участвуете в разработке суперхранилища данных под Oracle или DB2, то разберитесь по специальной литературе с нормальной формой Бойса — Кодда, четвертой и пятой нормальными формами.

2.3. Создание новой базы данных

Все дальнейшие действия по разработке приложения описаны для операционной системы Windows 7. Запустите Microsoft Access 2010 на вашем компьютере. Для этого нажмите кнопку **Пуск** и выберите в появившемся меню пункт **Все программы**. Появится список программ, установленных на компьютере. Выберите пункт **Microsoft Office**. В открывшемся меню найдите Microsoft Office Access 2010. Он установлен там по умолчанию. Сделайте щелчок левой кнопкой мыши. На экране появится стартовое окно с названием **Microsoft Access**. В этом окне можно создать новую пустую базу данных, создать базу данных с помощью шаблона или открыть одну из последних баз данных, с которыми уже велась работа.

В левой части стартового окна вы увидите список ранее открывавшихся баз данных, а в центре — комплект профессионально разработанных шаблонов для создания базы данных. Этот набор помогает сократить время и обеспечивает быст-

рый доступ к средствам для начала работы. Большинство шаблонов находится на Web-узлах компании Microsoft. Опустим работу с шаблонами и перейдем к созданию собственной базы данных "с нуля".

Определитесь с именем файла базы данных и папкой, в которой он будет расположен. Пусть имя файла — Real Estate 2010. Строка для ввода имени файла и поиска папки, в которую он будет помещен, находится в правом нижнем углу окна. Найдите там же кнопку **Создать** и щелкните по ней. Будет создана новая база данных и открыта таблица **Таблица1** в режиме таблицы (рис. 2.5).

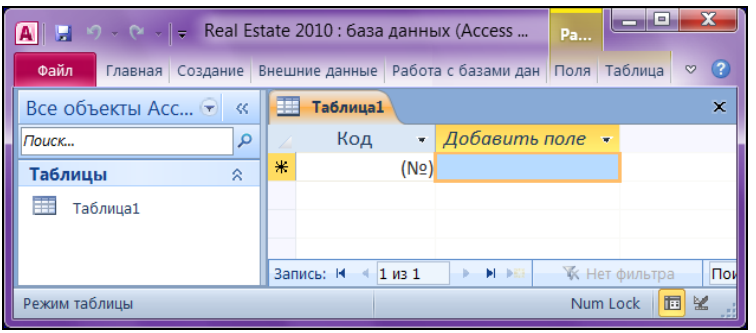


Рис. 2.5. Окно создаваемой базы данных Real Estate 2010

Закройте окно Microsoft Access. Создание таблиц мы рассмотрим в следующем разделе этой главы. База данных получила свое название и законное место на жестком диске. Отличительной особенностью Microsoft Access 2010 является то, что окно базы данных **Область навигации** (см. рис. 2.1) является отправной точкой, с которой начинается выполнение всех операций над объектами базы данных: таблицами, запросами, формами, отчетами, макросами и модулями. Не менее интересным фактом является и тот, что все эти элементы хранятся в одном-единственном файле. В данном случае — Real Estate 2010.accdb, чего не скажешь о других СУБД для персональных компьютеров.

В MS Access 2010 применяется несколько расширений файлов. Основные из них:

- accdb — расширение файла базы данных формата MS Access 2007/2010. Заменяет файлы с расширением mdb формата предыдущих версий;
- accde — расширение файлов MS Access 2007/2010, которые работают в режиме "исполнения". В accde-файлах удален весь исходный код. Работающий с accde-файлом может только выполнять код VBA, но не может изменять его. Accde-файлы пришли на смену файлам с расширением mde;
- accdt — расширение файлов шаблонов баз данных MS Access 2007/2010;
- adp — расширение файла проекта MS Access 2007/2010, который предназначен для работы с базой данных MS SQL Server. Формат файла не изменился со времен MS Access 2002/2003. В очередной версии пакета MS Office 15, вы-

пуск которой намечен ориентировочно на 2014 год, разработчики корпорации Microsoft обещают кардинально улучшить работу связки "MS Access — MS SQL Server";

- ade — расширение файлов проекта MS Access 2002/2003/2007/2010, которые работают в режиме "исполнения". В ade-файлах удален весь исходный код. Работающий с ade-файлом может только выполнять код VBA, но не может изменять его.

Познакомимся с интерфейсом Microsoft Access 2010. Этот интерфейс создан в результате многочисленных исследований и тестов на эффективность и практичность, проведенных компанией Microsoft в 2006—2009 годах. Целью его разработки было упрощение доступа к необходимым функциям MS Access. Революционный интерфейс пользователя в Microsoft Access 2007 содержал ряд новых элементов для работы с приложением. Эти новые элементы были введены для того, чтобы пользователь мог быстрее овладеть навыками работы с Microsoft Access и находить нужные команды. Новый дизайн упростил доступ к функциональным возможностям, которые раньше было трудно заметить из-за сложной структуры меню и панелей инструментов. В MS Access 2010 эти изменения получили дальнейшее развитие.

Самый важный элемент MS Access 2010 называется *лентой*. Это широкая полоса, находящаяся в верхней части окна программы, на которой расположены группы команд. На ленте имеются все команды, которые необходимы как для создания баз данных и приложений, так и для работы с ними (рис. 2.6).

Примечание

Вкладки ленты объединяют логически связанные команды. В Microsoft Access 2010 основные вкладки ленты — **Главная**, **Создание**, **Внешние данные** и **Работа с базами данных**.

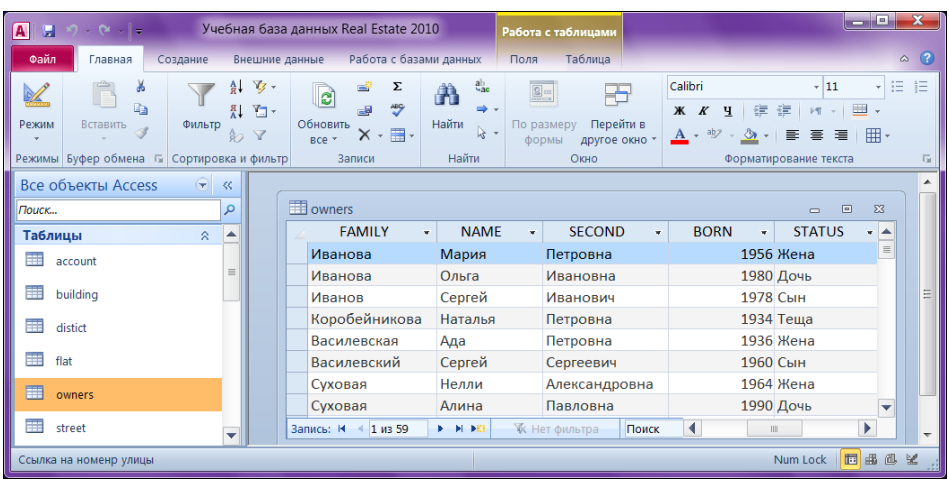


Рис. 2.6. Лента — основной элемент интерфейса Microsoft Access 2010

Каждая вкладка содержит группу связанных команд, которые могут открывать другие новые элементы интерфейса. Справа от основных вкладок ленты могут отображаться дополнительные. В данном случае (см. рис. 2.6) — это вкладки **Поля** и **Таблица**, т. к. в окне Microsoft Access открыта таблица `owners` в режиме таблицы. Иногда требуется выделить на экране дополнительное пространство для работы. В этих случаях можно свернуть ленту и оставить только строку с вкладками команд.

Совет

Чтобы свернуть ленту, дважды щелкните активную вкладку. Чтобы развернуть ее, снова дважды щелкните активную вкладку. Чтобы "прокрутить" ленту из начала в конец и обратно воспользуйтесь колесиком мыши. Свернуть ленту можно также с помощью пиктограммы  , которая расположена в правом верхнем углу окна MS Access 2010.

На ленте иногда могут отображаться не все значки или текст. Размер ленты оптимален для разрешения экрана 1280×1024 точек, когда приложение MS Access 2010 развернуто на весь экран. Например, полная вкладка **Главная** в приложении MS Access 2010 отображает текст и значки полностью. При уменьшении ленты группы на открытой вкладке начинают сжиматься горизонтально и превращаются в один значок. При сжатии программного окна остаются открытыми наиболее часто используемые команды или функции. Когда пользовательский интерфейс не отображается полностью, значки могут располагаться не в том порядке, как прежде. Значки, отображавшиеся ранее в одной строке, могут теперь отображаться в нескольких.

Еще один элемент интерфейса — панель быстрого доступа (рис. 2.7). Она размещена в левом верхнем углу окна Microsoft Access 2010 и представляет собой небольшую область, смежную с лентой, которая обеспечивает доступ к командам одним нажатием кнопки. В набор по умолчанию входят команды, которые обычно требуются чаще всего:  **Сохранить**,  **Отменить** и  **Вернуть**. Однако **Панель быстрого доступа** можно настраивать, добавляя другие часто используемые команды. В стандартном виде эта панель находится над вкладками команд ленты, но по желанию пользователя может быть перенесена и под ленту, что и показано на рис. 2.7.

Для настройки панели быстрого доступа сделайте щелчок левой кнопкой мыши по стрелке раскрытия списка в правой части панели. Откроется окно **Настройка панели быстрого доступа**. Выберите в нем пункт **Другие команды**. В диалоговом окне **Параметры Access** выделите команду, которую требуется добавить, и нажмите кнопку **Добавить**. Для удаления команды выделите ее в списке, расположенном справа, и нажмите кнопку **Удалить**. Можно также дважды щелкнуть по команде в списке. По завершении нажмите кнопку **OK**.

Как уже отмечалось ранее, в интерфейсе Microsoft Access 2010 есть еще один элемент — **Область навигации**. При открытии имеющейся или создании новой

базы данных имена объектов базы данных появляются в **Области навигации**. К объектам базы данных относятся таблицы, формы, отчеты, страницы, макросы и модули.

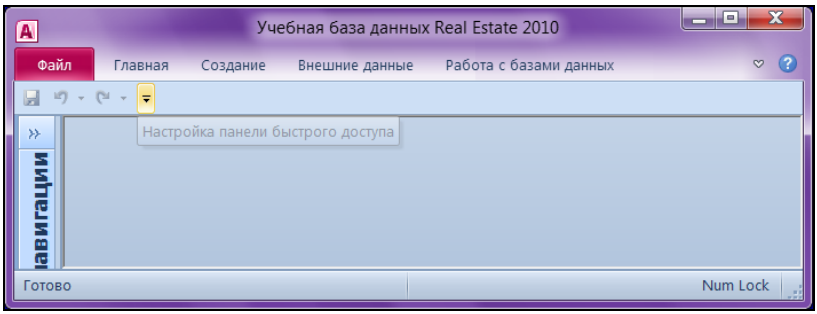


Рис. 2.7. Панель быстрого доступа Microsoft Access 2010

Если раньше для выполнения задач использовалось окно базы данных, то в MS Access 2010 теперь для этой цели служит **Область навигации** (см. рис. 2.1). Например, чтобы изменить значение строки в таблице, отображенной в режиме таблицы, следует открыть таблицу из **Области навигации**. По умолчанию **Область навигации** появляется при открытии базы данных в MS Access 2010 и баз данных, созданных в предыдущих версиях. Можно отключить отображение **Области навигации** по умолчанию, настроив соответствующий параметр. Для этого выберите кнопку **Файл**. В нижней части открывшегося окна щелкните по кнопке **Параметры**. Откроется окно **Параметры Access**. Щелкните по пункту **Текущая база данных**. В разделе **Навигация** снимите флажок в поле **Область навигации.**

2.4. Создание проекта MS Access

Запустите Microsoft Access 2010 на вашем компьютере. Для этого нажмите кнопку **Пуск** и выберите в появившемся меню пункт **Все программы**. Появится список программ, установленных на компьютере. Выберите пункт **Microsoft Office.** В открывшемся меню найдите Microsoft Office Access 2010. Он установлен там по умолчанию. Сделайте щелчок левой кнопкой мыши. На экране появится стартовое окно **Microsoft Access.**

Определитесь с именем файла проекта MS Access, который будет работать в качестве клиента с базой MS SQL Server, и папкой, в которой он будет расположен. Строка для ввода имени файла и поиска папки для размещения проекта находится в правом нижнем углу окна. Обязательно укажите после имени файла расширение adp.

Найдите там же кнопку **Создать** и щелкните по ней. Появится окно запроса на подключение к серверу (рис. 2.8).

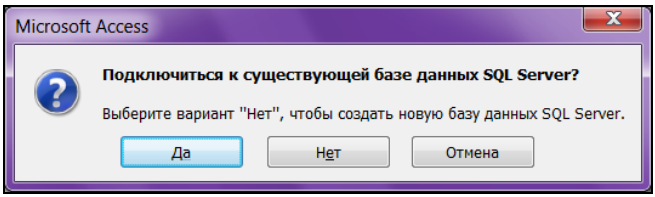


Рис. 2.8. Запрос на подключение к существующей базе данных MS SQL Server

Выберите кнопку **Да**. Появится диалоговое окно **Свойства канала передачи данных** (рис. 2.9). Укажите в нем имя экземпляра сервера, имя пользователя, его пароль и выберите базу данных на сервере. Не забудьте проверить соединение, нажав соответствующую кнопку.

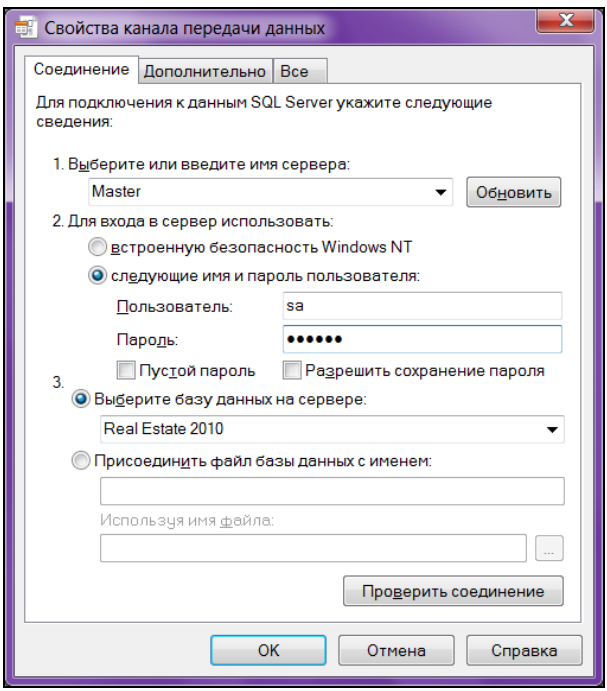


Рис. 2.9. Запрос на выбор сервера и базы данных

2.5. Создание таблиц

Существует несколько способов создания таблиц в Microsoft Office Access 2010:

- создание новой пустой таблицы;
- создание таблицы на основе списка на узле SharePoint;

- создание таблицы с помощью импорта внешних данных;
- создание таблицы при помощи конструктора.

Первые два способа — для самых нетерпеливых новичков в работе с базами данных, испытывающих сильное желание немедленно создать свою первую таблицу. Для создания новой пустой таблицы выберите на второй вкладке ленты **Создание** пункт **Таблица**. На экране появится новая пустая таблица. В ней два столбца: **Код** и **Добавить поле**. Поле **Код** содержит уникальный номер каждой строки этой таблицы и заполняется автоматически по мере добавления записей в таблицу. Замените название столбца **Добавить поле** на свое, например `STREET` (название улицы), и можете сразу составить весь список улиц города. Тип поля `STREET` будет назначен по умолчанию — текстовый с длиной 255 символов, если в настройках MS Office Access 2010 не установлен другой размер.

Для создания новой таблицы на основе списка на узле SharePoint выберите на второй вкладке ленты **Создание** пункт **Списки SharePoint**. Откроется меню шаблонов с пунктами **Контакты**, **Задачи**, **Вопросы** и т. д. Выбирайте нужные, указывая URL-адрес, и проектируйте свою таблицу!

Третий способ — создание таблицы с помощью импорта внешних данных — позволяет импортировать таблицу, расположенную в другом месте. В этом случае в новой таблице текущей базы данных будет создана копия импортированных данных.

Если вы заняты разработкой своего первого приложения и ваши намерения достаточно серьезны, то воспользуйтесь конструктором таблиц. Выберите на второй вкладке ленты **Создание** пункт **Конструктор таблиц**. Откроется окно конструктора с именем новой таблицы **Таблица1**. В окне три колонки. Первые две (**Имя поля** и **Тип данных**) будут использоваться приложением, а третья (**Описание**) предназначена только для разработчика. Не оставляйте ее пустой! Опишите подробно назначение поля таблицы. Позже обязательно поймете важность этого совета.

Каждое поле таблицы должно иметь уникальное имя, но в различных таблицах можно использовать одинаковые имена полей. В табл. 2.11 приведены основные типы данных полей Microsoft Access 2010.

Имена полей должны содержать не более 64 символов и могут включать любые комбинации символов за исключением точки, восклицательного знака и квадратных скобок. Используйте в именах полей только латинские буквы и не применяйте пробелы. В этом случае у вас не будет проблем с конвертацией таблиц Microsoft Access в таблицы других СУБД и вам не придется прибегать к помощи квадратных скобок при работе с полями таблиц в Visual Basic for Applications (VBA).

Примечание

Длину текстового поля по умолчанию можно изменить. Для этого выберите кнопку **Файл**. В нижней части открывшегося окна щелкните по кнопке **Параметры**. Откро-

ется диалоговое окно **Параметры Access**. Щелкните по пункту **Конструкторы объектов**. В разделе **Конструктор таблиц** установите требуемую длину текстового поля. В этом же разделе можно назначить по умолчанию и длину числового поля.

Таблица 2.11. *Типы данных полей таблиц MS Access 2010*

Вид данных	Тип данных	Описание
Символьный	Текстовый	Текст или числа, не требующие проведения расчетов. Максимальная длина — 255 символов. По умолчанию длина текстового поля устанавливается равной максимальной длине
	Поле MEMO	Поля типа MEMO предназначены для хранения больших текстовых данных. Длина поля может достигать 64 Кбайт. Поле не может быть ключевым или индексированным. Поля MEMO полезны для хранения больших объемов информации. При работе с Office Access 2010 можно задать свойство (**Только добавление**), при котором приложение Access сохраняет историю всех изменений поля MEMO. Историю изменений затем можно просмотреть
Числовой	Числовой	Содержит множество подтипов (размеров). От выбора размера зависит точность вычислений. Используйте целый тип для полей, которые используются в ссылках на другие таблицы базы данных
	Счетчик	Уникальные, последовательно возрастающие числа, автоматически вводящиеся в таблицу при добавлении каждой новой записи
	Логический	Содержит одно из двух возможных значений 0 — для представления значения "нет" и –1 (обратите внимание: *минус 1*) для "да"
	Денежный	Позволяет выполнять расчеты с точностью до 15 знаков в целой и до 4 знаков в дробной части
Дата и время	Дата/Время	Семь видов форматов для отображения даты и времени
Произвольный	Поле объекта OLE	Включает рисунок, фотографию, звукозапись, диаграммы, векторную графику, форматированный текст и т. п.
Адреса Web	Гиперсвязь	Содержит адреса Web-страниц
Произвольный	Вложение	Позволяет хранить документы и двоичные файлы любых типов в базе данных без излишнего увеличения ее объема. Для уменьшения общего объема данных вложения автоматически сжимаются. Этот тип данных используется, например, если нужно вложить в запись документ Microsoft Office Word 2007 или сохранить в базе данных набор цифровых изображений. В одной записи можно хранить несколько вложений

В MS Office Access 2010 имеется тип поля таблицы — **Вложение**. Вложение можно использовать для хранения нескольких файлов в одном поле таблицы, причем в этом поле можно хранить файлы самых разных типов. Например, в по-

ле документов, подтверждающих оплату за услуги, можно добавить к записи каждого лицевого счета одно или несколько платежных поручений, а также фотографию владельца и проживающих. Вложения также позволяют хранить данные более рационально. В более ранних версиях MS Access для хранения изображений и документов использовалась технология OLE (Object Linking and Embedding, связывание и внедрение объектов). По умолчанию с помощью технологии OLE создавалась растровая копия изображения или документа. Такие растровые файлы иногда могут быть слишком большими — значительно больше исходного файла. При просмотре изображения или документа из базы данных с помощью технологии OLE отображалось растровое изображение, а не исходный файл. При использовании вложений документы и другие файлы, не являющиеся изображениями, открываются в соответствующих программах, так что эти файлы можно редактировать непосредственно в приложении MS Access 2010. Кроме того, технология OLE требует применения программ, называемых OLE-серверами. Например, если в базе данных Access хранятся файлы изображений в формате JPEG, для каждого компьютера, на котором запущена эта база данных, требуется отдельная программа, зарегистрированная как OLE-сервер для изображений в формате JPEG. Напротив, в MS Access 2010 вложенные файлы сохраняются в исходных форматах без каких-либо вспомогательных изображений, и для просмотра изображений из базы данных установка дополнительного программного обеспечения не нужна.

Предупреждение

После того как в таблицу добавлено поле типа **Вложение**, можно вкладывать файлы в записи этой таблицы, не создавая форму для ввода данных. Кроме того, можно просматривать вложения без помощи формы. Однако помните, что для просмотра непосредственно из таблиц используются программы, в которых создавались эти файлы, или программы, поддерживающие файлы такого типа. Например, при открытии вложенного в таблицу документа MS Office Word запускается также приложение MS Word, и просмотр документа происходит в этом приложении, а не в Access. Если приложение Word не установлено на компьютере, появится диалоговое окно с предложением выбрать программу для просмотра файла.

Создадим нашу первую таблицу `building`. Ее окончательная структура взята из табл. 2.8. Имейте в виду, что имя поля и его описание вводится с клавиатуры, а тип данных выбирается из списка. Список будет доступен только тогда, когда курсор попадет в колонку **Тип данных**. Рассмотрим действия по созданию таблицы подробнее (рис. 2.10).

1. Введите в первую колонку имя первого поля `STREET` и нажмите клавишу <Enter>. Курсор переместится во вторую колонку **Тип данных**. По умолчанию будет назначен тип **Текстовый**.
2. Раскройте список типов данных. Раскрыть список можно при помощи мыши. Выберите тип **Числовой** и нажмите клавишу <Enter>. Заполните колонку **Описание**. Не ленитесь, пишите подробнее!

3. Переведите курсор в область **Свойства поля**. По умолчанию в качестве размера поля стоит значение *Длинное целое*.
4. Сделайте щелчок мышью на этом поле. Раскроется список подтипов числового типа. Выберите подтип *Целое*.
5. Оставьте незаполненным **Формат поля**. Перейдите сразу к строке **Число десятичных знаков**. Для ссылки на номер улицы используем четыре десятичных знака. Это дает возможность работать с 9999 улицами, что вполне достаточно для города с миллионным населением.
6. Повторите шаги 1—5 для всех оставшихся полей таблицы.

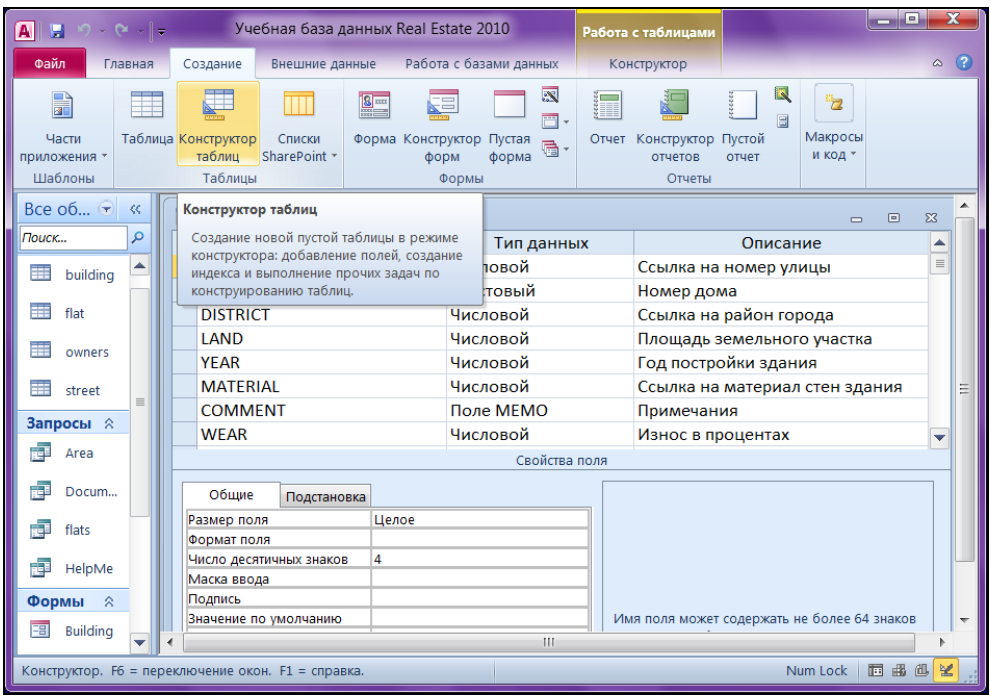


Рис. 2.10. Таблица `building` в режиме конструктора таблиц

После занесения данных обо всех полях таблицы просто закройте окно конструктора таблиц. Появится диалоговое окно, запрашивающее подтверждение на сохранение структуры таблицы (рис. 2.11). Нажмите кнопку **Да**, а в появившемся окне вместо названия **Таблица1** введите имя `building` и щелкните по кнопке **OK**.

Наша первая таблица появится в окне **Все объекты Access**. Заполнять сейчас созданную таблицу начинающему пользователю категорически не рекомендуется, да так и не делается! Посмотрите на содержимое таблицы `building` (табл. 2.12). В ней приведен фрагмент данных после запуска приложения в рабо-

ту. `building` — не одиночная таблица, она связана с другими таблицами базы данных Real Estate 2010. Что в ней за цифры, особенно в столбце `STREET`, скорее всего, до конца непонятно. Так что же заносить в качестве содержимого столбца `STREET`?

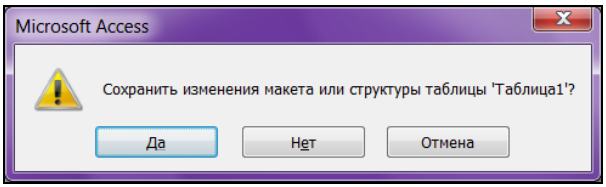


Рис. 2.11. Подтверждение на сохранение

Таблица 2.12. *Информация, содержащаяся в связанной таблице*

STREET	HOUSE	FLAT	STOREY	ROOMS	SQUARE	DWELL	BRANCH	ACCOUNT
14	102	1	1	3	60,8	40	20	3450
14	102	2	1	4	100	70	28	1000
14	102	3	1	4	78	60	16	4321
14	102	4	2	4	90	80	5	666
14	102	5	2	3	100	95	30	778
14	102	6	10	1	200	190	8	9787
14	102	7	10	7	170	150	10	879
179	104	1	1	1	30	20	9	23210
179	104	2	1	2	42	30	11	3267
179	104	3	1	1	27	20	6	6666
179	104	4	2	4	100	90	5	4587

Аналогичным образом создадим все наши таблицы: `flat`, `owners`, `account`, `street`, `district` и `wall`.

2.6. Создание первичных ключей и индексов

Простой первичный ключ — это индекс, созданный по ключевому полю таблицы.

Составной первичный ключ — это индекс, созданный по ключевой связке полей таблицы.

О том, как выбрать ключевое поле или назначить ключевую связку полей для таблицы, рассказано в *разд. 2.2*.

Предупреждение

Первичный ключ у любой таблицы может быть только один. Этого требует теория нормализации.

Кроме первичного ключа таблица может иметь любое количество обычных индексов. Среди них могут быть и уникальные, не допускающие повторяющихся значений. Их принято называть *индексами — кандидатами на роль первичного ключа*. В нашем примере такой индекс есть. Загляните в таблицу `flat` (см. рис. 2.4) и обратите внимание на поле `ACCOUNT` (номер лицевого счета квартиросъемщика). Это поле однозначно определяет положение любой квартиры в таблице.

Индекс можно построить по полю почти любого типа. К счастью, пользователь не обязан знать, за счет чего достигается такое огромное увеличение скорости поиска. Достаточно отметить поле как индексированное, а система Access 2010 позаботится обо всем остальном.

Хочу предостеречь вас от типичной ошибки начинающего разработчика — создания индексов по всем полям таблицы для достижения максимальной скорости поиска в сложных запросах. Во-первых, в этом просто нет необходимости, а во-вторых — возникнет серьезная задержка при добавлении записей в таблицу, т. к. системе придется перестраивать большое число индексов одновременно.

2.6.1. Создание обычного индекса по полю таблицы

Порядок создания, как простого индекса, так и уникального (индекса-кандидата) — один и тот же. Создадим уникальный индекс по полю `ACCOUNT` таблицы `flat`. Порядок действий следующий:

1. Откройте таблицу `flat` в режиме конструктора таблиц (рис. 2.12). Для этого в окне **Все объекты Access** базы данных Real Estate 2010 выделите таблицу `flat` и щелкните по ней правой кнопкой мыши. Появится меню. Выберите в нем второй пункт — **Конструктор**. Выделите поле `ACCOUNT`, нажав кнопку выделения поля в левой части бланка структуры таблицы.
2. Сделайте активным свойство **Индексированное поле**.
3. Поле `ACCOUNT` не может содержать повторяющиеся данные, поэтому в списке необходимо выбрать значение *Да (Совпадения не допускаются)*.
4. Закройте окно конструктора таблиц.
5. Появится диалоговое окно, сообщающее о том, что структура таблицы была изменена. Подтвердите сохранение.

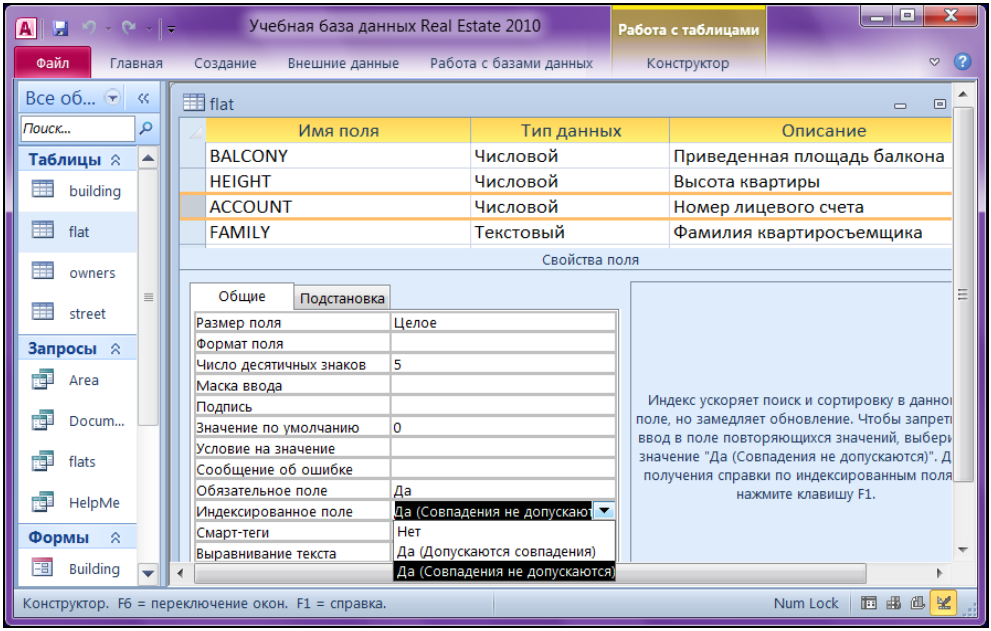


Рис. 2.12. Создание уникального индекса по полю ACCOUNT

Примечание

Для создания простого индекса в свойстве **Индексированное поле** следует указать *Да (Допускаются совпадения)*.

2.6.2. Создание простого первичного ключа

Создадим простой первичный ключ для таблицы account (лицевой счет). Ключевое поле, однозначно определяющее положение любой записи в этой таблице, также носит название ACCOUNT.

Для его создания выполните следующие действия:

1. Откройте таблицу account в режиме конструктора. Для этого в окне **Все объекты Access** базы данных Real Estate 2010 выделите таблицу account и щелкните по ней правой кнопкой мыши. Появится меню. Выберите в нем второй пункт — **Конструктор**.
2. Выделите поле ACCOUNT, нажав кнопку выделения поля в левой части бланка структуры таблицы. Строчка, относящаяся к этому полю, будет выделена рамкой (рис. 2.13).
3. Сделайте щелчок мышью по пиктограмме  **Ключевое поле** вкладки **Конструктор** ленты главного окна Microsoft Office Access 2010. Такой же ключ появится возле поля ACCOUNT (рис. 2.14).

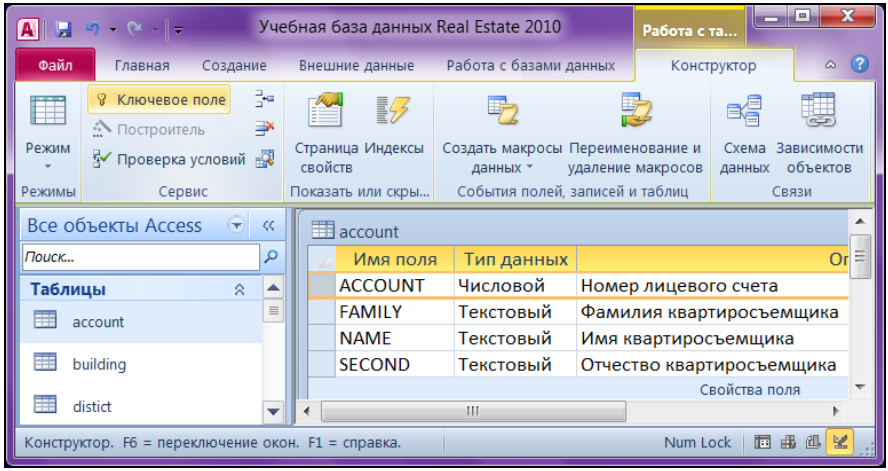


Рис. 2.13. Создание простого первичного ключа

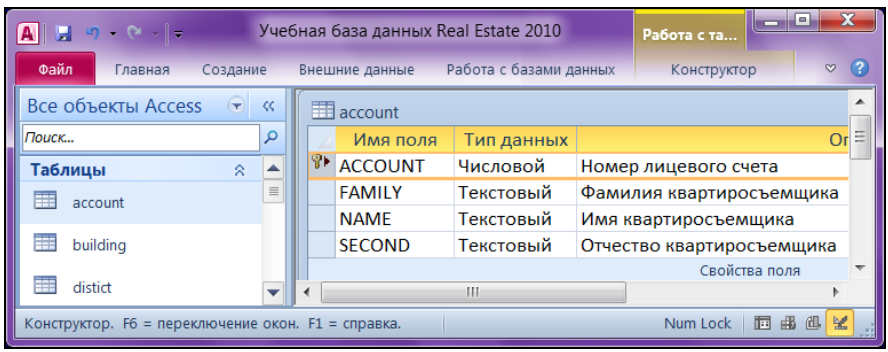


Рис. 2.14. Простой первичный ключ создан

4. Закройте окно конструктора таблиц.
5. Появится диалоговое окно, сообщающее о том, что структура таблицы была изменена. Подтвердите сохранение.

2.6.3. Создание составного первичного ключа

Создадим составной первичный ключ для таблицы `flat` (квартиры). Ранее ключевая связка полей для этой таблицы была определена нами так: `STREET` (номер улицы) плюс `HOUSE` (номер дома) плюс `FLAT` (номер квартиры).

1. Откройте таблицу `flat` в режиме конструктора. Для этого в окне **Все объекты Access** базы данных Real Estate 2010 выделите таблицу `flat` и щелкните по ней правой кнопкой мыши. Появится меню. Выберите в нем второй пункт — **Конструктор**.

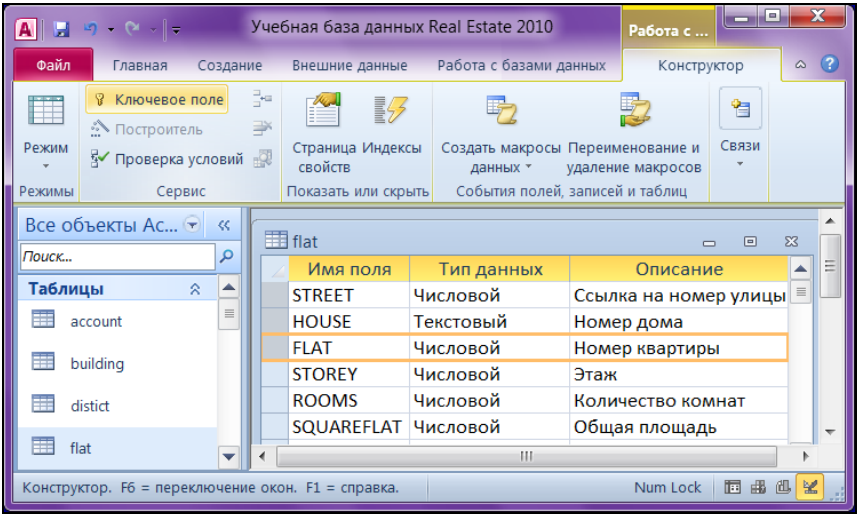


Рис. 2.15. Выделение связки полей таблицы

2. Выделите поле `STREET`, нажав кнопку выделения поля в левой части бланка структуры таблицы. Строчка, относящаяся к этому полю, будет выделена рамкой. Нажмите клавишу <Ctrl> и, удерживая ее, щелкните последовательно кнопки выделения напротив полей `HOUSE` и `FLAT`. Выделенных строчек станет три (рис. 2.15).

3. Сделайте щелчок мышью по пиктограмме  **Ключевое поле** вкладки **Конструктор** ленты главного окна Microsoft Office Access 2010. Стилизованное изображение ключа появится напротив всех трех выбранных полей. Имейте в виду: ключиков три, а первичный ключ таблицы — один, составной. Вы помните, что ранее по полю `ACCOUNT` таблицы `flat` мы уже создали обычный индекс. Давайте посмотрим, что получилось с индексированием нашей таблицы. Сделайте щелчок по пиктограмме  **Индексы**. Откроется окно **Индексы: flat** (рис. 2.16). Комментарии излишни!

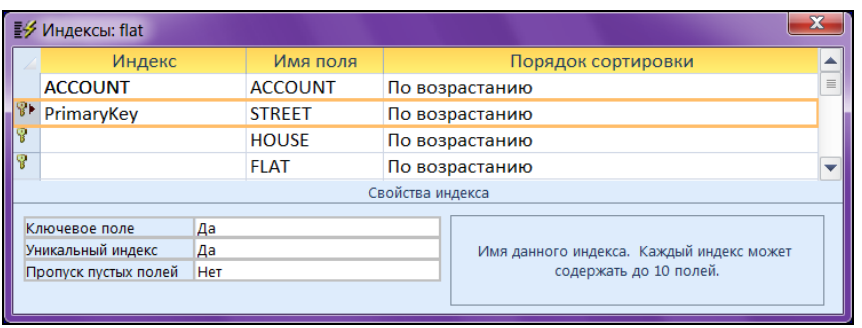


Рис. 2.16. Информация об индексах таблицы

4. Закройте окна индексов и конструктора таблицы и подтвердите сохранение сделанных изменений.

2.7. Контроль правильности ввода данных

Информация, накапливаемая в базе данных, должна обладать абсолютной достоверностью. Несоблюдение этого правила может порой привести к печальным последствиям. Например, отдел комплектации не сделает вовремя заказ на поставку необходимых материалов, владелец квартиры получит квитанцию для оплаты налога на автотранспорт, которого у него никогда не было, а пенсионеру будет отказано в выдаче страхового полиса и т. д. Даже самые опытные пользователи, заполняющие таблицы, могут допустить ошибку и занести неверные данные, что, скорее всего, и произошло в перечисленных выше случаях.

Разработчик программного комплекса просто обязан помочь пользователю избежать большинства ошибок при вводе информации. Далее приведены две возможности, которые любезно предоставили в наше распоряжение авторы Microsoft Access 2010.

2.7.1. Добавление условия на значение поля

Это условие позволяет проверить корректность данных только в одном поле, независимо от значений других полей. Рассмотрим пример, в котором на номер района наложено ограничение. Этот номер не может находиться вне диапазона от 1 до 9, даже если пользователь этого очень захочет (рис. 2.17).

Чтобы добавить условие на значение поля таблицы `district` (район), выполните следующие действия:

1. Откройте таблицу `district` в режиме конструктора. Для этого в окне **Все объекты Access** базы данных Real Estate 2010 выделите таблицу `district` и щелкните по ней правой кнопкой мыши. Появится меню. Выберите в нем второй пункт — **Конструктор**.
2. Поместите текстовый курсор в поле **Условие на значение**.
3. Наберите на клавиатуре `>=1 And <10` и нажмите клавишу <Enter>. Курсор переместится в поле **Сообщение об ошибке**.
4. Введите текст сообщения: *Номер района должен быть в пределах от 1 до 9 включительно*. Если текст не поместится в поле целиком, то система прокрутит его влево. Для перехода в начало текста сообщения нажмите клавишу <Home>.
5. Закройте окно конструктора таблицы и подтвердите сохранение сделанных изменений.

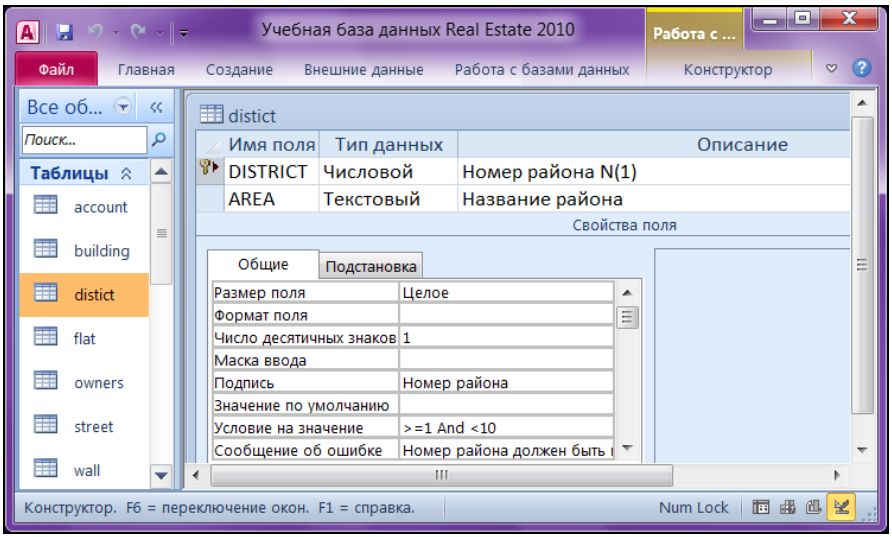


Рис. 2.17. Свойства поля `DISTRICT` (номер района города)

При попытке ввода номера района, который не находится в пределах диапазона 1—9, получим сообщение об ошибке и отказ программного комплекса от записи в таблицу сделанных изменений (рис. 2.18).

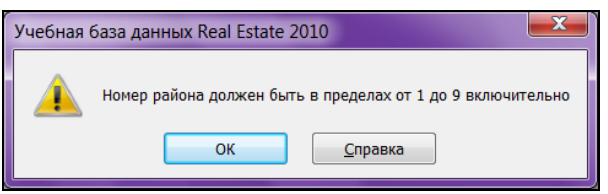


Рис. 2.18. Сообщение при ошибочных действиях оператора

2.7.2. Добавление условия на значение записи

Это условие позволяет сравнить значения нескольких полей сразу. Рассмотрим пример, в котором производится проверка соответствия общей площади квартиры сумме составляющих: жилой, вспомогательной и приведенной площади балкона. Для того чтобы добавить условие на значение записи, необходимо проделать следующие действия:

1. Откройте таблицу `flat` в режиме конструктора. Для этого в окне **Все объекты Access** базы данных Real Estate 2010 выделите таблицу `flat` и щелкните по ней правой кнопкой мыши. Появится меню. Выберите в нем второй пункт — **Конструктор**.

2. Сделайте щелчок мышью по пиктограмме  **Страница свойств** вкладки **Конструктор** ленты главного окна Microsoft Office Access 2010. Появится диалоговое окно свойств таблицы (рис. 2.19). Это же окно можно активизировать щелчком правой кнопки мыши в любом месте окна конструктора таблицы и выбором последнего пункта **Свойства** в появившемся меню.

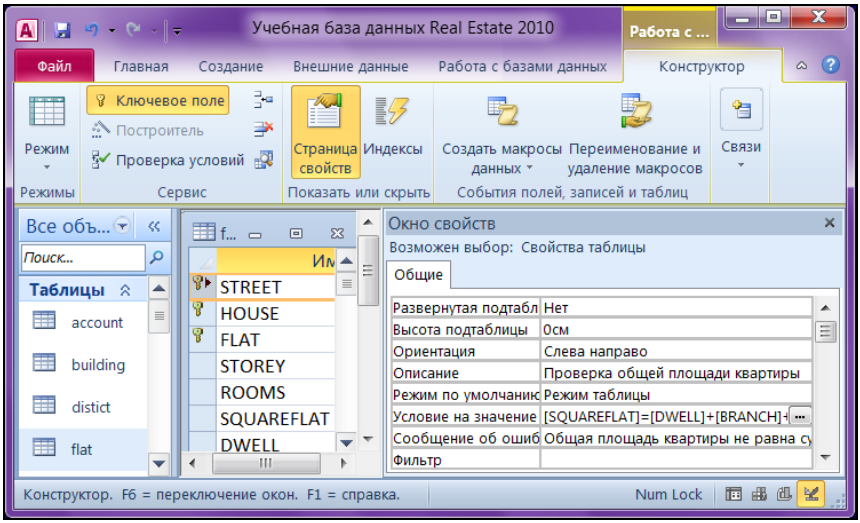


Рис. 2.19. Окно свойств таблицы вкладки **Конструктор**

3. Введите в поле **Описание** краткое назначение выполняемой проверки: *Проверка общей площади квартиры*.
4. Переместите курсор в поле **Условие на значение**. Появится кнопка  в его правой части. Нажмите ее.
5. Появится диалоговое окно **Построитель выражений**. В нем без труда вы найдете список всех полей таблицы `flat`. Выполните двойной щелчок левой кнопкой мыши по элементу `SQUAREFLAT`. В окне создаваемого выражения появится `[SQUAREFLAT]` (рис. 2.20).
6. Введите с клавиатуры знак равенства и выполните двойной щелчок по элементу `DWELL`, чтобы добавить его в выражение. Аналогичные действия по отношению к элементам `BRANCH` и `BALCONY` приведут вас к окончательному виду:

```
[SQUAREFLAT]=[DWELL]+[BRANCH]+[BALCONY]
```

7. Закройте окно **Построитель выражений**, щелкнув по кнопке **OK**.
8. Поместите текстовый курсор в поле **Сообщение об ошибке** в **Окне свойств** (см. рис. 2.19). Введите текст, который будет появляться всякий раз при нарушении условия равенства площадей: *Общая площадь квартиры не равна сумме составляющих*.

9. Закройте окно конструктора таблицы и подтвердите сохранение сделанных изменений.

В случае неравенства площадей при работе программного комплекса появится сообщение (рис. 2.21).

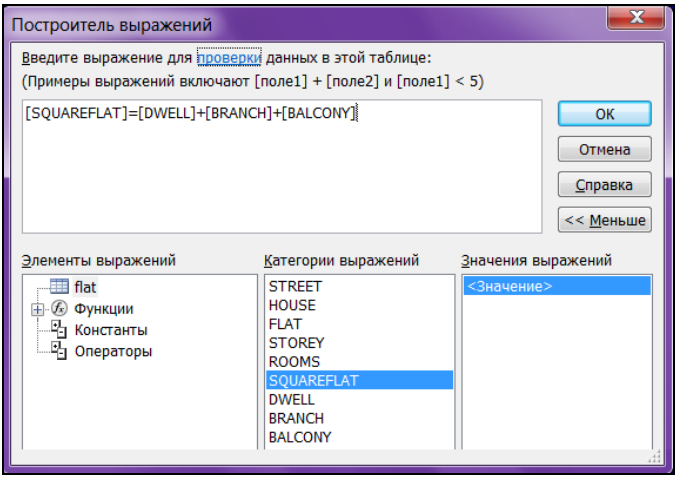


Рис. 2.20. Построитель выражений Microsoft Access 2010

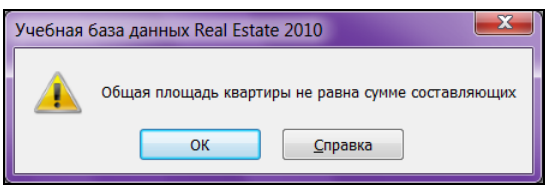


Рис. 2.21. Сообщение об ошибке

2.8. Создание связей между таблицами

Подведем итоги. База данных имеется. Таблицы доведены до третьей нормальной формы и помещены в базу. Первичный ключ есть у каждой таблицы. Индексы созданы. Типы связей между таблицами определены. Настало время создания связей между таблицами непосредственно в базе данных. Связи между таблицами назначают и просматривают в специальном окне **Схема данных** (рис. 2.22).

2.8.1. Создание связи "один-ко-многим"

Определим связь между таблицами `district` (районы) и `building` (здания). Это связь "один-ко-многим". В одном районе города расположено несколько зданий.

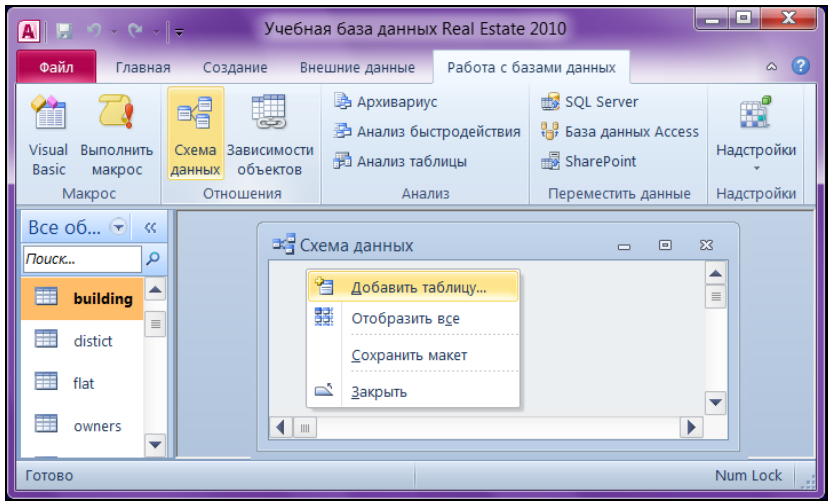


Рис. 2.22. Начальный этап создания схемы данных

1. Сделайте активной вкладку **Работа с базами данных** ленты главного окна Microsoft Access 2010. Выберите пиктограмму **Схема данных** (см. рис. 2.22).
2. Выполните щелчок правой кнопкой мыши в любом свободном месте появившегося окна **Схема данных**.
3. В появившемся меню выберите первый пункт — **Добавить таблицу**.
4. Появится диалоговое окно **Добавление таблицы**. Раскройте вкладку **Таблицы**.
5. В списке таблиц выберите `district` (районы) и нажмите кнопку **Добавить**. Нажмите кнопку **Закрыть**.
6. Таблица `district` появится в окне **Схема данных** (рис. 2.23). Выполните аналогичные действия с таблицей `building` (Здания).
7. Связь между таблицами `district` и `building` строится по значению одноименных полей `DISTRICT`. Поместите указатель мыши над полем `DISTRICT` (оно ключевое и поэтому выделено в списке полей стилизованным изображением ключа), нажмите левую кнопку мыши и, не отпуская ее, "перетащите" появившийся значок поля на поле `DISTRICT` таблицы `building`. Отпустите левую кнопку мыши. Появится диалоговое окно **Изменение связей** (рис. 2.24).
8. Поставьте флажок **Обеспечение целостности данных** и нажмите кнопку **Создать** для подтверждения создания связи и перехода в окно **Схема данных**. Microsoft Access 2010 использует назначенные связи при создании форм, запросов и отчетов, которые требуют данных из рассмотренных выше таблиц.

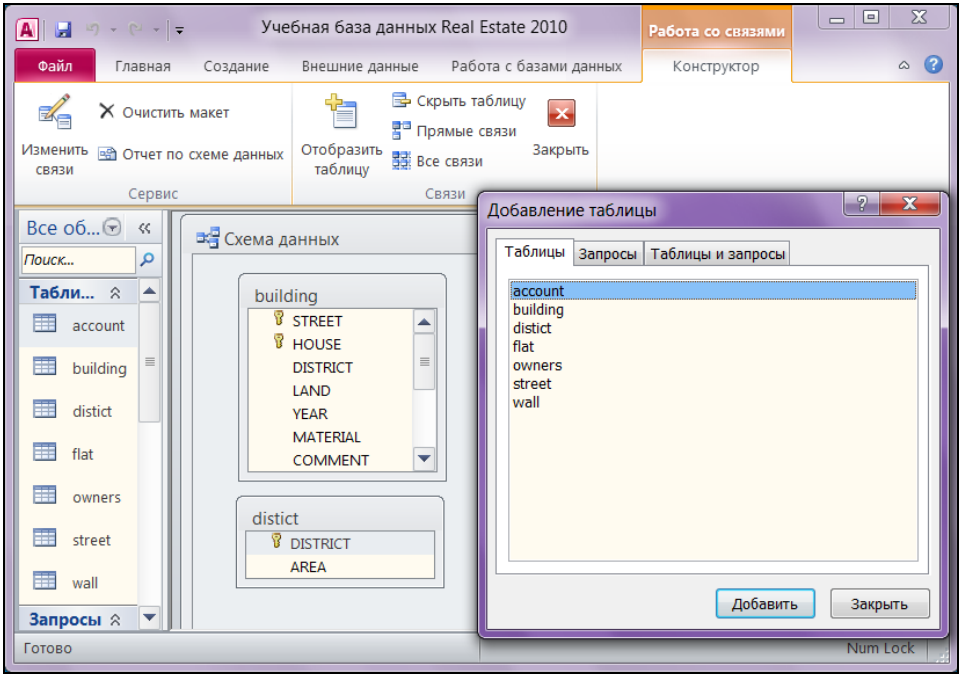


Рис. 2.23. Добавление таблицы `building` в схему данных

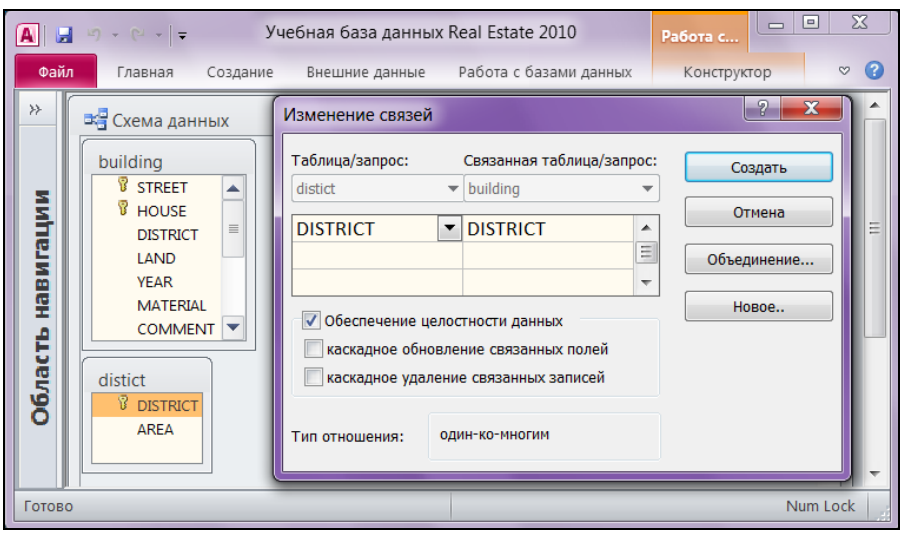


Рис. 2.24. Создание связи "один-ко-многим" между таблицами `district` и `building`

Предупреждение

Со стороны таблицы "один" началом "перетаскивания" обязательно должно быть ключевое поле, а со стороны "многие" — индексированное. Причем индексированное поле в атрибуте **Индексированное поле** должно иметь атрибут: *Да (Допускаются совпадения)*.

Важной особенностью MS Access 2010 является автоматическое обеспечение ссылочной целостности данных. Если на связь между таблицами наложены условия ссылочной целостности, то добавление в связанную таблицу записи, для которой нет соответствующих записей в главной таблице, становится невозможным.

Проверка целостности данных может осуществляться и программными средствами. Например, при добавлении в таблицу `building` описания нового здания вы можете проверить, имеется ли в таблице `district` район, в котором расположено это здание. Однако более правильным является определение условия целостности данных на уровне базы данных, т. к. в этом случае ни одно приложение не может нарушить целостность данных. С базой данных может работать несколько приложений, в том числе и не только ваших.

Примечание

Если разработчик поставит флажок **каскадное обновление связанных полей**, то у него появится возможность исправить номер района лишь в таблице `district`, а в таблице `building` все связанные записи система MS Access исправит автоматически.

Поставленный флажок **каскадное удаление связанных записей** позволит вам смело удалить район в таблице `district`, а все описания зданий этого района Access 2010 удалит без вашего участия. Задумайтесь о том, нужен ли вам этот флажок? Удалили одну запись, а лишились четверти всей базы данных!

Рассмотрим создание связи между таблицами в случае, когда одна из них имеет составной первичный ключ. Посмотрите на рис. 2.4. Для этой цели подходят таблицы `flat` и `owners`. У таблицы `flat` ключевая связка полей выглядит так: `STREET+HOUSE+FLAT`.

Для построения связи между этими таблицами выполните следующие действия:

1. Поместите указатель мыши над полем `STREET` таблицы `flat` и сделайте щелчок левой кнопкой мыши.
2. Нажмите клавишу <Shift> и, не отпуская ее, сделайте сначала щелчок по полю `HOUSE`, а затем `FLAT`. Отпустите клавишу <Shift>. Будет выделена группа из трех полей: `STREET+HOUSE+FLAT`.
3. Поместите указатель мыши над выделенной группой из трех полей таблицы `flat`, нажмите левую кнопку мыши и, не отпуская ее, "перетащите" появившийся значок связки полей в любое место таблицы `owners`.

4. Отпустите левую кнопку мыши. Появится диалоговое окно **Изменение связей** (рис. 2.25). В отличие от связывания таблицы, имеющей простой первичный ключ (см. рис. 2.24), где связь `DISTRICT-DISTRICT` была установлена автоматически, в данном случае необходимо явно указать, какие связки полей из таблиц `flat` и `owners` участвуют при создании связи между этими таблицами.
5. Используя поле с раскрывающимся списком, установите нужные вам связи.
6. Поставьте все три флажка, обеспечивающие ссылочную целостность. Каскадное удаление связанных записей здесь вполне уместно. В последнее время часто две соседние квартиры приобретает одна семья. Без третьего флажка (рис. 2.25) удалить данные по квартире и одновременно по проживающим нельзя. Поставьте его. Этим вы значительно облегчите жизнь пользователю вашей программы.

Окончательный вид связи между таблицами `flat` и `owners` будет таким, как на рис. 2.25.

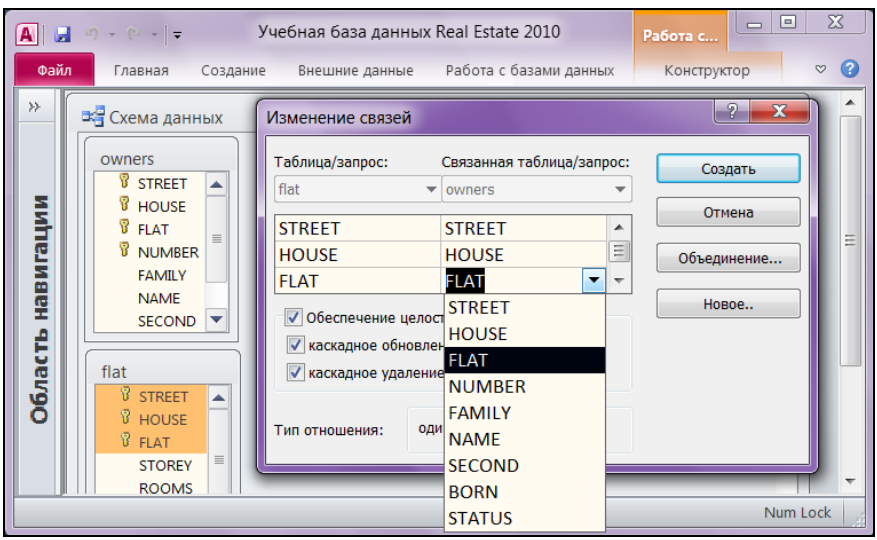


Рис. 2.25. Создание связи "один-ко-многим" между таблицами `flat` и `owners` с использованием составного первичного ключа

2.8.2. Создание связи "один-к-одному"

Такая связь должна быть установлена между таблицами `account` и `flat`, т. к. каждая квартира имеет свой уникальный номер лицевого счета. Для построения связи выполните следующие действия:

1. Поместите указатель мыши над полем `ACCOUNT` (оно ключевое и поэтому выделено в списке полей стилизованным изображением ключа).

2. Нажмите левую кнопку мыши и, не отпуская ее, перетащите появившийся значок поля на поле ACCOUNT таблицы flat. Отпустите левую кнопку мыши. Появится диалоговое окно **Изменение связей**.
3. Поставьте флажок **Обеспечение целостности данных** и нажмите кнопку **Создать** для подтверждения создания связи и перехода в окно **Схема данных**.

Предупреждение

Началом перетаскивания обязательно должно быть ключевое поле (таблица account). Конечная цель — индексированное поле (таблица flat). Индексированное поле в атрибуте **Индексированное поле** должно иметь значение: *Да (Совпадения не допускаются)*.

2.9. Устранение проблем, возникающих при создании ключей

Вы создали свою первую таблицу, и вам, конечно же, не терпится занести в нее данные, тем более что MS Access без труда позволяет это сделать. Не торопитесь! Завершите создание базы данных, назначьте первичные ключи и установите связи между таблицами. Нарушение этого порядка может доставить вам массу неприятностей. Вот одна из них.

В таблицу flat (квартиры) занесена 16 291 запись. Первичный ключ у этой таблицы, конечно же, не создан. Начальство торопит с выполнением проекта, да к тому же выделило работника для занесения информации. Вот вам — результат! Информация обо всех квартирах занесена, вы пытаетесь создать первичный ключ у этой таблицы, а в ней есть повторяющиеся записи. Наборщик занес одну или несколько квартир более одного раза. Не удивительно, ведь ваш программный комплекс его никак не контролировал. Не отчаивайтесь, а выполните следующие действия.

1. Сделайте активной вторую страницу вкладки с названием **Создание** ленты главного окна Microsoft Access 2010. Щелкните по пиктограмме **Мастер запросов**  , расположенной в разделе **Макросы и код**. Появится окно (рис. 2.26).
2. Выберите пункт **Повторяющиеся записи** и нажмите кнопку **OK**.
3. Появится список таблиц базы данных. Выберите в нем таблицу flat и нажмите кнопку **Далее**.
4. В появившемся диалоговом окне **Поиск повторяющихся записей** занесите в окно **Поля с повторами** три поля: STREET, HOUSE, FLAT, используя кнопку > (рис. 2.27).
5. Нажмите кнопку **Готово**, и результат перед вами (рис. 2.28).

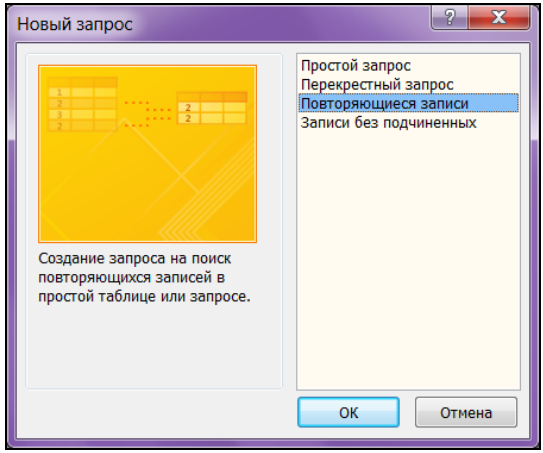


Рис. 2.26. Создание нового запроса для поиска повторяющихся записей в таблице

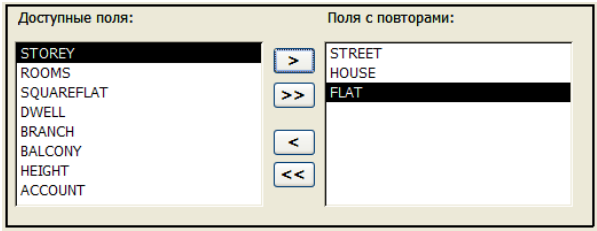


Рис. 2.27. Выбор полей

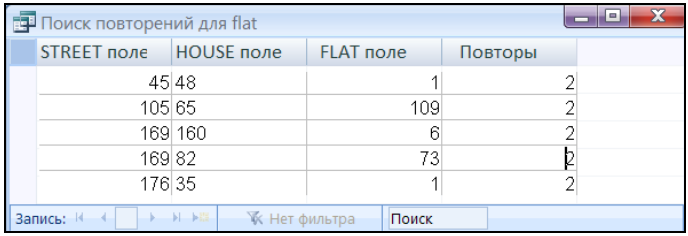


Рис. 2.28. Повторяющиеся записи в таблице `flat`

Осталось удалить из таблицы `flat` пять квартир, которые занесены по два раза. Для этого вернитесь в окно базы данных Real Estate 2010, удалите запрос, который называется "Поиск повторений для flat", и начните все с самого начала. Как ни странно, это самый быстрый путь к цели. Выполните пункты 1—4, а в пункте 5 вместо кнопки **Готово** нажмите кнопку **Далее**. Вам будет дана возможность выбрать дополнительные поля, которые будут отображены вместе с повторяющимися значениями. Советую вам выбрать все поля, нажав кнопку [>>]. После

этого нажмите кнопку **Готово**, и на экране появятся все записи, которые дублируют друг друга по составному ключу (рис. 2.29).

Поиск повторений для flat

STREET	HOUSE	FLAT	STOREY	ROOMS	SQUAREFLAT	DWELL	BRANCH	BALCONY	HEIGHT
45	48	1	1	2	42,4	28,8	13,6	0	2,5
45	48	1	1	1	30,4	18	12,4	0	2,5
105	65	109	1	1	31,8	11,3	20,5	0	2,5
105	65	109	1	4	89,7	60,5	24,6	4,6	2,5
169	160	6	2	1	7,5	7,5	0	0	3,26
169	160	6	2	2	54,1	36,7	17,4	0	3,26
169	82	73	1	1	33	17	15,2	0,8	2,51
169	82	73	1	3	67,8	42,3	22,6	2,9	2,51
176	35	1	1	1	38,5	19,7	15,2	3,6	2,5
176	35	1	1	2	54,3	30,4	20,3	3,6	2,5

Рис. 2.29. Расшифровка повторяющихся записей в таблице `flat`

На рис. 2.29 хорошо видны ошибки, которые сделал наборщик информации. Посмотрите на первые две строчки. Квартира номер 1, находящаяся в доме 48, расположенном на улице со ссылочным номером 45, один раз занесена как двухкомнатная с площадью 42,4 м^2, а во второй раз — как однокомнатная с площадью 30,4 м^2. Очевидно, что одну из этих квартир следует удалить. Разберитесь — какую именно.

Совет

Для удаления записи в таблице, находящейся в режиме просмотра, выделите ее, нажав кнопку выделения строки таблицы в левой части бланка. Вся строка после этого будет выделена рамкой. Нажмите клавишу <Delete>. Правильно ответьте на запрос системы (рис. 2.30).

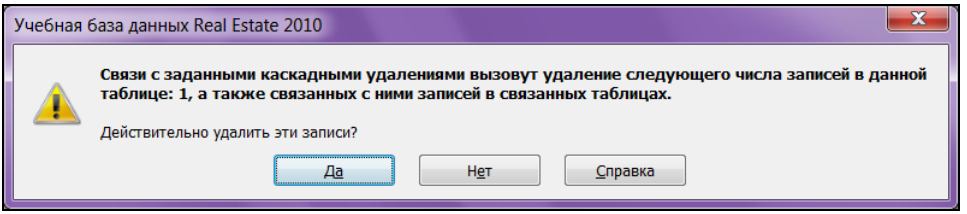


Рис. 2.30. Запрос на подтверждение удаления записи

2.10. Устранение связи "многие-ко-многим"

В качестве примера рассмотрим функционирование фирмы "Столица" (см. вариант 1 заданий на компакт-диске). Особенностью работы этой фирмы является

посредническая деятельность — стыковка поставщиков товаров и покупателей. Один поставщик связан с несколькими покупателями. Один покупатель, в свою очередь, связан с несколькими поставщиками. По понятным причинам поставщик "не знает" покупателя, и наоборот. Вот вам предпосылка создания связи "многие-ко-многим" между двумя таблицами: поставщики и покупатели (рис. 2.31).

Поставщики

ИНН поставщика	Название поставщика	Штрих-код	Название товара	Дата выпуска	Цена за 1 ед.	К-во
252206156	«Океан-2»	46052461	Печенье	01.06.2008	95-00	200
772654321	«Октябрь»	78690543	Печенье	12.07.2008	123-50	450
772654321	«Октябрь»	78695544	Шоколад	15.05.2008	43-00	500
772654321	«Октябрь»	78694455	Конфеты	14.06.2008	105-00	350
278765433	«Амур пиво»	46052462	Печенье	14.07.2008	97-00	600

∞ ∞

Покупатели

ИНН покупателя	Название покупателя	Название товара	Дата покупки	Цена за 1 ед.	К-во	Штрих-код
276546373	«Татьяна»	Печенье	18.07.2008	105-00	100	46052462
276546373	«Татьяна»	Печенье	18.07.2008	140-00	150	78690543
273894756	«Светлый-3»	Печенье	21.07.2008	110-00	50	46052461

Рис. 2.31. Пример связи "многие-ко-многим"

Как мы уже знаем, реляционная модель не позволяет непосредственно реализовать связь типа "многие-ко-многим". Кроме этого, штрихкод товара (рис. 2.32), являясь его однозначной идентификацией, не может быть назначен в качестве первичного ключа. В случае если через какое-то время будет закуплена еще одна партия такого же товара, то в таблице поставщиков появится строчка, имеющая идентичный штрихкод. Аналогичная ситуация возможна и с покупателем, который приобретет такой же товар из другой партии. И еще одна проблема: один клиент практически никогда не покупает всю партию целиком, и перед фирмой встает проблема учета остатка товара.

Рис. 2.32. Штрихкод

Все эти проблемы (реляционной модели и алгоритма) легко решаются добавлением двух промежуточных таблиц: "Поставленные товары" и "Проданные товары". На рис. 2.33 показан фрагмент схемы данных превращения связи "многие-ко-многим" в несколько связей "один-ко-многим" с добавлением этих промежуточных таблиц.

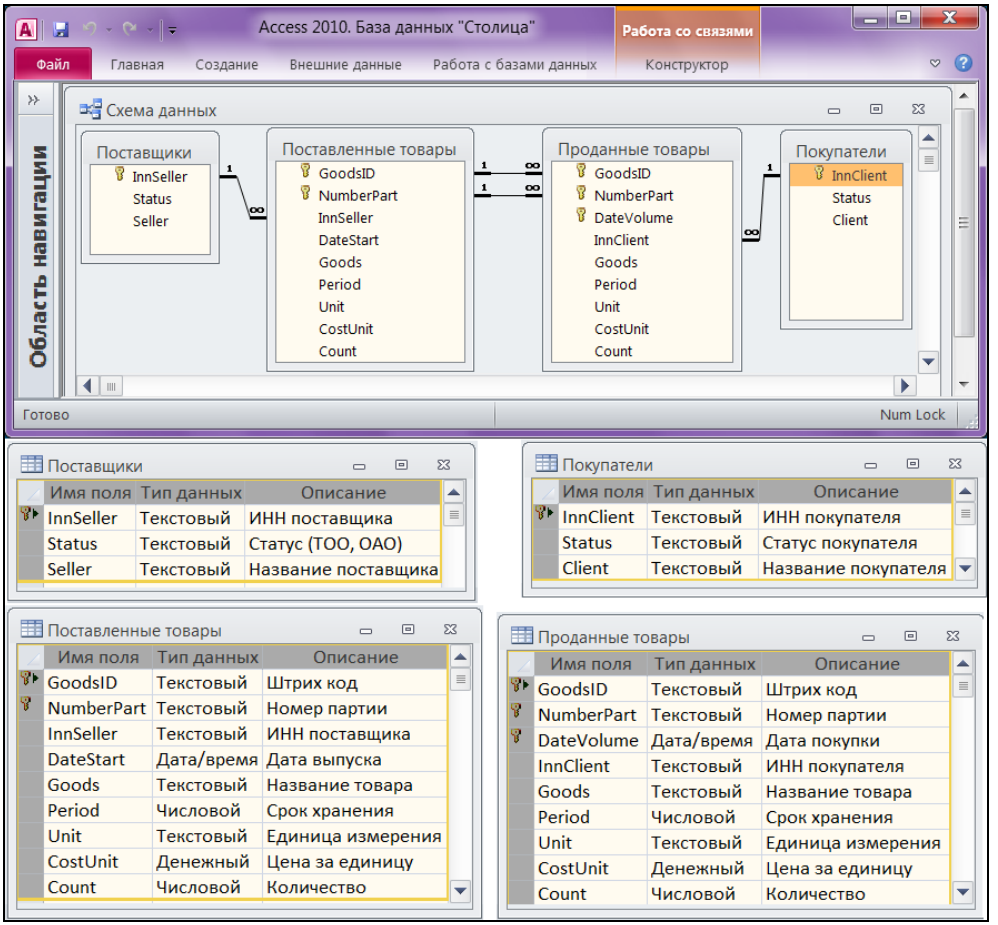


Рис. 2.33. Фрагмент схемы данных

Преобразование связи "многие-ко-многим" в несколько связей "один-ко-многим" и "многие-к-одному" можно сделать следующим образом. Вместо того чтобы рассматривать множество поставщиков, поставляющих товары многим клиентам, будем рассматривать одного поставщика и множество поставляемых им товаров ("один-ко-многим"). Далее будем рассматривать деление товара одной партии на множество частей, подлежащих продаже ("один-ко-многим"). В результате

получим множество товаров, приобретаемых одним покупателем ("многие-к-одному").

Появление таблицы "Проданные товары" позволяет решить внутреннюю проблему учета остатков товара в фирме "Столица".

Примечание

Особенностью реализации предложенной схемы является наличие составного первичного ключа в таблице "Поставленные товары" по связке полей "штрих-код товара" плюс "номер партии" (`GoodsID + NumberPart`).

ГЛАВА 3

Создание форм и отчетов MS Access 2010

MS Access 2010 дает нам возможность ввода и редактирования данных непосредственно в режиме таблицы. Однако для конечного пользователя программного комплекса этого явно недостаточно — из-за его низкой квалификации. Пользователь должен работать с законченным программным продуктом. Его не интересует ни реляционная модель данных, ни то, что информация хранится в таблицах, и что существуют первичные ключи и триггеры. Он желает знать только одно: щелчком по какой кнопке он добьется желаемого результата. Так предоставим же ему эту возможность.

Форма MS Access 2010 — это объект базы данных, который можно использовать для ввода, изменения или отображения данных из таблицы или запроса. Форма может использоваться как стартовая точка вашего приложения. Для автоматизации часто выполняемых действий формы содержат так называемые элементы управления, с помощью которых осуществляется доступ к данным. Формы можно рассматривать как окна, через которые пользователи могут просматривать и изменять базу данных. Рационально построенная форма ускоряет работу с базой данных, поскольку пользователям не требуется искать то, что им нужно. Внешне привлекательная форма — достойный элемент интерфейса. Она делает работу с базой данных более приятной и эффективной, кроме того, она может помочь в предотвращении неверного ввода данных. В MS Access 2010 предусмотрен ряд средств, помогающих быстро создавать формы, а также новые типы форм и функциональные возможности, благодаря которым база данных становится более практичной.

3.1. Автоматическое создание формы на основе таблицы

В направлении полной автоматизации работы с данными резко улучшает положение дел такая замечательная способность MS Access, как автоматическое создание форм. Перейдите на вторую вкладку ленты **Создание** в раздел **Формы**

(рис. 3.1). В нашем распоряжении несколько способов отображения информации из таблиц в формах:

- создание формы с помощью инструмента **Форма**;
- создание формы при помощи инструмента **Пустая форма**;
- создание Web-формы, в которой отображаются несколько записей, при помощи инструмента **Несколько элементов**;
- создание разделенной формы при помощи инструмента **Разделенная форма**;
- создание формы в виде сводной диаграммы при помощи инструмента **Сводная диаграмма**;
- создание формы в виде сводной таблицы при помощи инструмента **Сводная таблица**.

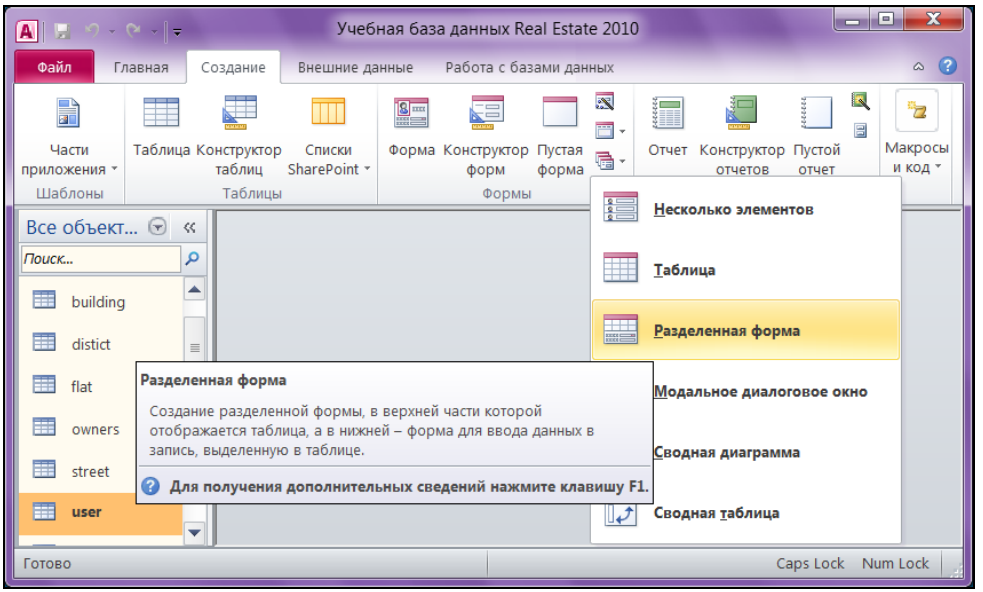


Рис. 3.1. Раздел **Формы** вкладки **Создание** ленты MS Access 2010

Для создания формы с помощью инструмента **Форма** в области навигации выберите таблицу с данными, которые должны отображаться в форме, и сделайте щелчок мышью по пиктограмме этого инструмента. MS Access 2010 создаст форму и отобразит ее в режиме макета. В режиме макета можно внести изменения в структуру формы при одновременном отображении данных. При необходимости можно настроить размер полей в соответствии с данными. Созданная форма со стандартной Access-линейкой навигации по записям готова к работе.

Заслуживает особого внимания инструмент **Разделенная форма**. Созданная им форма одновременно отображает данные в режиме формы и в режиме таблицы.

Эти два отображения связаны с одним и тем же источником данных и всегда синхронизированы друг с другом. При выделении поля в одной части формы выделяется то же поле в другой части. Данные можно добавлять, изменять или удалять в каждой части формы. Работа с разделенной формой дает преимущества обоих типов формы в одной форме. Например, можно воспользоваться нижней (табличной) частью формы, чтобы быстро найти запись, а затем просмотреть или изменить запись в верхней части формы. Для отображения нужных записей можно воспользоваться их сортировкой. Работают и все другие стандартные инструменты MS Access. На рис. 3.2 показана форма, созданная на основе таблицы `user` (работники) учебной базы данных Real Estate 2010.

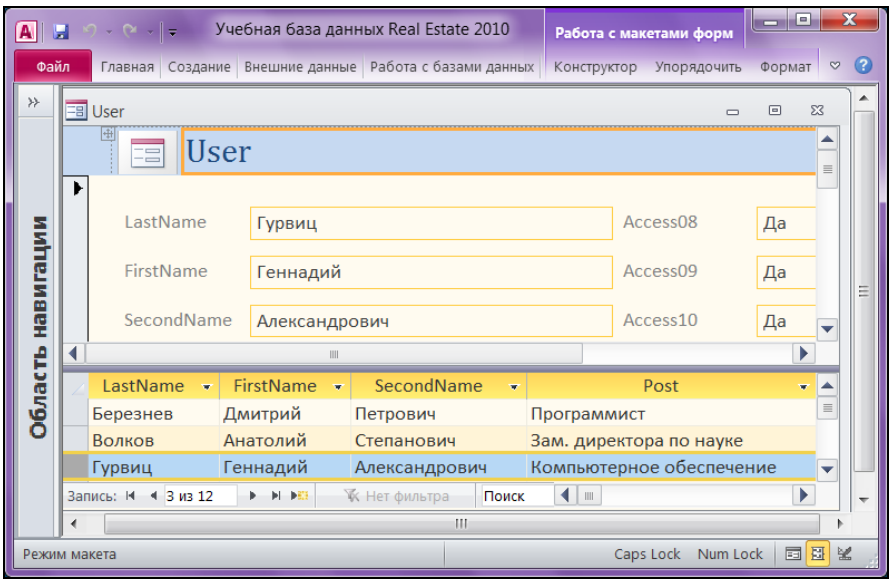


Рис. 3.2. Результат работы инструмента **Разделенная форма**

Примечание

В каком бы месте MS Office Access 2010 вы бы не находились, нажмите клавишу <Alt>, и на экране появятся названия клавиш, нажав которые можно вызвать тот или иной пункт интерфейса. Повторное нажатие <Alt> погасит подсказки.

Совет

Запустите несколькими щелчками мыши и все остальные инструменты быстрого создания форм. Отдельное внимание уделите диаграммам. Посмотрите на результаты их работы. Они просто незаменимы в процессе общения с заказчиком на этапе технического задания. Работодатель, будучи не очень осведомленным в деталях нашей профессии, несомненно, примет эти манипуляции за вашу "ночную" разработку, ведь на экране фамилии, названия товаров и др. из области деятельности его родного предприятия, да еще в таком презентабельном виде!

3.2. Применение мастера для создания формы

MS Access 2010 имеет в своем арсенале еще одно средство для быстрого создания формы — мастер форм. С его помощью можно создавать формы как на основе одной таблицы или запроса, так и на основе нескольких связанных таблиц. Освоить работу с мастером — хорошая идея, которая приведет к значительной экономии времени разработчика, но это не значит, что мастера — это всегда лучший способ. Более солидные результаты дает создание формы с помощью мастера с последующим усовершенствованием ее в режиме конструктора.

Мастер форм разбивает процесс создания формы на несколько этапов. На каждом из них выбираются определенные параметры в предложенном диалоговом окне. Если на одном из этапов сделана ошибка и необходимо изменение уже выбранных параметров, то мастер всегда позволяет вернуться к предыдущему шагу.

Создадим при помощи мастера форму `Building` (здания). Она отображает информацию из главной таблицы `building` и двух таблиц-справочников: `street` (улицы) и `district` (районы).

1. Для запуска мастера форм выберите пункт **Мастер форм**, расположенный на вкладке ленты **Создание** в разделе **Формы** (см. рис. 3.1).
2. Появится первое окно мастера (рис. 3.3). Раскройте поле со списком **Таблицы и запросы**. Выберите в нем таблицу `building`.

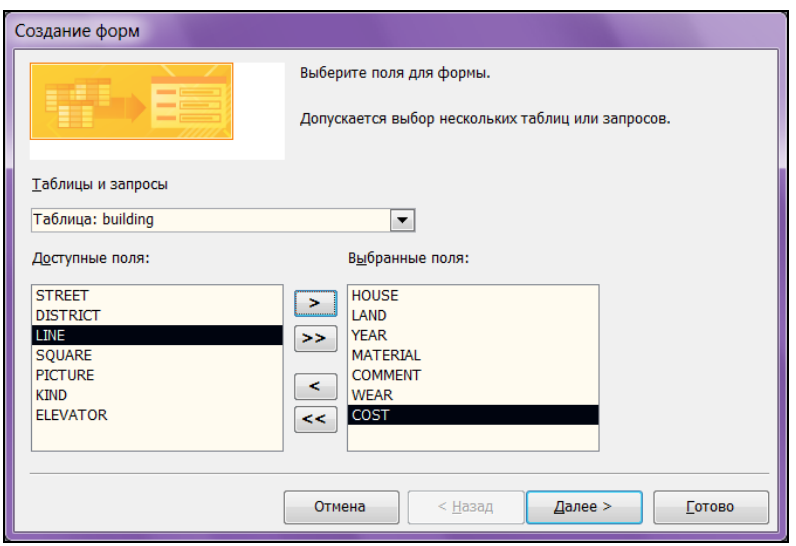


Рис. 3.3. Первый шаг работы мастера форм

3. В поле со списком **Доступные поля** отображены все поля выбранной таблицы `building`. Выберите только те из них, которые следует отобразить в создаваемой форме. Используйте для этого кнопку [>]. Чтобы добавить все поля из таблицы, примените кнопку [>>]. В таблице `building` не следует выбирать поля `STREET` и `DISTRICT`. Они содержат числа-ссылки, а вовсе не названия улиц и районов. Им в форме нечего делать. Понятную пользователю информацию об этих объектах надо взять в таблицах `street` и `district`. Это поля `NAME`, `SIGN` и `AREA`.
4. Для перехода ко второму шагу работы мастера форм нажмите кнопку **Далее**. Появится второе окно мастера (рис. 3.4). Существует несколько видов форм в зависимости от представления на них данных. Некоторые из них: в один столбец, ленточный, табличный, выровненный, сводная таблица, сводная диаграмма. Мастер предлагает нам выбрать один, но только из четырех. Пусть это будет первый вид: **в один столбец**.
5. На третьем шаге требуется указать название формы и выбрать опцию переключателя **Дальнейшие действия**. Опций две:
   - **Открыть форму для просмотра и ввода данных**;
   - **Изменить макет формы**.
6. Если вы хотите внести свои изменения в форму, созданную мастером, то выберите **Изменить макет формы** и нажмите кнопку **Готово**.
7. Рядом с созданной формой появится окно **Список полей**. Двойной щелчок по нужному полю из связанной таблицы — и оно появится в форме. Добавим `AREA` (название района) и `NAME` (название улицы).

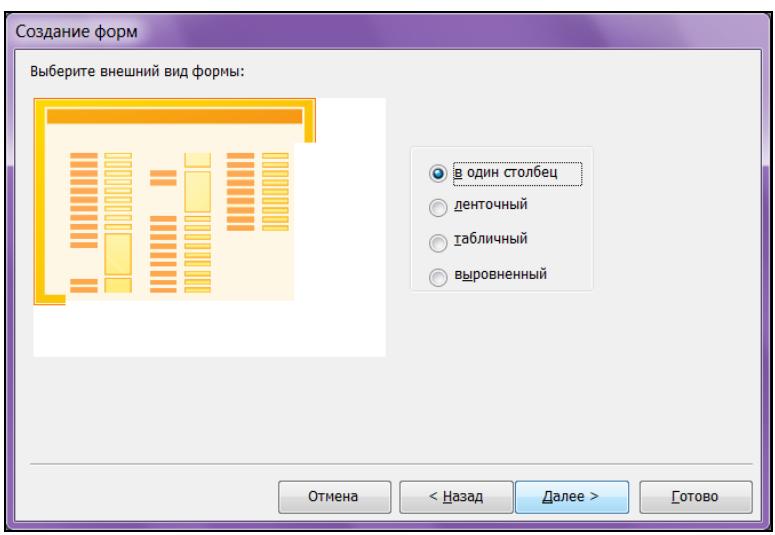


Рис. 3.4. Второй шаг работы мастера форм

8. Остается заменить английские названия полей русскими. Щелкните по нужному полю и вводите текст на русском языке.

В результате мы получили форму в соответствии с выбранными параметрами. На рис. 3.5 представлена форма `Building` в режиме формы. Она готова к использованию, но в таком виде лучше ее не включать в состав программного комплекса. Требуется определенная доработка. Ее можно выполнить в режиме конструктора форм.

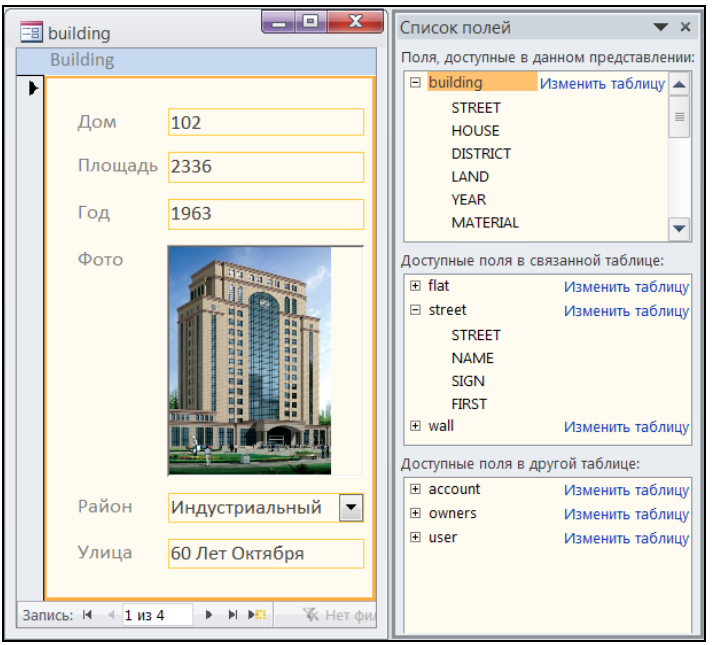


Рис. 3.5. Результат работы мастера форм

Любая форма, так же как и таблица базы данных MS Access 2010, может быть выведена на печать. Работа с формами может вестись в пяти режимах:

- в режиме формы;
- в режиме конструктора;
- в режиме таблицы;
- в режиме сводной таблицы;
- в режиме сводной диаграммы.

Для выбора режима работы с формой найдите ее в области навигации и сделайте двойной щелчок мышью по ее имени. Форма откроется в режиме формы. Этот режим является основным. В нем всегда работают рядовые пользователи про-

граммного комплекса (просмотр записей, редактирование, добавление и удаление). Сделайте щелчок по форме правой кнопкой мыши. Появится контекстное меню. В нем вы найдете все вышеперечисленные режимы.

Распечатать форму можно, находясь в любом режиме работы с ней. Внешний вид распечатанной формы всегда соответствует текущему режиму. Распечатка содержит столько "снимков" формы, сколько записей в соответствующей таблице. Для вывода на печать сделайте щелчок по главной кнопке MS Access 2010 — кнопке **Файл**. В открывшемся меню выберите пункт **Печать**. Чтобы увидеть, в каком виде форма будет на бумаге, перейдите в режим предварительного просмотра.

3.3. Создание простой формы в режиме конструктора

В этом разделе мы создадим форму в режиме конструктора, размещая в ней элементы различных типов. Перед вами самый распространенный вид формы (рис. 3.6), созданной в MS Access 2010.

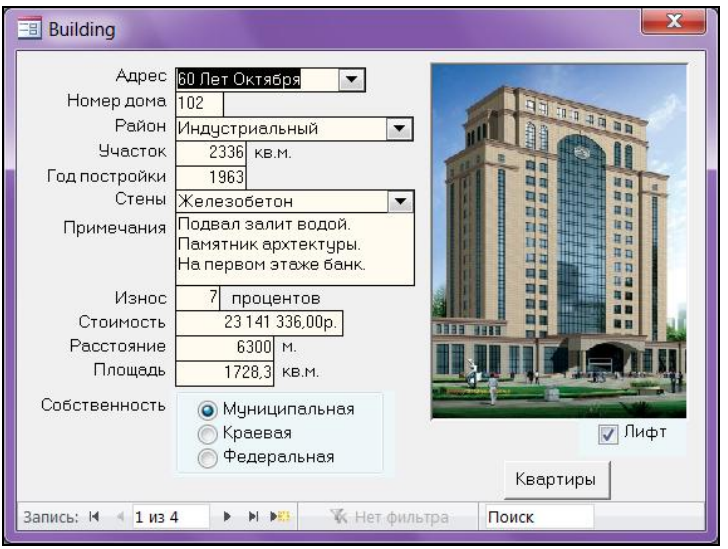


Рис. 3.6. Наша первая форма, созданная при помощи конструктора форм. Форма открыта в режиме формы

В данном случае она используется для отображения информации о зданиях и связана с одноименной таблицей `building`. В любой момент времени в этой форме отображается информация только по одному зданию. Линейка прокрутки, размещенная в нижней части формы, дает возможность отобразить в форме лю-

бую запись из таблицы. В центре линейки прокрутки находится номер записи. На рис. 3.6 — это запись номер 1.

Откроем эту форму в режиме конструктора форм. Для этого в области навигации (окно базы данных Real Estate 2010) выберите раздел **Формы**. Щелкните правой кнопкой мыши по форме `Building`, в появившемся контекстном меню выберите пункт **Конструктор** (рис. 3.7).

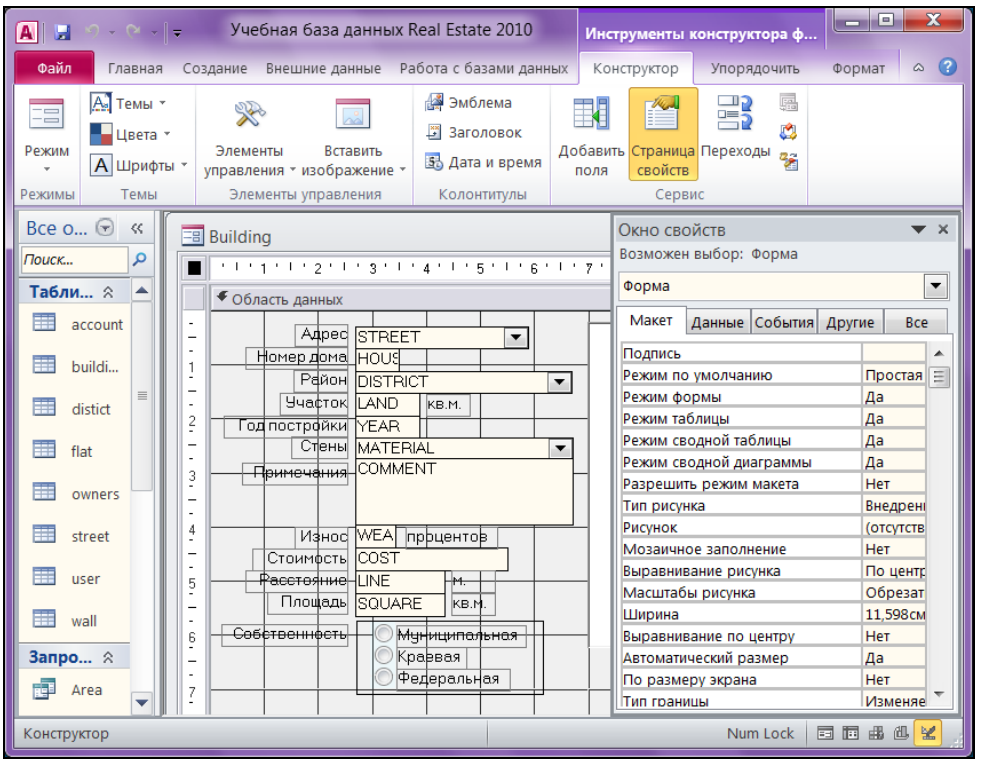


Рис. 3.7. Наша первая форма, созданная при помощи конструктора форм. Форма открыта в режиме конструктора

На экране появятся три дополнительные вкладки главной ленты MS Access 2010: **Конструктор**, **Упорядочить** и **Формат**, а также форма `Building` в режиме конструктора. На вкладке ленты в разделе **Элементы управления** расположена панель инструментов. Она предназначена для размещения в форме выбранных элементов и содержит их стилизованные изображения. Функции этих кнопок будут рассмотрены в следующих разделах.

В разделе ленты **Сервис** — пять элементов. Выберите элемент **Страница свойств**, и окно свойств появится на экране дисплея. Для этих же целей можно применить клавишу <F4>. Каждый объект в MS Access, включая непосредствен-

но базу данных, имеет свойства. Наша форма не исключение. Имеются различные категории свойств формы. В MS Access 2010 они представлены на пяти вкладках:

- **Макет** — свойства, которые принадлежат способу отображения объекта;
- **Данные** — свойства, которые принадлежат данным объекта, независимо от того, каким способом они получены;
- **События** — свойства, которые принадлежат событиям и связанным с ними процедурам;
- **Другие** — свойства, которые принадлежат характеристикам объекта или его признакам;
- **Все** — все категории и свойства объекта.

3.3.1. Подготовка к конструированию

Теперь, когда вы в общих чертах знаете о том, с чем придется иметь дело, предлагаю начать конструирование формы с самого начала, останавливаясь подробно на каждом создаваемом элементе.

1. Выберите вторую вкладку ленты MS Access 2010 — **Создание**.
2. Сделайте щелчок левой кнопкой мыши по значку **Конструктор форм**. Появится новая пустая форма и три дополнительные вкладки: **Конструктор**, **Упорядочить** и **Формат**. Активной будет вкладка **Конструктор**.
3. Выберите на этой вкладке значок **Страница свойств**. Появится **Окно свойств**. Активной в этом окне должна быть вторая вкладка **Данные**.
4. Перейдите на первую строчку **Источник записей** и сделайте щелчок по кнопке  .
5. Поле со списком раскроется. В нем будут перечислены все таблицы, входящие в текущую базу данных.
6. Выберите таблицу `building`. Из этой таблицы и будет отображать данные наша первая форма.
7. В окне свойств перейдите на вкладку **Макет**. Установите свойство **Кнопки размеров окна** в *Отсутствуют*. Если пользователь в процессе работы с формой раскроет ее окно на весь экран, то для доказательства вашего профессионального мастерства придется выполнить пересчет координат расположения элементов в форме, а это очень трудоемкое занятие.

На рис. 3.8 приведен вид окна MS Access 2010 целиком. Система готова к построению формы!

Каждому полю таблицы `building` в форме, которую мы позже назовем так же — `Building`, соответствует определенный элемент. Начнем с первого поля — `STREET`.

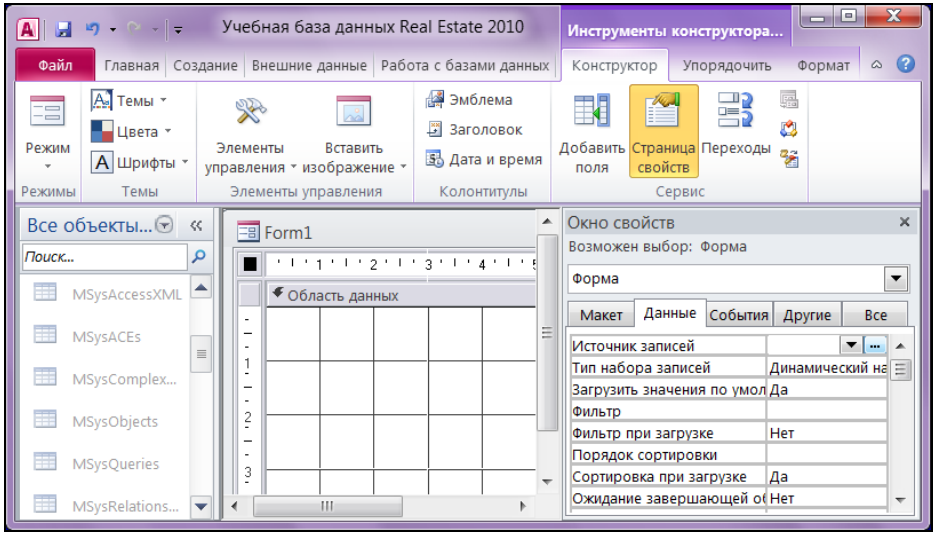


Рис. 3.8. Создание заготовки для построения формы

3.3.2. Изменение цвета формы

Цвет фона формы является основным параметром, определяющим ее внешний вид. Существует множество причин, в силу которых разработчики изменяют цвет формы. Некоторые меняют цвет, чтобы сделать ее просто более привлекательной. Другие — чтобы выделить группу объектов. Третьи — для условного форматирования данных в полях. При выборе цвета фона всегда нужно помнить о том, для каких целей предназначена создаваемая форма и как она будет использоваться. Не стоит забывать и о том, что программный комплекс будет находиться целый день перед глазами работника. Не следует отдавать предпочтение ярким цветам.

Для изменения цвета фона формы или объекта:

1. Выделите раздел формы, например **Область данных** или объект, цвет фона которого нужно изменить, щелкнув левой кнопкой мыши по заголовку раздела или по элементу формы.
2. В окне свойств появятся значения всех свойств этого объекта. Перейдите на вкладку **Макет**.
3. Щелкните левой кнопкой мыши по свойству **Цвет фона**. В правой части строчки свойства появится кнопка [...]. Нажмите ее. Откроется окно выбора цвета.
4. Если стандартных цветов этого окна недостаточно, сделайте щелчок мышью по строчке **Другие цвета**. Откроется окно **Цвета** с двумя вкладками: **Обыч-**

ные и **Спектр** (рис. 3.9). Перейдите на вторую вкладку и сделайте свой выбор.

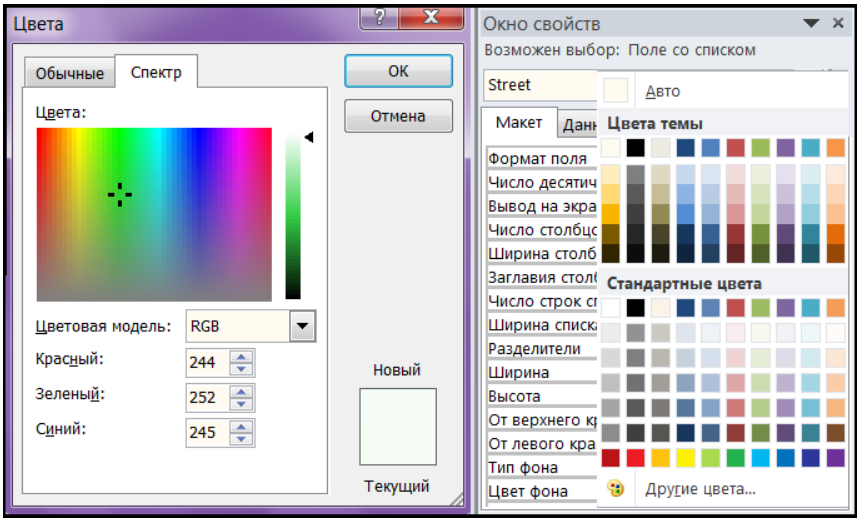


Рис. 3.9. Выбор цвета фона объекта формы

Предупреждение

Цвет фона раздела формы MS Access 2010 устанавливается независимо от цвета фона других разделов. Чтобы изменить цвет всех разделов, задайте новый цвет для каждого из них отдельно. Цвет фона формы не влияет на цвет фона элементов управления, размещенных в форме.

Совет

Проще и удобнее всего изменить цвет фона области данных формы можно, сделав щелчок правой кнопкой мыши по области данных (не попав по какому-либо элементу формы) и выбрав в появившемся контекстном меню пункт **Цвет заливки/фона**.

3.3.3. Изменение фонового рисунка формы

Если вас не устраивает однородный цвет фона формы, можно в качестве фона использовать рисунок. MS Office Access 2010 "понимает" рисунки, хранящиеся во всех известных на момент выхода в свет этой версии графических форматах.

Чтобы задать фоновый рисунок для формы:

1. В режиме конструктора выделите всю форму. Для этого сделайте щелчок левой кнопкой мыши по квадратику в левом верхнем углу конструктора форм  . В центре этого квадрата появится метка  .

2. Сделайте доступным окно свойств формы. Для этого на вкладке **Конструктор** главной ленты MS Access 2010 в разделе **Сервис** выберите значок **Страница свойств**.
3. Перейдите на первую вкладку **Макет**.
4. Выберите свойство **Рисунок**. Во второй колонке этого свойства увидите его значение: *Отсутствует*. Сделайте щелчок левой кнопкой мыши по этому свойству. Появится кнопка [...].
5. Активируйте ее щелчком мыши. Откроется окно **Выбор рисунка** (рис. 3.10).
6. Сделайте свой выбор и закройте окно щелчком мыши по кнопке **OK**.

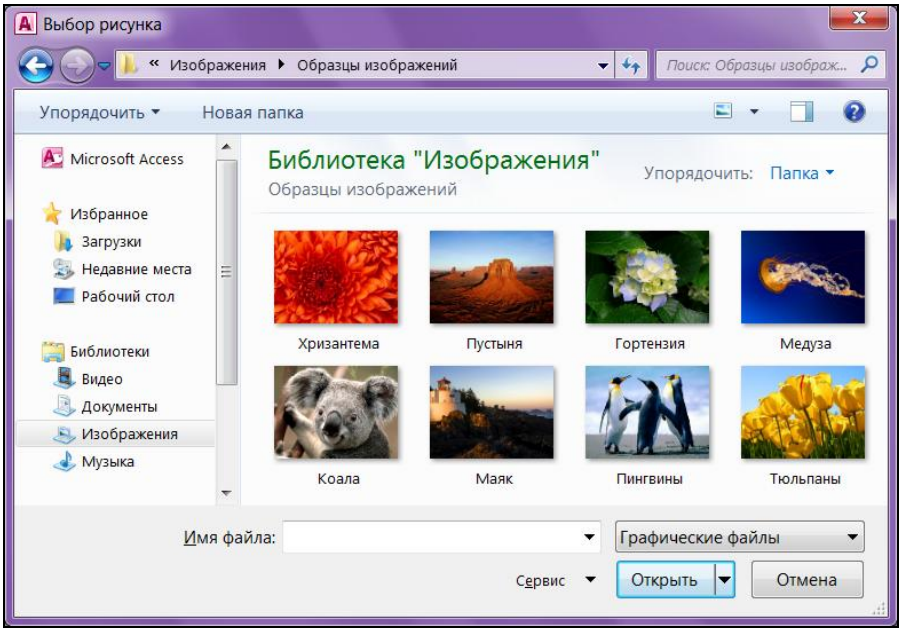


Рис. 3.10. Диалоговое окно **Выбор рисунка**

Предупреждение

Действие фонового рисунка распространяется, в отличие от цвета фона, на все разделы формы. Для его удаления просто удалите все символы из текстового поля значения свойства.

Используя свойство **Масштабы рисунка** можно установить его размеры. Для вывода рисунка в исходном виде в раскрывающемся списке этого свойства выберите значение *Обрезать*. При выборе значения *Увеличить* произойдет растяжение или сжатие рисунка с сохранением пропорций до максимально возможных размеров, при которых не происходит обрезка рисунка. При выборе значения

Растянуть рисунок будет сжат до размеров формы. Изменение пропорций рисунка в этом случае неизбежно.

Совет

Чтобы определить положение рисунка, воспользуйтесь свойством **Выравнивание рисунка**. Чтобы фоновый рисунок центрировался относительно формы, а не окна формы, установите его значение **По центру формы**, а не **По центру**.

Примечание

Используя свойство **Мозаичное заполнение**, можно добиться повторяющегося изображения рисунка по всей форме. Для этого установите значение этого свойства равным *Да*.

3.3.4. Панель инструментов *Элементы управления* вкладки *Конструктор*

Панель инструментов **Элементы управления** вкладки **Конструктор** в MS Access 2010 содержит кнопки, предназначенные для разработки форм (рис. 3.11). В табл. 3.1 описано назначение этих кнопок.

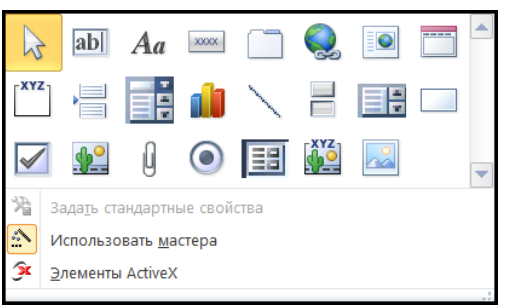


Рис. 3.11. Панель инструментов **Элементы управления**

Таблица 3.1. *Назначение кнопок панели инструментов* ***Элементы управления***

Кнопка	Описание
	Кнопка **Выбрать**. Выделение прямоугольных областей рукописных штрихов, фигур и текста
	Кнопка **Поле**. Используется для отображения, ввода и изменения данных в источнике записей формы или отчета для вывода результатов вычислений, а также для приема данных, вводимых пользователем
	Кнопка **Надпись**. Создает элемент управления, в котором в форме или отчете выводится поясняющий текст. Надписи могут содержать гиперссылки. MS Access автоматически присоединяет подписи к создаваемым элементам управления

Таблица 3.1 *(продолжение)*

Кнопка	Описание
	Кнопка **Кнопка**. Создает элемент управления для вызова другой формы, отчета, макроса, процедуры или функции VBA
	Кнопка **Вкладка**. Применяется для создания формы с несколькими вкладками. На вкладку можно добавлять другие элементы управления
	Кнопка **Гиперссылка**. Применяется для создания ссылки на Web-страницу, рисунок или адрес электронной почты
	Кнопка **Группа переключателей**. Используется для размещения набора флажков, переключателей или выключателей
	Кнопка **Вставить разрыв страницы**. Применяется для указания начала нового экрана в форме или в отчете
	Кнопка **Поле со списком**. Создает составной элемент управления, объединяющий поле и раскрывающийся список. Чтобы ввести значение в поле базовой таблицы, можно ввести значение в поле в элементе управления или выбрать значение в списке
	Кнопка **Диаграмма**. Отображает данные MS Access в форме или отчете в виде диаграммы
	Кнопка **Линия**. Используется в формах или отчетах для отделения особенно важных разделов формы или отчета
	Кнопка **Выключатель**. Создает отдельный элемент управления, присоединенный к логическому полю в базе данных MS Access 2010 или к столбцу типа `Bit` в проекте MS Access, работающему с MS SQL Server 2008
	Кнопка **Список**. Создает список, допускающий прокрутку. Если форма открыта в режиме формы, то выбранное в списке значение можно ввести в новую запись или использовать для изменения существующей записи
	Кнопка **Прямоугольник**. Используется для создания графических объектов для привлечения внимания к важным данным в форме или отчете
	Кнопка **Флажок**. Создает отдельный элемент управления, присоединенный к логическому полу в базе данных. Флажок в пользовательском окне или входящий в группу параметров является свободным элементом управления
	Кнопка **Свободная рамка объекта**. Используется для отображения в форме или отчете свободного объекта OLE. Этот объект остается неизменным при переходе от записи к записи
	Кнопка **Вложение**. Применяется для связи с полем таблицы типа **Вложение**
	Кнопка **Переключатель** (или радиокнопка). Создает отдельный элемент управления, присоединенный к логическому полю в базе данных MS Access 2010 или к столбцу типа `Bit` в проекте MS Access, работающему с MS SQL Server 2008
	Кнопка **Подчиненная форма/отчет**. Предназначена для вывода в форме или отчете данных из нескольких таблиц

Таблица 3.1 *(окончание)*

Кнопка	Описание
	Кнопка **Присоединенная рамка объекта**. Предназначена для отображения в форме или отчете объектов OLE (набор рисунков). При переходе от записи к записи в форме или отчете выводятся разные объекты
	Кнопка **Рисунок**. Используется для отображения неизменяемого рисунка в форме или отчете. Рисунок не является объектом OLE. После размещения рисунка в форме его изменения не допускаются

В формах MS Access 2010 применяются три типа элементов управления.

- *Присоединенные элементы управления*, связанные с полем источника данных для формы. Это может быть поле таблицы, запрос и даже значение другого элемента управления текущей или любой другой формы. Присоединенные к таблице элементы отображают и позволяют изменить значение поля, с которым они связаны. Элементы, присоединенные к другим элементам, не могут изменить значения "донора". Самыми распространенными присоединенными элементами являются текстовые поля. Выключатели, переключатели и флажки связывают с логическим полем таблицы. Элемент OLE — с графическим объектом, видео- и звуковым файлом и т. д. Все присоединенные элементы при "рождении" получают связанные с ними метки. Значение метки представляет собой значение свойства **Подпись**, относящегося к вкладке **Макет**. Метку всегда можно удалить.
- *Свободные элементы управления* не зависят от источника данных формы. Свободные текстовые поля используются для ввода данных, например, для получения значения, которое будет использоваться в выражении. Прямоугольники и линии — для оформления внешнего вида, а OLE — для добавления графики в форму или отчет. Не все свободные элементы имеют метки.
- *Вычисляемые элементы управления* используют в качестве источника данных в выражении. В выражениях могут использоваться как поля таблиц, так и свободные элементы.

3.3.5. Панель инструментов вкладки *Упорядочить*

Панель инструментов **Упорядочить** содержит кнопки и раскрывающиеся меню, которые облегчают выбор параметров форматирования. На рис. 3.12 приведена только часть этой панели — **Размер и порядок**. Почти все пиктограммы и пункты этой панели изначально недоступны и визуально погашены. В нужный момент времени система MS Access обеспечивает к ним доступ. Рис. 3.12 сделан в тот момент, когда в форме была выделена группа элементов. Доступны все пункты всех меню за исключением одного — **Разгруппировать.**

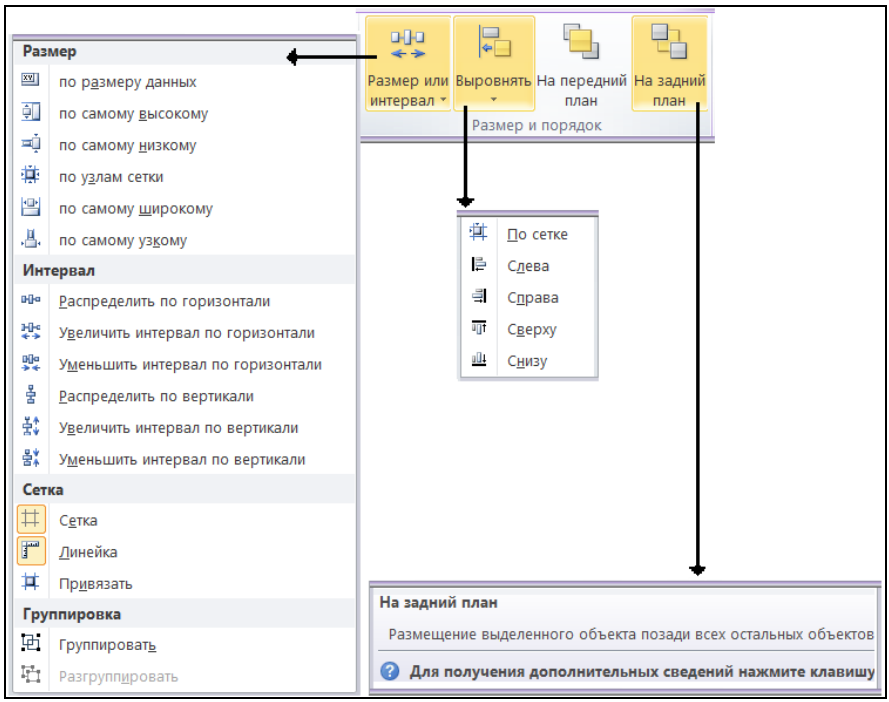


Рис. 3.12. Панель инструментов вкладки **Упорядочить**

3.3.6. Создание поля со списком

Улицу, на которой расположено здание, пользователь должен выбрать из списка улиц города, а не вносить с клавиатуры ее название. В MS Access грамотно реализован выбор значения из очень длинных списков. Вы можете ввести один или несколько символов названия элемента списка (в данном случае улицы) — MS Access сам найдет в списке нужный элемент по первым символам и занесет его в текстовое поле. Поэкспериментируйте с формой `Building` из файла Real Estate Часть I.accdb. Уверен, что вам очень понравится такая реализация выбора. А сейчас — непосредственно построение.

1. Убедитесь, что в разделе **Элементы управления** вкладки **Конструктор** кнопка  **Использовать мастера** нажата. Если нет — выделите ее щелчком левой кнопки мыши. В противном случае вы очень долго будете обвинять во всех грехах свой безупречно работающий компьютер. Построитель этого элемента так и не запустится.

2. Нажмите на панели элементов кнопку  **Поле со списком**. Поместите указатель мыши над активной областью формы. Он превратится в значок поля со списком  , снабженный крестиком в левом верхнем углу. Прицельтесь по-

лучше и сделайте щелчок левой кнопкой мыши. Увидите следующую картину (рис. 3.13).

3. Одновременно с этим откроется первое диалоговое окно мастера списков **Создание полей со списком**. Вам будет предложено выбрать источник значений (рис. 3.14). Наши названия улиц хранятся в таблице `street`, поэтому сделайте щелчок левой кнопкой мыши по первой кнопке переключателя **Объект "поле со списком"...**.

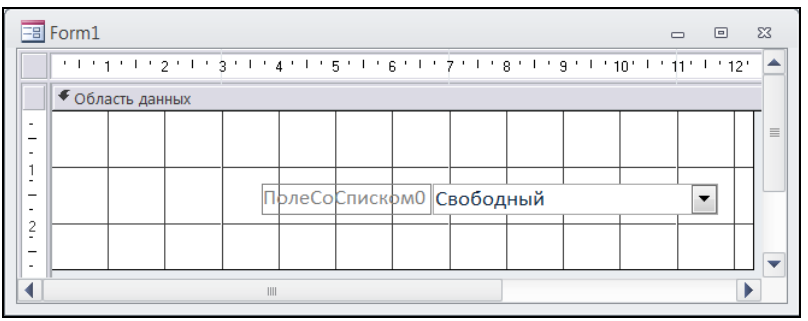


Рис. 3.13. В области данных формы появился первый элемент

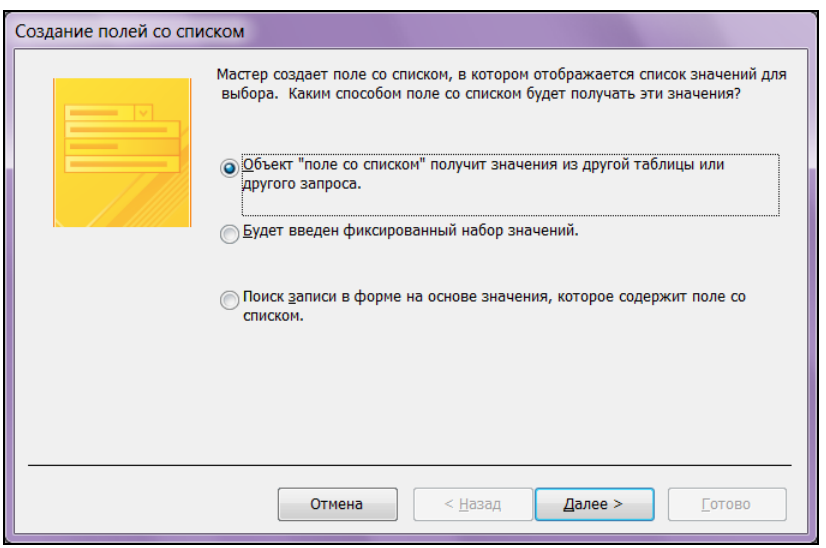


Рис. 3.14. Первое диалоговое окно мастера построения поля со списком

4. Нажмите кнопку **Далее**. Появится второе диалоговое окно, в котором отображен список всех таблиц, входящих в базу данных. Выберите таблицу `street` и нажмите кнопку **Далее**.

5. В появившемся третьем диалоговом окне MS Access предлагает выбрать поля таблицы `street`, значения которых будут отображаться в раскрывающемся поле со списком. Пусть это будут поля `NAME` и `SIGN`.
6. Четвертое диалоговое окно (рис. 3.15) предлагает определить порядок отображения улиц в поле со списком. Если строк в таблице более десятка, то сортировка — обязательный момент. Отсортировать записи можно максимум по четырем полям. В нашем случае достаточно двух. Это название улицы и признак. Для текстовых полей порядок сортировки — по алфавиту.

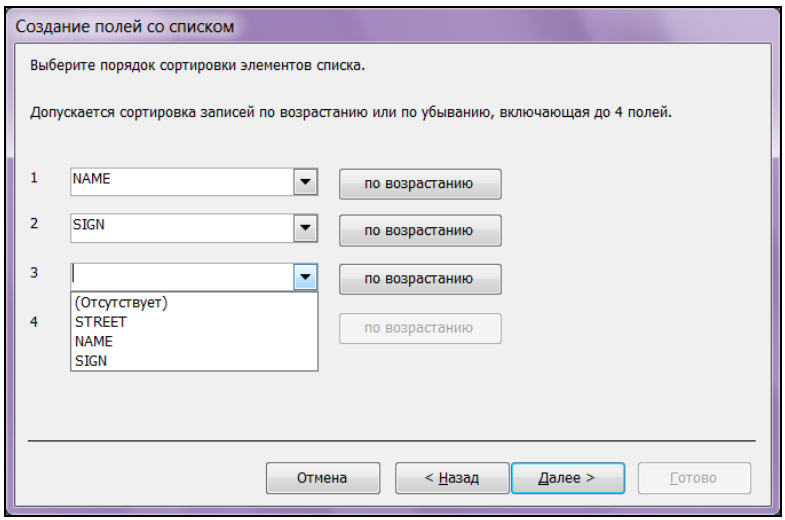


Рис. 3.15. Четвертое диалоговое окно мастера построения поля со списком

7. В пятом диалоговом окне поставьте флажок **Скрыть ключевой столбец (рекомендуется)**. Пользователю ни к чему видеть, под каким номером в таблице `street` стоит та или иная улица. Здесь же вам предоставлена возможность установить ширину колонок (рис. 3.16).
8. Нажмите кнопку **Далее**. Очень ответственный момент. Надо указать поле таблицы `building`, в которое будет записываться ссылка на улицу из таблицы `street`.
9. В последнем диалоговом окне предлагается ввести надпись, которая будет стоять рядом с раскрывающимся списком. Напишите слово "Адрес" и нажмите кнопку **Готово**.

Предупреждение

Если при создании заготовки для формы *(см. разд. 3.3.1)* вы пропустили пункты 4—6 и не назначили таблицу, с которой должна работать форма, то: на первом шаге работы мастера (см. рис. 3.14) вы не найдете третий переключатель **Поиск записи в форме на основе значения, которое содержит поле со списком**. К тому же,

мастер построения поля со списком пропустит пятый шаг. Будет создан бесполезный "свободный" элемент. В этом случае создайте окружение формы и начните построение поля со списком с самого начала.

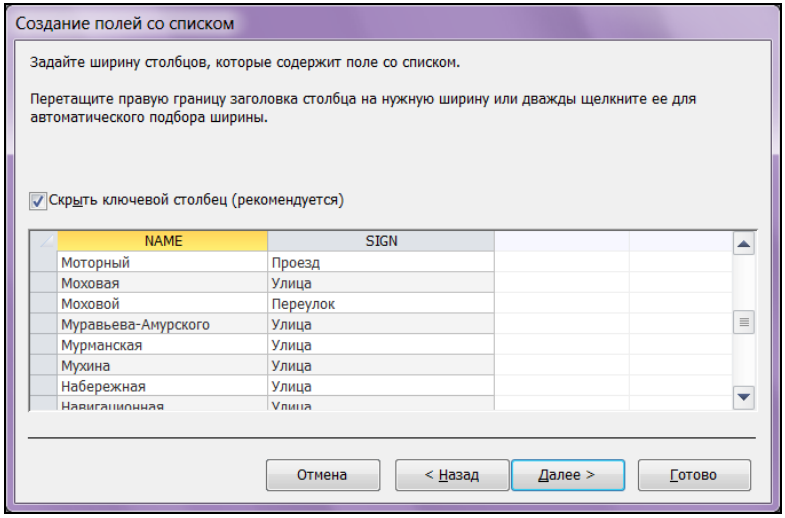


Рис. 3.16. Выбор ширины колонок раскрывающегося списка

Не беда, если вам не удалось сразу расположить элемент в нужном месте формы. MS Access 2010 предоставляет пользователю массу возможностей для форматирования. Вот только основные из них.

- Во-первых, щелкните правой кнопкой мыши по надписи "Адрес" и выберите в раскрывшемся меню пункт **Размер**, а в появившемся подменю — пункт **По размеру данных**.
- Во-вторых, переместите поле со списком и его метку в нужное место на форме. Для этого поместите указатель мыши в любую точку на границе выделенного элемента, отличную от маркеров изменения размеров. Нажмите левую кнопку мыши и, не отпуская ее, перетащите элемент на новое место.
- В-третьих, более точно "выставить" элемент управления и его метку на форме можно при помощи клавиш-стрелок при нажатой клавише <Ctrl>.

3.3.7. Создание текстовых полей

Создадим текстовое поле для отображения номера дома. Это однострочное текстовое поле, присоединенное к полю `HOUSE` таблицы `building`. Порядок действий следующий.

1. Если на экране отсутствует окно списка полей таблицы `building`, выберите на вкладке **Конструктор** главной ленты MS Access 2010 в разделе **Сервис** значок **Добавить поля**.

2. В списке полей выделите поле HOUSE. Нажмите левую кнопку мыши и перетащите выделенный элемент на форму. В активной области формы указатель мыши превратится в символ текстового поля с маленьким крестиком, как это было при создании поля со списком, и стрелкой. Стрелка показывает на левый верхний угол самого поля, а не его метки. Если подведете курсор слишком близко к левому краю формы, то можете потерять метку текстового поля. На экране ее не будет видно, хотя среди элементов формы она будет присутствовать.
3. Перетащите текстовое поле за маркер перемещения к метке и уменьшите ширину текстового поля.

У вас есть возможность изменить размеры как элемента, так и метки при помощи клавиатуры. Выделите нужный объект и клавишами-стрелками при нажатой клавише <Shift> добейтесь необходимых результатов. При помощи клавиатуры такие действия выполняются гораздо точнее, чем при использовании мыши.

В качестве метки MS Access 2010 применяет название перетаскиваемого поля таблицы. Название метки можно изменить, причем несколькими способами. Проще всего щелкнуть по выделенной метке левой кнопкой мыши. Указатель мыши превратится в текстовый курсор. Далее используйте клавиатуру для ввода надписи.

Настала очередь создания поля **Примечания** (рис. 3.17). Это поле ввода с полосой прокрутки применяется для включения в форму длинных текстовых полей.

1. Если на экране отсутствует окно списка полей таблицы building, выберите на вкладке **Конструктор** главной ленты MS Access 2010 значок **Добавить поля**.

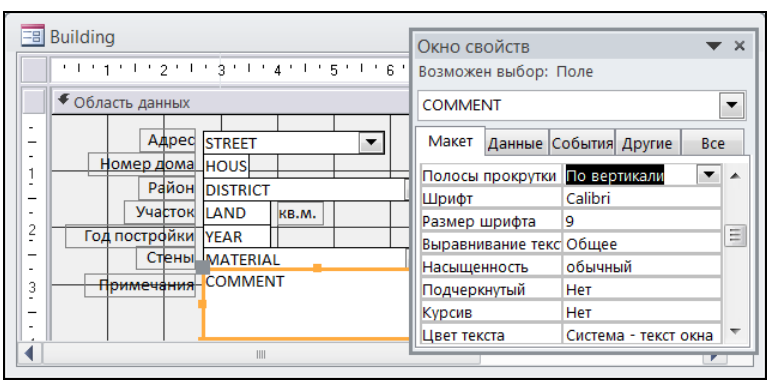


Рис. 3.17. Создание поля ввода с линейкой прокрутки

2. В списке полей выделите поле COMMENT. Нажмите левую кнопку мыши и перетащите выделенный элемент на форму. В активной области формы указатель мыши превратится в символ текстового поля. Отпустите левую кнопку мыши.

3. Удалите метку поля и измените размеры текстового поля до требуемых.
4. Сделайте щелчок правой кнопкой мыши по созданному полю. Выберите в открывшемся меню пункт **Свойства**.
5. В появившемся окне свойств для элемента "COMMENT" выберите вкладку **Макет**. Найдите в списке этой вкладки свойство **Полосы прокрутки**. Раскройте список свойств и выберите значение *По вертикали*.

Предупреждение

Вертикальная полоса прокрутки такого поля ввода в режиме формы отображается только тогда, когда поле получает фокус.

3.3.8. Создание поля типа *Флажок*

Если в таблице находится поле логического типа, принимающее значение только *да* или *нет*, то использование флажка сделает форму более выразительной и удобной. В нашем случае речь пойдет о поле `ELEVATOR` (лифт). Выделите в списке полей таблицы это поле и перетащите на форму. Остается изменить название метки. Щелкните по ней левой кнопкой мыши. Указатель превратится в текстовый курсор. Используйте клавиатуру для ввода надписи.

Этого же результата можно добиться и другим способом. Сделайте щелчок правой кнопкой мыши по созданной метке. Выберите в открывшемся меню пункт **Свойства**.

В появившемся диалоговом окне выберите вкладку **Макет**. Найдите в списке этой вкладки свойство **Подпись**. Введите текст, который хотите увидеть рядом с флажком, — *Лифт*.

3.3.9. Создание поля типа *Группа переключателей*

В таблице `building` (здания) имеется поле `KIND` — вид собственности. Собственность может быть только четырех типов: муниципальная, краевая, федеральная и частная. Тип этого поля — числовой. В качестве значений могут выступать только числа от 1 до 4. Конечно, можно было создать еще одну таблицу, назначить ключевое поле, установить отношение "один-ко-многим" с таблицей `building` по этому ключу, а для отображения в форме использовать поле со списком. К счастью, MS Access 2010 предлагает нам более легкий, а с позиции интерфейса и более красивый путь решения задачи такого типа, когда значений поля немного и их количество в процессе эксплуатации программного комплекса меняться не будет. Порядок создания группы переключателей следующий.

1. Убедитесь, что кнопка  с подсказкой "Использовать мастера" нажата. Если нет — "выделите" ее щелчком левой кнопки мыши. Нам понадобится работа построителя.

2. На этой же панели выберите пиктограмму **Группа переключателей**. Типичная ошибка начинающих — выбор пиктограммы **Переключатель**. Помните! Набором переключателей в MS Access группу переключателей сделать нельзя!
3. Поместите указатель мыши над активной областью формы. Он превратится в значок группы переключателей с крестиком в левом верхнем углу. Нажмите левую кнопку мыши и, удерживая ее в нажатом состоянии, переместите курсор по диагонали так, чтобы получилась рамка требуемого размера. Отпустите левую кнопку мыши. Автоматически запустится построитель группы переключателей.
4. Сделайте подписи у переключателей (рис. 3.18) и нажмите кнопку **Далее**. Теперь назначим переключатель, используемый по умолчанию. Большинство зданий города находится в муниципальной собственности. Предоставим возможность пользователю не щелкать мышью по этому элементу при занесении муниципального здания. Он наверняка будет нам за это благодарен (рис. 3.19).

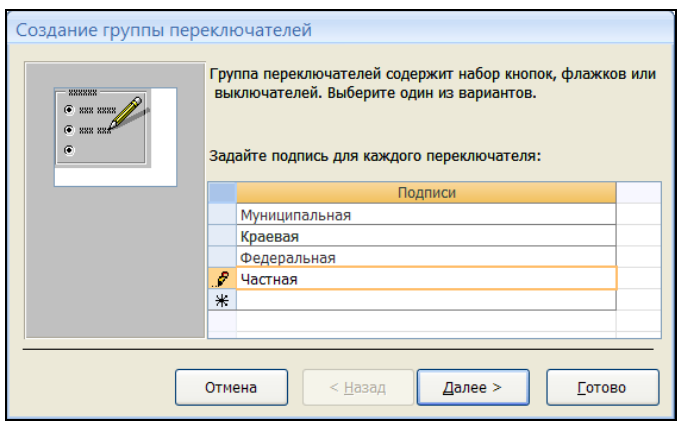


Рис. 3.18. Создание подписей у группы переключателей

5. Третий шаг построителя посвящен привязке подписи к значению (рис. 3.20). Если ничего не менять, то цифра 1 будет соответствовать муниципальной собственности, 2 — краевой, 3 — федеральной, а 4 — частной.
6. На четвертом шаге (рис. 3.21) необходимо указать поле таблицы `building`, в которое будет занесена выбранная цифра. Сделайте щелчок мышью по второй кнопке и не ошибитесь с выбором. Это поле `KIND` (вид собственности).
7. Пятый шаг построителя (рис. 3.22) — оформление внешнего вида группы переключателей. Здесь нам предоставляется возможность увидеть, что будет

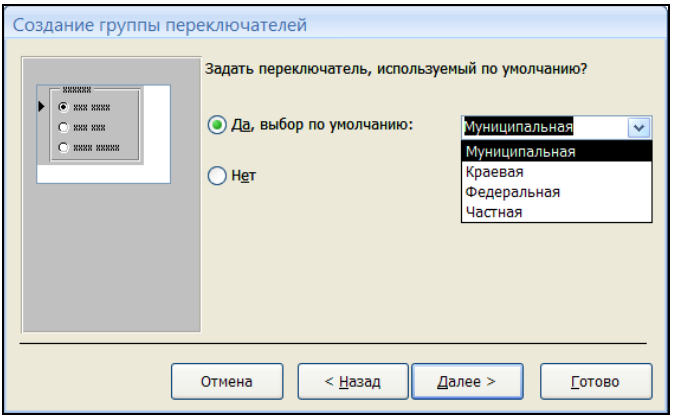


Рис. 3.19. Выбор значения по умолчанию

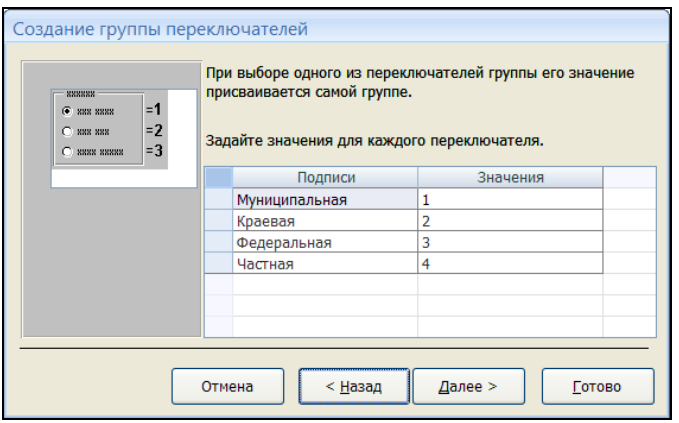


Рис. 3.20. Задание значений для каждого переключателя

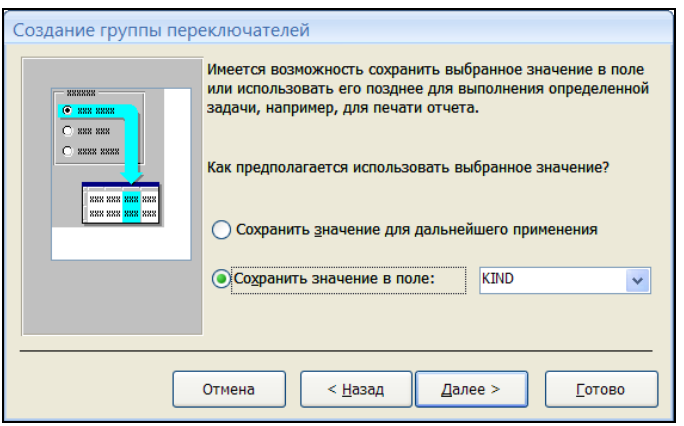


Рис. 3.21. Четвертый шаг работы построителя группы переключателей

в форме, если мы сделаем тот или иной выбор. Смелее выбирайте тип оформления и смотрите на левую часть окна (образец).

8. На последнем шаге необходимо ввести подпись для созданной группы переключателей. Если подпись в таком виде не нужна, то введите пустое значение, а в форму в нужном месте поместите элемент **Надпись** с заголовком *Собственность*.

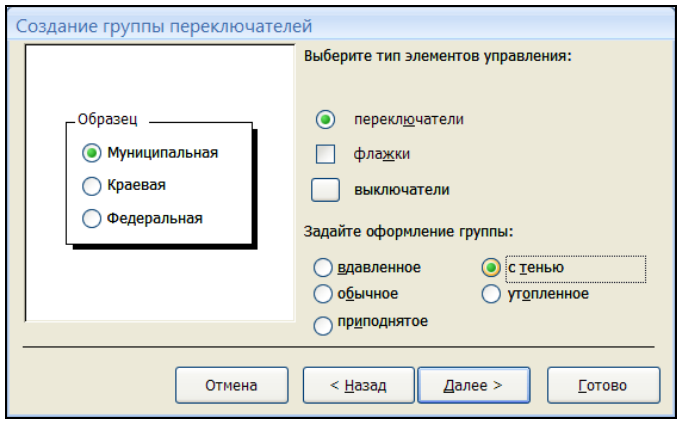


Рис. 3.22. Оформление внешнего вида группы переключателей

3.3.10. Отображение фотографий в форме

Все современные настольные СУБД дают нам возможность отображать фотографии, содержащиеся в таблицах баз данных. В Microsoft Access как фотографии, так и другие объекты OLE отображаются в присоединенной рамке объекта. Этот элемент управления представляет собой контейнер OLE, в котором могут отображаться растровые и векторные изображения, звуковые объекты, рисунки с анимацией, видеообъекты и т. п.

Добавим присоединенную рамку объекта в нашу первую форму (см. рис. 3.6). Напомню, что для хранения фотографий зданий мы предусмотрели поле `PICTURE` (поле OLE) в таблице `building`.

1. Выберите на панели элементов пиктограмму  **Присоединенная рамка объекта**. Не перепутайте ее с очень похожей на нее свободной рамкой объекта  .

2. Поместите указатель мыши над активной областью формы. Он превратится в значок присоединенной рамки объекта  с крестиком в левом верхнем углу. Нажмите левую кнопку мыши и, удерживая ее в нажатом состоянии, перемес-

тите курсор по диагонали так, чтобы получилась рамка требуемого размера. Отпустите левую кнопку мыши.

3. Удалите подпись возле этого элемента управления, которую сгенерировал компьютер: "Присоединенный OLE" (рис. 3.23). Нет никакого смысла переименовывать ее во что-то типа "Фотография" или нечто подобное. И без нее все предельно ясно.

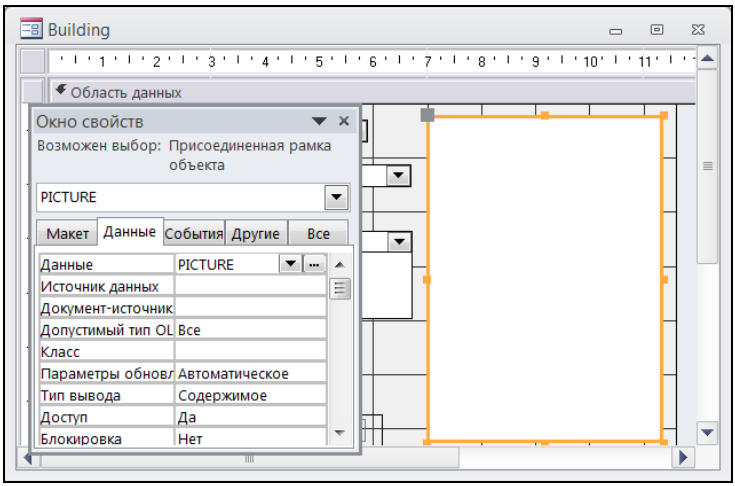


Рис. 3.23. Размещение фотографии здания в форме `Building`

4. Сделайте щелчок правой кнопкой мыши по только что созданному объекту и в появившемся меню выберите пункт **Свойства**. Настала очередь привязать к объекту поле таблицы. Выберите вторую вкладку **Данные**.

5. Последний этап — масштабирование. В MS Access существуют три способа масштабирования графических объектов внутри присоединенной рамки объекта. Для этого предназначено свойство **Установка размеров** (рис. 3.24). Его легко найти на первой вкладке **Макет** окна свойств.

Если в качестве значения этого свойства выбрать *Фрагмент*, то фотография будет отображена в ее исходной пропорции. Если фото не помещается в рамку целиком, то оно урезается снизу и справа. Второй способ — *Вписать в рамку*. Фотография будет "втиснута" в очерченное ранее пространство. Масштаб по высоте и ширине фотографии будет установлен отдельно, чтобы заполнить рамку полностью. Третий способ — *По размеру рамки*. Масштаб фотографии будет увеличен или уменьшен по ширине и по высоте так, чтобы фото целиком поместилось в рамку, и при этом его исходная пропорция сохранилась бы.

Наша первая форма создана. Позже мы разместим в ней еще один элемент — кнопку **Квартиры** для вызова другой формы (`Flats`), а сейчас попробуем доба-

вить запись в таблицу `building`, соответствующую еще одному зданию, с помощью нашей формы.

1. Сделайте активной **Область навигации** базы данных Real Estate 2010.
2. В разделе **Формы** найдите форму `Building`. Запустите ее на выполнение, сделав двойной щелчок мышью по названию.
3. Появится информация о первом здании (см. рис. 3.6). Найдите на линейке записей пиктограмму **Новая (пустая запись)**. Теперь можно заносить информацию об очередном здании. Займемся его фотографией.

а *б* *в*

Рис. 3.24. Варианты свойства **Установка размеров**: *а — Фрагмент*; *б — Вписать в рамку*; *в — По размеру рамки*

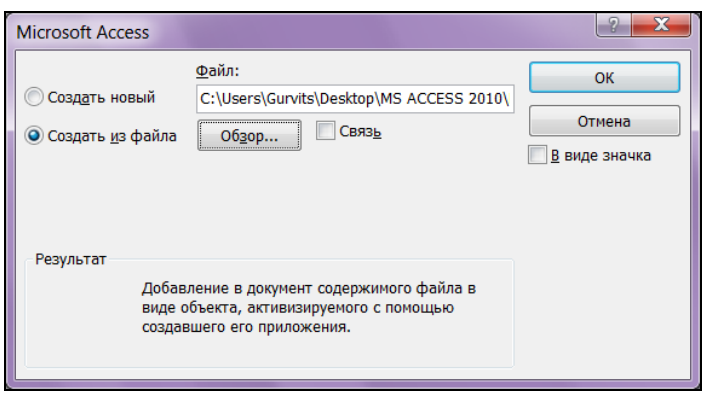


Рис. 3.25. Занесение фотографии здания

4. Сделайте щелчок правой кнопкой мыши по месту, где она должна располагаться в форме.
5. В появившемся меню выберите пункт **Вставить объект**. Появится диалоговое окно **Microsoft Access** для вставки объекта (рис. 3.25).
6. Выберите опцию **Создать из файла**, нажмите кнопку **Обзор** и с помощью появившегося диалогового окна операционной системы вашего компьютера **Обзор** укажите файл, в котором хранится фото здания. Не устанавливайте флажок **Связь**. Мы не будем редактировать эту фотографию с помощью какого-либо графического редактора. Связь таблицы с файлом и автоматическое обновление изображения нас не интересуют.

3.3.11. Перемещение элементов формы и изменение размеров

Для придания привлекательного вида форме вам непременно потребуется изменять расположение и размеры элементов, размещенных в ней.

При выделении элемента управления вокруг него должна появиться рамка, имеющая маркер перемещения (левый верхний угол) и семь маркеров изменения размеров. Текстовые поля, переключатели и флажки имеют связанные с ними метки. Метки выделяются одновременно с элементами управления. Для того чтобы выделить элемент, сделайте по нему щелчок левой кнопкой мыши. Правила работы с отдельным элементом приведены далее.

- Чтобы *переместить элемент управления вместе с меткой*, поместите указатель мыши в любую точку на его границе. Указатель должен превратиться в элемент перемещения ↔. Нажмите левую кнопку мыши и, удерживая ее, переместите элемент на новое место. Как только элемент окажется в требуемом месте, отпустите левую кнопку мыши.
- Для *перемещения элемента управления отдельно от метки* поместите указатель мыши над маркером перемещения. Указатель должен превратиться в элемент перемещения. Нажмите левую кнопку мыши и, удерживая ее, переместите элемент на новое место. Как только элемент окажется в требуемом месте, отпустите левую кнопку мыши.
- Для того чтобы *изменить одновременно ширину и высоту элемента управления*, установите указатель мыши над одним из трех угловых маркеров изменения размеров. Указатель должен превратиться в наклонную двунаправленную стрелку ↗. Нажмите левую кнопку мыши и, удерживая ее, измените размеры элемента так, как требуется.
- Чтобы *изменить только высоту или ширину элемента управления*, установите указатель мыши над одним из четырех боковых маркеров изменения размеров. Указатель должен превратиться в двухстороннюю стрелку. Нажмите ле-

вую кнопку мыши и, удерживая ее, измените размеры элемента до требуемого уровня.

Выделение нескольких элементов управления, расположенных в форме, дает возможность разработчику задать для них общие свойства, а также перемещать и изменять их размеры одновременно.

А теперь рассмотрим правила выделения группы элементов.

- Охватите при помощи мыши группу элементов прямоугольником. Она окажется выделенной, а указатель мыши, размещенный над ней, превратится в элемент перемещения ↔. В группу попадут все элементы управления, попавшие хотя бы частично в область выделения.
- Сделайте щелчок левой кнопкой мыши по одному из элементов, которые надо объединить в группу. Нажмите и удерживайте клавишу <Shift>. Последовательно щелкните по каждому элементу будущей группы.

Примечание

При выделении элемента управления, имеющего метку, она выделяется вместе с ним.

После выделения группы элементов на вкладке главной ленты MS Access 2010 **Упорядочить** (см. рис. 3.12) станет доступным ряд пунктов: **Размер**, **Интервал**, **Сетка** и т. д.

3.3.12. Задание последовательности перехода для элементов формы

Основное назначение формы — ввод информации. При занесении данных переход от одного элемента к другому осуществляется при помощи мыши или клавиатуры. Щелчок левой кнопкой мыши сразу делает активным нужный элемент. При помощи клавиатуры также можно передать управление элементу, расположенному в форме, но только в порядке "общей очереди". Для этого используются клавиши <Enter> и в меньшей степени — <Tab>, хотя именно она для этого и создана.

Порядок перехода определяется при разработке формы в режиме конструктора. В очередь элементы выстраиваются по мере появления в форме. Почти всегда очередность требуется изменить.

Для изменения последовательности перехода между элементами формы сделайте следующее:

1. Откройте форму в режиме конструктора. Для этого в области навигации (окно базы данных) выберите раздел **Формы**. Щелкните правой кнопкой мыши по имени формы. В появившемся контекстном меню выберите пункт **Конструктор**.

2. На экране появятся три дополнительные вкладки главной ленты MS Access 2010: **Конструктор**, **Упорядочить** и **Формат**. Перейдите на вкладку **Конструктор**.
3. Сделайте щелчок левой кнопкой мыши по значку **Переходы**. Появится окно с названием **Последовательность перехода** (рис. 3.26).
4. Для изменения очередности обхода элементов выберите из них нужный и сделайте щелчок по кнопке, находящейся слева от названия. Кнопку обязательно отпустите. Элемент будет выделен черным цветом.
5. Перетащите его в нужное место последовательности.

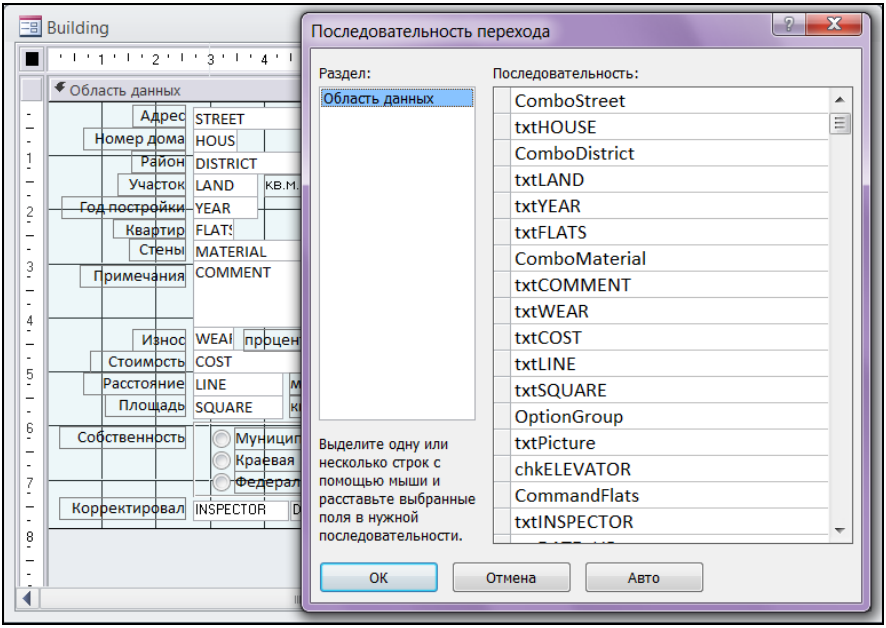


Рис. 3.26. Диалоговое окно **Последовательность перехода**

Примечание

Если во время выделения элемента не отпустить кнопку мыши, а сразу начать перетаскивание, то система выделит соседний элемент, за ним — следующий и т. д. Это сделано для того, чтобы выделить группу элементов. Сигнал к началу перемещения (одного элемента или группы) — повторное нажатие кнопки мыши.

3.4. Создание сложной формы

Наша первая форма `Building` дает возможность работать с таблицей зданий. С ее помощью мы можем комфортно добавлять, удалять и корректировать информацию, относящуюся к какому-либо зданию. Настала пора добиться такой же ком-

фортности при работе с квартирами, расположенными в здании, и с проживающими в них собственниками. В принципе мы могли бы создать еще три формы для раздельного отображения информации по квартирам, лицевым счетам и по проживающим в этих квартирах собственникам, но ввиду небольшого количества полей, содержащихся в таблицах `flat`, `owners` и `account`, есть смысл поместить всю эту информацию в одной сложной форме. Дадим ей имя `Flats` (квартиры).

3.4.1. Создание запроса

Первая проблема, которую необходимо решить на пути создания сложной формы, заключается в следующем. В нашей форме должна отображаться информация не обо всех квартирах в городе, а только о тех, которые расположены в выбранном для просмотра здании. Поэтому форма `Flats` будет связана не с таблицей `flat`, а с запросом `Flats`, в который мы поместим данные о квартирах, находящихся в одном конкретном здании, и лицевых счетах ответственных квартиросъемщиков. Порядок его создания следующий.

1. Выберите вторую вкладку главной ленты MS Access 2010 — **Создание**.
2. Сделайте щелчок левой кнопкой мыши по значку **Конструктор запросов**, который находится в разделе **Макросы и код**. Появится окно конструктора запросов и окно с названием **Добавление таблицы**.
3. Первая вкладка этого окна содержит список таблиц текущей базы данных. Выберите в нем сначала таблицу `flat` и нажмите кнопку **Добавить**, а затем таблицу `account`. Закройте окно **Добавление таблицы**. Появится дополнительная вкладка главной ленты MS Access — **Конструктор**.
4. Расположите последовательно все поля из таблиц `flat` и `account` в запросе, как это показано на рис. 3.27 (для этого "зацепите" по очереди каждое поле и перетащите их в нижнюю часть окна запроса или просто выберите нужные поля из раскрывающегося списка). Не пропустите ни одного, иначе вас ждут неприятности при занесении новых записей по квартирам. Поле `ACCOUNT` (номер лицевого счета), связывающее эти таблицы, заносится только один раз.
5. Определите условия отбора записей из таблиц `flat` и `account` в запрос (рис. 3.28). Он должен содержать данные только по одному конкретному зданию.
6. Установите порядок сортировки записей, попавших в запрос. Лучше всего, если квартиры в нашей форме будут отображаться в порядке возрастания номеров. Это поле `FLAT`.
7. Сохраните созданный запрос под именем `Flats`.

Условия отбора записей по полям `STREET` и `HOUSE`:

```
[Forms]![Building]![Street]
[Forms]![Building]![House]
```

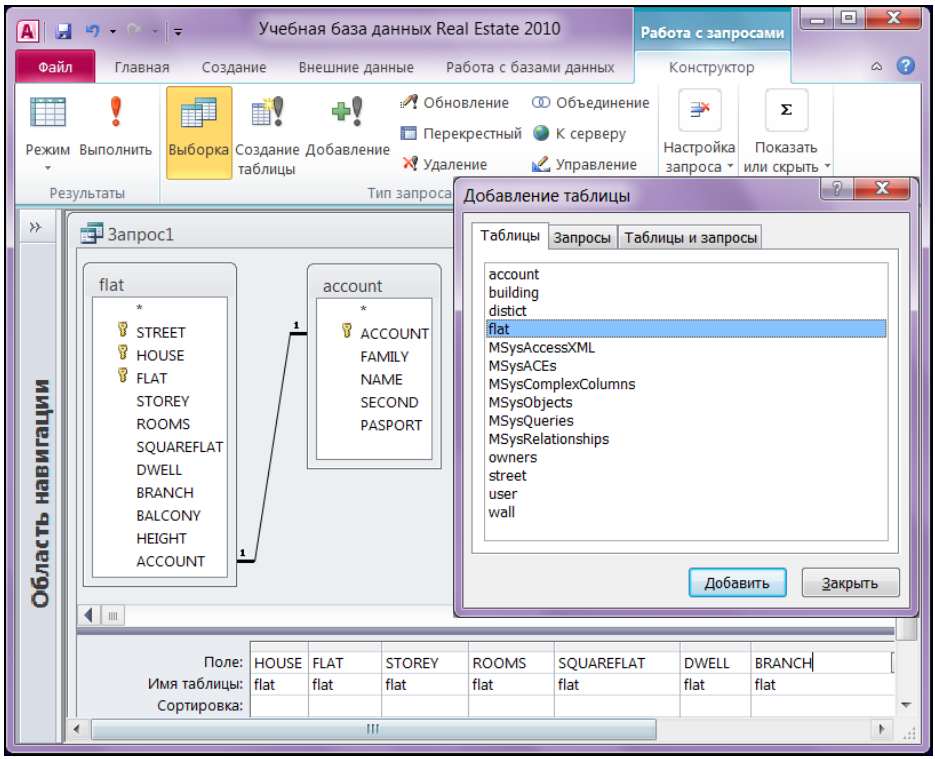


Рис. 3.27. Конструктор запросов

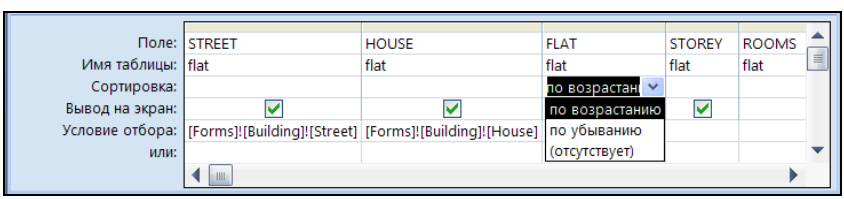


Рис. 3.28. Определение условий отбора записей из исходных таблиц и сортировка

можно перевести так. В запрос должны попасть только те квартиры, которые находятся в здании с номером улицы, отображенном в форме `Building` в объекте `Street` (адрес) и с номером дома, отображенном в форме `Building` в объекте `House` (номер дома). Для полной ясности посмотрите на рис. 3.28.

Сделайте щелчок правой кнопкой мыши в свободном месте окна конструктора запросов. Появится контекстное меню. Первым пунктом в нем должен быть пункт **Режим SQL**. Щелкните по нему правой кнопкой мыши. Появится текст запроса на языке SQL, сгенерированный конструктором запросов (листинг 3.1).

Листинг 3.1. Текст запроса `Flats` на языке SQL

```
SELECT flat.STREET, flat.HOUSE, flat.FLAT,
       flat.STOREY, flat.ROOMS, flat.SQUAREFLAT,
       flat.DWELL, flat.BRANCH, flat.BALCONY,
       flat.HEIGHT, account.ACCOUNT, account.FAMILY,
       account.NAME, account.SECOND
FROM account INNER JOIN flat ON account.ACCOUNT = flat.ACCOUNT
WHERE (((flat.STREET)=[Forms]![Building]![Street])
       AND  ((flat.HOUSE)=[Forms]![Building]![House]))
ORDER BY flat.FLAT;
```

Обратите внимание на сложность текста запроса и на то, с какой легкостью мы сформировали его с помощью конструктора запросов, не зная ни одной конструкции и ни одного служебного слова языка SQL. Начинающему пользователю MS Access вовсе необязательно знать этот язык, но мы к нему обязательно вернемся в *части II*.

3.4.2. Знакомство с событиями

Вторая проблема, которая может привести в замешательство начинающего разработчика, возникнет при занесении данных по новой квартире. Дело в том, что в нашей второй форме `Flats` не отображаются данные адреса, и если компьютеру не указать адрес квартиры, то в таблицу `flat` будет добавлена запись с техническими характеристиками квартиры и с пустым адресом. Понятно, что для конечного пользователя такая квартира будет навсегда потеряна, т. к. не будет отображаться в форме при повторном запуске на выполнение. Чтобы избежать этой ошибки, необходимо добавить в событие **До вставки** формы `Flats` процедуру, написанную на языке Visual Basic для приложений (Visual Basic for Applications, VBA). Текст этой процедуры приведен в листинге 3.2. Если вы хотите написать хорошее приложение на MS Access, то знание основ VBA вам просто необходимо. Этому посвящена *глава 5*.

Листинг 3.2. Текст процедуры обработки события *До вставки* формы `Flats`

```
Private Sub Form_BeforeInsert(Cancel As Integer)
        Me!STREET = [Forms]![Building]![STREET]
        Me!HOUSE = [Forms]![Building]![HOUSE]
End Sub
```

Первую и последнюю строчку этой процедуры MS Access сгенерирует сам. Вторая и третья строчки — на совести разработчика. Для ссылки на активную форму используется ключевое слово `Me` (см. рис. 3.29). После имени формы добавляют название элемента, разделив их восклицательным знаком.

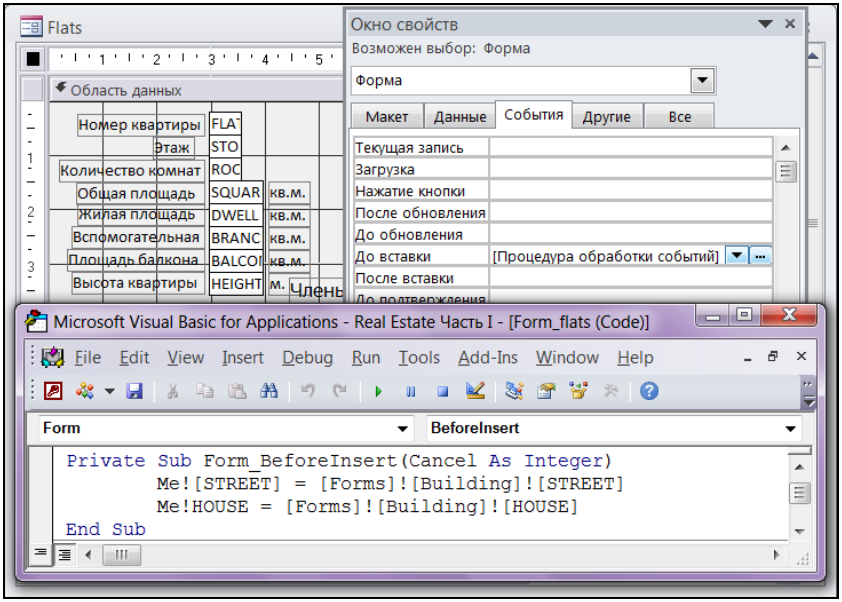


Рис. 3.29. Форма `Flats` (квартиры) в конструкторе

Теперь квартира, вновь заносимая в базу, получит адрес (`Street`+`House`) и будет появляться в форме в нужном месте и в нужное время.

3.4.3. Создание подчиненной формы

Подчиненная форма — это форма, находящаяся внутри другой формы. Первичная форма (в нашем случае `Flats`) называется главной. Подчиненные формы очень удобны для вывода информации из таблиц или запросов, связанных отношением "один-ко-многим". Одна квартира — несколько проживающих. При использовании формы с подчиненной формой для ввода новых записей, текущая запись в главной форме сохраняется при входе в подчиненную форму. Это гарантирует, что записи из таблицы или запроса на стороне "многие" будут иметь связанную запись в таблице или запросе на стороне "один". MS Access автоматически сохраняет каждую запись, добавляемую в подчиненную форму, и никакие специальные приемы типа обработки события **До вставки** не требуются.

Сначала посмотрим на конечный результат. На рис. 3.30 показаны две формы: `Building` (информация по зданию) и `Flats` (данные по квартире). Форма `Flats` содержит подчиненную форму `Owners` (члены семьи квартиросъемщика).

Порядок создания подчиненной формы следующий.

1. Откройте первичную форму `Flats` в режиме конструктора.
2. Убедитесь, что на вкладке **Конструктор** главной ленты в разделе **Элементы управления** кнопка  с подсказкой "Использовать мастера" нажата. Если

нет — выделите ее щелчком левой кнопки мыши. Нам понадобится работа построителя подчиненных форм.

3. Нажмите на панели элементов кнопку **Подчиненная форма**. Наведите указатель мыши на то место первичной формы, где вы планируете поместить левый верхний угол подчиненной формы. Указатель мыши превратится в значок подчиненной формы с крестиком в левом верхнем углу.

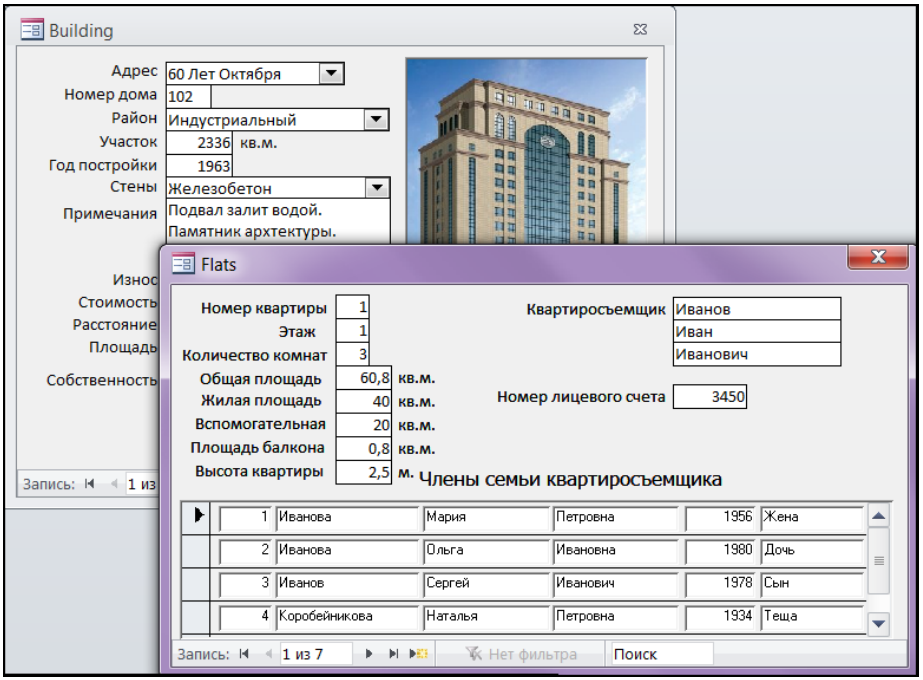


Рис. 3.30. Основные формы программного комплекса Real Estate 2010

4. Нажмите левую кнопку мыши и, удерживая ее в нажатом состоянии, переместите курсор по диагонали так, чтобы получилась рамка требуемого размера. Отпустите левую кнопку мыши. Автоматически запустится построитель подчиненной формы.

5. Первый шаг работы мастера подчиненных форм — определение данных, которые надо включить в подчиненную форму. Установите переключатель **Имеющиеся таблицы и запросы** и щелкните по кнопке **Далее**. Появится окно для выбора таблиц и полей. Выберите все поля из таблицы `owners` (рис. 3.31).

6. Второй шаг — определение полей связи между главной и подчиненной формами. Основа главной формы `Flats` — одноименный запрос, в который попали данные из двух таблиц: `flat` и `account`. Подчиненная форма основана на

данных из таблицы `owners`. Никогда ранее мы не устанавливали отношений между запросом `Flats` и таблицей `owners`, поэтому все, что может предложить мастер подчиненных форм на этом шаге, нам не подходит. Сделайте щелчок по кнопке **Самостоятельное определение**. Ключевая связка выглядит так: `STREET+HOUSE+FLAT` (рис. 3.32).

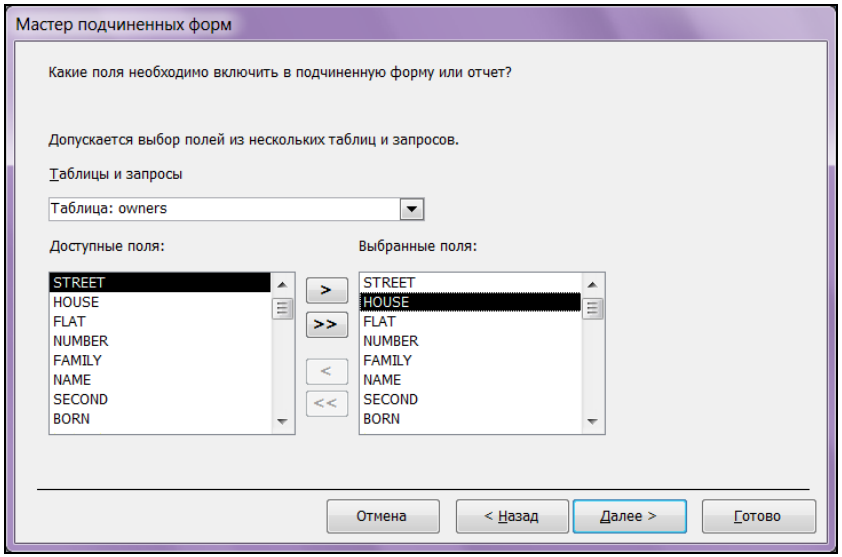


Рис. 3.31. Первый шаг работы мастера подчиненных форм

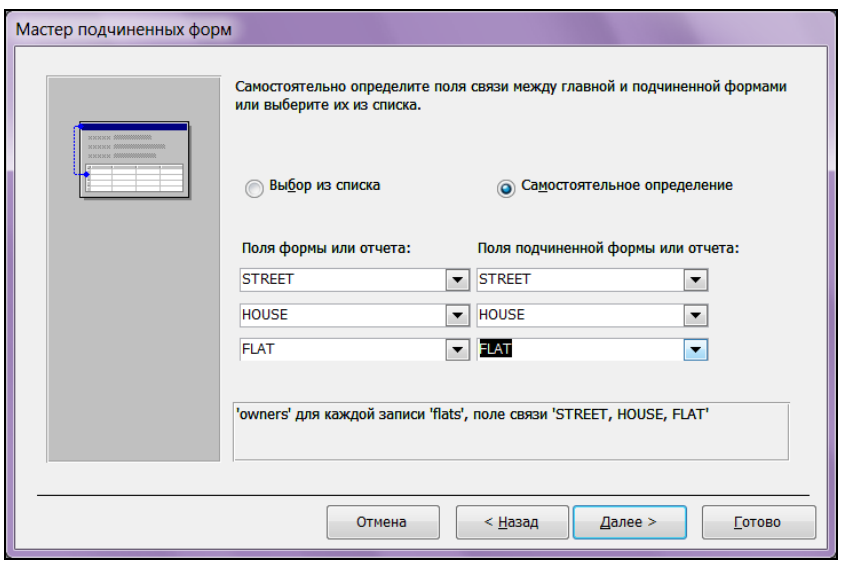


Рис. 3.32. Определение полей связи между главной и подчиненной формами

7. Третий шаг — выбор имени для подчиненной формы. Под этим именем она появится в списке форм базы данных Real Estate 2010. Мы уже условились, что назовем ее `Owners`. Введите это имя и щелкните по кнопке **Готово**.

Закройте первичную форму `Flats`. Дальнейшая работа будет происходить с формой `Owners`. Откройте ее в режиме конструктора. То, что сгенерировал мастер подчиненных форм, необходимо подвергнуть некоторому улучшению. Посмотрите сами (рис. 3.33).

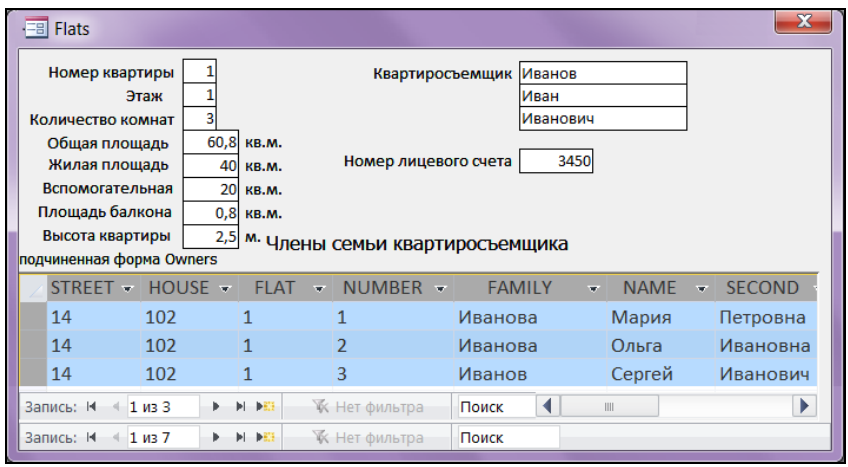


Рис. 3.33. Подчиненная форма, созданная при помощи мастера

В заголовки колонок вынесены названия полей таблицы `owners`. Конечному пользователю они ни о чем не говорят. Наличие ключевых полей `STREET`, `HOUSE` и `FLAT` в подчиненной форме вносит настоящую путаницу. Кнопки перехода по записям подчиненной формы в данном случае просто мешают работе. Не пытайтесь изменить шрифт заголовков колонок. Форма сгенерирована в режиме таблицы, у вас просто ничего не получится. Выполним доработку формы.

1. Для начала сделаем подчиненную форму ленточной. В окне свойств этой формы выберем вкладку **Макет**, найдем свойство **Режим по умолчанию** и изменим его значение: *Ленточные формы*.

2. Удалим заголовок формы и ключевые поля `STREET`, `HOUSE` и `FLAT`. Для удаления элемента сделайте по нему щелчок левой кнопкой мыши и нажмите клавишу <Delete>.

3. Уберем совсем кнопки перехода по записям. Для этого на вкладке **Макет** в качестве значения свойства **Кнопки навигации** поставьте *Нет*.

4. Немного отформатируем объекты формы. Для этого поместите указатель мыши в любую точку на границе выделенного элемента, отличную от маркеров изменения размеров. Нажмите левую кнопку мыши и, не отпуская ее, перета-

щите элемент на новое место. Более точно выставить элемент управления на форме можно при помощи клавиш-стрелок при нажатой клавише <Ctrl>, а поточнее изменить размеры элемента можно при помощи клавиш-стрелок при нажатой клавише <Shift>.

После проделанных манипуляций получим форму, изображенную на рис. 3.30.

3.4.4. Добавление кнопки в форму для вызова другой формы

В предыдущем разделе нами была создана сложная форма `Flats` для отображения информации о квартирах, расположенных в здании. Если вы полностью следовали моим советам и выполнили эту работу на компьютере, то вместо картинки, изображенной на рис. 3.30, после запуска формы на выполнение получили требование ввести какой-то параметр (рис. 3.34).

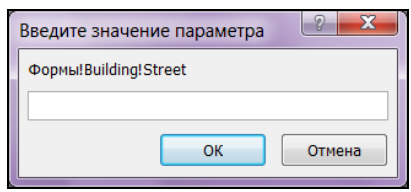


Рис. 3.34. Требование параметра

В этом нет ничего удивительного. Запускать на выполнение форму `Flats` (квартиры) имеет смысл только из формы `Building` (здание), т. к. два параметра адреса — улицу и номер дома — сформированный нами запрос по квартирам берет именно из этой формы. Добавим кнопку с названием **Квартиры** в форму `Building`.

1. Откройте форму `Building` в режиме конструктора.
2. Убедитесь, что на панели элементов кнопка  с подсказкой "Использовать мастера" нажата. Если нет — выделите ее щелчком левой кнопки мыши.
3. Сделайте щелчок левой кнопкой мыши по пиктограмме на панели элементов  **Кнопка**.
4. Наведите указатель мыши на то место формы `Building`, где вы планируете поместить левый верхний угол кнопки. Указатель мыши превратится в значок кнопки с крестиком в левом верхнем углу.
5. Нажмите левую кнопку мыши и, удерживая ее в нажатом состоянии, переместите курсор по диагонали так, чтобы получилась рамка требуемого размера. Отпустите левую кнопку мыши. Автоматически запустится построитель кнопки.

6. Появится диалоговое окно, показанное на рис. 3.35. Выберите категорию и действия так, как показано на этом рисунке, и нажмите кнопку **Далее**.

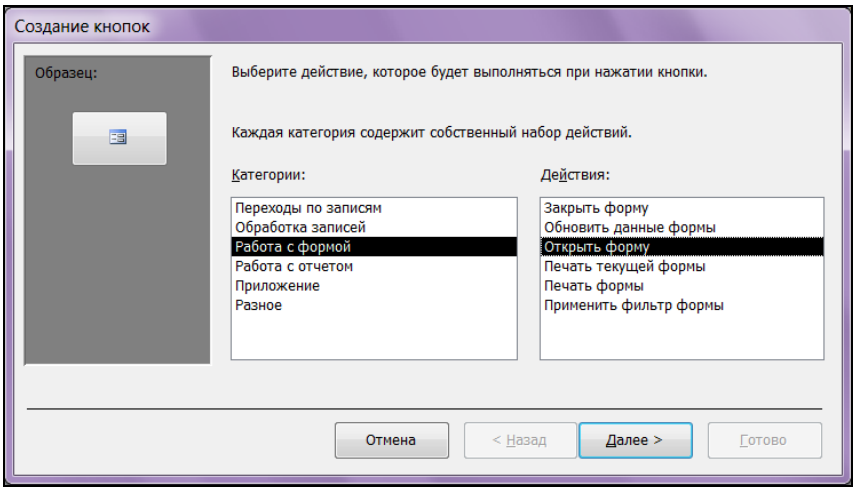


Рис. 3.35. Первый шаг работы мастера кнопок

7. На втором шаге работы мастера кнопок необходимо выбрать форму, которая будет запущена на выполнение после щелчка мышью по кнопке. Это форма — `Flats` (рис. 3.36).

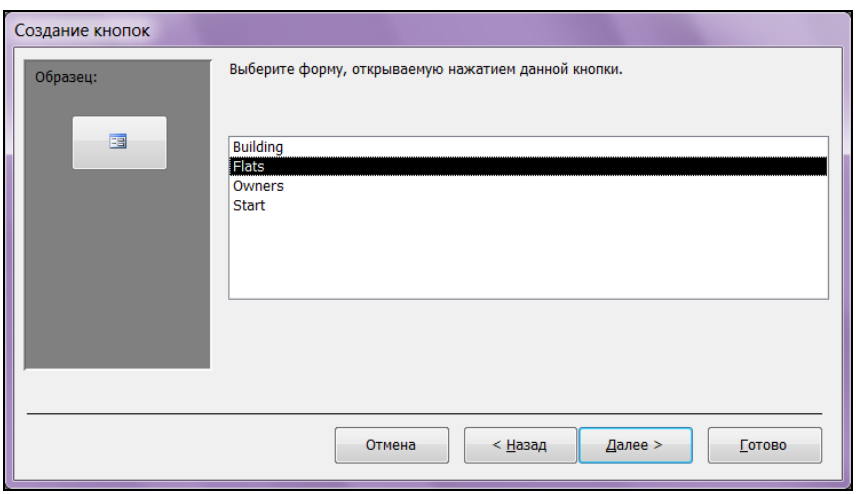


Рис. 3.36. Второй шаг работы мастера кнопок

8. Третий шаг — отбор сведений для отображения в форме. Так как запускаемая форма `Flats` отображает данные, полученные в результате выполнения

запроса по одному конкретному зданию, и никакого дополнительного отбора не требует — оставьте появившийся переключатель в исходном положении: **Открыть форму и показать все записи**.

9. Четвертый шаг — оформление внешнего вида кнопки. На ней можно разместить как надпись, так и рисунок, который находится в файле. При размещении рисунка Access предложит стандартное окно для поиска файла с рисунком.
10. На последнем шаге нам предлагается задать имя кнопки как объекта для дальнейшей ссылки на нее. Введите `ButtonFlats` или что-нибудь более информативное. Сделайте щелчок левой кнопки мыши по кнопке **Готово**. Если последний шаг пропустить, то компьютер сам присвоит созданной кнопке имя, под которым она и будет фигурировать как объект VBA в текстах, сгенерированных MS Access или написанных разработчиком.

Настало время посмотреть на то, что наколдовал Wizard. Так называется любой построитель (мастер) в оригинальной версии MS Access. Дословный перевод этого слова — колдун, волшебник. Запустите форму `Building` в режиме конструктора и сделайте двойной щелчок по созданной кнопке **Квартиры**. Откроется окно свойств этого объекта. Перейдите на вкладку **События** и выберите **Нажатие кнопки**. Результаты работы "колдуна" представлены в листинге 3.3.

Листинг 3.3. Код процедуры нажатия кнопки

```
Private Sub ButtonFlats_Click()
On Error GoTo Err_ ButtonFlats_Click
    Dim stDocName As String
    Dim stLinkCriteria As String
    stDocName = "flats"
    DoCmd.OpenForm stDocName, , , stLinkCriteria
Exit_ ButtonFlats_Click:
    Exit Sub
Err_ ButtonFlats_Click:
    MsgBox Err.Description
    Resume Exit_ ButtonFlats_Click
End Sub
```

Код процедуры, которая будет запущена на выполнение после щелчка по кнопке, можно несколько упростить (листинг 3.4).

Листинг 3.4. Код процедуры нажатия кнопки после доработки

```
Private Sub ButtonFlats_Click()
    DoCmd.OpenForm "Flats"
End Sub
```

Единственное отличие в работе этих процедур заключается в том, что в случае возникновения ошибки при работе формы `Flats` об этом в первой процедуре сообщит MS Access 2010, а во второй — Visual Basic for Applications 7.0. В обоих случаях будет выведен один и тот же текст, причем VBA укажет еще и номер ошибки.

Итак, кнопка для вызова формы `Flats` из формы `Building` создана. Основная часть нашего программного комплекса готова к работе. Давайте в очередной раз подумаем о конечном пользователе. Попробуйте полностью от начала до конца заполнить карточку здания. Уверен, вы будете не очень довольны тем, что после заполнения очередного поля и нажатия клавиши <Enter> курсор перепрыгивает вовсе не на следующее поле, а совсем в другой угол формы, и вам приходится прибегать к помощи мыши, чтобы вернуть его в нужное место.

Исправим сложившееся положение. Откройте форму `Building` в режиме конструктора. Сделайте щелчок левой кнопкой мыши по значку  **Переходы** вкладки **Конструктор**. Появится диалоговое окно **Последовательность перехода**, в котором отображен список всех объектов, имеющихся в форме. Причем отображены они в той последовательности, в какой попали в форму (см. рис. 3.26). С помощью полосы прокрутки найдите кнопку с именем `ButtonFlats`. У нас есть возможность перетащить любой объект в любое место окна. Выделите его щелчком левой кнопки мыши и творите!

Вполне очевидно, что наилучший порядок перехода фокуса между элементами формы — естественный. Если все элементы формы расположены один под другим, то нажмите кнопку **Авто**, и статус-кво будет восстановлен. После внесения изменений не закрывайте окно щелчком левой кнопкой мыши по крестику, расположенному в правом верхнем углу окна, т. к. в этом случае вся ваша работа пойдет насмарку. Для сохранения сделанных изменений обязательно нажмите кнопку **OK**.

3.5. Первые результаты

Мы создали две формы, которые позволяют добавлять и корректировать данные по описаниям зданий, квартир и проживающих в них, но не предусмотрели возможность удаления и поиска информации. Поспешу вас обрадовать — эти функции реализованы в Microsoft Access его создателями. Ни одна другая СУБД не имеет их в своем арсенале в таком виде. Если эта реализация устраивает разработчика программного комплекса, то больше ничего делать не требуется. Пользуйтесь готовым!

3.5.1. Удаление записи, отображаемой в форме

Для удаления записи из таблицы, которая связана с формой:

1. Сделайте эту запись активной. Для этого откройте нужную форму в режиме формы и при помощи линейки навигации по записям найдите требуемую.

2. На главной ленте MS Access 2010 выберите вкладку **Главная**, а в ней в разделе **Записи** — пункт **Удалить**. Откроется дополнительное меню (рис. 3.37).
3. В появившемся меню выберите второй пункт **Удалить запись**.
4. Появится запрос на подтверждение удаляемой записи. Нажмите кнопку **Да**.

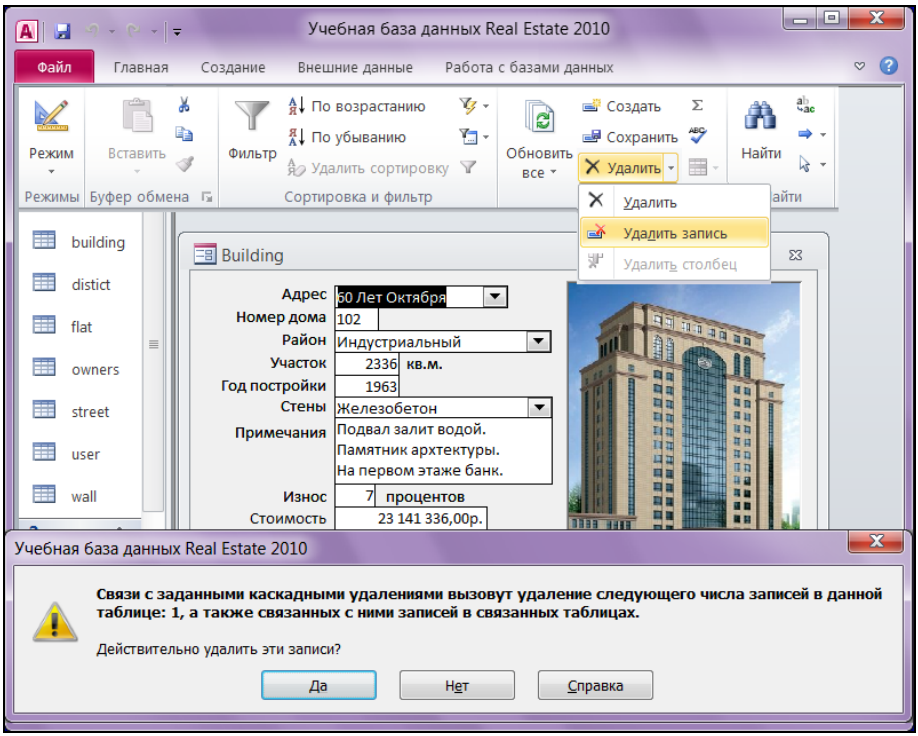


Рис. 3.37. Удаление записи, отображенной в форме

Предупреждение

Если при формировании связей между таблицами вы указали режим **Каскадное удаление связанных записей**, то вместе со зданием будут удалены все квартиры и проживающие в них, если нет — то MS Access выдаст сообщение о невозможности удаления этого здания, но только при наличии в нем квартир и проживающих.

3.5.2. Поиск в MS Access 2010

Возможности поиска, предоставляемые Microsoft Access, необычайно широки. Рассмотрим лишь некоторые из них.

Задача: требуется найти все здания, расположенные не в центральном районе, 1963 года постройки с износом до 30%. Площадь земельного участка должна

быть до 2500 м^2. Здания должны находиться в краевой собственности и иметь лифт. Наличие фотографии обязательно.

Решение: запустите на выполнение форму `Building`. На ленте MS Access 2010 выберите вкладку **Главная**, а на ней в разделе **Сортировка и фильтр** — значок **Параметры расширенного фильтра**. Появится контекстное меню. Вторым в нем должен быть пункт **Изменить фильтр**. Выберите его. Появится окно **Building: фильтр** (рис. 3.38). В этом окне мы можем задать условие, которому должно отвечать значение любого поля, находящегося в форме.

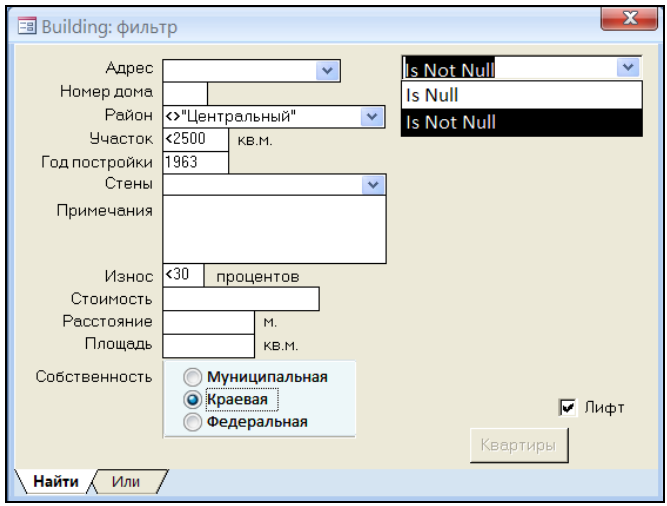


Рис. 3.38. Окно установки фильтра по зданиям

Обратите внимание на профессионализм авторов MS Access 2010.

- В список, из которого вы выбираете значение поля, занесены только те значения, которые встречаются в базе данных. Например, если в базе нет зданий 1909 года постройки, то среди предложенных вариантов выбора этот год отсутствует.
- Переключатель, который при занесении здания позволял выбрать лишь один вид собственности, сейчас доступен полностью.
- Наличие фотографии — значение **Is Not Null**. Что может быть оригинальнее?
- Для формирования очень сложных условий внизу окна имеется вкладка **Или**. Сделайте по ней щелчок левой кнопкой мыши, если в этом есть необходимость, и задайте еще группу условий. Таких групп условий может быть сколько угодно.

После ввода всех нужных условий снова выберите значок **Параметры расширенного фильтра**, а в открывшемся меню на этот раз — пункт **Применить**

фильтр. Снова появится форма `Building`, но отображаться в ней будут только два здания, удовлетворяющие условиям нашего примера, и надпись: "С фильтром".

Не забудьте снять фильтр, когда поставленная задача будет решена. Для этого предназначен тот же пункт меню, только теперь он носит название **Удалить фильтр**.

Предупреждение

Устанавливать фильтр можно только при числе записей в фильтруемых таблицах до двух-трех тысяч из-за значительного замедления работы компьютера, особенно в сети.

3.5.3. Проверка орфографии

И еще об одной приятной неожиданности, доставленной нам разработчиками Microsoft Access, — проверке орфографии, причем на довольно высоком уровне. Этим также не могут похвастаться другие СУБД. На рис. 3.6 в поле **Примечания** занесен текст, содержащий слово с орфографической ошибкой. Интересно, а вы ее заметили? Поручим компьютеру найти эту ошибку.

1. Поместите указатель мыши в поле **Примечания**.
2. На ленте MS Access 2010 выберите вкладку **Главная**. В разделе **Записи** найдите значок  **Орфография**. Сделайте по нему щелчок левой кнопкой мыши.

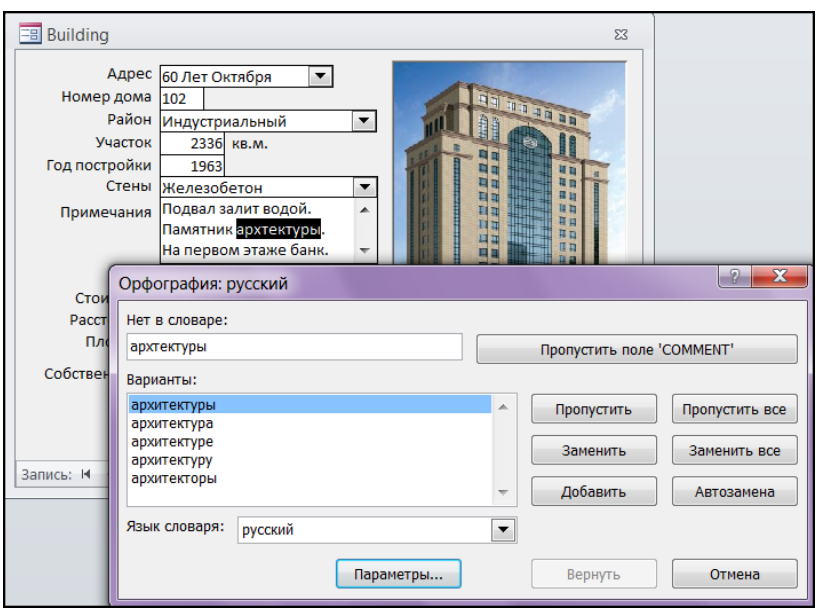


Рис. 3.39. Орфографическая ошибка при заполнении поля примечания

3. Появится диалоговое окно (рис. 3.39). Дальше — смотрите сами. Видно, кто виноват и что делать.

Примечание

Если конкретное поле не указано, то проверка орфографии будет проведена по всем полям формы.

3.6. Создание стартовой формы

Для запуска приложения Real Estate 2010 служит форма-заставка (рис. 3.40). Ее имя — `Start`. Форма `Building` активизируется после щелчка по кнопке ее запуска с изображением здания. Код события **Нажатие кнопки** с изображением здания приведен в листинге 3.5. Листинг 3.6 содержит код процедуры выхода из программного комплекса. Окно MS Access 2010 остается после этого открытым.

Рис. 3.40. Стартовая форма приложения Real Estate 2010

Листинг 3.5. Код процедуры запуска формы `Building`

```
Private Sub Кнопка2_Click()
' Кнопка Далее
On Error GoTo Err_Кнопка2_Click
    Dim stDocName As String
    Dim stLinkCriteria As String

    DoCmd.Close     ' Закрытие формы
    stDocName = "Building"
    DoCmd.OpenForm stDocName, , , stLinkCriteria
```

```
Exit_Кнопка2_Click:
    Exit Sub

Err_Кнопка2_Click:
    MsgBox Err.Description
    Resume Exit_Кнопка2_Click
End Sub
```

Листинг 3.6. Код процедуры нажатия кнопки с надписью STOP

```
Private Sub Кнопка1_Click()
   ' Кнопка STOP
   ' Выход из программного комплекса
   ' Возврат в MS Access
   DoCmd.Close     ' Закрытие формы
End Sub
```

Если в параметрах запуска MS Access 2010 сделать ссылку на эту форму, то после щелчка мышью по файлу Real Estate Часть I.accdb пользователь сразу увидит приложение в работе. Для этого выберите в главном окне MS Access 2010 кнопку **Файл**, а в левом нижнем углу открывшегося окна — кнопку **Параметры**. Выбор пункта **Текущая база данных** показан на рис. 3.41.

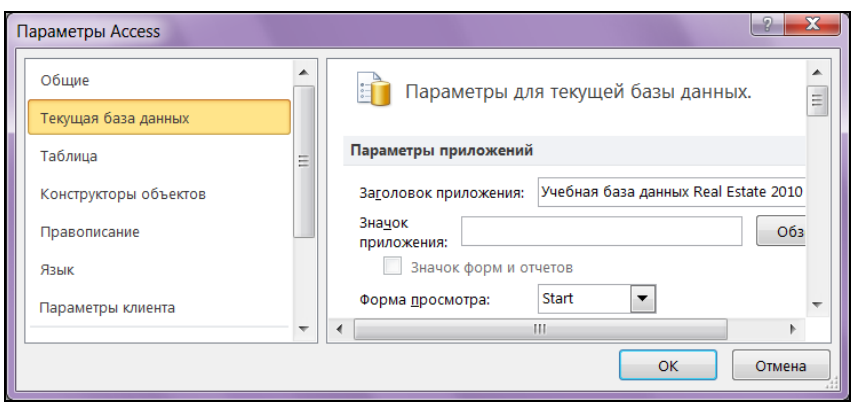


Рис. 3.41. Установка параметров запуска базы данных Real Estate 2010

Поле со списком **Форма просмотра** позволяет выбрать форму, которая будет выводиться на экран при открытии базы данных. Задайте заголовок и значок приложения. Например — building.ico. Наша первая форма носит название `Start`. Она и будет запущена на выполнение после открытия базы данных. Здесь же можно выполнить настройку главного окна MS Access 2010 при работе с базой данных. Просто поставьте или снимите флажки в нужном месте.

3.7. Создание простого отчета

Основная сфера применения форм — обеспечение возможности просмотра отдельных или небольших групп связанных записей. Отчеты же представляют собой наилучшее средство отображения информации из базы данных в виде печатного документа. Разработка отчета очень похожа на разработку формы. Используется та же панель элементов (за исключением двух), тот же список полей и окно свойств. В этом разделе мы построим относительно несложный отчет, пройдя шаг за шагом всю цепочку его создания.

Заставим наш программный комплекс выдавать на печать всю информацию, которая имеется в базе данных по отдельно взятой квартире: адрес, технические характеристики, ответственного квартиросъемщика, номер лицевого счета и список проживающих. Отчет будет запускаться из формы `Flats` (квартиры) и носить имя `Document`.

MS Access 2010 предоставляет пользователю возможность создавать множество различных отчетов любой степени сложности. Работа по созданию отчета всегда начинается с выбора источника, из которого будут извлекаться записи отчета. Отчет может представлять собой как простой список, так и подробную сводку данных, представленных в виде официального документа того или иного ведомства.

Однако в любом случае сначала определяют поля, которые должны войти в отчет, и в каких таблицах или запросах находятся эти поля.

Стандартное средство MS Access **Отчет** — самый быстрый способ создания отчета, потому что с его помощью отчет формируется сразу же, без уточнения дополнительной информации. В отчете будут представлены все записи базовой таблицы или запроса. Для демонстрации этого средства создадим отчет по улицам города. Для этого отображения данных из основных таблиц оно не подходит. Его назначение — информация из таблиц-справочников.

1. В области переходов щелкните таблицу `street`, на основе которой будем создавать отчет.
2. На вкладке **Создание** ленты MS Access 2010 в разделе **Отчеты** щелкните пиктограмму  **Отчет**. MS Access немедленно создаст отчет и отобразит его в режиме макета (рис. 3.42).
3. После просмотра отчет можно сохранить, а затем закрыть и его, и источник записей — таблицу или запрос. В следующий раз при его открытии MS Access отобразит в нем самые последние данные из таблицы `street`.

Полученный отчет очень далек от совершенства, он позволяет лишь быстро просмотреть базовые данные.

Вторая возможность — мастер отчетов — средство MS Access 2010, помогающее создать отчет на основании ответов, полученных на заданные пользователю во-

просы. Мастер отчетов предоставляет больше возможностей относительно выбора полей для включения в отчет. При этом разработчик может указать способ группировки и сортировки данных, а также включить в отчет поля из нескольких таблиц или запросов, но только в том случае, если отношения между этими таблицами и запросами заданы заранее.

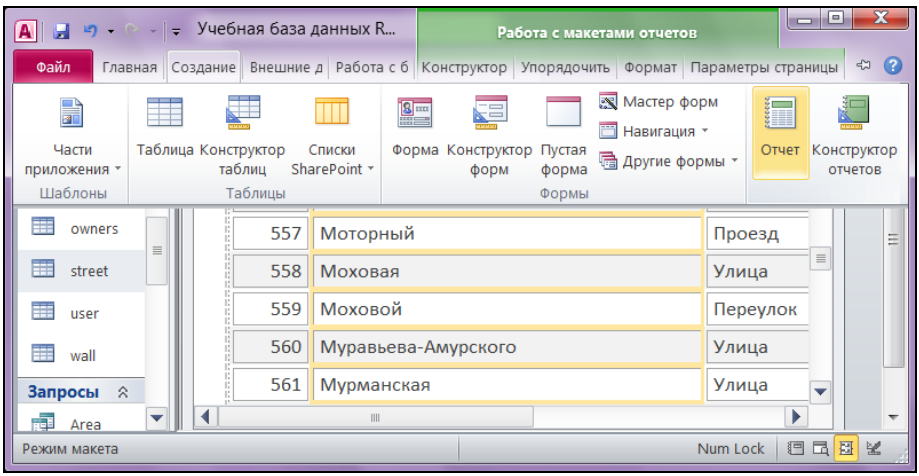


Рис. 3.42. Создание отчета при помощи стандартного средства **Отчет**

Пропустим описание работы с мастером и перейдем к работе с профессиональным средством создания отчетов — конструктором.

3.8. Создание отчета в режиме конструктора

В MS Access 2010 отчет разбит на разделы. Разделы отчета можно увидеть только в режиме конструктора. Чтобы созданные отчеты правильно работали, необходимо четко представлять назначение каждого раздела. Перечислим типы разделов и укажем назначение каждого из них.

- **Заголовок отчета.** Выводится на печать только один раз в начале отчета. В заголовок включается информация, обычно помещаемая на обложке: название отчета и дата. Заголовок отчета печатается перед верхним колонтитулом.
- **Верхний колонтитул.** Печатается вверху каждой страницы. Верхний колонтитул используется в тех случаях, когда нужно, чтобы название отчета повторялось на каждой странице.
- **Заголовок группы.** Размещается перед каждой новой группой записей. Используется для печати названия группы. Например, если отчет сгруппирован по зданиям, в заголовках групп можно указать их адрес.

- **Область данных.** Этот раздел печатается один раз для каждой строки данных из источника записей. В нем размещаются элементы управления, составляющие основное содержание отчета.
- **Примечание группы.** Печатается в конце каждой группы записей. Примечание группы можно использовать для печати сводной информации по группе.
- **Нижний колонтитул.** Печатается внизу каждой страницы. Используется для нумерации страниц и для печати постраничной информации.
- **Примечание отчета.** Печатается один раз в конце отчета. Примечание отчета можно использовать для печати итогов и другой сводной информации по всему отчету.

3.8.1. Подготовка к конструированию

На вкладке главной ленты MS Access 2010 **Создание** в разделе **Отчеты** найдите конструктор отчетов. После его запуска на ленте появятся четыре дополнительные вкладки: **Конструктор**, **Упорядочить**, **Формат** и **Параметры страницы**, а в центре экрана — окно конструктора отчетов. Новый пустой отчет содержит три раздела: верхний и нижний колонтитулы, между которыми находится область данных. Вы можете изменить размер любого раздела. Линейки с сантиметровыми делениями по верхнему и левому краям отчета помогают расположить данные на странице.

Если линейки отсутствуют, то для их появления на экране выберите вкладку главной ленты **Упорядочить**, в разделе **Размер и порядок** щелкните значок **Размер или интервал**. Появится меню. Найдите в нем в разделе **Сетка** пункт **Линейка**.

Верхний и нижний колонтитулы будут напечатаны соответственно вверху и внизу каждой страницы. Их можно убрать совсем с помощью пункта меню **Обработка событий...**, которое появится на экране, если сделать щелчок правой кнопкой мыши в любом месте отчета (рис. 3.43). Выберите в нем пункт **Колонтитулы страницы**. Расположенный чуть ниже пункт **Заголовок/примечание отчета** даст возможность создать заголовок, который будет напечатан только в начале отчета на первой странице.

Не забудьте про сетку. Она облегчит процесс конструирования. Вид заготовки отчета показан на рис. 3.43, который представляет собой композицию из фрагмента главной ленты и четырех окон:

- окно **Отчет1** обязано своему появлению значку **Конструктор отчетов**, расположенному на вкладке **Создание** ленты MS Access 2010;
- окно **Обработка событий** — щелчку правой кнопкой мыши в любом месте окна **Отчет1**;
- окно **Список полей** вызывается при помощи значка **Добавить поля** из раздела **Сервис** вкладки **Конструктор**;

♦ **Окно свойств** появляется на экране после активации значка **Страница свойств** из того же раздела.

Примечание

Одновременно на экране дисплея эти окна никогда не появляются.

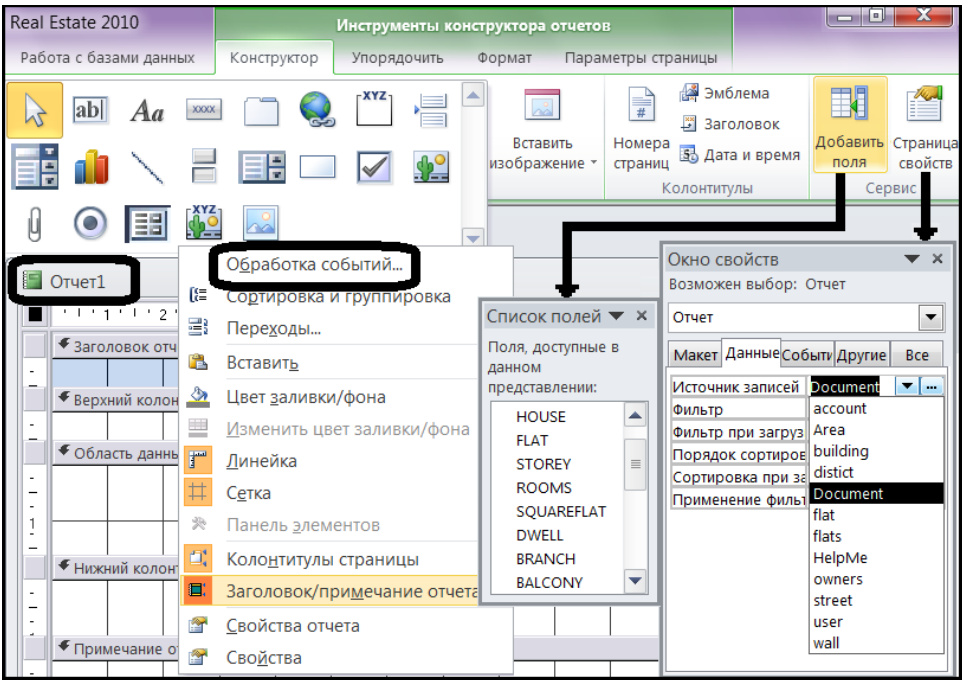


Рис. 3.43. Конструктор отчетов

3.8.2. Создание запроса

Конструктор отчетов практически всегда использует в своей работе данные базового запроса. При создании запроса нам понадобится информация из запроса `Flats` и таблиц `street` и `owners`. Напомню, что запрос `Flats` содержит данные по техническим характеристикам всех квартир, находящихся в отдельном конкретном здании, которое отображается в форме `Building`. Назовем базовый запрос так же, как и отчет, который будет построен на его основе, — `Document`.

1. Выберите вторую вкладку главной ленты MS Access 2010 — **Создание**.
2. Сделайте щелчок левой кнопкой мыши по значку **Конструктор запросов**, который находится в разделе **Макросы и код**. Появится окно конструктора запросов и окно с названием **Добавление таблицы**.

3. Первая вкладка этого окна содержит список таблиц текущей базы данных. Вторая — список запросов. Выберите в ней запрос `Flats` и нажмите кнопку **Добавить**. Таким же образом включите в новый запрос таблицы `street` и `owners`, расположенные на первой вкладке. Закройте окно **Добавление таблицы**.
4. Параметры объединения таблицы `street` и запроса `Flats` MS Access установил сам. В запрос будут включены только те записи из запроса `Flats` и таблицы `street`, в которых связанные поля совпадают. Объединить `Flats` и `owners` предстоит нам.
5. Поместите указатель мыши над полем `STREET` запроса `Flats`. Нажмите левую кнопку мыши и, не отпуская ее, перетащите появившийся значок поля на поле `STREET` таблицы `owners`. Проделайте аналогичную операцию с полями `HOUSE` и `FLAT` этого же запроса и таблицы. Верхняя часть окна конструктора запросов примет вид, представленный на рис. 3.44. Щелчок правой кнопкой мыши по линии, соединяющей поля из двух таблиц, приведет к появлению окна **Параметры объединения**. В нашем случае просто проверьте точность своих действий по "перетаскиванию" и оставьте переключатель, расположенный в нижней части окна, в первом положении.

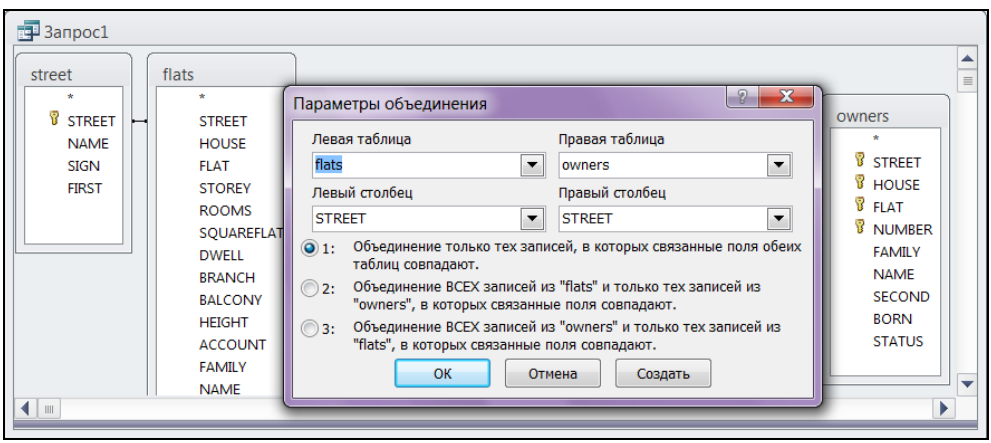


Рис. 3.44. Верхняя часть окна конструктора запросов после добавления таблиц и их объединения

6. Расположите последовательно все поля, которые хотите включить в запрос `Document` из таблиц `street`, `owners` и запроса `Flats`, в первой строчке нижней части окна конструктора запросов. Не удивляйтесь, если на втором десятке список полей в конструкторе MS Access 2010 закончится. Воспользуйтесь инструментом **Вставить столбцы**. Его можно найти в разделе **Настройка запроса** вкладки **Конструктор**.

7. В запрос `Document` должны попасть данные только по одной конкретной квартире. Ее номер отображен в объекте `FLAT` формы `Flats`. Для этого добавим условие отбора по полю `FLAT`:

   ```
   [Forms]![Flats]![FLAT]
   ```

8. Установите порядок сортировки записей, попавших в запрос `Document`. Они должны отображаться в документе по возрастанию порядковых номеров проживающих. Текст запроса приведен в листинге 3.7.

9. Сохраните созданный запрос под именем `Document` (рис. 3.45). Не забудьте, что корректно он будет запускаться только из формы `Flats` (квартиры), которая, в свою очередь, не может быть запущена без формы `Building` (здание).

Примечание

Условия отбора записей по полю `FLAT`: `[Forms]![Flats]![FLAT]` можно перевести так. В запрос должны попасть данные только по конкретной квартире. Ее номер следует взять в объекте `FLAT` формы `Flats`.

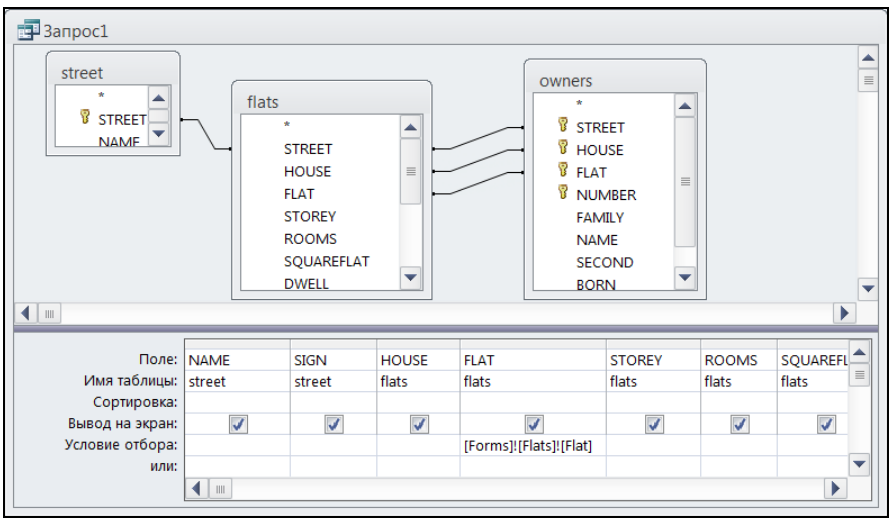


Рис. 3.45. Конечный вид окна конструктора запросов

Сделайте щелчок правой кнопкой мыши в свободном месте окна конструктора запросов. Появится контекстное меню. Первым пунктом в нем должен быть пункт **Режим SQL**. Щелкните по нему правой кнопкой мыши. Появится текст запроса на языке SQL, сгенерированный конструктором запросов.

Листинг 3.7. Вид базового запроса

```
SELECT street.NAME, street.SIGN, flats.HOUSE, flats.FLAT,
       flats.STOREY, flats.ROOMS, flats.SQUAREFLAT, flats.DWELL,
```

```
      flats.BRANCH, flats.BALCONY,
      flats.HEIGHT, flats.ACCOUNT, flats.FAMILY, flats.NAME,
      flats.SECOND, owners.NUMBER, owners.FAMILY, owners.NAME,
      owners.SECOND, owners.BORN, owners.STATUS,street.FIRST
FROM street INNER JOIN (flats INNER JOIN owners ON
      (flats.HOUSE = owners.HOUSE)
      AND (flats.FLAT = owners.FLAT)
      AND (flats.STREET = owners.STREET))
          ON street.STREET = flats.STREET
WHERE (((flats.FLAT)=[Forms]![Flats]![Flat]))
ORDER BY owners.NUMBER;
```

Обратите внимание на сложность текста запроса и на то, с какой легкостью мы сформировали его с помощью конструктора запросов, не зная ни одной конструкции и ни одного служебного слова языка SQL. Начинающему пользователю MS Access вовсе необязательно знать этот язык, но мы к нему обязательно вернемся в *части II*.

3.8.3. Добавление элементов в отчет

В отчетах MS Access 2010 применяются три типа элементов управления.

- *Присоединенные элементы управления*, связанные с полем источника данных для отчета. Это может быть поле таблицы, запрос и даже значение другого элемента управления. Самыми распространенными присоединенными элементами являются текстовые поля. Выключатели, переключатели и флажки связывают с логическим полем таблицы. Элемент OLE — с графическим объектом, видео- и звуковым файлом и т. д. Все присоединенные элементы при "рождении" получают связанные с ними метки. Значение метки представляет собой значение свойства **Подпись**, относящегося к вкладке **Макет**. Метку всегда можно удалить.
- *Свободные элементы управления* не зависят от источника данных отчета. Прямоугольники и линии — для оформления внешнего вида, а OLE — для добавления графики в отчет. Не все свободные элементы имеют метки.
- *Вычисляемые элементы управления* используют в качестве источника данных в выражении. В выражениях могут использоваться как поля таблиц, так и свободные элементы.

Для создания отчета, показанного на рис. 3.46, выполните следующие действия:

1. Откройте окно свойств отчета. Для этого перейдите на вкладку **Конструктор** и сделайте щелчок по значку **Страница свойств**, расположенному в разделе **Сервис**.
2. В окне свойств перейдите на вторую вкладку — **Данные.**

3. Выберите первое свойство — **Источник записей**. Раскройте поле со списком и найдите в нем запрос `Document`.
4. Сделайте щелчок правой кнопкой мыши в любом месте окна конструктора отчетов. Появится меню **Обработка событий**. Выберите пункт **Заголовок/примечание отчета**.
5. В этом же меню щелкните пункты **Сетка** и **Линейка**.
6. Перейдите на вкладку **Конструктор** для добавления в отчет полей запроса `Document`. Их список откроется в окне **Список полей** после щелчка по значку **Добавить поля**.
7. Поместите надпись в самом верху заголовка отчета и введите в нее текст "Справка". Выделите надпись щелчком мыши. Вокруг нее появятся маркеры.
8. Перейдите на вкладку **Формат**. В разделе **Шрифт** установите шрифт Arial Cyr размером 10 пунктов. Подчеркните текст и сделайте его выделенным. Здесь же можно назначить цвет текста и цвет фона.
9. Перейдите на вкладку **Упорядочить**. В разделе **Размер и порядок** щелкните по значку **Размер или интервал**. Откроется меню.
10. Выберите в нем пункт  **по размеру данных**. Размер элемента управления будет настроен в соответствии с назначенным шрифтом.

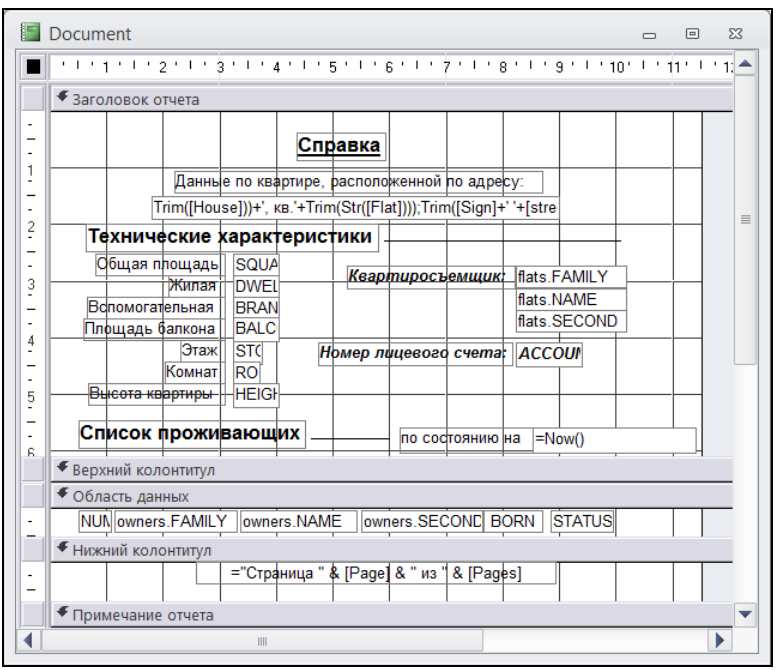


Рис. 3.46. Окончательный вид отчета в конструкторе отчетов

11. В этом же меню выберите пункт  **по узлам сетки**. Он предназначен для изменения места расположения одного или нескольких элементов путем выравнивания по узлам сетки.
12. Выполните аналогичные действия по размещению второй строки заголовка отчета — "Данные по квартире, расположенной по адресу".
13. Перетащите поля NAME, SIGN, HOUSE и FLAT из списка полей (вкладка **Конструктор**, значок **Добавить поля**) в заголовок отчета.
14. Выполните форматирование. Для этого выберите нужные элементы управления, щелкая по ним левой кнопкой мыши при нажатой клавише <Shift>. Откройте вкладку ленты **Упорядочить**. Выберите значок **Размер или интервал**. Откроется меню. В разделах **Размер** и **Интервал** вы найдете 12 пунктов меню для форматирования.
15. Перетащите поля NUMBER, FAMILY, NAME, SECOND, BORN и STATUS в область данных. Удалите подписи, которые к ним сгенерирует Access, и выполните форматирование. Окончательный вид отчета в конструкторе показан на рис. 3.46.

3.8.4. Включение в отчет даты, времени и номеров страниц

Любой документ, выдаваемый организацией, обязательно должен иметь в своем составе дату, а в некоторых случаях и время выдачи. Это единственный показатель актуальности сведений, которые он содержит. Большие отчеты должны иметь пронумерованные страницы. В MS Access 2010 есть несколько способов добиться желаемого результата.

Для добавления номера страницы в область верхнего или нижнего колонтитулов выполните следующие действия.

Откройте отчет в режиме конструктора.

1. На вкладке ленты **Конструктор** выберите значок **Номера страниц**. Он находится в разделе **Колонтитулы**. Откроется диалоговое окно **Номера страниц** (рис. 3.47).
2. Выберите формат, расположение и выравнивание для номеров страниц.
3. Снимите флажок **Отображать номер на первой странице**, если номер на первой странице не нужен.
4. Нажмите кнопку **OK**. Номера страниц будут добавлены в отчет.

Для добавления номеров страниц в другую область отчета, а также для вставки даты и времени создания документа применяется более общий способ — окно **Построитель выражений** (рис. 3.48).

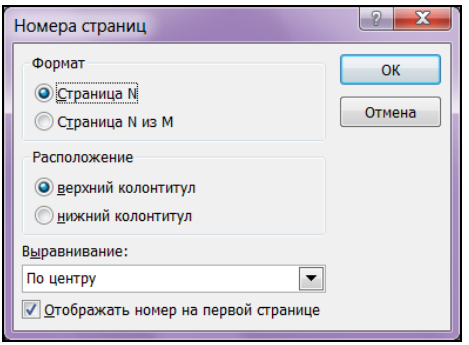


Рис. 3.47. Вставка номера страницы

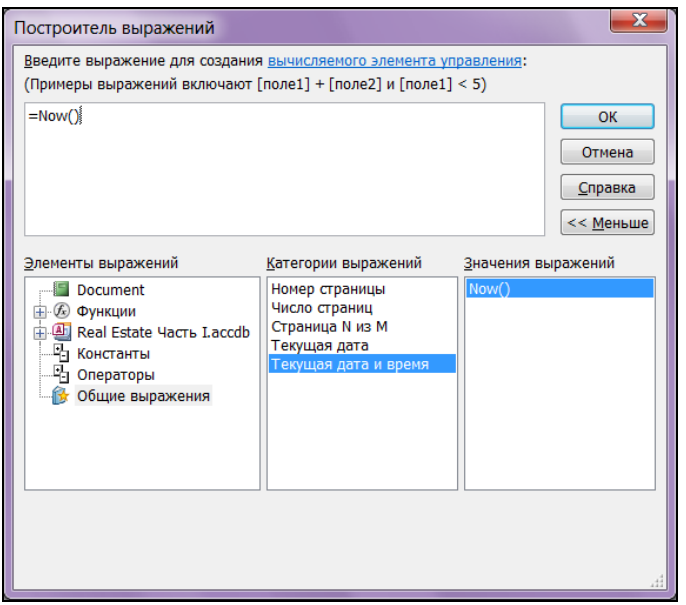


Рис. 3.48. Добавление в отчет даты и времени создания

Добавим в заголовок отчета надпись: "по состоянию на", после которой поместим дату создания отчета. Для этого:

1. Откройте отчет в режиме конструктора.
2. На вкладке **Конструктор** ленты MS Access 2010 выберите кнопку **Поле**. Она находится в разделе **Элементы управления**. Расположите курсор в области заголовка отчета. Курсор превратится в стилизованное изображение поля таблицы с крестиком в левом верхнем углу.
3. Сделайте щелчок левой кнопкой мыши и, удерживая ее, добейтесь требуемого размера поля. Кнопку отпустите.

4. Откройте окно свойств. Для этого сделайте щелчок по кнопке **Страница свойств** вкладки **Конструктор**.
5. В окне свойств выберите вторую вкладку **Данные**.
6. Перейдите к первой строчке **Данные** и нажмите кнопку ... . Появится окно построителя выражений (см. рис. 3.48).
7. В левом списке **Элементы выражений** выберите строчку **Общие выражения**, а в среднем **Категории выражений — Текущая дата и время**. В правом списке появится стандартная функция MS Access — `Now()`. Щелкните по кнопке **OK**. Эта функция со знаком = появится в поле значений свойства **Данные** создаваемого элемента.
8. Вместо названия надписи **Поле49:** напишите: "по состоянию на". Для этого щелчком мыши выделите надпись и внесите изменения прямо на месте.
9. Перейдите на вкладку **Формат**. В разделе **Шрифт** установите шрифт Arial Cyr размером 10 пунктов или любой другой. Подчеркните текст и сделайте его выделенным. Здесь же можно назначить цвет текста и цвет фона.
10. Перейдите на вкладку **Упорядочить**. В разделе **Размер и порядок** щелкните по значку **Размер или интервал**. Откроется меню. Выберите в нем пункт **по размеру данных**. Размер элемента управления будет настроен в соответствии с назначенным шрифтом.
11. В этом же меню выберите пункт **по узлам сетки**. Он предназначен для изменения места расположения одного или нескольких элементов путем выравнивания по узлам сетки. Выполните окончательное форматирование.

3.8.5. Добавление кнопки в форму для запуска отчета

Закончив работу, не торопитесь запускать отчет на выполнение. В автономном виде он не запустится. В этом нет ничего удивительного, поскольку отчет построен на базе запроса `Document`, который берет данные из формы `Flats` (квартиры), а та в свою очередь запускается на основе данных формы `Building` (здание), т. к. два параметра адреса — улицу и номер дома берет именно из этой формы.

Добавьте кнопку **Документ** в форму `Flats` и запускайте отчет только из этой формы. Текст процедуры VBA, сгенерированный мастером кнопок, для запуска этого отчета приведен в листинге 3.8.

Листинг 3.8. Текст обработки события *Нажатие кнопки*

```
Private Sub Кнопка40_Click()
On Error GoTo Err_Кнопка40_Click
    Dim stDocName As String
```

```
    stDocName = "Document"
    DoCmd.OpenReport stDocName, acPreview

Exit_Кнопка40_Click:
    Exit Sub

Err_Кнопка40_Click:
    MsgBox Err.Description
    Resume Exit_Кнопка40_Click
End Sub
```

Имя кнопки — `Кнопка40`. Это имя сгенерировано автоматически. У вас кнопка, скорее всего, будет иметь другой порядковый номер. Имя отчета — `Document`. Отчет запускается в режиме предварительного просмотра — `acPreview`. В случае возникновения ошибки при выполнении отчета о нем сообщит Microsoft Access, а не VBA — `MsgBox Err.Description`.

3.8.6. Вывод отчета MS Access на печать

При предварительном просмотре отчета в режиме целой страницы Microsoft Access выводит его так, как сделал бы принтер. По умолчанию стандартные поля с трех сторон составляют по 1,5 см, а слева — 1 см. У нас имеется возможность изменить параметры печати отчета. Для этого:

1. Откройте отчет в режиме конструктора.
2. Перейдите на последнюю вкладку ленты MS Access 2010 — **Параметры страницы** и сделайте щелчок по значку **Параметры страницы**.

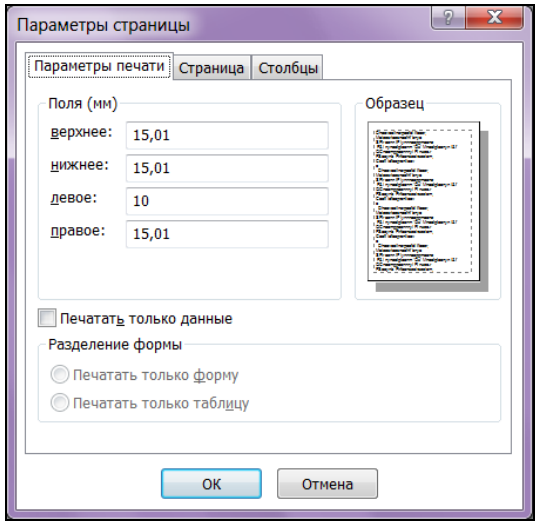


Рис. 3.49. Окно **Параметры страницы**

3. Откроется диалоговое окно **Параметры страницы** (рис. 3.49). Для того чтобы свести к минимуму расход бумаги при печати рабочих отчетов, установите флажок **Печатать только данные**. Верхний и нижний колонтитулы, а также заголовок отчета и примечание отчета печататься не будут.
4. Задайте в окне **Параметры страницы** поля страницы, выводимой на принтер.

Для того чтобы распечатать отчет, находясь в режиме предварительного просмотра (рис. 3.50), сделайте щелчок правой кнопкой мыши в любом месте отчета. Появится окно **Печать** вашей операционной системы. Можете распечатать отчет целиком или только необходимые страницы. Также имеется возможность указать количество экземпляров отчета и отправить отчет в файл для последующей распечатки.

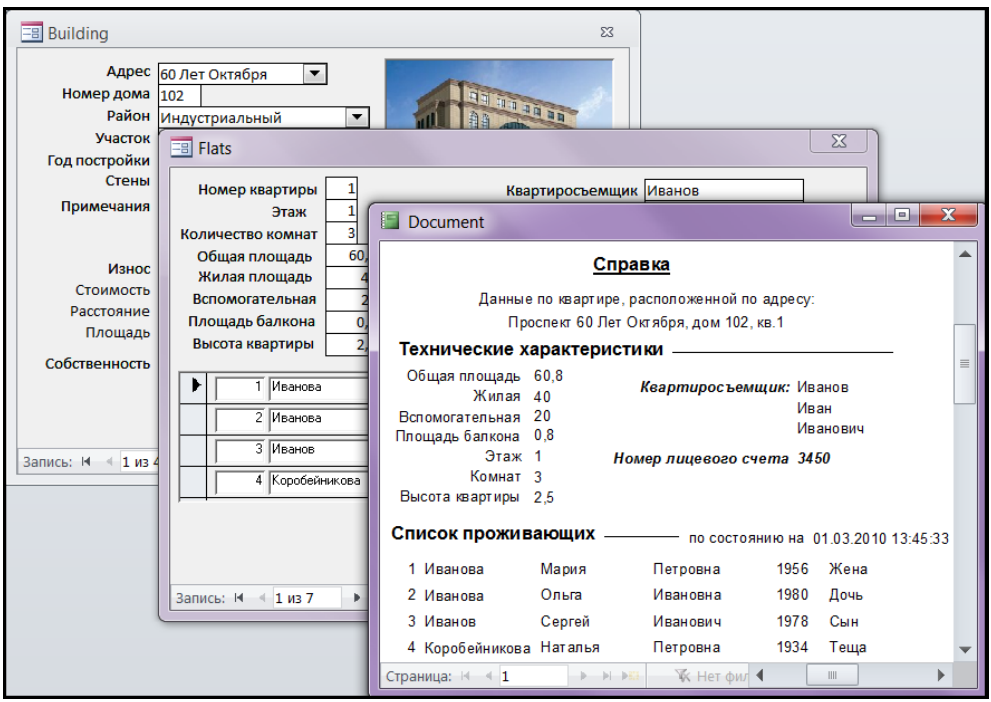


Рис. 3.50. Вид отчета в режиме предварительного просмотра

3.8.7. Добавление отчету интеллектуальности

Справка, полученная в результате вывода на печать отчета `Document`, представляет собой официальный документ. Инспектору остается только поставить на нем подпись и печать. Однако в таком виде адрес здания в ней всегда будет начинаться с признака, написанного с большой буквы (Улица, Переулок, Проспект

и т. д.), а это не совсем то, что нам нужно! Ведь в качестве признака может фигурировать слово, стоящее в адресе на втором месте и начинающееся с маленькой буквы (шоссе, проезд, бульвар и т. д.). Об этом уже шла речь в *главе 2*, и все необходимые атрибуты для правильной записи адреса нами уже предусмотрены. Вернемся к нашему отчету и внесем в него необходимые изменения (рис. 3.51).

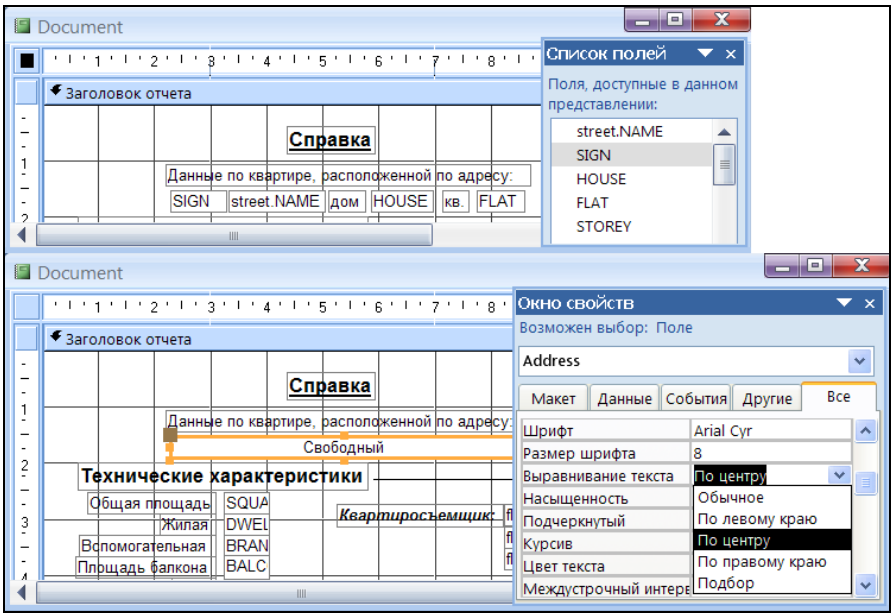


Рис. 3.51. Работа с адресом здания в справке

Удалим поля и надписи (`street.NAME`, `дом`, `HOUSE`, `кв.`, `FLAT`). Для этого надо сделать щелчок мышью по объекту и нажать клавишу <Delete>. Поле `SIGN` переименуем в `ADDRESS`, увеличив его в размерах и отцентрировав. Выделите поле мышью и клавишами-стрелками при нажатой клавише <Shift> добейтесь требуемых результатов. В окне свойств напротив свойства **Имя** поставьте `ADDRESS`. В поле объекта удалите надпись `SIGN`. В конструкторе отчета в поле `ADDRESS` появится надпись "Свободный". В окне свойств напротив свойства **Выравнивание текста** в раскрывающемся поле со списком найдем значение *По центру*. Сделаем поле `ADDRESS` вычисляемым, включив в состав выражения необходимые поля и надписи в требуемом порядке. Для этого в окне свойств найдем свойство **Данные** и щелчком по кнопке ... запустим построитель выражений (рис. 3.52).

Приведем некоторые сведения о построителе выражений. Окно построителя выражений состоит из четырех разделов.

- **Поле выражения.** В верхней части окна построителя расположено поле, в котором создается выражение. Ниже находится раздел, предназначенный

для создания элементов выражения и их последующей вставки в поле выражения. Допускается непосредственный ввод части выражения в поле выражения.

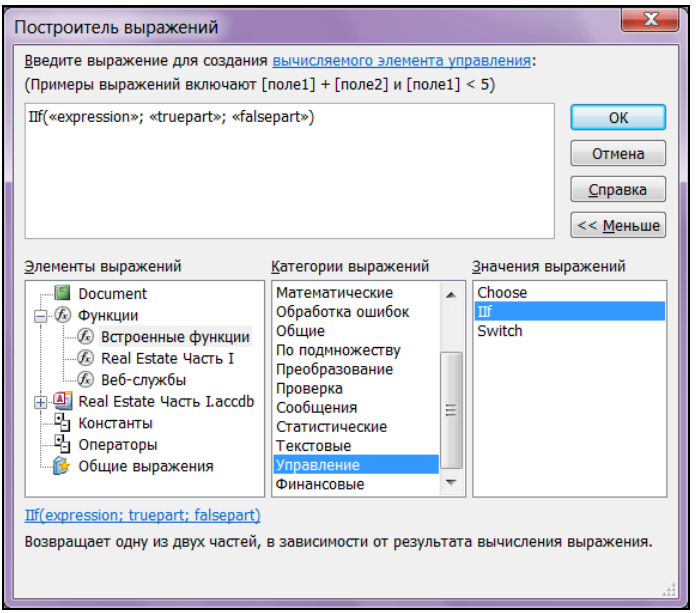


Рис. 3.52. Начало работы с построителем выражений

- **Элементы выражения.** В нижней части окна построителя находятся три списка.
  - В левом списке **Элементы выражений** выводятся папки, содержащие таблицы, запросы, формы, объекты базы данных, встроенные и определенные пользователем функции, константы, операторы и общие выражения.
  - В среднем списке **Категории выражений** задаются определенные элементы или типы элементов для папки, заданной в левом поле. Например, если выбрать в левом списке **Встроенные функции**, то в среднем списке появится перечень всех типов функций Microsoft Access.
  - В правом списке **Значения выражений** выводится перечень значений (если они существуют) для элементов, заданных в левом и среднем списках. Например, если выбрать в левом списке **Встроенные функции** и тип функции в среднем, то в правом списке будет выведен перечень всех встроенных функций выбранного типа.

Примечание

При вставке идентификатора в выражение построитель вставляет только те его части, которые требуются в текущем контексте. Например, при запуске построителя

выражений из окна свойств формы `Building` и вставке идентификатора для свойства **Вывод на экран** (Visible) будет вставлено только имя свойства — **Visible**. При использовании данного выражения вне контекста формы необходимо включать полный идентификатор: `Forms![Building].Visible`.

Нажмите клавишу <=>. Выражение, создаваемое при помощи построителя, как правило, начинается со знака присваивания.

Далее используем встроенную функцию:

```
IIF(expr; truepart; falsepart)
```

которая в зависимости от значения логического выражения `expr` назначит в нашем случае значением поля `ADDRESS` либо первую цепочку (название + признак + дом + квартира), либо вторую (признак + название + дом + квартира). Критерием выбора цепочки адреса является поле `FIRST` таблицы улиц (см. табл. 2.2).

Предупреждение

В таблицах Microsoft Access логическое поле имеет тип **Числовой**, значению `True` соответствует число –1 (минус один), а значению `False` — число 0 (ноль). В таблицах же Microsoft SQL Server логическое поле имеет тип `Bit` (`True` — один, `False` — ноль).

Совет

Чтобы избежать проблем при переводе приложения Microsoft Access на платформу SQL Server, не "привязывайтесь" к конкретным значениям полей таблиц, а используйте константы `True` и `False`, которые построитель выражений Microsoft Access сразу же переведет на русский язык (`Истина` и `Ложь`), если вы работаете с русскоязычной версией Microsoft Access 2010 (рис. 3.53).

Для окончательного построения выражения нам потребуется текстовая функция `Trim(stringexpr)`, убирающая концевые пробелы, и функция преобразования `Str(number)`, преобразующая число в символы.

В листинге 3.9 приведен законченный вид выражения для вычисляемого поля `ADDRESS`.

Листинг 3.9. Выражение для вычисляемого поля

```
=IIF([First]=Истина;
     Trim([street.Name]+' '+[Sign]+', дом '+(Trim([House]))+',кв.'+
     Trim(Str([Flat])));Trim([Sign]+' '+[street.Name])+
     ', дом '+(Trim([House]))+', кв.'+Trim(Str([Flat])))
```

На рис. 3.54 показана работа сконструированного нами выражения. Обратите внимание на правила записи выражений. Имена полей должны быть заключены в квадратные скобки, а текстовые константы — в апострофы или кавычки. Пара-

метры функций — в круглые скобки. Если параметров несколько, то они отделяются друг от друга точкой с запятой. В выражении общее число открывающих круглых скобок обязательно должно быть равно числу закрывающих.

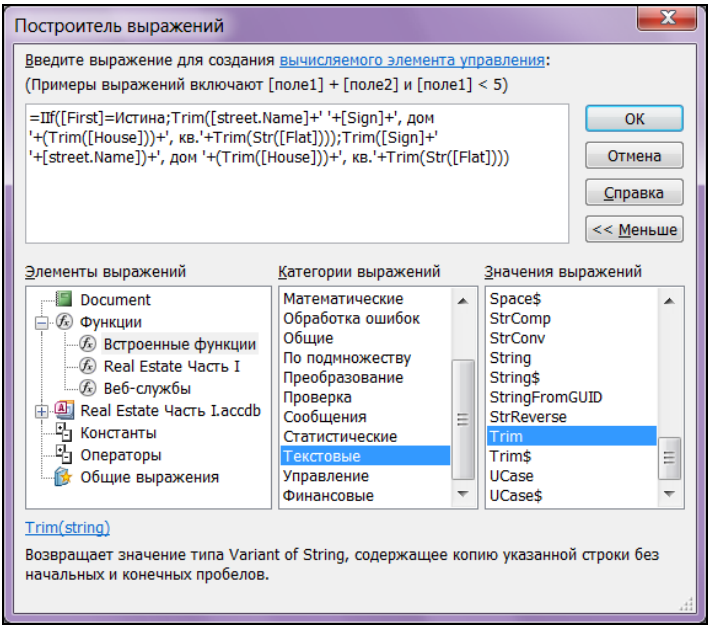


Рис. 3.53. Выражение для вычисляемого поля готово

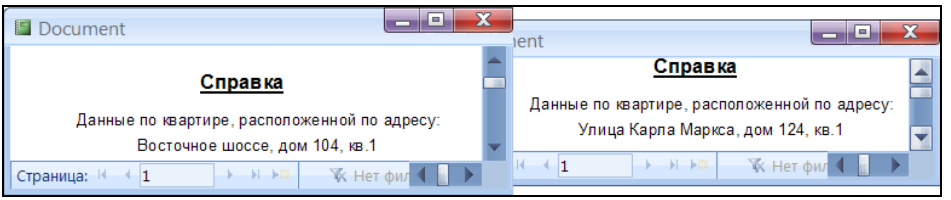


Рис. 3.54. Запись адреса по правилам русского языка

Чтобы работать с Microsoft Access более эффективно, необходимо научиться создавать простые выражения с использованием функций и операций. Выражения применяются, как правило, для проверки условий или для арифметических вычислений.

Выражение создается с помощью комбинации идентификаторов, операторов и значений, обеспечивающих получение необходимого результата. Выражения можно создавать самостоятельно или с помощью построителя выражений.

3.9. Операторы Microsoft Access для построения выражений

На этот раз рассмотрим процесс создания выражения без помощи построителя. Введите комбинацию идентификаторов, операторов и значений в элемент управления, в котором требуется получить результат. Например, следующее выражение увеличивает значение элемента управления `Cost` (стоимость здания) формы `Building` на 20%:

```
= [Forms]![Building]![Cost] * 1,2
```

Примечание

Оператор `!` указывает, что следующий за ним элемент является элементом, определяемым пользователем (элементом семейства). Перед выражениями, определяющими вычисляемые элементы управления, всегда следует помещать знак присваивания (=). Некоторые выражения дают логические результаты `Истина` или `Ложь`.

Арифметические операторы выполняют сложение, вычитание, умножение и деление (табл. 3.2).

Таблица 3.2. *Арифметические операторы*

Оператор	Описание	Пример
+	Складывает два операнда	[Dwell]+[Branch]
–	Вычитает один операнд из другого	[SquareFlat]-[Balcony]
– (унарный)	Меняет знак операнда	-345
*	Перемножает два операнда	[Cost]*[Wear]
/	Делит один операнд на другой	[Cost]/[Square]
\	Делит целый операнд на другой нацело	[Year]\2
Mod	Возвращает остаток от целочисленного деления	[Cost] Mod 12
^	Возводит операнд в степень	[Line]^2

Операторы сравнения сравнивают значения двух операндов и возвращают логические значения (`Истина` или `Ложь`), соответствующие результату сравнения (табл. 3.3).

Таблица 3.3. *Операторы сравнения*

Оператор	Описание	Пример	Результат
>	Больше	5678>3000	Истина
>=	Больше или равно	234>=2341	Ложь

***Таблица 3.3** (окончание)*

Оператор	Описание	Пример	Результат
<	Меньше	1000<1001	Истина
<=	Меньше или равно	6789<=6789	Истина
<>	Не равно	567<>567	Ложь

Логические операторы используются для объединения результатов двух или более сравнений в одно (табл. 3.4).

***Таблица 3.4.** Логические операторы*

Оператор	Описание	Примеры	Результат
And	Логическое "И" (конъюнкция)	Истина And Ложь	Ложь
		Истина And Истина	Истина
Or	Логическое "ИЛИ" (дизъюнкция)	Ложь Or Ложь	Ложь
		Истина Or Ложь	Истина
Not	Логическое отрицание	Not Ложь	Истина
		Not Истина	Ложь

3.10. Стандартные функции Microsoft Access

Функции предназначены для возврата значений в точку вызова. В MS Access 2010 имеется более 150 различных стандартных функций. Приведу особенно часто используемые из них.

Более подробные сведения вы можете получить из справочной системы MS Access 2010 (на русском языке) и справочника Access VBA (к сожалению, в последних версиях продукта — на английском).

Математических функций MS Access (табл. 3.5) вполне достаточно для большинства инженерных приложений.

***Таблица 3.5.** Математические функции*

№	Функция	Описание функции	Пример	Значение
1	Abs()	Возвращает абсолютную величину числа	Abs(-345.6)	345.6
2	Atn()	Возвращает арктангенс числа в радианах	Atn(1)	0.7853982

***Таблица 3.5** (окончание)*

№	Функция	Описание функции	Пример	Значение
3	`Cos()`	Возвращает косинус угла, в радианах	`Cos(1)`	0.5403023
4	`Exp()`	Возвращает значение экспоненты	`Exp(1)`	2.7182818
5	`Int()`	Округляет число до ближайшего минимального целого (см. примеры)	`Int(-15.2)` `Int(13.6)`	–16 13
6	`Log()`	Возвращает натуральный логарифм числа	`Log(10)`	2.302585
7	`Rnd()`	Возвращает случайное число в диапазоне от 0 до 1	`Rnd()`	0.2895625 (произвольное)

8	`Sgn()`	Возвращает 1 для положительного числа, 0 для нулевого, –1 для отрицательного числа	`Sgn(-10.1)`
			`Sgn(0)`
			`Sgn(10.1)`
				–1
				0
				1

9	`Sin()`	Возвращает синус угла, выраженного в радианах	`Sin(1)`	0.8414710
10	`Sqr()`	Возвращает квадратный корень	`Sqr(2)`	1.4142136

Другие функции, полезные для начинающего разработчика, приведены в табл. 3.6.

***Таблица 3.6.** Другие функции*

№	Функция	Описание функции	Пример	Значение
1	`Date()`	Возвращает текущую системную дату	`Date()`	12.08.2010
2	`Day()`	Возвращает день из значения даты	`Day(Date())`	12
3	`Month()`	Возвращает месяц из значения даты	`Month(Date))`	8
4	`Now()`	Возвращает дату и время из системных часов компьютера	`Now()`	12.08.2010 11:42:28
5	`Time()`	Возвращает время из системных часов компьютера	`Time()`	11:45:40
6	`Year()`	Возвращает год из значения даты	`Year(Date())`	2010
7	`Chr()`	Возвращает как текст знак, соответствующий коду ANSI	`Chr(37)`	% (процент)
8	`Lcase()`	Переводит текст в нижний регистр	`Lcase("aSD")`	asd
9	`Ucase()`	Переводит текст в верхний регистр	`Lcase("aSD")`	ASD

Таблица 3.6 *(окончание)*

№	Функция	Описание функции	Пример	Значение
10	`Mid()`	Возвращает подстроку из строки. Необходимо указать, с какого символа и сколько символов	`Mid("abcdef",2,3)`	bcd
11	`Rtrim()`	Удаляет пробелы после текста	`Rtrim("abcd ")`	abcd
12	`Ltrim()`	Удаляет пробелы перед текстом	`Ltrim(" abcd")`	abcd
13	`Trim()`	Удаляет пробелы до текста и после него	`Trim(" abcd ")`	abcd
14	`Str()`	Преобразует число в текст	`Str(1234.56)`	1234.56
15	`Val()`	Преобразует текст в число	`Val("1234.56")`	1234.56

ГЛАВА 4

Дополнительные возможности MS Access 2010

Microsoft Access 2010 — это уже девятая версия продукта, впервые появившегося в 1992 году. По данным компании Microsoft, до 2010 года в мире продано более 100 миллионов копий Access всех версий, что дает основание считать эту СУБД самой популярной системой управления базами данных для персональных компьютеров. Этой популярностью Access полностью обязан своим разработчикам, которые, наряду с прекрасным исполнением (абсолютно вся информация хранится в одном файле), простотой освоения, эффективностью работы и эффектным доступом к данным из других источников, предоставили пользователю богатейшие дополнительные возможности, делающие работу с Microsoft Access еще более эффективной и комфортной. Вот некоторые из них.

4.1. Сжатие базы данных

Вы удалили форму или таблицу из базы данных Microsoft Access. Обратите внимание, размер файла этой базы данных остался прежним! При удалении записи из таблицы место, которое она занимала в базе, также автоматически не освобождается и не используется для хранения новой записи. Более того, после таких удалений база данных хранится в дезорганизованном виде, хотя и остается полностью работоспособной. Чтобы уменьшить размер файла базы данных и увеличить ее быстродействие, воспользуйтесь служебной программой. Средства сжатия и восстановления, начиная с MS Access 2002, усовершенствованы и интегрированы в единый процесс, что делает их более защищенными и эффективными. Для запуска служебной программы сжатия базы данных:

1. Запустите MS Access 2010.
2. Откройте базу данных. Для этого на вкладке **Файл** стартового окна MS Access найдите нужную базу, если она там есть, или воспользуйтесь пунктом **Открыть**. Появится окно базы данных.

3. Щелкните заголовок вкладки **Файл.** В появившемся меню система сделает активным пункт **Сведения**, который содержит несколько значков.
4. Выберите среди них третий: **Сжать и восстановить.**

Предупреждение

После окончания процесса сжатия активизируется окно базы данных. Сжатая база данных хранится под тем же именем, что и перед сжатием.

При работе с базой данных размером до 10—15 Мбайт имеет смысл включить эту опцию в настройки MS Access. Тогда, закрывая базу данных, вы автоматически запустите утилиту сжатия. Время ее работы при таком размере составляет 2—3 секунды. Для этого:

1. На вкладке **Файл** окна MS Access 2010 выберите пункт **Параметры**.
2. Появится окно **Параметры Access**. Выберите в нем пункт **Текущая база данных.**
3. В разделе **Параметры приложений** поставьте флажок **Сжимать при закрытии**.
4. Для завершения процесса настройки нажмите кнопку **OK**.

4.2. Преобразование базы данных в формат MS Access 2007/2010

Функция преобразования базы данных в новый формат появилась впервые только в Microsoft Access 2000. Более ранние версии не позволяли выполнять такие преобразования, а разработчикам настоятельно рекомендовалось иметь в своем распоряжении несколько версий этого продукта и вести разработку на той, которая есть у заказчика. Процесс преобразования файла базы данных предыдущих версий (2000 и 2002—2003) к виду Microsoft Access 2007/2010 предельно прост.

1. Запустите MS Access 2010.
2. Откройте базу данных. Для этого на вкладке **Файл** стартового окна MS Access найдите нужную базу, если она там есть, или воспользуйтесь пунктом **Открыть.**
3. Появится окно **Улучшение базы данных** (рис. 4.1). Щелкните по кнопке **Да.**
4. Появится диалоговое окно **Сохранение**, в котором надо указать название файла базы данных и папку, в которую будет помещена приведенная к формату MS Access 2007/2010 база.

Примечание

Окно (см. рис. 4.1) появится только в том случае, если открываемая база данных создана в формате предыдущих версий MS Access.

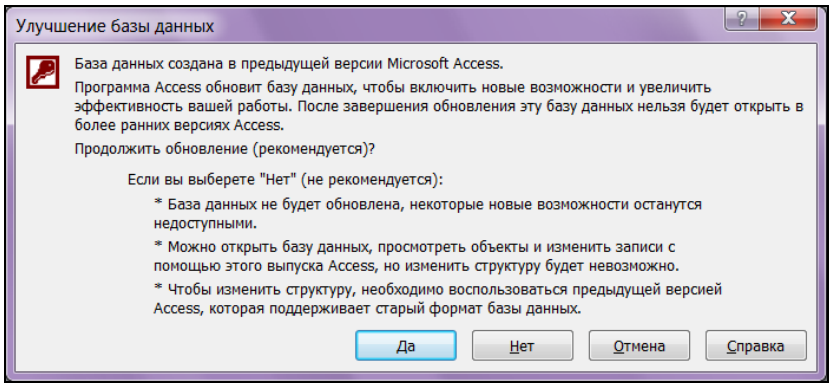


Рис. 4.1. Окно преобразования базы данных

Преобразование базы данных формата MS Access 2007/2010 к предыдущим версиям (2000 и 2002—2003) возможно, но корректно воспользоваться им можно только в случае, если при разработке не использовались новые возможности последней версии. Для преобразования выполните следующие действия.

1. Откройте базу данных. Для этого на вкладке **Файл** стартового окна MS Access найдите нужную базу, если она там есть, или воспользуйтесь пунктом **Открыть**.
2. Щелкните на заголовке вкладки **Файл**. В появившемся меню система сделает активным пункт **Сведения**, который содержит несколько значков.
3. Выберите пункт меню **Доступ**, а в изменившейся правой части окна — тип формата базы данных Access 2002/2003 или Access 2000.
4. Появится окно **Сохранение** операционной системы вашего компьютера. Выберите папку и укажите имя файла, в котором требуется сохранить преобразованную базу данных, и нажмите кнопку **Сохранить**.

4.3. Анализ быстродействия базы данных

Для оптимизации быстродействия базы данных Microsoft Access 2010 применяется специальная служебная программа — анализатор быстродействия. Для ее запуска:

1. Откройте базу данных.
2. Перейдите в окне MS Access 2010 на четвертую вкладку главной ленты с названием **Работа с базами данных**.
3. В разделе **Анализ** выберите значок **Анализ быстродействия**. Появится одноименное диалоговое окно (рис. 4.2).
4. Выберите вкладку, соответствующую типу объекта базы данных.

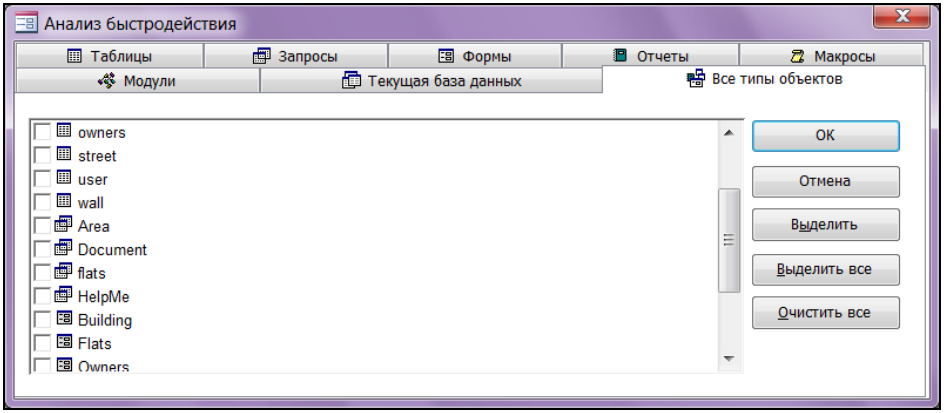


Рис. 4.2. Окно для анализа быстродействия базы данных

Анализатор быстродействия выдает три типа рекомендаций по оптимизации производительности: советы, предложения и мысли. При выделении элемента в списке **Результаты анализа** сведения о предлагаемом решении выводятся в отдельном окне **Анализ быстродействия**. Операции оптимизации, как правило, подразумевают определенные компромиссы, которые следует иметь в виду, приступая к оптимизации. Для получения дополнительных сведений о рекомендации выберите ее в списке и просмотрите информацию в области **Примечания**. Microsoft Access 2010 может автоматически выполнять рекомендации типа "совет" и "предложение". Рекомендации типа "мысль" выполняются вручную. Очень часто они кажутся примитивными. Например, анализатор советует заменить тип поля `HOUSE` таблицы `flat` с **Текстовый** на *Длинное целое*. А как быть в этом случае с номером дома в виде 104а? Очень просто: добавьте еще одно поле в таблицу, в котором и будет находиться литерная составляющая номера дома.

4.4. Сохранение базы данных в виде accde-файла

Если база данных содержит программы Microsoft Visual Basic, то при сохранении ее в виде accde-файла будут скомпилированы все модули, удалены все изменяемые исходные программы, а конечная база данных будет сжата. Программы Visual Basic будут по-прежнему выполняться, но их нельзя будет просматривать или изменять, благодаря чему уменьшится размер базы данных. Кроме того, будет оптимизировано использование памяти, что повысит быстродействие.

Сохранение базы данных как accde-файла делает невозможным выполнение следующих действий:

- просмотр, изменение или создание форм, отчетов или модулей в режиме конструктора;

- добавление, удаление или изменение ссылок на библиотеки объектов или базы данных;
- изменение программы с помощью свойств или методов Microsoft Access или модели объектов VBA (accde-файл не содержит текстов исходных программ);
- импорт и экспорт форм, отчетов или модулей.

Примечание

Любые таблицы, запросы, страницы доступа к данным или макросы в базах данных, являющихся accde-файлами, могут быть импортированы в другую базу данных MS Access.

Обязательно сохраните копию исходной базы данных MS Access 2010. В базе данных MS Access, сохраненной как accde-файл, нельзя изменять структуру форм, отчетов или модулей. Чтобы изменить структуру этих объектов, следует сделать это в исходной базе данных, а затем снова сохранить ее как accde-файл. При сохранении в виде accde-файла базы данных, содержащей таблицы, возникают сложности согласования различных версий данных в случае последующего изменения структуры форм, отчетов или модулей.

В будущих версиях Microsoft Access открывать, преобразовывать или выполнять программы в accde-файлах, скорее всего, будет невозможно. Единственным способом преобразования accde-файла MS Access 2007/2010 в формат новых версий будет открытие исходной базы данных MS Access, в которой был создан accde-файл, ее преобразование и последующее сохранение преобразованной базы данных MS Access в виде accde-файла (если компания Microsoft не придумает что-нибудь новое).

Accde-файлы создаются из файлов accdb с помощью описанной далее процедуры.

1. Откройте базу данных, которую требуется сохранить в виде файла accde. Для этого на вкладке **Файл** стартового окна MS Access найдите нужную базу, если она там есть, или воспользуйтесь пунктом **Открыть**.
2. Щелкните на заголовке вкладки **Файл**. В появившемся меню система сделает активным пункт **Сведения**, который содержит несколько значков.
3. Выберите пункт меню **Доступ**, а в изменившейся правой части окна — пункт **Создать ACCDE**.
4. Появится окно **Сохранение** операционной системы вашего компьютера. Выберите папку и укажите имя файла, в котором требуется сохранить преобразованную базу данных, и нажмите кнопку **Сохранить**.

4.5. Анализ данных в Microsoft Excel

Еще одна дополнительная возможность, предоставленная нам разработчиками Microsoft Access. Практически любой объект этой СУБД можно передать в MS Excel и там проанализировать. В этом случае можно использовать всю

мощь электронных таблиц от богатейшего набора стандартных функций до построения прекрасных диаграмм. Более всего это средство подходит для работы с запросами.

Рассмотрим пример. Отдел социальной защиты администрации Центрального района города прислал запрос с просьбой предоставить списки всех граждан, стоящих на учете в вашей организации, старше 55 лет.

Для создания такой выборки:

1. В области навигации базы данных Real Estate 2010 откройте раздел **Запросы**.
2. Щелчком правой кнопки мыши выделите объект, который следует передать в Excel. Пусть это будет запрос `HelpMe`, в котором содержатся данные о проживающих (рис. 4.3). Текст запроса приведен в листинге 4.1.

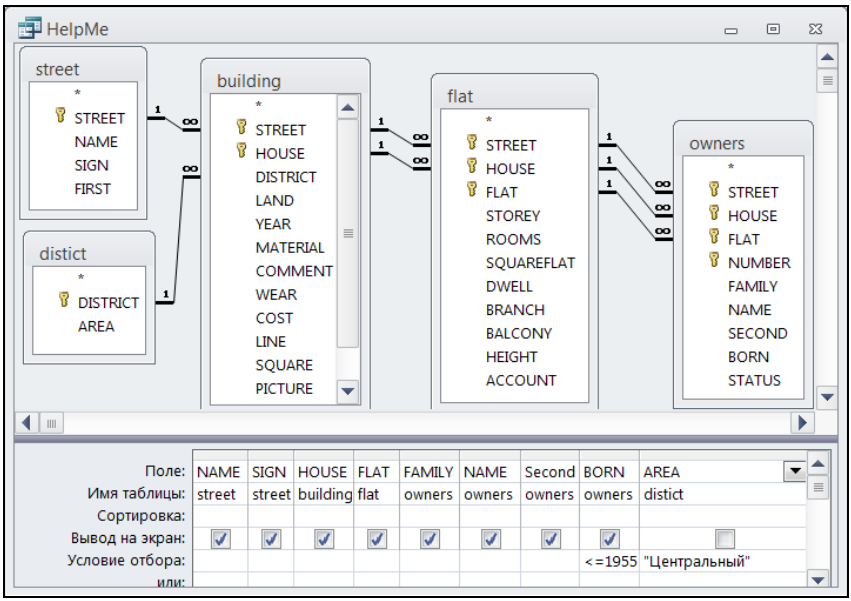
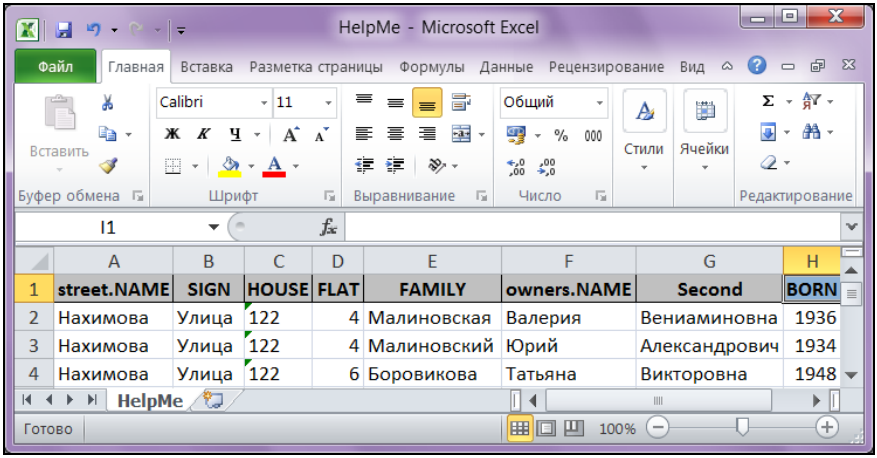


Рис. 4.3. Запрос в конструкторе запросов

3. В открывшемся контекстном меню выберите третий пункт **Экспорт**.
4. Появится еще одно меню со списком назначения. Найдите в нем **Excel**.
5. Откроется диалоговое окно **Экспорт — Электронная таблица Excel**.
6. В разделе параметров экспорта поставьте флажки **Экспортировать данные с макетом и форматированием** и **Открыть целевой файл после завершения операции экспорта**. Нажмите кнопку **OK**.
7. Автоматически запустится MS Excel 2010, в котором в виде таблицы будут отображены данные по выбранному объекту (рис. 4.4).



	A	B	C	D	E	F	G	H
1	street.NAME	SIGN	HOUSE	FLAT	FAMILY	owners.NAME	Second	BORN
2	Нахимова	Улица	122	4	Малиновская	Валерия	Вениаминовна	1936
3	Нахимова	Улица	122	4	Малиновский	Юрий	Александрович	1934
4	Нахимова	Улица	122	6	Боровикова	Татьяна	Викторовна	1948

Рис. 4.4. Запрос `HelpMe`, переданный в Microsoft Excel 2010

Листинг 4.1. Текст SQL-запроса

```
SELECT street.NAME, street.SIGN, building.HOUSE, flat.FLAT,
       owners.FAMILY, owners.NAME, owners.SECOND, owners.BORN
FROM ((street INNER JOIN building ON street.STREET = building.STREET)
      INNER JOIN flat ON (building.HOUSE = flat.HOUSE)
      AND (building.STREET = flat.STREET))
      INNER JOIN owners ON (flat.FLAT = owners.FLAT)
      AND (flat.HOUSE = owners.HOUSE)
      AND (flat.STREET = owners.STREET)
WHERE (((district.AREA)="Центральный")
      AND ((owners.BORN)<=1955));
```

4.6. Повышение быстродействия Microsoft Access

Соблюдение следующих правил поможет повысить производительность MS Access 2010.

- При работе с базами данных, которые не применяются другими пользователями, устанавливайте Microsoft Access и все свои базы данных на собственном жестком диске, а не на сетевом сервере.
- Чтобы быть единственным пользователем базы данных, откройте ее в монопольном режиме. Для этого в диалоговом окне **Открытие файла базы данных** нажмите стрелку рядом с кнопкой **Открыть** и выберите в списке вариант **Монопольно**.

- Чтобы освободить память, закройте неиспользуемые приложения.
- Увеличьте оперативную память компьютера.
- Регулярно удаляйте ненужные файлы, выполняйте сжатие базы данных, а после этого проводите дефрагментацию диска с помощью служебной программы дефрагментации. Для запуска этой программы нажмите кнопку **Пуск** в Windows, последовательно выберите пункты **Программы | Стандартные | Служебные | Дефрагментация диска.**
- Замените рисунок или фоновый узор, выбранный для рабочего стола Windows, на однородный экран.
- Не используйте программы сжатия диска или переместите базы данных на несжатый диск.

Предупреждение

Если с вашим программным обеспечением работает несколько компьютеров, соединенных в локальную вычислительную сеть, то большинство вышеизложенных рекомендаций не даст желаемого результата. Настало время разделить данные и приложение.

4.7. Разделение данных и приложения

Изначально MS Access 2010 хранит все объекты данных и элементы интерфейса в одном файле accdb. При разработке однопользовательского приложения, которое размещается целиком на "персоналке", это очень удобно. Если перед вами стоит задача обеспечить работу нескольких пользователей с программным комплексом, то поместите этот файл на один компьютер и обеспечьте доступ к нему всех работающих. Каждый раз, когда пользователю понадобится какой-нибудь объект, например форма или отчет, MS Access будет пересылать по локальной вычислительной сети весь файл accdb. В реальных условиях действующего предприятия — это неоправданное увеличение нагрузки на сеть, снизить которую довольно просто. Разделим базу данных на две части. Для этого:

1. Откройте базу данных, которую требуется разделить на две составляющие.
2. На вкладке главной ленты **Работа с базами данных** в разделе **Переместить данные** выберите значок  **База данных Access.**
3. Появится окно мастера, сообщающее о том, что операция может потребовать много времени и в принципе закончиться неудачно. Сделайте резервную копию исходной базы данных.
4. В диалоговом окне **Создание базы данных с таблицами** выберите папку, в которую требуется сохранить файл данных, затем введите имя файла в поле **Имя файла** и нажмите кнопку **Разделение.**
5. Появится сообщение об успешном выполнении операции.

Предупреждение

Если исходная база данных зашифрована, то, несмотря на это, новая база данных с таблицами будет открыта для всех пользователей. Добавьте для нее пароль сразу после разделения.

Не заблуждайтесь! То, что мы сделали — это не переход на платформу "клиент-сервер". Несомненно, скорости работы этот прием добавит нашему приложению. Компоненты интерфейса (формы, отчеты, меню) теперь хранятся на рабочей станции и не передаются по сети, но работа с данными по-прежнему — на уровне файлового сервера. Для выполнения любого запроса требуется практически полная передача "клиенту" всех необходимых ему таблиц и индексных файлов с "сервера".

Примечание

В нашем случае файл с данными получит имя Real Estate Часть I_be.accdb. Имя файла "клиента" останется прежним — Real Estate Часть I.accdb.

Поместите файл с данными (Real Estate Часть I_be.accdb) на самой мощной машине вашей локальной сети, а "клиента" растиражируйте по рабочим станциям. Следующий этап — настройка "клиента". Для этого:

1. Запустите на выполнение файл Real Estate Часть I.accdb ("клиент").
2. В области навигации откройте раздел **Таблицы**. У каждой таблицы в левом верхнем углу появилась стрелочка. Теперь все наши таблицы — связанные. Это означает, что они находятся не в этой базе данных. К тому же ссылка на действительное расположение таблиц ошибочна. Ведь мы переместили файл с данными. И не только его. "Клиент" тоже переехал!
3. Удалите все связанные таблицы. Для этого, удерживая клавишу <Shift>, сделайте щелчок мышью по первой и последней таблицам в списке таблиц.
4. Нажмите клавишу <Delete>.
5. Правильно ответьте на запрос системы для подтверждения удаления.
6. Перейдите на третью вкладку ленты — **Внешние данные**. В разделе **Импортировать и связать** выберите значок **Access.**
7. Появится окно **Внешние данные — база данных Access.**
8. Укажите полный путь к файлу Real Estate Часть I_be.accdb ("сервер"). Лучше всего воспользоваться для этого кнопкой **Обзор.**
9. В разделе окна с названием **Укажите когда и где сохранять данные в текущей базе данных** отметьте переключатель **Создать связанную таблицу для связи с источником данных**. Изменения, сделанные на "сервере", будут отображаться у "клиента", и наоборот.
10. Появится окно **Связь с таблицами**. Щелкните по кнопке **Выделить все** (рис. 4.5) и нажмите кнопку **OK.**

11. Все таблицы появятся в области переходов. Это связанные таблицы с правильными ссылками на "сервер".

Рис. 4.5. Выбор таблиц для назначения связей

Предупреждение

Если в области навигации у значка таблицы отсутствует стрелочка в левом верхнем углу, то это означает, что вы просто импортировали ее на рабочую станцию. Все изменения отныне останутся только у "клиента". Выполните набор инструкций 1—11 заново и более внимательно.

В завершение раздела — достоинства разделения данных и приложения MS Access 2010 в развернутом виде:

- пользовательский интерфейс работает значительно быстрее;
- у разработчика появляется возможность создавать временные таблицы-выборки на рабочей станции, не беспокоясь о конфликтах и блокировках;
- значительно упрощается установка новых версий приложения;
- файл с данными, расположенный на сервере, потерять теперь значительно сложнее.

4.8. Просмотр и изменение свойств документа MS Access 2010

Свойства документа — это метаданные (данные, которые описывают другие данные). Например, записи в таблице являются данными, а их число может служить примером метаданных. К свойствам документа MS Access относятся заголовок, имя автора, тема и ключевые слова, определяющие раздел или содержимое документа. Указав соответствующие значения для полей свойств документа, эти документы можно будет упорядочить и легко находить в дальнейшем.

Свойства документа делятся на следующие типы:

- стандартные (автор, название, тема и др.);
- автоматически обновляемые свойства (размер файла, дата создания или последнего изменения файла) и статистические сведения. По этим свойствам можно, например, найти все файлы, созданные после конкретной даты;
- пользовательские свойства. Имя пользовательского свойства можно выбрать из предлагаемого списка или определить самостоятельно;
- свойства для организации. Если в организации настроена область сведений о документе, то документы пользователя могут иметь определенные для его организации свойства.

Для просмотра и изменения свойств текущего документа сделайте следующее:

1. Откройте базу данных.
2. Нажмите кнопку **Файл**. Появится меню. Выберите в нем пункт **Сведения**.
3. В правом верхнем углу окна найдите ссылку **Просмотр и изменение свойств базы данных**.
4. Появится диалоговое окно **Свойства**. В нем пять вкладок (рис. 4.6).
5. Откройте вкладки для выбора свойств, которые требуется просмотреть или изменить.

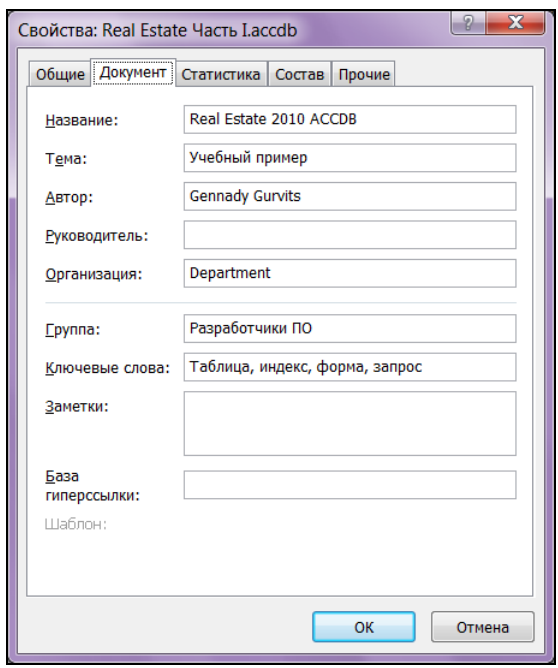


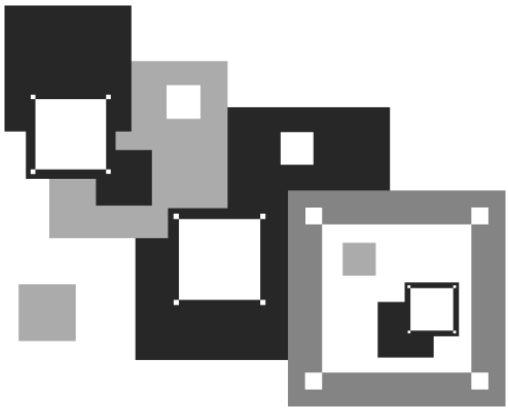
Рис. 4.6. Изменение свойств документа MS Access 2010

Для создания пользовательских свойств документа перейдите на вкладку **Прочие.** Этим свойствам можно назначать текстовые, числовые значения или значения даты/времени, а также значения "да" или "нет". Имя пользовательского свойства можно выбрать из предлагаемого списка или определить самостоятельно.

4.9. Импортирование объекта в свою базу данных

Проблемы, решение которых вы найдете в *частях II* и *III*, встретились автору в процессе работы над собственными приложениями. Не исключено, что эти решения заинтересуют вас. Используйте их в своих разработках. Любой объект (таблица, форма, отчет, запрос, модуль и т. д.) может быть импортирован во внешнюю базу данных. Для этого:

1. Откройте базу-источник в MS Access 2010.
2. Найдите нужный объект в области навигации.
3. Откройте третью вкладку ленты MS Access — **Внешние данные.**
4. В разделе **Экспорт** выберите пункт **Access**. Откроется диалоговое окно **Экспорт — База данных Access.**
5. Щелкните по кнопке **Обзор**. Найдите в нем файл базы-назначения.



ЧАСТЬ II

ДАЛЬНЕЙШЕЕ РАЗВИТИЕ ВАШЕГО ПРИЛОЖЕНИЯ

В которой вы научитесь делать то, что нужно, и так, как можно

ГЛАВА 5

Основные сведения о Visual Basic for Applications

Visual Basic for Applications (VBA) — это мощный инструмент разработки приложений. Как и другие средства, например, MS Visual C++, MS Visual C#, MS VB, MS Visual FoxPro, Borland Delphi, он предоставляет разработчику возможность создать полностью законченные программные продукты. VBA встроен во все приложения Microsoft Office, AutoCAD, CorelDRAW и множество других. В MS Office 2010 используется версия VBA с номером 7.0. Изучив VBA, вы сможете создавать свои программные комплексы не только в Microsoft Access. Необходимо лишь дополнительно освоить иерархию объектов выбранного приложения.

5.1. Среда Visual Basic for Applications

Создав базу данных Real Estate 2010, мы автоматически получили в свое распоряжение проект VBA с таким же именем. Запустите MS Access 2010, откройте базу данных Real Estate 2010, перейдите на вкладку **Работа с базами данных.** Выбор самой левой пиктограммы — **Visual Basic** — откроет окно (рис. 5.1). VBA имеет собственную среду разработки, в которой имеется меню, окна и другие элементы, использующиеся при работе с проектом. При переходе от MS Access к другому основному приложению вам практически не потребуется времени, чтобы научиться применять в нем VBA.

Наш проект VBA носит имя базы данных Real Estate и содержит формы, модули класса и модули кода. На рис. 5.1 представлены четыре основных окна среды VBA. Это:

- окно проекта (заголовок окна: **Project — sample**);
- окно текста (на рисунке заголовка окна нет, но хорошо видны две надписи: **General** и **StartMainMenu**);
- окно свойств (заголовок: **Properties — ModuleMain**);
- окно отладки (**Immediate**).

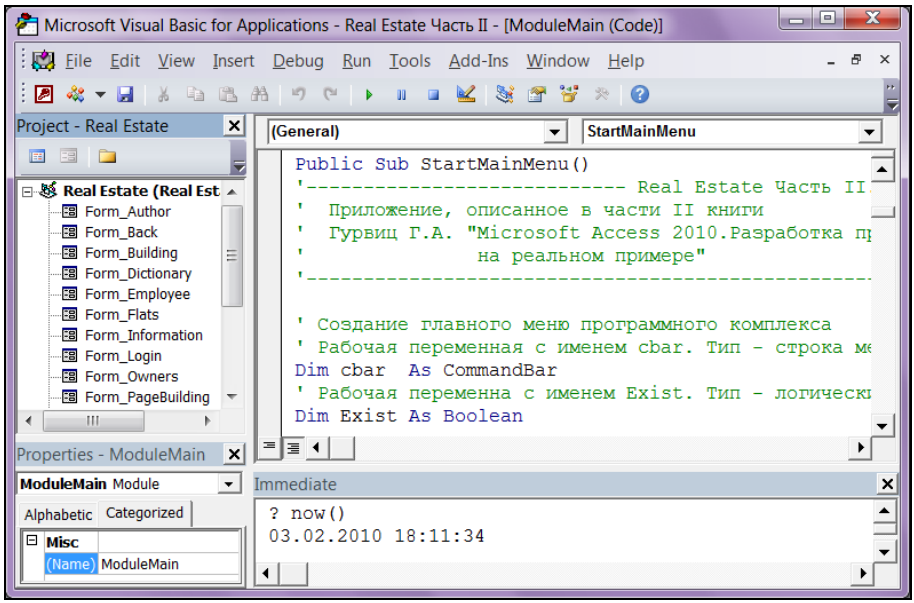


Рис. 5.1. Окно Microsoft Visual Basic for Applications

В окне проекта можно легко выбрать требуемый объект. Для его редактирования сделайте по объекту двойной щелчок мышью. В окне кода появится его текст, а в окне **Properties** — свойства объекта. Окно свойств предназначено для задания свойств объектов. Например, можно указать в окне свойств формы фон, заголовок, номер темы в файле справки и т. д. Всего свойств — несколько десятков. На первой вкладке свойства расположены в алфавитном порядке (**Alphabetic**). Выбор второй вкладки этого окна (**Categorized**) отсортирует свойства по категориям.

Окно редактирования кода предназначено для ввода текста процедур VBA. Код внутри модуля сгруппирован в отдельные разделы — процедуры. Редактор VBA может отображать текст в двух режимах: просмотр всего модуля или просмотр отдельной процедуры. Для переключения этих режимов применяются кнопки  (отдельная процедура) и  (все процедуры), расположенные в левом нижнем углу окна редактора кода.

В режиме просмотра всего модуля процедуры отделяются одна от другой горизонтальной чертой. Если этот разделитель отсутствует на экране, то вернуть его на место можно следующим способом:

1. В главном меню VBA выберите пункт меню **Tools**. Появится всплывающее меню.
2. Щелкните по строчке **Options**. Откроется одноименное диалоговое окно.
3. Поставьте флажок **Procedure separator**.

Два раскрывающихся списка (на рис. 5.1 в них отображены надписи **General** и **StartMainMenu**) предназначены для выбора объекта и его составляющих соответственно. Если в VBA идет работа с кодом обработки событий формы, то левый список содержит объект формы, а правый — перечень событий, допустимых для выбранного объекта.

Примечание

Выбор объекта и события инициирует создание в редакторе кода первой и последней строчки процедуры.

5.2. Интеллектуальные возможности редактора текстов

Редактор кода VBA наделен своими создателями способностью автоматически завершать написание составных частей строки текста (операторов, параметров, свойств). Он сам предлагает разработчику список компонентов, которые могут завершить введенную инструкцию или отобразить на экране сведения о процедуре или функции после введения имени (рис. 5.2).

Если подсказки для выбора свойств вам мешают, сделайте следующее:

1. В главном меню VBA выберите пункт меню **Tools**. Появится всплывающее меню.
2. Щелкните по строчке **Options**. Откроется одноименное диалоговое окно.
3. Снимите флажок **Auto Data Tips**.

Тем не менее, возможность вызова всплывающей подсказки у вас осталась даже при снятом флажке. Воспользуйтесь в нужный момент нажатием комбинации клавиш <Ctrl>+<I>.

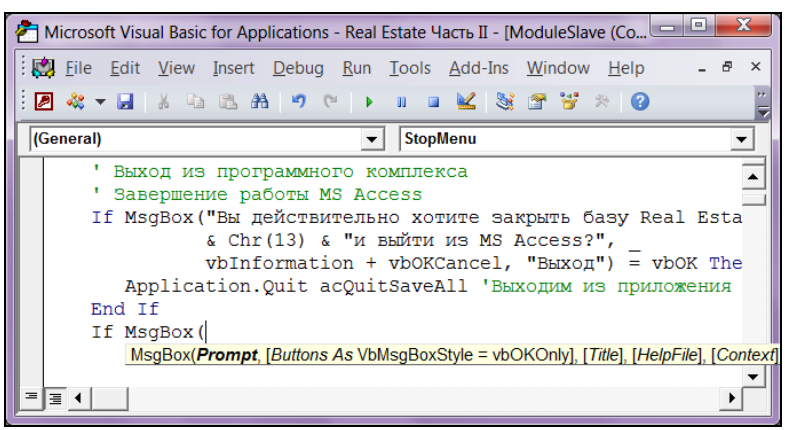


Рис. 5.2. Подсказка редактора кода VBA

Редактор кода выполняет автоматическую проверку синтаксиса введенной строки после нажатия клавиши <Enter>. Если строка выделяется красным цветом — ищите ошибку.

Еще одна возможность редактора кода VBA заключается в вызове нужного в данный момент раздела справки. Вы изучаете текст какого-либо модуля, и назначение определенного ключевого слова VBA вызывает затруднения. Расположите курсор на этом слове и нажмите клавишу <F1>. Появится окно со справкой.

5.3. Переменные, типы данных и константы

Переменная — это область памяти компьютера, имеющая имя. В Visual Basic for Applications переменные используются для хранения данных в оперативной памяти. После создания переменная указывает на одну и ту же область памяти до тех пор, пока не будет уничтожена. Разработчику не нужно знать, где находится эта область. Достаточно лишь сослаться на имя переменной. При выборе имени переменной следует руководствоваться правилом:

- имя должно начинаться с буквы;
- имя не может содержать пробел и точку, а также `@`, `&`, `!`, `$`, `#`;
- имя не должно быть длиннее 255 символов;
- не рекомендуется назначать имена, совпадающие с названием функций и методов VBA.

Перед началом работы с переменной ее нужно объявить. Сделать это можно при помощи операторов `Dim`, `Public`, `Private` и `Static`. В зависимости от используемых операторов и месте их появления в коде программы, переменной будет назначена область видимости и время жизни. В VBA существуют три области видимости переменной.

- **Переменная уровня проекта.** Описывается с помощью инструкции `Public`. Также называется *открытой*. Является доступной для всех процедур проекта VBA. Пример:

  ```
  Public Account As Integer
  ```

  Объявлена переменная с именем `Account` типа `Integer`. Ее значением может быть только целое число.

- **Переменная уровня модуля.** Описывается с помощью инструкции `Dim` или `Private`. Эти инструкции равнозначны. Является доступной только в том модуле, в котором описана. Все процедуры этого модуля ее "видят". Из других модулей проекта она не видна. Описание такой переменной необходимо разместить перед описанием всех процедур модуля. Пример:

  ```
  Privat LastName As String
  Dim LastName As String
  ```

Объявлена переменная с именем `LastName` (фамилия) типа `String`. Ее значением может быть только строка символов.

- **Переменная уровня процедуры.** Описывается с помощью инструкции `Dim` или `Static`. Также называется *локальной*. Является доступной только в той процедуре, в которой описана. Все другие процедуры как этого модуля, так и других модулей ее не "видят". Если используется конструкция `Static`, то переменная сохраняет свое значение, пока выполняются другие процедуры этого модуля. Если применяется конструкция `Dim`, то после завершения работы процедуры значение такой переменной всегда теряется. Пример:

```
Static Period As Long
Dim Period As long
```

Объявлена переменная с именем `Period` типа `Long`. Ее значением может быть только длинное целое.

Увидеть значение переменной в тот или иной период времени можно в *окне отладки*. Откройте окно VBA.

Для этого, находясь в главном окне MS Access 2010:

1. Выберите вкладку **Работа с базами данных**, в левой части которой расположена пиктограмма **Visual Basic**. Щелкните по ней.
2. Запустится Microsoft Visual Basic (рис. 5.3). Сделайте щелчок правой кнопкой мыши по пункту **Debug** главного меню этого окна.

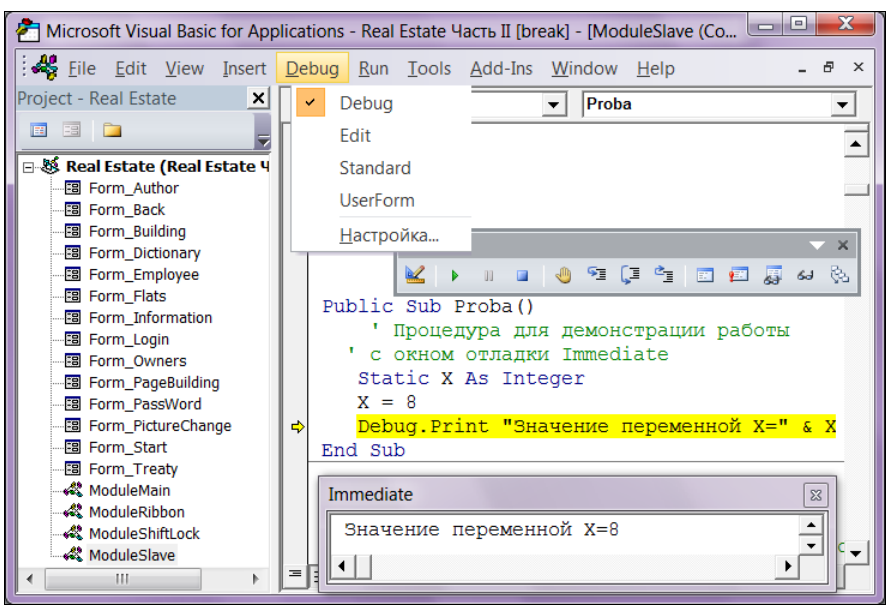


Рис. 5.3. Использование окна отладки в Visual Basic for Applications

3. Откроется контекстное меню из пяти пунктов. Выберите первый — **Debug**.
4. Появится панель инструментов отладчика.
5. Щелкните по пиктограмме  **Immediate Window (Ctrl+G)**. В появившееся окно **Immediate** отныне будут выводиться все значения, которые вы укажете после фразы `Debug.Print`.

Выполним отладку процедуры `Proba`, которая находится в модуле `ModuleSlave`. Выделите этот модуль в окне состава проекта и найдите текст процедуры `Proba`. Щелчок мышью по пиктограмме  **Step Info (F8)** запустит на выполнение отмеченную строчку кода. Если это `Debug.Print`, то в окне **Immediate** появится результат.

VBA дает возможность описывать и использовать переменные различных типов. Выбор типа переменной основан на требованиях разрабатываемого приложения. В VBA имеются строки, числа, даты, объекты, логические значения, а также общий тип данных — `Variant`. Он применяется по умолчанию и может содержать данные любого типа, кроме строк фиксированной длины.

В табл. 5.1 приведены основные типы данных VBA. Переменные `Byte`, `Integer` и `Long` не могут иметь дробных значений. Переменные типа `Byte` не могут иметь отрицательных значений. Тип переменных `Currency` (денежный) предпочтительнее использовать в валютных операциях. Скорость работы с этим типом выше, чем с `Single` или `Double`.

Таблица 5.1. *Основные типы данных в Visual Basic for Applications*

Категория	Тип данных	Размер	Диапазон
Числовые типы	`Byte`	1 байт	От 0 до 255
	`Integer`	2 байта	От –32 768 до 32 767
	`Long`	4 байта	От 2 147 483 648 до 2 147 483 647
	`Single`	4 байта	Для отрицательных чисел: от -3.402823×10^{38} до -1.401298×10^{-45} Для положительных чисел: от 1.401298×10^{-45} до 3.402823×10^{38}
	`Double`	8 байтов	Для отрицательных чисел: от $-1.79769313486232\times10^{308}$ до $-4.94065645841247\times10^{-324}$ Для положительных чисел: от $4.94065645841247\times10^{-324}$ до $1.79769313486232\times10^{308}$
	`Currency`	8 байтов	От –922 337 203 685 477.5808 до 922 337 203 685 477.5807

Таблица 5.1 *(окончание)*

Категория	Тип данных	Размер	Диапазон
Логический тип	`Boolean`	2 байта	`False` или `True`. Если используется 0, то он интерпретируется как `False`. Любое другое значение — `True`. При обратном преобразовании `True` рассматривается как –1
Тип даты	`Date`	8 байтов	От 1 января 100 года до 31 декабря 9999 года
Тип объект	`Object`	4 байта	Ссылка на объект
Строковые типы	`String`	Длина строки плюс 10 байтов	Строки переменной и фиксированной длины

Примечание

Большинство из рассматриваемых в таблице типов переменных VBA находится в полном соответствии с типами полей базы данных MS Access 2010 (см. табл. 2.11).

Для обязательного объявления всех переменных в начало модуля надо поместить директиву:

```
Option Explicit
```

Использование этой директивы не допускает возможности работы с необъявленными переменными. Если этого не сделать, то ничего страшного не произойдет, но не забывайте, что скупой платит дважды, а ленивый — трижды. У вас на клавиатуре есть несколько интересных клавиш. Одна из них с буквой "С". Если в имени переменной один раз укажете русскую букву, а другой — латинскую, то правильной работы программного комплекса можете вообще не добиться.

Константа — это область памяти компьютера, имеющая имя. В Visual Basic for Applications константы, как и переменные, используются для хранения данных в оперативной памяти. Различие в том, что изменить значение константы нельзя. В VBA можно как описать собственные константы, так и применять стандартные, которых в языке великое множество. При описании константы необходимо сразу присвоить ей значение, например:

```
Public Const Pi As Double=3.141592653589793
```

Описав константу один раз, можно использовать ее в разных выражениях столько раз, сколько это требуется. Область видимости констант VBA регламентируется теми же правилами, что и у переменных.

5.4. Стандартные константы на примере функции *MsgBox()*

В подавляющем своем большинстве стандартные константы Visual Basic for Applications — это числа, использующиеся в качестве аргументов функций или методов. Для примера рассмотрим функцию `MsgBox()` — самую часто используемую функцию VBA. Для работы с ней применяется 21 константа. В листинге 5.1 приведен текст процедуры завершения работы программного комплекса Real Estate, в которой используется эта функция.

Листинг 5.1. Текст процедуры завершения работы программного комплекса

```
Public Sub StopMenu()
' Выход из программного комплекса
' Завершение работы MS Access
If MsgBox("Вы действительно хотите закрыть базу Real Estate " _
   & Chr(13) & "и выйти из MS Access?", _
   VbInformation+VbOKCancel+VbDefaulButton1, "Выход") = VbOK Then
Application.Quit acQuitSaveAll   ' Выходим из приложения
End If
End Sub
```

Текст, следующий за символом ' до конца строки, представляет собой *комментарий* и игнорируется транслятором. Нужен исключительно самому разработчику или продолжателям его дела для пояснения смысла написанного кода. Комментарий может начинаться и не с самого начала строки.

Применение символов пробела и знака подчеркивания (_) означает, что следующая строка кода является продолжением предыдущей.

Обратите внимание на константы VBA. Их в этом примере используется четыре, причем значения трех складываются:

```
VbInformation + VbOKCancel + VbDefaulButton1
VbOK
```

Visual Basic for Applications "знает", что:

```
vbInformation=64    ' используется значок информации (табл. 5.3)
vbOKCancel=1        ' отображаются кнопки OK и Отмена (табл. 5.2)
VbDefaulButton1=0   ' номер выделенной (OK) кнопки (табл. 5.4)
```

И если нажата кнопка **OK** (рис. 5.4), то значение `VbOK=1`.

Текст вызова функции `MsgBox()` можно написать короче (листинг 5.2), указав в коде непосредственно цифры.

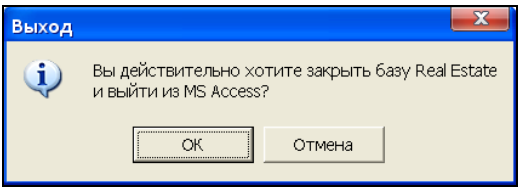


Рис. 5.4. Окно **Выход**

Листинг 5.2. Текст процедуры без констант VBA

```
If MsgBox("Вы действительно хотите закрыть базу Real Estate " _
       & Chr(13) & "и выйти из MS Access?", _
       65, "Выход") = 1 Then
       Application.Quit acQuitSaveAll 'Выходим из приложения
End If
```

Однако это не следует делать по двум причинам:

- имена констант значительно улучшают восприятие текста программы;
- создатели VBA не гарантируют, что значения числовых аргументов в будущих версиях продукта не изменятся, в отличие от имен.

В табл. 5.2 приведены константы, которые можно указать в качестве аргументов функции `MsgBox()` для определения состава кнопок в диалоговом окне. В табл. 5.3 приведены константы, определяющие отображаемые в диалоговом окне значки. В табл. 5.4 представлены константы, определяющие основную кнопку в диалоговом окне, в табл. 5.5 — константы, соответствующие выбранной кнопке.

Таблица 5.2. *Значения констант для выбора состава кнопок*

Константа	Значение	Отображение кнопок
`VbOkOnly`	0	OK
`VbOkCancel`	1	OK, Отмена
`VbAbortRetryIgnore`	2	Прервать, Повтор, Пропустить
`VbYesNoCancel`	3	Да, Нет, Отмена
`VbYesNo`	4	Да, Нет
`VbRetryCancel`	5	Повтор, Отмена

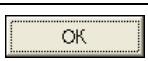
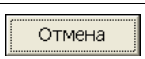
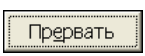
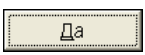
Таблица 5.3. *Значения констант для выбора значков*

Константа	Значение	Значок
VbCritical	16	
VbQuestion	32	
VbExclamation	48	
VbInformation	64	

Таблица 5.4. *Значения констант для определения основной кнопки*

Константа	Значение	Номер основной кнопки
VbDefaultButton1	0	1
VbDefaultButton2	256	2
VbDefaultButton3	512	3
VbDefaultButton4	768	4

Таблица 5.5. *Значения констант для идентификации нажатой кнопки*

Константа	Значение	Отображение кнопок
VbOk	1	
VbCancel	2	
VbAbort	3	
VbRetry	4	
VbIgnore	5	
VbYes	6	
VbNo	7	

5.5. Стандартные функции и выражения

Функции VBA предназначены для возврата значений в точку вызова. В VBA имеется более 150 различных стандартных функций. Особенно часто используемые из них приведены в табл. 3.5 и 3.6. Более подробные сведения вы можете получить из справочной системы MS Access 2010 и справочника Access VBA 7.0.

В листинге 5.3 приведен текст процедуры, которая использует стандартную функцию VBA для построения таблицы значений тригонометрической функции синус. Значение синуса аргумента вычисляется по алгоритму, реализованному создателями библиотеки стандартных функций VBA. Это — разложение в ряд Тейлора с удержанием количества членов ряда, достаточного для достижения заданной точности. Рядовой пользователь может даже ничего и не слышать об этой премудрости. Тем не менее желаемого результата он достигнет (листинг 5.4).

Листинг 5.3. Таблица значений функции синуса

```
Public Sub DemoSin()
' Процедура для демонстрации работы
' со стандартной функцией
   Static X As Single
   Static y As Single
   For X = 0 To 1 Step 0.1
       y = Sin(X)
       Debug.Print "X=" & X & " Sin(X)=" & Y
   Next X
End Sub
```

Листинг 5.4. Результаты работы процедуры `DemoSin`

```
X=0           Sin(X)=0
X=0,1         Sin(X)=9,983342E-02
X=0,2         Sin(X)=0,1986693
X=0,3         Sin(X)=0,2955202
X=0,4         Sin(X)=0,3894183
X=0,5         Sin(X)=0,4794255
X=0,6         Sin(X)=0,5646425
X=0,7         Sin(X)=0,6442177
X=0,8         Sin(X)=0,7173561
X=0,9         Sin(X)=0,783327
```

Выражение VBA представляет собой комбинацию ключевых слов, операторов, переменных и констант. Результат выражения — число, строка или логическое значение.

Ключевое слово — это набор символов, распознаваемый как элемент языка программирования VBA. Например: `Case`, `If`, `Sub`, `Step` и т. д.

Оператор — это набор символов, предназначенный для объединения простых выражений в более сложные. Различают арифметические операторы, операторы сравнения и логические операторы (см. табл. 3.2—3.4).

Перед выражениями, определяющими вычисляемые элементы управления, всегда расположен знак присваивания (=). Некоторые выражения дают логические результаты `Истина` или `Ложь`.

5.6. Массивы

В отличие от переменной, которая содержит один объект (строку, число, дату и т. д.), массив может содержать целый набор связанных между собой данных. Эти данные имеют общее имя. Обратиться к конкретному значению массива позволяет его номер. Например, можно применить переменную `GrossSalary` для хранения величины заработной платы сотрудника:

```
GrossSalary = 19800
```

Для хранения сведений о зарплате служащих всего отдела из 10 человек можно объявить десять переменных (`GrossSalary1`, `GrossSalary2`, `GrossSalary3`, ...), а можно — один массив из десяти элементов:

```
Dim GrossSalary(10) As Integer
```

Все значения в массиве должны обязательно принадлежать к одному типу данных. В скобках после имени указывают количество элементов. В этом примере объявлен *одномерный массив*. Нумерация элементов начинается с нуля (по умолчанию). Первый элемент массива может иметь номер, отличный от нуля. Если поместить в начало модуля оператор `Option Base 1`, то нумерация элементов массива начнется с единицы. В операторе `Dim` можно явно задать нижнюю границу. Это значение может быть любым при условии, что оно меньше верхней границы:

```
Dim GrossSalary(1 To 10) As Integer
GrossSalary(5)= 19800    ' Зарплата пятого работника
```

Массив может быть не только одномерным. В VBA применяются двумерные, трехмерные массивы и массивы с большим числом размерностей. Допускается использование 60 индексов. В следующем примере объявлен двумерный массив (матрица) с именем `Matric` из ста целых чисел: десять строк и десять столбцов. Нумерация элементов начинается с единицы.

```
Dim Matric(1 To 10, 1 To 10) As Integer
```

Часто размер массива заранее неизвестен. В этом случае в операторе `Dim` после имени массива ставят пустые скобки и описание типа, а количество элементов и диапазон не задаются. Перед использованием массива выполняется оператор

`ReDim`. В листинге 5.5 приведен пример работы с массивом, размер которого заранее неизвестен.

Листинг 5.5. Пример работы с массивом

```
Dim Sample() As Variant
' Задаем размерность: двумерный, 5 строк (0:4) и 6 столбцов (0:5)
' Нумерация элементов начинается с нуля
ReDim Sample(4,5)
' Присваиваем нужные значения
Sample(0,2)= 3
Sample(1,3)= 8
' Вычисляем результат
Summary= Sample(0,2)+ Sample(1,3)
' Изменение размерности. Теперь — одномерный (1:11)
ReDim Sample(1 To Summary)
' Присваиваем нужные значения
Sample(3)= "Иванов"
```

При выполнении оператора `ReDim` все значения, хранящиеся в массиве, теряются. Однако VBA предоставляет возможность изменить размеры массива в сторону увеличения без потери данных. Увеличим размер массива `Sample` на один элемент до (1:12). Для этого применяется ключевое слово `Preserve`. Стандартная функция `Ubound` возвращает число элементов массива.

```
ReDim Preserve Sample(Ubound(Sample) + 1)
```

5.7. Инструкции Visual Basic for Applications

Значения терминов, применяемых в компьютерной науке, на сегодняшний день, к сожалению, окончательно не установлены. Один и тот же термин может использоваться для обозначения разных объектов. В *разд. 3.9* шла речь об операторах сравнения, арифметических и логических. Инструкции VBA иногда также называют *операторами*.

5.7.1. Оператор присваивания

Оператор присваивания присваивает значение выражения переменной, константе или свойству объекта. Оператор присваивания обязательно содержит в своем составе знак равенства. В результате выполнения двух следующих операторов переменная `Sign` получит значение 7.

```
Sign = 4
Sign = Sign + 3
```

Для присваивания переменной ссылки на объект в операторе присваивания используется служебное слово `Set`. В следующем примере (листинг 5.6) переменной `cBar` присваивается ссылка на новое головное меню программного комплекса, созданное методом `Add` (добавить) коллекции `CommandBars` (панели команд).

Листинг 5.6. Создание панели инструментов

```
Set cbar = CommandBars.Add(Name:="MainMenu", _
       Position:=msoBarTop, MenuBar:=True, Temporary:=False)
' MainMenu — имя главного меню программного комплекса
' Значение константы msoBarTop=1. Меню появится вверху окна
' Если MenuBar:=True  — будет создана строка меню
' Если MenuBar:=False — будет создана панель инструментов
' Temporary:=False — строка меню не временная и
' останется после завершения работы
```

5.7.2. Оператор *With*

Инструкция `With` освобождает разработчика от утомительной необходимости использовать большое количество повторений имени одного и того же объекта при работе с его свойствами и методами. Второе назначение инструкции `With` — структурирование года. Код становится более понятным.

С использованием `With`:

```
With oExcel.Selection.Borders(xlEdgeLeft)
           .LineStyle = xlContinuous
           .Weight = xlThin
           .ColorIndex = xlAutomatic
EndWith
```

Без использования `With`:

```
oExcel.Selection.Borders(xlEdgeLeft).LineStyle = xlContinuous
oExcel.Selection.Borders(xlEdgeLeft).Weight = xlThin
oExcel.Selection.Borders(xlEdgeLeft).ColorIndex = xlAutomatic
```

5.7.3. Управление выполнением программы

Программа на VBA не является монолитным набором команд, каждый раз производящим одни и те же вычисления. Ее поведение корректируется в зависимости от входных данных, внешних условий (клавиатура, мышь и др.) и аварийных ситуаций. VBA предоставляет в распоряжение разработчика три инструкции, позволяющие управлять ходом выполнения программы. Они носят название *управляющих конструкций ветвления* и дают возможность проверить некоторое условие и, в зависимости от результата проверки, выполнить ту или иную группу операторов.

Инструкция `If...Then` вычисляет условие, и если оно равно `True`, то выполняется оператор, следующий за ключевым словом `Then`. Синтаксис оператора `If` допускает запись как в одну, так и в несколько строк. Для выполнения одного оператора, следующего за конструкцией `If...Then`, лучше использовать запись в одну строку:

```
If CountAvto=1 Then NewStatement= NewStatement+1
```

Хотя можно применить и другой вариант. Читать такой код легче, но длина его больше:

```
If CountAvto=1 Then
   NewStatement= NewStatement+1
EndIf
```

Инструкция `If...Then...Else` выполняет блок операторов, следующий за ключевым словом `Then`, если значение условия после `If` равно `True`. В противном случае выполняется блок операторов, стоящих после ключевого слова `Else` (листинг 5.7).

Листинг 5.7. Инструкция `If...Then...Else`

```
If QtyToHold > 0 Then
   TransactionTypeHold = Hold_TransactionType
   Quantity = QtyToHold
   AddHold = EditTransactionOrderID
   Quantity = QtyRequested
Else
   QuantityGranted = 0
   Account = Reseller
End If
```

Инструкция `Select Case` имеет определенные преимущества перед несколькими операторами `If...Then...Else`. Она облегчает чтение и сопровождение кода. Оператор `Select Case` вычисляет выражение, расположенное вверху всей структуры. Результат вычисления сравнивается с несколькими значениями, и если совпадет с одним из них, то выполняется соответствующий блок операторов (листинг 5.8).

Листинг 5.8. Инструкция `Select Case`

```
Select Case DomainFunction
    Case DLookup
         ' Если значение переменной DomainFunction равно
         ' значению переменной DLookup
         DomainFunctionWrapper = ExprDomainCriteria
         QuantityGranted = 0
         Account = Reseller
```

```
    Case 14, 17
         ' Если значение переменной DomainFunction равно 14 или 17
         DomainFunctionWrapper = DomainCriteria
         Account = Seller
    Case 4 To 6
         ' Если значение переменной DomainFunction от 4 до 6
         DomainFunctionWrapper = DSumExpr
    Case Else
         ' Неожиданный аргумент функции домена
         Debug.Assert False
End Select
```

Если выражению соответствуют значения более чем одного оператора `Case`, то выполняется только первый блок операторов.

Если выражению не соответствует ни одно из значений оператора `Case`, то выполняется блок операторов, стоящих после `Else`.

5.7.4. Операторы цикла

Операторы цикла предназначены для повторного выполнения одной и более строк кода. В VBA имеется богатый выбор средств организации циклов, которые можно разделить на две основные группы:

- циклы с перечислением `For...Next`;
- циклы с условием `Do...Loop`.

Оператор `For...Next` обеспечивает выполнение группы инструкций указанное число раз, пока переменная цикла изменяется от начального значения до конечного с заданным шагом. Если шаг не указан, то он считается равным единице. В следующем примере (листинг 5.9) трехмерный массив заполняется случайными числами. Используются три вложенных цикла.

Листинг 5.9. Вложенные циклы

```
Option Base 1
Dim SampleArray (10,10,10) As Single
Dim i As Integer, j As Integer, k As Integer
Randomize
For i=1 To 10
    For j=1 To 10
        For k=1 To 10
            SampleArray(i,j,k) = Rnd()
        Next k
    Next j
Next i
```

Эту же задачу можно решить, применив вместо трех вложенных циклов со счетчиками всего один `For Each...Next` (листинг 5.10).

Листинг 5.10. Цикл `For Each...Next`

```
Dim SampleArray (10,10,10) As Single
Dim ijk As Variant
Randomize
For Each ijk In SampleArray
    ' Для каждого элемента массива
    ijk = Rnd()
Next
```

В следующем примере (листинг 5.11) вычисляется сумма четных элементов одномерного массива при помощи `For...To...Step...Next`.

Листинг 5.11. Вычисление суммы четных элементов

```
Dim DemoArray(20) As Single
Dim i As Integer, Summa As Single
' Здесь массив должен быть заполнен
Summa = 0
For i = 2 To UBound(DemoArray) Step 2
    Summa = Summa + DemoArray(i)
Next
```

Как правило, выполнение цикла завершается, когда достигнуто условие прекращения его работы. Однако в некоторых случаях необходимо прекратить выполнение цикла досрочно, избежав выполнения лишних операторов (листинг 5.12). Например, если главное меню программного комплекса обнаружено в коллекции `CommandBars` (панели команд), то дальнейший перебор следует прервать. Для этого применяется связка ключевых слов `Exit For`.

Листинг 5.12. Досрочное завершение цикла

```
For Each cbar In CommandBars
    If cbar.Name = "MainMenu" Then
       ' Главное меню с именем MainMenu существует.
       ' Выйти из цикла
       Exit For
    End If
Next cbar
```

Оператор `Do` повторяет выполнение группы инструкций, пока условие имеет значение `True` (случай `While`) или пока оно не примет значение `True` (случай `Until`).

В первом случае используется следующий синтаксис:

```
Do While <условие>
  <операторVisualBasicForAppication1>
  Exit Do
  <операторVisualBasicForAppication2>
  ...
Loop
```

Во втором случае синтаксис оператора имеет вид:

```
Do Until <условие>
  <операторVisualBasicForAppication1>
  Exit DO
  <операторVisualBasicForAppication2>
  ...
Loop
```

В любом месте управляющей структуры `Do` может быть размещено любое число инструкций `Exit Do`, которые обеспечивают альтернативные возможности выхода из цикла.

В следующем примере (листинг 5.13) рассмотрено применение еще одного оператора цикла, реализованного в VBA. Это оператор `While...Wend`. Генерируется случайное число в диапазоне 1—10 до тех пор, пока не выпадет значение 7. Переменная `attempt` после завершения цикла будет содержать количество попыток.

Листинг 5.13. Игра в рулетку

```
Dim rank As Integer        ' Количество очков
Dim attempt As Integer     ' Попытка
Randomize
rank = Int(10 * Rnd()) + 1
attempt = 1
While rank <> 7
  attempt = attempt + 1
  ' Генерация случайного числа от 1 до 10
  rank = Int(10 * Rnd()) + 1
Wend
```

5.7.5. Оператор безусловного перехода

Программист, работающий на VBA, может задать переход на любую строку внутри процедуры без проверки каких-либо условий. Этой строке надо присвоить *метку*. Метка должна обязательно начинаться с буквы и заканчиваться двоеточием (листинг 5.14). Для демонстрации применения метки переработан предыдущий пример.

Листинг 5.14. Применение метки

```
Dim rank As Integer        ' Количество очков
Dim attempt As Integer     ' Попытка
Randomize
For i = 1 To 10
  ' Генерация случайного числа от 1 до 10
  rank = Int(10 * Rnd()) + 1
  attempt = attempt + 1
  If rank = 7 Then GoTo LabelEnd
Next
LabelEnd: < очереднойОператорVisualBasicForAppications>
```

5.8. Процедуры и функции

Процедуры и функции VBA представляют собой законченные фрагменты программного кода. Текст процедуры VBA заключается между операторами `Sub` и `End Sub`:

```
Sub <имяПроцедуры> (<Аргумент1>, <Аргумент2>, ...)
  <операторVisualBasicForAppications1>
  <операторVisualBasicForAppications2>
   ...
End Sub
```

Этот фрагмент может быть использован в программном комплексе любое число раз. Каждый раз заново писать его в нужном месте нет необходимости. Для того чтобы воспользоваться написанной процедурой, надо ее просто "вызвать". Процедура VBA без списка аргументов может быть вызвана с помощью команд меню, кнопок панелей инструментов, клавиш быстрого вызова или из другой процедуры. Кроме этого, процедуру без параметров можно связать с выполнением любого события: открытие формы, щелчок мыши и т. д. Процедуру с параметрами можно вызвать только из другой процедуры или функции.

Существуют два способа вызова процедуры. В первом случае указывается имя вызываемой процедуры и список значений аргументов без скобок. Этот способ применяется, если вызывающая процедура расположена в том же модуле, что и вызываемая. Пример вызова процедуры `Account` с передачей ей двух аргументов (константы и выражения):

```
Account 453, CountPages +1
```

Второй способ заключается в использовании оператора VBA `Call`:

```
Call Account(453, CountPages +1)
```

VBA допускает две разновидности передачи переменных процедуре или функции: по ссылке и по значению. При передаче по ссылке фактический и формаль-

ный параметры отождествляются. Это означает, что любое изменение формального параметра в процедуре повлечет за собой изменение фактического параметра. Другими словами, значение переменной, переданной в процедуру, может быть там изменено. Для этой цели применяется описатель `ByRef`. Он же подразумевается по умолчанию. При передаче по значению в процедуру передается само значение и, следовательно, величина переменной в процедуре изменена быть не может. Такую форму передачи обеспечивает описатель `ByVal`.

Рассмотрим вызов процедуры и передачу параметров на примере. Текст вызывающей процедуры `ProcedureOptionsMain` приведен в листинге 5.15.

Листинг 5.15. Вызывающая процедура

```
Sub ProcedureOptionsMain()
   Dim A As Integer
   Dim B As Integer
   Dim C As Integer
   A = 1000
   B = 2000
   C = 3000
   Call ProcedureOptionsSlave(A,B,C)
   Debug.Print A
   Debug.Print B
   Debug.Print C
End Sub
```

В процедуре описаны три переменные. Они получают значения, а затем передаются в качестве фактических параметров вызываемой процедуре. Текст вызываемой процедуры `ProcedureOptionsSlave` приведен в листинге 5.16.

Листинг 5.16. Вызываемая процедура

```
Sub ProcedureOptionsSlave(ByRef X, ByVal Y, Z)
   X = X + 1
   Y = Y + 1
   Z = Z + 1
   Debug.Print X
   Debug.Print Y
   Debug.Print Z
End Sub
```

Вызываемая процедура содержит три формальных параметра. Все они описаны по-разному. Значения всех трех в теле процедуры увеличиваются на единицу. В итоге в окне отладки появится шесть чисел. Сначала значения, полученные в

`ProcedureOptionsSlave`: 1001, 2001 и 3001. А потом — значения, которые были возвращены в главную процедуру: 1001, 2000 и 3001. Обратите внимание на тот факт, что значение переменной, переданной `ByVal` (по значению), не изменилось.

Имя функции VBA, в отличие от имени процедуры VBA, выступает в качестве переменной и используется для возвращения значения в точку вызова. Текст функции заключается между операторами `Function` и `End Function`.

```
Function <имяФункции> (<Аргумент1>, <Аргумент2>, ...)
  <операторVisualBasicForAppications1>
  <операторVisualBasicForAppications2>
   ...
  <ИмяФункции> = <ВозвращаемоеЗначение>
End Function
```

Функцию можно вызвать с помощью отдельного оператора VBA, имеющего следующий синтаксис:

```
Call <имяФункции> (<Список_фактических_значений>)
```

Для вызова функции применяется еще один способ: напишите ее имя со списком фактических значений аргументов непосредственно в нужном выражении VBA или в формуле в вычисляемых полях форм, запросов и отчетов MS Access 2010.

Рассмотрим текст функции `CurrentPath` из модуля `ModuleSlave`. Она предназначена для возвращения в точку вызова полного пути к папке, из которой запущен на выполнение программный комплекс (листинг 5.17). Функция используется в форме `Login` для поиска пути к папке с фотографиями работников.

Листинг 5.17. Определение полного пути к папке

```
Public Function CurrentPath() As String
  ' Функция для получения полного пути
  ' к папке, в которой находится наша база данных
  Dim NameBase As String
  Dim NameBaseShort As String
  Dim CountLetter As Integer
  ' Полное имя файла, в котором находится наша база данных
  NameBase = CurrentDb.NAME

  ' Имя файла без пути к нему возвращает
  ' Dir() — стандартная функция VBA
  NameBaseShort = Dir(NameBase)

  ' Количество символов, описывающих путь к файлу,
  ' без символа \
  CountLetter = Len(NameBase) — Len(NameBaseShort) — 1
```

```
  Debug.Print "Полный путь: " & NameBase
  Debug.Print "Длина полного пути: " & Len(NameBase) & " символа"
  Debug.Print "Короткий путь: " & NameBaseShort
  Debug.Print "Длина пути: " & Len(NameBaseShort) & " символов"

  ' Отделяем слева путь к файлу из CountLetter символов
  ' Left() — стандартная функция VBA
  CurrentPath = Left(NameBase, CountLetter)
End Function
```

В начале кода описаны три рабочие переменные: `NameBase`, `NameBaseShort` и `CountLetter`. Сначала определяется полное имя файла, затем краткое без пути к файлу, длина этих имен и, наконец, разность имен. Значение полученной строки символов получает сама функция `CurrentPath` в предпоследней строчке текста. Четыре оператора `Debug.Print` позволяют контролировать правильность работы кода функции при отладке. Откройте текст модуля `ModuleSlave` и найдите или наберите текст функции `CurrentPath`. Сделайте щелчок правой кнопкой мыши по пункту **Debug** главного меню окна Visual Basic. Откроется контекстное меню из пяти пунктов. Выберите первый — **Debug**. Появится панель инструментов отладчика. Щелкните по пиктограмме  **Immediate Window (Ctrl+G)**. В появившемся окне **Immediate** наберите:

```
Debug.Print CurrentPath
```

или

```
? CurrentPath
```

В окне **Immediate** появится отладочная информация (рис. 5.5).

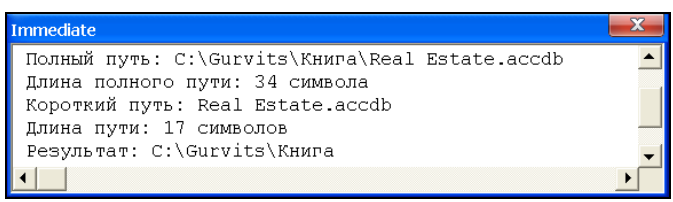


Рис. 5.5. Результат работы функции `CurrentPath`

Окно отладки может быть использовано не только для определения значений переменных, но и для их изменения. Например, для ввода нового значения переменной `CountLetter` можно ввести:

```
CountLetter = 32
```

Любой оператор, набираемый в окне **Immediate**, должен помещаться в одной строке. Нельзя ввести инструкцию из нескольких строк, но это ограничение лег-

ко обойти. Разделите инструкции двоеточиями. Такая форма записи применялась еще на первых "персоналках".

```
For CountLetter=1 To 5: Debug.Print CountLetter: Next CountLetter
```

Отладочные операторы `Debug.Print` можно оставить в готовой программе. Вывод в окно **Immediate**, которое отсутствует на экране дисплея, лишь незначительно замедлит работу программного комплекса. Если же снижение производительности вовсе не желательно, то вместо довольно утомительной процедуры установки символов комментария в начало каждой отладочной строки поместите в начале процедурного модуля описание глобальной константы:

```
Public Const Debugging = True
```

А все отладочные операторы разместите между `If` и `End If`:

```
If Debugging Then
   Debug.Print "Полный путь: " & NameBase
   Debug.Print "Длина полного пути: " & Len(NameBase) & " символа"
   Debug.Print "Короткий путь: " & NameBaseShort
   Debug.Print "Длина пути: " & Len(NameBaseShort) & " символов"
End If
```

Перед сдачей программного комплекса заказчику просто замените значение этой константы с `True` на `False`.

ГЛАВА 6

Использование SQL Access

Язык SQL (Structured Query Language) — один из языков, появившихся в результате разработки реляционной модели данных. В настоящее время этот язык получил очень широкое распространение и фактически превратился в стандартный язык реляционных баз данных. SQL поддерживают сотни СУБД различных типов, которые созданы для самых разнообразных платформ от персоналок до мэйнфреймов.

Понятно, что базовый стандарт не может предусмотреть все потребности пользователей, поэтому многие фирмы-производители СУБД предлагают собственные и часто непереносимые расширения SQL.

Существуют также специальные процедурные расширения SQL-диалектов. Они похожи на обычные процедурные языки, т. е. у них есть и нормальные переменные, и метки, и циклы, и все прочее, а также полностью поддерживается синтаксис SQL. Жесткого стандарта на процедурные расширения нет, поэтому фирмы-изготовители СУБД определяют синтаксис так, как считают нужным.

6.1. Назначение языка SQL

Язык SQL дает возможность разработчику баз данных:

- создавать базы данных и таблицы, полностью описывая их структуру;
- выполнять манипулирование данными, используя операции добавления, удаления и модификации;
- создавать и выполнять простые и сложные запросы к базе данных.

Язык SQL, обладая перечисленными выше серьезными возможностями, достаточно прост и легкодоступен для изучения. В его состав входят два основных компонента:

- язык DML (Data Manipulation Language) — используется для выборки данных и их обновления;

- язык DDL (Data Definition Language) — используется для описания структур баз данных и управления доступом к ним.

Диалект языка SQL для MS Access относится к непроцедурным языкам. Он не требует указания методов доступа к данным и поддерживает свободный формат записи операторов. У него нет ни переменных, ни меток, ни циклов, ни всего прочего, с чем привык работать нормальный программист. Надо четко представлять себе, что SQL оговаривает способ передачи данных в клиентскую программу, но никак не оговаривает то, как эти данные должны в клиентской программе обрабатываться и представляться пользователю.

6.2. Запросы на выборку. Оператор *SELECT*

Структура оператора `SELECT` отличается от остальных операторов языка DML (`INSERT`, `UPDATE` и `DELETE`) несколько большей сложностью, а сам оператор — частотой употребления, поэтому вначале рассмотрим возможности именно этого оператора. Для построения примеров с помощью операторов SQL будем использовать данные таблиц учебного примера Real Estate.

Результатом работы оператора `SELECT` в MS Access 2010 является выборка необходимых строк из базы данных и размещение их в динамическом объекте набора записей. Синтаксис оператора довольно прост:

```
SELECT <Список_столбцов>
 FROM <Список_таблиц>
 [WHERE <Условие_выборки>]
 [GROUP BY <Список_полей_группы>]
 [HAVING <Условие>]
 [ORDER BY <Список_атрибутов>]
```

Прописными буквами выделены ключевые слова. Курсивом обозначены места, в которых следует ввести имена своих таблиц, столбцов, значений и т. д. В квадратных скобках размещены необязательные элементы.

6.2.1. Предложение *FROM*

Ключевое слово `FROM` определяет имена таблиц, которые являются источником записей для создаваемого запроса. Для выборки всех столбцов таблицы улиц `tblStreet` базы данных Real Estate необходимо написать:

```
SELECT Street, Name, Sign, First
FROM tblStreet;
```

Примечание

В *частях II* и *III* все таблицы, созданные ранее, получили в своем названии приставку "tbl". Этот прием всегда используется при создании профессионального прило-

жения для того, чтобы отличить таблицу от запроса в окне построителя, т. к. в серьезных разработках их число может быть довольно значительным. Запрос же идентифицируют приставкой "qwr".

MS Access не предоставляет пользователю командной строки для непосредственного ввода SQL-конструкций. Но выход из этой ситуации есть — это конструктор запросов. Алгоритм работы следующий:

1. Откройте вторую вкладку **Создание** на ленте MS Access 2010.
2. Запустите конструктор запросов. Он расположен в разделе **Другие** этой вкладки.
3. Появится окно **Добавление таблицы**. Закройте его.
4. Выберите первую пиктограмму вкладки **Режим SQL**.
5. Откроется окно конструктора в режиме ввода кода языка SQL.
6. Сохраните запрос под именем "Учебный запрос" в базе данных.

Откройте "Учебный запрос" в режиме конструктора. Теперь у нас есть командная строка! Введите в нее текст запроса (рис. 6.1).

Рис. 6.1. Текст простейшего запроса

Точку с запятой в конце последней строки MS Access добавит сам. Перейдите в режим таблицы, сделав щелчок по пиктограмме **Режим**. Или закройте окно конструктора запросов и запустите запрос на выполнение (рис. 6.2).

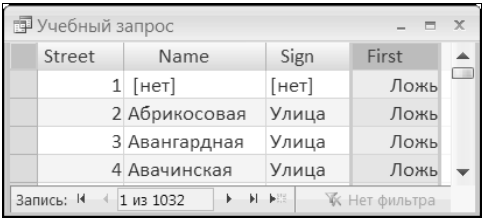


Рис. 6.2. Результат выполнения запроса по улицам

В запрос попали все улицы (1032 записи) из таблицы `tblStreet`. В предложении `SELECT` для выборки всех столбцов таблицы можно использовать звездочку (`*`). Результат будет такой же:

```
SELECT * FROM tblStreet;
```

6.2.2. Предложение *WHERE*

Использование в операторе `SELECT` предложения, определяемого ключевым словом `WHERE`, позволяет задавать выражение условия, принимающее значение `Истина` или `Ложь` для значений полей таблиц, к которым обращается оператор `SELECT`. Предложение `WHERE` определяет, какие строки указанных таблиц должны быть выбраны. В запрос включаются только те строки, для которых условие, указанное в предложении `WHERE`, принимает значение `Истина`.

Текст запроса, выполняющего выборку всех проживающих с фамилией (`Family`) Иванов, сведения о которых находятся в таблице `tblOwners`, таков:

```
SELECT Family, Name, Second FROM tblOwners
WHERE Family = 'Иванов'
   ORDER BY Name, Second;
```

В результате выполнения получим список из 79 проживающих (рис. 6.3). Выборка отсортирована по полям `Name` (имя) и `Second` (отчество).

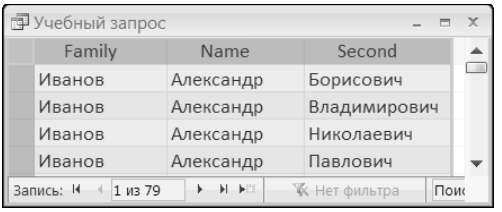


Рис. 6.3. Фамилию Иванов имеют 79 проживающих

6.2.3. Предложение *ORDER BY*

Как мы уже знаем, таблицы — это неупорядоченные наборы данных, и данные, которые выводятся из них, не обязательно появляются в какой-то определенной последовательности. Язык SQL использует ключевое слово `ORDER BY`, чтобы дать возможность упорядочить вывод. Оно упорядочивает вывод запроса согласно значениям в том или ином количестве выбранных столбцов. Вы можете использовать `ORDER BY` одновременно с любым числом столбцов. Обратите внимание, что во всех случаях столбцы, которые упорядочиваются, должны быть указаны в выборе `SELECT`. По умолчанию установлено возрастание. Текстовые поля будут отсортированы в алфавитном порядке. Столбцы типа **Дата/время** — в хронологическом.

В предыдущем примере сначала будет выполнена сортировка по имени, а затем по отчеству проживающего.

6.2.4. Предикат *DISTINCT*

SQL-запросы могут содержать строки с одинаковыми значениями атрибутов. Для исключения повторяющихся записей применяют ключевое слово DISTINCT. Внесем изменения в предыдущий запрос:

```
SELECT DISTINCT Family, Name, Second FROM tblOwners
   WHERE Family = 'Иванов'
   ORDER BY Name, Second;
```

Полученная выборка будет содержать 71 строку. В результаты запроса включаются только уникальные значения каждого из полей, перечисленных в операторе SELECT. Если две записи содержат абсолютно одинаковые фамилию, имя и отчество, то приведенная выше инструкция SQL возвращает только одну запись, содержащую данные о проживающем. Скорее всего, это будут разные люди. Кому придет в голову оформить оплату коммунальных услуг за одновременное проживание сразу в двух квартирах одного города? Да и институт прописки этого не позволит.

Ошибки заполнения базы данных здесь мы не рассматриваем, хотя в любом программном комплексе их достаточно!

6.2.5. Предикат *TOP*

Предназначен для возвращения заданного количества записей, находящихся в числе первых или последних в выборке. Запрос на отображение первых десяти фамилий в предыдущем примере имеет вид:

```
SELECT DISTINCT TOP 10 Family, Name, Second FROM tblOwners
WHERE Family = 'Иванов'
   ORDER BY Name, Second;
```

Если необходимо отобразить последние 10 записей, измените порядок сортировки (DESC — по убыванию):

```
SELECT DISTINCT TOP 10 Family, Name, Second From tblOwners
WHERE Family = 'Иванов'
   ORDER BY Name, Second DESC;
```

Если не включить предложение ORDER BY, в ответ на запрос будет выдан произвольный набор 10 записей из таблицы tblOwners, удовлетворяющий предложению WHERE.

Можно также использовать зарезервированное слово PERCENT для получения заданного процента первых или последних записей диапазона, заданного предложением ORDER BY.

6.2.6. Предложение *GROUP BY*

Объединяет записи с одинаковыми значениями, находящиеся в указанном списке полей, в одну запись:

```
SELECT <Список_столбцов>
  FROM <Список_таблиц>
  [GROUP BY <Список_полей_группы>]
```

Получим выборку всех однофамильцев с фамилией Иванов:

```
SELECT Family, Name, Second FROM tblOwners
WHERE Family = 'Иванов'
GROUP BY Family, Name, Second
   ORDER BY Name, Second;
```

Все записи, содержащие повторяющиеся фамилию, имя и отчество, будут объединены в одну. В выборке будет также 71 запись из 79 Ивановых, имеющихся в таблице `tblOwners`. Это не удивительно, т. к. этот запрос идентичен рассмотренному ранее:

```
SELECT DISTINCT Family, Name, Second FROM tblOwners
WHERE Family='Иванов'
   ORDER BY Name, Second;
```

только обладает большей возможностью для расширения функциональности. Воспользуемся этим. Сформируем выборку, содержащую список однофамильцев, у которых полностью совпадает имя и отчество. Эта выборка также должна содержать и число повторений записей.

6.2.7. Предложение *HAVING*

Определяет сгруппированные записи, которые должны отображаться в операторе `SELECT` с предложением `GROUP BY`. После того как записи будут сгруппированы предложением `GROUP BY`, предложение `HAVING` покажет те из них, которые отвечают его условиям.

Для поиска повторяющихся строк напишем запрос:

```
SELECT Family, Name, Second, Count(Family) AS Повторы
   FROM tblOwners
   WHERE Family = 'Иванов'
   GROUP BY Family, Name, Second
   HAVING Count(Name)>1 AND Count(Second)>1
   ORDER BY Name, Second;
```

В нашем случае из 79 Ивановых 7 имеют одинаковые имена и отчества, причем запись "Иванов Иван Иванович" встречается три раза (рис. 6.4).

В запрос добавлено дополнительное поле с названием `Повторы`. Для его формирования используется функция `Count()`. Она подсчитывает количество совпадающих записей. Аргумент представляет собой строковое выражение. Операндом в выражении может быть имя таблицы или функция, встроенная или определяемая пользователем, но не статистическая функция SQL. Подсчитать можно количество записей любого типа, включая текстовые.

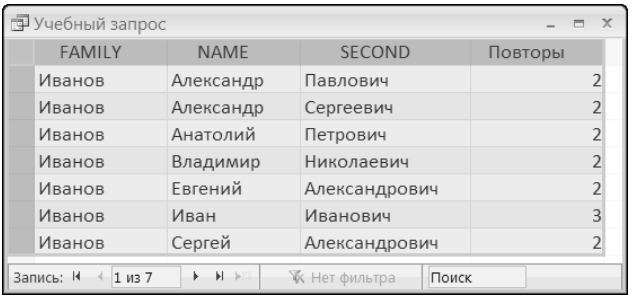


Рис. 6.4. Список однофамильцев

6.2.8. Выборка данных из нескольких таблиц

До сих пор мы рассматривали создание выборок только из одной таблицы. При работе с нормализованной базой данных непременно потребуется создавать запросы, использующие данные из нескольких таблиц. Их объединение обязательно. Оно выполняется по одному или нескольким столбцам. При объединении двух таблиц по 10 строк в каждой без объединения получится результат уже из 100 строк, а это практически всегда ни к чему.

В MS Access существуют два способа объединения таблиц. Первый способ основан на использовании ключевого слова `WHERE`, а второй — на связке ключевых слов `INNER-JOIN-ON`.

Сделаем выборку из двух таблиц: `tblBuilding` и `tblStreet`, содержащую адреса всех зданий и год их постройки:

```
SELECT tblStreet.Name, tblStreet.Sign,
       tblBuilding.House, tblBuilding.Year
FROM tblStreet, tblBuilding
WHERE tblstreet.Street=tblbuilding.Street
ORDER BY Year;
```

Результат запроса приведен на рис. 6.5. В запрос попали все здания из таблицы `tblBuilding` (334 штуки). Они отсортированы по году постройки. Без объединения таблиц результирующий запрос содержал бы 344 688 записей (334 здания × 1032 улицы).

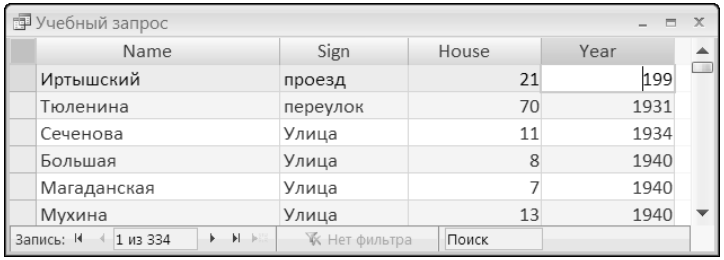


Рис. 6.5. Адреса зданий и год их постройки

Сразу заметна ошибка в первой строке. Год постройки здания слишком мал — 199. Попробуйте ее исправить — и у вас ничего не выйдет! Получен необновляемый набор записей.

Перепишем запрос с использованием связки ключевых слов `FROM-INNER-JOIN-ON`:

```
SELECT tblStreet.Name, tblStreet.Sign,
       tblBuilding.House, tblBuilding.Year
FROM tblStreet INNER JOIN tblBuilding
     ON tblStreet.Street=tblBuilding.Street
   ORDER BY Year;
```

Результат тот же, но набор записей — обновляемый! Исправьте год постройки здания. Такой запрос позволит это сделать. Исправления будут автоматически внесены в соответствующую запись таблицы `tblBuilding`.

Совет

Всегда используйте для объединения таблиц второй вариант. Он написан в соответствии с более новым стандартом языка SQL и предоставляет разработчику больше возможностей.

MS Access 2010 позволяет объединять в операторе `SELECT` данные более чем из двух таблиц. Синтаксис в этом случае выглядит так:

```
FROM (...(Таблица1 INNER JOIN Таблица2 ON Условие1) INNER JOIN
         (Таблица3 ON Условие2) INNER JOIN
         (Таблица4 ON Условие3) INNER JOIN ...)
```

В *главе 4* средствами конструктора запросов был создан запрос для отдела социальной защиты администрации Центрального района города. Результаты запроса были переданы в MS Excel. Цель достигнута. Заинтересованное ведомство получило требуемую информацию, но для придания отчету законченного вида специалисту администрации придется потратить значительное время на его доработку.

В отчет включены названия улиц (столбец `NAME` таблицы `tblStreet`) и имена владельцев квартир (столбец `NAME` таблицы `tblOwners`). Несомненно, столбцы в разных таблицах могут иметь одинаковые названия, но при размещении их в одной

выборке MS Access будет вынужден добавить к ним названия таблиц. Это, несомненно, не добавит отчету функциональности. К тому же адрес проживающего (улица, признак, номер дома и квартиры) и ФИО (фамилия, имя, отчество) в итоговом документе должны занимать только по одной колонке.

В языке SQL имеется возможность изменять названия итоговых столбцов и объединять несколько из них в один. Для переименования столбца в запросе применяется строка: `AS <Псевдоним>`, а для объединения полей — символ `&`. Текст SQL-запроса приведен в листинге 6.1. На рис. 6.6 показаны первые строки выборки. Всего строк — 206.

Листинг 6.1. Текст запроса для отдела социальной защиты

```
SELECT tblOwners.FAMILY & ' ' & tblOwners.NAME & ' ' &
       tblOwners.SECOND AS ФИО, tblStreet.NAME & ' ' &
       Lcase(tblStreet.SIGN) & ' ' & tblBuilding.HOUSE &
       ' , кв.' & tblFlats.FLAT AS Адрес, tblOwners.BORN
       AS [Дата рождения]
FROM ((tblStreet INNER JOIN tblBuilding ON
     tblStreet.STREET=tblBuilding.STREET)
     INNER JOIN tblFlats ON
     (tblBuilding.STREET=tblFlats.STREET) AND
     (tblBuilding.HOUSE=tblFlats.HOUSE))
     INNER JOIN tblowners ON
     (tblFlats.STREET=tblOwners.STREET) AND
     (tblFlats.HOUSE=tblOwners.HOUSE) AND
     (tblFlats.FLAT=tblOwners.FLAT)
WHERE Year(tblOwners.BORN)<1940 AND
      tblBuilding.DISTRICT=1
   ORDER BY tblOwners.FAMILY;
```

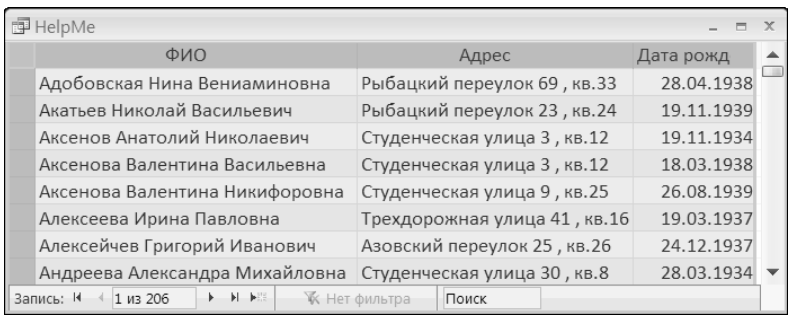


Рис. 6.6. Результаты выполнения запроса для отдела социальной защиты

6.2.9. Подчиненные запросы

В состав оператора SELECT, формирующего запрос к базе данных, могут входить подзапросы, результат выполнения которых используется для определения окончательного результата внешнего запроса. Подзапросы могут входить в состав конструкций WHERE и HAVING внешнего запроса и иметь несколько уровней вложения.

Воспользуемся этой возможностью языка SQL для поиска квартир в таблице tblFlats, в которых нет проживающих. Текст запроса приведен в листинге 6.2.

Листинг 6.2. Выборка пустующих квартир

```
SELECT *
FROM tblFlats
WHERE NOT EXISTS
   (SELECT * FROM tblOwners
       WHERE Street=tblFlats.Street AND
         House=tblFlats.House AND
         Flat=tblFlats.Flat)
   ORDER BY Street, House, Flat;
```

В составе запроса появились новые для вас ключевые слова: NOT EXIST. Они используются только с подзапросами. Результат их обработки представляет собой логическое True или False. По этим ключевым словам проверяется отсутствие строк в таблице tblOwners. Для проверки противоположного условия (строки есть) можно применить предикат EXIST.

В запрос попало 9 квартир из 18 475 имеющихся в таблице txtFlats. Несомненно, это ошибки набора информации. Исправить их сейчас практически невозможно, т. к. в выборке вместо названий улиц содержатся ссылки на них. Модифицируем запрос для придания ему надлежащего вида (листинг 6.3).

Листинг 6.3. Модификация запроса по пустующим квартирам

```
SELECT tblStreet.Name, tblStreet.Sign, tblFlats.House,
       tblFlats.Flat, tblFlats.Rooms
FROM tblStreet INNER JOIN tblFlats ON
     tblStreet.Street=tblFlats.Street
WHERE NOT EXISTS
     (SELECT * FROM tblOwners
               WHERE Street=tblFlats.Street AND
                     House=tblFlats.House AND
                     Flat=tblFlats.Flat)
   ORDER BY Name, House, Flat;
```

На рис. 6.7 приведена выборка квартир, для которых нет соответствующих записей в таблице проживающих.

В запросах такого типа вместо функции `EXIST` или `NOT EXIST` можно использовать стандартную функцию Microsoft Access `Count(*)`. Если в качестве параметра поставить знак `*`, то она возвратит количество записей в запросе (листинг 6.4).

Рис. 6.7. Результаты выполнения запроса по пустующим квартирам

Листинг 6.4. Применение `Count(*)` вместо `EXIST`

```
SELECT tblStreet.Name, tblStreet.Sign, tblFlats.House,
       tblFlats.Flat, tblFlats.Rooms
FROM tblStreet INNER JOIN tblFlats ON
     tblStreet.Street=tblFlats.Street
WHERE 1>
     (SELECT Count(*) FROM tblOwners
             WHERE Street=tblFlats.Street AND
                   House=tblFlats.House AND
                   Flat=tblFlats.Flat)
   ORDER BY Name, House, Flat;
```

Примечание

Время выполнения запроса с применением функции `Count(*)` несколько больше, чем с использованием ключевого слова `EXIST`.

6.3. Манипулирование данными

Язык DML (Data Manipulation Language), являясь компонентом SQL (Structured Query Language), используется для выборки данных и их обновления. В его состав входят четыре основных оператора: `SELECT`, `INSERT`, `UPDATE` и `DELETE`.

Оператор SELECT предназначен для выборки данных из таблиц и рассмотрен нами отдельно. Остальные же операторы позволяют вносить изменения в данные и поэтому будут рассмотрены в этом разделе.

6.3.1. Оператор *INSERT*

Предназначен для копирования строк из одной таблицы в другую, а также для добавления записи к таблице с использованием списка значений.

Синтаксис первой формы оператора (запрос на добавление) имеет вид:

```
INSERT INTO <Имя_таблицы>
SELECT <Инструкция>
```

Синтаксис второй формы оператора (добавление одной записи) имеет вид:

```
INSERT INTO <Имя_таблицы>
[(Столбец1[, Столбец2[, ...]])]
VALUES (Значение1[, Значение2[, ...])
```

Первая форма оператора используется для копирования содержимого одной таблицы в другую. Скопировать все здания из таблицы Building в таблицу tblBuilding можно при помощи оператора:

```
INSERT INTO tblBuilding
       SELECT * FROM Building
```

В состав оператора INSERT может входить любая конструкция SELECT, возвращающая выборку записей. Создание таких конструкций рассмотрено в предыдущем разделе.

Вторая форма предназначена для добавления записи в таблицу и заполнения этой записи значениями. Имена столбцов могут быть опущены. При этом значения должны следовать в том же порядке, что и столбцы в описании таблицы. Добавить новую улицу в таблицу улиц tblStreet можно при помощи оператора:

```
INSERT INTO tblStreet
(NumSTREET, Name, Sign, [First])
VALUES (1033, 'Генерала Деникина', 'Улица', False);
```

или в сокращенной форме:

```
INSERT INTO tblStreet
VALUES (1033, 'Генерала Деникина', 'Улица', False);
```

Название поля First (порядок следования в документах) MS Access 2010 автоматически заключил в квадратные скобки, т. к. в языке Access SQL имеется стандартная функция с таким же именем.

6.3.2. Оператор *UPDATE*

Применяется для создания запроса на изменение значения в одном или нескольких столбцах таблицы на основании заданных условий. Синтаксис оператора UPDATE имеет следующий вид:

```
UPDATE <Имя_таблицы>
SET Столбец1=Выражение1]
  [,Столбец2=Выражение2]
  [ ...]
WHERE <Условия_отбора>
```

Применять оператор UPDATE особенно удобно для внесения изменений в большое количество записей или когда записи, которые необходимо изменить, находятся в нескольких таблицах. Одновременно можно изменить несколько полей.

Обновление записей, выполненное с использованием запроса на обновление, нельзя отменить. Рекомендуется всегда делать резервные копии данных. Если записи будут удалены по ошибке, то их можно будет восстановить только из резервных копий.

Заменим фамилию работника в таблице tblUser:

```
UPDATE tblUser SET LastName = "Рощина"
       WHERE LastName= "Кудряшова";
```

В следующем примере изменяется столбец Flats (количество квартир) в таблице зданий. Количество квартир в здании подсчитывается на основании соответствующих записей в таблице tblFlats (квартиры). Текст запроса на изменение имеет вид:

```
UPDATE tblBuilding
       SET Flats =
          (SELECT Count(*) FROM tblFlats
                  WHERE Street=tblBuilding.Street
                  AND House=tblBuilding.House);
```

К сожалению, этот красивый запрос работать не будет. Язык SQL Access не поддерживает использование подчиненных запросов в предложении SET, поэтому попытка откорректировать поле FLATS в таблице tblBuilding обречена на неудачу, но выход есть:

- запишем запрос в виде:

  ```
  UPDATE tblBuilding SET Flats = DCountSQL(Street,House);
  ```

- напишем пользовательскую функцию DCountSQL (листинг 6.5) и поместим ее в модуль ModuleSlave.

Наша функция DCountSQL использует стандартную функцию DCount(), которая предназначена для определения числа строк в указанном наборе записей. Вызо-

вем ее для возврата числа записей в таблице tblFlats (квартиры), соответствующих зданию. Синтаксис функции имеет вид:

```
DCount(Выражение, Подмножество[, Условие])
```

Выражение — обязательный аргумент. Определяет поле, для которого требуется подсчитать записи. Может включать также элемент управления в форме, константу или функцию. Используем подстановочный знак звездочки (*). В этом случае функция DCount() подсчитает общее число записей, в том числе содержащих значения Null.

Подмножество — обязательный аргумент. Строковое выражение, определяющее набор записей, из которых состоит подмножество. Это может быть имя таблицы или запроса, не требующего параметра. В нашем случае — это таблица tblFlats.

Условие — необязательный аргумент. Строковое выражение, используемое для ограничения диапазона данных, обрабатываемых функцией DCount(). Например, *Условие* часто представляет собой предложение WHERE в выражении SQL, но без слова WHERE. Если аргумент *Условие* опущен, функция DCount() обрабатывает аргумент *Выражение* для всего подмножества. Любое поле, включенное в аргумент *Условие*, должно быть также и полем *Подмножества*. В противном случае функция DCount() возвращает значение Null.

Функция Count() была оптимизирована для ускорения подсчета записей в запросах. Используйте функцию Count() в выражении запроса вместо функции DCount(), а чтобы наложить ограничения, задайте дополнительные условия для результата. Используйте функцию DCount(), если требуется подсчитать записи в подмножестве из программного модуля или макроса либо в вычисляемом элементе управления.

Примечание

Для сцепления строк предпочтительнее использовать амперсанд. Следует избегать использования оператора сложения для каких-либо операций, кроме сложения чисел, за исключением случаев, когда действительно нужно, чтобы значение Null распространялось на все выражение.

Листинг 6.5. Пользовательская функция DCountSQL

```
Public Function DCountSQL(NumberStreet, NumberHouse) As Integer
' Возвращает количество квартир в здании
' NumberStreet — номер улицы
' NumberHouse  — номер дома
DCountSQL = DCount("*", "tblFlats", _
    "Street = " & NumberStreet & _
    "AND House = " & NumberHouse)
End Function
```

Представьте себе обычную ситуацию. Подчиненный запрос в предложении `SET` не поддерживается, а нужной стандартной функции в арсенале MS Access нет! Не отчаивайтесь, выход есть всегда.

- Запишем запрос в виде:

```
UPDATE tblBuilding SET Flats = CountSQL(Street,House);
```

- Запустим VBA и добавим ссылку на библиотеку: Microsoft ActiveX Data Objects 2.1 Library (пункт меню: **Tools | References**).
- Напишем пользовательскую функцию `CountSQL` (листинг 6.6) и поместим ее в модуль `ModuleSlave`.

Листинг 6.6. Пользовательская функция `CountSQL`

```
Public Function CountSQL(NumberStreet, NumberHouse) As Integer
' Возвращает количество квартир в здании
' NumberStreet — номер улицы
' NumberHouse  — номер дома
Dim CountSelectSQL As Integer      ' Количество квартир
' Создание ссылок на реальные объекты
Dim CN As ADODB.Connection
' CN — Объектная переменная — Соединение
' Объект ADO верхнего уровня
Dim RS As ADODB.Recordset
' RS — Объектная переменная, которая содержит набор записей
Set CN = CurrentProject.Connection   ' Открытие соединения
Set RS = New ADODB.Recordset         ' Создание набора
With RS
     ' Задание свойств объекта RS (Recordset)
     ' Источник: SQL-конструкция
     .Source = "SELECT * FROM tblFlats " & _
               "WHERE Street = " & NumberStreet & _
               " AND House= " & NumberHouse
     ' Указатель на открытое соединение
     .ActiveConnection = CN
     .CursorType = adOpenKeyset   ' Тип курсора
     ' Метод объекта RS
     ' Открывает набор данных активного объекта
     .Open
End With
CountSelectSQL = RS.RecordCount
' RecordCount — свойство объекта RS
' Возвращает количество записей
' Освобождение объектных переменных
Set RS = Nothing
Set CN = Nothing
```

```
' Количество записей в выборке
' CountSQL — имя процедуры-функции
CountSQL = CountSelectSQL
End Function
```

Мы воспользовались доступом к базе данных из VBA-кода. Visual Basic не является языком баз данных, поэтому в тексте пользовательской функции для работы с таблицами применяются предназначенные для этого объекты, их свойства и методы. Используется модель доступа к данным ADO (ActiveX Data Objects), которая позволяет работать с базами данных из приложения, поддерживающего Visual Basic for Applications.

6.3.3. Оператор *DELETE*

Создает запрос на удаление, который удаляет записи из таблицы. Синтаксис оператора DELETE имеет следующий вид:

```
DELETE
FROM <Имя_таблицы>
WHERE <Условие_отбора>
```

Оператор DELETE особенно удобен в тех случаях, когда требуется удалить много записей. Чтобы удалить из базы данных таблицу целиком, можно использовать метод EXECUTE с инструкцией DROP. Однако в этом случае вместе с таблицей будет утрачена и ее структура. При использовании оператора DELETE удаляются только данные, структура таблицы и все ее свойства, такие как атрибуты полей и индексы, остаются без изменений.

Примечание

Запрос на удаление удаляет всю запись, а не только данные, содержащиеся в определенных полях. Чтобы удалить значения из конкретных полей, следует создать запрос на обновление.

6.4. Определение данных при помощи SQL

MS Access имеет в своем составе превосходный инструмент для работы с таблицами — конструктор. Работа с ним была рассмотрена в *главе 2*. Однако если вам приходится часто иметь дело с "аналитикой", то в свой арсенал разработчика обязательно следует добавить средства языка DDL (Data Definition Language).

В состав языка DDL, который используется для описания структур баз данных и управления доступом к ним, входят четыре оператора:

- CREATE TABLE — создание новой таблицы;
- ALTER TABLE — изменение структуры таблицы;

- DROP — удаление таблицы или индекса;
- CREATE INDEX — создание индекса.

Они позволяют обойтись без конструктора там, где работу по созданию, модификации и удалению таблиц требуется автоматизировать.

6.4.1. Создание таблиц. Оператор *CREATE TABLE*

Оператор создает пустую таблицу. Такая таблица не имеет строк. Значения в нее вводятся с помощью оператора INSERT. Оператор CREATE TABLE определяет имя таблицы и множество поименованных столбцов в указанном порядке. Для каждого столбца должен быть определен тип и размер. Каждая создаваемая таблица должна иметь, по крайней мере, один столбец. Синтаксис оператора CREATE TABLE имеет следующий вид:

```
CREATE [TEMPORARY] TABLE <Имя_таблицы>
  (Столбец1 ТипДанных1 [(Размер1)] [CONSTRAINT <Индекс1>]
 [(Столбец2 ТипДанных2 [(Размер2)] [CONSTRAINT <Индекс2>]
 [ ...]]
 [ CONSTRAINT <Индекс_нескольких_полей> [, ...]])
```

Оператор CREATE TABLE включает в себя компоненты, описанные в табл. 6.1.

Таблица 6.1. *Компоненты оператора создания таблицы*

Компонент	Описание
Имя_таблицы	Имя создаваемой таблицы
Столбец1, *Столбец2*	Имена полей, которые создаются в новой таблице. Должно быть создано хотя бы одно поле
ТипДанных1, *ТипДанных2*	Тип данных компонента *Столбец* в новой таблице
Размер1, *Размер2*	Размер поля в знаках (только для полей с типом данных TEXT,BINARY и STRING)
Индекс1, *Индекс2*	Предложение CONSTRAINT, определяющее индекс по одному полю
Индекс_нескольких_полей	Предложение CONSTRAINT, определяющее индекс по нескольким полям

Предупреждение

Таблица, созданная с помощью атрибута TEMPORARY, доступна только в течение того сеанса, во время которого она была создана. По завершении сеанса она автоматически удаляется. К временной таблице могут иметь доступ несколько пользователей.

Типы данных, которые могут быть созданы при помощи оператора CREATE TABLE в MS Access 2010 приведены в табл. 6.2.

Таблица 6.2. *Типы данных, которые доступны в языке SQL версии MS Access 2010*

Тип данных	Описание
BYTE	Байт
SMALLINT или SHORT	Целое
INTEGER или LONG	Длинное целое
SINGLE	Одинарной точности с плавающей точкой
NUMERIC, FLOAT или DOUBLE	Двойной точности с плавающей точкой
CHAR(N), TEXT(N) или STRING(N)	Текстовый длиной *N* символов
BINARY(N)	Двоичный (*N* разрядов)
DATE или DATETIME	Дата/время
CURRENCY	Денежный
LOGICAL или BIT	Логический

Следующий пример демонстрирует создание таблицы tabStreet (улицы города). Ее структура приведена на рис. 6.8. По полю NumStreet (номер улицы) создан первичный ключ (NumStreet PRIMARY KEY).

```
CREATE TABLE tabStreet
       (NumStreet SMALLINT CONSTRAINT NumStreet PRIMARY KEY,
        Name CHAR (30),
        Sign CHAR(10),
        First LOGICAL)
```

Свяжем таблицу tabStreet отношением "один-ко-многим" с таблицей зданий tabBuilding (рис. 6.9). Связь будет установлена автоматически сразу после создания таблицы tabBuilding. Для этих целей напишем следующий оператор языка SQL:

```
CREATE TABLE tabBuilding
(NumStreet SMALLINT CONSTRAINT NumStreet REFERENCES tabStreet,
House SMALLINT,
Land SINGLE,
Elevator LOGICAL,
CONSTRAINT NumStreetAndHouse PRIMARY KEY (NumStreet, House))
```

Вторая строчка кода создает столбец NumStreet (номер улицы). Он назначен внешним ключом (CONSTRAINT NumStreet) для связи (REFERENCES) с таблицей tabStreet. В последней строке кода создается составной первичный ключ NumStreetAndHouse таблицы tabBuilding по двум полям: NumStreet + House. Этот

первичный ключ в организации связи с таблицей `tabStreet` не используется. Он создан для перевода таблицы зданий во вторую нормальную форму. На рис. 6.10 показан вид базы данных после запуска на выполнение двух созданных выше операторов SQL.

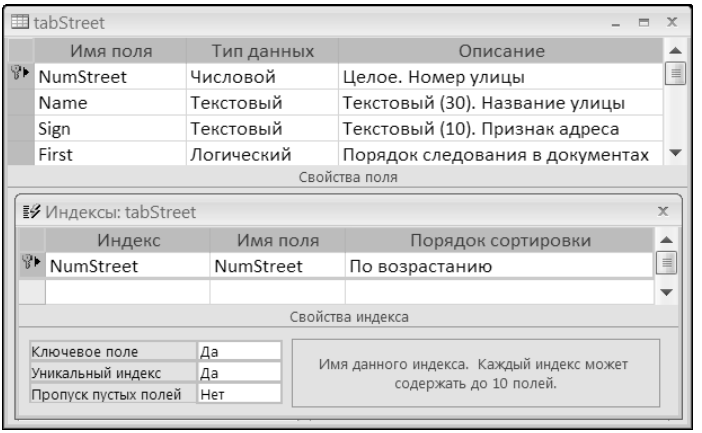


Рис. 6.8. Структура таблицы `tabStreet`

Рис. 6.9. Структура таблицы `tabBuilding`

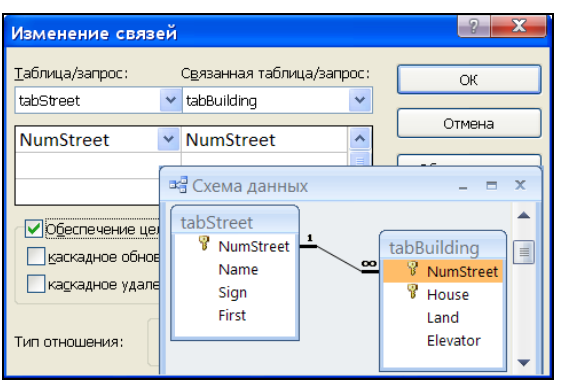


Рис. 6.10. Связь "один-ко-многим" установлена

6.4.2. Модификация таблиц. Оператор *ALTER TABLE*

Оператор ALTER TABLE предназначен для изменения структуры таблицы, созданной с помощью оператора CREATE TABLE или посредством конструктора таблиц. Применяя этот оператор, можно добавить или удалить новый столбец, ввести или удалить ограничение. Один оператор можно применить только к одному столбцу или индексу. Синтаксис оператора ALTER TABLE имеет следующий вид:

```
ALTER TABLE <Имя_таблицы>
ADD [COLUMN] <Имя_столбца> ТипДанных [(Размер)][NOT NULL]
    [CONSTRAINT <Индекс>]
ALTER COLUMN <Имя_столбца> ТипДанных [(Размер)]
DROP [COLUMN] <Имя_столбца>
     [CONSTRAINT <Имя_индекса>]
[ CONSTRAINT <Индекс_нескольких_полей> [, ...]])
```

Оператор CREATE TABLE включает в себя следующие компоненты (табл. 6.3).

Таблица 6.3. *Компоненты оператора модификации таблицы*

Компонент	Описание
Имя_таблицы	Имя создаваемой таблицы
Имя_столбца	Имя поля, которое будет добавлено в таблицу, удалено из нее или изменено в таблице
ТипДанных	Тип данных компонента *Имя_столбца* в модифицируемой таблице
Размер	Размер поля в знаках (только для полей с типом данных TEXT,BINARY и STRING)
Индекс	Имя создаваемого индекса
Имя_индекса	Имя удаляемого индекса
Индекс_нескольких_полей	Предложение CONSTRAINT, определяющее индекс по нескольким полям

Ключевое слово ADD COLUMN применяется для добавления в таблицу нового поля. Укажите имя поля, тип данных, а также размер для полей с типом данных TEXT, BINARY и STRING.

Если для поля определено свойство NOT NULL, то поле обязательно должно содержать допустимые данные.

Ключевое слово ALTER COLUMN предназначено для изменения типа данных существующего поля. Укажите имя поля, новый тип данных, а также размер для полей с типом данных TEXT, BINARY и STRING.

Ключевое слово ADD CONSTRAINT предназначено для добавления индекса набора полей. Используйте конструкцию DROP COLUMN для удаления поля. Требуется указать только имя поля. Конструкция DROP CONSTRAINT предназначена для удаления индекса набора полей. Требуется указать только имя индекса после зарезервированного слова CONSTRAINT.

Предупреждение

- Невозможно одновременно добавить или удалить несколько полей или индексов.
- Чтобы добавить индекс для одного поля или для набора полей в таблице, используйте инструкцию CREATE INDEX. Чтобы удалить индекс, созданный с помощью инструкции ALTER TABLE или CREATE INDEX, можно воспользоваться инструкцией ALTER TABLE или DROP.
- Свойство NOT NULL можно задавать для одного поля или внутри именованного предложения CONSTRAINT для одного или нескольких полей. Ограничение NOT NULL для поля можно задать только один раз. При попытке повторно определить это ограничение возникает ошибка выполнения.

Модифицируем таблицу tblBuilding, входящую в состав базы данных Real Estate 2010. Добавим в нее столбец FLATS (количество квартир в здании). Для выполнения этой операции напишем оператор:

```
ALTER TABLE tblBuilding
      ADD FLATS SHORT;
```

6.4.3. Удаление таблиц и индексов. Оператор *DROP*

Оператор DROP предназначен для удаления таблицы, процедуры или представления из базы данных либо индекса из таблицы. Существуют четыре формы оператора DROP:

```
DROP TABLE <Имя_таблицы>
DROP INDEX <Индекс> ON <Имя_таблицы>
DROP PROCEDURE <Имя_процедуры>
DROP VIEW <Имя_представления>
```

Компоненты оператора DROP приведены в табл. 6.4.

Таблица 6.4. *Компоненты оператора* DROP

Компонент	Описание
Имя_таблицы	Имя удаляемой таблицы
Индекс	Имя индекса, удаляемого из таблицы
Имя_процедуры	Имя удаляемой процедуры
Имя_представления	Имя удаляемого представления

Предупреждение

Перед удалением таблицы или индекса эту таблицу необходимо закрыть. Для удаления индекса можно также воспользоваться инструкцией ALTER TABLE.

6.4.4. Создание индекса. Оператор *CREATE INDEX*

Ключевое слово CONSTRAINT операторов CREATE TABLE и ALTER TABLE дает возможность работы с индексами. Наряду с этими операторами для создания первичного ключа или дополнительных индексов в существующих таблицах можно использовать оператор CREATE INDEX.

Синтаксис оператора CREATE INDEX имеет следующий вид:

```
CREATE [ UNIQUE ] INDEX <Имя_индекса>
  ON <Имя_таблицы> (Столбец1 [ASC|DESC]
                  [, Столбец2 [ASC|DESC], ...])
  [WITH { PRIMARY | DISALLOW NULL | IGNORE NULL }]
```

Компоненты оператора CREATE INDEX приведены в табл. 6.5.

Таблица 6.5. *Компоненты оператора создания индексов*

Компонент	Описание
Имя_таблицы	Имя существующей таблицы, для которой создается индекс
Имя_индекса	Имя создаваемого индекса
Столбец1, *Столбец2*	Имя одного или нескольких полей для индексации

Ключевое слово UNIQUE используется для запрещения появления повторяющихся значений в одном или нескольких индексированных полях.

Чтобы определить правила проверки значений данных, можно использовать необязательное предложение WITH.

Возможны следующие действия при создании индексов:

- для создания индекса по одному полю укажите имя поля в круглых скобках после имени таблицы;
- для создания индекса по нескольким полям укажите имена всех полей, включаемых в индекс;
- для создания убывающего индекса используйте ключевое слово DESC, в противном случае будет создан индекс по возрастанию.

Ключевое слово DISALLOW NULL запрещает пустые значения в одном или нескольких индексированных полях новых записей.

Ключевое слово `IGNORE NULL` разрешает пустые значения в одном или нескольких индексированных полях новых записей.

Ключевое слово `PRIMARY` создает первичный ключ. `UNIQUE` в этом случае можно опустить.

Предупреждение

Если в таблице уже есть первичный ключ, не используйте зарезервированное слово `PRIMARY` при создании в ней нового индекса. Это приведет к возникновению ошибки.

ГЛАВА 7

Разработка интерфейса приложения

Трудно переоценить роль интерфейса пользователя в современных программных комплексах. Удачный или неудачный интерфейс во многом предопределяет успех или неудачу всей разработки. Удобство интерфейса — вещь достаточно субъективная. То, что нравится одному пользователю, совсем не подойдет для другого. То, что обеспечивает комфортную работу служащих одного отдела, будет встречено в штыки в другом. Поэтому при проектировании системы лучше всего заранее включать в нее возможность настройки интерфейса на различные категории пользователей. Вот уже много лет автор применяет в своих разработках для создания интерфейса метод пересекающихся каскадов. Этот метод появился на свет в результате бесконечных переделок меню программного комплекса, обеспечивающего работу Департамента муниципальной собственности администрации одного из городов России.

7.1. Метод пересекающихся каскадов

Основное достоинство Windows-приложений — их стандартный вид. Если пользователь научился работать в одном из них, то можно считать, что он без труда освоит любое. Все приложения Windows используют одни и те же соглашения. Де-факто существует несколько общих принципов разработки приложений под Windows. Следуя им, разработчик получает ряд существенных преимуществ. Во-первых, разработанное приложение выглядит профессионально. Во-вторых, оно легко осваивается пользователем и согласуется с другими приложениями, и в-третьих, имеет современный дизайн.

Необходимо отметить, что следования только этим общим принципам для разработки полнофункционального корпоративного приложения недостаточно. Современный программный комплекс должен обеспечить максимум удобств каждому работнику любого подразделения предприятия. Основным препятствием комфортной работе в случае корпоративного приложения служит система защиты от несанкционированного доступа. Работники одного отдела не должны ви-

деть или корректировать некоторую часть информации другого отдела, а также выполнять чужие запросы и расчеты, реализованные в корпоративном приложении. Однако доступная информация всегда должна быть под рукой. Для того чтобы реализовать эти противоречивые требования, используйте *метод пересекающихся каскадов*.

В главном меню программного комплекса, выполненного в стиле Stage (сцена), каждое подразделение или его функционально независимая часть реализована в виде каскада. В нужном, по смыслу решаемой задачи, месте реализован переход в другой каскад (смежного отдела) с изменением доступа к той части информации, которая должна быть недоступна работнику другого подразделения. Таких пересечений может быть сколько угодно. Второй отличительной особенностью предложенного метода является отсутствие сообщений обработчика ошибок типа "В доступе к данным отказано", которые негативно сказываются на работе пользователя. Лучше в самом начале исключить направления выбора, приводящие в конечном итоге к вмешательству в действия оператора служб сетевой операционной системы.

Метод пересекающихся каскадов может быть с успехом применен при разработке прикладного программного обеспечения практически любой корпорации. На рис. 7.1 приведен пример использования метода пересекающихся каскадов для разработки прикладного программного обеспечения корпорации Real Estate. Подробно показана совместная деятельность только тех подразделений корпорации, которые обеспечивают сбор платы за аренду недвижимости корпорации, не используемой в основном производстве, связанном с выращиванием экзотических растений и доставкой их по всему миру. Это подразделения учета недвижимости, учета арендаторов и учета доходов корпорации. Каскады работы с программным комплексом всех этих подразделений на рисунке показаны одновременно, причем первых двух полностью, а каскад подразделения учета доходов — частично. В реальной же работе сотруднику подразделения доступен только каскад подразделения или еще более мелкий — каскад отдела. Поэтому даже комбинированный вид экрана рабочей станции содержит погашенные пункты меню (**Филиалы** и **Платежи**), с которых начинается работа других подразделений компании, не рассматриваемых в данном примере.

Начало работы сотрудника подразделения учета недвижимости — пункт **Здания**, находящийся в главном меню. Добавление, удаление и корректировка данных по объекту недвижимости входит в его обязанности. Вы видите, что все пункты всплывающего меню доступны. Проследим каскад поиска данных по оплате за аренду конкретного помещения:

- **Здания**;
- **Поиск здания по адресу**;
- **Список зданий, попавших в запрос**;
- **Просмотр информации по зданию**;

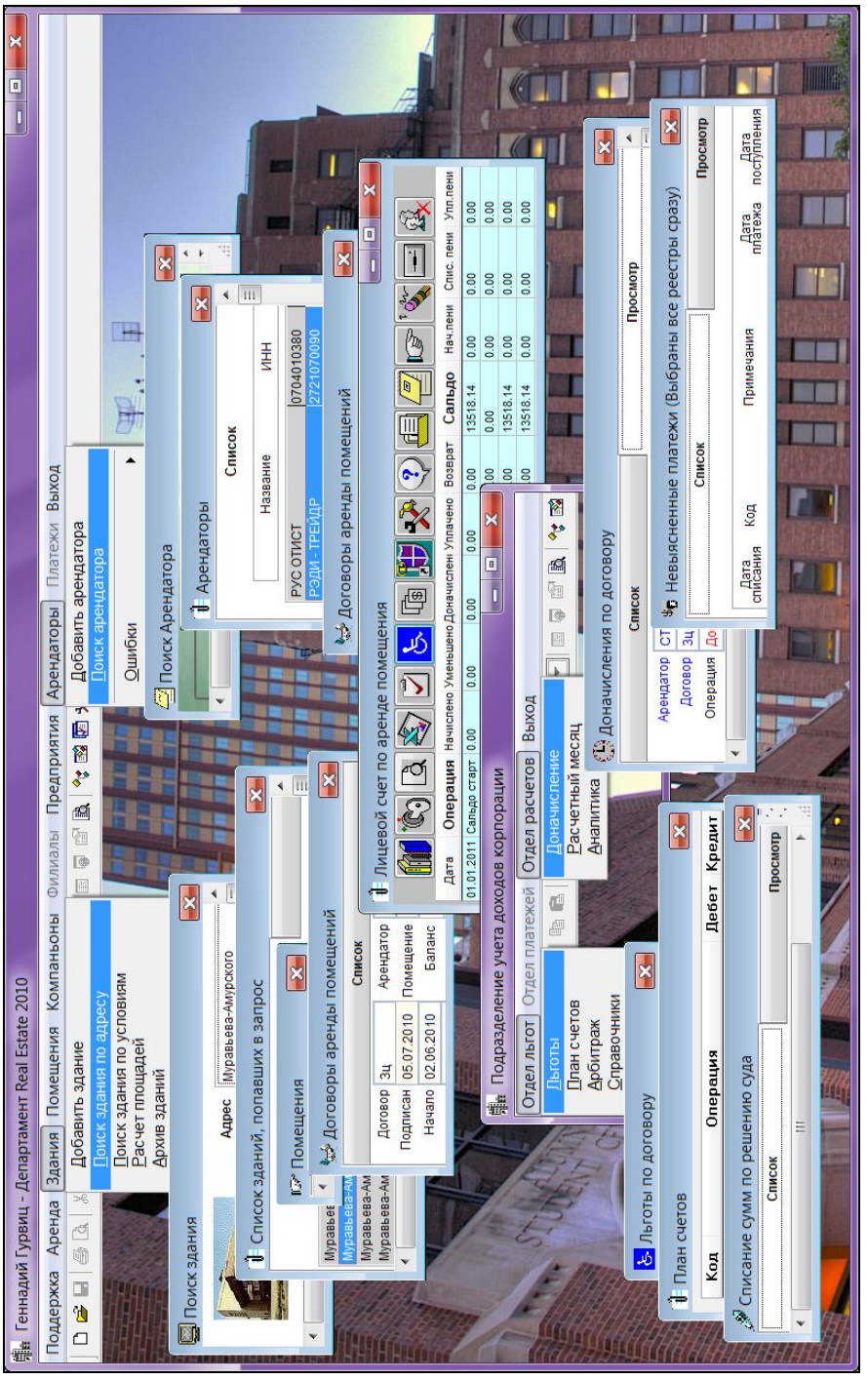


Рис. 7.1. Метод пересекающихся каскадов

- **Помещения**;
- **Договоры аренды помещений**;
- **Лицевой счет по аренде помещения**.

Далее начинается каскад подразделения учета доходов корпорации Real Estate, состоящий из каскадов трех отделов этого подразделения. Понятно, что деятельность работника подразделения учета недвижимости, попавшего в чужие каскады, функционально ограничена. Он может воспользоваться только правами просмотра некоторой части информации, да и то не во всех каскадах. Вход в каскад отдела платежей недоступен.

Каскад поиска данных по арендной плате, пройденный работником подразделения учета арендаторов начинается с пункта **Арендаторы** и заканчивается окном **Лицевой счет по аренде помещения**. Здесь каскады также пересекаются. Путь в другие каскады возможен, но также функционально ограничен.

Главный каскад подразделения учета доходов начинается с пункта основного меню **Платежи**. На рис. 7.1 он недоступен. Следует отметить, что переход работника этого подразделения в каскады других подразделений корпорации предусмотрен. Однако возможность корректировки информации, некоторых запросов и статистических расчетов для него будет исключена.

Реализовать возможности, которые дает применение метода пересекающихся каскадов, в реальном приложении можно только при согласованной совместной работе администратора базы данных и разработчика программного комплекса. На практике очень часто их взаимодействие носит чисто формальный характер, а это может погубить грамотно выполненную разработку. Просто вас, а не администратора базы данных обвинят в непрофессионализме, поскольку сотрудники предприятия в своей повседневной деятельности общаются только с разработчиком.

7.2. Создание меню программного комплекса

В меню объединяют последовательности и группы команд, одну из которых может выбрать работник для совершения очередного действия. Как правило, названия команд в меню достаточно информативны, так что пользователь может легко найти нужную ему команду. Команды для решения близких задач всегда объединяются в группу. В одном меню команды объединяют на основе одного из двух принципов: либо это различные действия над одним объектом, либо однотипные действия над различными объектами.

Приложения Microsoft Access 2010 позволяют без труда перестраивать главную ленту, управлять параметрами окон, выбирать нужные форматы представления данных на экране, создавать собственные диалоговые окна и размещать на них

разнообразные управляющие элементы. Но применения только стандартных возможностей недостаточно для создания полнофункционального приложения.

Согласно существующим стандартам проектирования интерфейса, работа программного комплекса должна начинаться с активизации головного меню, которое находится в верхней части окна приложения. Собственное головное меню для прикладной системы можно спроектировать визуально, используя средства Microsoft Access или язык программирования. Если добавления или изменения невелики, то можно модернизировать главную ленту, а при существенных изменениях (добавлении нескольких новых меню и подменю) целесообразно создавать собственную систему меню. Разумеется, в этом случае следует принимать во внимание будущих пользователей разрабатываемой системы, их образование, квалификацию, опыт работы с компьютером и т. д.

Для "среднего" пользователя — типичного сотрудника типичной организации — приложение MS Access 2010, несомненно, должно иметь собственный интерфейс, лишь в небольшой степени, использующий встроенные системные элементы, панели инструментов и диалоговые окна Access 2010. Поэтому в большинстве случаев разработчику следует создавать свое головное меню или ленту, перекрывающие частично или полностью на экране главную ленту Microsoft Access. На этом пути нам поможет VBA. На рис. 7.2 показано главное меню программного комплекса, расположенное на вкладке **Надстройки**. От ленты интерфейса Microsoft Access 2010 оставлены только вкладки **Файл**, **Главная** и **Панель быстрого доступа**. Причем доступных объектов на вкладке **Главная** совсем немного.

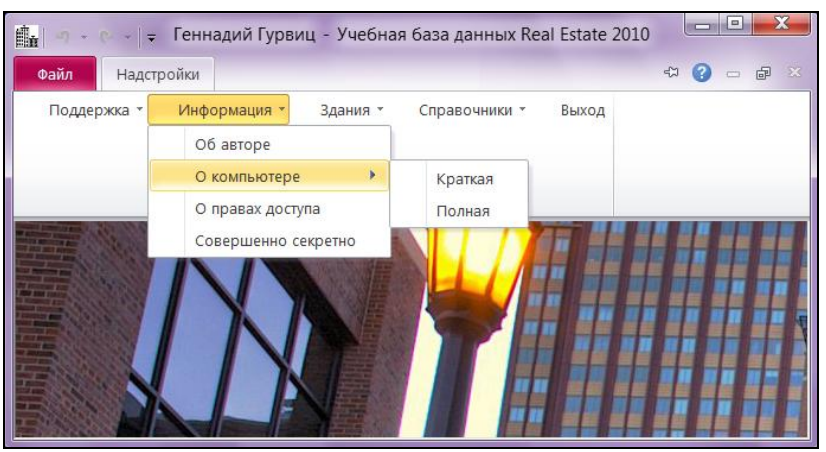


Рис. 7.2. Главное меню программного комплекса Real Estate 2010

Перед тем, как приступить к программированию, самым тщательным образом спроектируйте систему меню. Состав меню невозможно определить без учета конкретных задач, для решения которых предназначено разрабатываемое прило-

жение. Вначале нарисуйте вид меню на бумаге, обязательно обсудив его структуру с заказчиком, и только потом приступайте к программированию. Пока меню не превратилось в строчки кода, внести в него изменения гораздо проще.

Текст процедуры, создающей меню программного комплекса Real Estate 2010, приведен в листинге 7.1.

Листинг 7.1. Главное меню программного комплекса

```
Public Sub StartMainMenu()
' RealEstate.accdb
' Создание главного меню программного комплекса
' Рабочая переменная с именем cbar. Тип — строка меню
Dim cbar  As CommandBar
' Рабочая переменная с именем Exist. Тип — логический
Dim Exist As Boolean

Exist = False    ' Начальное значение переменной Exist
For Each cbar In CommandBars
    If cbar.NAME = "MainMenu" Then
       ' Главное меню с именем MainMenu уже создано
       ' Это возможно только в случае аварийного
       ' завершения предыдущего сеанса работы комплекса
       Exist = True
       ' Значение рабочей переменной Exist — истина
       Exit For
    End If
Next cbar

If Not Exist Then
   ' Значение переменной Exist — ложь
   ' Главное меню не создано
   ' Создаем его
   Set cbar = CommandBars.Add(NAME:="MainMenu", _
       Position:=msoBarTop, MenuBar:=True, Temporary:=False)
   ' MainMenu — имя главного меню программного комплекса
   ' Если MenuBar:=True  — будет создана строка меню
   ' Если MenuBar:=False — будет создана панель инструментов
   ' Если Temporary:=False — строка меню не временная и
   ' останется после завершения работы
End If
cbar.Enabled = True      ' Главное меню доступно
cbar.Visible = True      ' Главное меню видимо

With cbar
   With .Controls
```

```
With .Add(Type:=msoControlPopup)
     ' Создание всплывающего Popup меню
     ' Заголовок меню
     .Caption = "Поддержка"
     With .Controls
       ' Создание пункта меню
       With .Add(Type:=msoControlButton)
            ' Заголовок пункта меню
            .Caption = "Смена картинки"
            .Enabled = ChangePicture
            ' При выборе этого пункта меню
            ' запустить процедуру ChPicture
            ' Она находится в модуле ModuleSlave
            .OnAction = "ChPicture"
            ' Всплывающая подсказка
            .TooltipText = "Картинка окна программы"
       End With
       With .Add(Type:=msoControlButton)
            ' Заголовок пункта меню
            .Caption = "Смена пароля"
            .Enabled = ChangePassword
            ' При выборе этого пункта меню
            ' запустить процедуру Password
            ' Она находится в модуле ModuleSlave
            .OnAction = "Password"
            ' Всплывающая подсказка
            .TooltipText = "Пункт доступен всем"
       End With
       With .Add(Type:=msoControlButton)
            .Caption = "Календарь ежедневник"
            .Enabled = True
            .OnAction = "Calendar"
            .TooltipText = "Пункт доступен всем"
       End With
       With .Add(Type:=msoControlButton)
            .Caption = "Калькулятор"
            .Enabled = True
            .OnAction = "Calculator"
            .TooltipText = "Пункт доступен всем"
       End With
       With .Add(Type:=msoControlButton)
            .Caption = "Выход"
            .Enabled = True
            .OnAction = "StopMenu"
       End With
     End With
End With
```

```
With .Add(Type:=msoControlPopup)
     .Caption = "Информация"
     With .Controls
       With .Add(Type:=msoControlButton)
            .Caption = "Об авторе"
            .Enabled = True
            .OnAction = "Autor"
       End With
       With .Add(Type:=msoControlPopup)
            .Caption = "О компьютере"
            With .Controls
                 With .Add(Type:=msoControlButton)
                      .Caption = "Краткая"
                      .Enabled = True
                      .OnAction = "Shot"
                 End With
                 With .Add(Type:=msoControlButton)
                      .Caption = "Полная"
                      .Enabled = True
                      .OnAction = "Long"
                 End With
            End With
       End With
       With .Add(Type:=msoControlButton)
            .Caption = "О правах доступа"
            .Enabled = RightAccess
            .OnAction = "Access"
       End With
       With .Add(Type:=msoControlButton)
            .Caption = "Совершенно секретно"
            .Enabled = True
            .OnAction = "Absolutely"
       End With
     End With
End With

With .Add(Type:=msoControlPopup)
     .Caption = "Здания"
     With .Controls
       With .Add(Type:=msoControlButton)
            .Caption = "Поиск здания"
            .Enabled = SeekBuilding
            .OnAction = "Search"
       End With
       With .Add(Type:=msoControlButton)
            .Caption = "Добавить здание"
```

```
                .Enabled = AddBuilding
                .OnAction = "AddBuild"
            End With
        End With
    End With
    With .Add(Type:=msoControlPopup)
        .Caption = "Справочники"
        With .Controls
            With .Add(Type:=msoControlButton)
                .Caption = "Улицы города"
                .Enabled = StreetTown
                .OnAction = "Street"
            End With
            With .Add(Type:=msoControlButton)
                .Caption = "Районы города"
                .Enabled = DistrictTown
                .OnAction = "District"
            End With
            With .Add(Type:=msoControlButton)
                .Caption = "Материал стен"
                .Enabled = MaterialWall
                .OnAction = "Wall"
            End With
            With .Add(Type:=msoControlButton)
                .Caption = "Работники"
                .Enabled = Staff
                .OnAction = "Employee"
            End With
        End With
    End With

    With .Add(Type:=msoControlButton)
        .Style = msoButtonCaption
        .Caption = "Выход"
        .Enabled = True
        .OnAction = "StopMenu"
    End With
  End With
End With

End Sub
```

7.3. Создание модуля VBA

Программный код VBA хранится в модуле. Модуль состоит из секции объявлений, за которой следует как минимум одна процедура. Модули бывают двух типов: глобальные и модули классов. Как узнать, к какому типу относится модуль VBA? Откройте **Область навигации**. Глобальные модули представлены в разделе **Модули**. Модули классов хранятся в составе форм и отчетов и не отображаются в области навигации. Они создаются, как правило, автоматически вместе с "родительскими" объектами. Получить доступ к модулям классов можно через окно VBA (**Project**).

Примечание

Название модуля класса формируется из имени объекта с добавлением приставки `Form_` для форм и `Report_` для отчетов. Модуль класса появится в окне **Project Explorer** только после добавления хотя бы одной процедуры обработки события для объекта.

Поместим разработанный текст процедуры `StartMainMenu` в глобальный модуль с именем `ModuleMain` нашего приложения. Для этого:

1. Откройте главное окно Microsoft Access 2010.
2. Выберите вкладку **Работа с базами данных**, в левой части которой расположена пиктограмма **Visual Basic**. Щелкните по ней.
3. Запустится Microsoft Visual Basic. В его главном меню выберите пункт **Insert**.
4. Откроется всплывающее меню. Щелкните по второй строчке — **Module**. Теперь можно набирать текст модуля (рис. 7.3).
5. После завершения набора сделайте щелчок мышью по первому пункту меню Visual Basic, который носит название **File**. Откроется подменю.
6. Выберите в нем первый пункт со стилизованным изображением дискеты и названием применительно к нашему случаю: **Save Real Estate Часть II**. Появится окно для ввода названия модуля. Наберите `ModuleMain` (рис. 7.4).

В созданный модуль вы можете поместить все свои процедуры. Если они объявлены `Public`, то будут доступны из любой точки приложения. Запустить процедуру `StartMainMenu` на выполнение можно, указав ее имя в нужном месте кода без каких-либо ключевых слов и параметров.

Создать как глобальный модуль, так и модуль класса MS Access 2010 можно, не запуская среду VBA. Для этого выполните следующие действия:

1. Откройте главное окно Microsoft Access 2010.
2. Выберите вкладку **Создание**. Перейдите в раздел **Макросы и код**.
3. В правом верхнем углу раздела расположен значок **Модуль**. Щелкните по нему. Среда VBA запустится автоматически, и сразу откроется окно для ввода текста модуля.

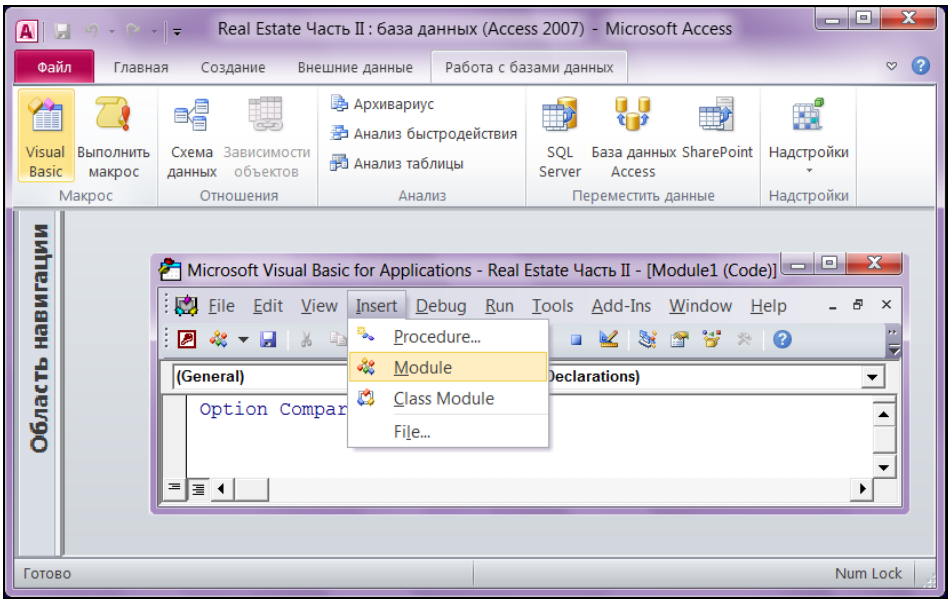


Рис. 7.3. Создание глобального модуля Microsoft Visual Basic for Applications

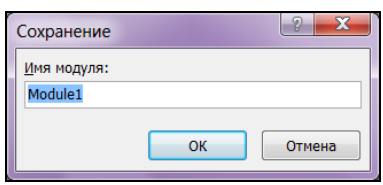


Рис. 7.4. Сохранение модуля VBA

4. Наберите текст и закройте окно VBA. Появится запрос на сохранение.
5. Введите имя модуля и нажмите кнопку **OK**.

Для завершения работы с меню и возвращения в среду Microsoft Access 2010 нам понадобится еще одна процедура. Ее текст приведен в листинге 7.2.

Листинг 7.2. Завершение работы с меню

```
Public Sub ResetMainMenu()
' Удаление главного меню программного комплекса
Dim cbar As CommandBar
' Если меню программного комплекса существует, удалить его
For Each cbar In CommandBars
      If cbar.Name = "MainMenu" Then
         CommandBars("MainMenu").Delete
         Exit For
```

```
      End If
Next cbar
End Sub
```

7.4. Изменение параметров запуска приложения

Для запуска приложения Real Estate служит форма-заставка. Ее имя — `Start`. Создание этой формы рассмотрено в *части I*. Внесем некоторые изменения в разработку. Если создается однопользовательское приложение, то щелчок по кнопке с изображением здания вполне может запускать главное меню. Код этого события приведен в листинге 7.3.

Листинг 7.3. Запуск меню программного комплекса

```
Private Sub BattonStart_Click()
    ' Кнопка начала работы
    DoCmd.Close     ' Закрытие этой формы
    StartMainMenu  ' Запуск главного меню
End Sub
```

В случае многопользовательской разработки следует перейти к назначению прав доступа. Для этой цели применяется форма `Login`. Ее создание будет рассмотрено в следующем разделе. Код вызова формы приведен в листинге 7.4.

Листинг 7.4. Запуск формы назначения прав доступа

```
Private Sub BattonStart _Click()
' Кнопка начала работы
On Error GoTo Err_ BattonStart _Click
    Dim stDocName As String
    Dim stLinkCriteria As String
    DoCmd.Close     ' Закрытие формы
    stDocName = "Login"
    DoCmd.OpenForm stDocName, , , stLinkCriteria

Exit_ BattonStart _Click:
    Exit Sub

Err_ BattonStart _Click:
    MsgBox Err.Description
    Resume Exit_ BattonStart _Click
End Sub
```

7.4.1. Установка параметров ленты и панелей инструментов

Форма `Login` должна быть запущена в модальном окне. Доступ ко всем остальным объектам программного комплекса должен быть временно запрещен. Для этого выберите в главном окне Microsoft Access 2010 вкладку **Файл**, а в левом нижнем углу открывшегося окна нажмите кнопку **Параметры**. Выбор пункта **Текущая база данных** показан на рис. 7.5.

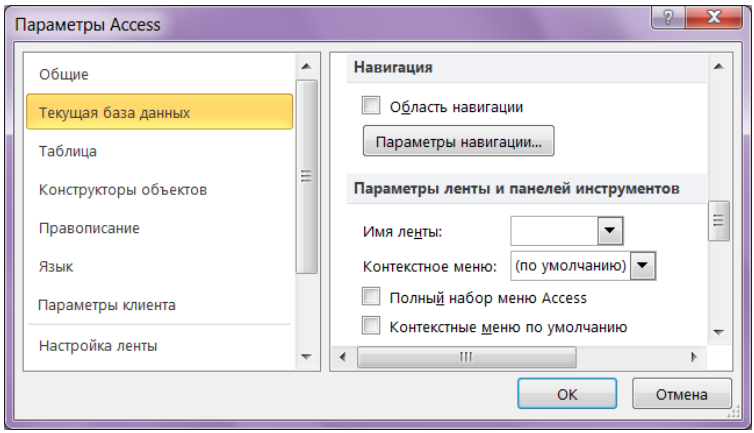


Рис. 7.5. Установка параметров ленты и панелей инструментов

Снимите все флажки. Главное меню программного комплекса примет вид, показанный на рис. 7.2.

Параметры запуска и действия, выполняющиеся при открытии приложения Microsoft Access 2010, которые определены в диалоговом окне **Параметры Access**, можно обойти, чтобы получить полный доступ к файлу Microsoft Access 2010. Для этого при открытии базы данных удерживайте нажатой клавишу <Shift>. В зависимости от параметров безопасности макросов, при загрузке появляются одно или несколько предупреждений о безопасности. Необходимо держать нажатой клавишу <Shift> до тех пор, пока не будут закрыты все окна с предупреждениями о безопасности.

7.4.2. Отключение действия клавиши <Shift>

В MS Access 2010 имеется возможность отключить клавишу обхода параметров запуска <Shift>, создав процедуру на Microsoft Visual Basic, которая задает для свойства `AllowBypassKey` базы данных значение `False`.

В примере, имеющемся в файле справки к MS Office Access 2010, приведена процедура `SetBypassProperty` (листинг 7.5), которая пытается установить свойст-

во AllowBypassKey, и если свойство не найдено, использует метод CreateProperty, чтобы добавить это свойство к семейству Properties.

Листинг 7.5. Работа со свойством AllowBypassKey

```
Sub SetBypassProperty()
Const DB_Boolean As Long = 1
    ChangeProperty "AllowBypassKey", DB_Boolean, False
End Sub

Function ChangeProperty(strPropName As String, _
    varPropType As Variant, _
    varPropValue As Variant) As Integer

    Dim dbs As Object, prp As Variant
    Const conPropNotFoundError = 3270

    Set dbs = CurrentDb
    On Error GoTo Change_Err
    dbs.Properties(strPropName) = varPropValue
    ChangeProperty = True

Change_Bye:
    Exit Function

Change_Err:
    If Err = conPropNotFoundError Then ' Property not found.
        Set prp = dbs.CreateProperty(strPropName, _
            varPropType, varPropValue)
        dbs.Properties.Append prp
        Resume Next
    Else
        ' Unknown error.
        ChangeProperty = False
        Resume Change_Bye
    End If
End Function
```

Порядок действий для отключения клавиши <Shift> при старте программного комплекса следующий:

1. Поместите процедуру SetBypassProperty в любой модуль вашего приложения. (В примере Real Estate это модуль ModuleShiftLock.)
2. Откройте его текст в редакторе VBA.

3. В окне отладки **Immediate** наберите `SetBypassProperty`.
4. Закройте базу данных.

Примечание

Отныне объекты вашей базы данных *лишь теоретически* недоступны не только для злоумышленника, но и для самого разработчика. Чтобы в этом убедиться, зайдите на любой хакерский сайт. Сразу найдете несколько утилит для возвращения значения `True` свойству `AllowBypassKey`.

7.4.3. Удаление ошибочно созданного меню

Иногда в процессе отладки созданное меню из-за ошибок остается на вкладке **Надстройки** и не удаляется даже после завершения сеанса работы с Microsoft Access 2010. Убрать строку меню достаточно просто. Сделайте по строке заголовка меню щелчок правой кнопкой мыши и в появившемся контекстном меню выберите пункт **Удалить настраиваемую панель инструментов.**

Совет

"Снос" всех подобных настроек можно автоматизировать. Для этого в процедуру загрузки стартовой формы поместите код:

```
Private Sub Form_Load()
  ' Удаление всех настроек предыдущих сеансов MS Access 2007
  Application.SetOption "ShowWindowsInTaskbar", False
End Sub
```

7.4.4. Подключение дополнительных библиотек

Если при запуске меню программного комплекса появится сообщение об ошибке (рис. 7.6) — подключите библиотеку Microsoft Office 14.0 Object Library (рис. 7.7). Для этого:

1. Выберите четвертую вкладку ленты главного окна MS Access 2007. Ее название — **Работа с базами данных.**
2. Запустите Visual Basic, щелкнув по первому значку, расположенному в разделе **Макрос**.
3. Выберите в главном меню VBA пункт **Tools**. Откроется всплывающее меню.
4. Сделайте щелчок по пункту **References**. Появится окно со списком библиотек.
5. Поставьте флажок рядом с Microsoft Office 14.0 Object Library.
6. В той же области окна находится еще одна важная библиотека для работы с объектами базы данных: Microsoft Office 14.0 Access Database Engine Object Library.
7. Отметьте ее тоже флажком.

Примечание

При следующем открытии окна со списком библиотек отмеченные компоненты появятся в самом его начале.

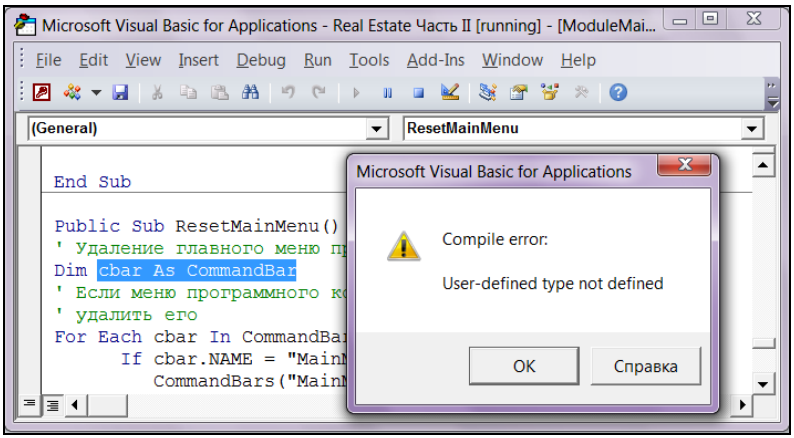


Рис. 7.6. Сообщение об ошибке во время запуска меню программного комплекса

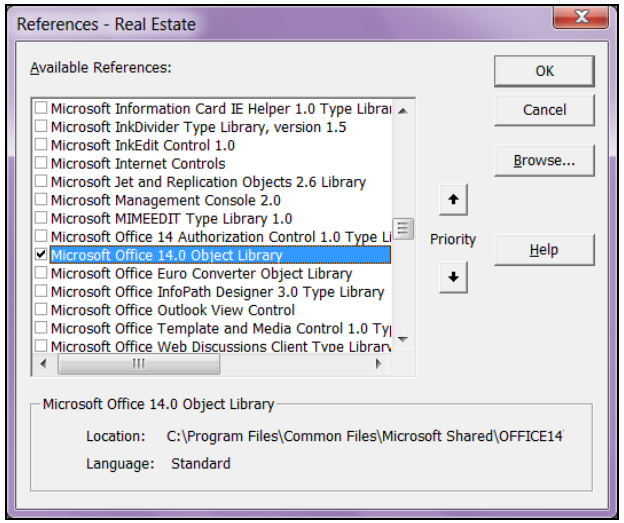


Рис. 7.7. Подключение библиотеки Microsoft Office 14.0 Object Library

7.4.5. Отключение функции блокирования процедур и макросов

Тексты на Visual Basic for Applications и макросы, хранящиеся в базе данных Microsoft Access, в силу каких-либо причин могут содержать вредоносный код.

Все приложения Microsoft Office изначально настроены на отключение макросов и процедур при запуске файла. Для правильного функционирования приложения необходимо изменить первоначальную настройку этой функции. Для этого:

1. На главной ленте Microsoft Access 2010 щелкните по вкладке **Файл**. Откроется меню.
2. Нажмите кнопку **Параметры**. Она расположена в правом нижнем углу вкладки. Появится окно **Параметры Access**.
3. В левой части появившегося окна выберите пункт **Центр управления безопасностью**. Появится диалоговое окно.
4. Нажмите кнопку **Параметры центра управления безопасностью**.
5. В появившемся окне выберите пункт **Параметры макросов**.
6. В разделе параметров (рис. 7.8) установите переключатель **Включить все макросы...**.

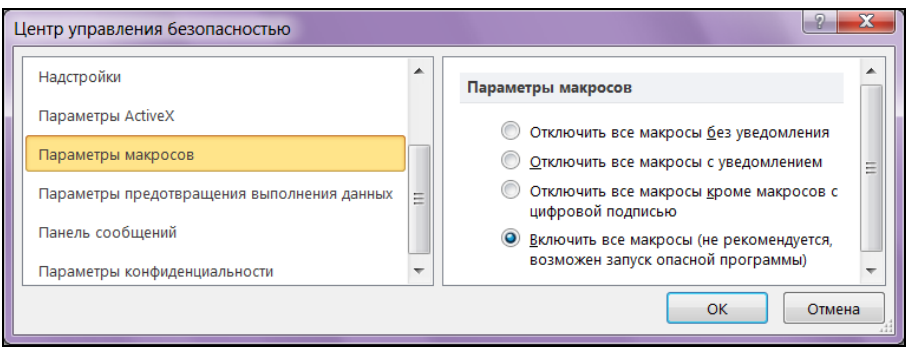


Рис. 7.8. Отключение функции блокирования макросов и процедур VBA

Предупреждение

При установке программного комплекса на компьютер заказчика не забудьте, что отключение функции блокирования процедур и макросов следует выполнить заново. Эти настройки не сохраняются в accdb-файле. Это прерогатива MS Office Access 2010.

7.5. Улучшение интерфейса приложения

Экран компьютера целый день перед глазами работника предприятия. Для удобства работы окно программного комплекса максимально раскрыто. Рабочий стол Windows 7 с его замечательными возможностями настройки не виден. Фон окна MS Access не отличается изысканностью. Унылая картина. Не правда ли? Чтобы

хоть как-то разнообразить ее, предоставим пользователю возможность выбора картинки для фона главного окна программного комплекса. На рис. 7.9 показан вид приложения Real Estate 2010 в начале его работы. Форма смены картинки запущена.

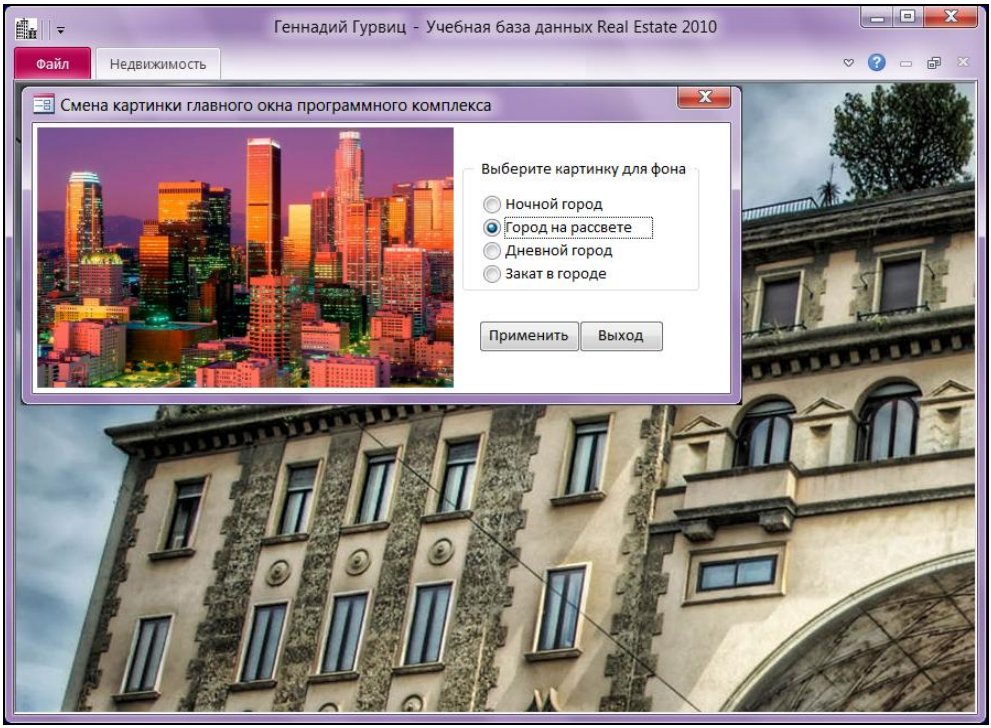


Рис. 7.9. Смена картинки главного окна программного комплекса

7.5.1. Форма назначения фона главного окна

В MS Access 2010, впрочем, как и в предыдущих версиях, отсутствует возможность размещения своих "обоев" для оформления главного окна. В данном разделе предложен способ решения этой проблемы.

- Сначала необходимо подобрать картинки и разместить их в отдельной папке. Приложение Real Estate хранит их в папке \Desktop. Картинки должны иметь достаточный размер, необходимый для полного замещения пространства главного окна. Дисплеи постоянно совершенствуются, увеличивается их разрешение, изменяется соотношение ширины и высоты, поэтому необходимо работать на перспективу. В папке \Desktop на компакт-диске вы найдете несколько картинок на тему предметной области приложения размером 1920×1200, что на момент написания раздела вполне достаточно.

- Затем следует создать новую форму любого размера без единого объекта. В программном комплексе Real Estate 2010 такая форма получила название `Back`. В табл. 7.1 приведен список только измененных свойств этой формы. Листинг 7.6 содержит код, который выполняется при загрузке формы `Back`.
- Последний этап заключается в запуске формы на выполнение. Это можно сделать после завершения работы формы `Start`, включив в самый конец кода кнопки с изображением здания (`ButtonStart`) строку: `DoCmd.OpenForm "Back"`.

Таблица 7.1. *Список измененных свойств формы* `Back`

Номер	Свойство	Значение
1	Тип границы	Отсутствует
2	Область выделения	нет
3	Кнопки навигации	нет
4	Полосы прокрутки	Отсутствуют
5	Кнопка закрытия	нет
6	Кнопки размеров окна	Отсутствуют

Листинг 7.6. Текст события *Загрузка формы* `Back`

```
Private Sub Form_Load()
  ' Картинка находится вне базы данных
  Form.PictureType = 1
  ' Путь к файлу с картинкой (Дневной город)
  Form.PICTURE = CurrentProject.Path & "\DESKTOP\Afternoon.jpg"
  ' Развернем форму на весь экран
  DoCmd.Maximize
End Sub
```

7.5.2. Форма смены картинки главного окна

Для изменения фона окна программного комплекса предназначена форма `PictureChange` (см. рис. 7.9), которая содержит четыре объекта:

- свободную рамку объекта `PictureBack` для размещения картинки в режиме предварительного просмотра;
- группу переключателей `OptionGroup` с названиями доступных картинок;
- кнопку **Применить** `CommandSave` для переноса картинки в главное окно программного комплекса;
- кнопку **Выход** `CommandExit` для завершения работы формы.

Измененных свойств в форме смены картинки главного окна немного. Необходимо убрать область выделения, кнопки навигации, полосы прокрутки и кнопку размеров окна.

В листинге 7.7 приведен код, выполняющийся при загрузке формы `PictureChange`.

Листинг 7.7. Текст события *Загрузка формы* `PictureChange`

```
Private Sub Form_Load()
  ' Назначение файла справки
  Me.HelpFile = CurrentProject.Path & "\RealEstate.chm"
  ' Дневной город
  Form.PictureBack.PICTURE = CurrentProject.Path & _
              "\DESKTOP\Afternoon.jpg"
End Sub
```

Свободная рамка объекта должна иметь размеры с коэффициентом пропорциональности картинки 1920/1200 = 1,6. Для объекта `PictureBack` в свойствах указана ширина 8 см, а высота 5 см. Среди измененных свойств присутствуют также **Установка размеров** — по размеру рамки и **Выравнивание рисунка** — по центру.

Группа переключателей `OptionGroup` создается при помощи мастера. После создания к переключателям необходимо добавить код события **После обновления**, приведенный в листинге 7.8.

Листинг 7.8. Текст события *После обновления* объекта `OptionGroup`

```
Private Sub OptionGroup_AfterUpdate()
' Изменение свойства PICTURE объекта PictureBack формы PictureChange
Select Case OptionGroup.Value
   Case 1   ' Ночной город
     PictureBack.PICTURE = CurrentProject.Path & _
                 "\DESKTOP\Night.jpg"
   Case 2   ' Город на рассвете
     PictureBack.PICTURE = CurrentProject.Path & _
                 "\DESKTOP\BeforeDawn.jpg"
   Case 3   ' Дневной город
     PictureBack.PICTURE = CurrentProject.Path & _
                 "\DESKTOP\Afternoon.jpg"
   Case 4   ' Закат в городе
     PictureBack.PICTURE = CurrentProject.Path & _
                 "\DESKTOP\SunSet.jpg"
End Select
End Sub
```

Последний листинг 7.9 содержит код, который запускается на выполнение при нажатии кнопки **Применить**.

Листинг 7.9. Текст события *Нажатие кнопки* объекта `CommandSave`

```
Private Sub CommandSave_Click()
Select Case OptionGroup.Value
' Изменение свойства PICTURE формы Back
   Case 1   ' Ночной город
     Forms!Back.PICTURE = CurrentProject.Path & _
                 "\DESKTOP\Night.jpg"
   Case 2   ' Город на рассвете
     Forms!Back.PICTURE = CurrentProject.Path & _
                 "\DESKTOP\BeforeDawn.jpg"
   Case 3   ' Дневной город
     Forms!Back.PICTURE = CurrentProject.Path & _
                 "\DESKTOP\Afternoon.jpg"
   Case 4   ' Закат в городе
     Forms!Back.PICTURE = CurrentProject.Path & _
                 "\DESKTOP\SunSet.jpg"
End Select
End Sub
```

Листинги 7.8 и 7.9 используют значение свойства `OptionGroup.Value` группы переключателей, в котором сохраняется номер выбранного переключателя. По умолчанию в нем установлен номер 3, соответствующий картинке "Дневной город". Она находится в файле Afternoon.jpg. Листинги практически идентичны за исключением того, что первый изменяет картинку в окне предварительного просмотра, а второй — в главном окне MS Access, в нашем случае в форме `Back`.

ГЛАВА 8

Обеспечение информационной безопасности приложения

Чем большую ценность представляет собой информация, находящаяся в распоряжении предприятия, тем острее стоят вопросы его доверия к своим сотрудникам и тем совершеннее должны быть средства отслеживания доступа к корпоративным информационным ресурсам, бесконтрольное использование которых может причинить серьезный вред. Любое предприятие не может сегодня успешно функционировать без создания надежной системы защиты информации, включающей не только организационно-нормативные меры, но и технические программно-аппаратные средства контроля. Все решения по защите от несанкционированного доступа и установление личной ответственности работников принимаются руководителем предприятия, т. к. именно он по действующему ныне законодательству несет в первую очередь ответственность за утрату конфиденциальной информации. Не сомневайтесь — ответственность за доступ к информационной системе предприятия будет возложена на вас. При защите информационной системы от несанкционированного доступа могут быть использованы два принципа управления доступом к защищаемым ресурсам: дискреционный и мандатный.

8.1. Дискреционный принцип управления доступом

Каждому зарегистрированному пользователю устанавливаются права доступа к объектам системы, которые прописываются в правилах разграничения доступа (ПРД). Эти правила должен утвердить руководитель предприятия. Обязательно побеспокойтесь об этом. Данный вариант управления доступом позволяет для любого пользователя системы создать изолированную программную среду, т. е. ограничить его возможности по запуску программ, указав в качестве разрешенных к запуску только те программы, которые действительно необходимы для выполнения пользователем своих служебных обязанностей. Таким образом, программы, не входящие в этот список, пользователь запустить не сможет.

8.2. Мандатный принцип управления доступом

Принцип управления доступом к ресурсам основан на сопоставлении уровня конфиденциальности каждого ресурса и полномочий конкретного зарегистрированного пользователя по доступу к ресурсам информационной системы с заданным уровнем конфиденциальности.

Для организации мандатного управления доступом для каждого пользователя системы устанавливается некоторый уровень допуска к конфиденциальной информации (уровень отдела, уровень департамента, уровень дирекции и т. д.), а каждому ресурсу присваивается так называемая метка конфиденциальности или код доступа. При этом разграничение доступа к информации осуществляется путем сравнения уровня допуска пользователя и метки конфиденциальности ресурса и принятии решения о предоставлении или не предоставлении доступа.

В рассматриваемом нами примере Real Estate реализован дискреционный принцип управления доступом (формы `Login` и `Employee`). Внимательно изучив работу форм, вы без больших переделок сможете реализовать мандатный принцип управления доступом. Просто уберите доступ любого пользователя (даже администратора) к объектам типа **Флажок** ☑ (вторая страница формы `Employee`).

Наряду с контролем доступа система защиты программного комплекса должна содержать подсистему учета сделанных пользователем изменений. Один из способов реализации этой подсистемы вы найдете в *разд. 8.5* с описанием работы формы `Employee`.

8.3. Форма контроля доступа к приложению

Как только пользователь получит право доступа к работе с программным комплексом, ему автоматически предоставляются различные привилегии. Они могут включать разрешение на доступ к определенным таблицам, формам, запросам, отчетам и расчетам, которые выполняет комплекс. Привилегии предоставляются работникам для того, чтобы они могли решать задачи, входящие в круг их должностных обязанностей. К нарушению защиты может привести предоставление излишних полномочий, поэтому каждому работнику администратор должен назначить только те привилегии, без которых его работа невозможна.

Научимся делать доступными каждому пользователю только те пункты меню и кнопки, с которыми ему положено работать в соответствии с правилами разграничения доступа (ПРД), существующими на предприятии. Для этих целей в программном комплексе Real Estate предназначена форма `Login`. Эта форма запускается на выполнение до активации меню программного комплекса и либо вообще запрещает регистрацию работника, если его данных (фамилия и пароль) нет в базе, либо ограничивает доступ в соответствии с ПРД.

Рассмотрим форму `Login` в процессе работы (рис. 8.1). Перед вами самый распространенный вид формы, созданной в MS Access. Обратите внимание на то, что до завершения процесса регистрации главное меню программного комплекса отсутствует. От ленты MS Access 2010 осталась одна вкладка **Главная** и та — в недоступном состоянии.

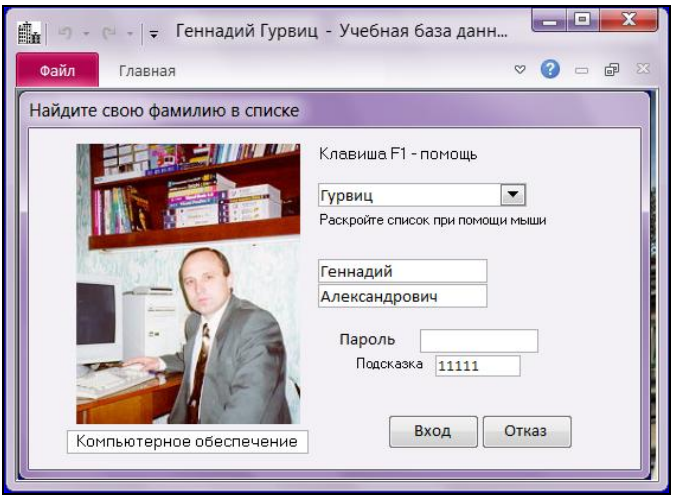


Рис. 8.1. Форма контроля доступа к программному комплексу

Панель быстрого доступа также погашена. Это сделали настройки параметров Access (см. рис. 7.5) и значение *Да* свойства **Всплывающее окно** формы `Login`. Полностью отказываться от ленты не стоит. Пункты **Фильтр**, **Найти**, **Орфография** становятся доступными после завершения работы формы `Login` и могут быть с успехом применены при работе с программным комплексом. Да и работа с буфером обмена Windows тоже не помешает.

Для создания формы выберите на второй вкладке ленты MS Access 2010 **Создание** пункт **Конструктор форм**. Откроется окно конструктора с именем новой формы `Form1`. Процесс создания формы самым подробным образом был рассмотрен в *части I*.

Сейчас откройте уже готовую форму `Login` в конструкторе (рис. 8.2). Найдите в области навигации учебной базы данных Real Estate 2010 в разделе форм имя `Login`. Щелкните по нему правой кнопкой мыши. Появится контекстное меню. Вторым в нем должен быть пункт **Конструктор**. Выберите его. На экране появится форма в режиме конструктора. Щелчок правой кнопкой мыши в свободном от изображения формы месте приведет к появлению еще одного контекстного меню. В нем следует выбрать последний пункт **Свойства**. В появившемся окне свойств увидите пять вкладок: **Макет**, **Данные**, **События**, **Другие** и **Все**.

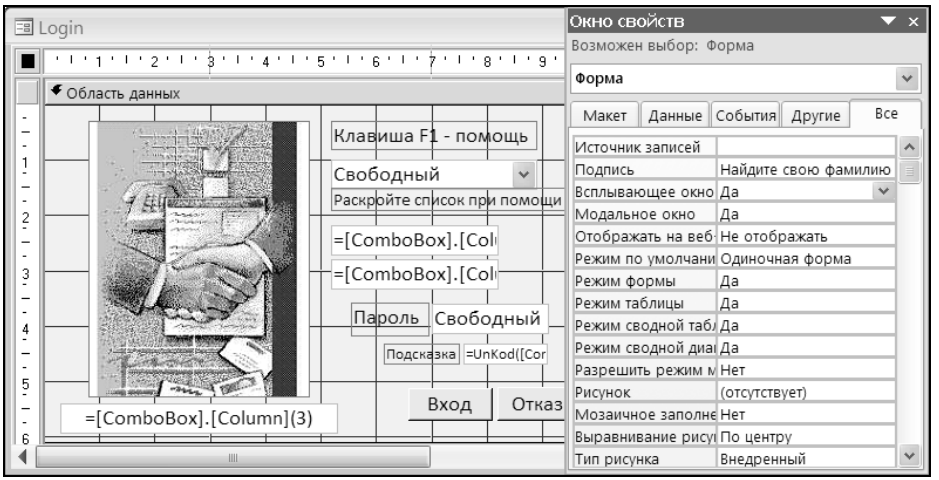


Рис. 8.2. Форма контроля доступа к программному комплексу в режиме конструктора

В окне свойств несколько десятков строчек. Все содержат значения по умолчанию. В нашем случае следует изменить всего несколько из них. Список измененных свойств приведен в табл. 8.1.

Таблица 8.1. *Список измененных свойств формы* `Login`

Номер	Свойство	Значение
1	**Подпись**	*Найдите свою фамилию в списке*
2	**Всплывающее окно**	*Да*
3	**Модальное окно**	*Да*
4	**Отображать на веб-узле SharePoint**	*Не отображать*
5	**Разрешить режим макета**	*Нет*
6	**Область выделения**	*Нет*
7	**Кнопки перехода**	*Нет*
8	**Полосы прокрутки**	*Отсутствуют*
9	**Кнопки размеров окна**	*Отсутствуют*
10	**Кнопка закрытия**	*Нет*
11	**Контекстные меню**	*Нет*

К несомненным достоинствам MS Access при работе с окном свойств следует отнести русифицированный интерфейс разработчика, а к недостаткам — отсутствие выделения свойств объектов, которые были изменены разработчиком.

В MS Access постоянно наступают различные события, которые можно считать полезными лишь в том случае, если знаешь, как на них реагировать и в каком порядке эти события выполняются. Более того, каждый тип элементов управления Access имеет свой набор событий. Обработать событие и отреагировать на него позволяет вкладка **События** окна свойств формы.

При загрузке формы выполняется следующая цепочка событий:

1. **Открытие** (`Form_Open`).
2. **Загрузка** (`Form_Load`).
3. **Изменение размера** (`Form_Resize`).
4. **Включение** (`Form_Activate`).
5. **Получение фокуса** (`Form_Got Focus`).

Для двух из них: **Открытие** (`Form_Open`) и **Загрузка** (`Form_Load`) написан код процедуры обработки событий, который запускается на выполнение при наступлении события. Это листинги 8.1 и 8.2.

Листинг 8.1. Текст события *Открытие* формы `Login`

```
Private Sub Form_Open(Cancel As Integer)
  ' Инициализация глобальных переменных,
  ' предназначенных для управления доступом к приложению
  ' Запуск процедуры
  Adjustment ' Процедура из модуля ModuleMain
End Sub
```

Листинг 8.2. Текст события *Загрузка* формы `Login`

```
Private Sub Form_Load()
  ' Назначение файла справки
  Me.HelpFile = CurrentPath() & "\RealEstate.chm"
End Sub
```

RealEstate.chm — имя файла контекстно-зависимой справки к приложению. Функция `CurrentPath()` возвращает полный путь к этому файлу и описана в предыдущей главе. Текст процедуры `Adjustment` находится в модуле `ModuleMain` и приведен в листинге 8.3.

Листинг 8.3. Текст процедуры `Adjustment`

```
Public Sub Adjustment()
  ' Начальные значения глобальных переменных
  FAMILY = "Разработчик комплекса"
```

```
  SuperVisor = False          ' Признак идентификации
  ChangePicture = False       ' Смена картинки
  ChangePassword = False      ' Смена пароля
  SetDeleted = False          ' Удаленные записи
  CountRecords = False        ' Объем базы данных
  RightAccess = False         ' Права доступа
  SeekBuilding = False        ' Поиск зданий
  AddBuilding = False         ' Добавление зданий
  WorkFlats = False           ' Работа с квартирами
  AccountWork = False         ' Лицевые счета
  WordExcel = False           ' Отчеты
  StreetTown = False          ' Адресный план
  DistrictTown = False        ' Работа с районами
  MaterialWall = False        ' Материал стен
  Staff = False               ' Администрирование
End Sub
```

Форма `Login` работает с таблицей `tblUser`, которая включена в базу данных Real Estate 2010. Состав полей таблицы `tblUser` приведен в табл. 8.2. Первичный ключ таблицы создан при помощи связки полей `LastName + FirstName + SecondName`. В основе его создания лежит предпосылка, что на предприятии не могут работать люди с полностью совпадающими фамилией, именем и отчеством. Если это вас не устраивает, то воспользуйтесь дополнительным полем — счетчик.

Таблица 8.2. *Состав таблицы пользователей*

№	Поле	Тип	Размер	Описание
1	LastName	**Текстовый**	15	Фамилия пользователя
2	FirstName	**Текстовый**	12	Имя
3	SecondName	**Текстовый**	10	Отчество
4	Post	**Текстовый**	25	Занимаемая должность
5	PassWord	**Текстовый**	10	Зашифрованное значение пароля
6	File	**Текстовый**	8	Имя файла с фотографией
7	Access01	**Логический**	1	Настройка картинки в главном окне
8	Access02	**Логический**	1	Возможность смены своего пароля
9	Access03	**Логический**	1	Показ удаленных записей
10	Access04	**Логический**	1	Просмотр объема базы данных
11	Access05	**Логический**	1	Просмотр своих прав доступа
12	Access06	**Логический**	1	Возможность поиска зданий
13	Access07	**Логический**	1	Возможность добавления зданий

***Таблица 8.2** (окончание)*

№	Поле	Тип	Размер	Описание
14	Access08	**Логический**	1	Возможность работы с квартирами
15	Access09	**Логический**	1	Работа с лицевыми счетами
16	Access10	**Логический**	1	Возможность работы с отчетами
17	Access11	**Логический**	1	Работа с адресным планом
18	Access12	**Логический**	1	Работа со списком районов
19	Access13	**Логический**	1	Доступ к материалу стен зданий
20	Access14	**Логический**	1	Установка прав доступа всем работникам
21	Inspector	**Текстовый**	15	Фамилия предоставившего доступ
22	Date_up	**Дата/время**	8	Дата последней корректировки
23	Time_up	**Текстовый**	10	Время последней корректировки
24	Range	**Числовой**	1	Типовой код доступа

Значения полей Access01—Access14 присваиваются глобальным переменным в результате работы формы Login, и на основании того, False это или True, обеспечивается доступ к пунктам меню, формам и расчетам программного комплекса.

Доступ к пункту меню обеспечивается использованием одной-единственной строчки. В следующем примере таких строчек две (по числу пунктов меню), и они выделены жирным шрифтом:

```
With .Add(Type:=msoControlPopup)
     .Caption = "Справочники"
     With .Controls
          With .Add(Type:=msoControlButton)
               .Caption = "Улицы города"
               .Enabled = StreetTown
               .OnAction = "Street"
          End With
          With .Add(Type:=msoControlButton)
               .Caption = "Районы города"
               .Enabled = DistrictTown
               .OnAction = "District"
          End With
     End With
End With
```

Управление доступом к справочнику улиц обеспечивает глобальная переменная StreetTown, а к справочнику районов — DistrictTown.

Доступ к объектам формы обеспечивает это же свойство — `Enabled`, только установленное для объекта формы. В следующем примере показано, как "связать" кнопку занесения новой записи `CommandAdd` с глобальной переменной `AddBuilding`. Установите это свойство при загрузке формы:

```
Private Sub Form_Load()
  CommandAdd.Enabled = AddBuilding
End Sub
```

Для выбора фамилии работника нам понадобится элемент управления формы **Поле со списком**. Выберите на панели значок  , а в нужном месте активной области формы при помощи левой кнопки мыши отведите место для этого объекта.

Предупреждение

Необходимо создать свободный элемент управления. Он не должен быть связан ни с одним полем таблицы.

В табл. 8.3 приведен список свойств объекта **ComboBox**. Для его создания не запускайте построитель. Нам понадобится запрос по всем полям таблицы `tblUser`. Всего полей — 24. Построитель сообщит, что работа возможна лишь с двадцатью. Просто напишите в свойстве **Источник строк** фразу `Select * from tbtlUser` — и проблема решена. В таблице показаны только измененные свойства.

Таблица 8.3. *Список измененных свойств поля со списком*

Свойство	Значение
Имя	ComboBox
Ширина списка	*5см*
Ширина столбцов	*2,5см;2,5см*
Источник строк	*Select * from tblUser*
Число строк списка	*10*
Всплывающая подсказка	*Раскройте список при помощи мыши*
Ограничиться списком	*Да*
Только значения источника строк	*Да*
Переход по TAB	*Да*

В результате запроса в поле со списком будут загружены значения всех полей таблицы `tblUser`. Практически все они и нужны в нашем случае, за исключением даты и времени последней корректировки пароля (поля 22 и 23). Но и они могут понадобиться, если разработчик желает предусмотреть смену пароля через определенный промежуток времени. Показывать же на экране необходимо лишь имя

и фамилию (1 и 2 столбец). Управлять отображением позволяет свойство **Ширина столбцов**. Если столбец не нужен на экране — поставьте *0см* в месте его появления. В нашем случае можно поступить еще проще. Укажите ширину списка — *5см*, а ширину столбцов: *2,5см;2,5см*. Дальнейшие нули писать не надо. Остальные поля будут просто отсечены.

После выбора в списке фамилии работника необходимо заполнить данными о нем (имя, отчество, должность, фото и т. д.) другие управляющие элементы формы. Здесь нам поможет номер колонки поля со списком **ComboBox.**

Для текстового поля `txtFirstName` (имя работника) в качестве значения свойства **Данные** укажите:

```
=[ComboBox].[Column](1)
```

Колонки поля со списком MS Access нумерует с нуля. Нулевая колонка — фамилия, первая — имя, вторая — отчество и т. д. Теперь имя работника — связанный элемент. Как только в поле со списком **ComboBox** будет выбран очередной работник, в текстовом поле `txtFirstName` появится его имя.

Для отображения фотографии работника в объекте формы (свободная рамка объекта) с именем `PictureUser` в качестве свойства **Данные** поставьте:

```
=CurrentPath() & "\PHOTO\" & [ComboBox].[Column](5) & ".jpg"
```

Опишем подробнее составные части этого выражения:

- функция `CurrentPath()` возвращает полный путь к файлу базы данных и описана в предыдущей главе;
- PHOTO — папка, в которой хранятся фотографии работников;
- в пятой колонке поля со списком содержится значение шестого поля таблицы `tblUser`. Это поле `File` (имя файла с фотографией);
- JPG — формат файла изображения.

Для того чтобы при запуске формы `Login` список фамилий сразу был раскрыт, сделайте следующее:

1. На вкладке **Конструктор** главной ленты MS Access 2010 выберите значок  **Переходы** для изменения последовательности перехода по элементам управления формы.
2. В появившемся диалоговом окне **Последовательность перехода** поставьте первым элемент **ComboBox**. Он первым получит фокус после активации формы.
3. Перейдите в окно свойств этого элемента.
4. Выберите вкладку **События**.
5. Сделайте активным событие **Получение фокуса**. Текст процедуры на VBA приведен в листинге 8.4.

Листинг 8.4. Обработка события *Получение фокуса*

```
Private Sub ComboBox_GotFocus()
   ' Используется метод Dropdown
   ' для раскрытия поля со списком
   ComboBox.Dropdown
End Sub
```

Текстовый элемент `txtHint` (подсказка) связан с четвертой колонкой поля со списком:

```
=UnKod([ComboBox].[Column](4))
```

В реальном приложении он без сомнения должен быть опущен. В учебном примере содержит расшифрованный пароль выбранного пользователя.

Совет

Не удаляйте этот объект из формы. Просто погасите его. Для этого в качестве значения свойства **Вывод на экран** поставьте *Нет*. Свойство расположено на вкладке **Макет**.

Пароль пользователя хранится в зашифрованном виде в поле `PassWord` таблицы `tblUser`. Для его зашифровки используется функция `CrKod()`, а для расшифровки — `UnKod()`. Тексты функций приведены в листингах 8.5 и 8.6.

Листинг 8.5. Зашифровка пароля

```
Public Function CrKod(cPassword) As String
   ' Возвращает зашифрованный пароль
   ' cPassword — незашифрованный пароль (входной параметр)
   Dim cLetter As String             ' Один символ пароля
   Dim cEncryptedPassword As String  ' Зашифрованный пароль
   Dim i As Integer                  ' Параметр цикла
   If Not IsNull(cPassword) Then
      ' Пустой пароль не зашифровывается
      ' Убираем концевые пробелы в пароле
      cPassword = Trim(cPassword)
      ' Начальное значение зашифрованного пароля
      cEncryptedPassword = ""
      For i = 1 To Len(cPassword)
          ' Отделяем очередной символ
          cLetter = Mid(cPassword, i, 1)
          cEncryptedPassword = cEncryptedPassword + _
                    Chr(Asc(cLetter) — i)
      Next i
   End If
```

```
   ' Возвращаемое значение зашифрованного пароля
   CrKod = cEncryptedPassword
End Function
```

Листинг 8.6. Расшифровка пароля

```
Public Function UnKod(cEncryptedPassword) As String
   ' Возвращает расшифрованный пароль
   ' cEncryptedPassword — зашифрованный пароль
   ' (входной параметр)
   Dim cLetter As String             ' Один символ пароля
   Dim cPassword As String           ' Расшифрованный пароль
   Dim i As Integer                  ' Параметр цикла
   If Not IsNull(cEncryptedPassword) Then
      ' Пустой пароль не расшифровывается
      ' Убираем концевые пробелы в зашифрованном пароле
      cEncryptedPassword = Trim(cEncryptedPassword)
      ' Начальное значение расшифрованного пароля
      cPassword = ""
      For i = 1 To Len(cEncryptedPassword)
          ' Отделяем очередной символ
          cLetter = Mid(cEncryptedPassword, i, 1)
          cPassword = cPassword + Chr(Asc(cLetter) + i)
      Next i
   End If
   ' Возвращаемое значение расшифрованного пароля
   UnKod = cPassword
End Function
```

В тексте используются стандартные функции VBA:

- `Mid()` — возвращает подстроку из строки, начиная с символа `i` длиной в один символ;
- `Chr()` — возвращает символ, заданный ANSI-кодом;
- `Asc()` — возвращает ANSI-код символа;
- `Trim()` — удаляет из строки начальные и конечные пробелы.

Функции реализуют шифрование пароля методом замены. Заменяется каждый символ пароля. При замене учитывается позиция символа в строке.

Текстовый элемент `txtParole` (пароль) — свободный элемент управления. Предназначен для ввода пароля. Его отличительной особенностью является свойство **Маска ввода**. Откройте список значений этого свойства и установите при помощи окна построителя **Создание масок ввода** маску **Пароль**. При работе с формой `Login` каждый символ пароля будет отображаться символом "звездочка" (`*`).

Последний этап — добавление в форму двух кнопок: **Вход** и **Отказ**. Для этого:

1. Откройте форму `Login` в режиме конструктора.
2. Сделайте щелчок левой кнопкой мыши по пиктограмме  **Кнопка** на панели элементов.
3. Наведите указатель мыши на то место формы, где вы планируете поместить левый верхний угол кнопки. Указатель мыши превратится в значок кнопки с крестиком в левом верхнем углу.
4. Нажмите левую кнопку мыши и, удерживая ее в нажатом состоянии, переместите курсор по диагонали так, чтобы получилась рамка требуемого размера. Отпустите левую кнопку мыши.
5. Если запустится построитель кнопки — прекратите его работу. Текст обработки события `Click` (щелчок по кнопке) придется написать без его помощи.

Листинги 8.7 и 8.8 содержат коды обработки события для кнопок **Вход** и **Отказ** соответственно.

Листинг 8.7. Код события `Click` кнопки *Вход*

```
Private Sub Кнопка2_Click()
' Кнопка Вход
Dim Parole As String          ' Пароль из базы данных
Dim ParoleEnter As String     ' Введенный пароль
If IsNull([Forms]![Login]![ComboBox].Column(0)) Then
   ' Если фамилия не выбрана — Null.
   ' Вернуть работу в форму
   MsgBox "Найдите свою фамилию в списке.", _
       vbOKOnly + vbExclamation, "Внимание"
   Exit Sub
End If
' Значение пароля из таблицы tblUser
' содержится в пятой колонке объекта ComboBox.
' Нумерация колонок начинается с нуля
Parole = [Forms]![Login]![ComboBox].Column(4)
' Удаляем концевые пробелы
Parole = Trim(Parole)
If Len(Trim(Parole)) = 0 Then
   ' Если в таблице tblUser был стерт пароль
   ' (например, через ODBC)
   SuperVisor = False   ' Идентификация не выполнена
   MsgBox "Пароль в таблице идентификации отсутствует." _
       & Chr(13) & "Операция сравнения паролей " & _
       "не может быть выполнена", _
       vbOKOnly + vbExclamation, "Внимание"
   Application.Quit acQuitSaveAll ' Выходим из приложения
```

```
Else
   If IsNull([Forms]![Login]![txtParole]) Then
      ' В поле введенного пароля — Null
      MsgBox "Вы забыли ввести пароль.", _
              vbOKOnly + vbExclamation, "Внимание"
      ' Установка курсора в поле ввода пароля и возврат в форму
      [Forms]![Login]![txtParole].SetFocus
      Exit Sub
   End If
   ' Удаление концевых пробелов
   ParoleEnter = Trim([Forms]![Login]![txtParole])
   ' Зашифровка введенного пароля
   ParoleEnter = CrKod(ParoleEnter)
   ' Фамилия работника
   FAMILY = [Forms]![Login]![ComboBox].Column(0)
   If ParoleEnter = Parole Then
      SuperVisor = True   ' Идентификация выполнена
      ' Права доступа к пунктам меню и кнопкам форм.
      ' Смена картинки главного окна программного комплекса
      ChangePicture = [Forms]![Login]![ComboBox].Column(6)
      ' Возможность изменять пароль
      ChangePassword = [Forms]![Login]![ComboBox].Column(7)
      ' Возможность просмотра удаленных записей
      SetDeleted = [Forms]![Login]![ComboBox].Column(8)
      ' Информация о заполнении таблиц
      CountRecords = [Forms]![Login]![ComboBox].Column(9)
      ' Просмотр своих прав доступа
      RightAccess = [Forms]![Login]![ComboBox].Column(10)
      ' Сложный поиск зданий
      SeekBuilding = [Forms]![Login]![ComboBox].Column(11)
      ' Занесение здания
      AddBuilding = [Forms]![Login]![ComboBox].Column(12)
      ' Возможность обработки квартир
      WorkFlats = [Forms]![Login]![ComboBox].Column(13)
      ' Работа с лицевым счетом
      AccountWork = [Forms]![Login]![ComboBox].Column(14)
      ' Работа с отчетами
      WordExcel = [Forms]![Login]![ComboBox].Column(15)
      ' Работа с адресным планом
      StreetTown = [Forms]![Login]![ComboBox].Column(16)
      ' Работа с районами
      DistrictTown = [Forms]![Login]![ComboBox].Column(17)
      ' Материал стен зданий
      MaterialWall = [Forms]![Login]![ComboBox].Column(18)
```

```
      ' Работа с пользователями
      Staff = [Forms]![Login]![ComboBox].Column(19)
      MsgBox "Инспектор " & Trim(FAMILY) & " ! Доступ разрешен", _
             vbOKOnly + vbExclamation, "Результат идентификации"
   Else
      SuperVisor = False ' Идентификация не выполнена
      MsgBox "Инспектор " & Trim(FAMILY) & " ! Доступ запрещен", _
             vbOKOnly + vbExclamation, "Результат идентификации"
      Application.Quit acQuitSaveAll ' Выходим из приложения
   End If
End If
DoCmd.Close    ' Закрытие этой формы
StartMainMenu  ' Запуск главного меню
End Sub
```

Если работник щелкает по кнопке **Вход**, не выбрав свою фамилию в поле со списком, то в нулевой колонке **ComboBox** в качестве фамилии содержится значение `Null`. Процедура прекращает обработку и возвращает управление в форму (`Exit Sub`). Следует сказать, что сделать это пользователю довольно трудно, т. к. кнопки **Вход** и **Отказ** закрыты списком работников. В нем 10 строк, и он находится в раскрытом состоянии (метод `Dropdown`). Но рано или поздно кто-то из служащих предприятия обязательно сделает это, закрыв список, так ничего в нем и не выбрав.

Для исключения возможности банального удаления пароля злоумышленником в таблице `tblUser` (заметим, что способов получения доступа к таблице MS Access предостаточно) производится проверка длины пароля пользователя. Если пароль отсутствует, то работа программного комплекса прекращается. "Подсматривать" пароль через ODBC не имеет смысла, т. к. в таблице он зашифрован, а вводить его в форму требуется в раскодированном виде.

Пароль, указанный пользователем, зашифровывается и сравнивается с хранящимся в таблице. В случае совпадения все глобальные переменные, управляющие доступом к пунктам меню, формам и отчетам, получают значения, соответствующие полям `Access01`—`Access14` таблицы `tblUser`. Далее форма закрывается (`DoCmd.Close`) и запускается процедура активации меню программного комплекса (`StartMainMenu`). Эта процедура находится в модуле `ModuleMain`.

Листинг 8.8. Код события `Click` кнопки *Отказ*

```
Private Sub Кнопка1_Click()
   ' Кнопка Выход — отказ от регистрации.
   ' Завершение работы комплекса
   StopMenu ' Процедура находится в модуле ModuleSlave
End Sub
```

8.4. Форма изменения пароля

В правилах разграничения доступа (ПРД), действующих на предприятии или в организации, должен быть указан срок действия пароля пользователя. По истечении этого срока программный комплекс должен предложить работнику сменить пароль. Для этих целей создадим форму `PassWord`. Она может быть запущена работником на выполнение раньше истечения контрольного срока (пункт меню **Поддержка | Смена пароля**). Форма `PassWord` показана на рис. 8.3. В табл. 8.4 приведен список только измененных свойств этой формы.

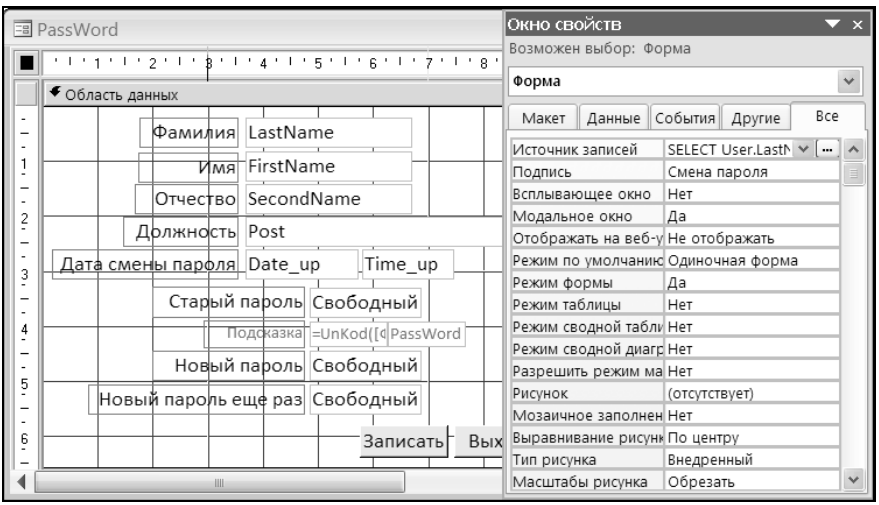


Рис. 8.3. Форма `PassWord` в режиме конструктора

Таблица 8.4. *Список свойств формы* `PassWord`

Номер	Свойство	Значение
1	**Источник записей**	`SELECT tblUser.LastName, tblUser.FirstName, tblUser.SecondName, tblUser.Post, tblUser.PassWord, tblUser.Date_up, User.Time_up FROM [tblUser] WHERE (((tblUser.LastName)=GetValueSQL()));`
2	**Подпись**	*Смена пароля*
3	**Модальное окно**	*Да*
4	**Отображать на веб-узле SharePoint**	*Не отображать*
5	**Режим таблицы**	*Нет*
6	**Режим сводной таблицы**	*Нет*

Таблица 8.4 *(окончание)*

Номер	Свойство	Значение
7	**Режим сводной диаграммы**	*Нет*
8	**Разрешить режим макета**	*Нет*
9	**Область выделения**	*Нет*
10	**Кнопки перехода**	*Нет*
11	**Полосы прокрутки**	*Отсутствуют*
12	**Кнопки размеров окна**	*Отсутствуют*
13	**Кнопка закрытия**	*Нет*
14	**Контекстные меню**	*Нет*

Форма смены пароля `PassWord` работает с запросом, в который помещаются данные о работнике, запустившем программный комплекс на выполнение (рис. 8.4). Для его создания:

1. Сделайте щелчок по кнопке [...], которая расположена в области данных свойства **Источник записей**. Запустится построитель запроса.

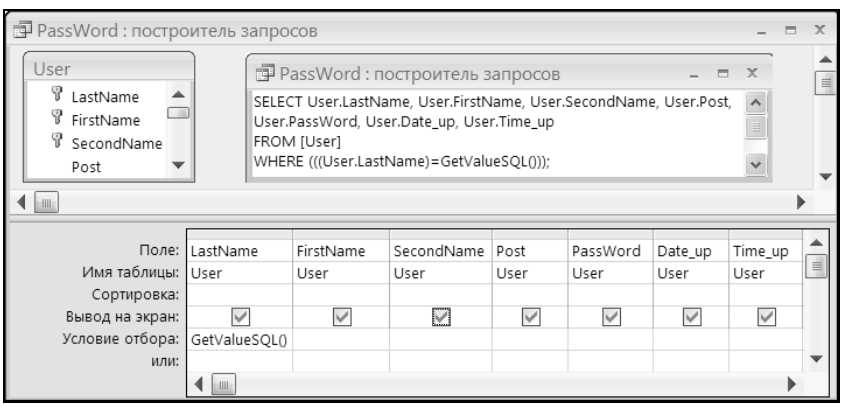


Рис. 8.4. Создание запроса для формы `PassWord`

2. Добавьте в запрос источник. Это таблица `tblUser`.
3. Расположите в нижней части окна построителя поля: `LastName`, `FirstName`, `SecondName`, `Post`, `PassWord`, `Date_Up` и `Time_Up`. Остальные поля таблицы не нужны.

4. В условии отбора по полю `LastName` (фамилия) укажите пользовательскую функцию `GetValueSQL()`.
5. Сохраните запрос.

Предупреждение

Построитель MS Access 2010 не разрешает использовать в запросе имена переменных. Он автоматически заключает их в кавычки, превращая, таким образом, в символьные константы.

Вот почему в условии отбора по полю `LastName` вместо глобальной переменной `FAMILY` (фамилия текущего пользователя) мы вынуждены поставить функцию `GetValueSQL()`. Она находится в модуле `ModuleSlave`, а текст ее представлен в листинге 8.9. В дальнейшем мы не будем применять этот "трюк" ввиду его громоздкости. Visual Basic for Applications предоставляет несколько других и гораздо более легких способов включения переменных в запрос. Один из них показан в конце листинга 8.10. Это запрос на модификацию, но пусть это вас не смущает. Принцип включения имен переменных в запрос на выборку и удаление точно такой же.

Листинг 8.9. Пользовательская функция

```
Public Function GetValueSQL()
   ' Возвращает значение переменной FAMILY
   ' Фамилия текущего пользователя
   ' Значение требуется вставить в SQL-запрос
   ' Напрямую это сделать MS Access не разрешает
   GetValueSQL = FAMILY
End Function
```

Окончательный вид формы показан на рис. 8.5.

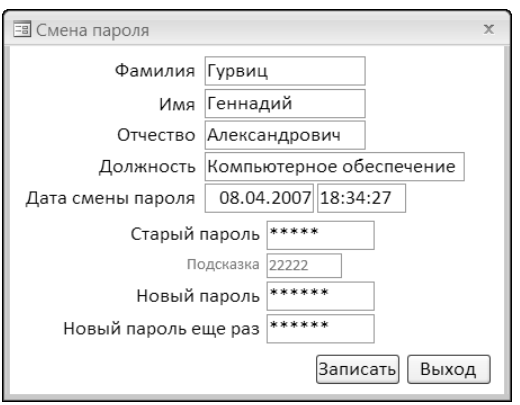


Рис. 8.5. Форма `PassWord`

Добавим код, который будет запущен на выполнение при наступлении события `Click` кнопки **Записать** формы `PassWord` (листинг 8.10).

Листинг 8.10. Код события *Нажатие кнопки*

```
Private Sub Кнопка2_Click()
  ' Кнопка Записать
  Dim OldPassword As String     ' Старый пароль
  Dim NewPasswordOne As String  ' Новый пароль
  Dim NewPsswordTwo As String   ' Новый пароль повторно
  Dim TextSQL As String         ' Текст SQL-запроса
  ' При ошибке перейти к метке: SqlUpdateErr
  On Error GoTo SqlUpdateErr
  If IsNull([Forms]![PassWord]![PassWord]) Then
    ' Старый пароль не введен
    MsgBox "Вы забыли ввести старый пароль.", _
           vbOKOnly + vbExclamation, "Внимание"
    ' Установка курсора в поле ввода старого пароля
    [Forms]![PassWord]![OldPassword].SetFocus
    ' Возврат в форму
    Exit Sub
  End If
  OldPassword = [Forms]![PassWord]![OldPassword]
  ' Удаляем концевые пробелы
  OldPassword = Trim(OldPassword)
  ' Зашифровка старого пароля
  OldPassword = CrKod(OldPassword)
  If OldPassword <> Trim([Forms]![PassWord]![CrPassWord]) Then
    ' Старый пароль введен неправильно
    MsgBox "Неправильно введен старый пароль.", _
           vbOKOnly + vbExclamation, "Внимание"
    ' Установка курсора в поле ввода старого пароля
    [Forms]![PassWord]![OldPassword].SetFocus
    ' Возврат в форму
    Exit Sub
  End If
  If IsNull([Forms]![PassWord]![NewPassword1]) Then
    ' Новый пароль не введен
    MsgBox "Вы забыли ввести новый пароль.", _
           vbOKOnly + vbExclamation, "Внимание"
    ' Установка курсора в поле ввода нового пароля
    [Forms]![PassWord]![NewPassword1].SetFocus
    ' Возврат в форму
    Exit Sub
  End If
```

```
If IsNull([Forms]![PassWord]![NewPassword2]) Then
  ' Подтверждение нового пароля не введено
  MsgBox "Вы забыли подтвердить новый пароль.", _
         vbOKOnly + vbExclamation, "Внимание"
  ' Установка курсора в поле подтверждения
  [Forms]![PassWord]![NewPassword2].SetFocus
  ' Возврат в форму
  Exit Sub
End If
' Удаление концевых пробелов в значениях пароля
NewPasswordOne = Trim([Forms]![PassWord]![NewPassword1])
NewPsswordTwo = Trim([Forms]![PassWord]![NewPassword2])
If NewPasswordOne <> NewPsswordTwo Then
  MsgBox "Два значения нового пароля не совпадают.", _
         vbOKOnly + vbExclamation, "Внимание"
  ' Установка курсора в поле подтверждения
  [Forms]![PassWord]![NewPassword2].SetFocus
  ' Возврат в форму
  Exit Sub
End If
If Len(NewPasswordOne) <= 4 Then
  MsgBox "Длина пароля — не менее 5 символов.", _
         vbOKOnly + vbExclamation, "Внимание"
  ' Установка курсора в поле нового пароля
  [Forms]![PassWord]![NewPassword1].SetFocus
  ' Возврат в форму
  Exit Sub
End If
' Зашифровка нового пароля
NewPasswordOne = CrKod(NewPasswordOne)
If CrKod(NewPasswordOne) = _
  Trim([Forms]![PassWord]![CrPassWord]) Then
  MsgBox "Этот пароль уже использовался.", _
         vbOKOnly + vbExclamation, "Внимание"
  ' Установка курсора в поле ввода старого пароля
  [Forms]![PassWord]![NewPassword1].SetFocus
  ' Возврат в форму
  Exit Sub
End If
' Формирование текста SQL-запроса
TextSQL = "UPDATE tblUser " & _
          "SET tblUser.PassWord = '" & NewPasswordOne & "'," & _
          "tblUser.Date_Up=Date(),tblUser.Time_Up=Time()" & _
          "WHERE tblUser.LastName = '" & FAMILY & "' "
' Debug.Print TextSQL
' Выполнение запроса
DoCmd.RunSQL TextSQL
```

```
  ' Закрытие формы
  DoCmd.Close
  Exit Sub

SqlUpdateErr:
  If Err.Number = 2501 Then
      MsgBox "Смена пароля не выполнена.", _
             vbOKOnly + vbExclamation, "Внимание"
  Else
     ' Сообщить при любой другой ошибке
     MsgBox "Возникла ошибка:" & vbCrLf & Err.Description & _
            vbCrLf & "Номер " & Err.Number, vbCritical
     Resume Next
  End If
End Sub
```

В тексте процедуры используются стандартные функции VBA:

- `IsNull()` — возвращает `True`, если значение не присвоено (аргумент `Null`);
- `Len()` — возвращает число символов в строке;
- `Trim()` — удаляет из строки начальные и конечные пробелы.

После запуска процедура выполняет семь проверок:

1. Введен ли старый пароль?
2. Совпадает ли он с имеющимся в базе?
3. Введен ли новый пароль?
4. Введено ли подтверждение значения нового пароля?
5. Совпадают ли два значения нового пароля?
6. Имеет ли новый пароль достаточную длину?
7. Не совпадает ли старый пароль с вновь введенным?

Этот джентльменский набор значительно добавляет интеллектуальности процедуре смены пароля, повышая информационную безопасность программного комплекса. В реальном приложении он может быть даже несколько расширен добавлением блокировки учетных записей пользователей после некоторого числа неудачных попыток ввода пароля.

Примечание

При проектировании процедуры регистрации следует оценить влияние системы защиты на конечных пользователей и выяснить:

- не приведет ли требование использовать более длинные пароли к тому, что их будут чаще хранить на бумаге, прикрепленной к дисплею компьютера?

- не вызовет ли политика защиты потерю эффективности труда и рост нагрузки на администратора системы, который только и будет заниматься сбросом забытых паролей?

Только после четкого представления влияния этих факторов на работу конкретного предприятия можно приступить к разработке программного кода.

После выполнения проверок введенное значение пароля зашифровывается (функция `CrKod()`) и формируется текст SQL-запроса на изменение данных в таблице `tblUser`. В него помещаются три значения: сам пароль, а также дата и время записи в базу данных.

Запрос модифицирует одну-единственную запись таблицы, в которой хранятся данные текущего пользователя. Его фамилия находится весь сеанс в глобальной переменной `FAMILY`.

SQL-запрос запускается на выполнение средствами VBA. Используется метод `RunSQL`. После запуска на экране появится системное сообщение (рис. 8.6).

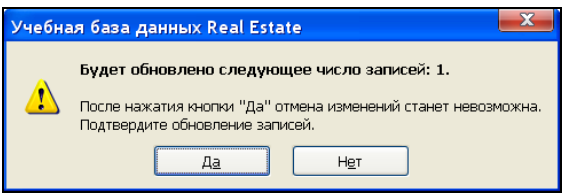


Рис. 8.6. Сообщение перед выполнением запроса

Если пользователь выберет для ответа кнопку **Нет**, то Microsoft Visual Basic выдаст сообщение об ошибке с номером 2501 (рис. 8.7). Оно для конечного пользователя абсолютно лишнее. Более того, дополнительные кнопки: **End** и **Debug** могут завести его в тупик. Поэтому в рассматриваемой процедуре предусмотрен целенаправленный перехват ошибки с номером 2501 и всех остальных, которые могут возникнуть при ее работе. Для этого после описания переменных в текст добавлена строка, предписывающая перейти к оператору, отмеченному меткой, в случае возникновения ошибки:

```
On Error GoTo SqlUpdateErr
```

где `SqlUpdateErr` — метка.

Этот оператор помещен в самый конец процедуры и формирует сообщение программного комплекса, а не системы. При работе в штатном режиме управление ему вообще не передается, т. к. перед ним размещена строка выхода из процедуры: `Exit Sub`.

Теперь вместо окна **Microsoft Visual Basic** (см. рис. 8.7) появится сообщение (рис. 8.8). "Заблудиться" в программном комплексе пользователь не сможет (дополнительные кнопки отсутствуют), но текст, по-прежнему, малопонятен. Добавим оператор условного перехода для разделения ошибки 2501 и всех остальных,

которые теоретически могут возникнуть. О них будет сообщать окно (рис. 8.8), а о том, что прервано выполнение макрокоманды `RunSQL` — окно (рис. 8.9).

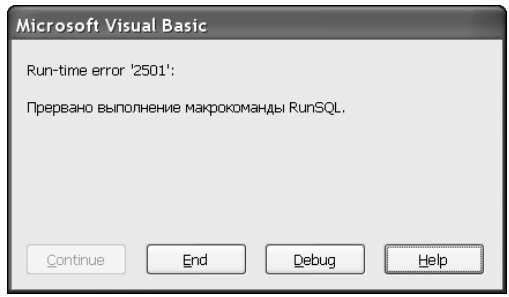


Рис. 8.7. Реакция системы на отказ от выполнения смены пароля

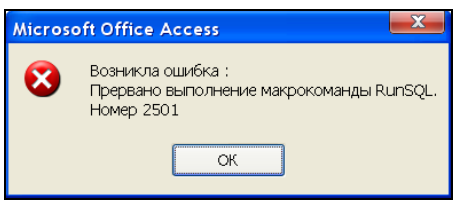


Рис. 8.8. Перехват системной ошибки с указанием ее содержания и номера

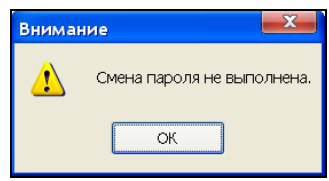


Рис. 8.9. Целенаправленный перехват ошибки 2501

Совет

Для отладки процедур программного комплекса, связанных с перехватом и обработкой ошибок, воспользуйтесь методом `Raise` объекта `Err`, который генерирует ошибку с кодом, заданным параметром `Number`. Описание ошибки задает второй параметр `Description` этого метода. Параметр `Description` можно опустить.

В следующем примере создается фиктивная ошибка отказа принтера. Вместо сообщения системы "Printer error" генерируется текст "Принтер не готов".

```
Err.Raise Number:=482, Description:="Принтер не готов"
```

Вставьте эту строчку именно в том месте кода программы, где хотите увидеть отказ принтера и "разобраться" с этой не существующей реально ошибкой.

В MS Office Access 2010 имеется возможность отказаться от запроса системы на подтверждение изменения записей в таблице. Для этого:

1. На главной ленте MS Access щелкните по вкладке **Файл**. Появится окно, внизу которого сделайте щелчок по кнопке **Параметры**.
2. В появившемся окне **Параметры Access** выберите пункт **Параметры клиента** (рис. 8.10).

3. В правой части окна снимите флажок **Подтверждение запросов на изменение**.

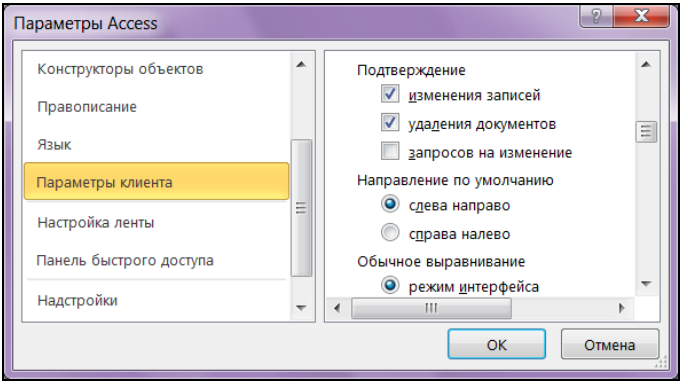


Рис. 8.10. Изменение стандартных настроек работы MS Access 2010

Последний штрих при создании формы `PassWord` (смены пароля пользователя) — запретите конечному пользователю изменять значения полей с фамилией, именем, отчеством, должностью и датой корректировки. Для этого выделите объект или группу целиком и воспользуйтесь свойствами, приведенными в табл. 8.5.

Таблица 8.5. *Список свойств доступа к объектам формы*

Свойство	Вкладка	Пояснение
Доступ (Enable)	**Данные**	*Нет* — гашение объекта на экране в режиме формы. Объект отображается серым цветом
Блокировка (Lock)	**Данные**	*Да* — запрет изменения данных в режиме формы
Вывод на экран (Visible)	**Макет**	*Нет* — объект в режиме формы не отображается

Совет

Используйте значение *Нет* свойства **Вывод на экран** для подсказки пароля. Не удаляйте его совсем после отладки. Подсказка может пригодиться для "выяснения" отношений с неквалифицированным, слабо владеющим клавиатурой, но излишне настойчивым пользователем. Просто сделайте ее на некоторое время видимой.

8.5. Форма назначения прав доступа к приложению

Форма `Employee` предназначена только для администратора программного комплекса. Именно этому работнику предприятия предстоит реализовать правила

доступа каждого сотрудника к пунктам меню, формам, отчетам и другим объектам работы с информационной базой.

Существует великое множество вариантов оформления подобных форм. Рассмотрим один из них — двухстраничную форму. На первой ее странице поместим список работников предприятия, а на второй — подробности по работнику, фамилия которого выбрана на первой странице.

Запустим конструктор форм. Создадим новую форму и откроем окно свойств. В табл. 8.6 приведены измененные свойства формы `Employee`.

Таблица 8.6. *Свойства формы* `Employee`

№	Свойство	Вкладка	Значение
1	**Подпись**	**Макет**	*Работники предприятия*
2	**Отображать на веб-узле SharePoint**	**Другие**	*Не отображать*
3	**Режим таблицы**	**Макет**	*Нет*
4	**Режим сводной таблицы**	**Макет**	*Нет*
5	**Режим сводной диаграммы**	**Макет**	*Нет*
6	**Разрешить режим макета**	**Макет**	*Нет*
7	**Область выделения**	**Макет**	*Нет*
8	**Кнопки перехода**	**Макет**	*Нет*
9	**Полосы прокрутки**	**Макет**	*Отсутствуют*
10	**Кнопки размеров окна**	**Макет**	*Отсутствуют*
11	**Контекстные меню**	**Другие**	*Нет*
12	**Выравнивание по центру**	**Макет**	*Да*
13	**Тип границы**	**Макет**	*Тонкая*

Разместим в форме набор вкладок (страниц). Воспользуемся для этого элементом управления  **Вкладка**. Он расположен в разделе **Элементы управления** ленты на вкладке **Конструктор**. Вначале MS Access 2010 создаст набор из двух вкладок. В дальнейшем их количество можно изменить. В нашем случае этого не потребуется. Также непосредственно в самой форме разместим кнопку **Выход**. Она будет видна на любой вкладке. В табл. 8.7 приведен список объектов формы.

Совет

Создайте кнопку рядом с набором вкладок и передвиньте ее в область этого набора.

Таблица 8.7. *Список объектов формы* `Employee`

Объект			Имя объекта	Пояснение	Данные
Набор вкладок (имя объекта — **PageFrame**)	**Вкладка** (**Page1**) Список работников	**Список**	`ListBox`	Работники предприятия	Запрос
		Кнопка	`CommandAdd`	Добавить	Код VBA
		Поле	`txtCountWorker`	Число записей	Функция `DCount()`
	Вкладка (**Page2**) Подробности	**Поле**	`txtLastName`	Фамилия	`LastName`
		Поле	`txtFirstName`	Имя	`FirstName`
		Поле	`txtSecondName`	Отчество	`SecondName`
		Поле	`txtPost`	Должность	`Post`
		Поле	`txtFile`	Файл с фото	`File`
		Поле	`txtPassWord`	Пароль	`PassWord`
		Группа переключателей	`OptionGroup`	Типовой набор прав доступа	`Range`
		Свободная рамка объекта	`PictureUser`	Фотография работника	Функция
		Флажок	`ChkAccess01`	Права доступа	`Access01`
		...	...		...
		Флажок	`chkAccess14`		`Access14`
		Поле	`txtInspector`	Фамилия инспектора	`Inspector`
		Поле	`txtDate_up`	Дата корректировки	`Date_up`
		Поле	`txtTime_up`	Время	`Time_up`
		Кнопка	`CommadeSave`	Сохранить	Код VBA
		Кнопка	`CommandDel`	Удалить	Код VBA
		Кнопка	`CommandDelPassword`	Стереть пароль	Код VBA
Кнопка			`CommandExit`	Выход	Код VBA

Перейдем на первую вкладку формы и создадим на ней при помощи элемента управления  список `ListBox`. Свойства, которые нужно установить для списка, приведены в табл. 8.8.

Таблица 8.8. *Свойства объекта* `ListBox`

Номер	Свойство	Вкладка	Значение
1	**Имя**	**Другие**	`ListBox`
2	**Источник строк**	**Данные**	`SELECT tblUser.LastName, tblUser.FirstName, tblUser.SecondName, tblUser.Post FROM tblUser;`
3	**Присоединенный столбец**	**Данные**	*1*
4	**Число столбцов**	**Макет**	*4*
5	**Ширина столбцов**	**Макет**	*2,5см;2,3см;3см;4см*

В качестве источника данных объекта `ListBox` используется запрос. В него включены только те поля из таблицы `tblUser`, которые будут отображены в списке. Поле `txtCounWorker`, в котором будет отображаться количество записей в таблице `tblUser` в качестве свойства данные (вкладка **Макет**), должно иметь значение:

```
=DCount("*";"tblUser")
```

Стандартная функция `Dcount()` возвратит общее их количество. Вид первой вкладки создаваемой формы в конструкторе приведен на рис. 8.11.

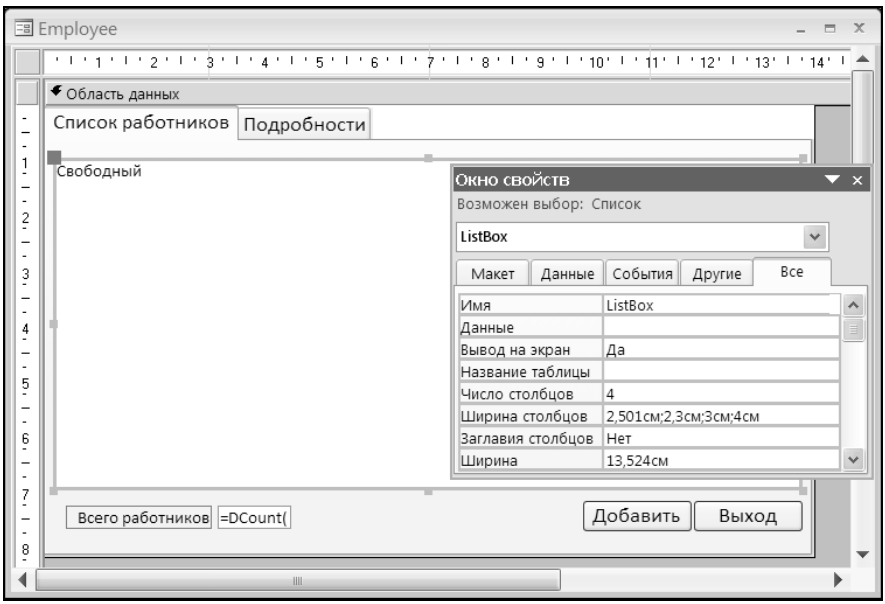


Рис. 8.11. Первая вкладка формы `Employee` в конструкторе

В листинге 8.11 приведен код обработки события загрузки формы. Для работы формы нам понадобятся четыре глобальные переменные. Они описаны как `Public` в модуле `ModuleMain`:

```
Public SelectLastName As String     ' Выбранная фамилия
Public SelectFirstName As String    ' Выбранное имя
Public SelectSecondName As String   ' Выбранное отчество
Public Indicator As Integer         ' Указатель действий
```

При загрузке формы эти переменные получат начальные значения.

Листинг 8.11. Код события *Загрузка* формы `Employee`

```
Private Sub Form_Load()
  ' Загрузка формы
  Indicator = 0          ' Указатель действий
  ' Indicator=1            Режим добавления записи
  ' Indicator=2            Режим корректировки записи
  ' Indicator=3            Выход без сохранения изменений
  ' Фамилия выбранного работника
  SelectLastName = ""
  ' Имя выбранного работника
  SelectFirstName = ""
  ' Отчество выбранного работника
  SelectSecondName = ""
End Sub
```

На рис. 8.12 показан вид первой вкладки формы `Employee` в процессе работы.

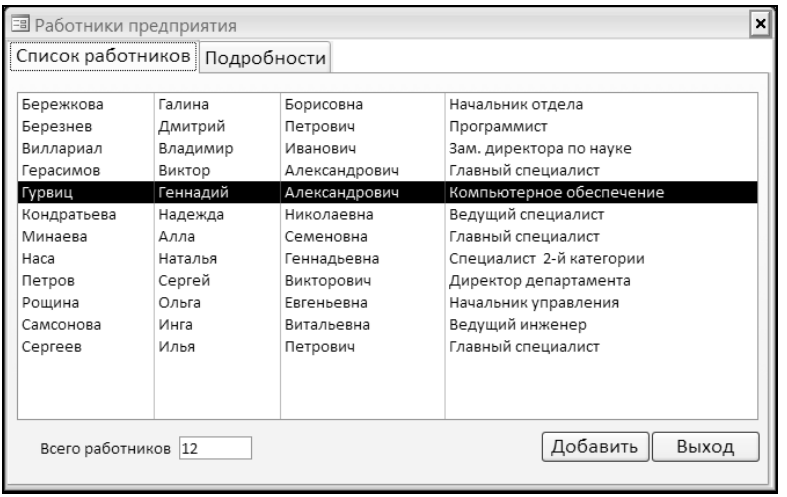


Рис. 8.12. Первая вкладка формы `Employee` в режиме формы

Выполнив щелчок мышью по фамилии нужного работника, администратор программного комплекса попадет на вторую вкладку формы. Код события `Click` приведен в листинге 8.12.

Листинг 8.12. Код события *Нажатие кнопки* объекта `ListBox`

```
Private Sub ListBox_Click()
Dim TextSQL As String ' Строка запроса
Indicator = 2         ' Режим корректировки включен
' Смотри событие "До обновления" этой формы Form_BeforeUpdate.
' Если среди работников нет однофамильцев,
' можно работать только с фамилией
' TextSQL = "Select * from tblUser where LastName = '" & _
' ListBox.Value & "'"
' Фамилия выбранного работника
SelectLastName = [ListBox].Column(0)
' Имя выбранного работника
SelectFirstName = [ListBox].Column(1)
' Отчество выбранного работника
SelectSecondName = [ListBox].Column(2)
TextSQL = "Select * from tblUser where LastName = '" & _
           SelectLastName & "'" & _
           " AND FirstName = '" & SelectFirstName & "'" & _
           " AND SecondName = '" & SelectSecondName & "'"
Me.RecordSource = TextSQL
Page2.Visible = True
Page2.SetFocus
' Чтобы запретить корректировку ФИО,
' снимите символ комментария с трех следующих строк
' txtLastName.Locked = True
' txtFirstName.Locked = True
' txtSecondName.Locked = True
End Sub
```

В коде формируется текст запроса на выборку только одной записи из таблицы `tblUser`, которая соответствует выбранному сотруднику. Если программный комплекс создается из расчета, что сотрудников с одной и той же фамилией в организации не будет, то при создании запроса можно воспользоваться свойством `Value` объекта `ListBox`. Для этого необходимо поставить *1* в поле **Присоединенный столбец** объекта `ListBox`. Первый столбец — это фамилия. В случае с однофамильцами необходимо использовать ключ таблицы `tblUser`, в качестве которого выступает связка полей:

```
LastName + FirstName + SecondName
```

а значения для поиска содержат переменные `SelectLastName`, `SelectFirstName` и `SelectSecondName`. Как присвоить значения этим переменным? Воспользуемся свойством `Column` объекта `ListBox`. Нулевая колонка свойства — фамилия (первый столбец объекта), первая колонка свойства — имя работника (второй столбец объекта) и вторая колонка свойства — отчество (третий столбец объекта):

```
SelectLastName = [ListBox].Column(0)
SelectFirstName = [ListBox].Column(1)
SelectSecondName = [ListBox].Column(2)
```

Вид второй вкладки формы `Employee` показан на рис. 8.13.

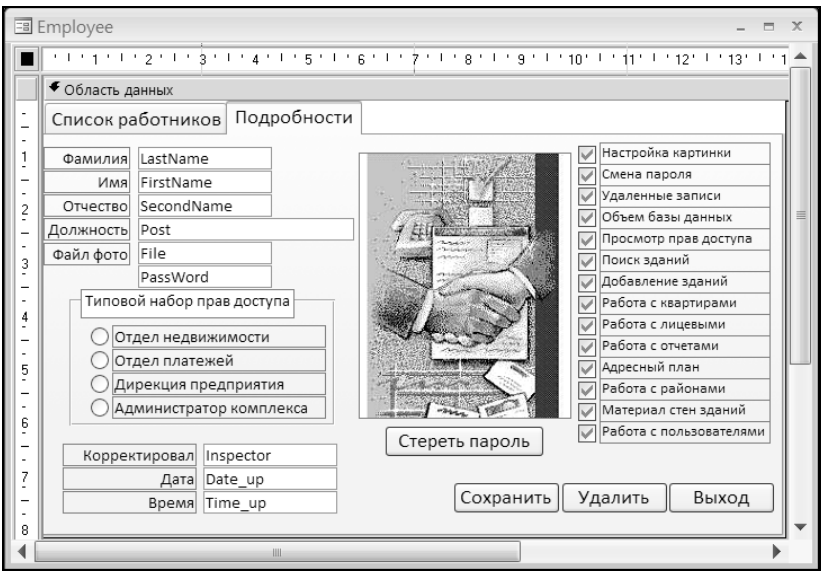


Рис. 8.13. Вторая вкладка формы `Employee` в режиме конструктора

Создание объектов **Поле**, **Флажок**, **Кнопка**, **Переключатель** и т. д. подробно описано в *части I*. Остановимся лишь на тех особенностях, которые свойственны объектам только этой формы. Измененные свойства объектов приведены в табл. 8.9.

Таблица 8.9. *Свойства объектов второй вкладки формы* `Employee`

Объект	Свойство	Вкладка	Значение
`PassWord`	**Вывод на экран**	**Макет**	*Нет*
`OptionGroup`	**Значение по умолчанию**	**Данные**	*0*
`txtInspector`	**Блокировка**	**Данные**	*Да*
`txtDate_up`	**Блокировка**	**Данные**	*Да*

Таблица 8.9 *(окончание)*

Объект	Свойство	Вкладка	Значение
txtTime_up	**Блокировка**	**Данные**	*Да*
PictureUser	**Данные**	**Данные**	=CurrentPath() & "\PHOTO\" & [txtFile].Value & ".jpg"

Вторая вкладка формы работает с обновляемым запросом. Изменение данных в полях формы повлечет изменение значений соответствующей записи в таблице tblUser при наступлении события **До обновления** (Form_BeforeUpdate) этой формы. Запретим пользователю изменять поля txtDate_up и txtTime_up, поставив *Да* для свойства **Блокировка**.

Значение свойства **Значение по умолчанию** объекта **Переключатель** (OptionGroup) равно *1*. Это означает, что если пользователь не изменял выбор типового набора прав доступа, то в базу данных попадет "Отдел недвижимости". Это неправильно. Поставим в качестве значения *0* — будет легко отследить пустое поле при добавлении нового пользователя. На рис. 8.14 показан вид второй вкладки формы в процессе работы программного комплекса.

Рис. 8.14. Вторая вкладка формы Employee в режиме формы

Форма Employee дает возможность реализовать как дискреционный, так и мандатный принцип управления доступом к приложению. В листинге 8.13 приведен код события **После обновления** объекта OptionGroup. Код реализует мандатный принцип управления. Запустите форму, выберите категорию (мандат) и глобальные переменные, связанные с объектами chkAccess01—chkAccess14, автоматически получат соответствующие значения. После вмешательства администратора

программного комплекса (ручная установка флажков) будет реализован дискреционный принцип.

Листинг 8.13. Код события *После обновления* объекта `OptionGroup`

```
Private Sub OptionGroup_AfterUpdate()
' Общие права доступа для всех категорий работников
chkAccess01.Value = True   ' Настройка картинки
chkAccess02.Value = True   ' Смена пароля
chkAccess03.Value = False  ' Удаленные записи
chkAccess04.Value = False  ' Объем базы данных
chkAccess05.Value = True   ' Просмотр прав доступа
chkAccess06.Value = False  ' Поиск зданий
chkAccess07.Value = False  ' Добавление зданий
chkAccess08.Value = False  ' Работа с квартирами
chkAccess09.Value = False  ' Работа с лицевыми
chkAccess10.Value = False  ' Работа с отчетами
chkAccess11.Value = False  ' Адресный план
chkAccess12.Value = False  ' Работа с районами
chkAccess13.Value = False  ' Материал стен зданий
chkAccess14.Value = False  ' Работа с пользователями
' Права доступа для отдельных подразделений
Select Case OptionGroup.Value
   Case 1   ' Отдел недвижимости
     chkAccess06.Value = True ' Поиск зданий
     chkAccess07.Value = True ' Добавление зданий
     chkAccess08.Value = True ' Работа с квартирами
     chkAccess10.Value = True ' Работа с отчетами
     chkAccess11.Value = True ' Адресный план
     chkAccess13.Value = True ' Материал стен
   Case 2   ' Отдел платежей
     chkAccess06.Value = True ' Поиск зданий
     chkAccess08.Value = True ' Работа с квартирами
     chkAccess09.Value = True ' Работа с лицевыми
   Case 3   ' Дирекция предприятия
     chkAccess06.Value = True ' Поиск зданий
     chkAccess08.Value = True ' Работа с квартирами
     chkAccess10.Value = True ' Работа с отчетами
    Case 4   ' Администратор комплекса
     chkAccess03.Value = True ' Удаленные записи
     chkAccess04.Value = True ' Объем базы данных
     chkAccess14.Value = True ' Изменение прав
End Select
End Sub
```

Листинг 8.14. Код события *Нажатие кнопки* объекта `CommandSave`

```
Private Sub CommandSave_Click()
' Кнопка сохранить
Dim NewPasswordOne As String ' Пароль работника
NewPasswordOne = "00000"     ' Пять нулей
If IsNull(txtLastName.Value) Then
   ' Фамилия работника не введена
   MsgBox "Вы забыли ввести фамилию.", _
       vbOKOnly + vbExclamation, "Внимание"
   ' Установка курсора в поле ввода фамилии
   txtLastName.SetFocus
   ' Возврат в форму
   Exit Sub
End If
If IsNull(txtFirstName.Value) Then
   ' Имя работника не введено
   MsgBox "Вы забыли ввести имя работника.", _
       vbOKOnly + vbExclamation, "Внимание"
   ' Установка курсора в поле ввода имени
   txtFirstName.SetFocus
   ' Возврат в форму
   Exit Sub
End If
If IsNull(txtSecondName.Value) Then
   ' Отчество работника не введено
   MsgBox "Вы забыли ввести отчество работника.", _
       vbOKOnly + vbExclamation, "Внимание"
   ' Установка курсора в поле ввода отчества
   txtSecondName.SetFocus
   ' Возврат в форму
   Exit Sub
End If
If IsNull(txtPost.Value) Then
   ' Должность работника не указана
   MsgBox "Вы забыли ввести должность работника.", _
       vbOKOnly + vbExclamation, "Внимание"
   ' Установка курсора в поле ввода должности
   txtPost.SetFocus
   ' Возврат в форму
   Exit Sub
End If
If IsNull(txtFile.Value) Then
   ' Имя файла с фотографией не указано
   MsgBox "Вы забыли ввести имя файла.", _
       vbOKOnly + vbExclamation, "Внимание"
```

```
   ' Установка курсора в поле ввода файла
   ' Имя картинки Нет фото
   txtFile.Value = "NoFoto"
   txtFile.SetFocus
   ' Возврат в форму
   Exit Sub
End If
If OptionGroup.Value = 0 Then
   ' Типовой набор прав не указан
   MsgBox "Вы забыли ввести набор прав доступа.", _
       vbOKOnly + vbExclamation, "Внимание"
   ' Установка курсора в поле ввода прав доступа
   OptionGroup.SetFocus
   ' Возврат в форму
   Exit Sub
End If
txtInspector.Value = FAMILY
txtDate_up = Date
txtTime_up = Time()
If Page2.Caption = "Занесение нового работника" Then
   ' Для нового работника — пароль пять нулей
   NewPasswordOne = CrKod(NewPasswordOne)
   txtPassWord.Value = NewPasswordOne
End If
DoCmd.Close    ' Закрытие формы
End Sub
```

При написании кода для кнопки **Удалить** (листинг 8.15) реализованы некоторые особенности работы конечного пользователя. Учтена ситуация, когда пользователь, начав корректировку ключевых полей (фамилия, имя или отчество), вдруг решает удалить из списка сведения о работнике. В этом случае в качестве параметров запроса на удаление нельзя использовать данные из объектов `txtLastName`, `txtFirstName` и `txtSecondName`. Воспользуемся значениями из глобальных переменных `SelectLastName`, `SelectFirstName` и `SelectSecondName`. Они содержат фамилию, имя и отчество работника до начала корректировки.

Листинг 8.15. Код события *Нажатие кнопки* объекта `CommandDel`

```
Private Sub CommandDel_Click()
' Кнопка Удалить
Dim TextSQL As String ' Строка запроса
' При ошибке перейти к метке: SqlUpdateErr
On Error GoTo SqlUpdateErr
TextSQL = "DELETE from tblUser where LastName = '" & _
           SelectLastName & "'" & _
```

```
          " AND FirstName = '" & SelectFirstName & "'" & _
          " AND SecondName = '" & SelectSecondName & "'"
' Выполнение запроса
DoCmd.RunSQL TextSQL
' См. событие "До обновления" этой формы — Form_BeforeUpdate
Indicator = 3
DoCmd.Close    ' Закрытие формы
SqlUpdateErr:
   If Err.Number = 2501 Then
      MsgBox "Удаление работника не выполнено.", _
      vbOKOnly + vbExclamation, "Внимание"
   End If
End Sub
```

Процедура формирует текст запроса на удаление. Текст содержит строка `TextSQL`.

SQL-запрос запускается на выполнение средствами VBA. Используется метод `RunSQL`. В процедуре обрабатывается ошибка с номером 2501. Она возникнет в том случае, если пользователь откажется от подтверждения своих действий, нажав кнопку **Нет** в ответ на запрос системы. Значение глобальной переменной `Indicator` равно 3. Выход из формы (при этом значении) пропустит обработку события **До обновления**.

Особое место среди кнопок, размещенных на второй странице формы, занимает кнопка **Стереть пароль** (листинг 8.16). Она нужна только администратору программного комплекса в случае, если работник забыл свой пароль. Подсказать забытый пароль пользователю администратор не может. Это сделано в целях безопасности (излишние полномочия администратору не нужны), а вот удалить и предоставить возможность работнику занести новый пароль администратор обязан.

Листинг 8.16. Код события *Нажатие кнопки* объекта `CommandDelPassWord`

```
Private Sub CommandDelPassword_Click()
Dim TextSQL As String        ' Строка запроса
Dim NewPasswordOne As String ' Новый пароль
NewPasswordOne = "00000"     ' Пять нулей
' Зашифровка пяти нулей
NewPasswordOne = CrKod(NewPasswordOne)
' Формирование текста SQL-запроса
' SelectLastName, SelectFirstName, SelectSecondName -
' глобальные переменные. Они получили значения
' при обработке события DblClick поля со списком ListBox
If MsgBox("Вы действительно хотите стереть пароль? ", _
      vbInformation + vbYesNo, "Внимание") = vbYes Then
```

```
    TextSQL = "UPDATE tblUser" & _
    " SET tblUser.PassWord = '" & NewPasswordOne & "' " & _
    " WHERE LastName = '" & SelectLastName & "' " & _
    " AND FirstName = '" & SelectFirstName & "'" & _
    " AND SecondName = '" & SelectSecondName & "'"
    ' Debug.Print TextSQL
    DoCmd.RunSQL TextSQL
    Indicator = 3  ' См. событие "До обновления" для этой формы -
                   ' Form_BeforeUpdate
    DoCmd.Close    ' Закрытие формы
End If
End Sub
```

Программный код для этой кнопки предусматривает не только стирание пароля, но и занесение нового (5 нулей рядом) в зашифрованном виде. Это сделано на тот случай, если злоумышленник каким-либо способом получит доступ к таблице `tblUser`. Уничтожение пароля одного из пользователей ему ничего не даст, т. к. при регистрации под чужим именем сработает код кнопки **Вход** формы `Login`, и на экране нарушитель получит сообщение (рис. 8.15). Попытка подсмотреть пароль в таблице `tblUser` также ничего не даст. Он хранится там в зашифрованном виде.

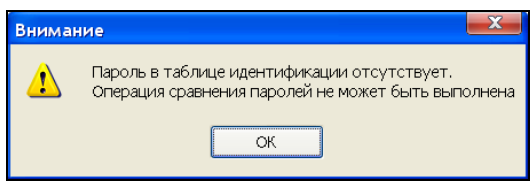


Рис. 8.15. Попытка стереть пароль, минуя программный комплекс

В листинге 8.17 приведен код кнопки **Добавить**. Кнопка расположена на первой вкладке формы `Employee`. Для корректной работы создается запрос, в который заведомо не попадет ни одна запись. Для этого выполняется поиск работника с несуществующей фамилией (фамилия отсутствует). Таких записей в таблице нет в принципе, т. к. ключ не может содержать пустое значение. Текст запроса помещен в строковую переменную `TextSQL`. Источник данных формы — теперь этот "пустой" запрос:

```
Me.RecordSource = TextSQL
```

Листинг 8.17. Код события *Нажатие кнопки* объекта `CommandAdd`

```
Private Sub CommandAdd_Click()
' Кнопка Добавить
Dim TextSQL As String ' Строка запроса
```

```
Indicator = 1          ' Режим добавления включен
' См. событие "До обновления" этой формы — Form_BeforeUpdate

' Вариант 1. Запрос, не возвращающий ни одной записи.
' Ключ не может содержать пустое значение.
' Запись будет заведомо пустая
' TextSQL = "SELECT * from tblUser WHERE LastName = ''"
' Источник записей формы — запрос TextSQL
' Me.RecordSource = TextSQL

' Вариант 2. Штатная работа Access.
' Добавление новой записи
DoCmd.GoToRecord acDataForm, "Employee", acNewRec
' Вывод на экран второй вкладки
Page2.Visible = True
' Изменение заголовка второй вкладки
Page2.Caption = "Занесение нового работника"
' Кнопка Стереть пароль недоступна
CommandDelPassword.Enabled = False
' Кнопка Удалить недоступна
CommandDel.Enabled = False
' Передача управления полю Фамилия
txtLastName.SetFocus
' Первая вкладка (Список работников) невидима
Page1.Visible = False
End Sub
```

Изменение заголовка второй вкладки на "Занесение нового работника" и удаление с экрана первой вкладки поможет пользователю легко сориентироваться в ситуации занесения новой записи. Форма примет вид, показанный на рис. 8.16. Кнопки **Удалить** и **Стереть пароль** погашены.

Для добавления новой записи в форму `Employee` вместо "пустого" запроса:

```
TextSQL = "SELECT * from tblUser WHERE LastName = ''"
Me.RecordSource = TextSQL
```

можно применить более простую конструкцию VBA:

```
DoCmd.GoToRecord acDataForm, "Employee", acNewRec
```

И даже ее можно упростить, опустив первые два параметра. VBA "поймет", что дело следует иметь с активным объектом, а это форма `Employee`:

```
DoCmd.GoToRecord ,,acNewRec
```

Общий синтаксис такой конструкции имеет вид:

```
DoCmd.GoToRecord тип_объекта, имя_объекта, запись, смещение
```

Здесь:

- *тип_объекта* — может принимать ряд значений, указанных в табл. 8.10;
- *имя_объекта* — имя объекта, работающего с записью;
- *запись* — может принимать ряд значений, указанных в табл. 8.11;
- *смещение* — номер текущей записи (применяется с `acGoTo`).

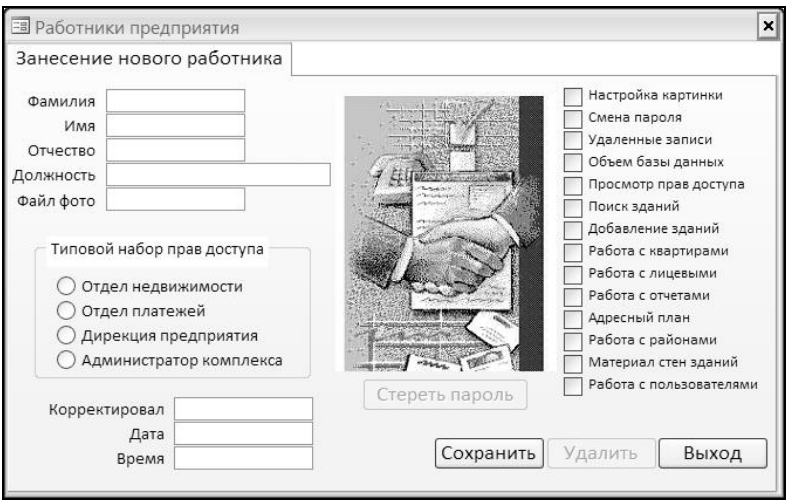


Рис. 8.16. Вид формы при занесении нового работника

Таблица 8.10. *Значения констант, определяющих тип объекта*

Константа	Значение	Описание
`acActiveDataObject`	–1	Активный объект, работающий с записью (принимается по умолчанию)
`acDataForm`	2	Форма
`acDataFunction`	10	Пользовательская функция (только для adp-файла)
`acDataQuery`	1	Запрос
`acDataReport`	3	Отчет
`acDataServerView`	7	Представление (только для adp-файла)
`acDataStoredProcedure`	9	Хранимая процедура (только для adp-файла)
`acDataTable`	0	Таблица

Таблица 8.11. *Значения констант, определяющих, какая запись будет назначена текущей*

Константа	Значение	Описание
acFirst	2	Сделать текущей первую запись
acGoTo	4	Сделать текущей запись с указанным номером
acLast	3	Сделать текущей последнюю запись
acNewRec	5	Сделать текущей новую запись
acNext	1	Сделать текущей следующую запись
acPrevious	0	Сделать текущей предыдущую запись

В листинге 8.18 приведен код события **Нажатие кнопки** объекта `CommandExit`. Кнопка расположена непосредственно в форме и доступна из ее любого места. Значение индикатора (`Indicator=3`) дает возможность отказаться от всех изменений и завершить работу с формой без запроса на подтверждение.

Листинг 8.18. Код кнопки *Выход*

```
Private Sub CommandExit_Click()
  ' Кнопка Выход.
  ' Выход без сохранения изменений.
  ' См. событие "До обновления" этой формы — Form_BeforeUpdate
  Indicator = 3
  DoCmd.Close    ' Закрытие формы
End Sub
```

Последний листинг 8.19 позволяет пользователю пересмотреть внесенные изменения перед сохранением записи. В зависимости от того, в каком состоянии находится значение переменной `Indicator`, пользователь получает соответствующий запрос на подтверждение своих действий или не получает его вообще.

Листинг 8.19. Код события *До обновления* формы `Employee`

```
Private Sub Form_BeforeUpdate(Cancel As Integer)
' До обновления
Select Case Indicator
  Case 1 ' Штатный режим добавления записи
    If MsgBox("Добавить новую запись в базу данных? ", _
      vbInformation + vbYesNo, "Сохранение") = vbNo Then
     ' Отказ от изменений
      DoCmd.RunCommand acCmdUndo
    End If
```

```
  Case 2 ' Штатный режим корректировки
    If MsgBox("Сохранить сделанные изменения? ", _
      vbInformation + vbYesNo, "Сохранение") = vbNo Then
    ' Отказ от изменений
      DoCmd.RunCommand acCmdUndo
    End If

  Case Else
    ' Во всех остальных случаях данные не обновлять.
    ' Отказ от изменений
    DoCmd.RunCommand acCmdUndo
End Select
End Sub
```

ГЛАВА 9

Создание основных форм приложения

Разрабатываемое нами приложение предназначено для использования в локальной вычислительной сети. С ним будет работать несколько человек одновременно. Настало время выбрать стратегию урегулирования многопользовательских конфликтов. Они могут возникать тогда, когда пользователи мешают друг другу. Обычно это происходит в том случае, если сразу несколько человек пытаются получить доступ к одним и тем же данным и изменить их.

9.1. Разработка многопользовательского приложения

MS Access 2010 хорошо работает в многопользовательских средах. Рассмотрим ситуации, в которых возникают конфликты, и те инструменты MS Access, которые помогают их предотвратить или разрешить. Сначала разберем механизмы блокировки, которые предоставляет MS Access. После этого перейдем к изучению стратегии для отдельных таблиц.

Если ваше приложение обслуживает локальную сеть, нельзя исключать возможность того, что сразу несколько пользователей захотят редактировать одни и те же данные. Конечно, если вам повезет, этого не произойдет. Однако вероятность подобных столкновений увеличивается по мере того, как растет число пользователей системы. Чем их больше, тем чаще возникают конфликты. Если вы не хотите полагаться на случай, то вставьте в свое приложение коды, благодаря которым оно сможет должным образом урегулировать возникающие проблемы.

Управление межпользовательскими конфликтами в MS Access состоит в следующем: задаются команды установок и блокировок, которые будут служить "арбитражем" для конкурирующих запросов. Блокировка снабжает пользователя "замком", который не дает другим пользователям изменять его данные. Хотелось бы, чтобы это происходило автоматически, без применения кода. До определенной степени это действительно возможно.

Прикладному программисту, которому надо для начальства быстро положить на стол некоторые данные, не извлекаемые стандартными отчетами приложения, не требуется забивать себе голову транзакциями и блокировками уже хотя бы потому, что оператор `SELECT`, который является основой для построения любых отчетов, не выполняет блокировок данных. В 90% случаев разработчику достаточно представлять себе, что он работает с системой один на один, ничего не зная про остальных пользователей. То же касается и пользователей: хорошо спроектированное приложение работает так, что СУБД сделает все сама: "заблокирует" кого надо и "откатит" кого нужно. MS Access автоматически управляет запросами и по умолчанию посылает пользователю сообщения при конфликтах.

Конфликты могут возникнуть как минимум в трех ситуациях: при редактировании и запросе данных, а также сопровождении базы данных.

- *Конфликты при редактировании.* Случается, что два или более пользователя одновременно пытаются редактировать одни и те же данные. Возможно ли сделать это в вашем приложении? Или оно "разрешит" редактирование одному из пользователей, а остальные будут ждать своей очереди?
- *Конфликты при запросе.* Эта ситуация возникает, когда один из пользователей запускает длительный запрос или отчет, и важно, чтобы данные в нем были непротиворечивы. Если вы позволите другим пользователям в это время редактировать данные, в них могут быть введены новые значения, которые сделают результаты обрабатываемого запроса недействительными.
- *Конфликты при сопровождении.* Иногда необходимо создать новые индексы или резервную копию базы данных, а эти действия могут совершаться только одним пользователем. Что, если в это время в системе будут другие пользователи, которые помешают вашей программе получить эксклюзивный доступ к нужным таблицам? Способы разрешения подобных конфликтов необходимо предусмотреть в любом приложении баз данных, т. к. подобные столкновения практически неизбежны.

При разработке стратегии разрешения конфликтов между пользователями надо учитывать:

- требования заказчика;
- вероятность конфликтов;
- требования к организации сопровождения приложения.

У каждого заказчика свои особенности характера. Возможно, он потребует, чтобы в любой момент времени доступ к определенному фрагменту информации имел только один человек или процесс. Например, у одного из ваших клиентов может быть деловое правило, гласящее, что с лицевым счетом `account` в каждый момент времени работает только один инспектор. Тогда ваше приложение должно отвечать этому условию. Другой клиент может предъявить прямо противоположное требование. Поэтому всегда старайтесь выяснить, есть ли у пользователя специальные пожелания.

В некоторых случаях вероятность конфликтов невелика. Например, вряд ли в многотысячном списке квартир `flat` два пользователя захотят одновременно редактировать одну и ту же запись. А вот в маленьких таблицах, которые постоянно используются, конфликты гораздо более вероятны.

При проектировании блокировок часто забывают, что каждое рабочее приложение должно обладать некоторыми средствами сопровождения. В их функции входят диагностика, обновление и переиндексация. Часто такие действия требуют эксклюзивного доступа к системе. Во всех этих случаях для других пользователей доступ должен быть запрещен.

Итак, когда один из пользователей редактирует некоторые данные, другие пользователи не должны иметь к ним доступа. Если вы приняли такое решение, можете использовать блокировку — инструмент, который есть во всех базах данных. Взаимодействуя с вашей операционной системой и/или локальной вычислительной сетью, MS Access устанавливает и снимает блокировки на основании запросов, которые поступают от запущенных экземпляров ваших приложений.

Необходимо гарантировать согласованность записи в базу данных. Поэтому MS Access, проводя изменения в некотором наборе данных, должен блокировать его. Возможно, это целая таблица, структуру которой вы меняете. Возможно, это заголовок таблицы, в котором обновляется количество строк таблицы. Возможно, это отдельная строка, которую нужно изменить. Обработчик базы данных MS Access гарантирует, что в конкретный момент времени работать с определенным множеством данных может только один пользователь, даже если вы не задаете такое требование специально.

9.2. Типы блокировок в MS Access

MS Access поддерживает два режима блокировки. Работа с ними предполагает минимум кодов или не требует их вообще.

- *Оптимистическая блокировка*. Используется по умолчанию. Запись блокируется только на время сохранения, а в процессе редактирования остается незаблокированной. Пользователь может редактировать вовсе не ту запись, которую видит на экране своего дисплея. Это случится в том случае, когда его более расторопный коллега, приступивший к редактированию записи позже, выполнит ее сохранение раньше.
- *Пессимистическая блокировка*. Это блокировка изменений. Позволяет только одному пользователю писать в заблокированные данные. Другие пользователи могут читать их, несмотря на блокировку, но не писать в них.

Для задания уровня блокировки данных в MS Access 2010:

1. Выберите в главном окне Microsoft Access 2010 вкладку **Файл**. Откроется окно.

2. Сделайте щелчок по кнопке **Параметры**. Она расположена в левом нижнем углу окна.
3. В левой части появившегося окна выберите пункт **Параметры клиента** (рис. 9.1).
4. Установите требуемые параметры и нажмите кнопку **OK**.

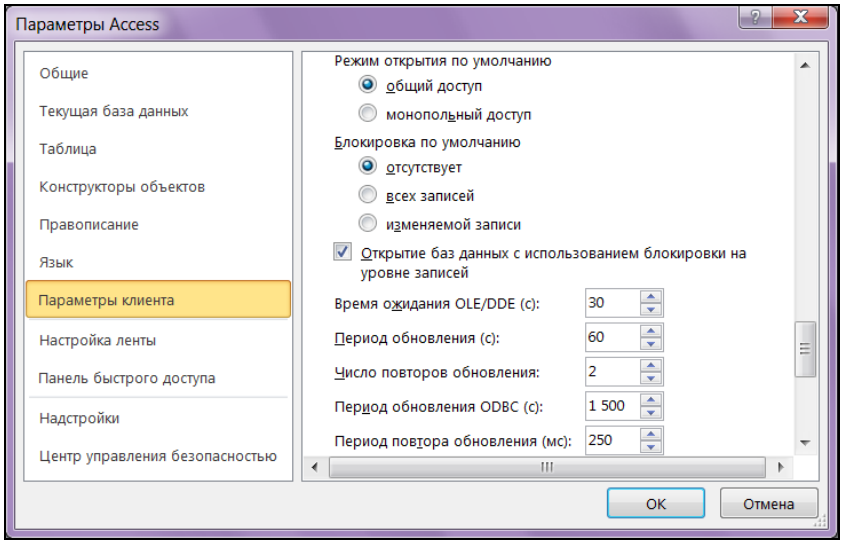


Рис. 9.1. Окно управления блокировками

Рассмотрим подробно назначение управляющих элементов в этом окне.

- **Режим открытия по умолчанию**:
  - **общий доступ** — открытие существующей базы данных для совместного использования. Этот параметр выбран по умолчанию;
  - **монопольный доступ** — открытие существующей базы данных для монопольного использования одним пользователем.
- **Блокировка по умолчанию**:
  - **отсутствует** — оптимистическая блокировка. Записи открыты для внесения изменений;
  - **всех записей** — пессимистическая блокировка всех записей в открытой форме или таблице, блокировка записей в любых базовых таблицах. Записи будут заблокированы, пока открыты объекты;
  - **изменяемой записи** — пессимистическая блокировка изменяемой в данный момент записи.
- **Открытие баз данных с использованием блокировки на уровне записей** — задание блокировки на уровне записей по умолчанию для текущей ба-

зы данных. Если флажок снят, то устанавливается по умолчанию блокировка на уровне страниц.

- **Время ожидания OLE/DDE (с)** — период повторения попыток выполнить операцию OLE или DDE после неудачной попытки. Допустимыми являются значения от 0 до 300. Значение по умолчанию — 30.
- **Период обновления (с)** — число секунд, после которых Microsoft Access автоматически обновит записи в режиме таблицы или формы. Допустимыми являются значения от 0 до 32 766. Значение по умолчанию — 60. При нулевом значении обновление не происходит.
- **Число повторов обновления** — количество сохранений измененной записи, заблокированной другим пользователем, которое Microsoft Access пытается выполнить. Возможные значения от 0 до 10. Значение по умолчанию — 2.
- **Период обновления ODBC (с)** — период, после которого происходит автоматическое обновление данных, получаемых через ODBC-подключение. Этот параметр действует только при работе с базой данных общего доступа в сети. Возможные значения от 0 до 32 766. Значение по умолчанию — 1500. При нулевом значении обновление не происходит.
- **Период повтора обновления (мс)** — число миллисекунд, по истечении которых MS Access пытается сохранить измененную запись, заблокированную другим пользователем. Допустимыми являются значения от 0 до 1000. Значение по умолчанию — 250.

Предупреждение

Выбранный здесь вариант блокировки применяется к данным в таблицах и программах, использующих объект `Recordset` для перебора записей, и не распространяется на запросы с использованием инструкций SQL. При работе с формами настройки блокировки по умолчанию не действуют. В этом случае следует обратиться к свойству формы **Блокировка записей**. Его можно найти на вкладке **Данные**. Значений у этого свойства три: *Отсутствует*, *Всех записей* и *Изменяемой записи*.

9.3. Создание многопользовательских форм

Мы перешли к созданию многопользовательских форм. Этот раздел поможет вам пройти через все этапы создания такой формы. На конкретном примере рассмотрим последствия применения блокировки, обработку ошибок и сообщения пользователю при выбранном типе блокировки. Самыми распространенными операциями в многопользовательских формах являются: добавление, редактирование и удаление записей. Многопользовательские формы должны иметь дело с каждой из этих операций.

Добавление требует наименьших усилий. При добавлении записи никакого конфликта между пользователями не возникает.

На этапе редактирования при оптимистической блокировке конфликты могут возникать, если один из пользователей пытается редактировать и обновлять строку в то время, когда ее уже редактирует кто-то другой. Подчеркну, что столкновения такого рода вероятны только при оптимистической блокировке. При подобном конфликте MS Access 2010 возлагает решение этого вопроса на пользователя, который изменяет запись (рис. 9.2).

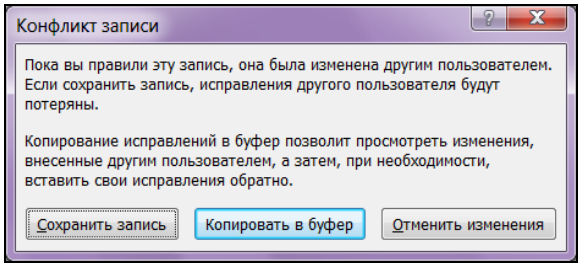


Рис. 9.2. Оптимистическая блокировка в действии

Конфликт, который возникает при удалении, является несколько более сложным. Возможно, хотя и маловероятно, что два пользователя захотят удалить одну и ту же строку одновременно. Поскольку они оба хотят совершить одинаковое действие, это проблемы не составляет. Однако не исключено, что один из пользователей попытается удалить строку, которую другой в это время редактирует. Поэтому лучше всего задать в процедуре, отвечающей за удаление, следующую функцию: отследить тот факт, что другой пользователь работает над строкой, и если это так, не выполнять удаление.

Продолжим работу по совершенствованию нашего приложения, созданного в *части I*. Модернизируем форму `Building`, приспособив ее для работы в многопользовательской среде. Выберем вариант пессимистической блокировки изменяемой записи: пока один пользователь редактирует запись, остальные — могут ее только просматривать.

Изменим три свойства формы, приведенные в табл. 9.1, и напишем одну процедуру обработки событий.

Таблица 9.1. *Измененные свойства формы* `Building`

Свойство	Вкладка	Значение
Область выделения	**Макет**	*Да*
Блокировка записей	**Данные**	*Изменяемой записи*
Интервал таймера	**События**	*100000*
Таймер	**События**	Процедура обработки событий

Чтобы узнать о том, работает ли еще кто-нибудь из пользователей в данное время с записью, отображенной в форме, посмотрите на вид значка в области выделения (рис. 9.3). Возможны три варианта. Они показаны рядом с формой.

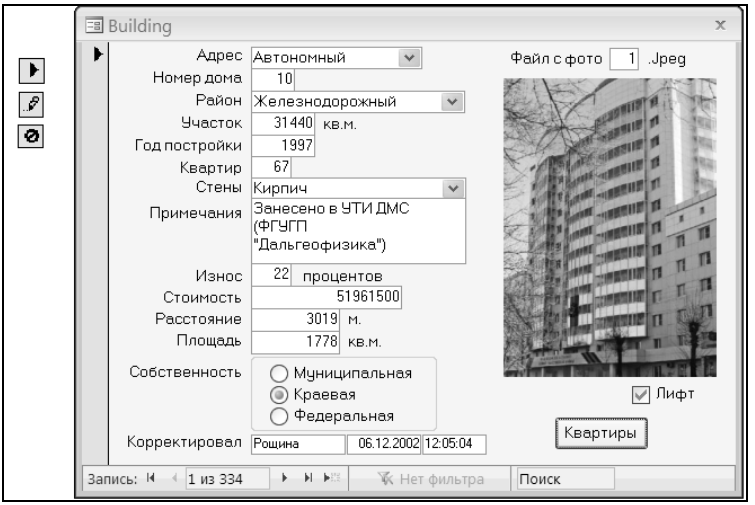


Рис. 9.3. Вид индикатора работы с записью

Индикатор в виде треугольника говорит о том, что запись находится в режиме просмотра, и никто из пользователей не занимается ее корректировкой. Попробуйте откорректировать какое-либо поле в форме, и вид индикатора сразу изменится. Вы увидите стилизованное изображение карандаша. Запись свободна — можете вносить изменения. Если же треугольник при попытке корректировки превратится в перечеркнутый круг, ждите — кто-то из пользователей ее корректирует в данный момент времени. Сразу же возникает вполне уместный вопрос: "Сколько ждать?" А что, если коллега отвлекся или вообще забыл про свой компьютер! Для этого следует предусмотреть автоматическое закрытие его формы после истечения определенного интервала времени. Изменим значение свойства формы **Интервал таймера** — это число миллисекунд до наступления события **Таймер**. Его код приведен в листинге 9.1. Для завершения редактирования данных по зданию вполне достаточно 100 секунд.

Листинг 9.1. Код события *Таймер*

```
Private Sub Form_Timer()
' Закрытие формы по истечении числа миллисекунд,
' заданных в свойстве "Интервал таймера"
If Not CurrentProject.AllForms("Flats").IsLoaded Then
   ' Закрыть только в случае, если пользователь
   ' не перешел к работе с квартирами
```

```
   DoCmd.Close
End If
End Sub
```

Дорога для работы остальных пользователей открыта. Подумаем о виновнике блокировки. Он уже не занимается корректировкой, а перешел вниз по каскаду и работает с квартирами? Форму `Building` принудительно закрывать нельзя, т. к. форма `Flats` использует ссылки на объекты формы `Building`. Выполним проверку: загружена форма `Flats` или нет, и только в случае ее отсутствия считаем, что работник "забыл" про свой компьютер.

Продолжим изучение ситуации. Работа с квартирами закончена, и форма `Building` снова активна. Таймер по-прежнему обновляется каждые 100 секунд. А что если работник вернулся к зданиям на последних секундах этого периода? Форма `Building` закроется у него перед самым носом! И, скорее всего, вы получите несколько неприятный отзыв о себе, как о разработчике. В листинге 9.2 приведен код обработки события активизации формы после возвращения из путешествия по каскаду.

Листинг 9.2. Код события *Включение*

```
Private Sub Form_Activate()
  ' Сброс интервала таймера при активизации формы
  Me.TimerInterval = 100000
End Sub
```

Установим интервал таймера снова равным 100 секундам. Вот вам и сброс его значения. Осталось предусмотреть ситуацию, когда пользователь занимается изучением информации по зданиям и не интересуется квартирами. Для изучения одного здания 100 секунд вполне достаточно, а нескольких? В листинге 9.3 приведен код обработки события, наступающего при переходе с записи на запись. Задача решена.

Листинг 9.3. Код события *Текущая запись*

```
Private Sub Form_Current()
  ' Сброс интервала таймера при переходе с записи на запись
  Me.TimerInterval = 100000
End Sub
```

Чтобы узнать о том, освободилась запись или нет, другим пользователям вовсе не обязательно постоянно переходить в режим корректировки. Перечеркнутый круг автоматически превратится в треугольник максимум через 60 секунд (рис. 9.1, параметр **Период обновления (с)**) после того, как пользователь, вы-

полнявший корректировку, завершит ее, или сработает таймер, установленный в форме.

Примечание

Меньший период обновления несколько увеличивает нагрузку на сеть, но делает работу рядовых пользователей более комфортной.

Если свойство **Область выделения** имеет значение *Нет* — ничего страшного не произойдет. Значков-индикаторов не будет, но звуковой сигнал при попытке редактирования заблокированной записи останется. В обоих случаях MS Access не станет генерировать никакой ошибки.

Рассмотрим вариант оптимистической блокировки записи, при котором конфликты не происходят вообще. Оптимистическая блокировка используется в MS Access по умолчанию, она проста в реализации, и обычно предпочтение отдают именно ей из-за высокой степени доступности данных, поскольку право долговременного либо исключительного доступа к ним никому не предоставляется. При открытии записи для редактирования остальные пользователи также могут открывать ее для редактирования, причем преимущество сохранения внесенных изменений имеет первый пользователь. Хотя оптимистическая блокировка проста в реализации и обычно не порождает проблем доступа пользователей к своим данным, при ее использовании одним из наиболее важных вопросов работы с базами данных в многопользовательской среде является вопрос о том, чьи изменения следует сохранять.

Исправим свойство формы **Блокировка записей** на *Отсутствует*. Что может произойти в этом случае? Пользователь, приступивший к корректировке первым, отвлекся. Второй пользователь выполнил исправления быстрее и сохранил изменения раньше первого. Первый при завершении корректировки получит на экране дисплея окно **Конфликт записи** (см. рис. 9.2). Очень хорошее решение — кнопка **Копировать в буфер**. Вставьте фрагмент буфера обмена в тот же Microsoft Word и увидите результаты работы второго пользователя в виде хорошо читаемой таблицы (табл. 9.2). В ней список всех объектов формы и их значения.

Таблица 9.2. *Изменения, сделанные вторым пользователем*

ComboStreet	txtHOUSE	ComboDistrict	txtLAND	txtYEAR
Автономный	10	Железнодорожный	31000	1996

Сравните данные таблицы с рис. 9.3. Второй пользователь исправил площадь земельного участка и год постройки здания.

9.4. Еще один вариант оформления главной формы

В 1995 году с появлением новой версии Windows компания Microsoft предложила миру разработчиков баз данных приложение Tastrade, интерфейс которого был выполнен в основном с использованием набора вкладок (Page Frame). Пользователи быстро оценили возможности, которые дает работа с этим объектом. Формирование запроса к базе данных, просмотр записей, попавших в запрос, работа с нужной записью — и все это в одной форме! Переход на соседние вкладки возможен, но разрешен только там, где это требуется по смыслу решаемой задачи. Объект **Набор вкладок** имеется и в MS Access 2010, только событий при работе с ним, в отличие от других средств разработки, предусмотрено меньше. Возможно, что именно из-за этого Page Frame и не получил в MS Access широкого применения. Предлагаю вашему вниманию форму для работы со зданиями в исполнении **Набор вкладок**. Это форма `PageBuilding`.

Научить конечного пользователя работе с фильтром MS Access — непростая задача. Мощный инструмент требует серьезного подхода. Сложные выборки, выполняемые далеко не каждый день, для фильтра Access — не проблема, но в повседневной работе пользователи применяют его не часто, предпочитая для поиска "прокручивать" список объектов. Примените методику этого раздела, и они будут вам благодарны. На рис. 9.4 показана форма `PageBuilding` после запуска.

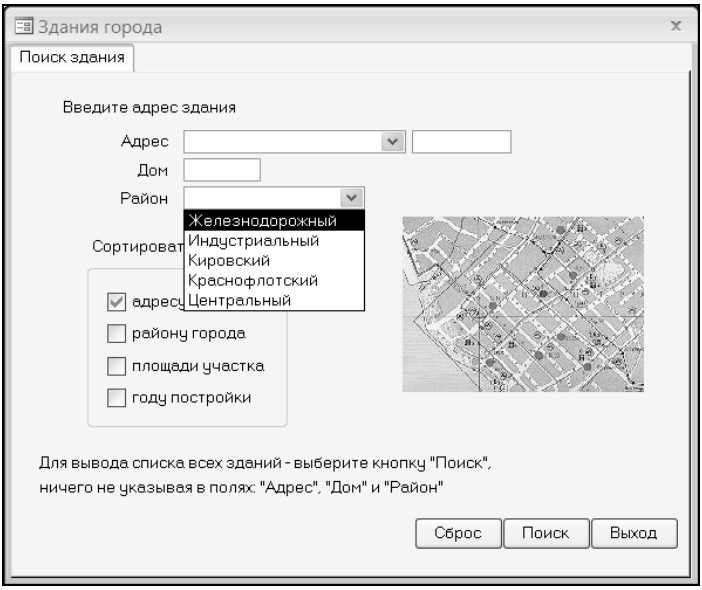


Рис. 9.4. Первая вкладка формы

В числе параметров поиска представлены только самые часто используемые: улица, номер дома, район и порядок сортировки. Перегружать страницу не следует — получите тот же фильтр MS Access, только в авторском исполнении. Для поиска всех зданий, расположенных на улице, выберите ее название, оставив номер дома пустым. Также можно поступить и с районом города. Вторая и третья страницы формы в режиме первого поиска на экране не отображаются. Заголовки этих вкладок и сами вкладки будут появляться в процессе развития событий работы со зданиями. Особая роль отведена кнопке **Сброс**. Она предназначена для отмены установленных параметров поиска. Поиск можно пропустить вообще. Нажатие кнопки **Поиск** без установки параметров поиска даст список всех зданий, имеющихся в базе. На рис. 9.5 показана вторая страница формы, предназначенная для просмотра списка зданий, попавших в запрос. Всего зданий — 334.

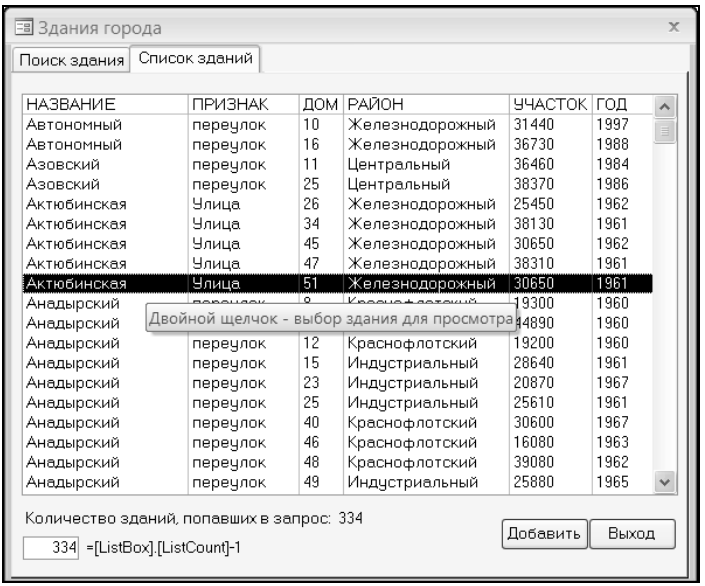


Рис. 9.5. Вторая вкладка формы

Двойной щелчок мышью по выбранному зданию позволяет выполнить переход на третью вкладку формы (рис. 9.6).

Теперь видны и доступны все три вкладки. Это обеспечивает возврат к выбранному списку зданий для детального изучения следующего объекта недвижимости или повторение запроса к базе с указанием новых параметров поиска.

В случае повторного поиска третья вкладка исчезнет с экрана и появится только после выбора здания для просмотра. В зависимости от режима (добавление записи, удаление или корректировка) доступны только нужные в данный момент вкладки и кнопки. Например, при занесении нового здания (кнопка **Добавить**

второй вкладки) с экрана убираются все вкладки, кроме третьей, которая становится первой с названием "Занесение нового здания", и гасятся кнопки **Удалить** и **Квартиры** (рис. 9.7).

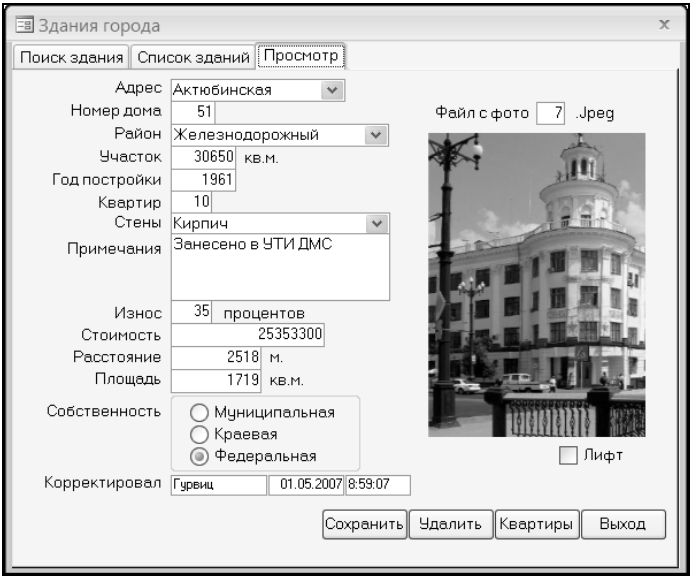


Рис. 9.6. Третья вкладка формы

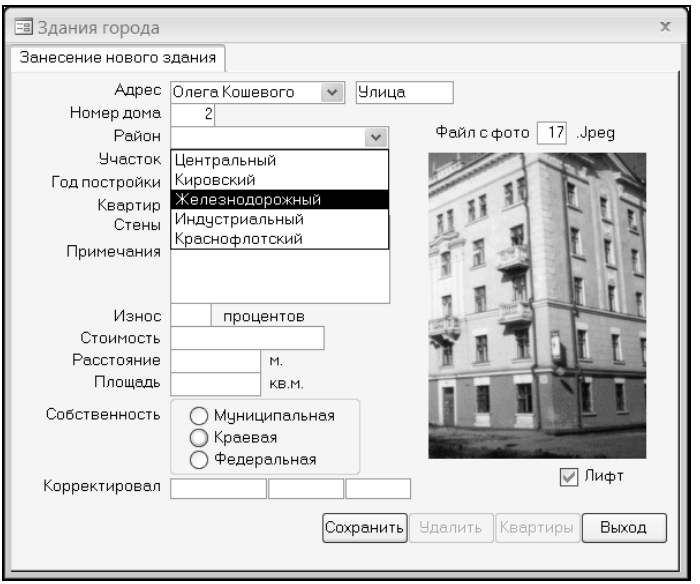


Рис. 9.7. Вид формы при занесении нового здания

9.4.1. Первая страница формы — поиск здания

Для создания страничной формы выберите в разделе **Элементы управления** вкладки **Конструктор** инструмент  **Вкладка** и разместите его в форме. Для этого в нужном месте активной области формы при помощи левой кнопки мыши отведите место этому объекту. Вкладок по умолчанию в наборе будет создано две. Далее порядок действий следующий:

1. Добавьте третью вкладку. Щелчок правой кнопкой мыши по заголовку набора вкладок откроет контекстное меню.
2. Пятым в нем должен быть пункт **Вставить вкладку**. Выберите его. Тот же результат даст выбор значка  **Вставить вкладку**, расположенного в разделе **Инструменты управления** вкладки **Конструктор**.
3. Переименуйте вкладки. Выберите их по очереди. В **Окне свойств** в разделе **Другие** измените свойство **Имя** на *Page1*, *Page2* и *Page3* соответственно.
4. Перейдите на вкладку **Макет**. Измените значения свойства **Подпись** на *Поиск здания*, *Список зданий* и *Просмотр*.
5. Для второй и третьей вкладок значение свойства **Вывод на экран** измените на *Нет*.
6. Дайте имя набору вкладок — `PageFrame`.

На рис. 9.8 приведена первая вкладка формы в режиме конструктора, а в табл. 9.3 — список объектов первой вкладки формы `PageBuilding`.

Таблица 9.3. *Список объектов первой вкладки формы* `PageBuilding`

Объект	Имя объекта	Пояснение	Данные
Поле со списком	`ComboBoxSeekStreet`	Адрес	`SELECT tblStreet.STREET, tblStreet.NAME, tblStreet.SIGN, tblStreet.FIRST FROM tblStreet ORDER BY tblStreet.[NAME], tblStreet.[SIGN];`
Поле	`txtSignSeek`	Признак адреса	`=[ComboBoxSeekStreet].[Column](2)`
Поле	`txtSeekHouse`	Дом	
Поле со списком	`ComboBoxSeekDistrict`	Район	`SELECT tblDistrict.DISTRICT, tblDistrict.AREA FROM tblDistrict ORDER BY tblDistrict.[AREA];`
Группа переключателей	`OptionSort`	Сортировать по	Значение по умолчанию: 1 Сортировка по адресу
Рисунок	`PicStreet`	Картинка	
Кнопка	`CommandKill`	Сброс	Код VBA (листинг 9.5)
Кнопка	`CommandSeek`	Поиск	Код VBA (листинг 9.6)
Кнопка	`CommandExit`	Выход	Код VBA (см. компакт-диск)

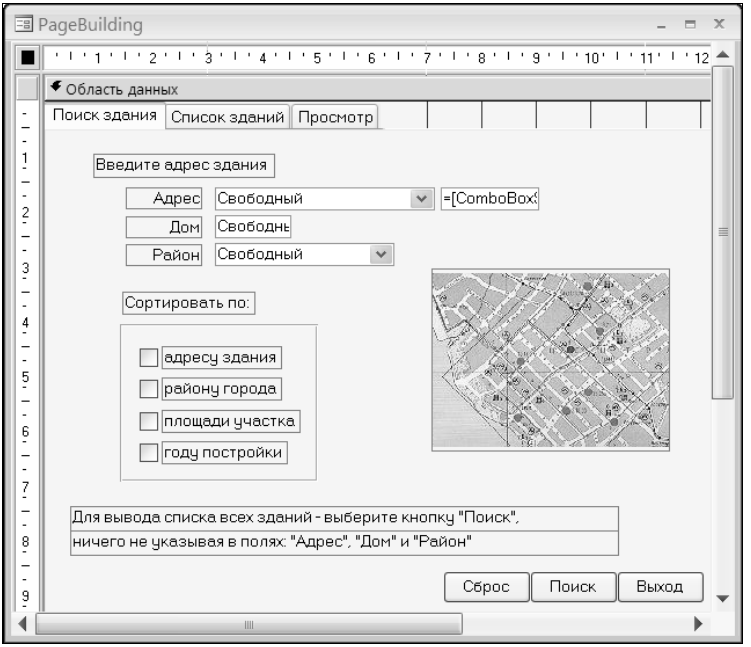


Рис. 9.8. Первая вкладка формы в режиме конструктора

В листинге 9.4 приведен текст начала модуля формы `PageBuilding`. Создав форму MS Access, мы получаем в свое распоряжение модуль VBA (рис. 9.9). Его имя формируется автоматически с добавлением приставки "Form_" к имени формы.

Просмотреть текст модуля очень просто. Форму для этого открывать необязательно.

1. Перейдите на четвертую вкладку главной ленты MS Access 2010 **Работа с базами данных**.
2. Запустите Visual Basic for Applications.
3. В открывшемся окне VBA сделайте видимым окно проекта. Пункт меню **View** строка **Project Explorer**.
4. Найдите в нем модуль `Form_PageBuilding` и сделайте по нему двойной щелчок мышью.

Листинг 9.4. Заголовок модуля `Form_PageBuilding`

```
Option Explicit
Option Compare Database
' Переменные уровня проекта
Public SelectStreet As Integer     ' Выбранная улица
Public SelectHouse As Integer      ' Выбранный номер дома
```

```
' Переменные уровня модуля
Dim SelectSeekStreet As Integer   ' Номер улицы для поиска
Dim SelectSeekHouse As Integer    ' Номер дома для поиска
Dim SelectSeekDistrict As Integer ' Номер района
Dim SelectSort As Integer         ' Порядок сортировки
Dim SQLText As String             ' Текст запроса
Dim Indicator As Integer          ' Индикатор выбора
```

В заголовке объявлены переменные уровня проекта и уровня модуля, применяемые в тексте этого модуля. Следом расположены все процедуры обработки событий и пользовательские функции и процедуры, если они имеются.

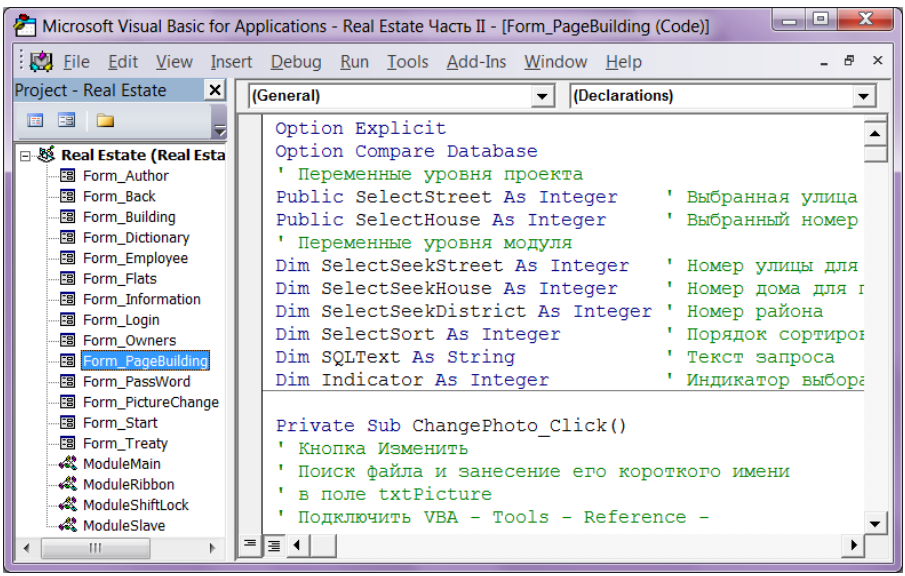


Рис. 9.9. Текст модуля `Form_PageBuilding`

Из формы зданий запускаются другие формы: квартир, лицевых счетов и т. д. В них должны быть известны координаты здания (номер улицы и номер дома), поэтому две переменные объявлены с атрибутом `Public`. Это переменные `SelectStreet` и `SelectHouse`. В листинге 9.5 приведен код обработки события **Нажатие кнопки**, который выполняется при необходимости подготовки нового поиска.

Листинг 9.5. Сброс параметров поиска

```
Private Sub CommandKill_Click()
   ' Кнопка "Сброс".
   ' Сброс введенных значений для поиска.
```

```
    ' Выбранная улица
    ComboBoxSeekStreet.Value = Null
    ' Номер дома
    txtSeekHouse.Value = Null
    ' Выбранный район
    ComboBoxSeekDistrict.Value = Null
    ' Порядок сортировки — по адресу
    OptionSort.Value = 1
End Sub
```

В листинге 9.6 формируется текст запроса для свойства **Данные** (RowSource) объекта `ListBox` второй вкладки формы. Этот запрос не обновляемый, т. к. для связывания таблиц применяется служебное слово `WHERE`, а не `FROM-INNER-JOIN-ON`. Текст изящнее, а обновлять данные в этой выборке не требуется. Строка запроса состоит из трех частей. В первой части определяется набор полей для выборки и им присваиваются русские названия: дом, район, участок, год и т. д. В таком виде они и будут размещены на второй странице набора вкладок. Вторая часть запроса может вообще отсутствовать, если пользователь не введет параметры поиска, а может содержать в самом общем случае три параметра (улица, дом и район). Все они начинаются с ключевого слова `AND`. Третья часть запроса содержит порядок сортировки: `ORDER BY`. Сортировка по адресу назначена по "умолчанию".

Листинг 9.6. Кнопка *Поиск* — основная кнопка первой вкладки

```
Private Sub CommandSeek_Click()
' Кнопка "Поиск"
Dim CountBuilding As Integer
' Количество зданий, попавших в запрос
If IsNull(ComboBoxSeekStreet.Value) Then
    ' Улица не имеет значения для пользователя
    SelectSeekStreet = 0
Else
    ' Номер улицы для поиска
    SelectSeekStreet = ComboBoxSeekStreet.Value
End If
If IsNull(txtSeekHouse.Value) Then
    ' Номер дома не имеет значения для пользователя
    SelectSeekHouse = 0
Else
    ' Номер дома для поиска.
    ' Номер дома преобразован из строки символов в число
    SelectSeekHouse = Val(txtSeekHouse.Value)
End If
If IsNull(ComboBoxSeekDistrict.Value) Then
    ' Район не имеет значения для пользователя
    SelectSeekDistrict = 0
```

```
Else
   ' Номер района для поиска
   SelectSeekDistrict = ComboBoxSeekDistrict.Value
End If
' Порядок сортировки
SelectSort = OptionSort.Value

' Формирование строки запроса.
' Основная часть строки
SQLText = "SELECT tblBuilding.Street, " & _
    "tblStreet.Name AS НАЗВАНИЕ, " & _
    "tblStreet.Sign AS ПРИЗНАК, " & _
    "tblBuilding.House AS ДОМ, tblDistrict.AREA AS РАЙОН, " & _
    "tblBuilding.Land AS УЧАСТОК, tblBuilding.Year AS ГОД " & _
 "FROM tblStreet, tblBuilding, tblDistrict " & _
    "WHERE  tblStreet.Street = tblBuilding.Street " & _
    "AND tblDistrict.District = tblBuilding.District "

' Дополнительная часть строки содержит параметры выбора
If SelectSeekStreet <> 0 Or SelectSeekHouse <> 0 Or _
   SelectSeekDistrict <> 0 Then
   If SelectSeekStreet <> 0 Then
      ' Если выбрана улица
      SQLText = SQLText & " AND tblBuilding.Street =" & _
                SelectSeekStreet
   End If
   If SelectSeekHouse <> 0 Then
      ' Если введен номер дома
      SQLText = SQLText & " AND tblBuilding.House =" & _
                SelectSeekHouse
   End If
   If SelectSeekDistrict <> 0 Then
      ' Если выбран район
      SQLText = SQLText & " AND tblBuilding.District =" & _
                SelectSeekDistrict
   End If
End If

' Продолжение основной части строки — сортировка
Select Case SelectSort
Case 1  ' По адресу здания
  SQLText = SQLText & " ORDER BY Name, Sign, House"
Case 2  ' По району города
  SQLText = SQLText & " ORDER BY tblDistrict.Area"
Case 3  ' По площади земельного участка
  SQLText = SQLText & " ORDER BY tblBuilding.Land"
```

```
Case 4  ' По году постройки
  SQLText = SQLText & " ORDER BY tblBuilding.Year"
End Select
' Устанавливаем свойства для поля со списком зданий.
' Тип источника данных (таблица или запрос)
ListBox.RowSourceType = "Table/Query"
' Источник данных поля
ListBox.RowSource = SQLText

' Подсчет количества зданий, попавших в запрос
CountBuilding = CountQuery(SQLText)
If CountBuilding = 0 Then
   ' В запросе нет записей
   MsgBox "Зданий, отвечающих условиям вашего запроса, в базе нет." & _
          " Повторите запрос, изменив требования.", _
          vbOKOnly + vbExclamation, "Внимание"
   ' Возврат на первую вкладку для повторения запроса
   Exit Sub
End If

' Изменение надписи внизу второй вкладки
CountRecords.Caption = _
     "Количество зданий, попавших в запрос: " & Str(CountBuilding)
' Выделение первой записи объекта "Поле со списком"
ListBox.Selected(1) = True
' Вторая вкладка видима
Page2.Visible = True
' Переход на вторую вкладку
Page2.SetFocus
End Sub
```

Предупреждение

Выполняя запрос, сформированный в процедуре, MS Access проверяет синтаксис, но не интерпретирует созданную разработчиком инструкцию SQL. Всегда помните об этом. Автору пришлось потратить достаточно неприличное количество времени для того, чтобы заставить этот запрос заработать. Дело в элементарной невнимательности. Посмотрите на рис. 2.1. Таблица районов города носит название `distict` (пропущена буква "r" в слове "district"). Конечно, при создании инструкции SQL автор писал правильно — `district`, а MS Access упорно отказывался находить хотя бы одно здание из нескольких сотен, абсолютно ничего не сообщая об ошибке типа: "не могу найти указанную таблицу!". В *данной части* эта таблица переименована, а в файле базы данных — оставлена без изменений, как напоминание о том, что в нашей работе мелочей не бывает.

Если в запрос не попадет ни одно здание, то пользователь получит сообщение о неудачной попытке поиска (рис. 9.10), и управление будет возвращено первой вкладке.

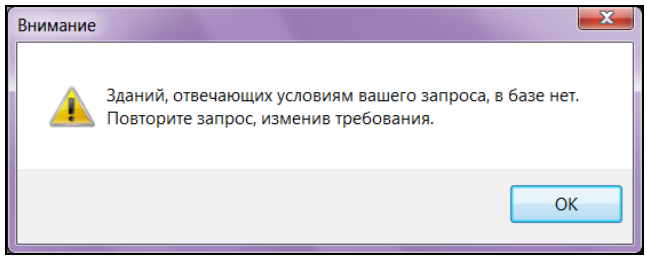


Рис. 9.10. Неудачный поиск здания

В случае удачного поиска выполняется переход на вторую вкладку и изменяется заголовок надписи, расположенной внизу вкладки. В него добавляется количество записей, попавших в выборку.

9.4.2. Доступ к данным из VBA. Microsoft ADO

Для подсчета количества записей, попавших в выборку, написана пользовательская функция `CountQuery()`. У нее один параметр: `SQLText` — строковая переменная с текстом запроса. Текст функции приведен в листинге 9.7. Для доступа к данным она использует технологию ActiveX Data Objects (ADO).

Листинг 9.7. Подсчет количества записей в выборке

```
Public Function CountQuery(SQLText) As Integer
' Возвращает количество записей в запросе.
' SQLText — строка с текстом запроса
Dim CountSelectSQL As Integer      ' Количество записей
' Создание ссылок на реальные объекты
Dim Connect As ADODB.Connection
' Connect — объектная переменная, тип соединения.
' Объект ADO верхнего уровня
Dim rsCount As ADODB.Recordset
' rsCount — объектная переменная, содержит набор записей

' Для создания соединения с текущей базой данных
'Set Connect = CurrentProject.Connection

' Для создания соединения с внешней базой данных
Set Connect = New ADODB.Connection
' Только для MS Access 2010 (файл accdb)
With Connect
     .ConnectionString = _
     "Provider=Microsoft.ACE.OLEDB.12.0;" & _
     "Data Source=" & CurrentPath() & "\Real Estate.accdb"
```

```
     .Open
End With

Set rsCount = New ADODB.Recordset     ' Создание набора
With rsCount
     ' Задание свойств объекта rsCount (Recordset)
     ' Источник: SQL-конструкция
     .Source = SQLText
     ' Указатель на открытое соединение
     .ActiveConnection = Connect
     .CursorType = adOpenKeyset    ' Тип курсора
     ' Метод объекта RS
     ' Открывает набор данных активного объекта
     .Open
End With
CountSelectSQL = rsCount.RecordCount
' RecordCount — свойство объекта rsCount.
' Возвращает количество записей.
' Освобождение объектных переменных
Set rsCount = Nothing
Set Connect = Nothing
' Количество записей в выборке
' CountQuery — имя функции
CountQuery = CountSelectSQL
End Function
```

Технология ADO представляет собой одну из моделей доступа к базам данных. В этом разделе будет рассмотрена лишь самая незначительная часть возможностей, которые предоставляет эта технология разработчику. Но их вполне достаточно, чтобы получить первые значимые результаты. Модель ADO содержит независимый набор объектов и коллекций (рис. 9.11).

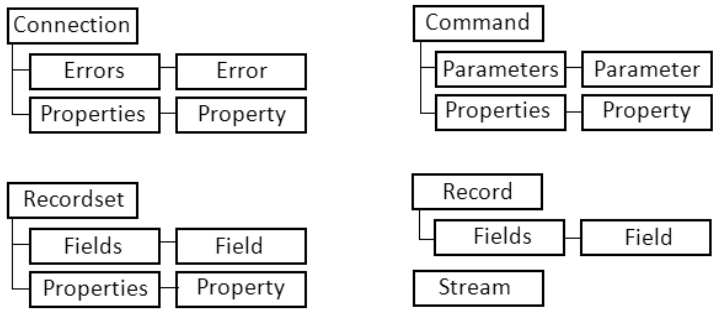


Рис. 9.11. Объекты и коллекции модели доступа к данным ADO

При помощи объектов ADO разработчик может:

- создавать базы данных и изменять их структуру;
- подключаться к базам данных, которые расположены на любых серверах сети;
- манипулировать данными, хранящимися в таблицах баз данных.

Где "живут" эти объекты и как с ними работать? Конечно же, в библиотеках! Просмотрим их список.

1. Откройте базу данных и запустите VBA.
2. В его главном окне выберите пункт **View**.
3. В появившемся меню — пункт **Object Browser**. Появится окно **Object Browser**.
4. В раскрывающемся списке, который расположен в левом верхнем углу окна, можно увидеть список библиотек: Access, ADODB, Office и т. д.
5. Выберите ADODB (рис. 9.12). В левом списке появится перечень объектов, а в правом — список свойств и методов выбранного объекта.

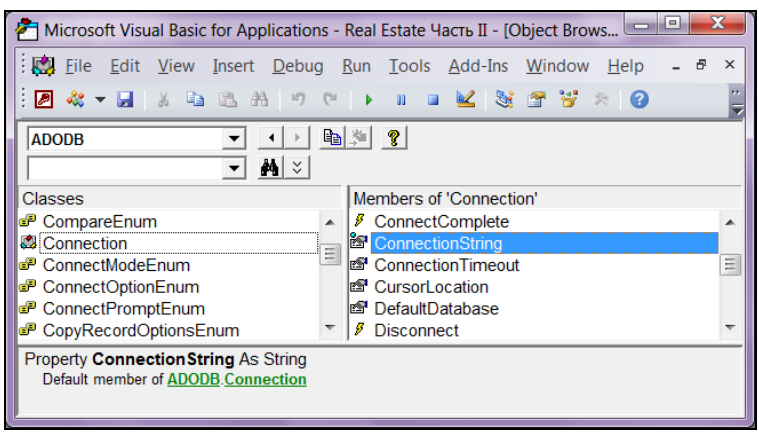


Рис. 9.12. Объект `Connection` библиотеки ADODB и его свойство `ConnectionString`

Не сомневаюсь, что первые строчки кода (см. листинг 9.7) вам теперь полностью понятны. Объявляется переменная `Connect` как объект `Connection` из библиотеки ADODB. Соединению назначается драйвер Microsoft.ACE.OLEDB.12.0 для связи с файлом базы данных MS Access 2010, которая называется Real Estate Часть II. Соединение открывается — `Open`. После создания соединения с базой данных можно приступить к формированию выборки данных:

```
Dim Connect As ADODB.Connection
Set Connect = New ADODB.Connection
```

```
With Connect
     .ConnectionString = _
     "Provider=Microsoft.ACE.OLEDB.12.0;" & _
     "Data Source=" & CurrentPath() & "\Real Estate.accdb"
     .Open
End With
```

Далее в тексте объявляется переменная `rsCount` как объект `Recordset` из библиотеки ADODB (рис. 9.13). Набор данных предписывается сформировать на основе SQL-запроса. Текст запроса находится в строке `SQLText`. Запрос нужно получить, используя соединение `Connect`. Тип выборки `CursorType` определяет константа `adOpenKeyset`.

```
Dim rsCount As ADODB.Recordset
Set rsCount = New ADODB.Recordset
With rsCount
     .Source = SQLText
     .ActiveConnection = Connect
     .CursorType = adOpenKeyset
     .Open
End With
```

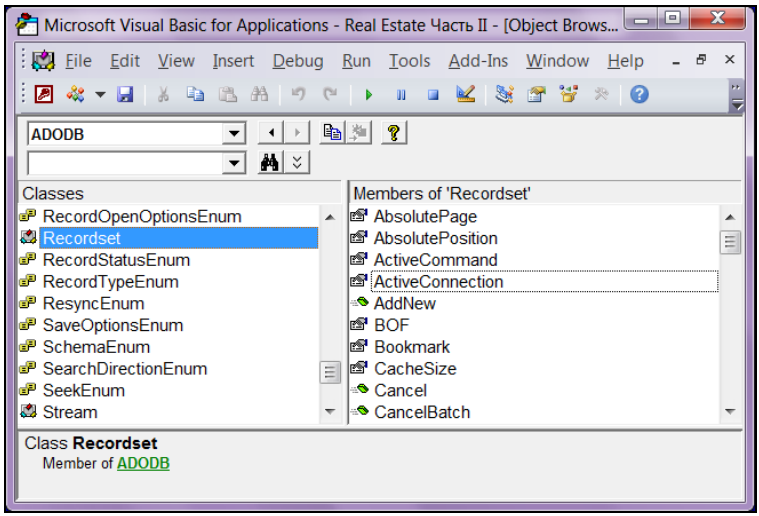


Рис. 9.13. Объект `Recordset` библиотеки ADODB и его свойства `ActiveConnection` и `CursorType`, используемые в процедуре

9.4.3. Вторая страница — просмотр списка

Основной объект второй вкладки — поле со списком `ListBox`. Мы уже имели дело с объектом такого типа, поэтому подробно не будем на нем останавливаться.

Отмечу лишь необходимость сброса присоединенного столбца этого объекта. Невнимательная работа с построителем может установить свойство **Присоединенный столбец**, а это приведет к тому, что при переходе от второй вкладки к третьей (от просмотра всего списка к работе с отдельной записью) активная запись будет все время меняться в зависимости от того, какой столбец (адрес, дом или район) присоединен. Для того чтобы избежать нареканий со стороны заказчика, не поленитесь — откройте еще раз окно свойств объекта `ListBox`, найдите на вкладке **Данные** свойство **Присоединенный столбец** и в качестве значения поставьте *0*.

В листинге 9.8 приведен код события, которое наступает при выборе здания для просмотра или корректировки. Это событие — двойной щелчок мыши.

Листинг 9.8. Выбор строки в объекте `ListBox`

```
Private Sub ListBox_DblClick(Cancel As Integer)
Dim TxtSQL As String ' Строка запроса
Indicator = 2        ' Режим корректировки включен
' См. событие "До обновления" этой формы — Form_BeforeUpdate.
' Номер выбранной улицы
SelectStreet = [ListBox].Column(0)
' Номер выбранного дома
SelectHouse = [ListBox].Column(3)
' Строка запроса
TxtSQL = "SELECT * FROM tblBuilding WHERE Street = " & _
          SelectStreet & " AND House = " & SelectHouse
' Источник данных формы — теперь этот запрос
Me.RecordSource = TxtSQL
' Третья вкладка видима и активна
Page3.Visible = True
Page3.SetFocus
End Sub
```

В тексте используются две переменные уровня проекта для хранения значений выбранной улицы (`SelectStreet`) и номера дома (`SelectHouse`). Эти значения выбираются из колонок поля со списком. Нулевая колонка (первый столбец) представляет собой не отображающийся на экране, из-за нулевого размера, номер улицы. Третья колонка (четвертый столбец) — номер дома.

Для подсчета количества записей в запросе мы применили пользовательскую функцию с использованием технологии ADO. Хочу предложить вашему вниманию значительно более короткий путь, но к цели он приведет, только если с данными, полученными в результате выполнения запроса, будет работать поле со списком. Это как раз наш `ListBox`. У этого объекта есть свойство `ListCount`

(количество записей). Создайте поле txtCoundBld внизу второй вкладки формы. Поставьте в свойстве **Данные** этого поля:

```
=ListBox.Count -1
```

если у объекта ListBox в свойстве **Заглавия столбцов** указано *Да*, и

```
=ListBox.Count
```

если свойство **Заглавие столбцов** имеет значение *Нет*. Задача решена!

В листинге 9.9 приведен код VBA, который запускается на выполнение после щелчка по кнопке **Добавить**. Это начало занесения нового здания.

Листинг 9.9. Код кнопки добавления здания

```
Private Sub CommandAdd_Click()
' Кнопка "Добавить"
Dim TextSQL As String ' Строка запроса
Indicator = 1         ' Режим добавления включен
' См. событие "До обновления" этой формы — Form_BeforeUpdate.
' Запрос, не возвращающий ни одной записи.
' Номер улицы не может быть отрицательным.
' Запись будет заведомо пустой
TextSQL = "SELECT * from tblBuilding WHERE Street = -1"
' Источник записей формы — запрос TextSQL
Me.RecordSource = TextSQL
' Первая вкладка (Поиск) невидима
Page1.Visible = False
' Вторая вкладка (Список зданий) невидима
Page2.Visible = False
' Вывод на экран третьей вкладки
Page3.Visible = True
' Изменение заголовка второй вкладки
Page3.Caption = "Занесение нового здания"
' Кнопка Квартиры недоступна
CommandFlats.Enabled = False
' Кнопка Удалить недоступна
CommandDel.Enabled = False
' Передача управления полю со списком (Улица)
ComboStreet.SetFocus
End Sub
```

В тексте создается запрос, не возвращающий ни одной записи. Сделать его не сложно. Попробуем найти здание, расположенное на улице с отрицательным номером. Попытка заведомо неудачная. Таких зданий просто не может быть в таблице building, т. к. номер улицы заносится в нее из справочника улиц street. Откройте таблицу улиц в конструкторе и убедитесь сами. Поле STREET содержит

Условие на значение >=0. Так как созданный запрос обновляемый, то проблем с переносом информации по новому зданию в таблицу building не возникнет.

9.4.4. Третья страница — работа с записью

Третья вкладка формы PageBuilding предназначена для просмотра сведений по выбранному зданию или для занесения нового здания. Выбором режима управляет переменная Indicator. Это переменная уровня модуля. Из других модулей не видна, но доступна из любого места текущей формы. В табл. 9.4 приведен список объектов третьей вкладки и даны необходимые пояснения. На рис. 9.14 приведена третья вкладка в режиме конструктора.

Таблица 9.4. *Список объектов третьей вкладки формы* PageBuilding

Объект	Имя объекта	Пояснение	Данные
Поле со списком	ComboStreet	Номер улицы	SELECT [tblStreet].[STREET], [tblStreet].[NAME], [tblStreet].[SIGN] FROM tblStreet; Присоединенный столбец=1
Поле	txtSign	Признак адреса	=[ComboStreet].[Column](2) **Блокировка** — *Да*
Поле	txtHouse	Дом	HOUSE
Поле со списком	ComboDistrict	Район	SELECT [tblDistrict].DISTRICT, [tblDistrict].AREA FROM tblDistrict; **Присоединенный столбец** — *1*
Поле	txtLAND	Площадь участка	LAND
Поле	txtYear	Год постройки	YEAR
Поле	txtFLATS	Количество квартир в здании	FLATS
Поле со списком	ComboMaterial	Материал стен	SELECT [tblWall].[MATERIAL], [tblWall].[WALL] FROM tblWall; **Присоединенный столбец** — *1*
Поле	txtComment	Примечания	COMMENT
Поле	txtWear	Износ	WEAR
Поле	txtCost	Стоимость	COST
Поле	txtLine	Расстояние от центра	LINE
Поле	txtSquare	Площадь	SQUARE

Таблица 9.4 *(окончание)*

Объект	Имя объекта	Пояснение	Данные
Группа переключателей	OptionGroup	Вид собственности	KIND
Поле	txtInspector	Корректировал	INSPECTOR **Блокировка** — *Да*
Поле	txtDate_up	Дата корректировки	DATE_UP **Блокировка** — *Да*
Поле	txtTime_up	Время корректировки	TIME_UP **Блокировка** — *Да*
Поле	txtPicture	Имя файла с фотографией	PICTURE
Рисунок	Picture	Фотография здания	=CurrentPath() & "\Building\" & [txtPicture].[Value] Вкладка **Макет**, **Рисунок** — NoFoto.jpg
Флажок	chkELEVATOR	Лифт	ELEVATOR
Кнопка	ChangePhoto	Изменить фотографию	Код VBA (листинг 9.10)
Кнопка	CommandSave	Сохранить	Код VBA (листинг 9.11)
Кнопка	CommandDel	Удалить	Код VBA (листинг 9.12)
Кнопка	CommandFlats	Квартиры	Код VBA (листинг 9.13)
Кнопка	CommandExit	Выход	Код VBA (листинг 9.14)

Отдельного внимания на третьей страницы вкладки заслуживает фотография здания и кнопка **Изменить**, расположенная над фото и имеющая к нему самое непосредственное отношение. Фотография "привязана" к текстовому полю txtPicture, в котором хранится имя файла с фотографией. Если файл не указан, то срабатывает свойство объекта Picture **Рисунок** — это файл с картинкой (NoFoto.jpg).

Все файлы помещены в папку Building. При работе с большим количеством изображений удобнее использовать методику работы с отдельными файлами, а не помещать их в поле **Вложение** или **Поле объекта OLE**. В этом случае у вас гораздо больше возможностей для манипуляции изображениями, но без дополнительного кода не обойтись. Напишем его (листинг 9.10). В очередной раз нам поможет технология ADO. Библиотеку Microsoft Office 12.0 Object Library мы уже подключили, когда создавали главное меню программного комплекса. Из нее понадобится объект FileDialog, представляющий собой диалоговое окно выбора файла (рис. 9.15).

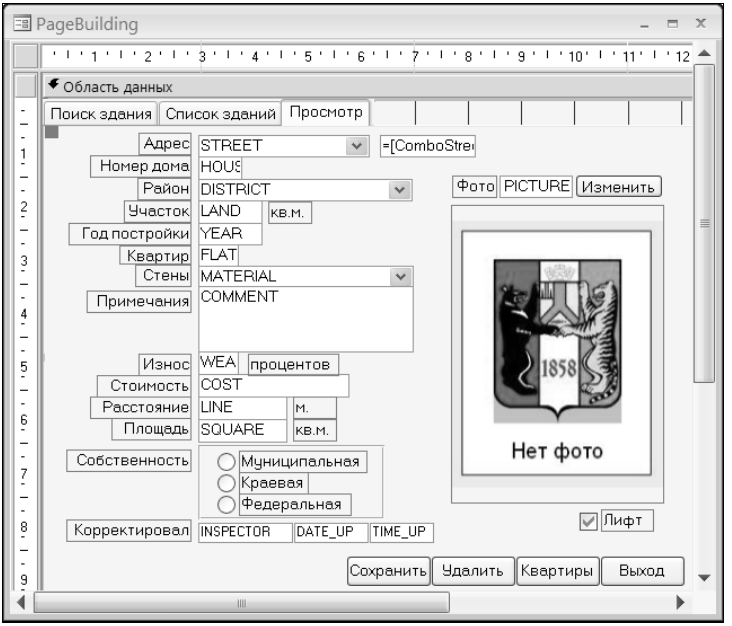


Рис. 9.14. Третья вкладка формы в режиме конструктора

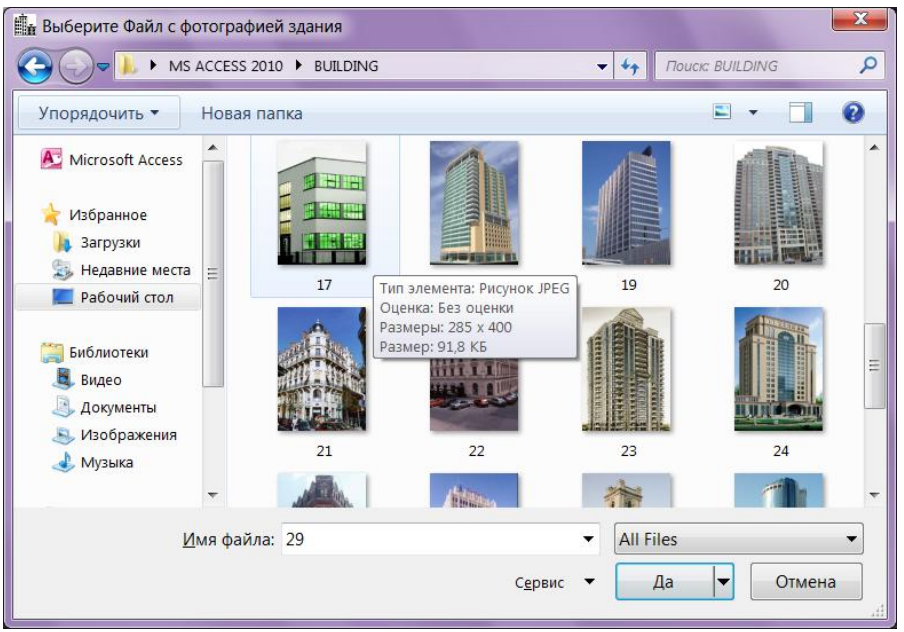


Рис. 9.15. Объект `FileDialog` библиотеки Office в работе

Листинг 9.10. Код кнопки изменения фотографии здания

```
Private Sub ChangePhoto_Click()
' Кнопка "Изменить".
' Поиск файла и занесение его короткого имени в поле txtPicture.
' Подключить: VBA — Tools — Reference -
'             Microsoft Office 14.0 Object Library

' Переменная fDialog типа "объект" — ссылка на окно стандартного
' диалога открытия файла для любого из приложений MS Office 2010
Dim fDialog As Office.FileDialog

' Имя файла с указанием полного пути к нему
Dim varFile As Variant  ' тип Varint — обязательно

' Создание окна диалога
Set fDialog = Application.FileDialog(msoFileDialogFilePicker)
' Изменение свойств объекта и применение методов
With fDialog
      ' Запретить выбор нескольких файлов сразу
      .AllowMultiSelect = False
      ' Заголовок окна
      .Title = "Выберите Файл с фотографией здания"
      ' Удалить фильтр, если он установлен
      .Filters.Clear
      ' Отображать в окне все файлы
      .Filters.Add "All Files", "*.*"
      If .Show = True Then
         ' Выбор файла с фотографией сделан
         For Each varFile In .SelectedItems
            ' Имя файла без полного пути к нему.
            ' Если нужен полный путь — уберите Dir()
            txtPicture.Value = Dir(varFile)
         Next
      Else
         MsgBox "Ни один файл с фотографией здания не выбран.", _
                vbOKOnly + vbExclamation, "Внимание"
      End If
   End With
End Sub
```

Назначим дополнительные свойства переменной `fDialog`, которая ссылается на объект `FileDialog`. Укажем заголовок окна, соответствующий данному действию: "Выберите Файл с фотографией здания". Запретим выбор нескольких файлов одновременно. Такая возможность тоже имеется и просто необходима, если с ок-

ном "связывается" поле со списком, а не текстовое поле. Назначим тип файлов, которые будут отображаться в окне (*.* — все). И, наконец, отправим в `txtPicture` имя выбранного файла с фотографией без указания полного пути к нему: `Dir(varFile)`.

В листинге 9.11 приведен код события **Нажатие кнопки** объекта `CommandSave`. Перед записью информации о здании выполняется ряд проверок. Пользователю нравится, когда система грамотно пишет о его ошибках и предлагает совершить действие, установив курсор в то место, где это действие следует выполнить. Несомненно, MS Access не даст занести в таблицу запись с пустым значением ключа, если улица или номер дома не выбраны, и даже сообщит об этом (рис. 9.16).

Листинг 9.11. Нажатие кнопки *Сохранить*

```
Private Sub CommandSave_Click()
' Кнопка Сохранить
If IsNull(ComboStreet.Value) Then
   ' Улица не введена
   MsgBox "Вы забыли про адрес здания.", _
       vbOKOnly + vbExclamation, "Внимание"
   ' Установка курсора в поле адреса
   ComboStreet.SetFocus
   ' Возврат в форму
   Exit Sub
End If
If IsNull(txtHOUSE.Value) Then
   ' Номер дома не введен
   MsgBox "Вы забыли ввести номер дома.", _
          vbOKOnly + vbExclamation, "Внимание"
   ' Установка курсора в поле номера дома
   txtHOUSE.SetFocus
   Exit Sub
End If
If IsNull(ComboDistrict.Value) Then
   MsgBox "Вы забыли ввести район.", _
          vbOKOnly + vbExclamation, "Внимание"
   ComboDistrict.SetFocus
   Exit Sub
End If
If IsNull(txtLAND.Value) Then
   MsgBox "Вы забыли ввести площадь участка.", _
          vbOKOnly + vbExclamation, "Внимание"
   txtLAND.SetFocus
   Exit Sub
End If
```

```
If IsNull(txtYEAR.Value) Then
   MsgBox "Вы забыли ввести год постройки.", _
       vbOKOnly + vbExclamation, "Внимание"
   txtYEAR.SetFocus
   Exit Sub
End If
If Me.Dirty Then
   ' Если Dirty = True, то данные редактировались.
   ' В этом случае добавляем реквизиты
   txtINSPECTOR.Value = FAMILY ' Инспектор
   txtDATE_UP = Date           ' Дата
   txtTIME_UP = Time()         ' Время
   ' Сохранение записи без закрытия формы
   DoCmd.RunCommand acCmdSaveRecord
   ' Теперь инициировано событие "До обновления".
   ' См. процедуру Sub Form_BeforeUpdate
   Page2.SetFocus
End If
End Sub
```

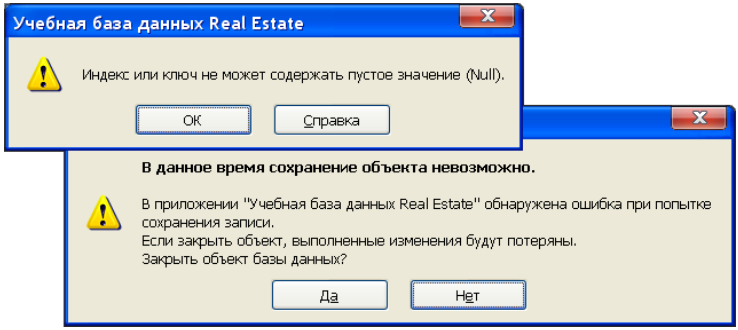


Рис. 9.16. Системное сообщение о неправильных действиях пользователя

Однако системное сообщение, скорее всего, обескуражит рядового пользователя. Посудите сами: ключ, индекс — и `Null`. Не дадим ему возможности читать сообщения MS Access 2010:

```
If IsNull(ComboStreet.Value) Then
   MsgBox "Вы забыли про адрес здания.", _
           vbOKOnly + vbExclamation, "Внимание"
   ComboStreet.SetFocus
   Exit Sub
End If
```

В случае редактирования данных в запись должна быть добавлена информация по специалисту, выполнившему корректировку. Для этого проверяется значение

свойства `Dirty` текущей формы и для `True` выполняется модификация значений соответствующих полей: `txtInspector`, `txtDate_up` и `txtTime_up`. Свойство `Value` в операторе присваивания можно не указывать. В завершение запись сохраняется в таблице без закрытия формы и управление передается второй вкладке:

```
DoCmd.RunCommand acCmdSaveRecord
Page2.SetFocus
```

Листинг 9.12. Кнопка удаления выбранного здания

```
Private Sub CommandDel_Click()
' Кнопка "Удалить"
Dim TextSQL As String ' Строка запроса
' При ошибке перейти к метке: SqlUpdateErr
On Error GoTo SqlUpdateErr
TextSQL = "DELETE from tblBuilding WHERE Street = " & _
          SelectStreet & " AND House = " & SelectHouse
' Выполнение запроса
DoCmd.RunSQL TextSQL
' См. событие "До обновления" этой формы — Form_BeforeUpdate
Indicator = 3
DoCmd.Close    ' Закрытие формы
SqlUpdateErr:
   If Err.Number = 2501 Then
      MsgBox "Удаление здания не выполнено.", _
             vbOKOnly + vbExclamation, "Внимание"
   End If
End Sub
```

Листинг 9.13. Кнопка вызова информации по квартирам

```
Private Sub CommandFlats_Click()
' Кнопка Квартиры создана при помощи мастера кнопок
On Error GoTo Err_CommandFlats_Click
Dim stDocName As String
Dim stLinkCriteria As String
stDocName = "Flats"
DoCmd.OpenForm stDocName, , , stLinkCriteria
Exit_CommandFlats_Click:
    Exit Sub
Err_CommandFlats_Click:
    MsgBox Err.Description
    Resume Exit_CommandFlats_Click
End Sub
```

Листинг 9.14. Кнопка завершения работы с формой

```
Private Sub CommandExit_Click()
' Кнопка Выход
DoCmd.Close    ' Закрытие формы
End Sub
```

Кнопка **Выход** расположена не на вкладке `Page3`, и даже не на наборе вкладок `PageFrame`. Она помещена непосредственно в форму и доступна из любого места формы. Для достижения этого эффекта расположите ее рядом с объектом `PageFrame` и перетащите на него. MS Access поймет, чего вы от него добиваетесь.

Листинги 9.15 и 9.16 содержат код обработки двух событий формы: **До обновления** и **После обновления** соответственно.

Листинг 9.15. Код события *До обновления* формы `PageBuilding`

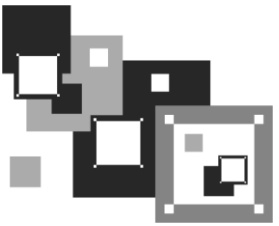
```
Private Sub Form_BeforeUpdate(Cancel As Integer)
' До обновления
Select Case Indicator
  Case 1 ' Штатный режим добавления записи
    If MsgBox("Добавить новую запись в базу данных? ", _
      vbInformation + vbYesNo, "Сохранение") = vbNo Then
      DoCmd.RunCommand acCmdUndo
    End If
    ' Первая вкладка (Поиск) видима
    Page1.Visible = True
    ' Вторая вкладка (Список зданий) видима
    Page2.Visible = True
    Page2.SetFocus
    ' Гашение третьей вкладки
    Page3.Visible = False
    ' Изменение заголовка третьей вкладки
    Page3.Caption = "Просмотр"
    ' Кнопка Квартиры доступна
    CommandFlats.Enabled = True
    ' Кнопка Удалить доступна
    CommandDel.Enabled = True

  Case 2 ' Штатный режим корректировки
    If MsgBox("Сохранить сделанные изменения? ", _
      vbInformation + vbYesNo, "Сохранение") = vbNo Then
      DoCmd.RunCommand acCmdUndo
    End If
```

```
  Case Else
    ' Во всех остальных случаях данные не обновлять
    DoCmd.RunCommand acCmdUndo
End Select
End Sub
```

Листинг 9.16. Код события *После обновления* формы `PageBuilding`

```
Private Sub Form_AfterUpdate()
  ' После обновления.
  ' Обновление запроса для поля со списком
  ListBox.Requery
End Sub
```



ГЛАВА 10

Взаимодействие с другими приложениями

Отчеты, выполненные в конструкторе отчетов MS Access 2010, несмотря на свою универсальность, значительно уступают таблицам MS Excel и документам MS Word в плане комфортности работы с отчетом. В этой главе пойдет речь о расширении функциональных возможностей вашего Access-приложения за счет организации взаимодействия с другими продуктами MS Office. Все составные части этого пакета разработаны таким образом, что работать с ними могут не только конечные пользователи, но и разработчики, обращаясь к ним как к серверам модели COM (Component Object Model, модель составных объектов). На практике это выглядит следующим образом. В нужный момент времени из вашего приложения запускается MS Excel, Word или другой продукт, ему передается выборка данных, с которыми оперирует исходное приложение, обрабатывается, но уже средствами привлеченных продуктов, и выдается конечному пользователю. Этот метод работы широко применяется программистами, пишущими на Delphi, Visual FoxPro и других языках. Для Access-разработчиков задача упрощается. В основе всей линейки MS Office лежит VBA, с которым читатель уже знаком *(см. главу 5)*. Знание других языков не требуется.

10.1. Передача данных в Microsoft Excel

MS Excel наделен замечательной возможностью записи действий пользователя в макрос. Другими словами, все, что делает пользователь в приложении, может быть автоматически запротоколировано в виде текста на языке VBA и после небольшой доработки по существующим правилам запущено на выполнение, но уже непосредственно из исходного приложения и с его исходными данными. Конечный пользователь может даже и не догадаться, что имеет дело с другим продуктом, если вы "погасите" все меню и панели инструментов, оставив на экране его дисплея только результаты своей работы.

10.1.1. Запись макроса

Превратимся на время в конечного пользователя. Запустите MS Excel 2010. Перейдите на вкладку **Вид** ленты главного окна. Найдите значок **Макросы** и щелкните по треугольничку, расположенному ниже значка. Появится меню из трех пунктов. Выберите в нем второй — **Запись макроса**. Откроется окно для ввода имени макроса. Щелкните по кнопке **OK**. Теперь все ваши действия будут протоколироваться. Введите в первые три ячейки текст, выделите их и установите вид шрифта и размер (рис. 10.1). После завершения работы выберите в том же месте главного окна пункт **Остановить запись**.

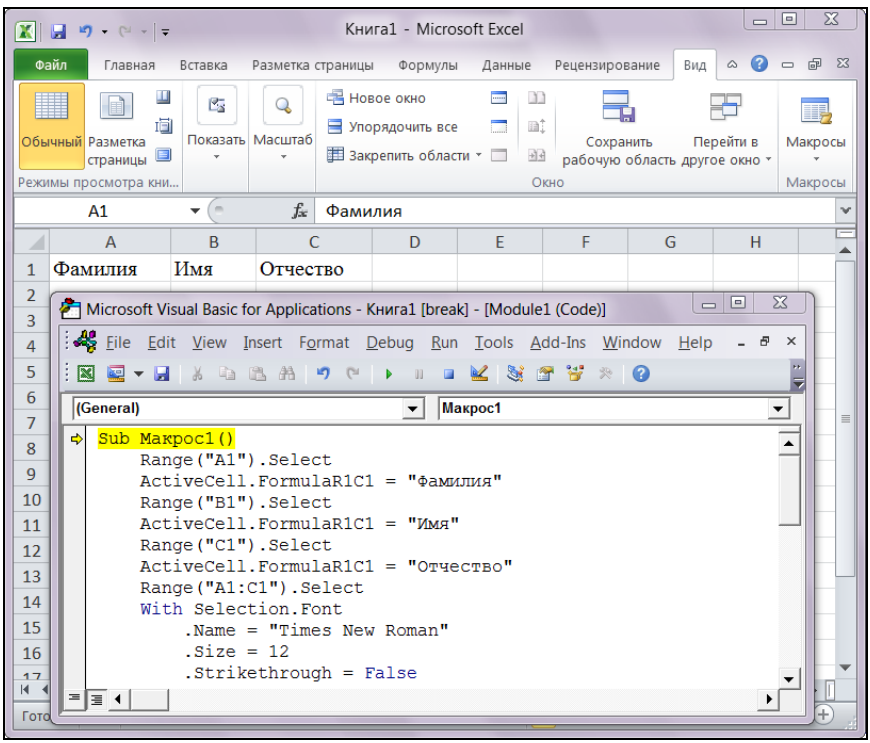


Рис. 10.1. Протокол действий конечного пользователя

Оперируя все той же пиктограммой **Макросы**, просмотрим созданный текст (листинг 10.1). Для этих же целей можно воспользоваться комбинацией клавиш <Alt>+<F8>, которая выводит на экран список всех макросов в документе.

Листинг 10.1. Код, записанный макрорекордером

```
Sub Макрос1()
    Range("A1").Select
    ActiveCell.FormulaR1C1 = "Фамилия"
```

```
    Range("B1").Select
    ActiveCell.FormulaR1C1 = "Имя"
    Range("C1").Select
    ActiveCell.FormulaR1C1 = "Отчество"
    Range("A1:C1").Select
    With Selection.Font
        .Name = "Times New Roman"
        .Size = 12
        .Shadow = False
        .Underline = xlUnderlineStyleNone
        .ThemeColor = xlThemeColorLight1
        .ThemeFont = xlThemeFontNone
    End With
End Sub
```

При записи макросов все продукты MS Office используют множество системных констант. В нашем листинге их три: `xlUnderlineStyleNone`; `xlThemeColorLight1`; `xlThemeFontNone`.

Если передачей документов в MS Office занимаются не VBA-приложения, то для их корректной работы приходится прибегать к различного рода уловкам для извлечения констант из соответствующих библиотек. В нашем случае этого делать не требуется.

10.1.2. Подключение библиотеки Microsoft Excel 14.0 Object Library

Модуль VBA увидит все константы MS Excel (их имена начинаются с двух символов — "xl"), если разработчик подключит дополнительную библиотеку.

Сделать это можно следующим образом:

1. Выберите четвертую вкладку ленты главного окна MS Access 2010. Ее название — **Работа с базами данных**.
2. Запустите Visual Basic, щелкнув по первому значку, расположенному в разделе **Макрос**.
3. Выберите в главном меню VBA пункт **Tools**. Откроется всплывающее меню.
4. Сделайте щелчок по пункту **References**. Появится окно со списком библиотек (рис. 10.2).
5. Поставьте флажок рядом с Microsoft Excel 14.0 Object Library.
6. Нажмите кнопку **OK**.

При дальнейшем запуске accdb- или adp-файла эта библиотека будет подключаться автоматически даже на другом компьютере.

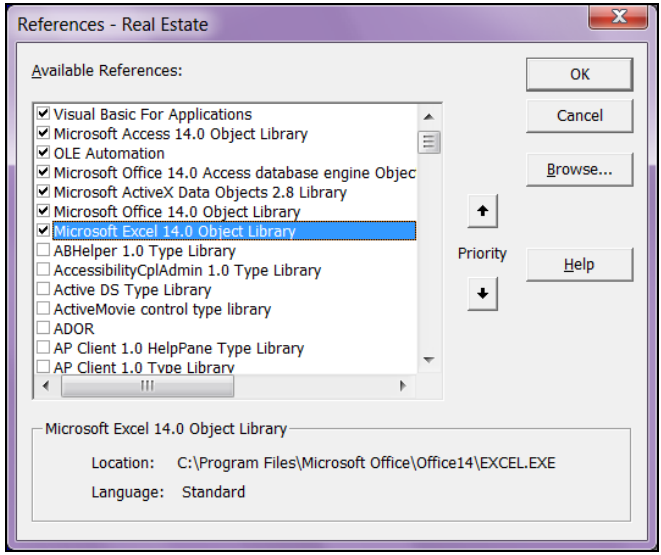


Рис. 10.2. Подключение библиотеки Microsoft Excel 14.0 Object Library

10.1.3. Использование кода макроса MS Excel в приложении MS Access

Алгоритм, который превращает код макроса MS Excel в код, правильно работающий в MS Access, предельно прост. Скопируйте текст макроса, оформите его как процедуру VBA и немного откорректируйте синтаксис (листинг 10.2).

Листинг 10.2. Откорректированный код макроса

```
Public Sub Макрос1()
Dim oExcel As Object
...

With oExcel
    .Range("A1").Select
    .ActiveCell.FormulaR1C1 = "Фамилия"
    .Range("B1").Select
    .ActiveCell.FormulaR1C1 = "Имя"
    .Range("C1").Select
    .ActiveCell.FormulaR1C1 = "Отчество"
    .Range("A1:C1").Select
    With .Selection.Font
         .NAME = "Times New Roman"
         .Size = 12
```

```
        .Shadow = False
        .Underline = xlUnderlineStyleNone
        .ThemeColor = xlThemeColorLight1
        .ThemeFont = xlThemeFontNone
    End With
End With
End Sub
```

Внимательно сравните оба листинга. Добавлено описание объекта. Его имя — `oExcel`. Тремя точками отмечено место в коде, где этот объект должен быть создан. Появился оператор `With`, и перед всеми составными идентификаторами добавлена точка. Как видите, изменения минимальны.

10.1.4. Создание объекта *Application*

Для создания объекта `oExcel` применяется функция `CreateObject`, которая возвращает ссылку на приложение MS Excel (листинг 10.3). Далее MS Excel делается видимым, его окно раскрывается на весь экран, добавляется рабочая книга и устанавливается основной шрифт для нее. Если этого не сделать, то при работе приложения в разных версиях Windows получите проблемы с форматированием ячеек из-за разных типов шрифта, назначенного по умолчанию самой операционной системой. Для ускорения вывода на экран Excel-отчета можно указать:

```
oExcel.Application.Visible = False
```

В этом случае пользователь ничего не будет знать о процессе передачи данных, а сам процесс останется за кадром. Хочу отметить, что большинству работающих с программным комплексом нравится наблюдать за "проделками" VBA, а несколько лишних секунд ничего не значат по сравнению с достаточно эффектным зрелищем.

Листинг 10.3. Создание объекта

```
Public Sub ReportExcel()
   ' Создание объекта
   Set oExcel = CreateObject("Excel.Application")
   ' Сделать окно Microsoft Excel видимым
   oExcel.Application.Visible = True
   ' Полное открытие окна MS Excel
   oExcel.Application.WindowState = xlMaximized
   ' Добавляем рабочую книгу
   oExcel.WorkBooks.Add
   ' При закрытии рабочей книги без сохранения
   ' ошибка формироваться не будет
   oExcel.DisplayAlerts = False
```

```
    ' Масштаб изображения 75%
    oExcel.ActiveWindow.Zoom = 75
    ' Заголовок окна Excel
    oExcel.Caption = "Лицевой счет"
    ' Установка шрифта для всей книги Excel
    oExcel.Cells.Font.NAME = "Arial"
    oExcel.Cells.Font.Size = 7
End Sub
```

10.1.5. Отчет, создаваемый комплексом Real Estate

Щелчок по кнопке **Счет**, размещенной в форме `Flats`, запускает процедуру формирования лицевого счета по квартире. Это бухгалтерский документ, который невозможно создать стандартными средствами MS Access (рис. 10.3).

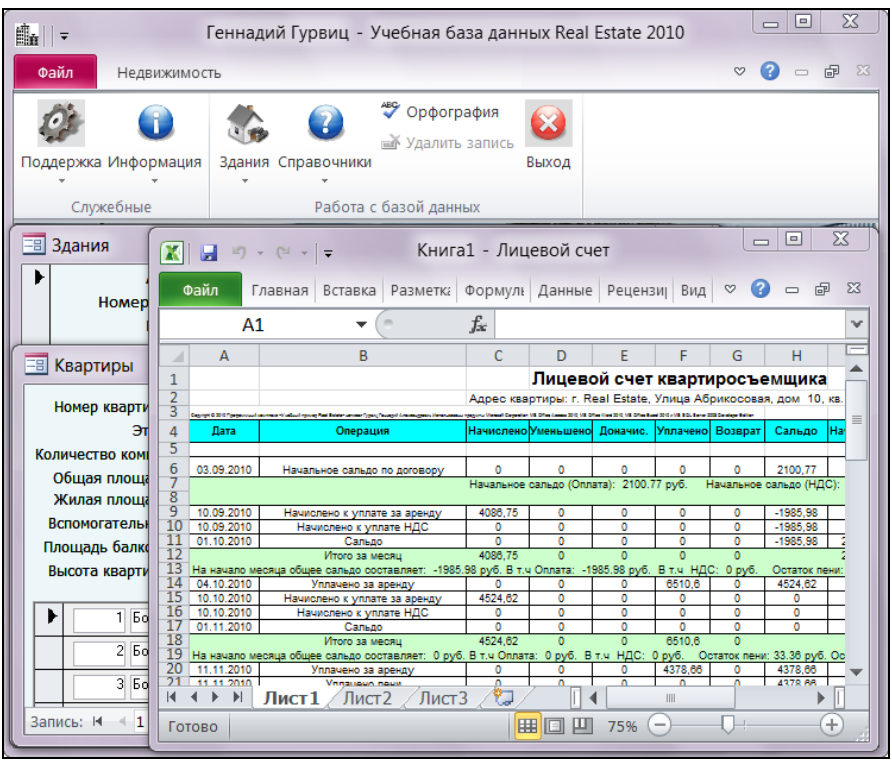


Рис. 10.3. Окончательный вид отчета, сформированного в MS Excel 2010

Мы не будем углубляться в процессы доступа к рабочей книге, ее сохранения и настройки окон, чтения и записи информации в ячейки, работы с объектами в

книге Excel. Внимательно изучите код, представленный в листинге 10.4. Комментариев в нем достаточно. Содержимое листинга можно найти в модуле `ModuleSlave` приложения Real Estate.

Листинг 10.4. Код процедуры, формирующей отчет MS Excel 2010

```
Public Sub AccountExcel()
' Передача лицевого счета в MS Excel.
' Глобальные переменные
' SelectAddressStreet   — Номер выбранной улицы
' SelectAddressHouse    — Номер выбранного дома
' SelectAddressFlat     — Номер выбранной квартиры
' Выборка всех денежных сумм находится в таблице tblExcel.
' tblExcel не имеет отношения к таблицам базы Real Estate.
' В реальном приложении ее заменяет несколько таблиц,
' на основе которых формируется набор RSset
Dim RSset As ADODB.Recordset  ' Набор данных
Set RSset = New ADODB.Recordset
' Рабочие переменные. Их тип — Variant
Dim I, NachAll, YmAll, DonaAll, YpAll, ReturnAll, NachPeniAll
Dim SpPeni, UpPeni, SaldoAll, OstPenu, OstSht
Dim AddressFlat, RightAddress, nRow
Dim SignSaldoStart, SldRussia, SldKray, StartRow
Dim oExcel As Object
' Создание объекта
Set oExcel = CreateObject("Excel.Application")
' Локальные переменные
NachAll = 0       ' Начислено за месяц
YmAll = 0         ' Уменьшено за месяц
DonaAll = 0       ' Доначислено за месяц
YpAll = 0         ' Уплачено за месяц
ReturnAll = 0     ' Возвращено за месяц
NachPeniAll = 0   ' Начислено пени за месяц
SpPeni = 0        ' Списано пени за месяц
UpPeni = 0        ' Уплачено пени за месяц
SaldoAll = 0      ' Сальдо на начало месяца
OstPenu = 0       ' Остаток пени
OstSht = 0        ' Остаток штрафа

' Сделать окно Microsoft Excel видимым
oExcel.Application.Visible = True
' Убираем панели инструментов (только для Excel 2003).
' Стандартная
oExcel.Application.CommandBars("Standard").Visible = False
' Форматирование
oExcel.Application.CommandBars("Formatting").Visible = False
```

```
' Размеры окна и его место на экране дисплея
oExcel.Application.WindowState = xlMaximized
' oExcel.Application.WindowState = xlNormal
' Установить требуемые — по желанию разработчика
' oExcel.Application.Top = 65
' oExcel.Application.Left = 10
' oExcel.Application.Width = 470
' oExcel.Application.Height = 290
' Добавляем рабочую книгу
oExcel.WorkBooks.Add
' При закрытии рабочей книги без сохранения
' ошибка формироваться не будет
oExcel.DisplayAlerts = False
' Масштаб изображения 75%
oExcel.ActiveWindow.Zoom = 75
' Заголовок окна Excel
oExcel.Caption = "Лицевой счет"
' Ориентация альбомная, поля по 1,5 см, бумага А4
oExcel.ActiveSheet.PageSetup.LeftMargin = 42
oExcel.ActiveSheet.PageSetup.RightMargin = 42
oExcel.ActiveSheet.PageSetup.TopMargin = 42
oExcel.ActiveSheet.PageSetup.BottomMargin = 42
oExcel.ActiveSheet.PageSetup.Orientation = xlLandscape
oExcel.ActiveSheet.PageSetup.PaperSize = xlPaperA4
' Для всей Excel-таблицы
oExcel.Cells.Font.NAME = "Arial"
oExcel.Cells.Font.Size = 7
' Заголовок отчета
oExcel.Range("D1").Select
oExcel.ActiveCell.Font.Bold = True
oExcel.ActiveCell.Font.Size = 12
oExcel.ActiveCell.FormulaR1C1 = "Лицевой счет квартиросъемщика"
oExcel.Range("C2").Select
oExcel.ActiveCell.VerticalAlignment = xlTop
oExcel.ActiveCell.Font.Size = 8
' Определение названия улицы и признака адреса по номеру.
' Набор RSset будет содержать только одну строку
With RSset
     ' Задание свойств объекта RSset (Recordset).
     ' Источник: SQL-конструкция
     .Source = "SELECT * FROM tblStreet " & _
               "WHERE STREET = " & SelectAddressStreet
     ' Указатель на открытое соединение
     .ActiveConnection = CurrentProject.Connection
     .CursorType = adOpenKeyset   ' Тип курсора
```

```
     .Open
End With
' Порядок следования в адресе.
' Про индексы RSset написано в разд. 14.8
If RSset(3) = False Then
   ' Признак адреса стоит первым
   RightAddress = Trim(RSset(2)) & " " & Trim(RSset(1))
Else
    ' Признак адреса стоит вторым
   RightAddress = Trim(RSset(1)) & " " & Trim(RSset(2))
End If
RSset.Close ' Закрытие набора данных
AddressFlat = "Адрес квартиры: г. Real Estate, " & _
    RightAddress & ", дом " & _
    Str(SelectAddressHouse) & ", кв. " & _
    Str(SelectAddressFlat) & "."
' Занесение адреса квартиры
oExcel.ActiveCell.FormulaR1C1 = AddressFlat
oExcel.Range("A3").Select
oExcel.ActiveCell.VerticalAlignment = xlBottom
oExcel.ActiveCell.Font.Size = 3
' Микротекст Copyright
oExcel.ActiveCell.FormulaR1C1 = "Copyright © 2010 " & _
    "Программный комплекс <Учебный пример Real Estate> " & _
    "написал Гурвиц Геннадий Александрович " & _
    "Использованы продукты Microsoft Corporation: " & _
    "MS Office Access 2010, MS Office Word 2010, " & _
    "MS Office Excel 2010 и " & _
    "MS SQL Server 2008 Developer Edition"
' Надписи колонок
oExcel.Range("A4").Select
oExcel.ActiveCell.FormulaR1C1 = "Дата"
oExcel.Range("B4").Select
oExcel.ActiveCell.FormulaR1C1 = "Операция"
oExcel.Range("C4").Select
oExcel.ActiveCell.FormulaR1C1 = "Начислено"
oExcel.Range("D4").Select
oExcel.ActiveCell.FormulaR1C1 = "Уменьшено"
oExcel.Range("E4").Select
oExcel.ActiveCell.FormulaR1C1 = "Доначис."
oExcel.Range("F4").Select
oExcel.ActiveCell.FormulaR1C1 = "Уплачено"
oExcel.Range("G4").Select
oExcel.ActiveCell.FormulaR1C1 = "Возврат"
```

```
oExcel.Range("H4").Select
oExcel.ActiveCell.FormulaR1C1 = "Сальдо"
oExcel.Range("I4").Select
oExcel.ActiveCell.FormulaR1C1 = "Нач. пени"
oExcel.Range("J4").Select
oExcel.ActiveCell.FormulaR1C1 = "Количество дней пени"
oExcel.Range("K4").Select
oExcel.ActiveCell.FormulaR1C1 = "Сп. пени"
oExcel.Range("L4").Select
oExcel.ActiveCell.FormulaR1C1 = "Упл.пени"
oExcel.Range("M4").Select
oExcel.ActiveCell.FormulaR1C1 = "Ост.пени"
oExcel.Range("N4").Select
oExcel.ActiveCell.FormulaR1C1 = "Ост.штр"
' Устанавливаем ширину колонок
oExcel.Columns("A:A").ColumnWidth = 8
oExcel.Columns("B:B").ColumnWidth = 25
oExcel.Columns("C:C").ColumnWidth = 7
oExcel.Columns("D:D").ColumnWidth = 7
oExcel.Columns("E:E").ColumnWidth = 7
oExcel.Columns("F:F").ColumnWidth = 6
oExcel.Columns("G:G").ColumnWidth = 6
oExcel.Columns("H:H").ColumnWidth = 7
oExcel.Columns("I:I").ColumnWidth = 6
oExcel.Columns("J:J").ColumnWidth = 18
oExcel.Columns("K:K").ColumnWidth = 6
oExcel.Columns("L:L").ColumnWidth = 6
oExcel.Columns("M:M").ColumnWidth = 6
oExcel.Columns("N:N").ColumnWidth = 6
' Для всех заголовков столбцов жирный шрифт
oExcel.Rows("4:4").Font.Bold = True
oExcel.Rows("4:4").RowHeight = 15
oExcel.Rows("4:4").Font.Size = 7
oExcel.Rows("4:4").VerticalAlignment = xlCenter
oExcel.Rows("4:4").Interior.ColorIndex = 8
oExcel.Rows("4:4").HorizontalAlignment = xlCenter
' Обводим линиями шапку таблицы
oExcel.Range("A4:N4").Select
oExcel.Selection.Borders(xlDiagonalUp).LineStyle = xlNone
' Левые вертикальные линии
With oExcel.Selection.Borders(xlEdgeLeft)
        .LineStyle = xlContinuous
        .Weight = xlThin
        .ColorIndex = xlAutomatic
End With
```

```
' Верхние горизонтальные линии
With oExcel.Selection.Borders(xlEdgeTop)
        .LineStyle = xlContinuous
        .Weight = xlThin
        .ColorIndex = xlAutomatic
End With
' Нижние горизонтальные линии
With oExcel.Selection.Borders(xlEdgeBottom)
        .LineStyle = xlContinuous
        .Weight = xlThin
        .ColorIndex = xlAutomatic
End With
' Правые вертикальные линии
With oExcel.Selection.Borders(xlInsideVertical)
        .LineStyle = xlContinuous
        .Weight = xlThin
        .ColorIndex = xlAutomatic
End With
' Правая вертикальная в последней ячейке
With oExcel.Selection.Borders(xlEdgeRight)
     .LineStyle = xlContinuous
     .Weight = xlThin
     .ColorIndex = xlAutomatic
End With
' Открытие таблицы-выборки лицевого счета
RSset.Open "tblExcel", CurrentProject.Connection
nRow = 5 ' Дальнейший вывод с пятой строки
' Сальдо по бюджетам: федеральный, краевой
SignSaldoStart = 0
SldRussia = 0
SldKray = 0
' Цикл по записям лицевого счета
For I = 0 To RSset.RecordCount — 1
  ' Переход к I-й записи набора RSset
  RSset.Move I, adBookmarkFirst
  Select Case RSset.Fields("CONTENTS").Value
    Case "Сальдо старт"
       SignSaldoStart = 0
       nRow = nRow + 1  ' Номер текущей строчки таблицы Excel
       ' Вывод строки счета. Находится в модуле ModuleSlave
       Call WriteLine(nRow, oExcel, RSset)
       oExcel.Rows(Trim(Str(nRow)) & ":" & _
              Trim(Str(nRow))).Font.Size = 7
       oExcel.Rows(Trim(Str(nRow)) & ":" & _
              Trim(Str(nRow))).RowHeight = 13.2
```

```
      oExcel.Rows(Trim(Str(nRow)) & ":" & _
             Trim(Str(nRow))).HorizontalAlignment = xlCenter
      oExcel.Range("A" & Trim(Str(nRow)) & ":" & _
                "N" & Trim(Str(nRow))).Select
      ' Левые вертикальные линии
      With oExcel.Selection.Borders(xlEdgeLeft)
              .LineStyle = xlContinuous
              .Weight = xlThin
              .ColorIndex = xlAutomatic
      End With
      ' Верхние горизонтальные линии
      With oExcel.Selection.Borders(xlEdgeTop)
              .LineStyle = xlContinuous
              .Weight = xlThin
              .ColorIndex = xlAutomatic
      End With
      ' Нижние горизонтальные линии
      With oExcel.Selection.Borders(xlEdgeBottom)
              .LineStyle = xlContinuous
              .Weight = xlThin
              .ColorIndex = xlAutomatic
      End With
      ' Правые вертикальные линии
      With oExcel.Selection.Borders(xlInsideVertical)
              .LineStyle = xlContinuous
              .Weight = xlThin
              .ColorIndex = xlAutomatic
      End With
      ' Правая вертикальная в последней ячейке
      With oExcel.Selection.Borders(xlEdgeRight)
              .LineStyle = xlContinuous
              .Weight = xlThin
              .ColorIndex = xlAutomatic
      End With
      SignSaldoStart = 1
Case "В т.ч. Аренда"
      SldRussia = RSset.Fields("SALDO").Value
Case "В т.ч. НДС"
      SldKray = RSset.Fields("SALDO").Value
      If SignSaldoStart = 1 Then
         ' Добавление строчки после Сальдо старт
         nRow = nRow + 1
         oExcel.Range("C" + Trim(Str(nRow))).Select
         oExcel.ActiveCell.FormulaR1C1 = _
            "Начальное сальдо (Оплата):  " & _
```

```
        Trim(Str(SldRussia)) & _
        " руб.     Начальное сальдо (НДС):  " & _
        Trim(Str(SldKray)) & " руб."
    oExcel.Rows(Trim(Str(nRow)) & ":" & _
            Trim(Str(nRow))).Font.Size = 7
    oExcel.Rows(Trim(Str(nRow)) & ":" & _
            Trim(Str(nRow))).Interior.ColorIndex = 35
    nRow = nRow + 1
    oExcel.Rows(Trim(Str(nRow)) & ":" & _
            Trim(Str(nRow))).Interior.ColorIndex = 35
    StartRow = nRow
End If
If SignSaldoStart = 0 Then
    oExcel.Range("A" & Trim(Str(StartRow + 1)) & ":N" & _
           Trim(Str(nRow))).Select
    oExcel.Selection.Borders(xlDiagonalUp).LineStyle = xlNone
    oExcel.Selection.HorizontalAlignment = xlCenter
    oExcel.Selection.Font.Size = 7
    ' Левые вертикальные линии
    With oExcel.Selection.Borders(xlEdgeLeft)
            .LineStyle = xlContinuous
            .Weight = xlThin
            .ColorIndex = xlAutomatic
    End With
    ' Верхние горизонтальные линии
    With oExcel.Selection.Borders(xlEdgeTop)
            .LineStyle = xlContinuous
            .Weight = xlThin
            .ColorIndex = xlAutomatic
    End With
    ' Нижняя горизонтальная линия в последней строке
    With oExcel.Selection.Borders(xlEdgeBottom)
            .LineStyle = xlContinuous
            .Weight = xlThin
            .ColorIndex = xlAutomatic
    End With
    ' Правые вертикальные линии
    With oExcel.Selection.Borders(xlInsideVertical)
            .LineStyle = xlContinuous
            .Weight = xlThin
            .ColorIndex = xlAutomatic
    End With
    ' Правая вертикальная в последней ячейке
    With oExcel.Selection.Borders(xlEdgeRight)
            .LineStyle = xlContinuous
```

```
        .Weight = xlThin
        .ColorIndex = xlAutomatic
End With
' Нижние горизонтальные линии в каждой строке
' не выводить, если строчка всего одна
If StartRow + 1 <> nRow Then
      With oExcel.Selection.Borders(xlInsideHorizontal)
        .LineStyle = xlContinuous
        .Weight = xlThin
        .ColorIndex = xlAutomatic
      End With
 End If
 ' Добавление строчки после Сальдо
 nRow = nRow + 1
 oExcel.Range("B" & Trim(Str(nRow))).Select
 oExcel.ActiveCell.FormulaR1C1 = "Итого за месяц"
 oExcel.Range("C" & Trim(Str(nRow))).Select
 oExcel.ActiveCell.FormulaR1C1 = NachAll
 oExcel.Range("D" & Trim(Str(nRow))).Select
 oExcel.ActiveCell.FormulaR1C1 = YmAll
 oExcel.Range("E" & Trim(Str(nRow))).Select
 oExcel.ActiveCell.FormulaR1C1 = DonaAll
 oExcel.Range("F" & Trim(Str(nRow))).Select
 oExcel.ActiveCell.FormulaR1C1 = YpAll
 oExcel.Range("G" & Trim(Str(nRow))).Select
 oExcel.ActiveCell.FormulaR1C1 = ReturnAll
 oExcel.Range("I" & Trim(Str(nRow))).Select
 oExcel.ActiveCell.FormulaR1C1 = NachPeniAll
 oExcel.Range("K" & Trim(Str(nRow))).Select
 oExcel.ActiveCell.FormulaR1C1 = SpPeni
 oExcel.Range("L" + Trim(Str(nRow))).Select
 oExcel.ActiveCell.FormulaR1C1 = UpPeni
 oExcel.Rows(Trim(Str(nRow)) & ":" & _
        Trim(Str(nRow))).Font.Size = 7
 oExcel.Rows(Trim(Str(nRow)) & ":" & _
        Trim(Str(nRow))).Interior.ColorIndex = 35
 oExcel.Rows(Trim(Str(nRow)) & ":" & _
  Trim(Str(nRow))).HorizontalAlignment = xlCenter
 ' Еще одна строчка сводных данных
 nRow = nRow + 1
 StartRow = nRow
 oExcel.Range("A" & Trim(Str(nRow))).Select
 oExcel.ActiveCell.FormulaR1C1 = "На начало месяца" & _
 " общее сальдо составляет:  " & _
 Trim(Str(SaldoAll)) & " руб. В т.ч." & _
```

```
          " Оплата:  " & Trim(Str(SldRussia)) & _
          " руб.  В т.ч.  НДС:  " & Trim(Str(SldKray)) & _
          " руб. " & "   Остаток пени: " & _
          Trim(Str(OstPenu)) & " руб. Остаток штрафа: " & _
          Trim(Str(OstSht)) & " руб."
          oExcel.Rows(Trim(Str(nRow)) & ":" & _
                 Trim(Str(nRow))).Font.Size = 7
          oExcel.Rows(Trim(Str(nRow)) & ":" & _
                 Trim(Str(nRow))).Interior.ColorIndex = 35
          NachAll = 0       ' Начислено за месяц
          YmAll = 0         ' Уменьшено за месяц
          DonaAll = 0       ' Доначислено за месяц
          YpAll = 0         ' Уплачено за месяц
          ReturnAll = 0     ' Возвращено за месяц
          NachPeniAll = 0   ' Начислено пени за месяц
          SpPeni = 0        ' Списано пени за месяц
          UpPeni = 0        ' Уплачено пени за месяц
       End If
Case "Сальдо"
     SignSaldoStart = 0
     nRow = nRow + 1
     ' Начислено пени
     NachPeniAll = NachPeniAll + RSset.Fields("NPE").Value
     ' Сальдо на начало месяца
     SaldoAll = RSset.Fields("SALDO").Value
     ' Остаток пени
     OstPenu = RSset.Fields("OSTPE").Value
     ' Остаток штрафа
     OstSht = RSset.Fields("OSTSH").Value
     ' Вывод строки счета
     Call WriteLine(nRow, oExcel, RSset)
Case Else
    nRow = nRow + 1
    ' Вывод строки счета
    Call WriteLine(nRow, oExcel, RSset)
    ' Уменьшено
    NachAll = NachAll + RSset.Fields("NACH").Value
    ' Доначислено
    DonaAll = DonaAll + RSset.Fields("YPDONA").Value
    ' Уплачено
    YpAll = YpAll + RSset.Fields("YPVS").Value
    ' Возвращено
    ReturnAll = ReturnAll + RSset.Fields("VPE").Value
    ' Начислено пени
    NachPeniAll = NachPeniAll + RSset.Fields("NPE").Value
```

```
      ' Списано пени
      SpPeni = SpPeni + RSset.Fields("YPNEDO").Value
      ' Уплачено пени
      UpPeni = UpPeni + RSset.Fields("RAYPVO").Value
  End Select
Next I
' Переход в начало отчета
oExcel.Range("A1").Select
' Устанавливаем защиту рабочего листа и книги.
' Пароль — текущее время.
' Защита от непунктуального работника.
' Пачка квитанций на клавиатуре — и ячейка изменена!
oExcel.ActiveSheet.Protect ("' + Time() + '")
oExcel.ActiveWorkbook.Protect ("' + Time() + '")
RSset.Close
Set RSset = Nothing
End Sub
```

Совет

В самом начале своей работы не пытайтесь основательно разобраться в объектной модели MS Excel, этого не требуется. Просто вспоминайте "фокус" с записью макроса. Проделайте вручную то, что должен сделать компьютер, и воспользуйтесь его творением. В большинстве случаев этого будет достаточно.

Как правило, код формирования Excel-отчета оформляется в виде процедуры с входными параметрами. Процедура `AccountExcel` написана без параметров. Исходные данные передаются ей через глобальные переменные (номер улицы, номер дома и номер квартиры). Все остальные данные формируются из таблиц базы данных на основании значений этих переменных. Для извлечения данных применяется объект `Recordset` из объектной модели доступа к данным (ADO):

```
Dim RSset As ADODB.Recordset  ' Набор данных
Set RSset = New ADODB.Recordset
With RSset
     ' Задание свойств объекта RSset (Recordset)
     ' Источник: SQL-конструкция
     .Source = "SELECT * FROM tblStreet " & _
               "WHERE STREET = " & SelectAddressStreet
     ' Указатель на открытое соединение
     .ActiveConnection = CurrentProject.Connection
     .CursorType = adOpenKeyset   ' Тип курсора
     .Open
End With
```

Для перемещения по записям набора `RSset` (`Recordset`) может применяться пять методов:

```
RSset.MoveFirst     ' Перемещение к первой записи набора
RSset.MoveNext      ' Перемещение к следующей записи набора
RSset.MovePrevious  ' Перемещение к предыдущей записи
RSset.Move          ' Перемещение к указанной записи
RSset.MoveLast      ' Перемещение к последней записи
```

Перемещение к указанной записи (метод `Move`) подробно рассматривается в *разд. 14.8*. Остальные четыре метода в комментариях не нуждаются.

Сделав при помощи нужной строчки кода (любой из пяти) требуемую запись текущей, можно извлекать из нее данные, например:

```
SaldoAll = RSset.Fields("SALDO").Value
oExcel.ActiveCell.FormulaR1C1 = SaldoAll
```

В кавычках указывается имя поля из набора `Recordset`.

Для размещения данных в Excel-отчете можно также использовать стандартный Access-прием, например:

```
oExcel.ActiveCell.FormulaR1C1 = [Forms]![Flats]![txtFlat].Value
```

где в качестве значения активной ячейки MS Excel используется значение текстового поля `txtFlat` формы `Flats`.

Совет

Если отчет предназначен для субъекта, который может стать потенциальным заказчиком вашей следующей разработки, включите в него микротекст "Copyright ©" с указанием ваших координат. Без работы не останетесь! Пакет заказов автора на разработку программного обеспечения и по сей день пополняется за счет этого законного трюка.

10.2. Передача данных в Microsoft Word

Нашей компании Real Estate на основании данных, имеющихся в базе, необходимо создать отчет, но не обычный, а представляющий собой документ на приватизацию квартиры. Внесение записей в документ от руки не допускается. Только подписи и печати договаривающихся сторон. Документ должен быть оформлен по всем правилам грамматики. Вид его зависит от исходных данных, следовательно, применение каких-либо шаблонов исключается. Задача серьезная, и решить ее нам поможет применение MS Word.

Выполним ряд подготовительных операций. Добавим в форму `Flats` кнопку **Документ**, код нажатия которой приведен в листинге 10.5.

Листинг 10.5. Код нажатия кнопки *Документ*

```
Private Sub CommandTreaty_Click()
' Кнопка "Документ".
' Глобальные переменные.
' Номер квартиры для приватизации
SelectAddressFlat = Me!txtFLAT.Value
' Количество комнат в квартире
SelectFlatRooms = Me!ROOMS.Value
' Проверка количества проживающих
If DCount("*", "tblOwners", _
    "STREET = " & SelectAddressStreet & _
    " AND HOUSE = " & SelectAddressHouse & _
    " AND FLAT = " & SelectAddressFlat) = 0 Then
    MsgBox "В квартире нет проживающих.", _
       vbOKOnly + vbExclamation, "Внимание"
Else
   ' Вызов формы для генерации договора приватизации
   DoCmd.OpenForm "Treaty"
End If
End Sub
```

Создадим форму `Treaty` для добавления внешних данных в документ (рис. 10.4). Следует отметить, что не вся информация для создания договора приватизации имеется в базе. Дадим возможность пользователю несколькими щелчками мыши сделать необходимые добавления.

Нажатие кнопки **Документ Word** (листинг 10.6) запускает на выполнение процедуру `TreatyWord`, которая находится в модуле `ModuleSlave`, и сообщает ей ряд фактических параметров: дату подписания договора, тип квартиры, форму собственности и т. д.

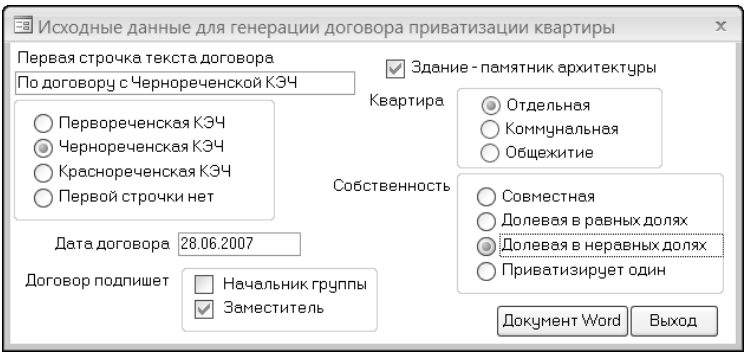


Рис. 10.4. Форма `Treaty` для ввода внешних данных

Листинг 10.6. Код нажатия кнопки *Документ Word*

```
Private Sub CommandSave_Click()
' Параметры, передаваемые в процедуру.
' Дата подписания договора
SelectDateTreaty = Me![txtDate].Value
' Фамилия, подписавшего договор
SelectChief = Me![OptionGroup2].Value
' Первая строка договора
SelectText = Me![txtSelectText].Value
' Здание — памятник архитектуры
If IsNull(Me![CheckBox1].Value) Then
   SelectMemorial = False
Else
   SelectMemorial = Me![CheckBox1].Value
End If
' Тип квартиры
SelectTypeFlat = Me![OptionGroup3].Value
' Форма собственности
SelectTypeKind = Me![OptionGroup3].Value
' Процедура находится в модуле ModuleSlave
Call TreatyWord(SelectDateTreaty, SelectChief, SelectText, _
     SelectMemorial, SelectTypeFlat, SelectTypeKind)
End Sub
```

Далее необходимо применить трюк, использовавшийся при работе с MS Excel, связанный с применением макроса. Наберем с клавиатуры образец договора приватизации, предварительно включив запись макроса. Посмотрим, что выдал макрорекордер. Первое, что бросается в глаза, — наличие объекта `Selection`. Его не было в MS Excel.

Для отделения одной порции данных от другой в MS Excel используется объект `Cell`. В MS Word — другой подход. В нем для выделения текста применяется как раз `Selection`, который в текстовом редакторе представляет собой выделенный текст. Если никакой текст не выделен, то `Selection` имеет дело с текущим положением курсора.

Следующий этап заключается в небольшой модификации полученного текста и помещения его в модуль MS Access. Для того чтобы MS Access не "ругался" на константы MS Word, сделайте следующее:

1. Выберите четвертую вкладку ленты главного окна MS Access 2010. Ее название — **Работа с базами данных**.
2. Запустите Visual Basic, щелкнув по первому значку, расположенному в разделе **Макрос**.
3. Выберите в главном меню VBA пункт **Tools**. Откроется всплывающее меню.

4. Сделайте щелчок по пункту **References**. Появится окно со списком библиотек.
5. Поставьте флажок рядом с Microsoft Word 14.0 Object Library (рис. 10.5).
6. Нажмите кнопку **OK**.

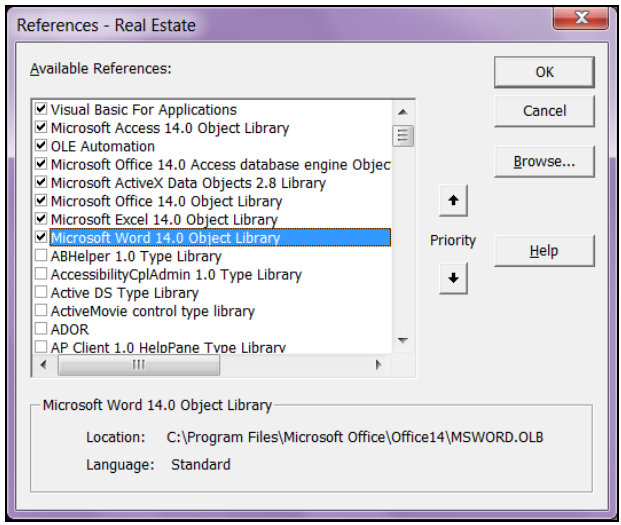


Рис. 10.5. Подключение библиотеки Microsoft Word 14.0 Object Library

Изменения, которые необходимо внести в текст макроса, как и в случае с MS Excel, минимальны. Необходимо создать объект `oWord`, сделать его видимым и добавить точку перед составными идентификаторами. Текст процедуры передачи данных в MS Word приведен в листинге 10.7.

Листинг 10.7. Код процедуры, формирующей отчет MS Word

```
Public Sub TreatyWord(SelectDateTreaty, SelectChief, _
       SelectText, SelectMemorial, SelectTypeFlat, _
       SelectTypeKind)
' Передача данных в Microsoft Word.
' Описание рабочих переменных
Dim I, SelectRoom, SelectRoom1, SelectRoom2, ChiefShot, _
       ChiefLong, DayText, DayMonthYear, lcText, _
       RightAddress, SelectHouse, SelectFlat
' Объект Word
Dim oWord As Object
Dim RSset As ADODB.Recordset  ' Набор данных
Set RSset = New ADODB.Recordset
' Решение "проблемы" русского языка
Select Case SelectFlatRooms
```

```
   Case 1
        SelectRoom = "одной комнаты"
        SelectRoom1 = "однокомнатной"
        SelectRoom2 = "одну комнату"
   Case 2
        SelectRoom = "двух комнат"
        SelectRoom1 = "двухкомнатной"
        SelectRoom2 = "две комнаты"
   Case 3
        SelectRoom = "трех комнат"
        SelectRoom1 = "трехкомнатной"
        SelectRoom2 = "три комнаты"
   Case Else
        SelectRoom = Str(SelectFlatRooms) + " комнат"
        SelectRoom1 = Str(SelectFlatRooms) + " комнатной"
        SelectRoom2 = Str(SelectFlatRooms) + " комнат"
End Select
Select Case SelectChief
   Case 1
        ChiefShot = "Рощин В.Н."
        ChiefLong = "Рощина Владимира Николаевича"
   Case 2
        ChiefShot = "Симонова Л.И."
        ChiefLong = "Симоновой Людмилы Ивановны"
End Select
' Дата прописью
DayMonthYear = ""
' Процедура находится в модуле ModuleSlave
Call Detail(SelectDateTreaty, DayMonthYear)
' SelectDateTreaty — дата
' DayMonthYear — дата прописью
' Создание объекта MS Word
Set oWord = CreateObject("Word.Application")
' Сделать окно MS Word видимым
oWord.Visible = True
' Заголовок окна MS Word
oWord.Caption = "Договор на передачу квартиры в собственность "
oWord.Documents.Add
' Ориентация книжная, бумага A4
' Поля (1 см = 28 пунктов)
With oWord.ActiveDocument.PageSetup
     .LineNumbering.Active = False
     .Orientation = wdOrientPortrait
     .LeftMargin = 48
     .RightMargin = 40
```

```
     .TopMargin = 56
     .BottomMargin = 56
End With
' Включить расстановку переносов
With oWord.ActiveDocument
     .AutoHyphenation = True
     .HyphenateCaps = True
     .ConsecutiveHyphensLimit = 0
End With
With oWord.Selection
     ' Параметры шрифта
     .Font.NAME = "Times New Roman"
     .Font.Size = 14
     .Font.Bold = wdToggle
     .ParagraphFormat.Alignment = wdAlignParagraphCenter
     .TypeText ("Договор")
     .TypeParagraph
     .Font.Size = 12
     lcText = "на передачу квартиры в собственность"
     .TypeText (lcText)
     If SelectMemorial = True Then
        ' Если здание — памятник архитектуры
        .TypeParagraph
        lcText = "в жилом доме — памятнике истории и культуры"
        .TypeText (lcText)
     End If
     .TypeParagraph
     .Font.Size = 12
     .Font.Bold = wdToggle
     .ParagraphFormat.Alignment = wdAlignParagraphJustify
     .TypeText ("г. Real Estate")
     .TypeParagraph
     .ParagraphFormat.Alignment = wdAlignParagraphCenter
     lcText = Trim(DayMonthYear) ' Дата прописью
     .TypeText (lcText)
     ' Первая строка договора
     If Len(Trim(SelectText)) <> 0 Then
        .TypeParagraph
        .ParagraphFormat.Alignment = wdAlignParagraphJustifyMed
        lcText = "       " & Trim(SelectText)
        .TypeText (lcText)
     End If
     .TypeParagraph
     .ParagraphFormat.Alignment = wdAlignParagraphJustifyMed
     lcText = "     Администрация города Real Estate в лице" & _
```

```
      " начальника Управления жилищного фонда " & ChiefLong & _
      " в соответствии с Положением об Управлении фонда " & _
      "города, утвержденным постановлением мэра города от " & _
      "31.12.2000 № 1000, именуемая в дальнейшем " & _
      "Продавец, с одной стороны и гр."
.TypeText (lcText)
.TypeParagraph
.Font.Bold = wdToggle
.Font.Underline = wdUnderlineSingle
.ParagraphFormat.Alignment = wdAlignParagraphCenter
' Набор RSset содержит выборку по проживающим в квартире
With RSset
   ' Задание свойств объекта RSset (Recordset).
   ' Источник: SQL-конструкция
   .Source = "SELECT * FROM tblOwners " & _
             "WHERE STREET = " & SelectAddressStreet & _
             " AND HOUSE = " & SelectAddressHouse & _
             " AND FLAT = " & SelectAddressFlat
   ' Указатель на открытое соединение
   .ActiveConnection = CurrentProject.Connection
   .CursorType = adOpenKeyset   ' Тип курсора
   .Open
End With
' Цикл по проживающим
For I = 0 To RSset.RecordCount — 1
    ' Переход к I-й записи
    RSset.Move I, adBookmarkFirst
    ' Используется сокращенная форма записи
    ' для доступа к полям набора RSset
    lcText = Trim(RSset(4)) & " " & Trim(RSset(5)) & " " & _
             Trim(RSset(6)) & " — " & Str(RSset(7)) & " г.р."
    .TypeText (lcText)
    .TypeParagraph
Next I
If RSset.RecordCount > 0 Then
   ' Печатать, если собственников более одного.
   ' Первая запись имеет индекс 0
   Select Case SelectTypeKind
      Case 1
           lcText = "(совместная собственность)"
      Case 2
           lcText = "(в равных долях)"
      Case 3
           lcText = "(долевая собственность)"
   End Select
```

```
   .TypeText (lcText)
End If
RSset.Close ' Закрытие набора данных
.TypeParagraph
.TypeParagraph
.Font.Bold = wdToggle
.Font.Underline = wdUnderlineNone
.ParagraphFormat.Alignment = wdAlignParagraphJustify
lcText = "именуемый в дальнейшем Покупатель, заключили " & _
         "настоящий договор о нижеследующем:"
.TypeText (lcText)
.TypeParagraph
With RSset
   ' Задание новых свойств объекта RSset (Recordset)
   ' Источник: данные по квартире
   .Source = "SELECT * FROM tblFlats " & _
             "WHERE STREET = " & SelectAddressStreet & _
             " AND HOUSE = " & SelectAddressHouse & _
             " AND FLAT = " & SelectAddressFlat
   ' Указатель на открытое соединение
   .ActiveConnection = CurrentProject.Connection
   .CursorType = adOpenKeyset   ' Тип курсора
   .Open
End With
RSset.MoveFirst ' Переход к первой записи набора
If SelectTypeFlat = 1 Then
   ' Квартира отдельная
   lcText = "       1. Продавец передал в собственность, " & _
   "а Покупатель приобрел квартиру, " & _
   "состоящую из " & Trim(SelectRoom) & " общей площадью " & _
   Trim(Str(RSset(5))) & _
   " кв. м., в том числе жилой " & _
   Trim(Str(RSset(6))) & " кв. м., по адресу: "
Else
   ' Квартира коммунальная
   lcText = "       1. Продавец передал в собственность, " & _
   "а Покупатель приобрел часть коммунальной квартиры " & _
   "общей площадью " & Trim(Str(RSset(5))) & _
   " кв. м., в том числе жилой " & _
   Trim(Str(RSset(6))) & " кв. м. " & _
   "Данная доля включает " & Trim(SelectRoom2) & _
   " площадью " & Trim(Str(RSset(5))) & _
   " кв. м., в том числе жилой " & _
   Trim(Str(RSset(6))) & " кв. м. и часть помещений" & _
   " общего пользования квартиры, " & _
   "пропорционально занимаемой жилой площади, по адресу: "
End If
```

```
.TypeText (lcText)
RSset.Close ' Закрытие набора данных
lcText = "г. Real Estate, "
' Определение названия улицы и признака адреса по номеру.
' Набор RSset будет содержать только одну строку
With RSset
   ' Задание свойств объекта RSset (Recordset)
   ' Источник: данные по улице
   .Source = "SELECT * FROM tblStreet " & _
          "WHERE STREET = " & SelectAddressStreet
   ' Указатель на открытое соединение
   .ActiveConnection = CurrentProject.Connection
   .CursorType = adOpenKeyset   ' Тип курсора
   .Open
End With
' Порядок следования в адресе.
' Про индексы RSset написано в разд. 14.8
If RSset(3) = False Then
   ' Признак адреса стоит первым
   RightAddress = Trim(RSset(2)) & " " & Trim(RSset(1))
Else
   ' Признак адреса стоит вторым
   RightAddress = Trim(RSset(1)) & " " & Trim(RSset(2))
End If
lcText = lcText & RightAddress & _
       ", дом " & Trim(SelectAddressHouse) & ", кв. " & _
       Trim(Str(SelectAddressFlat)) + "."
RSset.Close ' Закрытие набора данных
.Font.Underline = wdUnderlineSingle
.Font.Bold = wdToggle
.TypeText (lcText)
.TypeParagraph
.Font.Bold = wdToggle
.Font.Underline = wdUnderlineNone
.ParagraphFormat.Alignment = wdAlignParagraphJustifyMed
lcText = "       2. Покупатель приобрел право " & _
"собственности с момента государственной регистрации " & _
"права в едином государственном реестре. "
.TypeText (lcText)
.TypeParagraph
lcText = "       3. Права и обязанности, возникающие " & _
"из настоящего договора, Покупателю разъяснены."
.TypeText (lcText)
.TypeParagraph
lcText = "       4. В случае смерти Покупателя все права " & _
"и обязанности по настоящему договору " & _
"переходят к его наследникам на общих основаниях."
```

```
.TypeText (lcText)
.TypeParagraph
lcText = "       5. Пользование квартирой производится " & _
"Покупателем применительно к Правилам " & _
"пользования жилыми помещениями, содержания жилого дома " & _
"и придомовой территории в РСФСР."
If SelectMemorial = True Then
   lcText = lcText & " и Положению " & _
   "об охране и использовании памятников истории и " & _
   "культуры от 16.09.1982 № 865."
End If
.TypeText (lcText)
.TypeParagraph
.ParagraphFormat.Alignment = wdAlignParagraphJustifyMed
lcText = "       6. Настоящий договор составлен в трех " & _
  "экземплярах, из которых один выдается " & _
  "Покупателю, один — Продавцу, один остается в " & _
  "краевом учреждении юстиции " & _
  "по государственной регистрации прав на недвижимое " & _
  "имущество и сделок с ним."
.TypeText (lcText)
If SelectMemorial = True Then
.TypeParagraph
lcText = "       7. Покупатель обязан заключить Охранное " & _
  "свидетельство по использованию квартиры " & _
  "в доме-памятнике со специально уполномоченным " & _
  "государственным органом охраны памятников по " & _
  "установленной форме."
.TypeText (lcText)
End If
.TypeParagraph
.TypeParagraph
.ParagraphFormat.Alignment = wdAlignParagraphCenter
.Font.Bold = wdToggle
lcText = "Адреса сторон:"
.TypeText (lcText)
.TypeParagraph
.TypeParagraph
lcText = "Подпись Продавца"
.ParagraphFormat.Alignment = wdAlignParagraphJustify
.Font.Bold = wdToggle
.TypeText (lcText)
.TypeParagraph
' Выравнивание по правому краю
.ParagraphFormat.Alignment = wdAlignParagraphRight
```

```
      lcText = "Подпись Покупателя"
      .TypeText (lcText)
      .TypeParagraph
      .TypeParagraph
      .Font.NAME = "Courier new"
      .Font.Size = 12
      .ParagraphFormat.Alignment = wdAlignParagraphJustify
      lcText = "_________________ "
      .TypeText (lcText)
      .Font.Bold = wdToggle
      lcText = Trim(ChiefShot)
      .TypeText (lcText)
End With
' Переход в начало документа
oWord.ActiveWindow.ActivePane.VerticalPercentScrolled = 0
Set RSset = Nothing
End Sub
```

В тексте процедуры `TreateWord` вызывается процедура `Detail`, которая возвращает в точку вызова строку, представляющую собой дату прописью. Текст ее находится в модуле `ModuleSlave`, входящем в состав программного комплекса.

10.3. Создание системы оперативной справки

Существует множество программных продуктов, предназначенных для разработки собственной системы оперативной справки. Рассмотрим один из них — HTML Help Workshop. Это самостоятельный продукт фирмы Microsoft. Его первая версия входила в состав Visual FoxPro 6.0. Сейчас он не является составной частью какого-либо продукта. Самую последнюю версию можно найти на Web-сервере по адресу:

http://msdn.microsoft.com/en-us/library/ms669985(VS.85).aspx

В этой главе рассмотрена работа с HTML Help Workshop 1.3. Обязательно прочтите файл ReadMe, входящий в состав продукта. До сих пор в HTML Help Workshop имеется ряд проблем. В этом файле разработчики разъясняют способы их решения.

Для отображения системы оперативной справки HTML Help Workshop использует Internet Explorer. После компиляции системы справки HTML Help Workshop создает файл с расширением chm. Это файл контекстно-зависимой справки. Находясь в любом месте программного комплекса, пользователь может получить помощь, нажав клавишу <F1>.

Рассмотрим последовательно все этапы создания файла справки RealEstate.chm к нашему приложению.

10.3.1. Создание HTML-страниц

Для каждой страницы оперативной справки создайте отдельный HTML-файл. Все страницы поместите в папку с именем HTML папки HELP. Так же в папке HELP необходимо создать еще две: IMAGES — для хранения изображений, рисунков и файлов фона страниц; SOUND — для хранения мелодий, сопровождающих работу системы оперативной справки.

Для создания HTML-файлов можно использовать Microsoft Word любой версии до MS Word 2007. Применение последних версий не позволит сопроводить работу справки музыкой в фоновом режиме. В настоящее время наряду с выпуском трех новых средств на основе современных технологий для разработки приложений и создания Web-объектов — Microsoft® Office SharePoint® Designer, Microsoft® Expression™ Web Designer и Microsoft® Visual Studio, корпорация Microsoft продолжает и техническое обслуживание пользователей всех существующих версий FrontPage — специализированного средства для создания простых HTML-страниц.

Выполним создание HTML-страниц нашей системы оперативной справки при помощи Microsoft FrontPage.

Запустите его на выполнение. В главном меню выберите пункт **Формат**. В появившемся меню — пункт **Фон**. Появится окно **Свойства страницы** (рис. 10.6).

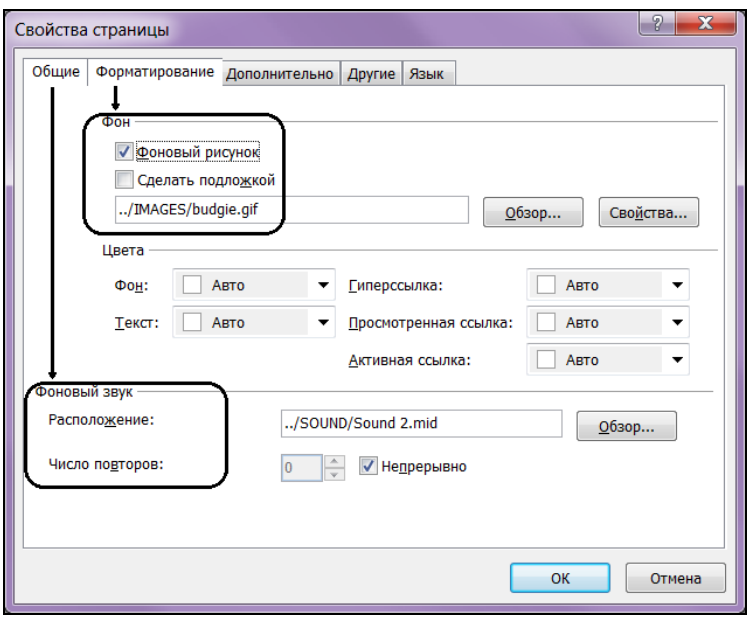


Рис. 10.6. Окно **Свойства страницы** Microsoft FrontPage

Побеспокойтесь о фоновом рисунке и фоновом звуке каждой страницы. Это позволит качественно поднять уровень оформления системы справки с минимальными затратами. Окно **Свойства страницы** содержит пять вкладок. Две из них: **Общие** и **Форматирование** условно открыты одновременно. Это сделано для наглядности.

Вкладка **Общие** служит для назначения странице фонового звука — мелодии, которая будет звучать все время, пока пользователь видит ее содержимое. Вкладка **Форматирование** предназначена для назначения фона страницы. После компоновки HTML-страницы выполните ее сохранение. Для этого в главном меню MS FrontPage выберите пункт **Файл**, в появившемся меню — пункт **Сохранить как**. Укажите тип файла: **Файлы HTML**. Установите гиперссылки между HTML-файлами. Ссылок должно быть столько, сколько необходимо для толковой работы.

10.3.2. Создание проекта

Запустите HTML Help Workshop. В его главном окне выберите пункт **File** (Файл). В открывшемся меню пункт **New** (Новый). Появится окно **New**. Выберите в нем первый пункт **Project** (Проект) и щелкните по кнопке **OK**. HTML Help Workshop предложит ввести имя проекта и включить в него уже имеющиеся у разработчика компоненты проекта. У нас пока ничего нет. Ограничимся именем RealEstate.hhp.

10.3.3. Включение страниц в HTML-проект

Откройте файл проекта RealEstate.hhp. На первой вкладке с именем **Project** (Проект) увидите семь пиктограмм (рис. 10.7).

Выберите вторую пиктограмму  **Add/Remove topic files** (Включение в проект HTML-страниц). Добавьте в проект все созданные на первом этапе HTML-страницы.

Определимся с первой пиктограммой  **Change project options** (Определение параметров проекта). Щелчок по ней откроет окно **Options** (Опции) с четырьмя вкладками. На вкладке **General** (Общие) определите:

- заголовок окна справки;
- файл страницы, которая будет отображена в окне при первом запуске системы оперативной справки;
- язык и шрифт.

На вкладке **Files** (Файлы) укажите название и расположение файла скомпилированной справки, а также файлов с содержанием тем справки и с указателями.

Вид вкладки **Compiler** (Компилятор) показан на рис. 10.8. Отметьте флажками необходимые объекты и действия.

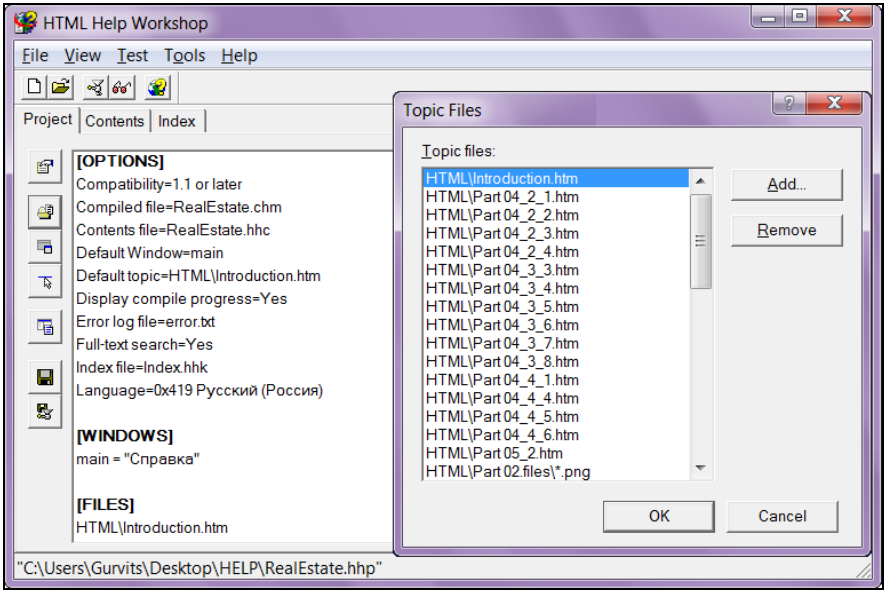


Рис. 10.7. Проект RealEstate.hhp в окне HTML Help Workshop

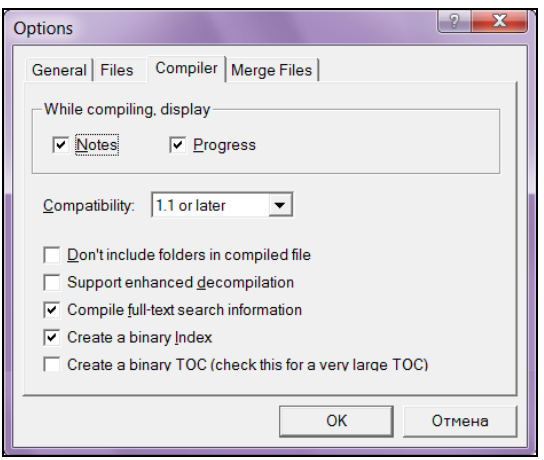


Рис. 10.8. Вкладка опций компилятора

10.3.4. Создание содержания справочной системы

Для создания содержания справочной системы перейдем на вторую вкладку HTML Help Workshop с названием **Contents** (Содержание). На ней находятся

11 пиктограмм (рис. 10.9). Первая пиктограмма **Contents properties** (Свойства содержания) предназначена для выбора внешнего вида вкладки с содержанием справочной системы. Выберите тип заголовка (значок папки или раскрытой книги). Можно также назначить свои значки.

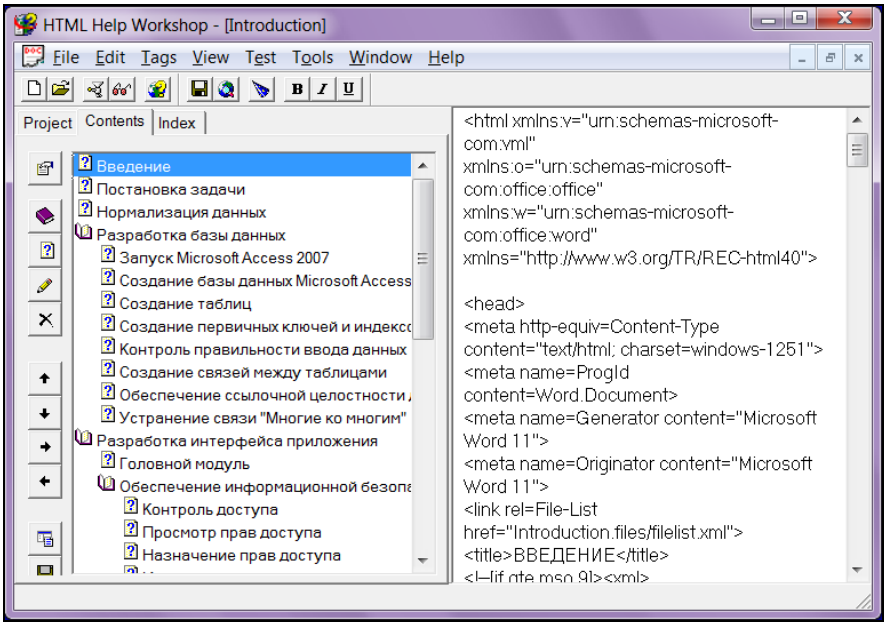


Рис. 10.9. Вторая вкладка HTML Help Workshop — **Contents**

Для добавления заголовка щелкните по второй пиктограмме **Insert a heading** (Вставить заголовок). Для добавления страницы предназначена третья пиктограмма **Insert a page** (Вставить страницу). Содержание тем справочной системы можно расположить в иерархическом виде. Допускается добавление заголовков нескольких уровней вложенности.

Для изменения уровня заголовка или страницы воспользуйтесь пиктограммами со стрелками.

После четырех этапов мы создали все основные компоненты справки. Отсутствуют только средства контекстно-зависимой справочной системы. Однако даже в таком виде наша оперативная справка работоспособна. Откомпилируйте ее, щелкнув по последней пиктограмме **Save all files and compile** (Сохранить все файлы и откомпилировать) первой вкладки **Project** (Проект) продукта HTML Help Workshop. При запуске на выполнение файла RealEstate.chm всегда будет отображаться назначенная стартовой HTML-страница **Introduction** (Введение).

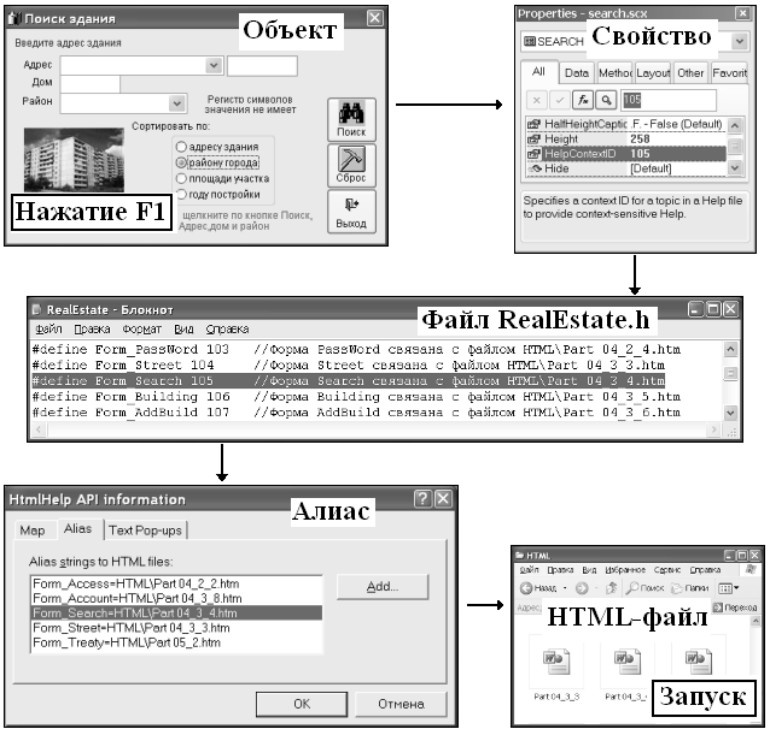


Рис. 10.10. Связь между нажатием клавиши <F1> и запуском нужной HTML-страницы

Для создания справочной системы, работающей совместно с приложением, необходимо связать объекты MS Access нашего приложения с файлами HTML-страниц. Как правило, это сами формы и объекты, расположенные в них. Посредниками в этом деле выступают псевдонимы тем, константы **HelpContextID** (индексы тем) и файл связи RealEstate.h. Вид цепочки связи показан на рис. 10.10.

10.3.5. Назначение псевдонимов тем

Для назначения псевдонимов тем, необходимых для создания контекстно-зависимой справочной системы, откройте окно **HtmlHelp API information** (рис. 10.11), выбрав четвертую пиктограмму  первой вкладки **Project** (Проект) продукта HTML Help Workshop.

Перейдите на вторую вкладку — **Alias** (Псевдоним). После нажатия кнопки **Add** (Добавить) откроется окно **Alias**. Укажите в нем псевдоним, имя файла и комментарий. Имя псевдонима вводится с клавиатуры. Имя HTML-файла выбирается из списка. Если файла в списке нет, вернитесь к третьему этапу и добавьте в список файлов отсутствующий.

Имя псевдонима HTML-файла лучше всего назначать осмысленно. Лениться не следует. Даже в небольшом файле справки можно запутаться с именами объектов и соответствующими им файлами. Про имена типа "A1" или "Aaa" следует забыть. Два примера имени псевдонима:

```
Form_Building                    (Форма просмотра списка зданий)
Form_Building_Combo1_Street      (Список улиц в форме зданий)
```

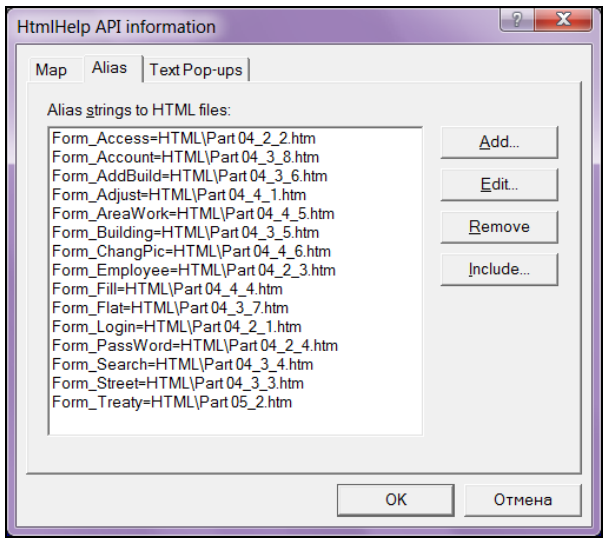


Рис. 10.11. Вторая вкладка окна **HtmlHelp API information**

10.3.6. Назначение индексов тем

Разберемся с индексами тем (свойство `HelpContextID` объекта MS Access). Запустите MS Access. Выберите объект. Это может быть `Form`, `CheckBox`, `ComboBox`, `CommandButton`, `EditBox`, `Grid`, `Image`, `Label`, `ListBox`, `Page` и др. Установите в качестве значения свойства `HelpContextID` выбранного объекта уникальное число. Сколько объектов вы хотите связать с файлами страниц — столько чисел. Каких — не важно, главное, чтобы они не повторялись. В примере Real Estate они пронумерованы по порядку, начиная с сотни.

Добавьте в текст события **Загрузка** каждой формы строчку назначения имени файла справки:

```
Private Sub Form_Load()
   ' Назначение файла справки
   Me.HelpFile = CurrentPath() & "\RealEstate.chm"
End Sub
```

10.3.7. Назначение связей

Назначим связи между псевдонимами и индексами тем. Информация об этом хранится в Map-файле. Это простой текстовый файл с расширением h. В нашем примере — RealEstate.h. Для его создания применим стандартную программу Windows — Блокнот. Несколько строчек из файла RealEstate.h:

```
#define Form_Login 100  //Форма Login связана
                        // с файлом HTML\Part 04_2_1.htm
#define Form_Access 101  //Форма Access связана с файлом HTML\Part 04_2_2.htm
#define Form_Street 104  //Форма Street связана с файлом HTML\Part 04_3_3.htm
```

Комментарий после числа, который начинается со знака `//`, можно вообще не писать.

Осталось включить имя этого файла в проект RealEstate.hhp. Запустите HTML Help Workshop. Откройте проект. Выберите четвертую пиктограмму  первой вкладки **Project**. Появится окно **HtmlHelp API information** (рис. 10.12). Добавьте Map-файл RealEstate.h.

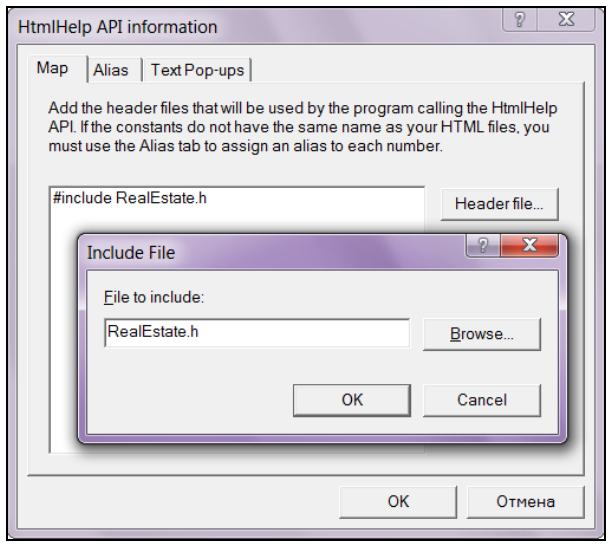


Рис. 10.12. Первая вкладка окна **HtmlHelp API information**

10.3.8. Компиляция файла справки

Откомпилируйте файл справки, щелкнув по последней пиктограмме  **Save all files and compile** (Сохранить все файлы и откомпилировать) первой вкладки **Project** (Проект) (см. рис. 10.7).

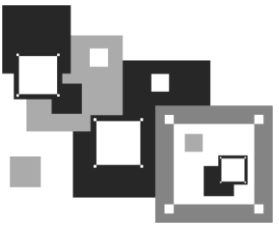
Предупреждение

HTML Help Workshop версии 1.3 (сборка 4.74.8702.0) не включает в проект файлы рисунков Microsoft Word с расширением png (Portable Network Graphics).

PNG — это формат графических файлов, поддерживаемый многими Web-обозревателями. Он обеспечивает сжатие и хранение графических изображений без потери графических данных при распаковке изображения. Формат PNG позволяет записывать сведения о прозрачности изображений, а также сведения для управления яркостью изображения на различных компьютерах. Данный формат используется для сохранения самых разных графических данных, от небольших изображений (таких как маркеры списков и объявления) до высококачественных фотографий. PNG-файлы хранятся в папках поддержки HTML-страниц.

Для решения этой проблемы откройте файл проекта RealEstate.hhp в программе Блокнот и добавьте строчки в любое место раздела `[FILES]`:

```
[FILES]
HTML\Part 02.files\*.png
HTML\Part 03_3.files\*.png
HTML\Part 04_1.files\*.png
HTML\Part 04_2.files\*.png
HTML\Part 04_2_4.files\*.png
HTML\Part 04_3_3.files\*.png
HTML\Part 05_2.files\*.png
HTML\Part 06.files\*.png
```



ГЛАВА 11

Создание пользовательской ленты

На смену меню MS Access и панелям инструментов, успешно применяемым на протяжении многих лет, в предыдущей версии продукта (MS Access 2007) пришла лента — полоса в верхней части главного окна. Осталась она и в Microsoft Access 2010, но в несколько измененном виде. При разработке профессионального приложения лента MS Access, как правило, заменяется пользовательской для того, чтобы скрыть существующие функциональные средства, которые могут повредить безопасности создаваемого приложения, и добавить новые. На этом пути разработчик может выбрать один из трех вариантов.

1. Создание меню приложения средствами языка VBA (Visual Basic for Applications).
2. Создание пользовательской ленты средствами языка XML (Extensible Markup Language).
3. Создание пользовательской ленты при помощи инструмента **Настройка ленты**, который является частью пользовательского интерфейса Microsoft Office Fluent, появившегося в MS Access 2010.

Первый вариант, используемый в версиях MS Access 97/2000/2002/2003, рассмотрен ранее *(см. разд. 7.2)*. При работе с ним в настройках MS Access 2010 гасятся все вкладки основной ленты, за исключением вкладок **Главная** и **Надстройки**. Вкладка **Главная** содержит непогашенные инструменты проверки орфографии, установки фильтра и удаления записей из форм, имеющих кнопки навигации по записям. Вкладка **Надстройки** содержит пользовательское меню (см. рис. 7.2). К недостаткам способа следует отнести наличие на экране вкладки **Файл**, в которой особую угрозу для безопасности приложения представляет пункт **Параметры**. Даже неквалифицированный пользователь может несколькими щелчками мыши вернуть назад главную ленту MS Access, **Область навигации** и получить полный доступ ко всем объектам приложения.

Второй вариант, появившийся в версии MS Access 2007, отличается наибольшей функциональностью. Позволяет создать ленту в максимальной степени удовле-

творяющую пожеланиям рядового пользователя. Главным недостатком варианта является необходимость изучения разработчиком языка XML.

Третий вариант — новшество, появившееся в последней версии продукта. Для работы с ним не требуется знание XML. Пользовательская лента создается при помощи специального инструмента и может быть выгружена в файл для переноса на компьютеры заказчика. Недостатков несколько.

- Действие ленты распространяется на все приложения, запускаемые под управлением MS Access, установленного на компьютере.
- Вкладка **Файл** выглядит как и в первом способе — пункт **Параметры** доступен.
- Отсутствует возможность погасить объекты, созданные на ленте, в соответствии с правами доступа конкретного пользователя.

В этой главе вам будет предложен прием, позволяющий убрать из меню **Файл** все пункты, представляющие угрозу безопасности создаваемого приложения, и остаться на прежних позициях создания меню средствами VBA. Значительная часть главы будет посвящена созданию пользовательской ленты с применением языка XML, а также инструмента **Настройка ленты**.

11.1. Создание таблицы *USysRibbons*

Всю информацию о пользовательских лентах MS Access 2010 хранит в системной таблице `USysRibbons`. После создания базы данных вы не найдете `USysRibbons` в разделе **Таблицы** области навигации. Ее необходимо создать или импортировать из другой базы данных. Например, Real Estate, которая находится на компакт-диске (файл Real Estate Часть II.accdb). Механизм импорта описан в *разд. 4.9*.

Перед созданием или импортированием таблицы `USysRibbons`, содержащей коды XML, необходимо вывести на экран наряду со списком таблиц, созданных разработчиком, список системных таблиц. Для этого:

1. Сделайте щелчок правой кнопкой мыши по заголовку окна **Область навигации**. Появится контекстное меню.
2. Выберите в нем пункт **Параметры навигации**. Откроется одноименное диалоговое окно. Установите флажок **Показывать системные объекты**, находящийся в разделе **Параметры отображения**.
3. Для завершения процесса нажмите кнопку **ОК**.

Примечание

Системные объекты будут отображаться не только в этой базе данных, но и во всех остальных, которые будут запущены позже на вашем компьютере под управлением MS Access 2010.

Таблица `USysRibbons` содержит три столбца (табл. 11.1). На рис. 11.1 показан вид таблицы для программного комплекса Real Estate. В таблицу занесены две ленты: первая — для создания меню программного комплекса средствами VBA, вторая — для работы только с кодом XML.

Таблица 11.1. *Структура системной таблицы* `USysRibbons`

Столбец	Тип данных
`ID` (идентификационный номер)	Счетчик (Длинное целое)
`RibbonName` (Имя ленты)	Текстовый (Размер 255)
`RibbonXml`	Поле MEMO (Размер 65536)

Столбец `ID` предназначен для хранения номера ленты. Столбец `RibbonName` содержит название ленты. Название может быть указано в параметрах текущей базы данных. В этом случае лента будет активирована при запуске приложения. Столбец `RibbonXml` предназначен для хранения кода ленты на языке XML.

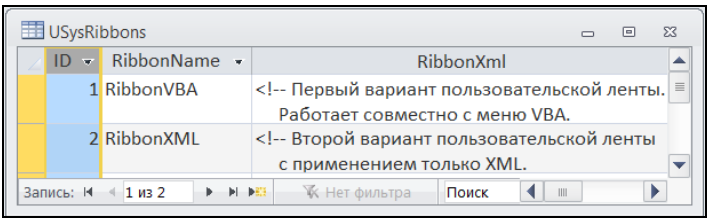


Рис. 11.1. Вид таблицы `USysRibbons` для программного комплекса Real Estate

11.2. Задание имени ленты

Пользовательская лента будет активирована при запуске приложения только в том случае, если разработчик укажет ее имя в параметрах настройки базы данных. Для этого после создания таблицы `USysRibbons` и добавления в нее хотя бы одной записи, содержащей название ленты, необходимо выполнить следующие действия.

1. Щелкните по кнопке **Файл**. Выберите пункт **Параметры**. Откроется диалоговое окно **Параметры Access**.
2. Выберите пункт **Текущая база данных**. Выполните прокрутку вниз до появления раздела **Параметры ленты и панелей инструментов**.
3. Откройте поле со списком **Имя ленты**. Оно содержит в случае таблицы `USysRibbons` программного комплекса Real Estate две ленты: `RibbonVBA` и `RibbonXML` (рис. 11.2). Выберите нужную.

4. Щелкните по кнопке **OK** для завершения процесса. Закройте и снова откройте базу данных.

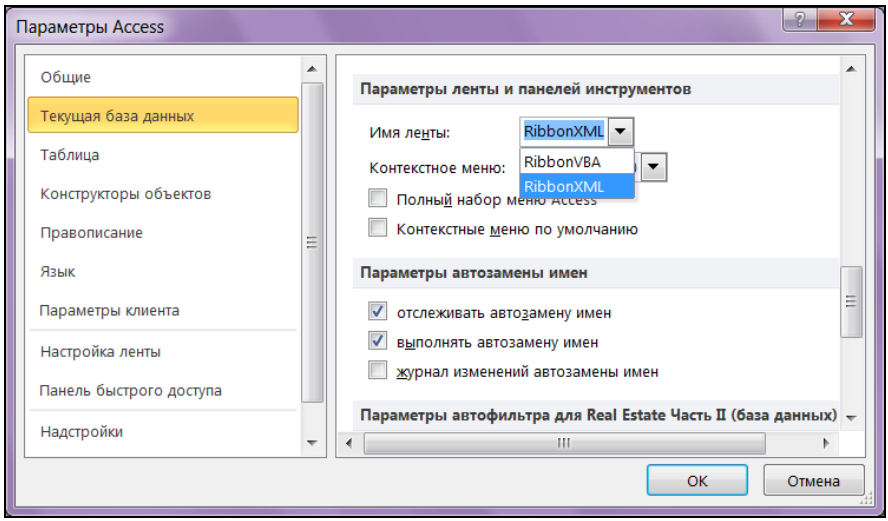


Рис. 11.2. Выбор конкретной ленты

Примечание

Запуск приложения при нажатой клавише <Shift>, если она не отключена, не активирует пользовательскую ленту. В этом случае будет запущена главная лента MS Access.

11.3. Добавление кода XML в таблицу

Заносить код ленты на языке XML можно непосредственно в MEMO-поле таблицы `USysRibbons`, но это крайне неудобный способ. Лучше всего воспользоваться специализированным редактором XML, который кроме возможности свертывания дескрипторов найдет еще и синтаксические ошибки. Логические ошибки, проблемы неправильных вызовов, отсутствующих параметров и т. д. — вне его компетенции. Ошибки этого плана исправляют только при запуске приложения на выполнение, поэтому воспользуемся стандартным приложением Windows — программой Блокнот. На рис. 11.3 в этом приложении открыт текст первой ленты программного комплекса Real Estate (файл RibbonVBA.txt с компакт-диска).

Скопировать код пользовательской ленты в таблицу `USysRibbons` не представляет никакого труда.

1. Выделите код XML целиком. Скопируйте его в буфер обмена Windows.
2. Откройте свою базу данных, а в ней таблицу `USysRibbons` в режиме таблицы.

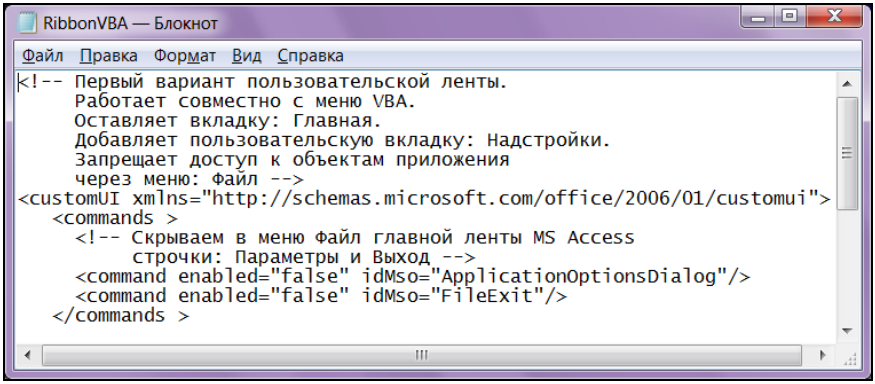


Рис. 11.3. Код XML для создания первой ленты программного комплекса

3. Вставьте содержимое буфера обмена в поле `RibbonXml` (см. рис. 11.1). Введите в эту же строку таблицы в поле `RibbonName` имя ленты.

Примечание

При возникновении ошибок в процессе отладки кода ленты выполните обратное копирование в Блокнот. Исправьте текст и после удаления содержимого MEMO-поля целиком повторите описанную ранее процедуру. Имя ленты заносить не надо.

11.4. Включение сообщений об ошибках

Перед первым запуском на выполнение созданной ленты необходимо включить отображение сообщений об ошибках пользовательского интерфейса надстроек. В случае возникновения ошибки при выключенном интерфейсе лента попросту не будет активироваться, а разработчик не получит никакого сообщения.

Для включения сообщений об ошибках необходимо выполнить следующие действия.

1. Щелкните по кнопке **Файл**. Выберите пункт **Параметры**. Откроется диалоговое окно **Параметры Access**.
2. Выберите пункт **Параметры клиента**. Выполните прокрутку вниз до появления раздела **Общие**.
3. Установите флажок **Показывать ошибки интерфейса пользователя надстроек**.
4. Щелкните по кнопке **OK** для завершения процесса.

Примечание

После завершения процесса отладки сообщения об ошибках можно не отключать, поскольку при переносе приложения на компьютеры заказчика эта опция не попадет в настройки его СУБД MS Access.

11.5. Вариант ленты с применением меню на VBA

Наряду с трудностями настройки сложной командной панели в предыдущих версиях MS Access, ей были свойственны и другие недостатки. Пользователи жаловались на проблемы, возникающие при создании собственного меню. Но шло время и большинство препятствий на этом пути разработчики преодолели. И вот теперь — все по-новому. Бал правит лента. А как быть с разработками, накопленными годами, да еще с таким трудом? Попробуем адаптировать меню, созданное средствами VBA (см. листинг 7.1), для работы в MS Access 2010, без каких-либо вопиющих брешей в системе безопасности приложения. Про пресловутую клавишу <Shift> позволю себе умолчать. Разбираться с дескрипторами языка XML и функциями обратного вызова вам конечно же не хочется.

Создайте таблицу `USysRibbons`. Занесите в нее одну запись, в MEMO-поле которой поместите текст из файла RibbonVBA.txt, имеющийся на компакт диске и представленный в листинге 11.1. Задайте имя ленты *(см. разд. 11.2)*. Оно должно совпадать со значением в поле `RibbonName`. Прочитайте комментарии, имеющиеся в листинге, и оставьте изучение языка XML до лучших времен.

Листинг 11.1. Текст ленты для работы с меню на VBA

```
<!-- Первый вариант пользовательской ленты.
     Работает совместно с меню VBA.
     Оставляет вкладку: Надстройки.
     Запрещает доступ к объектам приложения
     через меню: Файл -->
<customUI xmlns="http://schemas.microsoft.com/office/2009/07/customui">
   <commands >
     <!-- Скрываем в меню Файл строчки: Быстрая печать,
          Печать, Предварительный просмотр,
          Параметры и Выход -->
     <command enabled="false" idMso="FilePrintQuick"/>
     <command enabled="false" idMso="PrintDialogAccess"/>
     <command enabled="false" idMso="FilePrintPreview"/>
     <command enabled="false" idMso="ApplicationOptionsDialog"/>
     <command enabled="false" idMso="FileExit"/>
   </commands >
<!-- Оставляем главную ленту MS Access-->
<ribbon startFromScratch="false">
  <tabs>
    <!-- Убираем вкладку Главная -->
    <tab idMso="TabHomeAccess" visible="false"/>
    <!-- Убираем вкладку Создание -->
```

```
    <tab idMso="TabCreate" visible="false"/>
     <!-- Убираем вкладку Внешние данные -->
    <tab idMso="TabExternalData" visible="false"/>
     <!-- Убираем вкладку Работа с базами данных -->
    <tab idMso="TabDatabaseTools" visible="false"/>
  </tabs>
</ribbon>
</customUI>
```

На рис. 11.4 представлен вид программного комплекса при работе с лентой `RibbonVBA`. Для удобства все всплывающие меню совмещены.

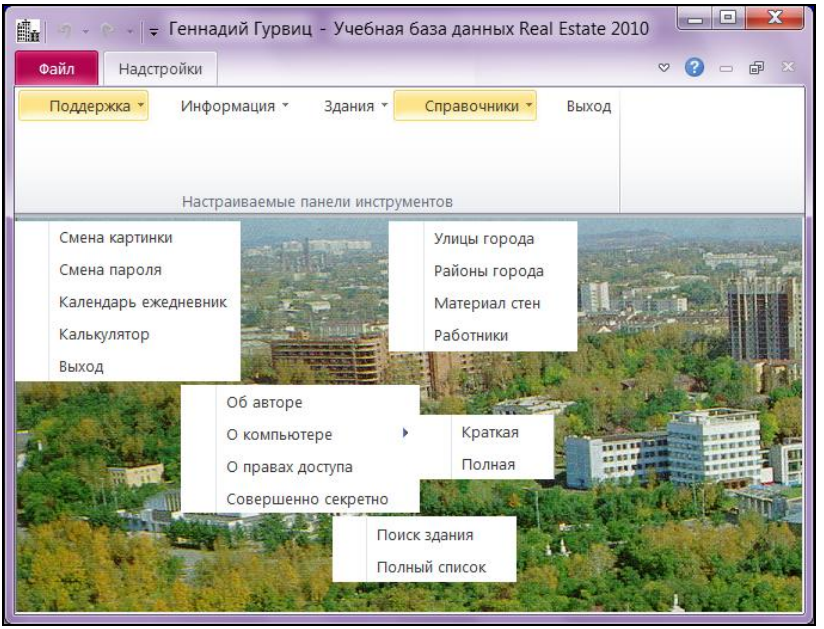


Рис. 11.4. Меню на языке VBA в работе

Лента `RibbonVBA` "разобралась" и с кнопкой **Файл**. На рис. 11.5 показано то, что осталось от ее содержимого: **Печать**, **Параметры конфиденциальности** и **Выход**, но все пункты погашены. Панель быстрого доступа также полностью погашена. Рядовой работник может только расположить ее под лентой или вернуть назад.

Если вы не хотите полностью лишить пользователя работы с принтером, а ему иногда потребуется выводить на печать отчеты из программного комплекса, то из листинга 11.1 уберите строчку:

```
<command enabled="false" idMso="PrintDialogAccess"/>
```

которая делает недоступным окно **Печать** операционной системы вашего компьютера, предназначенное для выбора принтера, числа копий и других параметров печати перед печатью.

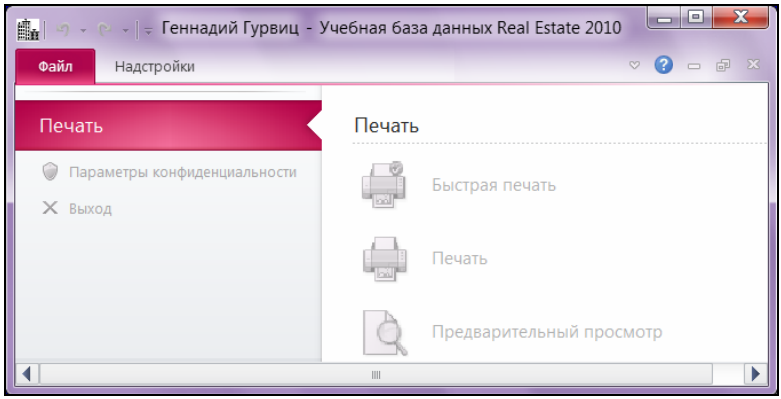


Рис. 11.5. Содержимое кнопки **Файл** после запуска на выполнение ленты `RibbonVBA`

Примечание

Если вы начали работу над собственным приложением с создания этой ленты, и меню на языке VBA еще нет, то перед его созданием подключите библиотеку Microsoft Office 14.0 Object Library. Этот процесс описан в *разд. 7.4.4*.

11.6. Вариант ленты с кодом XML

Microsoft Access 2010 позволяет не только убрать главную ленту с экрана дисплея, но и создать несколько пользовательских лент как для всего приложения, так и для отдельных форм и отчетов текущей базы данных. Все они должны быть помещены в таблицу `USysRibbons`. Назначение имени ленты для приложения описано в *разд. 11.2*. Если лента должна появляться вместе с формой или отчетом, то ее имя необходимо указать в окне свойств формы или отчета. Перейдите на вкладку **Другие**, найдите свойство **Имя ленты**, раскройте поле со списком. В нем появится список всех лент, имеющихся в таблице `USysRibbons`. Сделайте свой выбор и сохраните выполненные изменения.

Основное назначение пользовательской ленты для приложения Real Estate 2010 — заменить созданное ранее меню с применением кода VBA. Следует отметить, что возможности языка XML позволяют разработчику гораздо большее. На вкладках ленты можно разместить надписи, флажки, различные кнопки, раскрывающиеся списки, переключатели и т. д. Другими словами, все типы элементов, имеющиеся на вкладках главной ленты. Кроме этого на пользовательской ленте можно разместить и сами элементы, которые только можно найти на главной ленте.

Первый этап в создании пользовательской ленты заключается в проектировании ее внешнего вида на бумаге. Созданное ранее меню (см. рис. 11.4) послужит нам отправной точкой и позволит пропустить этот этап. Вариант ленты с применением кода на языке XML содержится в файле RibbonXML.txt на компакт-диске и приведен в листинге 11.2.

Листинг 11.2. Текст пользовательской ленты для приложения Real Estate

```
<!-- Второй вариант пользовательской ленты
     с применением только XML.
     Определение пространства имен XML.
     Указан документ, содержащий допустимые дескрипторы.-->
<customUI xmlns="http://schemas.microsoft.com/office/2009/07/customui"
                loadImage="MyICO"
                onLoad="OnLoad">
     <!-- Пространство имен xmlns в MS Office содержит
          элементы конструкции ленты.
          Атрибут loadImage указывает на процедуру: MyICO
          из глобального модуля ModuleRibbon, которая
          загружает значки для ленты.
          Атрибут onLoad указывает на процедуру: OnLoad
          из глобального модуля ModuleRibbon,
          которая передает в глобальную переменную
          объект текущей ленты.-->
 <commands >
    <!-- Скрываем в меню Файл строчки: Быстрая печать,
          Печать, Предварительный просмотр,
          Параметры и Выход -->
     <command enabled="false" idMso="FilePrintQuick"/>
     <command enabled="false" idMso="PrintDialogAccess"/>
     <command enabled="false" idMso="FilePrintPreview"/>
     <command enabled="false" idMso="ApplicationOptionsDialog"/>
     <command enabled="false" idMso="FileExit"/>
 </commands >
<!-- Убираем главную ленту MS Access -->
<ribbon startFromScratch="true">
 <tabs>
  <!-- Создание пользовательской вкладки
       Пользовательская лента будет содержать
       только одну вкладку с названием: Недвижимость -->
  <tab id="MenuRealEstate"
           label="Недвижимость">
       <!-- id — имя вкладки
           label — заголовок вкладки -->
```

```
<!-- Создание первого раздела вкладки -->
<group id="group1"
       label="Служебные">
       <!-- group1 — имя первого раздела вкладки
            label — подпись вкладки (см. внизу ленты)
             Всего разделов будет два:
             1. Служебные
             2. Работа с базой данных -->
 <!-- Первый раздел (Служебные) содержит два меню:
      1. Поддержка
      2. Информация -->
 <!-- Создание меню: Поддержка-->
 <menu id="MenuSupport" label="Поддержка"
                        size="large"
                        image="Support.gif">
          <!-- label — название меню
               size — размер элемента (large или normal)
               image — имя файла значка для меню: Поддержка -->
      <!-- Создание первого пункта меню. -->
      <button id="Button1"
              label="Смена картинки"
              onAction="XMLPicture"
              screentip="Изменение фона окна программы"
              supertip="Снижает утомляемость работника"/>
              <!-- id — имя пункта меню (кнопки)
                label — название пункта меню
                onAction — имя процедуры, которая будет
                запущена при выборе пункта меню.
                Эта процедура (XMLPicture) находится в
                глобальном модуле ModuleRibbon.
                screentip — краткая всплывающая подсказка
                supertip — подробная всплывающая подсказка-->
      <!-- Создание второго пункта меню. -->
      <button id="Button2"
              label="Смена пароля"
              onAction="XMLChPassword"/>
      <!-- Создание третьего пункта меню. -->
      <button id="Button3"
              label="Календарь ежедневник"
              onAction="XMLInformation"/>
      <!-- Создание четвертого пункта меню. -->
      <button id="Button4"
              label="Калькулятор"
              onAction="XMLInformation"/>
```

```
        <!-- Создание пятого пункта меню. -->
        <button id="Button5"
                label="Выход"
                onAction="XMLStopMenu"/>
    </menu>
    <!-- Создание меню: Информация -->
    <menu id="MenuInformation"
          label="Информация"
          size="large"
          image="Information.gif">
          <button id="Button6"
                  label="Об авторе"
                  onAction="XMLAuthor"/>
               <menu id="MenuComputer"
                     label="О компьютере">
                  <button id="Button7"
                          label="Краткая"
                          onAction="XMLInformation"/>
                  <button id="Button8"
                          label="Полная"
                          onAction="XMLInformation"/>
               </menu>
          <button id="Button9"
                  label="О правах доступа"
                  onAction="XMLEmployee"
                  getEnabled="XMLRightAccess"/>
                  <!-- Атрибут getEnabled указывает на процедуру,
                       которая устанавливает права доступа
                       к пункту меню (кнопке).
                       Эта процедура (XMLRightAccess) находится в
                       глобальном модуле ModuleRibbon-->
          <button id="Button10"
                  label="Совершенно секретно"
                  onAction="XMLAbsolutely"/>
    </menu>
  </group>
  <!-- Создание второго раздела вкладки -->
  <group id="group2" label="Работа с базой данных">
  <!-- Второй раздел вкладки содержит два меню:
          1. Здания
          2. Справочники
       Две системные кнопки:
          1. Орфография
          2. Удалить запись
       и кнопку: Выход -->
```

```
<!-- Создание меню: Здания -->
<menu id="MenuBuilding" label="Здания"
      size="large"
      image="Building.gif">
     <button id="Button11"
             label="Поиск здания"
             onAction="XMLPageBuilding"
             getEnabled="XMLSeekBuilding"/>
     <button id="Button12"
             label="Полный список"
             onAction="XMLAllBuildings"/>
</menu>
<!-- Создание меню: Справочники -->
<menu id="MenuDictionary"
         label="Справочники"
         size="large"
         image="Dictionary.gif">
      <button id="Button13"
              label="Улицы города"
              onAction="XMLDictionary"
              getEnabled="XMLStreetTown"/>
      <button id="Button14"
              label="Районы города"
              onAction="XMLDictionary"
              getEnabled="XMLDistrictTown"/>
      <button id="Button15"
              label="Материал стен"
              onAction="XMLDictionary"
              getEnabled="XMLMaterialWall"/>
      <button id="Button16"
              label="Работники"
              onAction="XMLEmployee"
              getEnabled="XMLRightAccess"/>
</menu>
<!-- Системная кнопка Орфография -->
<button idMso="SpellingAccess" visible="true"/>
<!-- Системная кнопка Удалить запись.
     Предназначена для удаления записи в форме,
     имеющей кнопки навигации по записям -->
<button idMso="RecordsDeleteRecord" visible="true"/>
<!-- Создание кнопки: Выход -->
<button id="ButtonExit"
        label="Выход"
        size="large"
        image="Exit.gif"
        onAction="XMLStopMenu"/>
```

```
    </group>
   </tab>
  </tabs>
</ribbon>
</customUI>
```

На рис. 11.6 вы видите результат работы кода XML, приведенного в листинге 11.2. Для удобства все пункты меню раскрыты и показаны одновременно. Сравните его с результатами работы пользовательского меню на языке VBA (см. рис. 11.4). Внешний вид меню практически не изменился. Удалось убрать непонятное слово **Надстройки**. Их заменила вкладка **Недвижимость**, однозначно трактующая предметную область приложения. Пользовательская лента содержит два раздела: **Служебные** и **Работа с базой данных**. Возможности VBA не позволяют этого сделать. Пользователь видит только надпись **Настраиваемые панели инструментов**, которая его просто сбивает с толку. Вид меню несколько улучшен в результате применения значков. И наконец, появились две системные кнопки: **Орфография** и **Удалить запись.**

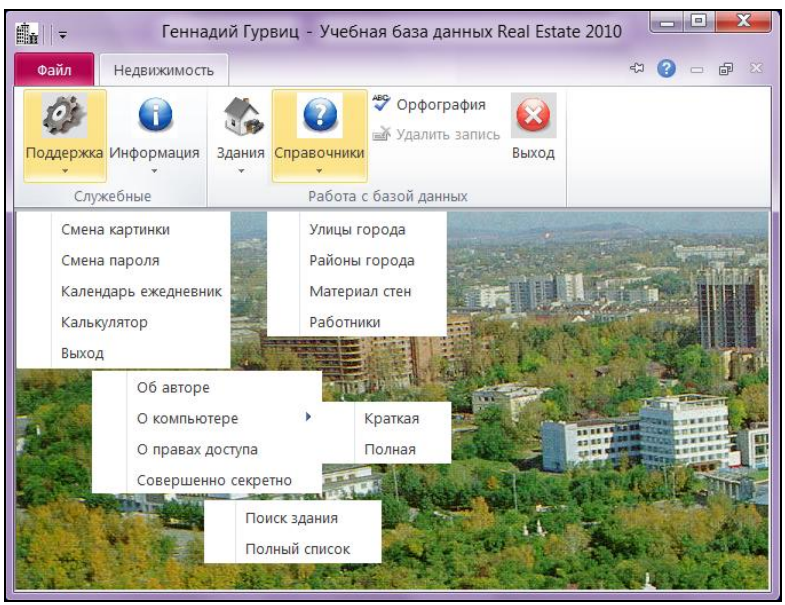


Рис. 11.6. Результат работы пользовательской ленты

Кнопка **Удалить запись** предназначена для удаления записей в формах, которые имеют кнопки навигации по записям. Кнопка становится доступной при появлении на экране формы, имеющей значение *нет* для свойства **Всплывающее окно**.

Если вы скопируете текст ленты `RibbonXML` в свою таблицу `USysRibbons` и запустите приложение на выполнение, то получите ряд сообщений об ошибках. Это свя-

зано с тем, что в коде XML используется ряд процедур обратного вызова, которые представляют собой своеобразный мостик между объектами ленты и формами MS Access или процедурами VBA. В глобальном модуле `ModuleRibbon` приложения Real Estate 2010, которое находится на компакт-диске в файле Real Estate Часть II.accdb, вы найдете все процедуры обратного вызова, используемые в тексте кода ленты. Остановимся на их работе подробнее.

Существует несколько типов процедур обратного вызова. С каждым типом связан соответствующий дескриптор языка XML. Каждый тип процедуры имеет специфическую конструкцию, которой необходимо строго придерживаться. Рассмотрим их в порядке появления в коде ленты.

```
loadImage="MyICO"
```

Процедура `MyICO` предназначена для определения местонахождения значков для пользовательской ленты. В приложении Real Estate 2010 все значки помещены в отдельную папку ICO. Текст процедуры имеет вид:

```
Public Sub MyICO(imageName As String, ByRef image)
  ' Определение местонахождения значков для пользовательской ленты
  Set image = LoadPicture(CurrentProject.Path & "\ICO\" & imageName)
End Sub
```

Поместив ее в глобальный модуль, можно забыть про то, где лента, запущенная на выполнение, будет искать значки. Просто в нужном месте укажите:

```
image="Support.gif"
```

где в кавычках задано имя файла значка.

Следующий атрибут

```
onLoad="OnLoad"
```

позволяет сохранить в глобальной переменной имя активированной ленты. Это необходимо для дальнейших манипуляций с лентами, если это потребуется в приложении.

Глобальную переменную, например `RealRibbon`, необходимо объявить в заголовке глобального модуля:

```
' Описание глобальных переменных
Public RealRibbon As IRibbonUI   ' Текущая лента
```

Текст процедуры `OnLoad` должен иметь вид:

```
Public Sub OnLoad(ribbon As IRibbonUI)
  Set RealRibbon = ribbon
End Sub
```

В приложении Real Estate 2010 переменная `RealRibbon` используется для обновления пользовательской ленты:

```
RealRibbon.Invalidate
```

Эта строчка расположена в самом конце кода, который вызывается на выполнение после нажатия кнопки **Вход** формы `Login`, когда завершена авторизация пользователя, в результате которой ему назначены права доступа к объектам программного комплекса. Для чего это сделано?

Проследим порядок запуска на выполнение нашего приложения. Двойной щелчок по значку accdb-файла вызывает к жизни СУБД MS Access, которая запускает на выполнение программный комплекс. Для Real Estate 2010 в настройках текущей базы данных в качестве стартовой формы (**Форма просмотра**) назначена форма `Start`, а в качестве пользовательской ленты (**Имя ленты**) — лента `RibbonXML`. В соответствии с выполненными настройками получим следующую цепочку.

1. Загрузка глобальных модулей приложения (`ModuleMain`, `ModuleRibbon` и т. д.), в результате которой система узнает имена глобальных переменных и тексты процедур. В частности, процедуры `Adjustment` (`ModuleMain`) и процедур обратного вызова (`ModuleRibbon`).
2. Загрузка формы `Start` с обработкой событий:
   - открытие формы;
   - загрузка формы;
   - изменение размера формы

   и т. д.
3. Загрузка пользовательской ленты `RibbonXML`, которая сделает недоступной главную ленту MS Access 2010 и выполнит свою настройку в соответствии с назначенными процедурами контроля доступа. Другими словами, сделает недоступными все свои объекты, для которых написаны процедуры `getEnabled` (см. на два десятка строк далее), т. к. авторизация пользователя еще не выполнена.
4. Загрузка формы `Login` после завершения пользователем работы формы `Start`. На этом этапе работник получит персонифицированные права доступа к объектам ленты.

Вот теперь пользовательскую ленту необходимо обновить в соответствии с полученными правами доступа. Однако такого действия с лентой MS Access не предусматривает. Поэтому указываем:

```
RealRibbon.Invalidate
```

`Invalidate` — сделать недействительной. Пользовательская лента `RibbonXML` станет недоступной. Она останется на экране, но не будет реагировать на щелчки мышью. И если в данный момент не вывести на экран другую ленту, то система "перерисует" `RibbonXML` заново в соответствии с правами доступа пользователя.

Для запуска на выполнение формы или отчета применяется атрибут `onAction`:

```
onAction="XMLPicture"
```

Текст процедуры XMLPicture имеет вид:

```
Public Sub XMLPicture(control As IRibbonControl)
  DoCmd.OpenForm "ChangePicture"
End Sub
```

В рассмотренном случае процедура обратного вызова XMLPicture запускает на выполнение форму ChangePicture, которая производит смену картинки окна MS Access. Таких процедур в приложении должно быть столько, сколько атрибутов onAction имеется в коде ленты. Приведем еще один вариант:

```
onAction="XMLStopMenu"
```

Процедура XMLStopMenu не вызывает на выполнение ни формы, ни отчета. Она выполняет выход из программного комплекса:

```
Public Sub XMLStopMenu(control As IRibbonControl)
   ' Выход из программного комплекса
   ' Завершение работы MS Access
   If MsgBox("Вы действительно хотите закрыть базу Real Estate " _
             & Chr(13) & "и выйти из MS Access?", _
             vbInformation + vbYesNo, "Выход") = vbYes Then
      Application.Quit acQuitSaveAll   ' Выходим из приложения
   End If
End Sub
```

Для контроля доступа к пунктам меню (объектам ленты) используется атрибут getEnabled:

```
getEnabled="XMLRightAccess"
```

В данном случае процедура обратного вызова XMLRightAccess назначает права доступа к пункту меню **О правах доступа**:

```
Public Sub XMLRightAccess(control As IRibbonControl, ByRef enabled)
   ' Назначение прав доступа к пункту меню "О правах доступа".
   ' RightAccess — глобальная переменная типа Boolean.
   ' Описана в глобальном модуле ModuleMain и
   ' может иметь значение False или True
   enabled = RightAccess
End Sub
```

Результат ее работы можно увидеть на рис. 11.7. Пункт меню **О правах доступа** недоступен (выделен серым цветом), т. к. глобальная переменная RightAccess получила свое значение False после завершения работы формы Login. Форма Login взяла его из поля Access05 таблицы tblUser. Значением поля Access05 распоряжается флажок chkAccess05 формы Employee (Работники предприятия), показанной на рисунке.

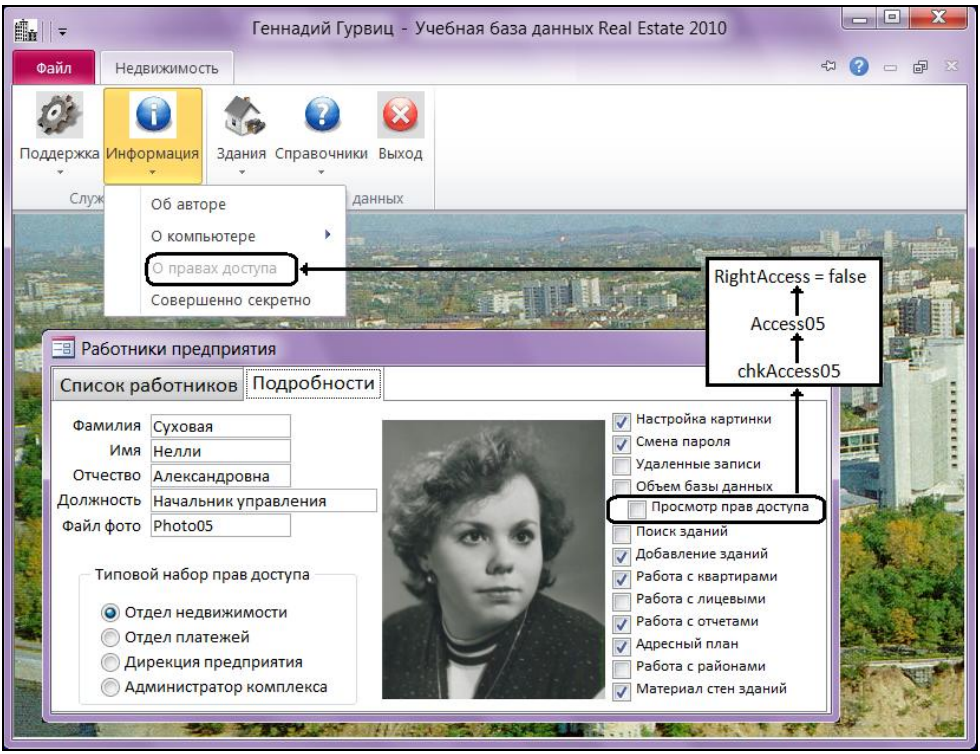


Рис. 11.7. Результат работы процедуры обратного вызова

Примечание

Язык XML — *регистрозависимый язык*, в отличие от VBA. Поэтому на данном этапе работы сохраняйте особую осторожность, манипулируя прописными и строчными буквами.

11.7. Создание ленты при помощи инструмента *Настройка ленты*

Сообщество пользователей MS Access с нетерпением ожидало появления специализированного инструмента создания и настройки ленты. И вот он в нашем арсенале (рис. 11.8). Разработчик может убрать с экрана вкладки главной ленты или часть их и создать собственные. Знание языка XML не требуется. Тем не менее, на момент написания этой главы инструмент далек от совершенства, и знание основ XML желательно, т. к. текст разработанной пользовательской ленты может быть выгружен в XML-файл, доработан и помещен в таблицу `USysRibbons`. После чего пользовательская лента будет появляться только в той базе данных,

которая содержит таблицу `USysRibbons` с созданной лентой, и в настройках — имя ленты.

Для запуска инструмента **Настройка ленты** выполните следующие действия:

1. Щелкните по кнопке **Файл**. Выберите пункт **Параметры**. Откроется диалоговое окно **Параметры Access**.
2. Выберите пункт **Настройка ленты**.
3. В правой части окна появится одноименный раздел.

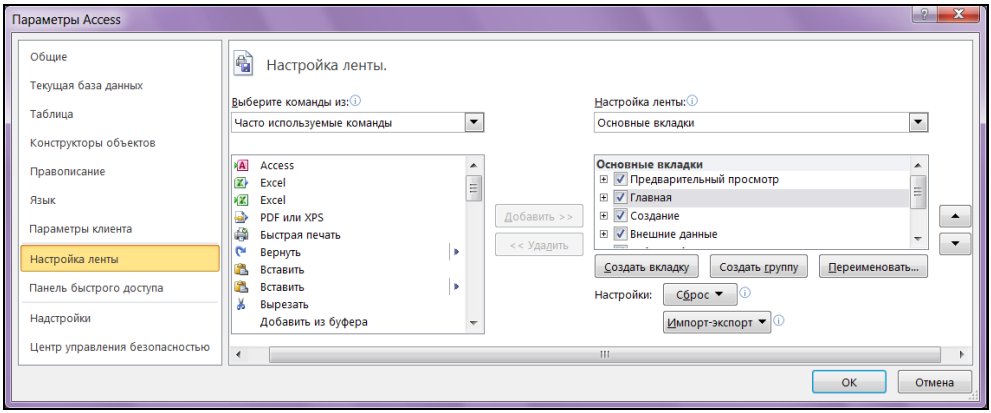
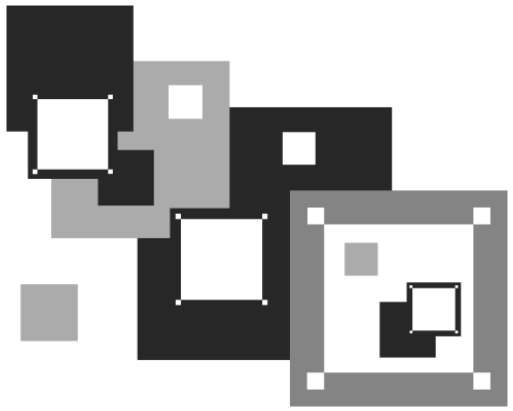


Рис. 11.8. Специализированный инструмент настройки ленты

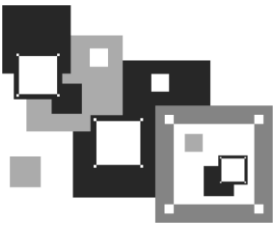
При помощи нового инструмента пользователь может создавать свои вкладки и группы команд, которые запускают на выполнение соответствующие макросы. Так же имеется возможность разместить на пользовательских вкладках любую команду, имеющуюся на главной ленте MS Access 2010.



ЧАСТЬ III

Перевод приложения в архитектуру "клиент-сервер"

В которой вы научитесь делать то, что нужно, и так, как нужно



ГЛАВА 12

Преобразование базы данных MS Access 2010 в базу MS SQL Server 2008

Ваша база данных после запуска в постоянную эксплуатацию значительно выросла в объеме и занимает несколько десятков, а то и сотен мегабайт, с ней работает полтора десятка пользователей. Замедление работы приложения Microsoft Access очень заметно, т. к. все его объекты, включая таблицы, индексные файлы, формы, запросы, отчеты, макросы и модули, хранятся в одном файле, который расположен на сервере. Значительно выросла нагрузка и на сетевое оборудование. Настала очередь следующего шага в развитии вашего приложения. Это переход на платформу "клиент-сервер" с целью оптимизации производительности, масштабируемости, безопасности, надежности, способности к восстановлению и доступности базы данных и приложения. Microsoft SQL Server с самого начала разрабатывался в архитектуре "клиент-сервер", где данные и индексы расположены на одном компьютере, доступ к которому осуществляется через сеть со многих клиентских компьютеров. Таким образом, обработка данных выполняется там, где она выполняется лучше всего — на сервере, что значительно снижает нагрузку на сеть. После выполнения запросов результаты отправляются на клиентский компьютер. Возможности SQL Server очень велики. Ваше предприятие теперь на многие годы может забыть о нехватке вычислительной мощности своей локальной вычислительной сети.

Второй и, на мой взгляд, даже более важной предпосылкой перехода в архитектуру "клиент-сервер" является обеспечение информационной безопасности приложения на более высоком уровне.

Следует отметить, что подбор пароля для доступа к базе данных MS Access 2010 (файл с расширением accdb) представляет собой серьезную задачу даже для опытного компьютерного злоумышленника. На многих хакерских сайтах имеются многочисленные утилиты, якобы решающие эту проблему. Скажу, что это не совсем так, вернее, совсем не так. Пароль длиннее семи символов практически не вскрывается при помощи персонального компьютера. В распоряжении хакеров

фактически только банальный "перебор" паролей. Но не все так хорошо! В нашем случае имеет место совсем другая ситуация, когда пользователю необходимо предоставить доступ к файлу MS Access и ограничить лишь его возможности работы с ним. Пароли, созданные штатными средствами MS Access и направленные на ограничение возможностей работы пользователя, вскрываются в считанные секунды. Не спасет вас и блокировка "замечательной" клавиши <Shift> при старте программного комплекса.

Алгоритм дальнейших действий достаточно прост.

1. Переведите базу данных в SQL Server 2008. Процесс конвертации рассматривается в этой главе.
2. Разработайте формы, обеспечивающие доступ к вашему приложению *(см. главу 8)*.
3. Клиентскую часть перед установкой на рабочие станции заказчика переведите в файл accde (без исходных текстов VBA) или ade (проект MS Access без исходных текстов).
4. Обстоятельно выполните администрирование базы в SQL Server 2008, обязательно состыковав два уровня обороны: приложения и базы данных.

Обеспечение доступа к данным теперь ложится на плечи администратора базы данных, а их обработка и доступ к результатам расчетов — на совести разработчика программного комплекса. Если в вашей организации за администрирование базы данных отвечаете не вы, а другой человек (очень часто системный администратор), то обязательно найдите с ним общий язык, в противном случае к переходу на "клиент-сервер" в данный момент не стоит приступать вообще.

12.1. Подготовка к преобразованию

В этой главе вашему вниманию будет предложен самый простой способ перевода базы данных MS Access в базу данных MS SQL Server с использованием мастера MS Access. Мастер преобразования в формат SQL Server, входящий в поставку MS Access 2010, способен преобразовать базу данных MS Access 2010 в новую или существующую базу данных Microsoft SQL версии 2008 и ниже.

Линейка продуктов Microsoft Office, в которую входит Access, всегда опережала другие разработки корпорации Microsoft. MS Access 97 не "видел" SQL Server 7.0, а MS Access 2000 ничего не знал про SQL Server 2000. Начиная с MS Office 2003, ситуация изменилась. MS Access 2003 неплохо работает с появившемся позже его SQL Server 2005, а Access 2007 с SQL Server 2008.

Перед преобразованием базы данных Access в формат SQL Server рекомендуется выполнить следующие действия:

1. Создайте резервную копию базы данных, несмотря на то, что мастер не удаляет из базы Access данные или объекты.

2. Убедитесь в наличии достаточного места на диске, где будет храниться преобразованная база данных SQL Server. Мастер работает быстрее, когда на диске много свободного пространства.
3. Создайте уникальные индексы. Для переноса в SQL Server таблица должна иметь уникальный индекс (первичный ключ), т. е. находиться во второй нормальной форме. Если этот индекс не существует, то успешно работающее приложение с базой данных MS Access при работе с базой SQL Server даст сбой. Данные в такой таблице не удастся обновить.

Перед запуском мастера преобразования в формат SQL Server не забудьте присвоить себе необходимые разрешения на доступ к базе Microsoft Access. Также необходимо иметь соответствующие разрешения на доступ к базе Microsoft SQL Server.

Для построения новой базы данных необходимо разрешение на создание базы данных, а также разрешение на доступ к системным таблицам в главной базе данных. Если SQL Server работает в смешанном режиме аутентификации, то лучше всего для этих целей знать пароль системного администратора (`sa`). Обратите внимание на то, что для работы мастера не требуется использование ODBC для подключения к Microsoft SQL Server. Его очередь может наступить позже, когда переведенная в SQL Server база данных по вашему желанию начнет работать с другим клиентом (например, Visual FoxPro или Delphi). Напомню, что ODBC (Open Database Connectivity, соединение по открытым базам) — это технология, которая позволяет приложениям обмениваться данными, несмотря на различия в применяемых системах управления базами данных. Если вы останетесь на позициях MS Access — MS SQL Server, то ODBC совсем не понадобится.

Для запуска мастера преобразования сделайте следующее:

1. Откройте базу данных Microsoft Access 2010.
2. Перейдите на четвертую вкладку главной ленты — **Работа с базами данных**.
3. Выберите в разделе **Переместить данные** пункт **SQL Server**.
4. Появится стартовое окно мастера преобразования.

12.1.1. Создание базы данных

На первом шаге мастер предлагает воспользоваться существующей базой данных SQL Server или создать новую. Если после генерации SQL Server вы не создавали никаких баз данных на этом сервере, то установите переключатель в положение **создать базу данных**.

12.1.2. Сбор сведений

Второе окно (рис. 12.1) предназначено для сбора сведений об SQL Server, порядке соединения с ним и выбора названия базы данных, которая будет создана на

нем. Microsoft SQL Server 2008 Developer Edition установлен на персональном компьютере, имеющем имя `Master`, менять его не следует.

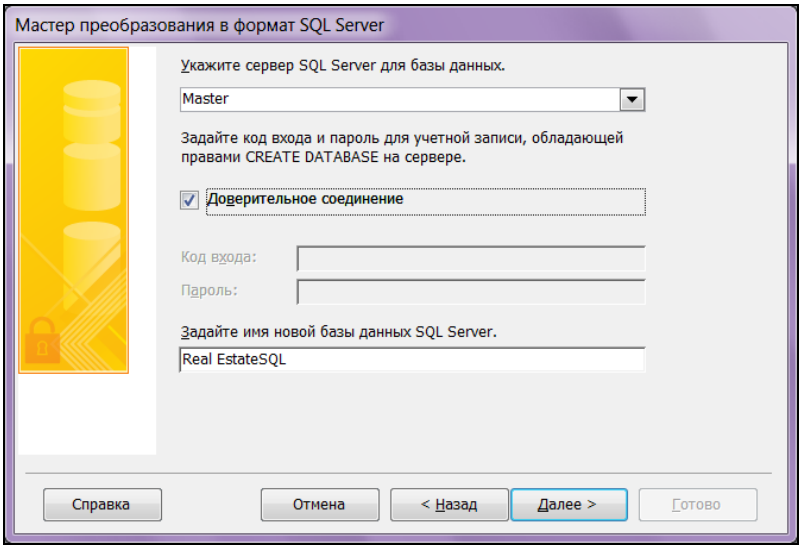


Рис. 12.1. Выбор экземпляра MS SQL Server для базы данных

Если же ваш компьютер включен в локальную сеть, в которой имеются другие серверы, и перед вами стоит задача выполнить перенос базы MS Access 2010 на другой SQL Server — найдите его имя, раскрыв при помощи мыши поле со списком, расположенное в верхней части окна. Для входа на наш SQL Server можно воспользоваться учетной записью его администратора `sa`, а можно использовать доверительные отношения и попасть на SQL Server под именем администратора операционной системы своего компьютера. Для этого поставьте флажок **Доверительное соединение**. В этом случае поля **Код входа** и **Пароль** станут недоступными. Этот вход мы обеспечили себе при генерации SQL Server 2008. Имя новой базы данных мастер преобразования сформирует сам, добавив окончание "SQL" к имени базы данных MS Access 2010 — Real Estate Часть II. Сформированное имя (Real Estate ЧастьIISQL) заменим более понятным: Real EstateSQL и перейдем следующему шагу, нажав кнопку **Далее**.

12.1.3. Выбор таблиц

Некоторое замедление в работе перед третьим шагом связано с проверкой заданного имени базы данных. Если база с таким именем уже существует, то система изменит его, добавив цифру 1 в самый конец.

Мастер преобразования предлагает выбрать таблицы базы данных Microsoft Access, которые будут помещены в базу данных Real EstateSQL на сервере.

В данном случае необходимо выбрать все таблицы. Для этого сделайте щелчок мышью по кнопке [>>]. Все таблицы появятся в правой части окна, а кнопка **Далее** станет доступной.

12.1.4. Выбор объектов

Переходим к четвертому шагу мастера преобразования (рис. 12.2). Реляционная база данных включает в себя не только таблицы, которые могут содержать условия проверки корректности данных. В ней хранятся индексы, связи между таблицами, триггеры, хранимые процедуры и т. д.

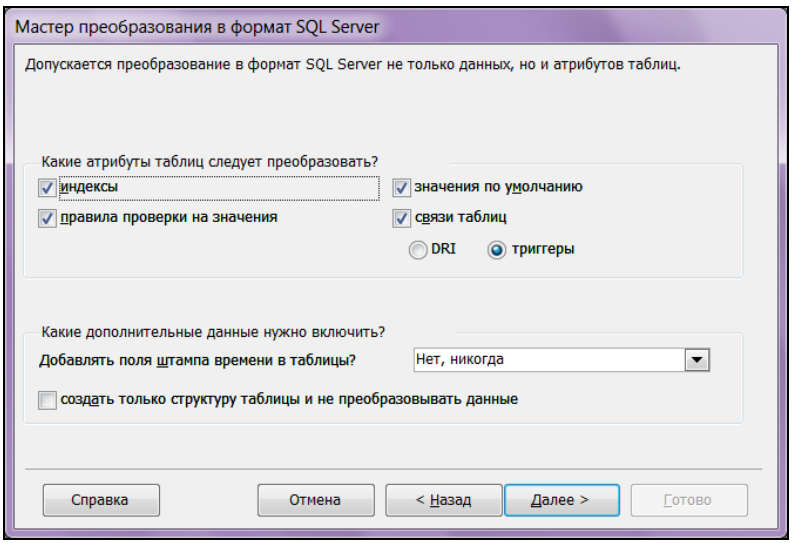


Рис. 12.2. Четвертый шаг преобразования

На очередном шаге нам предстоит определиться с объектами, которые следует конвертировать в базу SQL Server.

- **Индексы.** Если установлен флажок [✓] **индексы**, мастер преобразования в формат SQL Server преобразует все индексы. Первичные ключи Microsoft Access превратятся в некластеризованные, уникальные индексы Microsoft SQL Server и будут отмечены как первичные ключи SQL Server. Если выбрано связывание преобразуемой таблицы SQL Server с базой данных Access, то мастер преобразования добавит к имени индекса префикс "aaaaa". Microsoft Access выбирает в качестве первичного ключа первый индекс по алфавиту из списка доступных индексов, а префикс "aaaaa" гарантирует выбор нужного индекса. Имена всех остальных индексов останутся без изменений, кроме имен, в которых недопустимые символы заменяются символами "_". Уникальные и неуникальные индексы Microsoft Access становятся уникальными и неуникальными индексами SQL Server.

Предупреждение

Мастер преобразования в формат SQL Server может преобразовать существующий уникальный индекс, но не может создать индекс, если тот не существует.

- **Правила проверки на значения.** Если установлен этот флажок, мастер преобразует все условия на значения полей в ограничения CHECK. Условия на значение записей, в большинстве случаев, не конвертируются.
- **Значения по умолчанию.** Если установлен флажок **значения по умолчанию**, мастер преобразует все значения по умолчанию как значения ANSI (American National Standards Institute), а не как стандартные объекты, присоединенные к соответствующему полю SQL Server. Значения по умолчанию в SQL Server, в отличие от значений по умолчанию в MS Access, независимы от какого-либо конкретного поля или таблицы.
- **Связи таблиц.** Все моменты, связанные с установкой этого флажка, будут рассмотрены подробно несколько позднее.
- **Добавление в таблицы полей штампа времени.** Microsoft SQL Server использует поля timestamp для обозначения измененных записей (без указания времени изменения) путем создания поля с уникальными значениями и последующего обновления этого поля при каждом обновлении записи. Для связанной таблицы Access использует значение поля timestamp, чтобы перед обновлением поля определить, было ли оно изменено. Как правило, это поле обеспечивает наилучшее быстродействие и надежность. Существуют три варианта выбора.
  - При отсутствии полей timestamp SQL Server должен проверить все поля записи, чтобы определить, была ли она изменена, что снижает быстродействие.
  - Если выбрано значение по умолчанию **Да, определяется мастером**, мастер преобразования в формат SQL Server создает новые поля с типом данных timestamp в таблицах SQL Server, преобразованных из таблиц Microsoft Access, содержащих поля с типами данных "с плавающей точкой (4 байта или 8 байт)", поле MEMO или поле объекта OLE.
  - Третий вариант — выбор режима создания мастером полей timestamp для всех преобразуемых таблиц, независимо от типов данных полей, которые в них содержатся. Для этого следует выбрать вариант **Да, всегда**. Это повысит быстродействие преобразованных таблиц Access, которые могут не содержать полей MEMO, полей объектов OLE или полей с числами с плавающей точкой, но содержат поля других типов.
- Флажок **создать только структуру таблицы и не преобразовывать данные** используется в редких случаях. Например, вы недостаточно владеете графическими инструментами MS SQL Server и языком Transact-SQL, а новая база MS SQL Server должна быть готова немедленно. Создайте ее в MS Access и запустите мастер преобразования. Он отлично справится со своей работой!

Для преобразования связей (флажок **связи таблиц**) между таблицами и обеспечения ссылочной целостности данных можно использовать либо триггеры обновления, вставки и удаления, либо DRI (Declarative Referential Integrity, декларативная целостность данных). DRI действует так же, как целостность данных Microsoft Access, определяя ограничения по первичному ключу для базовых таблиц (сторона "один" отношения "один-ко-многим") и ограничения по внешнему ключу для внешних таблиц (сторона "многие" отношения "один-ко-многим").

Использование переключателя **DRI** предпочтительнее для начинающего свою работу с MS SQL Server. В этом случае можно вообще ничего не знать про триггеры. Рассмотрим процесс конвертации на примере связи между таблицами `tblBuilding` и `tblDistrict`. На рис. 12.3 приведены параметры этой связи, установленной в MS Access. Щелкните правой кнопкой мыши по линии, соединяющей таблицы, и выберите в контекстном меню пункт **Изменить связь**. Появится окно **Изменение связей**. Оно сообщает, что:

- между таблицами обеспечена ссылочная целостность данных;
- изменение номера района в таблице `tblDistrict` повлечет за собой соответствующее изменение номеров в таблице `tblBuilding`;
- каскадное удаление связанных записей в таблицах `tblDistrict` и `tblBuilding` не предусмотрено.

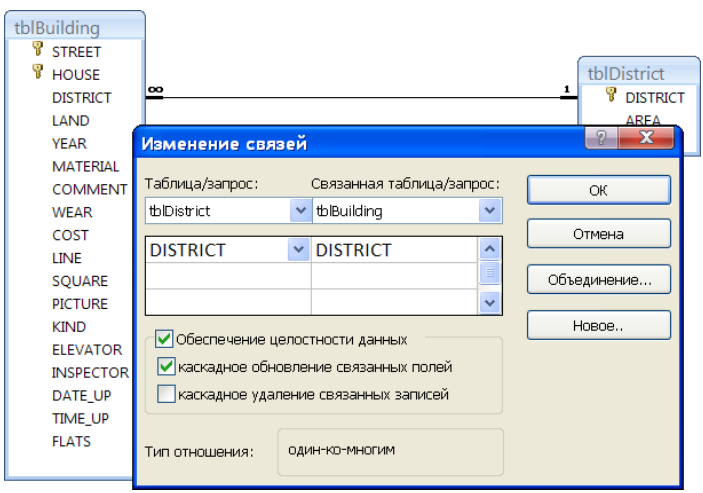


Рис. 12.3. Параметры связи, установленной между таблицами `tblDistrict` и `tblBuilding` в MS Access 2010

На рис. 12.4 представлены результаты работы мастера преобразования связи между таблицами `tblBuilding` и `tblDistrict` с использованием DRI. В MS SQL Server все несколько сложнее, чем в MS Access. Отобразить на экране дисплея сразу четыре окна, представленных на этом рисунке, вам не удастся. Это сбор-

ный рисунок, содержащий диаграмму базы данных, внешние ключи таблицы `tblBuilding`, поля связи между таблицами и окно сообщения об ошибке, которое появится на рабочей станции клиента (MS Access) при попытке удаления района в таблице `tblDistrict`.

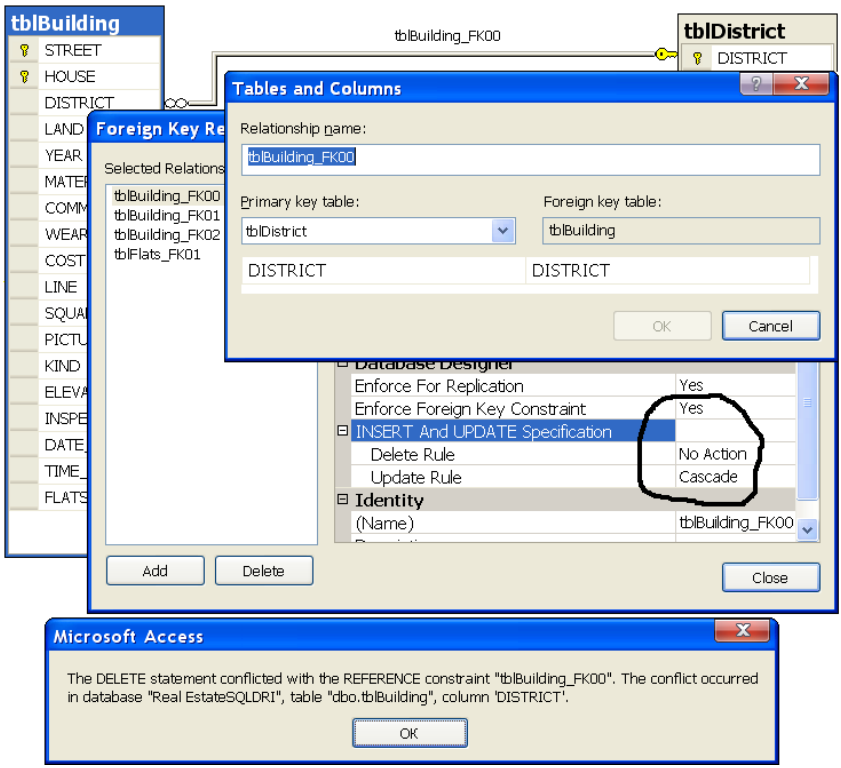


Рис. 12.4. Параметры связи между таблицами `tblDistrict` и `tblBuilding` в MS SQL Server 2008, преобразованной мастером с использованием DRI

Из рисунка видно:

- между таблицами обеспечена ссылочная целостность данных (введены ограничения внешнего ключа **Enforce Foreign Key Constraint — Yes**);
- изменение номера района в таблице `tblDistrict` повлечет за собой соответствующее изменение номеров в таблице `tblBuilding` (**Update Rule — Cascade**);
- каскадное удаление связанных записей в таблицах `tblDistrict` и `tblBuilding` не предусмотрено (**Delete Rule — No Action**).

Мастер преобразования прекрасно справился со своей задачей. Замечание только одно: в случае некорректных действий клиент получит на экране своего дисплея малопонятное сообщение на английском языке.

Использование переключателя **триггеры** (см. рис. 12.2). Если в связях между таблицами Microsoft Access определены каскадные обновления или удаления, и требуется сохранить эти возможности в преобразованных таблицах, установите флажок **связи таблиц** и переключатель **триггеры**, чтобы преобразовать все каскадные обновления или удаления как триггеры для поддержания целостности данных. Связь между таблицами не обязательно должна соответствовать одному триггеру. Каждое отношение может стать частью нескольких триггеров, а каждый триггер может содержать программу для эмуляции функциональности нескольких условий целостности данных. Триггеры вставки используются в дочерних таблицах, а триггеры удаления — в родительских таблицах.

На рис. 12.5 показано, что между таблицами `tblBuilding` и `tblDistrict` не обеспечена ссылочная целостность данных: **Enforce Foreign Key Constraint — No** (не введены ограничения внешнего ключа). Ссылочную целостность и каскадные обновления и добавления записей обеспечивают триггеры. Мастер преобразования создаст три триггера.

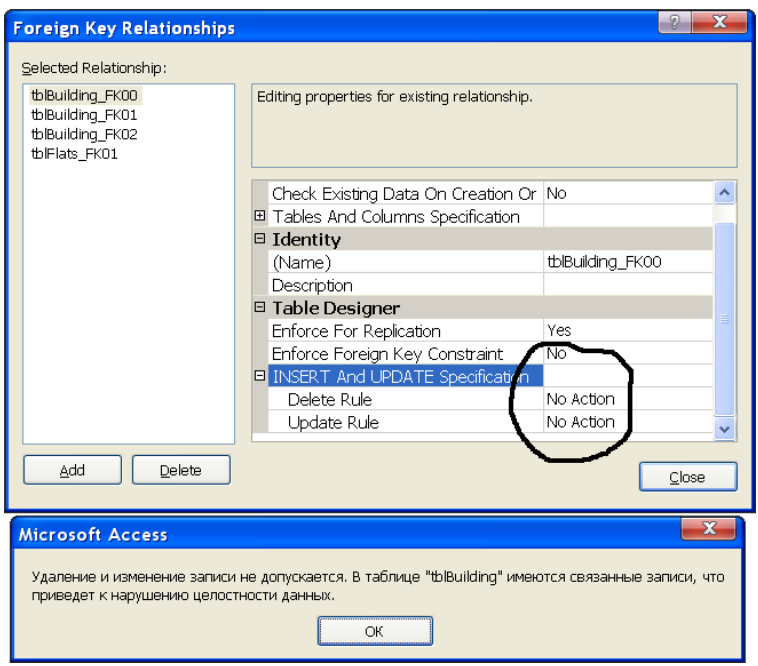


Рис. 12.5. Параметры связи между таблицами `tblDistrict` и `tblBuilding` в MS SQL Server 2008, преобразованной мастером с использованием триггеров

В листинге 12.1 приведен текст триггера, написанного на Transact-SQL и обеспечивающего запрет удаления района из таблицы районов `tblDistrict`, если в таблице зданий `tblBuilding` имеется хотя бы одно здание, расположенное в этом районе.

Листинг 12.1. Триггер на удаление (таблица tblDistrict)

```
ALTER TRIGGER T_tblDistrict_DTrig
ON tblDistrict FOR DELETE AS
SET NOCOUNT ON
/* * ЗАПРЕТ УДАЛЕНИЯ ПРИ ЗАВИСИМЫХ ЗАПИСЯХ В 'tblBuilding' */
IF (SELECT COUNT(*) FROM deleted, tblBuilding
  WHERE (deleted.DISTRICT = tblBuilding.DISTRICT)) > 0
    BEGIN
        RAISERROR 44445 'Удаление и изменение записи
        не допускаются. В таблице "tblBuilding" имеются
        связанные записи, что приведет к нарушению целостности
        данных.'
        ROLLBACK TRANSACTION
    END
```

Листинг 12.2 содержит код, обеспечивающий автоматическое изменение номера района в таблице зданий `tblBuilding`, после соответствующего изменения пользователем номера района в таблице `tblDistrict`.

Листинг 12.2. Триггер на изменение (таблица tblDistrict)

```
ALTER TRIGGER T_tblDistrict_UTrig
ON tblDistrict FOR UPDATE AS
SET NOCOUNT ON
/* * КАСКАДНЫЕ ОБНОВЛЕНИЯ В 'tblBuilding' */
IF UPDATE(DISTRICT)
    BEGIN
       UPDATE tblBuilding
       SET tblBuilding.DISTRICT = inserted.DISTRICT
       FROM tblBuilding, deleted, inserted
       WHERE deleted.DISTRICT = tblBuilding.DISTRICT
    END
```

Листинг 12.3 содержит код, обеспечивающий запрет на добавление здания в таблицу зданий `tblBuilding`, если в записи указана ссылка на несуществующий район в таблице `tblDistrict`.

Листинг 12.3. Триггер на добавление (таблица tblBuilding)

```
ALTER TRIGGER T_tblBuilding_ITrig
ON tblBuilding FOR INSERT AS
SET NOCOUNT ON
/* * ЗАПРЕТ ВСТАВКИ БЕЗ СОВПАДАЮЩЕГО КЛЮЧА В 'tblDistrict' */
```

```
IF (SELECT COUNT(*) FROM inserted) !=
   (SELECT COUNT(*) FROM tblDistrict,
    inserted WHERE (tblDistrict.DISTRICT = inserted.DISTRICT))
    BEGIN
        RAISERROR 44447 'Добавление и изменение записи
        не допускаются. Правила целостности данных
        требуют наличия связанной записи
        в таблице "tblDistrict".'
        ROLLBACK TRANSACTION
    END
```

При работе мастера преобразования с использованием триггеров сообщения о некорректных действиях пользователя с таблицами выводятся на русском языке и даже без вмешательства разработчика вполне понятны, хотя ничто не мешает вам привести их в полное соответствие с разработанным приложением. Например, при удалении района в таблице районов может появиться сообщение, показанное на рис. 12.6. Для этого замените:

```
RAISERROR 44445 'Удаление и изменение записи
не допускаются. В таблице "tblBuilding" имеются
связанные записи, что приведет к нарушению целостности
данных.'
```

текстом следующего содержания:

```
RAISERROR 44445 'Вы не можете удалить район,
т. к. в нем имеются здания.'
```

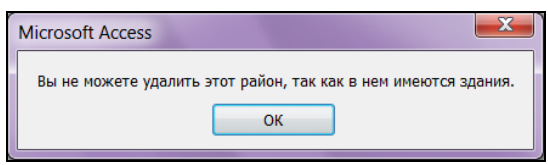


Рис. 12.6. Вид откорректированного сообщения

В любом случае преобразованную базу данных можно удалить целиком с помощью SQL Server Management Studio и начать процесс преобразования с самого начала. Если требуется повторно преобразовать только некоторые таблицы, необходимо сначала удалить эти таблицы и все другие, с которыми они связаны, начиная с таблицы на стороне "многие" в отношении "один-ко-многим". Первой среди удаляемых должна быть таблица, на первичный ключ которой нет ссылок.

Примечание

Мастер преобразования в формат SQL Server 2008 может установить отношения только между связанными таблицами, преобразованными в одно и то же время, но не может установить отношения между "старыми" и "новыми" таблицами.

12.2. Выбор способа преобразования

Одно из самых сложных — пятое окно (рис. 12.7). Сразу очень трудно догадаться о том, что же конкретно собирается сделать мастер в каждом из трех предложенных вариантах, хотя на первый взгляд все интуитивно понятно. Окно предназначено для выбора способа преобразования в формат SQL Server 2008.

Перечислю их в том порядке, в котором они представлены в окне:

- **создать новое приложение Access "клиент-сервер"** (связь клиента и сервера через OLE DB);
- **связать таблицы SQL Server с существующим приложением** — преобразовать все объекты базы данных MS Access для работы с базой данных SQL Server, что также позволит создать приложение типа "клиент-сервер" (связь через ODBC);
- **не изменять приложение** — преобразовать только данные из формата базы данных MS Access 2010 в формат базы данных SQL Server. Клиентскую часть в этом случае придется писать заново, используя другие, более мощные, программные средства (платформу .NET). На мой взгляд, это самый правильный выбор для дальнейшего развития приложения, которое переросло само себя. Да и пользователь, выросший на этом приложении, наверняка потребует его коренной переделки.

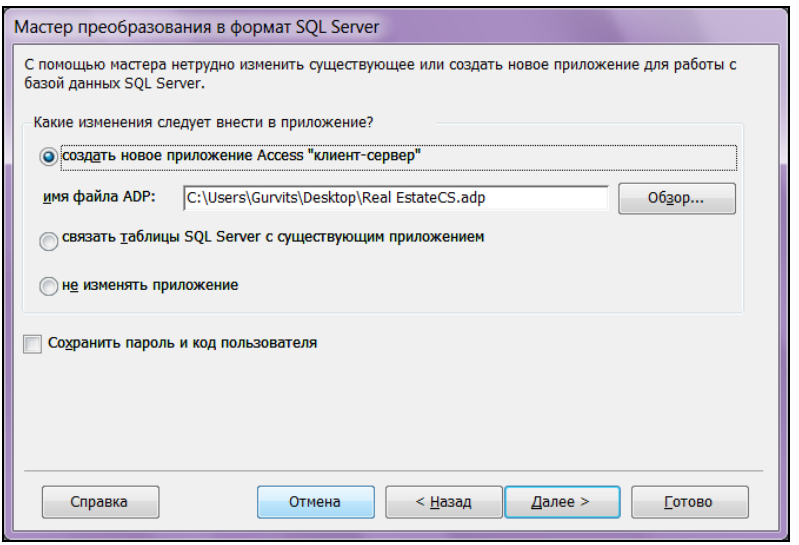


Рис. 12.7. Пятое окно мастера преобразования в формат SQL Server

Мастер преобразования получил всю необходимую информацию для преобразования базы данных. Сделайте щелчок по кнопке **Готово** для запуска процесса.

Рассмотрим все три способа преобразования более подробно, но в обратном порядке.

12.2.1. Создание базы SQL Server без изменения приложения

Это третий случай преобразования. Мастер создаст базу данных SQL Server, но не станет создавать проект Microsoft Access или модернизировать существующее приложение. Для работы с этим методом выберите переключатель **не изменять приложение** (см. рис. 12.7). В вашем распоряжении появится база данных под управлением MS SQL Server 2008, а также время на изучение новой платформы. Старое приложение продолжит функционирование в прежнем режиме. Этот вариант преобразования выходит за рамки данной книги.

12.2.2. Связь Access-приложения с базой данных SQL Server

В этом случае мастер преобразования создаст базу данных SQL Server и свяжет ее таблицы с текущей базой Access, которая будет изменена. Этот вариант уступает первому (**создать новое приложение Access "клиент-сервер"**) по скорости работы и дает несколько большую нагрузку на сетевые коммуникации, т. к. запросы Microsoft Access не превращаются в хранимые процедуры и представления Microsoft SQL-сервера, что ведет к лишним затратам времени на пересылку и конвертацию запросов к серверу во время работы приложения с диалекта Access SQL на вариант языка запросов SQL Server, который получил название Transact-SQL. Однако при использовании этого варианта можно отметить и определенные преимущества. Доводка accdb-файла до рабочего состояния в этом случае минимальна, а рассматриваемая нами база Real Estate 2010 вообще не требует никакого вмешательства со стороны разработчика.

Одним из первых стандартов доступа к базам данных был ODBC (Open Database Connectivity). Именно он и применяется при работе со связанными таблицами. После подключения к источнику данных ODBC MS Access сохраняет строку подключения в свойстве **Описание** каждой связанной таблицы. При создании строки подключения мастер не использует на компьютере клиента постоянный источник данных. Он указывает для каждой таблицы файл драйвера и имя сервера. А что делать нам при тиражировании клиента по рабочим станциям? Ведь при переносе клиентской части на другую машину связи с таблицами будут утеряны. Не прописывать же строку подключения для каждой таблицы заново! Выход прост — создание постоянного источника ODBC на каждой рабочей станции. Этот источник создается один раз и полностью автоматизирует процесс подключения к базе SQL Server. Создание подключения с использованием ODBC здесь не рассматривается из-за устаревшей технологии.

12.2.3. Создание нового приложения "клиент-сервер"

При выборе переключателя **создать новое приложение Access "клиент-сервер"** мастер преобразования в формат SQL Server создает новый проект Microsoft Access (рис. 12.8). Пользователю выводится приглашение указать его имя (по умолчанию используется имя текущей базы данных MS Access). Мастер добавляет суффикс "CS", а затем сохраняет его в той же папке, где расположена существующая база данных Access. Проект Microsoft Access 2010 соединяется с базой данных Microsoft SQL Server 2008 с помощью архитектуры компонентов OLE DB. Проект не содержит никаких данных или объектов определения данных: таблиц, представлений, триггеров, хранимых процедур и т. д. Эти объекты сохраняются в базе данных на SQL-сервере. Проект включает в себя только формы, отчеты, макросы и модули. Объекты базы данных из формата базы данных MS Access 2010 в формат проекта Access 2002—2010 преобразуются по правилам, описанным в следующих разделах.

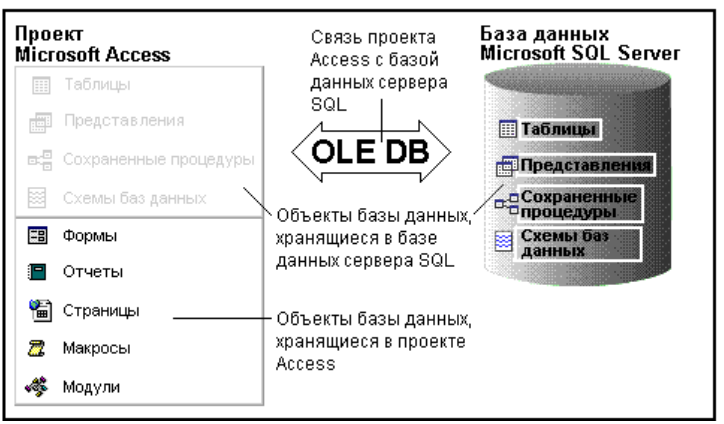


Рис. 12.8. Понятие проекта Microsoft Access

Запросы

Мастер преобразования в формат SQL Server изменяет запросы, превращая их в представления или хранимые процедуры, преобразуя синтаксис языка SQL Access в синтаксис языка Transact-SQL для SQL Server. Запросы на выборку преобразуются в представления. Запросы с сортировкой преобразуются в комбинацию представлений и хранимых процедур, что делает возможным вложение и сортировку (представления могут быть вложенными, но не могут содержать предложения `ORDER BY`, хранимые процедуры могут содержать предложения `ORDER BY`, но не могут быть вложенными). Запросы с параметрами, запросы, зависящие

от запросов с параметрами, а также запросы на изменение преобразуются в хранимые процедуры. В некоторых случаях может потребоваться преобразование запросов вручную, которые не преобразуются мастером. Запросы к серверу SQL, управляющие запросы и запросы на объединение не преобразуются.

Формы, отчеты и элементы управления

Мастер преобразования в формат SQL Server изменяет свойства **Источник записей**, **Данные** и **Источник строк** путем изменения имен таблиц, запросов, инструкций SQL или полей на эквивалентные имена таблиц, представлений, хранимых процедур или полей таблиц SQL Server.

Макросы и модули

В эти объекты базы данных мастер преобразования не вносит никаких изменений. После завершения работы мастера необходимо вручную преобразовать в модулях программы, использующие наборы записей, из формата объектов доступа к данным (DAO) в формат объектов данных ActiveX (ADO), а также проверить коды структур всех таблиц и запросов (мастер преобразования в формат SQL Server не преобразует инструкции SQL языка описания данных (Data Definition Language)).

12.3. Отчет мастера преобразования в формат SQL Server

Последним окном при работе мастера преобразования в формат MS SQL Server во всех трех случаях будет окно отчета (рис. 12.9).

Оно сообщает об ошибках, возникших в процессе конвертации. Полное описание этого процесса Microsoft Access 2010 может поместить в специальный файл. Побеспокойтесь о сохранении этой информации.

Сделайте щелчок правой кнопкой мыши в любом месте этого окна. Появится контекстно-зависимое меню. Выберите пункт **Экспорт**. На ваш выбор будет предложено несколько форматов от текстового до HTML. Отчет по базе Real Estate 2010 содержит двадцать страниц. К нему мы вернемся в *главе 14*.

12.4. ODBC, OLE DB, DAO, ADO, ADO.NET и просто .NET

В предыдущем разделе при переносе базы данных MS Access 2010 в SQL Server 2008 упоминались термины: ODBC, OLE DB и .NET. Остановимся на них подробнее.

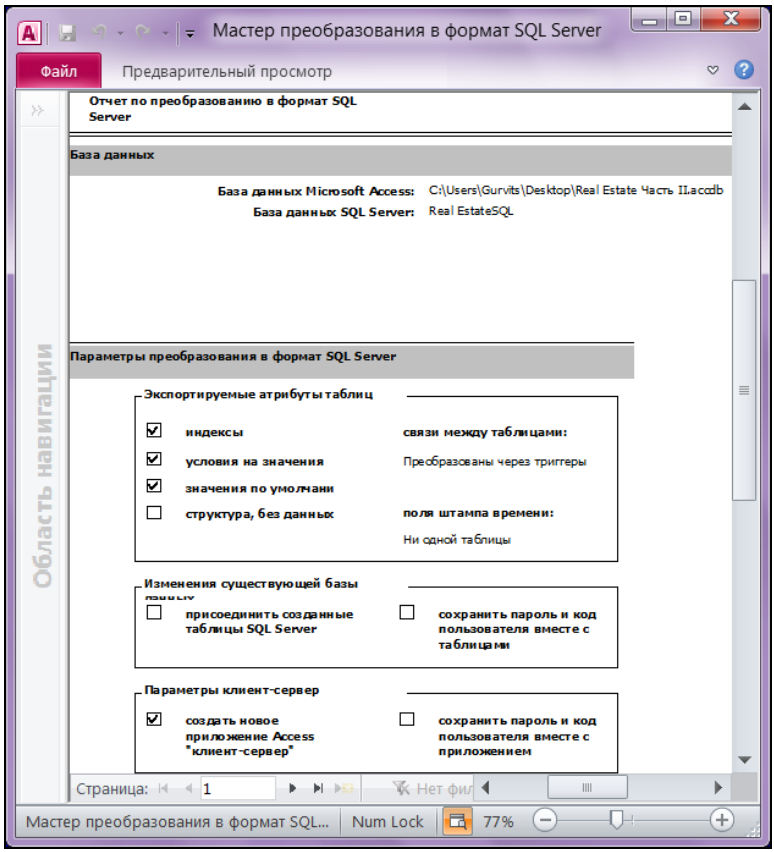


Рис. 12.9. Отчет о преобразовании в формат SQL Server

Для доступа к данным сервера со стороны клиента применяются три технологии.

- *Низкоуровневая технология.* Появилась первой. Использует способ включения в библиотеки клиентов специальных функций для работы с каждым конкретным сервером. Сколько серверов — столько наборов средств. Переход на другой сервер всегда требует радикального изменения приложения.
- *ODBC.* Появление этой технологии резко облегчило жизнь программисту. Он получил возможность использовать для доступа к данным стандартный интерфейс прикладного программирования API (Application Programming Interface) — набор подпрограмм для выполнения служб нижнего уровня операционной системы. ODBC взяла на себя перевод вызовов API в команды конкретного сервера. Предназначена для работы только с реляционными данными. В настоящий момент считается устаревшей технологией.
- *OLE DB.* В 2001—2003 годах пришла на смену ODBC. Сейчас она поддерживается подавляющим большинством производителей реляционных баз дан-

ных. MS Access 2010 по умолчанию использует OLE DB как свой внутренний способ доступа к данным. Главный недостаток ODBC заключается в том, что для выполнения простых задач программист должен вызвать значительное количество сложных функций. И, несмотря на то, что Microsoft и другие корпорации предлагают разработчикам высокоуровневые надстройки (а именно с ними мы и имеем дело), по своей сути ODBC — очень сложная технология. OLE DB основана не на API, а на COM (Component Object Model, модель составных компонентов). Это значительно облегчает разработку не только пользовательских приложений, но и средств доступа к данным OLE DB, называемых поставщиками или провайдерами. В их роли могут выступать любые COM-компоненты.

В 2002 году Microsoft представила платформу .NET. Обратите внимание: .NET — это не технология доступа к данным сервера. Это платформа, предназначенная для создания как обычных программ, так и Web-приложений. Программы Microsoft .NET работают в среде времени выполнения CLR (Common Language Runtime). Одной из основных идей .NET является совместимость компонентов приложения, написанных на разных языках. Например, компонент, написанный на C++, может обратиться к методу класса из библиотеки, написанной на Visual Basic или Delphi.

Наряду с этими технологиями существуют модели доступа к данным: DAO, ADO, ADO.NET и др. Так как VBA и языки, входящие в состав MS Visual Studio .NET, не являются языками баз данных, то для доступа к данным они вынуждены использовать предназначенные для этого объекты одной из моделей данных (DAO, ADO и ADO.NET) или сразу нескольких. Для работы языков баз данных (например, MS Visual FoxPro) эти объектные модели не требуются. Вот расшифровка названий этих моделей.

- Microsoft DAO (Data Access Objects) создана для обеспечения работы с базами данных приложения, поддерживающего VBA. Работает с драйвером Jet (MS Access) и с ODBC.
- Microsoft ADO (ActiveX Data Objects) пришла на смену DAO и работает с технологией доступа OLE DB.
- Microsoft ADO.NET (ActiveX Data Objects для платформы .NET) представляет собой модернизацию ADO. Часть ее объектов базируется на технологии OLE DB, а часть использует для доступа к данным технологии низкоуровневого доступа. ADO.NET — одна из библиотек компонентов платформы .NET.

На рис. 12.10 показана обобщенная схема доступа к данным MS SQL-сервера из приложений, написанных на различных языках программирования.

Работу базы данных MS Access 2010 (файл accdb) поддерживает драйвер Ace, пришедший на смену знаменитому Jet (рис. 12.11). MS Access 2010 не работает на платформе .NET, но имеет возможность взаимодействовать с компонентами .NET.

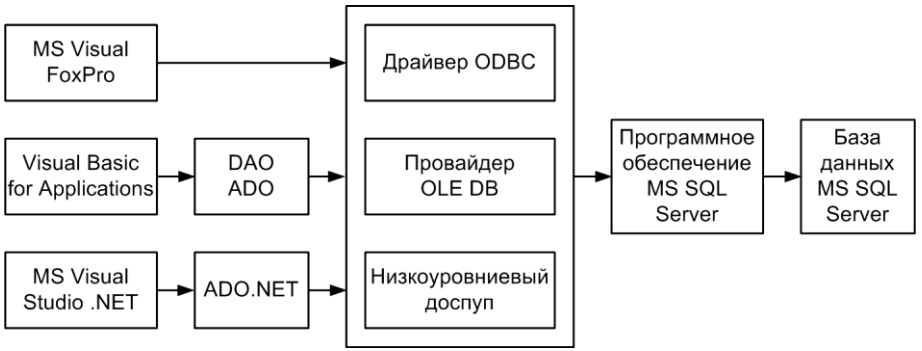


Рис. 12.10. Доступ к данным MS SQL-сервера из клиентских приложений

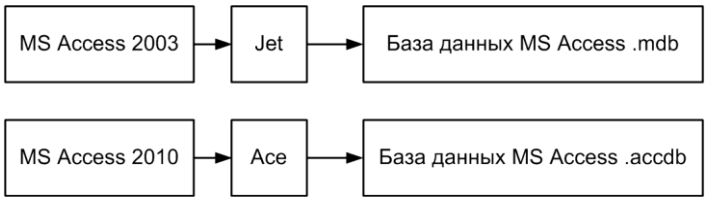


Рис. 12.11. Доступ MS Access к собственным данным

В *разд. 12.2.2* был рассмотрен способ перевода действующего приложения MS Access 2010 на платформу "клиент-сервер" с использованием связанных таблиц SQL Server 2008. Схема работы приложения со связанными таблицами приведена на рис. 12.12.

Рис. 12.12. Доступ MS Access 2010 к данным SQL Server при помощи "связывания" таблиц сервера

Почему компания Microsoft, отличающаяся достаточно жесткой линией в отношении перевода своих пользователей на новые архитектурные решения, посчитала нужным сохранить работу с проектом (файл adp) на прежнем уровне Access 2002/2003/2007 после трех лет работы над новой версией MS Access 2010 — пока остается загадкой. Что будет в следующей версии — покажет время.

ГЛАВА 13

Основные сведения об MS SQL Server 2008

MS SQL Server — замечательный инструмент. Он дает возможность решать самые разнообразные проблемы: от хранения информации на смартфонах до поддержки приложений, предусматривающих одновременный доступ множества пользователей к огромной базе данных. Работа с миллионами записей для MS SQL Server — заурядное явление.

13.1. Запуск MS SQL Server Management Studio

В распоряжение администратора баз данных создатели MS SQL Server 2008 предоставили несколько инструментов. Главный рабочий инструмент — SQL Server Management Studio. В его основу положена среда разработки Visual Studio .NET. Для запуска SQL Server Management Studio необходимо воспользоваться меню **Пуск** операционной системы вашего компьютера. Вызов выглядит так: **Пуск | Все программы | Microsoft SQL Server 2008 | SQL Server Management Studio**. Появится окно **Connect to Server** (Подключение к серверу). Выберите требуемый сервер, режим аутентификации, имя пользователя и пароль. Нажмите кнопку **Connect** (Подключиться). Появится окно, изображенное на рис. 13.1.

Интерфейс SQL Server Management Studio содержит несколько окон. Для их вывода на экран предназначен пункт **View** (Просмотр) главного меню. При первом запуске вы обязательно увидите окно обозревателя объектов **Object Explorer**. В нем расположено дерево объектов SQL-сервера. Путешествуя по дереву, можно дойти до полей любой таблицы, ее индексов, триггеров и т. д. Если в окне слишком много объектов — отфильтруйте их. Для отбора необходимых предназначена пиктограмма **Filter** (Фильтр) на панели инструментов. Другие окна появляются на экране автоматически в нужное время, в зависимости от действий, выполняемых пользователем. Принудительно любое окно можно вывести при помощи упомянутого выше пункта **View**. Рассмотрим их назначение.

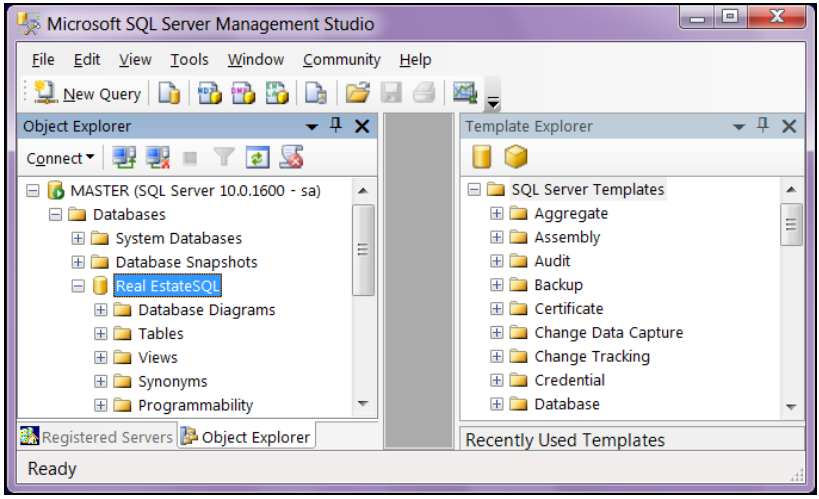


Рис. 13.1. Окно Microsoft SQL Server Management Studio

Окно зарегистрированных серверов **Registered Servers** предназначено для подключения к любому доступному серверу баз данных. Экземпляры MS SQL-сервера, имеющиеся на компьютере, с которого запущен инструмент SQL Server Management Studio, появляются автоматически. Для серверов с других компьютеров сети необходима регистрация. Сделайте щелчок правой кнопкой мыши в любом месте окна **Registered Servers** и выберите **New** (Новый) и **Server Registration** (Регистрация сервера).

Обозреватель шаблонов **Template Explorer** предоставляет в распоряжение разработчика заготовку кода для заданной операции. Хотите написать или откорректировать триггер — заготовки **Create** (Создать), **Alter** (Модифицировать) и **Drop** (Удалить) к вашим услугам. Измените в шаблоне необходимые параметры, указанные в угловых скобках, и код готов. Всего заготовлено 42 группы шаблонов для различных объектов. К ним можно добавить свои. Создайте файл с расширением sql и перенесите его в нужную папку. По умолчанию это папка:

```
C:\Program Files\Microsoft SQL Server\100\Tools\Binn\VSShell\Common7\
IDE\sqlworkbenchprojectitems\Sql
```

Окно обозревателя решений **Solution Explorer** дает возможность быстрого доступа к существующим проектам SQL-сервера.

Большинство параметров окна свойств **Properties Windows** доступно только для чтения. Окна бывают двух видов: для администрирования и разработки. Вид окна зависит от типа выбранного объекта.

13.2. Построение диаграммы базы данных

Вам, конечно же, хочется увидеть структуру конвертированной базы данных Real EstateSQL. Как она будет выглядеть в исполнении SQL Server 2008? Не менее наглядно, чем в MS Access 2010! Для вывода на экран структуры базы данных необходимо воспользоваться построителем диаграммы. Для этого:

1. В окне обозревателя объектов **Object Explorer** выберите интересующую нас базу данных.
2. Откройте список ее объектов, сделав щелчок мышью по значку "плюс" , расположенному слева от названия базы данных.
3. Щелкните правой кнопкой мыши по строке **Database Diagrams** (Диаграмма базы данных).
4. В появившемся контекстном меню выберите первый пункт **New Database Diagrams** (Новая диаграмма).
5. Появится первое окно мастера. Оно предназначено для выбора таблиц, которые требуется включить в диаграмму. Выберем все, кроме таблицы `tblUser`. Она не связана с остальными, и ее появление в схеме не требуется.

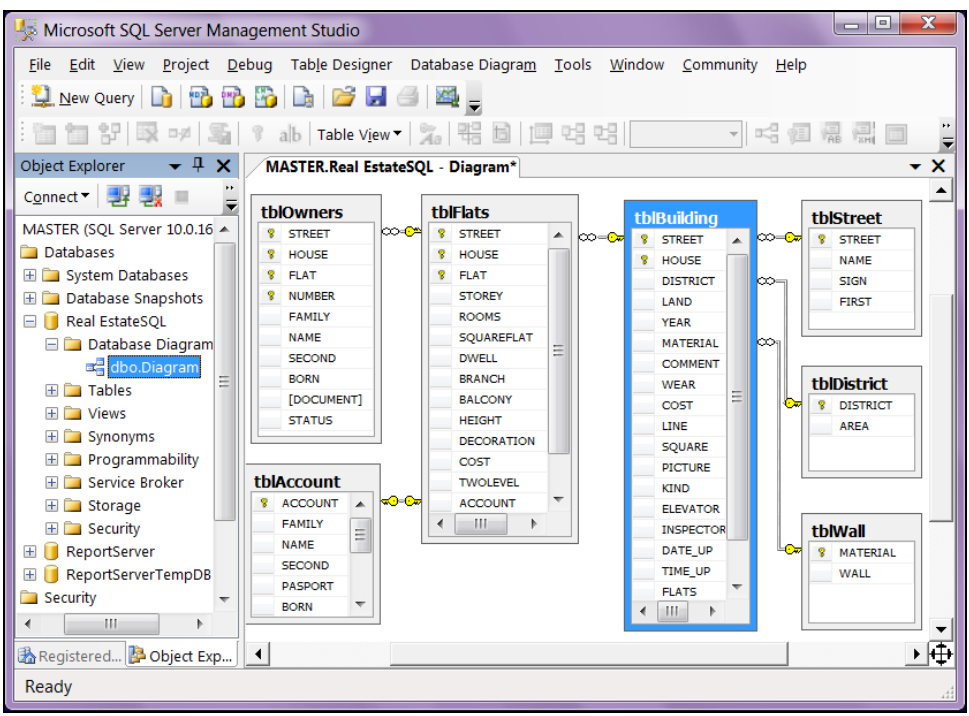


Рис. 13.2. Схема базы данных Real EstateSQL

6. Щелчок по кнопке **Close** (Закрыть) запустит процесс построения. Результаты работы мастера построения диаграмм показаны на рис. 13.2.

Возможности работы с объектами базы данных в окне диаграммы практически те же, что и у MS Access 2010:

- добавление таблицы в схему данных;
- создание новой таблицы с одновременным размещением ее в схеме;
- модификация таблицы, добавление и удаление полей, изменение типов данных;
- создание и удаление ключей и индексов;
- создание и удаление связей между таблицами;
- работа с ограничениями.

13.3. Схемы MS SQL Server 2008

В SQL Server 2008 каждая база данных состоит из схем "Schema". Схема содержит таблицы, индексы, триггеры, виды и т. д. Все объекты нашей базы данных Real Estate, созданной мастером преобразования, принадлежат одной схеме. Владельцем этих объектов является только она. Ее имя — dbo, что означает "владелец базы данных".

В SQL Server кроме схем базы данных существуют пользователи базы данных. Пользователи владеют схемами. Каждый пользователь всегда имеет назначенную схему по умолчанию и при обращении к объектам такой схемы может не указывать ее имя. Несколько пользователей могут владеть одной схемой через членство в той или иной группе Windows. Удаление пользователя не требует переименования объектов базы данных.

Схемы используются для ограничения видимости объектов. Объекты можно перемещать из одной схемы в другую в пределах одной и той же базы данных.

Предупреждение

Только слаженная совместная работа системного администратора, администратора сервера баз данных и разработчика программного комплекса может обеспечить использование всех возможностей, которыми обладает MS SQL Server 2008.

Для создания новой схемы:

1. В окне обозревателя объектов **Object Explorer** выберите интересующую нас базу данных.
2. Откройте список ее объектов, сделав щелчок мышью по значку "плюс"  , расположенному слева от названия базы данных.
3. Раскройте узел **Security** (Безопасность). Сделайте щелчок правой кнопкой мыши по узлу **Schemas** (Схемы). Появится меню.

4. Выберите в нем первый пункт **New Schema** (Новая схема). Появится окно, содержащее три страницы.
5. На первой странице **General** (Общие) в поле **Schema name** (Имя схемы) введите ее имя. В поле **Schema Owner** (Владелец схемы) укажите имя ее владельца.
6. Для поиска владельца предназначена кнопка **Search** (Поиск). После заполнения поля владельца нажмите кнопку **OK.**

Предупреждение

Созданную схему нельзя переименовать. Если такая необходимость возникла, то создайте новую схему, переместите в нее объекты из старой. Старую схему удалите.

13.4. Работа с таблицами

Основной объект схемы — таблица. Имя таблицы должно быть уникальным в пределах схемы. В различных схемах разные таблицы могут иметь одинаковые имена. Имя таблицы составное:

```
<Имя_базы_данных>.<Имя_схемы>.<Имя_таблицы>
```

Полное имя таблицы улиц `tblStreet` для рассматриваемого учебного примера имеет вид:

```
[Real EstateSQL].dbo.tblStreet
```

Имя базы данных Real EstateSQL заключено в квадратные скобки из-за пробела, присутствующего в составе имени.

Пользователь, владеющий схемой dbo (в нашем случае) по умолчанию, может обращаться к таблице улиц, указывая только последнюю часть имени: `tblStreet`.

13.4.1. Создание таблицы и ее модификация

Для создания таблицы в SQL Server Management Studio выполните следующие действия:

1. В окне обозревателя объектов **Object Explorer** откройте объект **Databases** (Базы данных), а затем узел требуемой базы данных.
2. Щелчок правой кнопкой мыши по объекту **Tables** (Таблицы) вызовет появление контекстного меню. Выберите в нем команду **New Table** (Новая таблица).
3. Откроется окно конструктора таблиц (рис. 13.3). Можно приступить к разработке структуры таблицы. В нашем распоряжении несколько окон.

Окно **Table** (Таблица) предназначено для ввода сведений о полях создаваемой таблицы. В нем три колонки: имя поля **Column Name**, тип данных **Data Type** и разрешение не вводить значение в поле при добавлении новой записи в таблицу **Allow Nulls**.

Занесите новое или выберите существующее поле таблицы, и в окне **Column Properties** появятся его свойства. Некоторые свойства недоступны для редактирования, если они унаследованы от объекта (база данных).

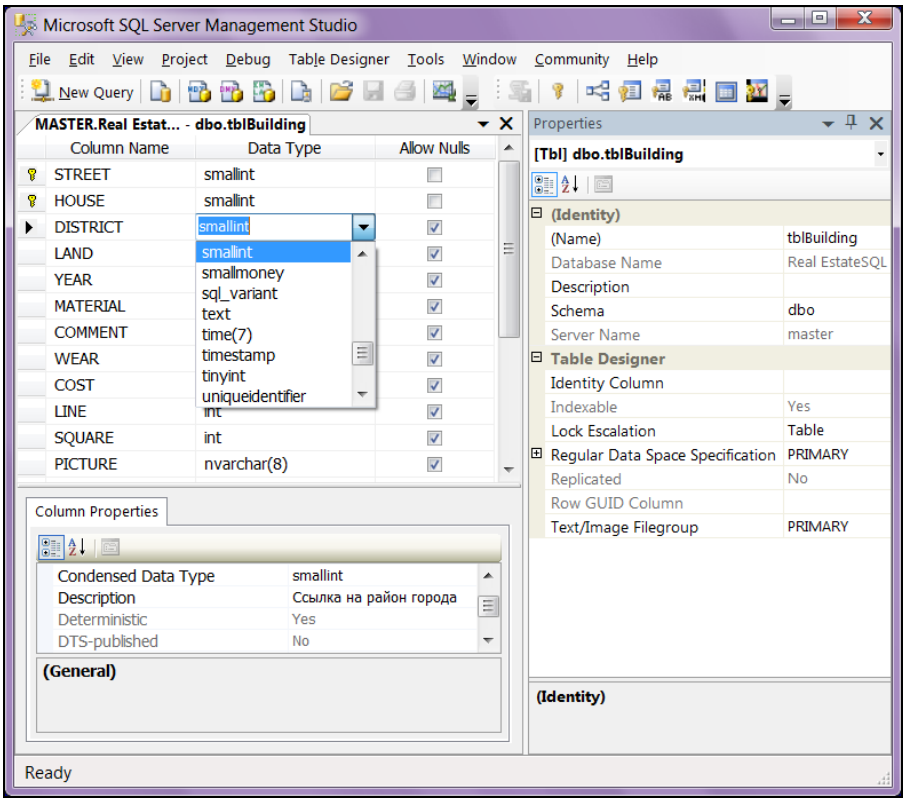


Рис. 13.3. Создание и модификация таблицы `tblBuilding` в конструкторе таблиц SQL Server Management Studio

Совет

Никогда не забывайте про свойство **Description** (Описание) окна **Column Properties** (Свойства полей). Щелчок по кнопке [...], расположенной в правой части строки, откроет дополнительное окно. Подробно напишите в нем о том, для чего поле создано.

Добавить поле в таблицу можно только в окне **Table**, а корректировать его название и тип данных можно как в этом окне, так и в окне **Column Properties**. При

выборе в качестве типа данных **Decimal** в окне **Column Properties** появятся дополнительные строчки: точность **Precision** и степень **Scale**, которые содержат максимальное количество десятичных знаков и максимальное количество знаков после десятичной точки.

Окно **Properties** (Свойства) предназначено для ввода имени таблицы и описания ее предназначения. Заполните строчки **Name** (Имя) и **Description** (Описание) соответственно. Для выбора схемы, которой будет принадлежать таблица, воспользуйтесь свойством **Schema** (Схема). После завершения работ по созданию таблицы щелкните по кнопке  **Save** (Сохранить) со стилизованным изображением дискеты, которая расположена на панели инструментов.

13.4.2. Просмотр информации о таблице

MS SQL Server Management Studio дает возможность пользователю получить исчерпывающую информацию о таблице. Для этого:

1. В окне обозревателя **Object Explorer** выберите интересующую нас базу данных.
2. Откройте список ее объектов, сделав щелчок мышью по значку "плюс"  , расположенному слева от названия базы данных.
3. Откройте узел **Tables** (Таблицы).
4. Щелкните правой кнопкой мыши по названию таблицы, информацию о которой требуется получить.
5. В появившемся контекстном меню выберите последний пункт **Properties** (Свойства).
6. Откроется окно **Table Properties** (Таблица свойств).

В появившемся окне несколько вкладок: общие **General**, разрешения **Permissions**, расширенные **Extended Properties** и т. д. На этих вкладках содержится:

- размер таблицы на жестком диске (строка пространство данных **Data space**);
- размер области, которую занимают индексы (строка пространство индексов **Index space**).
- количество строк в таблице (строка **Row count**);
- дата создания таблицы (строка **DateCreated**);
- дата последней модификации (строка **LastUpdated**);
- разрешения для таблицы;
- другая полезная информация.

13.4.3. Копирование, переименование и удаление таблиц

Для создания копии таблицы лучше всего использовать Transact-SQL. Запустим построитель запросов.

1. Находясь в MS SQL Server Management Studio, подключитесь к нужному серверу.
2. Выберите в стандартной панели инструментов **Standard** пиктограмму нового запроса **New Query**.
3. Если панель отсутствует на экране — воспользуйтесь правой кнопкой мыши для вызова контекстного меню. Щелкните по имени сервера в окне **Object Explorer** (Обозреватель объектов) и выберите пункт **New Query** (Новый запрос).
4. Появится окно конструктора запросов (рис. 13.4).

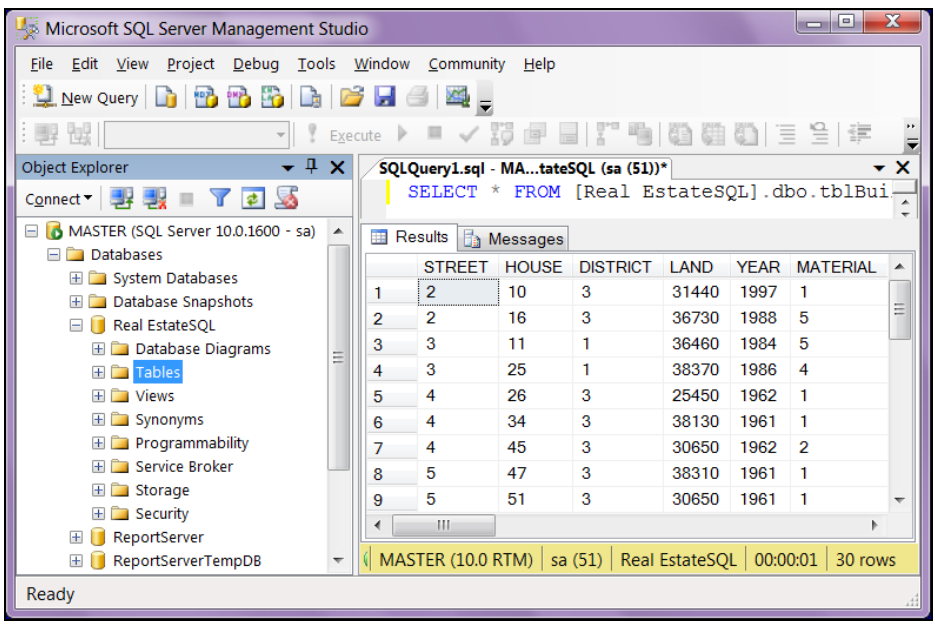


Рис. 13.4. Окно конструктора запросов MS SQL Server Management Studio

5. Окно состоит из двух частей. Верхняя часть предназначена для ввода запроса, а нижняя — для отображения его результатов. При первом запуске конструктора вторая часть окна не отображается на экране дисплея. Наберите текст запроса, нажмите кнопку **Execute** (Выполнить), которая расположена на панели инструментов, и нижняя часть окна с результатами выполнения появится на

экране. Вкладок в нижней части окна может быть две: **Results** (Результаты) и **Messages** (Сообщения).

Следующий пример демонстрирует копирование таблицы `tblBuilding` из схемы `dbo` в схему `Inspector`.

```
SELECT * INTO Inspector.tblBuilding FROM dbo.tblBuilding
```

В окне **Messages** (Сообщения) появится результат:

```
(30 row(s) affected)
```

Для удаления или переименования таблицы выполните следующие действия:

1. В окне обозревателя объектов **Object Explorer** откройте объект **Databases** (Базы данных), а затем узел требуемой базы данных.
2. Щелчок правой кнопкой мыши по объекту **Tables** (Таблицы) вызовет появление контекстного меню. Выберите в нем команду **View Dependencies** (Просмотреть зависимости). При удалении выбранной таблицы или ее переименовании указанные в этом окне зависимости между объектами будут разорваны. Внимательно изучите их. Закройте это окно.
3. Для удаления таблицы щелкните правой кнопкой мыши по ее имени и выберите в появившемся меню команду **Delete** (Удалить).
4. Для переименования таблицы щелкните правой кнопкой мыши по ее имени и выберите в появившемся меню команду **Rename** (Переименовать).

13.4.4. Просмотр значений данных в таблице

Просмотр содержимого таблицы можно осуществить двумя способами:

- с помощью MS SQL Server Management Studio и с применением инструкции Transact-SQL;
- с применением MS SQL Server Management Studio, которое выглядит следующим образом:
  - в окне обозревателя **Object Explorer** выберите интересующую нас базу данных;
  - откройте список ее объектов, сделав щелчок мышью по значку "плюс"  , расположенному слева от названия базы данных;
  - откройте узел **Tables** (Таблицы);
  - щелкните правой кнопкой мыши по названию таблицы, содержимое которой хотите просмотреть;
  - в появившемся контекстном меню выберите пункт **Edit Top 200 Rows** (Редактировать первые 200 строк);

- откроется окно **Table** (Таблица). В нижней части окна расположены кнопки перехода между записями таблицы.

При работе с запросом Transact-SQL результат будет тот же, но кнопок перехода на экране не будет (см. рис. 13.4).

13.5. Типы данных MS SQL Server 2008

Каждый столбец таблицы MS SQL-сервера должен иметь конкретный тип данных. От него зависит информация, которую можно хранить в этом столбце. После того как будет определен тип данных для столбца таблицы, изменить его в большинстве случаев будет нельзя. В табл. 13.1 приведены основные типы данных сервера.

Таблица 13.1. *Основные типы данных MS SQL Server 2008*

Категория	Тип данных	Размер	Диапазон
Числовые типы	`bit`	1 бит	0, 1 или `Null`
	`tinyint`	1 байт	От 0 до 255
	`smallint`	2 байта	От –32 768 до 32 767
	`int`	4 байта	От –2 147 483 648 до 2 147 483 647
	`bigint`	8 байт	От -2^{63} до 2^{63}
	`decimal`	5—17 байтов	От 10^{-38} до 10^{38}
	`real`	4 байта	От -3.4×10^{38} до 3.4×10^{38}
	`float`	4—8 байтов	От -1.79×10^{308} до -2.23×10^{-308}
	`money`	8 байтов	От –922 337 203 685 477.5808 до 922 337 203 685 477.5807
	`smallmoney`	4 байта	От –214 748.3648 до 214 748.3647
Дата и время	`smalldatetime`	4 байта	От 1 января 1900 года до 6 июня 2079 года с точностью 1 минута
	`datetime`	8 байтов	От 1 января 1753 года до 31 декабря 9999 года с точностью 3,33 мс
Специальный тип	`xml`	Переменный	Хранение данных XML
Строковые типы	`char`	1 байт на символ	Символьные данные постоянной длины (не Unicode) до 8000 символов
	`varchar`	1 байт на символ + 2 байта на указатель	Символьные данные переменной длины (не Unicode) до 8000 символов

Таблица 13.1 *(окончание)*

Категория	Тип данных	Размер	Диапазон
	varchar(max)	1 байт на символ + 2 байта на указатель	Символьные данные переменной длины (не Unicode) до 2 147 483 647 символов
	text	1 байт на символ	Символьные данные переменной длины (не Unicode) до 2 147 483 647 символов
	nchar	2 байта на символ	Символьные данные постоянной длины (Unicode) до 4000 символов
	nvchar	2 байта на символ + 2 байта на указатель	Символьные данные переменной длины (Unicode) до 4000 символов
	nvchar(max)	2 байта на символ + 2 байта на указатель	Символьные данные переменной длины (Unicode) до 1 073 741 823 символов
	ntext	2 байта на символ	Символьные данные переменной длины (Unicode) до 1 073 741 823 символов
Бинарные	binary	Размер в байтах	Бинарные данные фиксированной длины до 8000 байтов
	varbinary	Размер в байтах + 2 байта на указатель	Бинарные данные переменной длины до 8000 байтов
	varbinary(max)	Размер в байтах + 2 байта на указатель	Бинарные данные переменной длины до 2 147 483 647 байтов
	image	Размер в байтах	Бинарные данные переменной длины до 2 147 483 647 байтов

В MS SQL Server 2008 имеется тип данных — `xml`, который позволяет хранить данные не в виде строки, а непосредственно в формате XML. Пользователю предоставлена возможность создавать индексы по полям этого типа. Обратите внимание еще на три типа данных: `varchar(max)`, `nvchar(max)`, `varbinary(max)`. Они предназначены для хранения больших объектов размером до 2 Гбайт.

13.6. Преобразование типов данных

При работе с данными разных типов в большинстве случаев пользователю не придется выполнять явные преобразования. Так, например, MS SQL Server автоматически преобразует выражение к типу `int`, если в нем применяются данные `int`, `tinyint` и `smallint`. Однако в ряде случаев, когда без явного преобразования данных не обойтись, на помощь разработчику приходят функции `CAST()` и

CONVERT(). Их можно применять во всех операторах Transact-SQL, где используются выражения. Синтаксис функций следующий.

Функция CAST:

```
CAST(expression AS data_type [ (length ) ])
```

Функция CONVERT:

```
CONVERT(data_type [ ( length ) ], expression [, style ] )
```

Здесь:

- *expression* — любое, допустимое в Transact-SQL выражение;
- *data_type* — тип данных, в который требуется преобразовать выражение;
- *length* — длина типа данных (в ряде случаев не требуется);
- *style* — используется для отображения даты и времени в различных форматах.

Пример работы с функцией CAST показан на рис. 13.5.

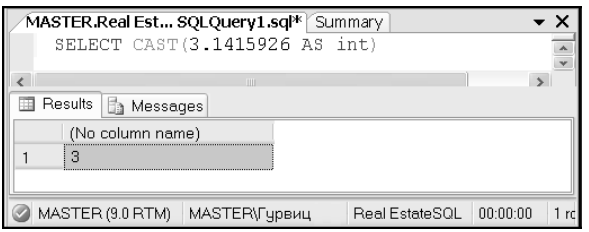


Рис. 13.5. Результат преобразования 3.1415926 к целому типу int

Еще один пример преобразования 3.1415926, но к денежному типу:

```
SELECT CAST(3.1415926 AS money)
```

Результат: 3.1416.

Основное назначение функции CONVERT() — преобразование типа данных "дата и время" в строковые типы. Отображение предусмотрено абсолютно для всех случаев, которые только могут встретиться на практике. Для этих целей предназначен аргумент style (табл. 13.2).

Таблица 13.2. *Номера стилей для функции CONVERT()*

Без столетия (уу)	Со столетием (уууу)	Стандарт	Отображение "вход/выход"
—	0 или 100	Принимаемый по умолчанию	mon dd yyyy hh:miAM (или PM)

Таблица 13.2 *(окончание)*

Без столетия (уу)	Со столетием (уууу)	Стандарт	Отображение "вход/выход"
1	101	США	mm/dd/yyyy
2	102	ANSI	yy.mm.dd
3	103	Английский/ французский	dd/mm/yy
4	104	Немецкий	dd.mm.yy
5	105	Итальянский	dd-mm-yy
6	106		dd mon yy
7	107		Mon dd, yy
8	108		hh:mm:ss
	9 или 109	Принимаемый по умолчанию с миллисекундами	mon dd yyyy hh:mi:ss:mmmAM (или PM)
10	110	США	mm-dd-yy
11	111	Япония	yy/mm/dd
12	112	ISO	yymmdd
	13 или 113	Европа с миллисекундами	dd mon yyyy hh:mm:ss:mmm (24-часовой цикл)
14	114		hh:mi:ss:mmm (24-часовой цикл)
	20 или 120	Канонический ODBC	yyyy-mm-dd hh:mi:ss (24-часовой цикл)
	21 или 121	Канонический ODBC с миллисекундами	yyyy-mm-dd hh:mi:ss.mmm (24-часовой цикл)
	126	ISO8601	yyyy-mm-dd Thh:mm:ss.mmm (no spaces)
	127	ISO8601 с часовым поясом Z	yyyy-mm-ddThh:mm:ss.mmmZ (без пробелов)
	130	Hijri (Кувейтский алгоритм календарной системы Хиджра)	dd mon yyyy hh:mi:ss:mmmAM
	131	Hijri (Кувейтский алгоритм календарной системы Хиджра)	dd/mm/yy hh:mi:ss:mmmAM

В следующем примере выводятся текущие дата и время:

- заданные неявно;
- заданные немецким стандартом (только дата без столетия и времени);
- заданные стандартом по умолчанию с миллисекундами;
- заданные для всех стандартов (только время без миллисекунд).

```
SELECT GETDATE(),
       CONVERT(CHAR(12),GETDATE(),4),
       CONVERT(CHAR(24),GETDATE(),109),
       CONVERT(CHAR(10),GETDATE(),108)

------ Кнопка  !Execute ------
2010-05-25 10:13:51.577
25.05.07
May 25 2010 10:13:51:577
10:13:51
```

13.7. Основы Transact-SQL

MS SQL Server 2008 использует для доступа к данным специализированный язык программирования Transact-SQL, который является диалектом языка SQL (Structured Query Language) стандарта ANSI SQL-92. Кроме этого, Transact-SQL дополнен конструкциями, увеличивающими мощность и гибкость этого языка. Далее будут рассмотрены основные элементы Transact-SQL.

13.7.1. Идентификаторы

Идентификаторы в Transact-SQL являются именами объектов, такими как переменная, таблица, поле таблицы, индекс, триггер, хранимая процедура и т. д. Идентификатор объекта создается при определении объекта и используется для ссылки на него. При выборе имени объекта следует руководствоваться такими правилами:

- имя должно начинаться с буквы или с символов: `@`, `$`, `_`, `#`;
- имя должно быть не длиннее 128 символов;
- имя не может содержать пробел, точку, запятую, `&`, круглые и фигурные скобки, а также ряд других специальных символов: `^`, `!`, `%`, `-`, `"`, `\`, `'`, `~`;
- имя не должно совпадать с зарезервированными словами языка.

В Transact-SQL существуют два типа идентификаторов:

- стандартные идентификаторы — это имена, назначенные по изложенным выше правилам;

- ограниченные идентификаторы — это имена, назначенные с нарушением правил. Использование таких идентификаторов нежелательно, но возможно. Эти имена должны быть заключены в квадратные скобки или двойные кавычки.

В следующем примере рассмотрен запрос на выборку всех записей из таблицы `tblStreet`:

```
SELECT * FROM [Real EstateSQL].dbo.tblStreet
```

В нем имя базы данных Real EstateSQL содержит пробел, поэтому в приведенном операторе `SELECT` оно заключено в квадратные скобки. Стандартный идентификатор может использоваться как с ограничителями, так и без них. Идентификатор, созданный не по правилам, обязательно должен быть заключен в ограничители.

Поговорим подробнее о применении ограничителей при работе с идентификаторами и символьными строками. Как вы уже знаете, в качестве ограничителей идентификаторов применяются квадратные скобки и двойные кавычки. Символьные строки ограничиваются этими же знаками. Когда и что из этого арсенала следует применять? Как не допустить ситуацию, в которой символьная строка, заключенная в двойные кавычки, будет принята за объект базы данных и наоборот?

Для решения этой проблемы в Transact-SQL существует конструкция:

```
SET QUOTED_IDENTIFIER {ON | OFF}
```

Попробуйте создать в MS SQL Server Management Studio хранимую процедуру, функцию или триггер. В начале текста увидите заготовку как минимум из двух строк:

```
SET QUOTED_IDENTIFIER ON
GO
```

Значение параметра `QUOTED_IDENTIFIER`, установленное в `ON`, заставляет MS SQL Server подчиниться следующим правилам использования ограничителей:

- символьные строки должны быть заключены в одинарные кавычки. Если строка содержит одинарную кавычку, то рядом с ней должна быть поставлена еще одна одинарная кавычка;
- двойные кавычки могут применяться только для ограничения идентификаторов.

При значении параметра `QUOTED_IDENTIFIER`, установленного в `OFF`, действуют другие правила:

- символьные строки могут быть заключены как в одинарные, так и в двойные кавычки. Если строка содержит одинарную кавычку, то строка должна быть ограничена двойными кавычками;
- двойные кавычки нельзя применять для ограничения идентификаторов.

13.7.2. Комментарии

В текст, написанный на языке Transact-SQL, разработчик может вставлять пояснения, содержащие любые символы. Такие пояснения носят названия *комментариев*. Комментарии игнорируются системой при исполнении написанного кода. Они предназначены только для пользователя. В Transact-SQL существуют два типа комментариев. В листинге 13.1 приведены комментарии обоих типов:

- *строчный комментарий* — начинается с двух знаков "минус", расположенных друг за другом. Все символы строки после двух минусов до ее конца будут игнорироваться транслятором;
- *блочный комментарий* — начинается со знаков `/*` и обязательно заканчивается знаками `*/`. Все символы между такими "скобками" также игнорируются транслятором.

Листинг 13.1. Работа с комментариями в Transact-SQL

```
/* Эти конструкции сгенерированы автоматически */
SET ANSI_NULLS ON
SET QUOTED_IDENTIFIER ON
GO

ALTER TRIGGER [T_tblFlats_UpTrig] -- Модификация триггера
ON [dbo].[tblFlats] FOR UPDATE, INSERT AS
SET NOCOUNT ON

Declare @SqFlat REAL        -- Площадь квартиры
Declare @SqBal REAL         -- Площадь балкона

IF UPDATE /* Поля модифицировались */ (SQUAREFLAT)
   OR UPDATE(DWELL)
      BEGIN
         SELECT @SqFlat =inserted.SQUAREFLAT,
                @SqBal=inserted.BALCONY
             FROM inserted
         IF @SqFlat,1) < @SqBal
             ROLLBACK TRANSACTION     -- Откат транзакции
      END
```

13.7.3. Переменные

Переменная — это область памяти, имеющая имя. Переменные предназначены для хранения одинарных значений и очень часто используются для передачи данных между командами. В Transact-SQL существуют переменные двух типов: локальные и глобальные.

С *локальной переменной* всегда ассоциируется определенный тип данных. Перед использованием переменную необходимо объявить. Локальная переменная объявляется в коде программы. Для этого используется следующий оператор:

```
DECLARE <имя_переменной> <тип_данных>
```

Имя локальной переменной должно начинаться с символа `@`. Это отличительный признак любой локальной переменной. Остальные символы назначаются в соответствии с правилами создания идентификаторов. В следующем примере объявлена переменная с именем `@Account` целого типа:

```
DECLARE @Account int
```

В одном операторе можно объявить несколько переменных, перечислив их через запятую:

```
DECLARE @LastName nvarchar(20),
        @FirstName nvarchar(15),
        @SecondName nvarchar(25)
```

Глобальные переменные определяются на уровне сервера. Они поддерживаются системой и доступны в любой момент времени. Имя глобальной переменной начинается с двух символов: `@@`.

Переменной можно присвоить значение двумя способами: с помощью команд `SET` и `SELECT`. При помощи `SET` присваивается конкретное значение, например:

```
DECLARE @ModelCar nvarchar(15)
SET @ModelCar='Carina'
```

При помощи `SELECT` переменной можно присвоить результат вычисления выражения. В следующем примере переменная `@SqFlat` получает значение суммы всех элементов столбца `SQUAREFLAT` таблицы `tblBuilding`.

```
DECLARE @SqFlat real
SELECT @SqFlat = SUM(SQUAREFLAT) FROM tblBuilding
```

13.7.4. Выражения

В состав выражения MS SQL Server могут входить переменные, константы, функции, поля таблиц, подзапросы, служебные слова, знаки операций и т. д. Другими словами, операнды (данные) и операторы (действия над данными). Операторы делятся на арифметические, операторы сравнения, логические и битовые. Арифметические операторы (табл. 13.3) выполняют сложение, вычитание, умножение и деление.

Таблица 13.3. *Арифметические операторы Transact-SQL*

Оператор	Описание
+	Складывает два операнда
–	Вычитает один операнд из другого

Таблица 13.3 *(окончание)*

Оператор	Описание
- (унарный)	Меняет знак операнда
*	Перемножает два операнда
/	Делит один операнд на другой
%	Возвращает остаток от целочисленного деления

Операторы сравнения сравнивают значения двух операндов и возвращают логические значения: `True` (если условие выполняется) или `False` (если условие не выполняется). Операторы сравнения приведены в табл. 13.4.

Таблица 13.4. *Операторы сравнения Transact-SQL*

Оператор	Описание
>	Больше
>=	Больше или равно
<	Меньше
<=	Меньше или равно
<> или !=	Не равно
!<	Не меньше
!>	Не больше

Логические операторы `AND`, `OR` и `NOT` используются для объединения результатов двух или более сравнений в одно (табл. 13.5).

Таблица 13.5. *Логические операторы Transact-SQL*

Оператор	Описание	Примеры	Результат
AND	Логическое "И" (конъюнкция)	True AND False	False
		True AND True	True
OR	Логическое "ИЛИ" (дизъюнкция)	False OR False	False
		True OR False	True
NOT	Логическое отрицание	NOT False	True
		NOT True	False

Для демонстрации работы выражений Transact-SQL запустим построитель запросов. Алгоритм запуска следующий:

1. Находясь в MS SQL Server Management Studio, подключитесь к нужному серверу.
2. Выберите в стандартной панели инструментов **Standard** пиктограмму нового запроса **New Query**.
3. Если панель отсутствует на экране — воспользуйтесь правой кнопкой мыши для вызова контекстного меню. Щелкните по имени сервера в окне **Object Explorer** (Обозреватель объектов) и выберите пункт **New Query** (Новый запрос).
4. Появится окно конструктора запросов (см. рис. 13.4).
5. Окно состоит из двух частей. Верхняя часть предназначена для ввода запроса, а нижняя — для отображения его результатов. При первом запуске конструктора вторая часть окна не отображается на экране дисплея. Наберите текст запроса, нажмите кнопку **! Execute** (Выполнить), которая расположена на панели инструментов, и нижняя часть окна с результатами выполнения появится на экране. Вкладок в нижней части окна может быть две: **Results** (Результаты) и **Messages** (Сообщения).

Кроме `AND`, `OR` и `NOT` в Transact-SQL существует еще 7 логических операторов.

- `ALL` сравнивает проверяемое значение со всеми значениями из набора. Если условие выполняется для всех значений, то возвращается `True`. В приведенном далее примере выясняется, есть ли хотя бы один проживающий с фамилией Иванов.

```
IF 'Иванов' <> ALL (SELECT FAMILY FROM tblAccount)
   PRINT 'Таких проживающих нет'
ELSE
   PRINT 'Фамилия есть в списке'
```

- `BETWEEN` возвращает `True`, если проверяемое значение находится в указанном диапазоне, и `False`, если нет. В следующем примере сделана выборка всех зданий, год постройки которых находится в интервале 1970—1980. Если проверяемое значение совпадает с указанными границами, то `BETWEEN` также возвращает значение `True`.

```
SELECT STREET,HOUSE,YEAR FROM tblBuilding
       WHERE YEAR BETWEEN 1970 AND 1980

------ Кнопка  !Execute ------
14  57  1977
14  58  1971
16  7  1980
17  13  1970
```

Такой же результат даст выборка:

```
SELECT STREET,HOUSE,YEAR FROM tblBuilding
       WHERE YEAR >= 1970 AND YEAR<=1980
```

- ANY (или SOME) сравнивает проверяемое значение со всеми значениями из набора. Если условие выполняется хотя бы для одного из значений, то возвращается True. В приведенном далее примере выясняется, есть ли хотя бы один проживающий с фамилией Иванов.

```
IF 'Иванов' = ANY (SELECT FAMILY FROM tblAccount)
   PRINT 'Фамилия есть в списке'
ELSE
   PRINT 'Таких проживающих нет'
```

- EXIST возвращает True, если подзапрос содержит хотя бы одну строку, и False в противном случае. В следующем примере выясняется, есть ли хотя бы один проживающий с фамилией Иванов.

```
IF  EXIST (SELECT * FROM tblAccount WHERE FAMILY='Иванов')
   PRINT 'Фамилия есть в списке'
ELSE
   PRINT 'Таких проживающих нет'
```

Примечание

Применяя оператор EXIST, можно проверить существование значений, удовлетворяющих множеству условий, а операторы ANY и SOME позволяют задать только одно логическое условие.

- IN сравнивает проверяемое значение со всеми значениями из набора. Если условие выполняется хотя бы для одного из значений, то возвращается True. В приведенном далее примере выясняется, есть ли хотя бы один проживающий с фамилией Иванов.

```
IF 'Иванов' IN (SELECT FAMILY FROM tblAccount)
   PRINT 'Фамилия есть в списке'
ELSE
   PRINT 'Таких проживающих нет'
```

Оператор IN фактически заменяет ANY, но функциональность его выше, т. к. в качестве значений может выступать не только подзапрос, но и перечисление величин, например:

```
IF 'Иванов' IN ('Петров','Сидоров')
   PRINT 'Фамилия есть в списке'
ELSE
   PRINT 'Таких проживающих нет'

------ Кнопка  !Execute ------
Таких проживающих нет
```

- `LIKE` возвращает `True`, если выражение совпадает с шаблоном, в противном случае возвращается `False`. В следующем примере выведен список квартиросъемщиков, чья фамилия начинается с букв "Ив".

```
SELECT FAMILY,NAME,SECOND FROM tblAccount
       WHERE FAMILY LIKE 'Ив%'
```

```
Иванов          Евгений        Владимирович
Иванченко       Наталья        Анатольевна
Иванова         Валентина      Михайловна
Иваненко        Клавдия        Михайловна
Иванищев        Александр      Сергеевич
Иванча          Тамара         Григорьевна
Иваночко        Владимир       Николаевич
Ивашинова       Мария          Ивановна
Ивашина         Надежда        Ивановна
Ивлева          Жанна          Георгиевна
```

В следующем примере выведен список квартиросъемщиков, в фамилии которых вторую, четвертую и шестую позицию занимает буква "а".

```
SELECT FAMILY,NAME,SECOND FROM tblAccount
WHERE FAMILY LIKE '_a_a_a%'
```

```
------ Кнопка  !Execute ------
Паламарчук      Вера           Владимировна
Тараканова      Анисья         Николаевна
Барабаш         Раиса          Антоновна
Гавага          Галина         Владимировна
Катаганов       Руслан         Назирович
Карабанова      Марианн        Леонидовна
```

13.7.5. Управляющие конструкции

Для изучения работы управляющих конструкций Transact-SQL запустим построитель запросов. Алгоритм запуска следующий:

1. Находясь в MS SQL Server Management Studio, подключитесь к нужному серверу.
2. Выберите в стандартной панели инструментов **Standard** пиктограмму нового запроса **New Query**.
3. Если панель отсутствует на экране — воспользуйтесь правой кнопкой мыши для вызова контекстного меню. Щелкните по имени сервера в окне **Object Explorer** и выберите пункт **New Query** (Новый запрос).
4. Появится окно конструктора запросов (см. рис. 13.4).

5. Окно состоит из двух частей. Верхняя часть предназначена для ввода запроса, а нижняя — для отображения его результатов. При первом запуске конструктора вторая часть окна не отображается на экране дисплея. Наберите текст запроса, нажмите кнопку **! Execute** (Выполнить), которая расположена на панели инструментов, и нижняя часть окна с результатами выполнения появится на экране. Вкладок в нижней части окна может быть две: **Results** (Результаты) и **Messages** (Сообщения).

Как и все алгоритмические языки, Transact-SQL содержит в своем составе операторные скобки. Синтаксис конструкции имеет следующий вид:

```
BEGIN
  <команда1>
  <команда2>
  ...
END
```

Операторные скобки применяются для объединения нескольких команд в блок. Такая группа воспринимается транслятором как одна команда. В следующем примере подсчитывается и выводится общее количество записей в двух таблицах: `tblBuilding` и `tblStreet`.

```
DECLARE @CountRecordALL smallint  -- Общее количество записей
DECLARE @CountRecordPart smallint -- Записей в одной таблице
BEGIN
   SELECT @CountRecordALL=COUNT(*)FROM tblBuilding
   SELECT @CountRecordPart=COUNT(*)FROM tblStreet
   SET @CountRecordALL=@CountRecordALL+@CountRecordPart
   PRINT @CountRecordALL
END

------ Кнопка  !Execute ------
1231
```

Блоки могут быть вложенными. В этом случае для улучшения восприятия кода тело блока смещается вправо.

Для организации разветвлений в алгоритме Transact-SQL представляет пользователю конструкцию `IF...ELSE`. В его трактовке вы не найдете служебного слова `THEN`, да и команд после слов `IF` и `ELSE` может быть только по одной. Синтаксис конструкции ветвления имеет вид:

```
IF <условие>
  <команда1>
ELSE
  <команда2>
```

Служебное слово `ELSE` и команда, следующая за ним, могут быть опущены. Если в какой-либо ветке алгоритма необходимо выполнить более чем по одной команде, то следует воспользоваться конструкцией `BEGIN...END`.

```
DECLARE @CountBuild smallint,@CountFlat smallint
SELECT @CountBuild=COUNT(*)FROM tblAccount
IF (@CountBuild<20000 AND MONTH(GETDATE())=6)
   -- Слишком маленький размер таблицы для июня
   BEGIN
      PRINT 'Это не та копия базы данных'
      SELECT @CountFlat=COUNT(*)FROM tblFlats
      IF @CountFlat BETWEEN 2000 AND 2200
         PRINT 'Скорее всего, это архив за апрель'
   END
ELSE
   PRINT 'Это архив за май'

------ Кнопка  !Execute ------
Это не та копия базы данных
```

Если разветвлений в алгоритме много и они одиночные, то для их замены и приведения текста программы к более компактному виду можно воспользоваться конструкцией CASE...END. Общий синтаксис конструкции следующий:

```
CASE <входное_выражение>
  WHEN <выражение1> THEN <выражение_результат>
  WHEN <выражение2> THEN <выражение_результат>
  ...
  ELSE <выражение_результат>
END
```

Обязательно обратите внимание на то, что в Transact-SQL конструкция фактически представляет собой функцию. У функции имеется один параметр (*входное_выражение*). Он указывается после CASE, но не в скобках. Функция возвращает результат, и поэтому связка CASE...END обязательно должна входить в состав какого-либо выражения Transact-SQL. Если в примере, рассмотренном далее, убрать PRINT, то система выдаст сообщение об ошибке, т. к. это будет неверный вызов функции.

Работает конструкция следующим образом. Если значение входного выражения и одно из вычисленных, стоящих после WHEN, совпадают, то возвращается выражение-результат, указанное после соответствующего THEN. Если ни одного совпадения не отмечено, то возвращается результат, стоящий после ELSE.

Если значение входного выражения совпадает более чем с одним значением выражения, стоящего после WHEN, то выполняется результат, соответствующий первому совпадению.

```
DECLARE @Rouble char(25)
SET @Rouble='6 руб'
PRINT
```

```
  CASE @Rouble
       WHEN '1 руб' THEN 'Один рубль'
       WHEN '2 руб' THEN 'Два рубля'
       WHEN '5 руб' THEN 'Пять рублей'
       ELSE 'Таких денег в кассе нет'
  END

------ Кнопка  !Execute ------
Таких денег в кассе нет
```

В следующем примере показано применение конструкции `CASE...END` непосредственно в SQL-запросе. Не следует забывать, что фактически это ее основное предназначение в Transact-SQL. В запросе формируется список улиц города, в котором добавлен столбец, содержащий сокращенное название признака объекта.

```
SELECT   STREET,NAME,PRIZNAK =
      CASE SIGN
         WHEN 'Улица' THEN 'Ул.'
         WHEN 'переулок' THEN 'пер.'
         WHEN 'проезд' THEN 'пр.'
         WHEN 'площадь' THEN 'пл.'
         WHEN 'шоссе' THEN 'ш.'
         ELSE 'нет'
      END
FROM tblStreet
ORDER BY NAME

------ Кнопка  !Execute ------
STREET NAME       SIGN      PRIZNAK
1  [нет]          [нет]      Нет
2  Абрикосовая    Улица      Ул.
3  Авангардная    Улица      Ул.
4  Авачинская     Улица      Ул.
5  Авиационная    площадь    пл.
6  Авроры         Улица      Ул.
7  Автобусная     Улица      Ул.
8  Автобусный     переулок   пер.
9  Автономный     переулок   пер.
10  Адмиральская  Улица      Ул.
11  Адмиральский  переулок   пер.
12  Азовский      переулок   пер.
13  Айвазовского  шоссе      ш.
14  Айвазовского  проезд     пр.
15  Акмолинская   Улица      Ул.
```

В Transact-SQL имеется конструкция для организации многократных повторений команд в программе: `WHILE...CONTINUE`. Она обеспечивает выполнение цикла только одного типа. Это цикл с предусловием. Как вы знаете, почти все языки программирования поддерживают три типа циклов: с предусловием, с постусловием и с параметром, но отсутствующие в Transact-SQL могут быть успешно заменены именно этим типом цикла. Синтаксис конструкции следующий:

```
WHILE <условие>
  {команда | блок}
[BREAK]
  {команда | блок}
[CONTINUE]
```

Цикл можно завершить принудительно. Для этого в нужном месте тела цикла нужно поместить служебное слово `BREAK`. В следующем примере вычисляются и выводятся квадрат и куб чисел от 1 до 5.

```
DECLARE @Number tinyint
PRINT 'Число    Квадрат    Куб'
SET @Number = 1
WHILE @Number<=5
      BEGIN
        PRINT LTRIM(STR(@Number)+STR(@Number*@Number)
              + STR(@Number*@Number*@Number))
        SET @Number=@Number+1
      END
------ Кнопка  !Execute ------
Число    Квадрат    Куб
1          1          1
2          4          8
3          9         27
4         16         64
5         25        125
```

Примечание

В каждом примере этого раздела приведены наборы команд Transact-SQL. Они посылаются на сервер одним пакетом при нажатии кнопки **! Execute** (Выполнить) на панели инструментов. Выполняются они так же, как единое целое. Если вам необходимо разбить такой пакет на отдельные участки кода и выполнить отдельно, один за другим, то поставьте в нужном месте текста ключевое слово `GO`.

В Transact-SQL MS SQL Server 2008 разработчики компании Microsoft включили возможность обработки исключений. Текущая версия позволяет обрабатывать некритические ошибки с помощью похожего на ставший уже стандартным синтаксис `TRY...CATCH`.

Вид его следующий:

```
BEGIN TRY
 -- Запрос, вызывающий опасения
END TRY

BEGIN CATCH
 -- Обработка ошибки, которая может возникнуть
END CATCH
```

13.8. Функции MS SQL Server 2008

В Microsoft SQL Server 2008 имеется набор стандартных встроенных функций. Многие из них отличаются от функций Microsoft Access. Математические функции MS SQL Server приведены в табл. 13.6.

Таблица 13.6. *Математические функции SQL Server 2008*

Функция	Описание функции	Пример	Значение
`Abs()`	Возвращает абсолютную величину числа	`Abs(-345.6)`	345.6
`Acos()`	Возвращает угол в радианах, косинус которого равен указанному значению	`Acos(0.5)`	1.0471976
`Asin()`	Возвращает угол в радианах, синус которого равен указанному значению	`Asin(0.5)`	0.5235988
`Atan()`	Возвращает угол в радианах, тангенс которого равен указанному значению	`Atan(1)`	0.7853982
`Atn2()`	Возвращает угол в радианах, тангенс которого равен частному от деления первого аргумента на второй, с учетом квадранта, задаваемого двумя аргументами	`Atn2(3,3)`	0.7853982
`Ceiling()`	Возвращает ближайшее целое число, большее или равное значению аргумента	`Ceiling(-15.2)` `Ceiling (13.6)`	–15 14
`Cos()`	Возвращает косинус угла, заданного в радианах	`Cos(1)`	0.5403023
`Cot()`	Возвращает котангенс угла, заданного в радианах	`Cot(1)`	0.6420926
`Degrees()`	Выполняет преобразование угла из радиан в градусы	`Degrees(1.0)`	57.29578
`Exp()`	Возвращает значение экспоненты	`Exp(1)`	2.7182818

Таблица 13.6 *(окончание)*

Функция	Описание функции	Пример	Значение
`Floor()`	Округляет число до ближайшего минимального целого (см. примеры)	`Floor(-15.2)` `Floor(13.6)`	–16 13
`IsNumeric`	Проверяет, имеет ли указанное выражение числовой тип, если имеет, то функция возвращает 1. В противном случае — 0	`IsNumeric(1.2)`	1
`Log()`	Возвращает натуральный логарифм числа	`Log(10)`	2.302585
`Log10()`	Возвращает десятичный логарифм числа	`Log10(100)`	2.0
`Pi()`	Возвращает значение π	`Pi()`	3.1415926
`Power()`	Возводит число в степень	`Power(3,3)`	27
`Radians()`	Выполняет преобразование угла из градусов в радианы	`Radians(90.0)`	1.5707963
`Rand()`	Возвращает случайное число в диапазоне от 0 до 1	`Rand()`	0.8529751 (произвольное)

`Sign()`	Возвращает 1 для положительного числа, 0 для нулевого, –1 для отрицательного числа	`Sign(-10.1)`
		`Sign(0)`
		`Sign(10.1)`
			–1
			0
			1

`Sin()`	Возвращает синус угла, выраженного в радианах	`Sin(1)`	0.8414710
`Sqrt()`	Возвращает квадратный корень	`Sqrt(2)`	1.4142136
`Square()`	Выполняет возведение в квадрат	`Square(1.2)`	1.44
`Tan()`	Возвращает тангенс угла, заданного в радианах	`Tan(1)`	1.5574077

Другие полезные функции приведены в табл. 13.7.

Таблица 13.7. *Другие функции MS SQL Server 2008*

Функция	Описание функции	Пример	Значение
`GetDate()`	Возвращает текущую системную дату и время	`GetDate()`	12.03.2010 09:33:21.153
`Day()`	Возвращает день из значения даты	`Day(GetDate())`	12
`Month()`	Возвращает месяц из значения даты	`Month(GetDate))`	3

Таблица 13.7 *(окончание)*

Функция	Описание функции	Пример	Значение
`Char()`	Возвращает символ, соответствующий коду ASCII	`Char(37)`	% (процент)
`Year()`	Возвращает год из значения даты	`Year(GetDate())`	2010
`Len()`	Возвращает длину строки в символах	`Len('Сервер')`	6
`Lower()`	Переводит текст в нижний регистр	`Lower('aSD')`	asd
`Upper()`	Переводит текст в верхний регистр	`Upper('aSD')`	ASD
`Replicate()`	Выполняет тиражирование строки указанное число раз	`Replicate('Ado',2)`	AdoAdo
`Substring()`	Возвращает подстроку из строки. Необходимо указать, с какого символа и сколько символов	`Substring('abcdef',2,3)`	bcd
`Rtrim()`	Удаляет пробелы после текста	`Rtrim('abcd ')`	abcd
`Ltrim()`	Удаляет пробелы перед текстом	`Ltrim(' abcd')`	abcd
`Reverse()`	Возвращает строку, символы которой записаны в обратном порядке	`Reverse('РОЗА')`	АЗОР
`Str()`	Преобразует число в текст	`Str(1234.56)`	1234.56

13.9. Ключи и индексы

Индексы предназначены для того, чтобы обеспечить быстрый доступ к данным, хранящимся в базе. В MS SQL Server 2008 управление индексами организовано отдельно от таблиц. В отличие от MS Access, MS SQL Server может создавать кластерные индексы. Кластерный индекс требует дополнительного дискового пространства, но работает быстрее, т. к. его создание упорядочивает данные таблицы физически. Как правило, кластерный индекс создается для первичного ключа. Кластерный индекс может быть только один для таблицы, поскольку физически отсортировать таблицу по двум критериям невозможно.

13.9.1. Создание индекса

Создать индекс может только пользователь, являющийся владельцем таблицы. Алгоритм создания следующий:

1. Запустите утилиту SQL Server Management Studio.
2. Появится окно **Connect to Server** (Соединение с сервером). Выберите требуемый сервер, режим аутентификации, имя пользователя и пароль. Нажмите кнопку **Connect** (Подключиться).
3. В окне обозревателя объектов **Object Explorer** откройте объект **Databases** (Базы данных), а затем узел требуемой базы данных.
4. Раскройте узел **Tables** (Таблицы). Сделайте щелчок правой кнопкой мыши по пункту **Indexes**. Появится контекстное меню.
5. Выберите в нем первый пункт **New Index...** (Новый индекс). Появится одноименное окно **New Index**. В нем поле **Table name** (Имя таблицы) погашено,

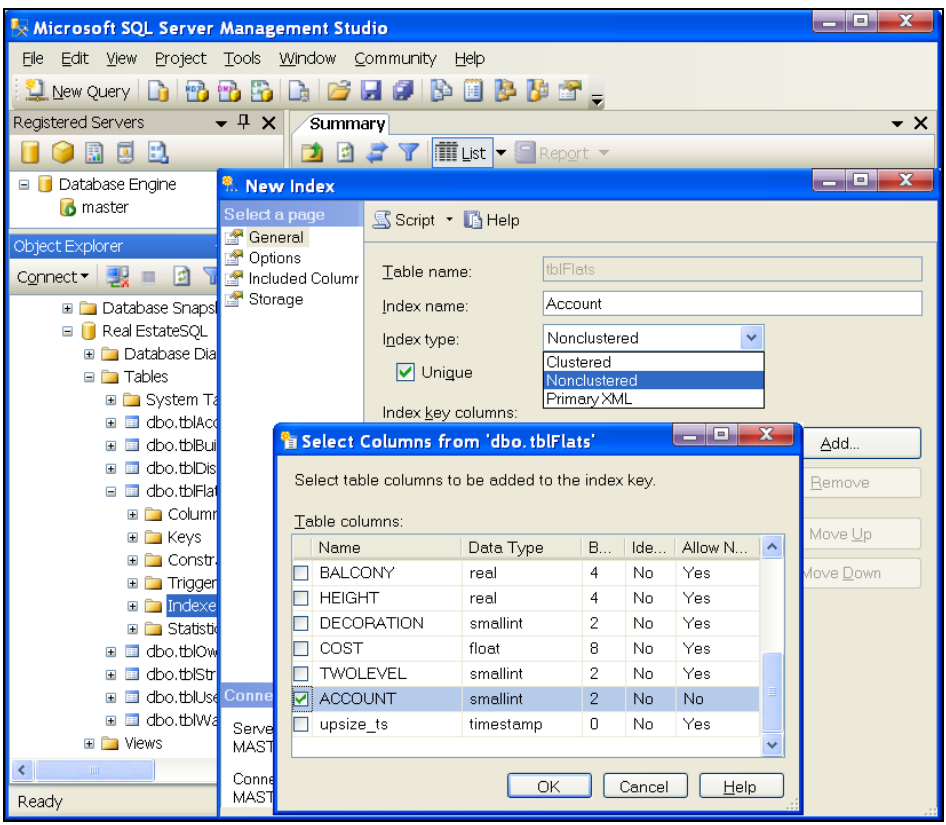


Рис. 13.6. Создание некластерного уникального индекса по полю `Account` для таблицы `tblFlats`

а надпись сообщает об имени таблицы, для которой создается индекс (рис. 13.6).

6. В поле **Index name** (Имя индекса) введите имя создаваемого индекса. Выберите его тип в раскрывающемся списке **Index type** (Тип индекса), также укажите, может ли индекс содержать повторяющиеся значения. Для этого предназначен флажок **Unique** (Уникальный).
7. Для выбора полей, которые будут включены в индекс, щелкните кнопку **Add** (Добавить). Появится окно **Select Columns from...**.
8. Отметьте флажком поля, которые следует включить в индекс, и нажмите кнопку **OK**.
9. Окно **Select Columns from...** закроется, а выбранные поля появятся в нижней части окна **New Index** (Новый индекс).
10. Сделайте щелчок по колонке порядка сортировки **Sort Order**. Появится поле со списком. Выберите в нем **Ascending** (По возрастанию) или **Descending** (По убыванию). Щелкните по кнопке **OK**.

13.9.2. Работа с индексами

В процессе эксплуатации базы данных или ее доработки вам непременно понадобится изменить свойства индекса, переименовать его или удалить. Сделать это можно следующим образом (рис. 13.7):

1. Запустите утилиту SQL Server Management Studio.
2. Появится окно соединения с сервером **Connect to Server**. Выберите требуемый сервер, режим аутентификации, имя пользователя и пароль. Нажмите кнопку **Connect** (Подключиться).
3. В окне обозревателя объектов **Object Explorer** откройте объект **Databases** (Базы данных), а затем узел требуемой базы данных.
4. Раскройте узел **Tables** (Таблицы). Выберите требуемую таблицу и сделайте по ее имени щелчок правой кнопкой мыши.
5. Появится контекстное меню. Выберите в нем пункт **Modify** (Изменить). Откроется окно конструктора таблицы.
6. Щелкните по значку **Manage Indexes and Keys**. Он расположен на панели инструментов. Появится окно **Indexes/Keys**. В левой части окна приведен список индексов, а в правой — их свойства.
7. Для изменения полей, включенных в индекс, перейдите в раздел **General** (Общие) и щелкните по строчке **Columns** (Столбцы). В правой части этой строки появится кнопка открытия окна **Index Columns.**
8. В нижней части окна **Indexes/Keys** имеются кнопки **Add** (Добавить) и **Delete** (Удалить) для добавления и удаления индексов.

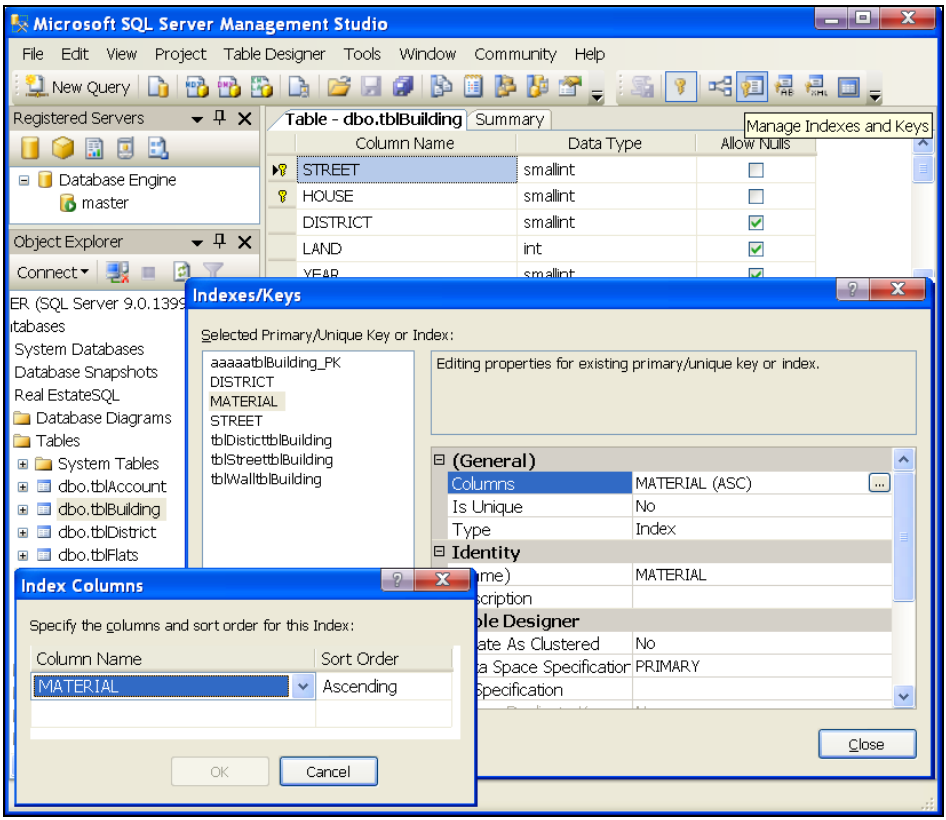


Рис. 13.7. Работа с индексами таблицы `tblBuilding`

13.9.3. Создание первичного ключа таблицы

В качестве первичного ключа можно указать поле таблицы или группу полей, однозначно определяющих положение любой записи. Первичный ключ для каждой таблицы может быть только один. Столбец или столбцы, входящие в первичный ключ, не могут содержать значение `NULL`. При создании первичного ключа MS SQL Server 2008 автоматически создаст уникальный индекс (Unique) для всех столбцов первичного ключа. По умолчанию будет создан кластерный индекс. Порядок создания первичного ключа следующий:

1. Запустите утилиту SQL Server Management Studio.
2. Появится окно **Connect to Server**. Выберите требуемый сервер, режим аутентификации, имя пользователя и пароль. Нажмите кнопку **Connect** (Подключиться).

3. В окне обозревателя объектов **Object Explorer** откройте объект **Databases** (Базы данных), а затем узел требуемой базы данных.
4. Раскройте узел **Tables** (Таблицы). Выберите требуемую таблицу и сделайте по ее имени щелчок правой кнопкой мыши.
5. Появится контекстное меню. Выберите в нем пункт **Modify** (Модифицировать). Откроется окно конструктора таблицы.
6. В третьей колонке конструктора **Allow Nulls** (Разрешить пустые значения) снимите флажки у полей, которые войдут в состав первичного ключа.
7. Сделайте щелчок по полю, на основе значений которого будет создан первичный ключ. Вся строчка будет выделена темным цветом.
8. Если в состав ключа необходимо включить несколько полей, то, удерживая нажатой клавишу <Ctrl>, щелкните мышью на затененном поле слева от имени столбца.
9. Сделайте щелчок мышью по кнопке **Set Primary Key** (Назначить первичным ключом), расположенной на панели инструментов. Эта кнопка имеет стилизованное изображение ключа. Именно этот значок и появится слева от названий полей, вошедших в ключ.

13.10. Создание ограничений для столбцов таблицы

Применение ограничений на значение поля таблицы дает возможность контролировать правильность заносимой информации в базу данных. Это ограничение позволяет проверить корректность данных только в одном поле. Рассмотрим пример создания ограничения на значение номера района, который не может находиться вне диапазона от 1 до 9 включительно.

Чтобы добавить условие на значение поля `District` таблицы `tblDistrict`, выполните следующие действия:

1. Запустите SQL Server Management Studio.
2. Появится окно соединения с сервером **Connect to Server**. Выберите требуемый сервер, режим аутентификации, имя пользователя и пароль. Нажмите кнопку **Connect** (Подключиться).
3. В окне обозревателя объектов **Object Explorer** откройте объект **Databases** (Базы данных), а затем узел требуемой базы данных.
4. Раскройте узел **Tables** (Таблицы). Выберите требуемую таблицу и раскройте ее узел.
5. Сделайте щелчок правой кнопки мыши по строке ограничений **Constraint**. Появится контекстное меню. Выберите в нем пункт **New Constraint...** (Новое ограничение). Появится диалоговое окно **Check Constraints** (рис. 13.8).

6. В строке **Name** (Имя) укажите имя ограничения. На него будет ссылаться система в сообщениях о нарушениях ограничения в процессе работы программного комплекса.
7. Сделайте щелчок мышью по второй части строки **Expression** (Выражение). Появится кнопка с изображением многоточия. Нажмите ее. Откроется окно **Check Constraint Expression**.
8. Введите текст ограничения и щелкните кнопку **OK**.
9. В строке описания **Description** укажите предназначение этого ограничения.
10. Сделайте щелчок по кнопке **Close** (Закрыть).

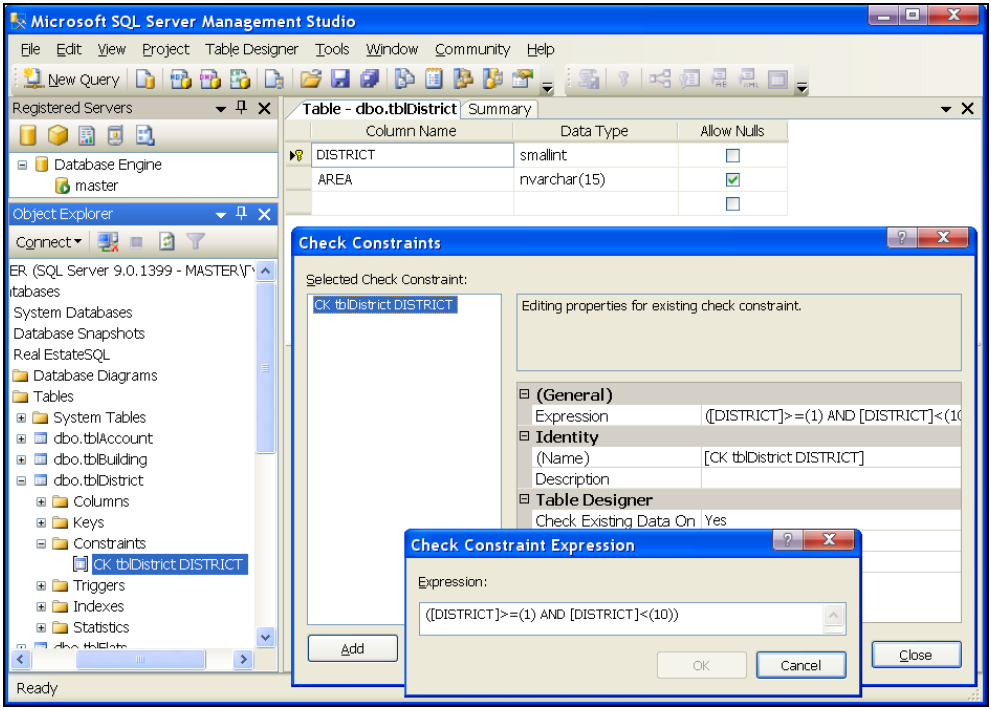


Рис. 13.8. Создание ограничения на значение поля таблицы `tblDistrict`

Если в процессе работы программного комплекса пользователь укажет номер района, выходящий за пределы диапазона (1—9), то на экране его дисплея появится сообщение (рис. 13.9).

Для рядового пользователя оно, конечно же, ни о чем не говорит. Поэтому вместо ограничений `CHECK` разработчики чаще применяют триггеры, которые легко научить разговаривать по-русски. О триггерах речь пойдет в *разд. 13.14*.

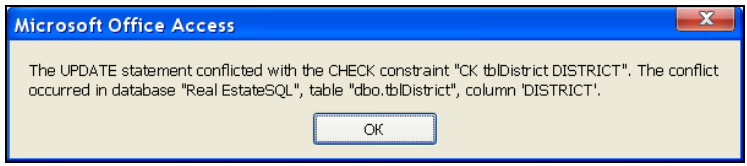


Рис. 13.9. Сообщение о неправильном значении столбца

13.11. Создание отношений между таблицами

Для создания связей между таблицами используются первичные и внешние ключи. Первичный ключ однозначно определяет положение любой записи и создается в главной таблице. В качестве первичного ключа используется один или несколько ее столбцов. Внешний ключ создается в подчиненной таблице, которая ссылается на данные главной таблицы. В качестве внешнего ключа используется один или несколько столбцов подчиненной таблицы. При изменении значения первичного ключа (главная таблица) возможны следующие варианты изменения внешнего ключа (подчиненная таблица):

- автоматическое каскадное изменение всех соответствующих значений;
- запрет на изменение значений первичного ключа, если в подчиненной таблице имеется хотя бы одна запись, связанная с изменяемой записью в главной таблице;
- изменение значений полей первичного ключа независимо от существования связанных записей в подчиненной таблице.

13.11.1. Создание связи "один-ко-многим"

Создадим связь между таблицами `tblDictrict` (районы) и `tblBuilding` (здания). Изучение предметной области разрабатываемого приложения показывает, что это отношение "один-ко-многим", т. к. в одном районе города может быть несколько зданий. В этой связи участвует простой первичный ключ, в состав которого входит одно поле. Алгоритм создания связи следующий:

1. Запустите SQL Server Management Studio.
2. Появится окно подключения к серверу **Connect to Server**. Выберите требуемый сервер, режим аутентификации, имя пользователя и пароль. Нажмите кнопку **Connect** (Подключиться).
3. В окне обозревателя объектов **Object Explorer** откройте объект **Databases** (Базы данных), а затем узел требуемой базы данных.
4. Раскройте узел **Database Diagrams** (Диаграмма базы данных). Он должен содержать диаграмму базы данных. Мы ее построили в начале этой главы. Если

диаграмма отсутствует — постройте ее. Создать связь между таблицами можно и без диаграммы, но этот путь менее нагляден и более трудоемок.

5. Сделайте двойной щелчок мышью по имени диаграммы. Она откроется в отдельном окне.
6. Если таблиц `tblDictrict` и `tblBuilding` в схеме нет, то добавьте их.
7. Поместите указатель мыши над изображением ключа в таблице `tblDictrict`. Ключ изображен слева от поля `DISTRICT`. Именно оно входит в состав первичного ключа. Нажмите левую кнопку мыши и не отпускайте ее полсекунды (по умолчанию — время срабатывания двойного щелчка `DblClick`). По истечении этого времени рядом с курсором появится крестик. Первичный ключ выбран.
8. Перетащите появившийся крестик на затененный квадрат, расположенный слева от поля `DISTRICT` таблицы `tblBuilding`, и сделайте щелчок левой кнопкой мыши. Откроются два окна: **Foreign Key Relationship** и **Tables and Columns** (рис. 13.10).

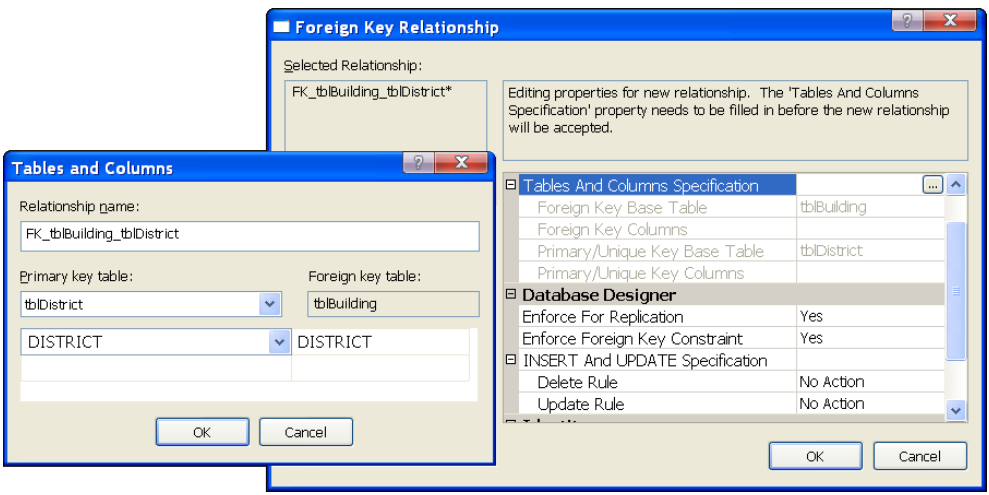


Рис. 13.10. Создание связи "один-ко-многим" между таблицами `tblDictrict` и `tblBuilding` с использованием простого первичного ключа

9. В окне **Tables and Columns** в поле **Relationship name** система сгенерировала имя внешнего ключа таблицы `tblBuilding`. Оно состоит из трех частей, разделенных символом подчеркивания: FK_tblBuilding_tblDistrict. Первые два символа "FK" означают **Foreign Key** (Внешний ключ), второе имя `tblBuilding` указывает подчиненную таблицу, а третье `tblDictrict` — главную таблицу в созданной связи.
10. Если все действия по "перетаскиванию" выполнены правильно, то никаких изменений в этом окне делать не требуется. Щелкните кнопку **OK** для закрытия окна **Tables and Columns**.

11. Активным станет первое окно **Foreign Key Relationship**. В нем уже установлен параметр **Enforce Foreign Key Constraint** в значение **Yes** — введены ограничения внешнего ключа. Вы можете ввести дополнительные параметры удаления и изменения записей. Для этого предназначены строчки **Delete Rule** (Удалить правило) и **Update Rule** (Изменить правило). Если в качестве значений этих параметров поставить **Cascade** (Каскад), то изменение и удаление записей в главной таблице `tblDictrict` автоматически приведут к изменению и удалению соответствующих записей в подчиненной таблице `tblBuilding`.

12. Если вы предпочитаете обеспечивать ссылочную целостность при помощи триггеров, то поставьте в окне **Foreign Key Relationship**:

    - **Enforce Foreign Key Constraint** в значение **Yes**;
    - **Delete Rule** в значение **No Action**;
    - **Update Rule** в значение **No Action**.

    И займитесь их написанием.

Примечание

Если ссылочная целостность обеспечивается одновременно как декларативно, так и с помощью триггеров, то в первую очередь сработает DRI, а до выполнения триггера дело не дойдет. Клиент получит системное сообщение на английском языке, если оно предусмотрено, а не сообщение триггера.

Рассмотрим создание связи между таблицами `tblFlats` (квартиры) и `tblOwners` (проживающие). В состав первичного ключа главной таблицы `tblFlats` входят три поля: `STREET`, `HOUSE` и `FLAT`. Внешний ключ подчиненной таблицы `tblOwners` также будет составным. Для построения связи между этими таблицами сделайте следующее:

1. Запустите утилиту SQL Server Management Studio.
2. Появится окно **Connect to Server**. Выберите требуемый сервер, режим аутентификации, имя пользователя и пароль. Нажмите кнопку **Connect** (Подключиться).
3. В окне обозревателя объектов **Object Explorer** откройте объект **Databases** (Базы данных), а затем узел требуемой базы данных.
4. Раскройте узел **Database Diagrams** (Диаграммы базы данных). Он должен содержать диаграмму базы данных. Мы ее построили в начале этой главы. Если диаграмма отсутствует — постройте ее. Создать связь между таблицами можно и без диаграммы, но этот путь менее нагляден и более трудоемок.
5. Сделайте двойной щелчок мышью по имени диаграммы. Она откроется в отдельном окне.
6. Если таблиц `tblFlats` и `tblOwners` в схеме нет, то добавьте их.

7. Поместите указатель мыши над изображением ключа, расположенного рядом с полем `STREET` в главной таблице `tblFlats`. Сделайте щелчок левой кнопкой мыши. Нажмите клавишу <Ctrl> и, не отпуская ее, сделайте щелчок по полю `HOUSE`, а затем `FLAT`. Отпустите клавишу <Ctrl>. Будет выделена группа из трех полей. Первичный ключ выбран.
8. Поместите указатель мыши над этой группой, нажмите левую кнопку мыши и, не отпуская ее, перетащите эту группу на затененный квадрат, расположенный слева от поля `STREET` таблицы `tblOwners`. Отпустите левую кнопку мыши. Откроются два окна: **Foreign Key Relationship** и **Tables and Columns** (рис. 13.11).

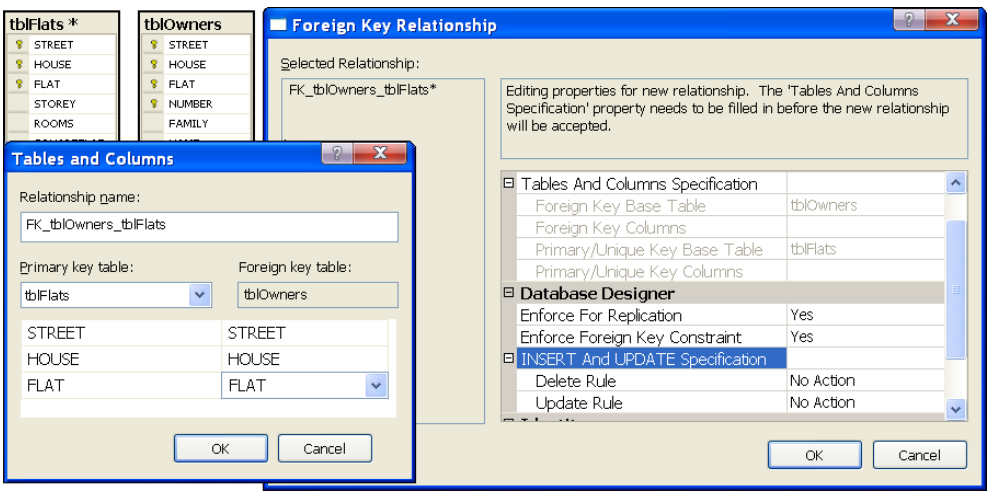


Рис. 13.11. Создание связи "один-ко-многим" между таблицами `tblFlats` и `tblOwners` с использованием составного первичного ключа

9. В окне **Tables and Columns** (Таблицы и столбцы) в поле **Relationship name** система сгенерировала имя внешнего ключа таблицы `tblOwners`. Оно состоит из трех частей, разделенных символом подчеркивания: `FK_tblOwners_tblFlats`. Первые два символа "FK" означают *Foreign Key* (Внешний ключ), второе имя `tblOwners` указывает подчиненную таблицу, а третье `tblFlats` — главную таблицу в созданной связи.
10. Если все действия по "перетаскиванию" выполнены правильно, то в окне **Tables and Columns** в разделе **Primary Key Table** (Таблица первичного ключа) будут указаны таблица `tblFlats` и три ее поля в качестве первичного ключа: `STREET+HOUSE+FLAT`, в разделе **Foreign Key Table** (Таблица внешнего ключа) — таблица `tblOwners` и только одно поле `STREET` в качестве внешнего ключа.

11. Восстановите статус-кво. Добавьте в состав внешнего ключа таблицы `tblOwners` еще два поля: `HOUSE` и `FLAT`. Щелкните кнопку **OK** для закрытия окна **Tables and Columns**.
12. Активным станет первое окно **Foreign Key Relationship**. В нем уже установлен параметр **Enforce Foreign Key Constraint** в значение **Yes** — введены ограничения внешнего ключа. Вы можете ввести дополнительные параметры удаления и изменения записей. Для этого предназначены строчки **Delete Rule** и **Update Rule**. Если в качестве значений этих параметров поставить **Cascade** (Каскад), то изменение и удаление записей в главной таблице `tblFlats` автоматически приведут к изменению и удалению соответствующих записей в подчиненной таблице `tblOwners`.

13.11.2. Создание связи "один-к-одному"

Такая связь должна быть установлена между таблицами `tblFlats` (квартиры) и `tblAccount` (лицевые счета). Каждая квартира может иметь только один лицевой счет или не иметь его вовсе, в то же время один лицевой счет принадлежит только одной квартире. Для построения связи выполните следующие действия:

1. Запустите утилиту SQL Server Management Studio.
2. Появится окно **Connect to Server** (Подключение к серверу). Выберите требуемый сервер, режим аутентификации, имя пользователя и пароль. Нажмите кнопку **Connect** (Подключиться).
3. В окне обозревателя объектов **Object Explorer** откройте объект **Databases** (Базы данных), а затем узел требуемой базы данных.
4. Раскройте узел **Tables** (Таблицы). Выберите таблицу `tblFlats`. Сделайте щелчок правой кнопкой мыши по пункту **Indexes** (Индексы). Появится контекстное меню.
5. Выберите в нем первый пункт **New Index...** (Новый индекс). Появится одноименное окно **New Index**. В нем поле **Table name** погашено, а надпись сообщает об имени таблицы, для которой создается индекс: `tblFlats`.
6. В поле **Index Name** введите имя создаваемого индекса: `ACCOUNT`.
7. Поставьте флажок **Unique**.
8. Для выбора поля `ACCOUNT`, которое будет включено в индекс, нажмите кнопку **Add**. Появится окно **Select Columns from...**. Сделайте выбор и щелкните по кнопке **OK**.
9. Окно **Select Columns from...** закроется, а выбранное поле `ACCOUNT` появится в нижней части окна **New Index**.
10. Сделайте щелчок по колонке порядка сортировки **Sort Order**. Появится раскрывающийся список. Выберите в нем **Ascending** (По возрастанию). Щелкните по кнопке **OK**.

11. Раскройте узел **Database Diagrams**. Он должен содержать диаграмму базы данных. Мы ее сформировали в начале этой главы. Если диаграмма отсутствует — постройте ее. Создать связь между таблицами можно и без диаграммы, но этот путь менее нагляден и более трудоемок.
12. Сделайте двойной щелчок мышью по имени диаграммы. Она откроется в отдельном окне.
13. Если таблиц `tblAccount` и `tblFlats` в схеме нет, то добавьте их.
14. Поместите указатель мыши над изображением ключа в таблице `tblAccount`. Ключ изображен слева от поля `ACCOUNT`. Именно оно входит в состав первичного ключа. Нажмите левую кнопку мыши и не отпускайте ее полсекунды (по умолчанию — время срабатывания двойного щелчка `DblClick`). По истечении этого времени рядом с курсором появится крестик. Первичный ключ выбран.
15. Перетащите появившийся крестик на затененный квадрат, расположенный слева от поля `ACCOUNT` таблицы `tblFlats`, и сделайте щелчок левой кнопкой мыши. Откроются два окна: **Foreign Key Relationship** и **Tables and Columns**.
16. В окне **Tables and Columns** в поле **Relationship name** система сгенерировала имя внешнего ключа таблицы `tblFlats`. Оно состоит из трех частей, разделенных символом подчеркивания: `FK_tblFlats_tblAccount`. Первые два символа "FK" означают *Foreign Key* (Внешний ключ), второе имя `tblFlats` указывает подчиненную таблицу, а третье `tblAccount` — главную таблицу в созданной связи.
17. Если все действия по "перетаскиванию" выполнены правильно, то никаких изменений в этом окне делать не требуется. Щелкните по кнопке **OK** для закрытия окна **Tables and Columns**.
18. Активным станет первое окно **Foreign Key Relationship**. В нем уже установлен параметр **Enforce Foreign Key Constraint** в значение **Yes** — введены ограничения внешнего ключа. Вы можете ввести дополнительные параметры удаления и изменения записей. Для этого предназначены строчки **Delete Rule** и **Update Rule**. Если в качестве значений этих параметров поставить **Cascade**, то изменение и удаление записей в главной таблице `tblAccount` автоматически приведут к изменению и удалению соответствующих записей в подчиненной таблице `tblFlats`.

13.12. Представления

Представление (вид) предназначено для объединения данных из нескольких связанных между собой таблиц. Для конечного пользователя оно выглядит как таблица или окно, отображающее информацию. Физически представление — это набор инструкций для SQL-сервера, на основе которых он производит выборку.

Другими словами, представление — это виртуальная таблица, содержащая столбцы и строки с данными, выбранными динамически из одной или нескольких таблиц и представленными пользователю как самостоятельные.

В MS Access представления называются *запросами*. Все, что вам известно о запросах, справедливо и для представлений. С представлением можно работать так же, как и с таблицей. Часто представления применяют для того, чтобы "спрятать" от рядового пользователя некоторые данные. Столбцы и строки определенных таблиц просто не включают в состав операторов `SELECT`, формирующих представление.

Все изменения, выполненные с данными представления, будут автоматически внесены в таблицы, на основе которых это представление создано, но при выполнении следующих условий:

- представление не должно иметь вычисляемых столбцов;
- представление должно содержать минимум одну таблицу в параметре `FROM`;
- команды `UPDATE` или `INSERT`, выполняющие действия над данными представления, должны изменять только столбцы, принадлежащие одной таблице;
- при создании представления не используются параметры `GROUP BY`, `DISTINCT`, `UNION` и `TOP`, а также функции агрегирования (`SUM`, `COUNT`, `MAX` и др.).

В MS SQL Server 2008 представления очень удобно создавать при помощи конструктора, входящего в состав Microsoft SQL Server Management Studio. Рассмотрим процесс создания представления на конкретном примере.

13.12.1. Пример

На вашем предприятии создан отдел мониторинга за зданиями только Кировского района, в котором процесс создания ТСЖ получил наибольшее распространение. Одна из задач этого отдела — контроль межевых дел земельных участков и процесса проведения капитальных ремонтов зданий. Другими словами, работник отдела мониторинга должен иметь дело с ограниченным набором полей базы данных Real EstateSQL. Для работы ему нужны только внешние параметры объектов недвижимости, такие как адрес, площадь земельного участка, год постройки, износ и материал стен.

Для решения этой задачи очень подходит представление MS SQL Server. Для его создания выполните следующие действия:

1. Запустите SQL Server Management Studio.
2. Появится окно **Connect to Server** (Подключение к серверу). Выберите требуемый сервер, режим аутентификации, имя пользователя и пароль. Нажмите кнопку **Connect** (Подключиться).
3. В окне обозревателя объектов **Object Explorer** откройте объект **Databases** (Базы данных), а затем узел требуемой базы данных.

4. Сделайте щелчок правой кнопкой мыши по узлу **Views**. Появится контекстное меню. Выберите в нем пункт **New View...**.

5. Откроются окно конструктора представлений и окно добавления новых объектов в создаваемое представление **Add Table**. В этом окне четыре вкладки: **Tables**, **Views**, **Functions** и **Synonyms**.

6. В состав текста нашего представления на Transact-SQL войдут только таблицы: `tblBuilding`, `tblStreet`, `tblDistrict` и `tblWall`. Выберите их по очереди из списка первой вкладки и нажмите кнопку **Add**.

7. Закройте окно **Add Table**. Выбранные таблицы со связями между ними появятся в верхней части основного окна конструктора представлений (рис. 13.12).

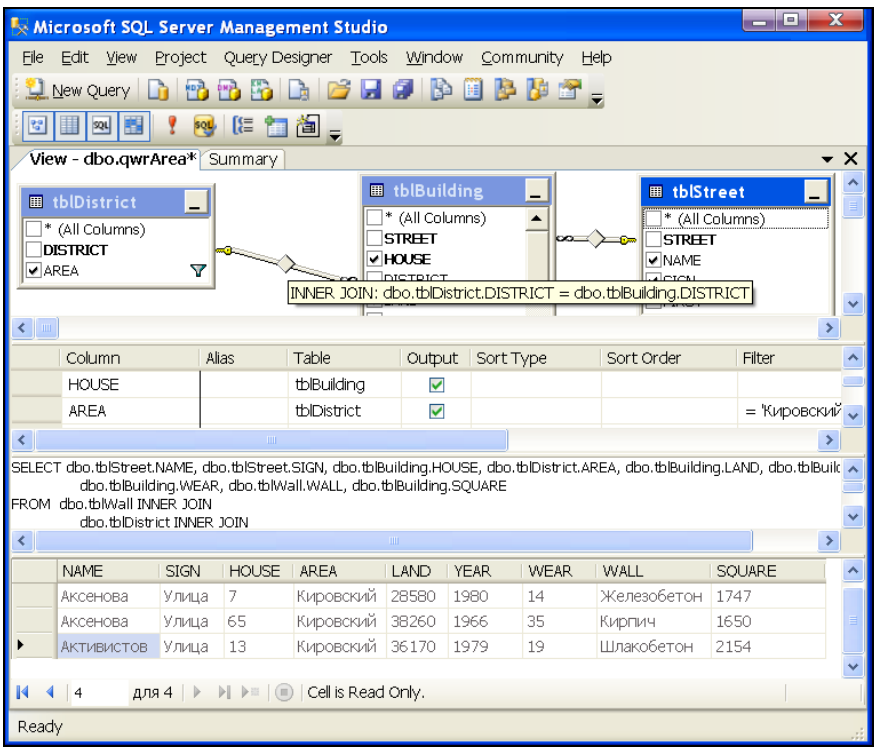


Рис. 13.12. Создание нового представления (View) с использованием Microsoft SQL Server Management Studio

8. Во второй части окна представлен интерфейс для выбора состава полей создаваемого представления. Третья колонка **Table** предназначена для выбора таблицы, а первая **Column** — для выбора поля из этой таблицы.

9. Определившись с составом полей, которые будут включены в текст представления, перейдите к строке AREA и в колонке **Filter** введите ='Кировский'.
10. В третьей части окна конструктор представлений все ваши действия превращает в текст на языке Transact-SQL. Он представлен в листинге 13.2.

Листинг 13.2. Код представления AREA на Transact-SQL

```
SELECT dbo.tblStreet.NAME,dbo.tblStreet.SIGN,
       dbo.tblBuilding.HOUSE,dbo.tblDistrict.AREA,
       dbo.tblBuilding.LAND,dbo.tblBuilding.YEAR,
       dbo.tblBuilding.WEAR,dbo.tblWall.WALL,
       dbo.tblBuilding.SQUARE
   FROM dbo.tblWall INNER JOIN
        dbo.tblDistrict INNER JOIN
        dbo.tblStreet INNER JOIN
        dbo.tblBuilding ON
        dbo.tblStreet.STREET = dbo.tblBuilding.STREET ON
        dbo.tblDistrict.DISTRICT = dbo.tblBuilding.DISTRICT
        ON dbo.tblWall.MATERIAL = dbo.tblBuilding.MATERIAL
   WHERE (dbo.tblDistrict.AREA = 'Кировский')
```

Не выходя из конструктора представлений, можно сразу запустить созданный текст на выполнение. Для этого сделайте щелчок мышью по значку с изображением восклицательного знака на панели инструментов. В окне конструктора появится его четвертая составляющая с результатами работы.

13.13. Хранимые процедуры

Хранимая процедура представляет собой набор команд Transact-SQL, который хранится непосредственно на сервере и представляет собой самостоятельный объект. Преимущество хранимых процедур заключается в том, что их выполняет сервер, и именно они — суть клиент-серверной архитектуры. Сервер — это всегда один из самых мощных компьютеров сети. Время выполнения на нем кода существенно меньше, чем на рабочей станции. К тому же сами данные расположены на том же компьютере, который выполняет обработку. Затраты времени на передачу по сети минимальны.

Хранимая процедура может состоять из значительного количества строк кода, но для ее выполнения необходимо указать серверу лишь ее имя и параметры, если они есть. База данных у сервера всегда под "рукой", а возвращать клиенту тоже требуется немного, только то, что он просил.

Хранимые процедуры бывают системными, пользовательскими и временными. Все действия по администрированию MS SQL Server 2008 выполняются в основ-

ном с использованием системных процедур. Вы также можете создать собственную системную хранимую процедуру, поместив ее в системную базу данных, и она будет "видна" для любой базы данных этого сервера. Пользовательская хранимая процедура представляет собой объект базы данных. В ней она и хранится. Непосредственный вызов пользовательской хранимой процедуры возможен только в контексте базы данных.

Для создания хранимой процедуры предназначена специальная конструкция Transact-SQL. Синтаксис ее имеет следующий вид:

```
CREATE { PROC | PROCEDURE } [schema_name.] procedure_name
 [ { @parameter [ type_schema_name. ] data_type }
 [ VARYING ] [ = default ] [ OUTPUT ]
 ] [ ,...n ]
 [ WITH
 { ENCRYPTION | RECOMPILE | EXECUTE AS Clause}]
 [ FOR REPLICATION ]
AS { <sql_statement> [;][ ...n ] | <method_specifier> }
[;]
```

В состав конструкции входят следующие аргументы:

- *schema_name* — имя схемы, которой принадлежит процедура;
- *procedure_name* — имя создаваемой процедуры. Имя системной процедуры должно начинаться с символов (sp_), а имя временной процедуры с двух знаков (##);
- *@parameter* — имя параметра хранимой процедуры, который будет использоваться для передачи данных. Параметры разделяются запятыми. Максимальное число параметров — 2100;
- *type_schema_name* — имя схемы, которой принадлежит тип данных;
- *data_type* — тип данных параметра;
- VARYING — используется совместно с OUTPUT и только с типом данных cursor;
- *default* — значение параметра по умолчанию;
- ENCRYPTION — код хранимой процедуры будет зашифрован;
- RECOMPILE — план выполнения хранимой процедуры при каждом ее выполнении будет создаваться заново;
- FOR REPLICATION — применяется при репликации данных;
- AS — начало тела хранимой процедуры;
- *sql_statement* — команды тела хранимой процедуры. Если команд несколько, то они заключаются в операторные скобки BEGIN...END.

Создадим хранимую процедуру, которая возвращает фамилии, имена и отчества всех проживающих в указанном здании. Здание идентифицируется двумя пара-

метрами: номером улицы и номером дома. Для ее создания выполните следующие действия:

1. Запустите SQL Server Management Studio.
2. Появится окно **Connect to Server** (Подключение к серверу). Выберите требуемый сервер, режим аутентификации, имя пользователя и пароль. Нажмите кнопку **Connect** (Подключиться).
3. В окне обозревателя объектов **Object Explorer** откройте объект **Databases**, а затем узел требуемой базы данных.
4. Откройте объект **Programmability**.
5. Сделайте щелчок правой кнопкой мыши по узлу **Stored Procedures** (Хранимые процедуры). Появится контекстное меню. Выберите в нем пункт **New Stored Procedure...** (Новая хранимая процедура). Появится окно с заготовкой текста хранимой процедуры.
6. Для облегчения ввода значений параметров предложенного шаблона нажмите комбинацию клавиш <Ctrl>+<Shift>+<M>. Появится окно **Specify Values for Template Parameters**. Все значения, введенные в этом окне, будут автоматически вставлены в код.
7. Введите код, приведенный в листинге 13.3.
8. Нажмите кнопку **! Execute** (Выполнить) на панели инструментов. Система проверит синтаксис хранимой процедуры и, если ошибок нет, запишет ее в базу данных.
9. Сделайте щелчок правой кнопкой мыши по узлу **Stored Procedures**. Появится контекстное меню. Выберите в нем пункт **Refresh** (Обновить).
10. Имя хранимой процедуры появится в окне **Object Explorer** в разделе **Stored Procedures**.

Листинг 13.3. Текст хранимой процедуры `Citizens`

```
set ANSI_NULLS ON
set QUOTED_IDENTIFIER ON
go
-- =============================================
-- Author:      Гурвиц
-- Create date: 01.05.2010
-- Description: Список проживающих в здании
-- =============================================
CREATE PROCEDURE [dbo].[Citizens]
       -- Add the parameters for the stored procedure here
       @StreetSQL smallint,
       @HouseSQL smallint
```

```
AS
BEGIN
   SELECT FAMILY,NAME,SECOND
          FROM dbo.tblOwners
          WHERE STREET=@StreetSQL AND HOUSE=@HouseSQL
END
```

Для проверки правильности работы хранимой процедуры `Citizens` сделайте щелчок мышью по кнопке **New Query**, расположенной на панели инструментов. Появится окно. Наберите в нем текст:

```
EXECUTE Citizens @StreetSQL=2, @HouseSQL=10
```

В нем `EXECUTE` — служебное слово, `@StreetSQL=2` — номер улицы, на которой расположено здание, `@HouseSQL=10` — номер дома. После щелчка по кнопке **! Execute** на экране появится окно с двумя вкладками. Первая вкладка **Results** содержит текст:

```
Борисенок       Владимир      Викторович
Борисенок       Егор          Владимирович
Верхотуров      Сергей        Викторович
Верхотурова     Ольга         Борисовна
Верхотуров      Александр     Сергеевич
Бляшкин         Александр     Иванович
Бляшкина        Зофия         Францевна
...
```

Вторая вкладка **Messages** сообщает, что в выборку попало 143 фамилии:

```
(143 row(s) affected)
```

13.14. Триггеры

В Transact-SQL можно создавать хранимые процедуры специального типа, которые запускаются на выполнение автоматически при осуществлении модификаций в таблице (удаление, добавление и корректировка записей). Эти процедуры получили название *триггеров*. Триггер всегда связан с конкретной таблицей. В триггере допускается использование большинства команд Transact-SQL, но существуют ограничения на некоторые операции. Например, создание и удаление таблиц, индексов, хранимых процедур, представлений и т. д. Для создания триггера используется конструкция:

```
CREATE TRIGGER [ schema_name. ]trigger_name
ON { table | view }
[ WITH [ ENCRYPTION ][ EXECUTE AS Clause ][ ,...n ] ]
{ FOR | AFTER | INSTEAD OF }
{ [ INSERT ] [ , ] [ UPDATE ] [ , ] [ DELETE ] }
```

```
[ WITH APPEND ]
[ NOT FOR REPLICATION ]
AS { <sql_statement> [;][ ...n ] | <method_specifier> }
[;]
```

В состав конструкции входят следующие аргументы:

- *schema_name* — имя схемы, которой принадлежит триггер;
- *trigger_name* — имя создаваемого триггера;
- {*table | view*} — имя таблицы или представления, для которых создается триггер;
- ENCRYPTION — код триггера будет зашифрован;
- EXECUTE AS — определяет уровень безопасности;
- FOR — ключевое слово, после которого указывается тип триггера;
- AFTER — будет создан стандартный триггер (по умолчанию);
- INSTEAD OF — триггер будет запускаться взамен команды, которая привела к его запуску;
- INSERT, UPDATE, DELETE — будет создан триггер на вставку, модификацию и удаление соответственно;
- WITH APPEND — применяется для обеспечения совместимости с ранними версиями MS SQL Server;
- NOT FOR REPLICATION — не вызывать триггер, если модификация данных выполняется средствами подсистемы репликации;
- AS — начало тела триггера;
- *sql_statement* — команды тела триггера. Если команд несколько, то они заключаются в операторные скобки BEGIN...END.

Создадим триггер, запускаемый на выполнение при изменении данных в таблице лицевых счетов tblAccount. Он должен обеспечить выполнение следующих действий:

- обновление номера лицевого счета в таблице квартир tblFlats (поле ACCOUNT) при изменении номера лицевого счета в таблице, для которой триггер создан (tblAccount);
- занесение данных о работнике, выполнившем последние изменения в таблице tblAccount. Эти данные должны содержать имя пользователя, дату и время корректировки, имя компьютера, с которого эта корректировка выполнена, и имя приложения, инициирующего корректировку (табл. 13.8).

***Таблица 13.8.** Дополнительные поля, которые должна иметь таблица `tblAccount` для хранения данных о работнике, выполнившем корректировку*

Название	Тип данных	Описание
`UserName`	`nvarchar(20)`	Учетная запись MS SQL Server 2008
`[DateTime]`	`datetime`	Дата и время последней корректировки
`HostName`	`nvarchar(20)`	Имя компьютера, с которого эта корректировка была выполнена
`ClientName`	`nvarchar(128)`	Имя приложения, выполнившего корректировку

Для создания триггера выполните следующие действия:

1. Запустите SQL Server Management Studio.
2. Появится окно **Connect to Server** (Подключение к серверу). Выберите требуемый сервер, режим аутентификации, имя пользователя и пароль. Нажмите кнопку **Connect** (Подключиться).
3. В окне обозревателя объектов **Object Explorer** откройте объект **Databases** (Базы данных), а затем узел требуемой базы данных.
4. Откройте узел **Tables** (Таблицы). Найдите в нем требуемую таблицу `tblAccount` и раскройте ее узел.
5. Сделайте щелчок правой кнопкой мыши по пункту **Triggers** (Триггеры). Появится контекстное меню. Выберите в нем строку **New Trigger...** (Новый триггер). Появится окно с заготовкой текста триггера.
6. Для облегчения ввода значений параметров предложенного шаблона нажмите комбинацию клавиш <Ctrl>+<Shift>+<M>. Появится окно **Specify Values for Template Parameters**. Все значения, введенные в этом окне, будут автоматически вставлены в код.
7. Введите код из листинга 13.4.
8. Нажмите кнопку **! Execute** (Выполнить) на панели инструментов. Система проверит синтаксис триггера и, если ошибок нет, запишет его в базу данных.
9. Сделайте щелчок правой кнопкой мыши по узлу **Triggers**. Появится контекстное меню. Выберите в нем пункт **Refresh** (Обновить).
10. Имя триггера появится в окне **Object Explorer** в разделе **Triggers**.

Листинг 13.4. Текст триггера `T_tblAccount_UTrig`

```
CREATE TRIGGER T_tblAccount_UTrig
ON dbo.tblAccount FOR UPDATE AS
SET NOCOUNT ON
/* * КАСКАДНЫЕ ОБНОВЛЕНИЯ В 'tblFlats' */
IF UPDATE(ACCOUNT)
```

```
    BEGIN
       UPDATE tblFlats
       SET tblFlats.ACCOUNT = inserted.ACCOUNT
       FROM tblFlats, deleted, inserted
       WHERE deleted.ACCOUNT = tblFlats.ACCOUNT
    END
/* * Занесение данных о последних изменениях в лицевом счете */
UPDATE tblAccount SET
  UserName=suser_sname(),-- Имя пользователя
  [DateTime]=getdate(),  -- Дата последней корректировки
  HostName=host_name(),  -- Имя машины, выполнившей корректировку
  ClientName=app_name()  -- Имя приложения
  FROM inserted JOIN tblAccount
       ON inserted.ACCOUNT=tblAccount.ACCOUNT
```

Проверить срабатывание триггера можно несколькими способами. Например, запустите проект MS Access 2010 и выполните любые изменения в таблице `tblAccount`. После этого просмотрите ее содержимое (рис. 13.13).

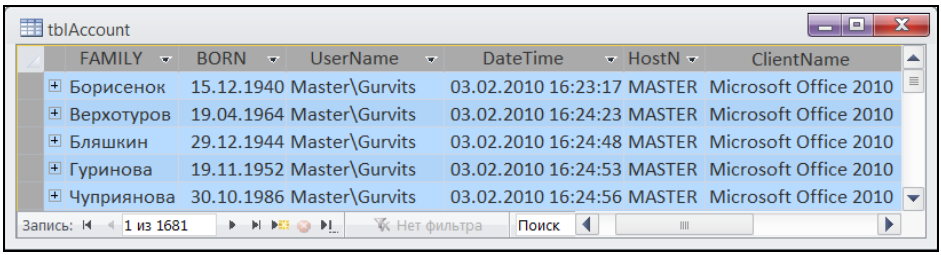
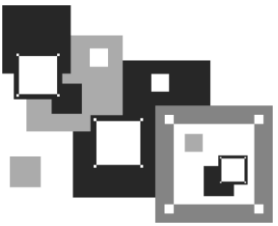


Рис. 13.13. Результаты работы триггера `T_tblAccount_UTrig`

Для отслеживания изменений, выполняемых в таблице `tblAccount`, триггер автоматически создает две служебные таблицы: `inserted` и `deleted`. Такие таблицы создаются для каждого триггера. Они доступны только для чтения. Какие данные в них содержатся?

- Триггер `INSERT`. При добавлении записи таблица `inserted` содержит добавленные строки. При успешном завершении работы триггера все они переносятся в таблицу базы данных. Таблица `deleted` при работе этого триггера не используется.
- Триггер `UPDATE`. При редактировании записи таблица `deleted` содержит строки, которые изменялись, но в состоянии до их изменения. Таблица `inserted` содержит строки в состоянии после их изменения. При успешном завершении работы триггера старые строки заменяются на новые.
- Триггер `DELETE`. При удалении записи таблица `deleted` содержит строки, которые требуется удалить. При успешном завершении работы триггера все они удаляются из таблицы базы данных. Таблица `inserted` при работе этого триггера не используется.



ГЛАВА 14

Внесение изменений в проект Microsoft Access

В *главе 12* был выполнен перенос Access-приложения на платформу "клиент-сервер". Мастер преобразования разделил базу данных MS Access 2010 на две части. Первая часть — база данных Real EstateSQL со всеми сопутствующими объектами попала в MS SQL Server 2008, а вторая, интерфейсная часть, была создана заново, на основе имеющейся (файл Real Estate Часть II.accdb), и размещена в файле проекта MS Access Real EstateCS.adp. Для связи этих двух частей MS Access 2010 по умолчанию использует OLE DB как свой внутренний способ доступа к данным (рис. 14.1).

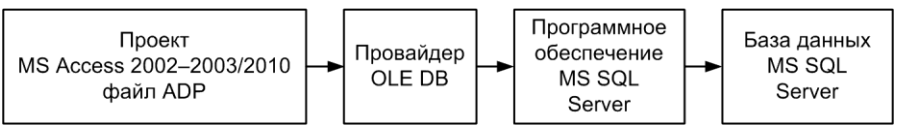


Рис. 14.1. Доступ к данным SQL-сервера из проекта MS Access

14.1. Преимущества работы с мастером преобразования

Вам вряд ли удастся успешно перенести свое приложение на платформу "клиент-сервер", если вы не будете знать особенности этого процесса. Все самые важные из них будут рассмотрены в процессе доработки программного продукта Real Estate, над которым уже потрудился мастер преобразования. В процессе изучения этой главы у разработчика может возникнуть вполне резонный вопрос: "Так стоит ли пользоваться мастером преобразования, если он не справляется со своей работой на все сто?" Не сомневайтесь — стоит! Он прекрасно выполняет перенос данных и конвертирует в новое приложение до 95% объектов, так что основной объем работы за нас будет сделан. Но без некоторой доработки все-таки не обойтись.

14.2. Перенесенные объекты и оставшиеся проблемы

Откроем отчет мастера преобразования, предусмотрительно сохраненный нами в RTF-файле во время работы мастера. В первую очередь просмотрите его внимательно от начала до самого конца. Выясните, все ли объекты данных Microsoft Access 2010 перенесены в базу данных SQL Server?

14.2.1. Таблицы

В начале листинга приведены сообщения о неудачных преобразованиях таблиц. Для примера Real Estate сообщение всего одно: "Таблица: `tblUser`. Таблица пропущена или сбой". Причину пропуска таблицы удастся найти только в конце листинга, когда до таблицы дойдет очередь. Мастер размещает описание своих действий над таблицами, расположенными в алфавитном порядке. Таблица `tblUser` находится в списке на предпоследнем месте. Сообщение:

```
Пропускается таблица 'tblUser'
Ошибка сервера 156: Incorrect syntax near the keyword 'File'.
```

сразу все проясняет. Поле `File`, в котором содержится имя файла с фотографией работника, должно быть переименовано, т. к. совпадает с ключевым словом Transact-SQL `File`. Применять такие имена нельзя. Переименуем поле в `FilePhoto` и повторим процесс конвертации заново. Других проблем при преобразовании таблиц нет. Все они успешно перенесены в базу MS SQL Server 2008.

14.2.2. Условия на значения полей и записей

Некоторые составляющие базы данных MS Access вообще не могут быть перенесены в базу данных SQL Server. Это часть ограничений на значения записей. Мастер выполняет попытку преобразования из синтаксиса ACE в синтаксис SQL Server, но если она заканчивается неудачно, то такое свойство не переносится вообще. В нашей базе данных имеется несколько условий на значения полей. Все они успешно превращены в ограничения `CHECK`. Ограничение номера района в исполнении MS SQL Server 2008 показано на рис. 14.2. А вот условие на значение записи таблицы `tblFlats`, проверяющее совпадение общей площади квартиры и ее составляющих, в базу SQL Server не попало:

```
Условие на [SQUAREFLAT]=[DWELL]+[BRANCH]+[BALCONY]
Не экспортировано.
```

Для того чтобы предотвратить занесение в базу некорректных данных, можно добавить эту проверку в код события **До обновления** формы `Flats` (листинг 14.1).

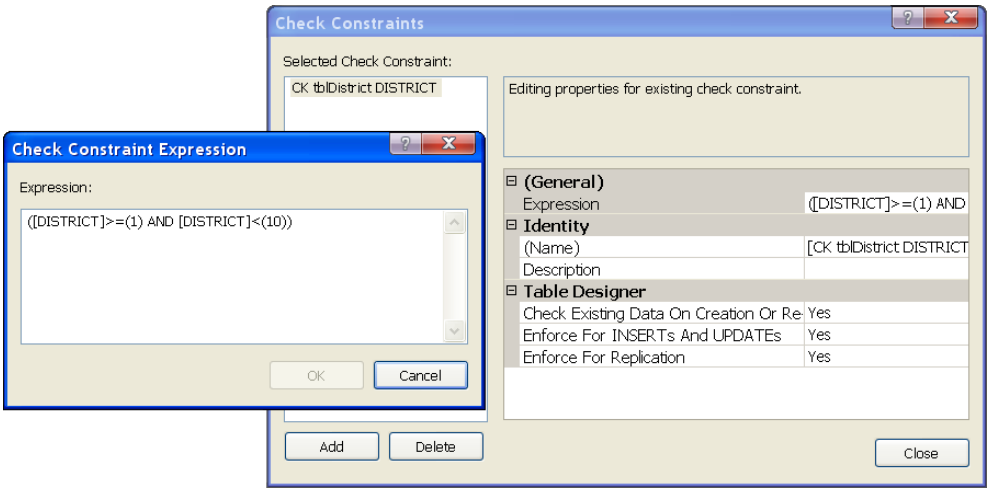


Рис. 14.2. Ограничение **Check**
"Номер района должен быть в интервале от 1 до 9"

Листинг 14.1. Код события *До обновления*

```
Private Sub Form_BeforeUpdate(Cancel As Integer)
If Round(txtSQUAREFLAT.Value, 1) <> _
         Round(txtDWELL.Value + txtBRANCH.Value + _
               txtBALCONY.Value, 1) Then
   ' Несовпадение площадей
   MsgBox "Общая площадь квартиры не равна сумме составляющих.", _
       vbOKOnly + vbExclamation, "Внимание"
   ' Установка курсора в поле общей площади
   txtSQUAREFLAT.SetFocus
   ' Возврат в форму
   Cancel = 1
End If
End Sub
```

В процедуре `Form_BeforeUpdate` значения общей площади и составляющих округляются до десятых долей квадратного метра (функция `Round()`). Это позволяет избежать проблем "мусора" разрядной сетки.

Рассмотрим другой вариант решения проблемы:

```
[SQUAREFLAT]=[DWELL]+[BRANCH]+[BALCONY]
```

Проверку совпадения площадей квартиры следует поручить серверу базы данных (триггер), а не пользовательскому приложению. Такая проверка надежнее, но для написания триггера необходимы знания языка Transact-SQL. Текст триггера приведен в листинге 14.2.

Листинг 14.2. Триггер `T_tblFlats_UTrig1`, проверяющий условие на значение записи

```
CREATE TRIGGER T_tblFlats_UTrig1
ON dbo.tblFlats FOR UPDATE, INSERT AS
SET NOCOUNT ON

-- Проверка соответствия общей площади квартиры и ее составляющих
Declare @SqFlat DECIMAL(7,1)  -- Общая площадь квартиры
Declare @SqDwell DECIMAL(7,1) -- Жилая площадь квартиры
Declare @SqBranch DECIMAL(7,1)-- Вспомогательная площадь
Declare @SqBal DECIMAL(7,1)   -- Приведенная площадь балкона
IF UPDATE(SQUAREFLAT) OR UPDATE(DWELL)
          OR UPDATE(BRANCH) OR UPDATE(BALCONY)
    BEGIN
       SELECT @SqFlat =inserted.SQUAREFLAT,
              @SqDwell=inserted.DWELL,
              @SqBranch=inserted.BRANCH,
              @SqBal=inserted.BALCONY
          FROM inserted
       IF @SqFlat <> (@SqDwell+@SqBranch+@SqBal)
          BEGIN
             RAISERROR ('Несоответствие площади квартиры
                         и ее составляющих',11,1)
             -- Откат транзакции
             ROLLBACK TRANSACTION
          END
    END
```

В теле триггера объявлены четыре локальные переменные: `@SqFlat`, `@SqDwell`, `@SqBranch` и `@SqBal`, которым присваиваются значения составляющих частей площади квартиры. Обратите внимание на тип переменных: `DECIMAL(7,1)`. Эти переменные получают значения, соответствующие откорректированным или вновь введенным значениям полей таблицы `tblFlats`, которые до завершения операции ввода или корректировки хранятся в служебной таблице сервера MS SQL с именем: `inserted`. Тип полей таблицы `tblFlats`, в которых хранятся площади, — `REAL`. В результате операций присваивания данные полей будут конвертированы в `DECIMAL` (десятичные с фиксированной точкой), для которых операция сравнения = (равно) будет иметь смысл и стандартная функция округления `ROUND()` не понадобится.

В случае несовпадения площадей принудительно генерируется сообщение без возникновения ошибки. Для генерации сообщения применяется команда `RAISERROR`, имеющая следующий синтаксис:

```
RAISERROR ( { msg_id | msg_str | @local_variable }
 { , severity, state }
 [ , argument [ ,...n ] ] )
 [ WITH option [ ,...n ] ]
```

В состав конструкции входят следующие аргументы:

- *msg_id* — номер ошибки. Если эта ошибка не системная, а пользовательская, то ее текст можно добавить в каталог sys.messages при помощи системной хранимой процедуры sp_addmessage, указав номер больше чем 50000, и вызывать многократно по этому номеру;
- *msg_str* — текст ошибки с максимальной длиной 2047 символов;
- *@local_variable* — вместо текста ошибки можно указывать локальную переменную типа char или varchar;
- *severity* — уровень серьезности ошибки. Пользователь может указывать уровень серьезности от 0 до 18. Ошибки уровней 20—25 считаются фатальными и не позволяют продолжить выполнение следующих операторов Transact-SQL;
- *state* — статус ошибки. Может принимать значения 1—127. Предназначен для идентификации однотипных ошибок;
- *argument* — применяется для подстановки в текст сообщения нужных значений. Параметров может быть до 20. Каждый должен быть локальной переменной и иметь тип: tinyint, smallint, int, char, varchar, nchar, nvarchar, binary, varbinary. Другие типы данных не поддерживаются;
- *option* — может принимать одно из трех значений:
  - LOG — сообщение будет занесено в системный журнал;
  - NOWAIT — сообщение будет без промедления отправлено клиенту;
  - SETERROR — устанавливает для *msg_id* (номер ошибки) значение системной переменной @@ERROR или значение 50000, в зависимости от уровня серьезности ошибки.

Если вы хотите использовать текст сообщения многократно, определитесь с номером этого сообщения, он должен быть больше чем 50000, и запустите в окне конструктора запросов (**New Query**) системную хранимую процедуру:

```
sp_addmessage 50001,11, 'Несоответствие площади квартиры
и ее составляющих'
```

Здесь 50001 — номер сообщения; 11 — уровень серьезности ошибки.

Теперь текст команды RAISERROR будет значительно короче:

```
RAISERROR (50001,11,1)
```

Для просмотра всех сообщений об ошибках (системных и пользовательских), имеющихся в каталоге sys.messages, можно выполнить оператор SELECT:

```
SELECT * FROM sys.messages
```

Для удаления сообщения из системного каталога применяется системная хранимая процедура sp_dropmessage с указанием номера удаляемого сообщения.

На рис. 14.3 все действия, описанные выше, выполнены в окне Microsoft SQL Server Management Studio. Обратите внимание на сообщение внизу окна: "Query completed with errors". Это не синтаксические ошибки, возникшие при выполнении конструкций, набранных в верхней части окна, а проделки RAISERROR.

Предупреждение

При переносе базы данных на другой MS SQL Server все сообщения, помещенные в системный каталог sys.messages, будут утеряны. Они не принадлежат базе данных, это достояние сервера. Заранее побеспокойтесь об их сохранности или не используйте их вовсе.

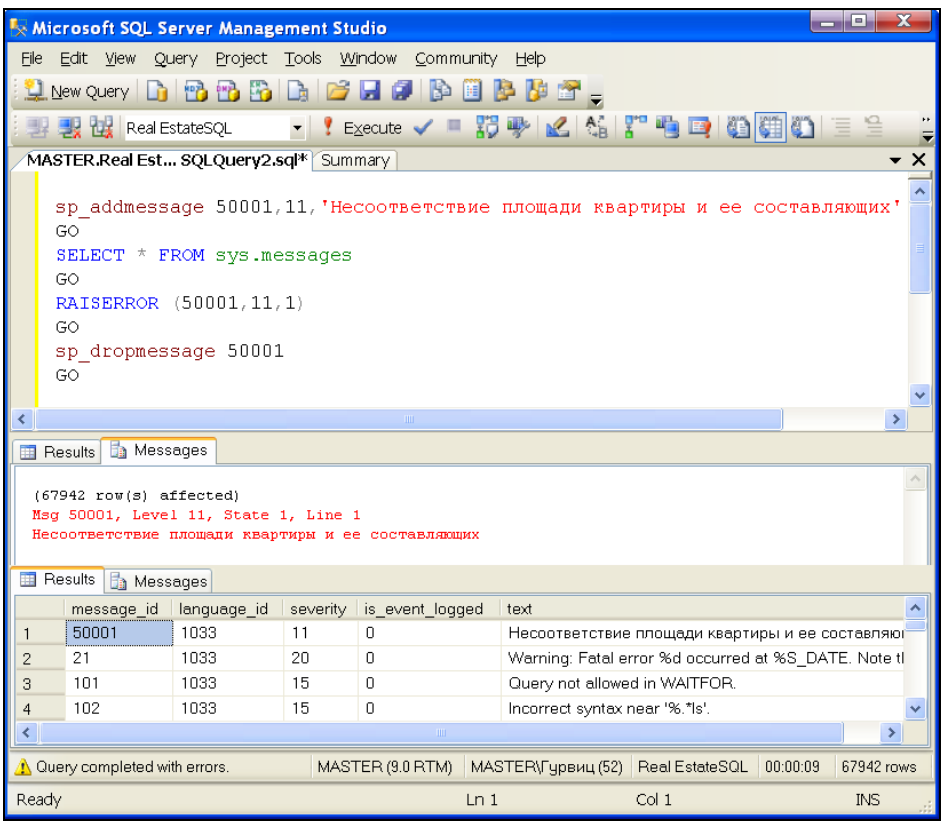


Рис. 14.3. Работа с конструкцией RAISERROR и каталогом сообщений sys.messages

14.2.3. Индексы и ключи

Мастер преобразования легко справляется с переносом индексов, первичных и внешних ключей, но оставлять результаты его деятельности без некоторой доработки не следует. Дело вот в чем. На рис. 14.4 представлен список ключей и индексов, которые созданы мастером для таблицы `tblBuilding`. На первый взгляд их слишком много, и это действительно так.

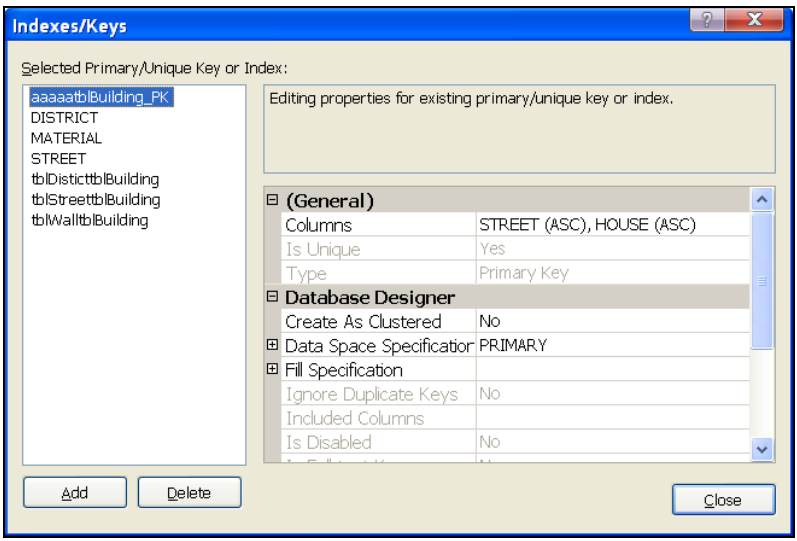


Рис. 14.4. Ключи и индексы, созданные мастером преобразования для таблицы `tblBuilding` (здания)

В MS SQL Server 2008 существует собственная система генерации имен первичных и внешних ключей, а также индексов. В состав имени первичного ключа всегда включаются символы "PK" (что означает Primary Key), а в состав внешнего ключа — "FK" (Foreign Key). Кроме этого в состав имени входят названия таблиц, которые фактически разъясняют, для каких целей тот или иной ключ или индекс создан. Изменять имена, "придуманные" системой, пользователь может, но делать этого на первых порах, скорее всего, не следует. Запутаться в технике работы с MS SQL Server гораздо проще, чем с MS Access.

При детальном рассмотрении состава полей таблицы `tblBuilding`, включенных в индексы, можно заметить значительное дублирование. Три последних индекса (`tblDistricttblBuilding`, `tblStreettblBuilding` и `tblWalltblBuilding`) отсутствовали в MS Access и полностью дублируют предыдущие (`DISTRICT`, `MATERIAL` и `STREET`), которые автоматически перенесены из базы данных MS Access. Вывод сомнению не подлежит — "старые" индексы необходимо удалить.

Совет

Кроме удаления ненужных индексов сделайте первичный ключ каждой таблицы кластерным, т. к. физическое упорядочивание таблицы так же увеличит скорость обработки запросов к базе данных.

14.2.4. Ссылочная целостность

Требуемый уровень функциональности базы данных Real Estate был обеспечен нами путем создания связей между таблицами. На эти связи были наложены определенные ограничения. Например, нами было запрещено удаление района из таблицы районов города, если в таблице зданий имеется хотя бы одно здание, относящееся к этому району (каскадное удаление). Или при изменении номера материала стен здания предписывалось исправить все ссылки на этот номер, имеющиеся в таблице зданий (каскадное обновление). А как обстоят дела со ссылочной целостностью в созданной мастером преобразования базой MS SQL Server?

Откройте таблицу `tblWall` (материал стен здания) в режиме просмотра. Исправьте номер первой записи (под номером 1 значится "Кирпич") на любой другой, кроме имеющихся. Например — семь. Откройте таблицу `tblBuilding` в режиме просмотра. В поле `Material` все единицы заменены на цифру 7! За счет чего? Для решения этой задачи мастер создал триггер. Его текст представлен в листинге 14.3.

Листинг 14.3. Триггер `T_tblWall_UTrig` на обновление записей в таблице `tblBuilding` при изменении ключевого поля в таблице `tblWall`

```
CREATE TRIGGER T_tblWall_UTrig ON [tblWall] FOR UPDATE AS
  SET NOCOUNT ON
  /* * КАСКАДНЫЕ ОБНОВЛЕНИЯ В 'tblBuilding' */
  IF UPDATE(MATERIAL)
      BEGIN
         UPDATE tblBuilding
         SET tblBuilding.MATERIAL = inserted.MATERIAL
         FROM tblBuilding, deleted, inserted
         WHERE deleted.MATERIAL = tblBuilding.MATERIAL
      END
```

Триггер — это специальный тип хранимой процедуры сервера MS SQL. Триггер запускается сервером автоматически при выполнении тех или иных действий с данными таблицы. Когда пользователь выполняет изменение данных, MS SQL Server запускает триггер. Если триггер завершает свою работу корректно, то изменения данных разрешаются.

Примечание

Если исходная база данных создана в MS Access в соответствии с правилами нормализации, то независимо от того, используется ли при конвертации механизм DRI или триггеров MS SQL Server, результирующая база будет работать правильно.

14.2.5. Запросы

Выясните, все ли запросы мастер преобразования перенес в новую базу данных? Из листинга, выданного в результате работы мастера преобразования, следует, что для базы данных Real Estate два запроса из четырех преобразованы успешно, а преобразование двух других закончилось неудачно.

Неудачно сгенерированные конструкции необходимо вручную довести до рабочего состояния, создав хранимые процедуры, функции и представления. В базу данных SQL Server не будут перенесены перекрестные запросы, запросы на выполнение с параметрами, запросы на выполнение, основанные на базе других запросов, и все запросы, базирующиеся на запросе, который не удалось перенести. Доработке запросов будет посвящен один из следующих разделов этой главы.

Насколько успешно функционируют преобразованные запросы в составе MS SQL Server 2008? Попробуем найти их в окне обозревателя объектов **Object Explorer** (рис. 14.5). В разделе **Views** находится только один из них: `dbo.qwrArea`.

Сделайте щелчок правой кнопкой мыши по его имени. Появится контекстное меню. Выберите в нем пункт **Modify**. Представление `dbo.qwrArea` откроется в окне конструктора представлений. Окно содержит четыре панели. На рис. 14.5 их три. Четвертая появится после запуска представления на выполнение.

Верхняя часть окна содержит диаграмму, на которой видны таблицы, включенные в состав представления, и связи между ними. Вторая панель содержит список полей исходных таблиц, включенных в представление. В нижней части окна представлен текст запроса, преобразованного в представление (листинг 14.4).

Листинг 14.4. Текст представления `dbo.qwrArea` на языке Transact-SQL

```
SELECT dbo.tblStreet.NAME,dbo.tblStreet.SIGN,dbo.tblBuilding.HOUSE,
       dbo.tblDistrict.AREA,dbo.tblBuilding.LAND,
       dbo.tblBuilding.YEAR,
       dbo.tblBuilding.WEAR, dbo.tblWall.WALL,
       dbo.tblBuilding.SQUARE
  FROM  dbo.tblWall INNER JOIN
        dbo.tblDistrict INNER JOIN
        dbo.tblStreet INNER JOIN
        dbo.tblBuilding ON dbo.tblStreet.STREET =
        dbo.tblBuilding.STREET ON dbo.tblDistrict.DISTRICT =
```

```
        dbo.tblBuilding.DISTRICT ON
        dbo.tblWall.MATERIAL = dbo.tblBuilding.MATERIAL
WHERE (dbo.tblDistrict.AREA = 'Кировский')
```

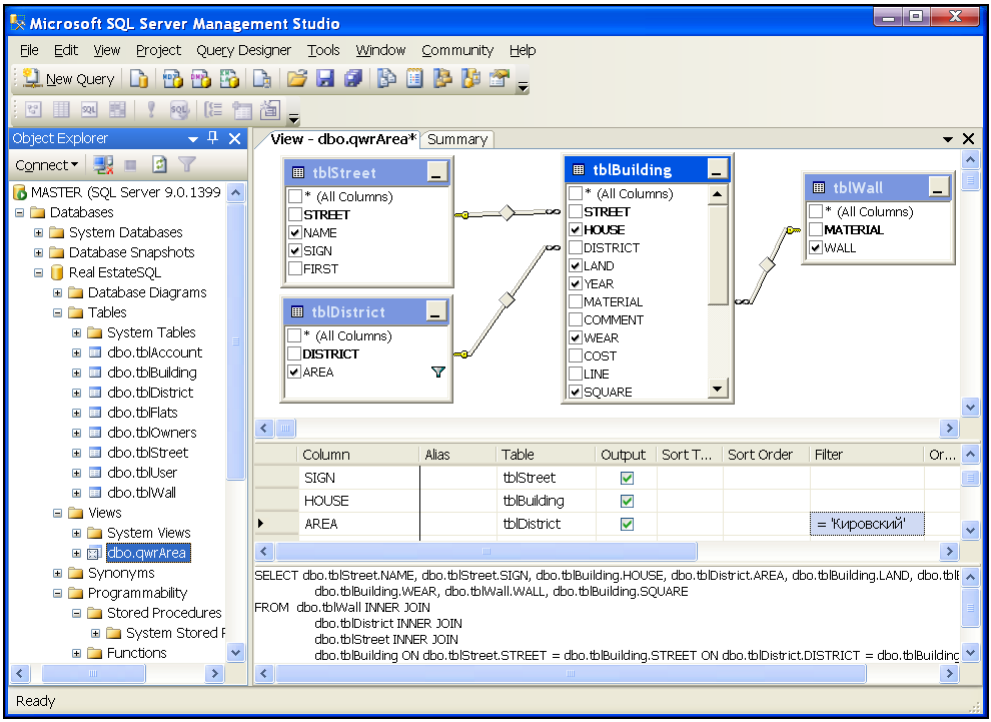


Рис. 14.5. Успешное преобразование запроса MS Access 2010 `qwrArea` в представление MS SQL Server 2008 `dbo.qwrArea`, выполненное мастером преобразования

Отличие диалекта MS Access SQL от Transact-SQL по конструкциям данного примера незначительно, именно поэтому мастер преобразования легко справился с поставленной задачей. Сравните этот текст с текстом MS Access (листинг 14.5). Новое для вас — только аббревиатура dbo (data base owner). Под ней скрывается одноименная схема MS SQL Server 2008, которой принадлежат представление и таблицы базы данных.

Листинг 14.5. Текст запроса `qwrArea` на языке MS Access SQL

```
SELECT tblStreet.NAME, tblStreet.SIGN, tblBuilding.HOUSE,
       tblDistrict.AREA, tblBuilding.LAND,
       tblBuilding.YEAR,
```

```
      tblBuilding.WEAR, tblWall.WALL,
      tblBuilding.SQUARE
FROM tblWall INNER JOIN
   (tblDistrict INNER JOIN
   (tblStreet INNER JOIN tblBuilding ON
   tblStreet.STREET=tblBuilding.STREET) ON
   tblDistrict.DISTRICT=tblBuilding.DISTRICT) ON
   tblWall.MATERIAL=tblBuilding.MATERIAL
WHERE (((tblDistrict.AREA)="Кировский"));
```

Поиски второго запроса `qwrFlats`, успешно преобразованного мастером, увенчаются успехом, ели вы в окне обозревателя объектов **Object Explorer** загляните в раздел **Programmability** и раскроете узел **Functions**. Этот запрос был конвертирован мастером в пользовательскую функцию `dbo.qwrFlats`, которая находится в разделе **Table-valued Functions** (рис. 14.6).

У функции два параметра:

- `@Forms_Building_Street1` (номер улицы);
- `@Forms_Building_House2` (номер дома),

а возвращает она выборку из двух таблиц — `tblFlats` и `tblAccount` (листинг 14.6).

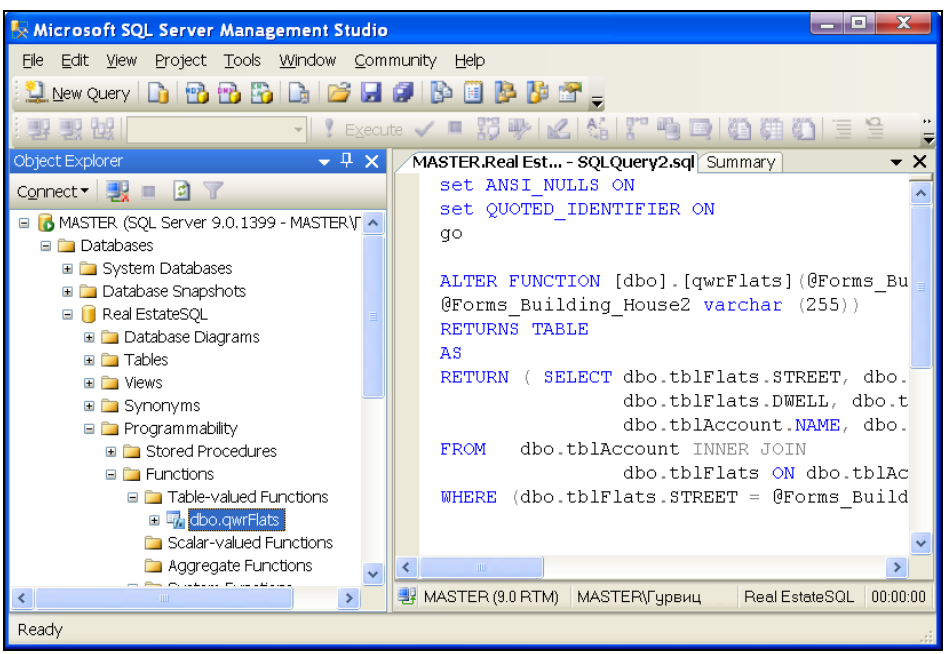


Рис. 14.6. Запрос MS Access 2010 `qwrFlats`, успешно преобразованный мастером в пользовательскую функцию `dbo.qwrFlats` MS SQL Server 2008

Листинг 14.6. Пользовательская функция `dbo.qwrFlats`

```
ALTER FUNCTION [dbo].[qwrFlats]
     (@Forms_Building_Street1 varchar (255),
      @Forms_Building_House2 varchar (255))
RETURNS TABLE
AS
RETURN ( SELECT dbo.tblFlats.STREET,dbo.tblFlats.HOUSE,
   dbo.tblFlats.FLAT,dbo.tblFlats.STOREY,dbo.tblFlats.ROOMS,
   dbo.tblFlats.SQUAREFLAT,
   dbo.tblFlats.DWELL, dbo.tblFlats.BRANCH,
   dbo.tblFlats.BALCONY, dbo.tblFlats.HEIGHT,
   dbo.tblAccount.ACCOUNT, dbo.tblAccount.FAMILY,
   dbo.tblAccount.NAME, dbo.tblAccount.SECOND
FROM dbo.tblAccount INNER JOIN
     dbo.tblFlats ON dbo.tblAccount.ACCOUNT =
     dbo.tblFlats.ACCOUNT
WHERE (dbo.tblFlats.STREET = @Forms_Building_Street1) AND
      (dbo.tblFlats.HOUSE = @Forms_Building_House2) )
```

Созданная мастером функция будет корректно работать без внесения в ее текст каких-либо изменений, но тип параметров функции `varchar(255)`, поставленный мастером "по максимуму", следует все-таки изменить на реальный тип передаваемых данных номера улицы и номера дома: `smallint`.

Для проверки работы функции `dbo.qwrFlats` типа **Table-valued Functions** сделайте щелчок мышью по кнопке **New Query**, расположенной на панели инструментов. В появившемся окне наберите текст:

```
DECLARE @sqlStreet smallint,
        @sqlHouse smallint
SET @sqlStreet=2
SET @sqlHouse=10
SELECT * FROM dbo.qwrFlats(@sqlStreet,@sqlHouse)
```

В отличие от хранимой процедуры, для запуска которой применяется служебное слово `EXECUTE`, необходимо воспользоваться конструкцией `SELECT`. Эту конструкцию можно упростить, указав вместо параметров их непосредственные значения:

```
SELECT * FROM dbo.qwrFlats(2,10)
```

После щелчка по кнопке **! Execute** (Выполнить) на экране появится окно с двумя вкладками (рис. 14.7). Функция `dbo.qwrFlats` вернула в точку вызова информацию по 67 квартирам, расположенным в этом здании.

Следует отметить некоторый прогресс, достигнутый разработчиками компании Microsoft при создании новой версии продуктов MS Access 2010 и MS SQL Server

2008. Запрос с именем `qwrFlats` мастером преобразования MS Access 2003 в SQL Server 2000 не конвертировался вообще.

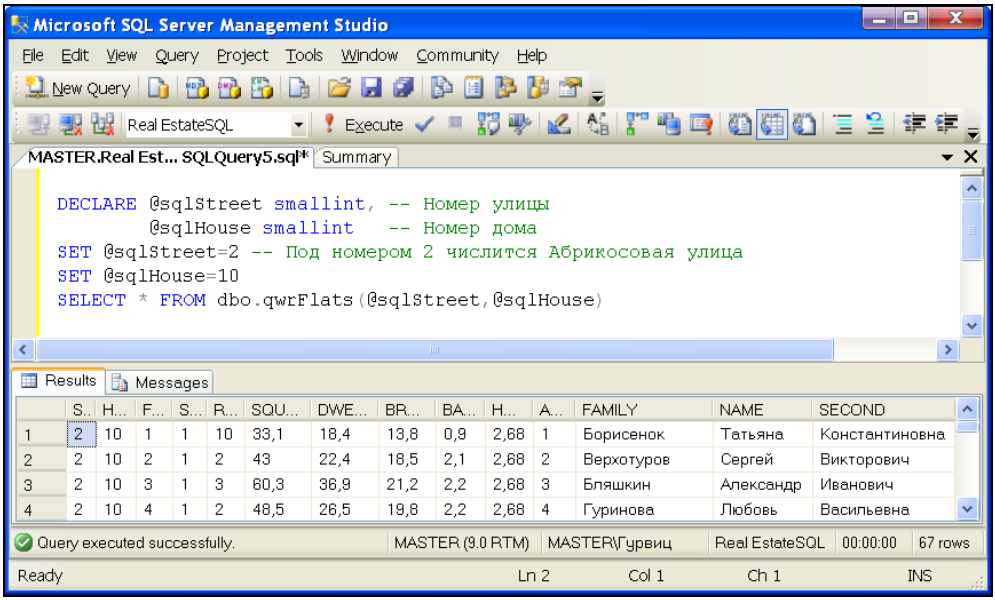


Рис. 14.7. Результаты работы пользовательской функции `dbo.qwrFlats`

Подведем итоги результатов преобразования объектов данных в формат MS SQL Server 2008. Не преобразованы одна таблица `tblUser` из-за присутствия в ее составе поля `File`, которое совпало (по вине разработчика) со служебным словом языка Transact-SQL, и одно условие на значение записи таблицы `tblFlats`. Из четырех запросов MS Access в базе MS SQL Server оказалось только два. Все остальные объекты успешно конвертированы. Результат достаточно неплохой, особенно если учитывать разницу в классе этих программных продуктов и отличия в идеологии при работе с базами данных.

А как обстоят дела с преобразованием интерфейса программного комплекса?

14.3. Первый запуск проекта MS Access

Запустим на выполнение файл Real EstateCS.adp, созданный мастером преобразования в той же папке, где был расположен (и остался там же без изменения) файл Real Estate Часть II.accdb (рис. 14.8).

Форма `Building` работает, причем в составе клиентской части. Посмотрите на линейку MS Access 2010. В ее составе две новые кнопки: последняя и предпоследняя.

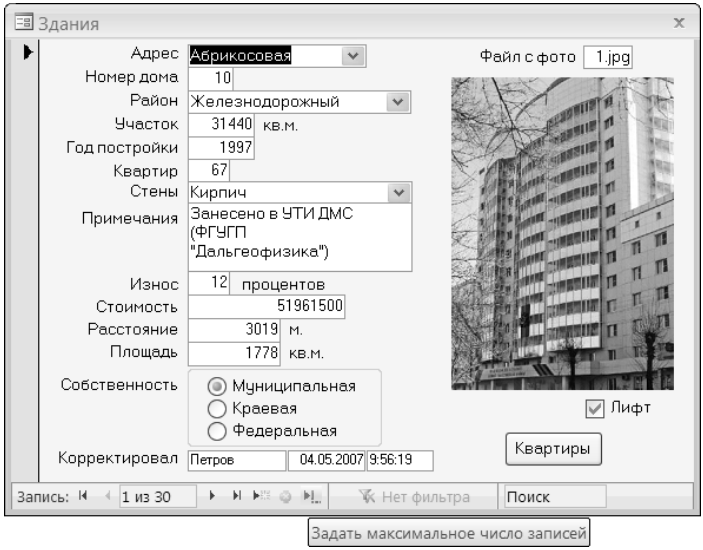


Рис. 14.8. Форма `Building` при работе в составе проекта MS Access

Последняя кнопка  предназначена для ограничения максимального числа записей, которые будут переданы на рабочую станцию. Щелчок по этой кнопке приведет к появлению окна, ограничивающего максимальное количество записей, переданных на рабочую станцию (рис. 14.9).

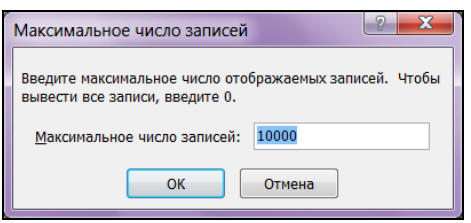


Рис. 14.9. Ограничение числа записей

Щелчок по предпоследней кнопке  позволит прервать пересылку выбранных записей с сервера на рабочую станцию. Во время пересылки эта кнопка светится красным цветом. После завершения операции — становится недоступной.

Запущенный проект MS Access Real EstateCS.adp автоматически подключен мастером к базе данных Real EstateSQL, расположенной на сервере `Master`. Если база данных будет перемещена, или проект MS Access требуется перенести на другой компьютер, то после запуска на выполнение файла Real EstateCS.adp выполните следующие действия:

1. Выберите на главной ленте MS Access 2010 кнопку **Файл**. Откроется диалоговое окно.

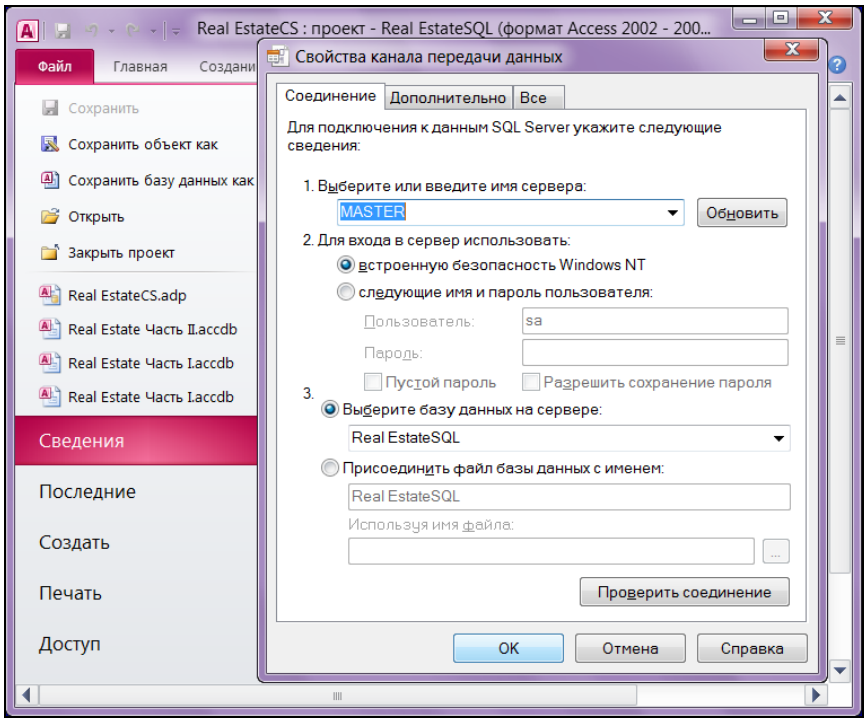


Рис. 14.10. Подключение проекта MS Access 2010 к базе данных MS SQL Server 2008

2. Выберите в нем строку **Сведения**. Правая часть окна изменится. Выберите значок **Управление сведениями о сервере для базы данных**.
3. В открывшемся меню найдите пункт **Подключение** и щелкните по нему. Появится диалоговое окно **Свойства канала передачи данных** (рис. 14.10).
4. Установите требуемые параметры и после проверки подключения нажмите кнопку **OK**.

14.4. Исправление мелких ошибок мастера преобразования

Проверим дальнейшую работоспособность проекта Real EstateCS.adp, созданного мастером преобразования. Скорее всего, ошибок будет немного, и вы заметите, что они повторяются. Это связано с индивидуальностью стиля разработки. К тому же их исправление не потребует значительных временных затрат.

Ошибка 1. При запуске формы `Login` (форма контроля доступа к приложению) фотография работника в форме не отображается, а VBA выдает сообщение об ошибке с номером 91 (рис. 14.11).

При этом отладчик указывает на строку в пользовательской функции (листинг 14.7):

```
NameBase = CurrentDb.NAME
```

Листинг 14.7. Пользовательская функция `CurrentPath()`

```
Public Function CurrentPath() As String
   ' Функция для получения полного пути
   ' к папке с фотографиями
   Dim NameBase As String
   Dim NameBaseShort As String
   Dim CountLetter As Integer
   ' Полное имя файла, в котором находится наша база данных
   NameBase = CurrentDb.NAME
   ' Имя файла без пути к нему возвращает
   ' Dir() — стандартная функция VBA
   NameBaseShort = Dir(NameBase)
   CountLetter = Len(NameBase) — Len(NameBaseShort) — 1
   ' Отделяем слева путь к файлу из CountLetter символов
   ' Left() — стандартная функция VBA
   CurrentPath = Left(NameBase, CountLetter)
End Function
```

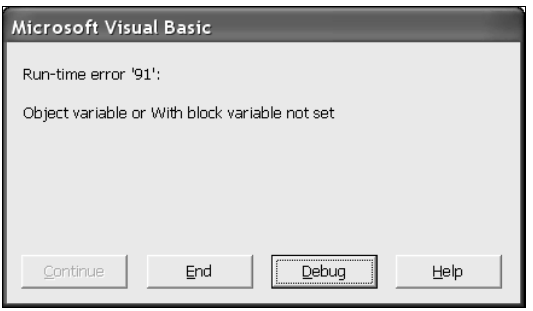


Рис. 14.11. Ошибка, связанная с отсутствием объекта `CurrentDb.Name`

Объект `CurrentDb.NAME` (имя текущей базы данных) неизвестен проекту MS Access. Он имеет дело с внешней базой данных. Посмотрим, для чего в Real Estate Часть II.accdb понадобился путь к текущей базе данных? Для формирования полного пути к папке с фотографиями работников. Эта папка расположена там же, где находится файл Real Estate Часть II.accdb, а теперь и Real EstateCS.adp. Вывод напрашивается сам собой: заменим значение, которое возвращала функция `CurrentPath()`, на `CurrentProject.Path`, а от услуг этой функции можем вообще отказаться. Но так как этот прием используется при работе с несколькими формами, то просто модифицируем вид функции (листинг 14.8).

Листинг 14.8. Пользовательская функция после исправления

```
Public Function CurrentPath() As String
   ' Функция для получения полного пути
   ' к папке, в которой находится наш проект ADP
   CurrentPath = CurrentProject.Path
End Function
```

Ошибка 2. При нажатии кнопки **Вход** формы `Login` (форма контроля доступа к приложению) появляется сообщение об ошибке (рис. 14.12). Отладчик указывает на ошибку с номером 13, которая возникла в строке:

```
ChangePassword = [Forms]![Login]![ComboBox].Column(7)
```

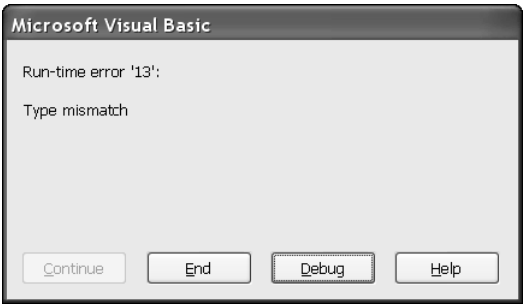


Рис. 14.12. Ошибка, связанная с несовпадением типов `Boolean` (VBA), `Logical` (значения 0 и –1 в Access) и `Bit` (значения 0 и 1 в MS SQL Server)

Переменная `ChangePassword` объявлена как глобальная:

```
Public ChangePassword As Boolean
```

а седьмая колонка объекта `ComboBox` формы `Login` связана с полем таблицы `tblUser` (имя поля `Access02`), в котором в MS Access хранились данные с типом **Логический**. При выполнении оператора присваивания VBA имеет место ошибка несовпадения типов. Какие же типы данных здесь не совпали? И почему связка VBA и MS Access работает, а VBA и SQL Server 2008 — нет?

Откроем таблицу `tblUser`, расположенную на сервере. Тип данных **Логический** поля с именем `Access02` мастер преобразования конвертировал в тип `Bit` MS SQL Server. Почему же VBA отказывается считать 1 (единицу) за `True`? Да потому, что для VBA `True` — это –1 (минус единица), чем как раз и располагает логическое поле MS Access. Выход, как всегда, очень прост:

```
If [Forms]![Login]![ComboBox].Column(7) = 0 Then
   ChangePassword = False
Else
   ChangePassword = True
End If
```

Ошибка 3. При нажатии кнопки **Поиск** формы `PageBuilding` (форма работы со зданиями) появляется сообщение об ошибке (рис. 14.13). Отладчик указывает на ошибку времени выполнения, которая возникла в строке:

```
SQLText = "SELECT tblBuilding.Street, " & _
    "tblStreet.Name AS НАЗВАНИЕ, " & _
    "tblStreet.Sign AS ПРИЗНАК, " & _
    "tblBuilding.House AS ДОМ, tblDistrict.AREA AS РАЙОН, " & _
    "tblBuilding.Land AS УЧАСТОК, tblBuilding.Year AS ГОД " & _
 "FROM tblStreet, tblBuilding, tblDistrict " & _
    "WHERE  tblStreet.Street = tblBuilding.Street " & _
    "AND tblDistrict.District = tblBuilding.District "
```

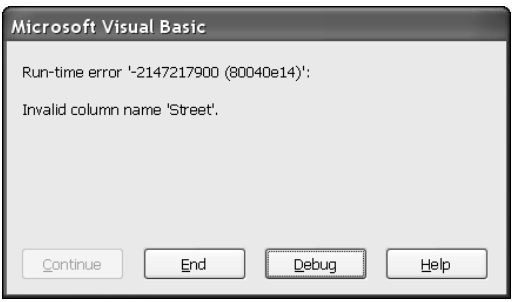


Рис. 14.13. Ошибка, связанная с тем, что MS SQL Server работает с учетом регистра. Имя поля в SQL-запросе должно быть написано как `STREET`

Чем же не понравилось поле `Street` таблицы `tblBuilding` транслятору VBA и почему он считает его ошибочным? Да потому, что такого поля в таблице `tblBuilding` базы данных Real Estate MS SQL Server `Master` нет! Есть поле `STREET`. А так как наш сервер работает с учетом регистра (это было задано при его генерации), то большие и маленькие буквы он считает разными. Это касается не только поля `Street`, так что в строку запроса придется внести несколько изменений:

```
SQLText = "SELECT tblBuilding.STREET, " & _
    "tblStreet.NAME AS НАЗВАНИЕ, " & _
    "tblStreet.SIGN AS ПРИЗНАК, " & _
    "tblBuilding.HOUSE AS ДОМ, tblDistrict.AREA AS РАЙОН, " & _
    "tblBuilding.LAND AS УЧАСТОК, tblBuilding.YEAR AS ГОД " & _
 "FROM tblStreet, tblBuilding, tblDistrict " & _
    "WHERE  tblStreet.STREET = tblBuilding.STREET " & _
    "AND tblDistrict.DISTRICT = tblBuilding.DISTRICT "
```

Ошибка 4. При нажатии кнопки **Поиск** формы `PageBuilding` (форма работы со зданиями) появляется сообщение об ошибке (рис. 14.14). Отладчик указывает на ошибку времени выполнения, которая возникла в строке пользовательской функции `CountQuery()` при выполнении метода `Open`, при открытии соединения с базой данных Real Estate Часть II (листинг 14.9).

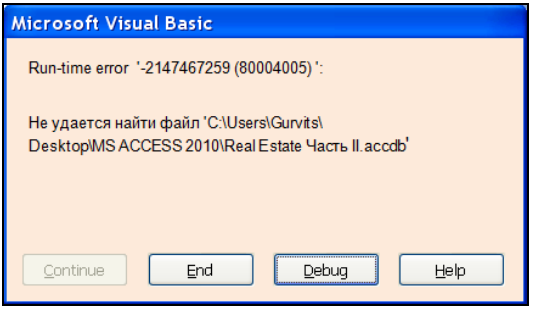


Рис. 14.14. Ошибка, связанная с тем, что проект MS Access пытается получить выборку из базы данных, которой уже нет

Листинг 14.9. Пользовательская функция `CountQuery()`, возвращающая количество зданий, попавших в запрос

```
Public Function CountQuery(SQLText) As Integer
' Возвращает количество записей в запросе.
' SQLText — строка с текстом запроса
Dim CountSelectSQL As Integer      ' Количество записей
' Создание ссылок на реальные объекты
Dim Connect As ADODB.Connection
' Connect — объектная переменная, тип "Соединение".
' Объект ADO верхнего уровня
Dim rsCount As ADODB.Recordset
' rsCount — объектная переменная, содержит набор записей

' Для создания соединения с внешней базой данных
Set Connect = New ADODB.Connection

' Только для MS Access 2007/2010 (accdb-файл)
With Connect
     .ConnectionString = _
     "Provider=Microsoft.ACE.OLEDB.12.0;" & _
     "Data Source=" & CurrentPath() & "\Real Estate.accdb"
     .Open
End With

...
End Function
```

Очень хорошо, что мы удалили "старую" базу данных из папки, где расположен проект Real EstateCS.adp. Если бы она осталась на месте, то этой ошибки не возникло, и все пользователи регулярно получали бы настоящую дезинформацию.

Выборка зданий делалась бы из базы MS SQL Server, а количество записей подсчитывалось по данным базы MS Access!

Для исправления ошибки следует указать другую строку подключения:

```
With Connect
     .ConnectionString = _
     "Provider=SQLOLEDB;" & _
     "Data Source=Master;" & _
     "Database=Real EstateSQL;" & _
     "Trusted_Connection=yes"
     .Open
 End With
```

Эта строка подключения будет работать не только в проекте MS Access. Ее с успехом можно использовать при подключении к базе данных MS SQL Server из любого другого приложения, понимающего VBA, например, AutoCAD. Ведь это не что иное, как информация для ADO. В нашем случае можно поступить еще проще. Замените предыдущий фрагмент кода одной строкой:

```
Set Connect = CurrentProject.Connection
```

Ошибка 5. При нажатии кнопки **Поиск** формы `PageBuilding` (форма работы со зданиями) никакие сообщения об ошибках не возникают, но и данные в поле со списком `ListBox` второй страницы этой формы не выводятся, хотя функция `CountQuery()` сообщает, что записи в выборке есть (рис. 14.15). Об этом говорит и надпись внизу формы: "Количество зданий, попавших в запрос: 30".

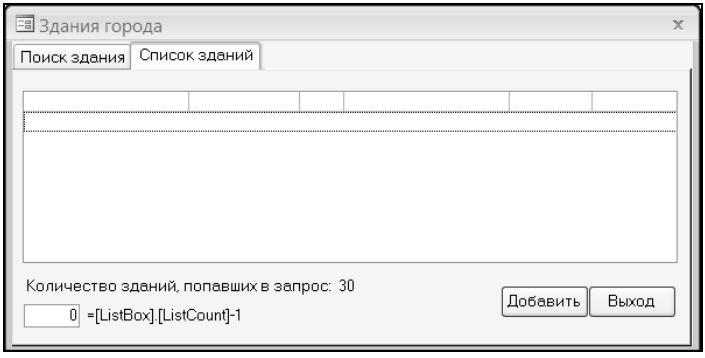


Рис. 14.15. Ошибка, связанная с тем, что проект MS Access пытается получить выборку из базы данных, которая перенесена

Относительно серьезная ошибка, заключающаяся в следующем. В качестве свойства `RowSourceType` в файле Real Estate Часть II.accdb указано (таблица или запрос):

```
ListBox.RowSourceType = "Table/Query"
```

В файле проекта Real EstateCS.adp для этого свойства следует указать (таблица, представление или хранимая процедура):

```
ListBox.RowSourceType = "Table/View/StoredProc"
```

Ошибка 6. При нажатии кнопки **Удалить**, расположенной на третьей вкладке формы `PageBuilding` (форма работы со зданиями), никакие сообщения об ошибках не возникают, но здание не удаляется, если в нем есть квартиры.

Это проделки триггера `T_tblBuilding_DTrig` на удаление для таблицы `tblBuilding` и обработчика ошибок процедуры `CommandDel_Click()` формы `PageBuilding`. Вот фрагмент кода, нуждающийся в исправлении:

```
' При ошибке перейти к метке: SqlUpdateErr
On Error GoTo SqlUpdateErr
...
SqlUpdateErr:
   If Err.NUMBER = 2501 Then
      MsgBox "Удаление здания не выполнено.", _
      vbOKOnly + vbExclamation, "Внимание"
   End If
```

При разработке accdb-приложения, для того чтобы оградить пользователя от англоязычного сообщения системы "Run-time error" при выборе кнопки **Нет** (см. рис. 8.6), мы "выпустили" на экран рабочей станции только сообщение приложения об ошибке с номером 2501. Тогда неверно считалось, что других ошибок при работе процедуры `CommandDel_Click()` не может быть в принципе. Знать бы, где упасть! Доработка текста может иметь следующий вид:

```
SqlUpdateErr:
  Select Case Err.Number
    Case 2501
         ' Отказ пользователя от удаления здания
         MsgBox "Удаление здания не выполнено.", _
         vbOKOnly + vbExclamation, "Внимание"
    Case 44445
         ' Ошибка срабатывания триггера удаления.
         ' См. текст триггера T_tblBuilding_DTrig
         MsgBox "Здание удалить нельзя. В нем есть квартиры!", _
         vbOKOnly + vbExclamation, "Внимание"
    Case Else
         ' Другие ошибки
         MsgBox Err.Description
  End Select
```

Ошибка 7. При запуске формы `PassWord` (форма изменения пароля) появляется сообщение об ошибке (рис. 14.16). Отладчик указывает на ошибку времени выполнения, которая возникла в строке:

```
SELECT tblUser.LastName, tblUser.FirstName,
       tblUser.SecondName, tblUser.Post,
       tblUser.PassWord, tblUser.Date_up,
       User.Time_up
FROM [tblUser] WHERE
     (((tblUser.LastName)=GetValueSQL()));
```

Это значение свойства **Источник записей** формы `PassWord`. Для того чтобы сообщить форме данные работника, была использована пользовательская функция `GetValueSQL()`, текст которой был приведен ранее в листинге 8.9.

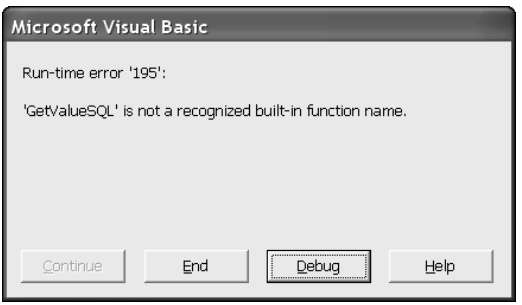


Рис. 14.16. Ошибка, связанная с тем, что в состав конструкции `SELECT` свойства **Источник записей** формы входит пользовательская функция `GetValueSQL()`

Понятно, что эта функция MS SQL-серверу не известна. Для решения проблемы не будем создавать аналогичную пользовательскую функцию для MS SQL Server и передавать ей в качестве параметра фамилию работника (глобальная переменная `FAMILY`). Есть способ проще. Удалим значение свойства **Источник записей** в окне свойств формы, а в процедуру обработки события **Загрузка** включим код, приведенный в листинге 14.10.

Листинг 14.10. Код события *Загрузка* для формы `PassWord`

```
Private Sub Form_Load()
Me.RecordSource = "SELECT tblUser.LastName, tblUser.FirstName, " & _
     "tblUser.SecondName, tblUser.Post, tblUser.PassWord, " & _
     "tblUser.Date_up, tblUser.Time_up " & _
     "FROM tblUser " & _
     "WHERE tblUser.LastName = '" & FAMILY & "'"
End Sub
```

Ошибка 8. При нажатии кнопки **Записать** формы `PassWord` (форма изменения пароля) появляется сообщение об ошибке (рис. 14.17). Отладчик указывает на ошибку времени выполнения, которая возникла в строке:

```
TextSQL = "UPDATE tblUser " & _
    "SET tblUser.PassWord = '" & NewPasswordOne & "'," & _
    "tblUser.Date_up=Date(),tblUser.Time_up=Time() " & _
    "WHERE tblUser.LastName = '" & FAMILY & "' "
```

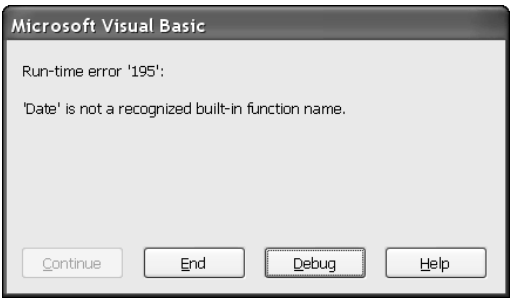


Рис. 14.17. Ошибка, связанная с тем, что в состав конструкции `UPDATE` входит стандартная функция `Date()` MS Access, а не `GETDATE()` MS SQL Server

В тексте сообщения об ошибке VBA сообщает имя функции, которая не известна MS SQL-серверу. Это функция текущей даты. Наряду с измененным значением пароля пользователя в таблицу `tblUser` форма `PassWord` записывает дату и время последней корректировки, а также фамилию работника, выполнившего корректировку.

При работе с базой данных MS Access такой подход вполне оправдан. Если же база данных находится под контролем MS SQL Server, то отслеживание изменений разумнее поручить триггеру базы данных, а не приложению. Тогда злоумышленник не сможет не "засветиться" при изменении каких-либо данных в таблице, даже если ему удастся подключиться к ней, минуя приложение. Такой подход рассмотрен в предыдущей главе (см. листинг 13.4).

Не будем создавать триггер для таблицы `tblUser`. Просто заменим функции MS Access на функции MS SQL Server. Однако единообразия на этом пути нет. В MS Access текущие дата и время возвращаются двумя функциями: `Date()` и `Time()`, а в MS SQL Server для этих целей предназначена одна функция `GetDate()`. В MS Access также можно было обойтись одной `Now()`, но этого сделано не было. Придется выкручиваться! Первую часть от даты/времени система "отстегнет" сама, а для отсечения текущего времени воспользуемся функцией `Convert()`:

```
TextSQL = "UPDATE tblUser " & _
    "SET tblUser.PassWord = '" & NewPasswordOne & "'," & _
    "tblUser.Date_up=GETDATE()," & _
    "tblUser.Time_up=CONVERT(CHAR(8),GETDATE(),14) " & _
    "WHERE tblUser.LastName = '" & FAMILY & "' "
```

Подведем итоги. Мелких ошибок сравнительно много, но они относятся всего к трем типам, причем всех их можно было избежать еще на стадии разработки accdb-файла.

- *Ошибки соединения с базой данных.* Не пишите `CurrentDb.Name`, а используйте `CurrentProject.Path` и `CurrentProject.Connection`. Свойство `RowSourceType = "Table/Query"` можно было вообще не писать. В этом случае философия умолчания MS Access сработает безупречно.
- *Ошибки регистра.* Не генерируйте MS SQL Server, независимый от регистра, а просто будьте внимательны при написании кода.
- *Ошибки стандартных и пользовательских функций.* Откажитесь от функций, входящих в состав запросов. VBA легко позволяет это сделать. Указывайте в них только конечные значения.

14.5. Доработка интерфейса программного комплекса

Правила обновления данных в MS Access 2010 несколько отличаются от правил обновления MS SQL Server 2008. Ведь ядро базы данных совсем иное.

Форма `Flats`, созданная нами для работы с квартирами в accdb-файле, базируется на запросе, содержащем данные из двух рабочих таблиц — `tblFlats` и `tblAccount`. Только их данные модифицируются во время работы этой формы. Вы можете возразить: "А как же данные по проживающим? Они тоже корректируются в этой форме!" Подчиненная форма `Owners`, входящая в состав формы `Flats`, — не в счет, у нее своя рабочая таблица `tblOwners`.

В составе adp-файла основу формы `Flats` составляет выборка, возвращаемая пользовательской функцией `qwrFlats` (см. листинг 14.4) MS SQL Server 2008 типа **Table-valued Functions**. Она также содержит данные из этих же таблиц, однако механизм взаимодействия с формой — другой. На пути к нему исправим еще одну неточность мастера преобразования. Запустим на выполнение форму `Flats`, щелкнув по кнопке **Квартиры**, которая находится в форме `Building` (рис. 14.18). MS Access потребует ввода параметра. Ничего удивительного, так решил мастер преобразования, указав в свойстве **Входные параметры** формы `Flats` следующую строку:

```
? = Forms_Building_Street, ? = Forms_Building_House
```

Это свойство можно найти на вкладке **Данные** окна свойств формы `Flats`. Не следует заставлять пользователя задумываться над вводом номера улицы и номера дома. Заменим строчку, сгенерированную мастером на следующую:

```
@Forms_Building_Street=[Forms]![Building]![Street],
@Forms_Building_House=[Forms]![Building]![House]
```

Параметры для работы функции `qwrFlats` программный комплекс теперь найдет в форме `Building`. После этого форма `Flats` будет запускаться в штатном режиме.

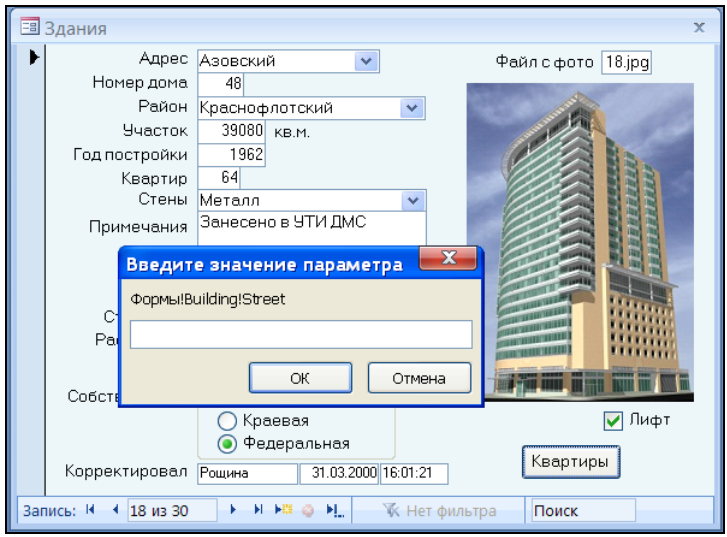


Рис. 14.18. Для запуска формы `Flats` требуется ввод параметров

14.5.1. Обновление данных в форме с двумя таблицами

Запустим форму `Flats` после выполненных исправлений. Напомню, что это следует делать только из формы `Building`, выполнив щелчок по кнопке **Квартиры**. На экране появится список квартир в здании. Удача? Нет! Добавлять и удалять записи в таблицу квартир `tblFlats` и таблицу лицевых счетов `tblAccount` эта форма не дает, хотя работа с проживающими в квартире (таблица `tblOwners`) выполняется нормально. Следует отметить, что редактирование абсолютно всех данных, отображенных в форме, работает корректно, поэтому, в принципе, форму можно оставить в составе программного комплекса для обеспечения деятельности подразделений, занимающихся только мониторингом, но это непрофессиональный подход.

14.5.2. Исправление формы для работы с квартирами

Почему же при работе с формой квартир `Flats` в составе adp-файла пользователь потерял возможность удаления старых и добавления новых записей? Причиной этому является пользовательская функция `qwrFlats` (см. листинг 14.4), созданная мастером преобразования на основе одноименного Access SQL-запроса. Именно выборка, сделанная ей на основе двух таблиц, лишена возможностей удаления и вставки записей в эти таблицы. Доработку формы `Flats` начнем с исправления

функции qwrFlats. Удалим список полей, относящихся к таблице tblAccount, и саму эту таблицу из секции FROM (листинг 14.11).

Листинг 14.11. Пользовательская функция qwrFlats после доработки

```
ALTER FUNCTION [dbo].[qwrFlats]
  (@Forms_Building_Street smallint,
   @Forms_Building_House smallint)
RETURNS TABLE
AS
RETURN (SELECT *
          FROM dbo.tblFlats
          WHERE (dbo.tblFlats.STREET = @Forms_Building_Street)
            AND (dbo.tblFlats.HOUSE = @Forms_Building_House))
```

Выборка данных только из одной таблицы tblFlats позволит функции qwrFlats (а заодно и форме Flats) модифицировать, удалять и добавлять записи в эту таблицу.

На следующем этапе удалим все те поля, отображенные в форме Flats, которые связаны с таблицей tblAccount, а для того чтобы не изменять технологию работы с квартирами, которую рядовые пользователи уже успели освоить до автоматизма, создадим подчиненную форму на основе таблицы tblAccount. Как мы уже убедились, механизм подчиненных форм переносится в клиент-серверную архитектуру без каких-либо доработок. Остановимся подробнее на создании второй подчиненной формы, расположенной в главной форме Flats. Напомню, что первая подчиненная форма Owners предназначена для отображения информации о проживающих в квартире.

Особенность предстоящей работы заключается в том, что создаваемая нами форма будет отображать всего одну запись из таблицы tblAccount. Одна квартира — один лицевой счет. Поэтому режим отображения формы будет определен как **Одиночная форма** без кнопок перехода. Алгоритм создания этой подчиненной формы следующий:

1. Откройте первичную форму Flats в режиме конструктора.
2. Убедитесь, что на панели элементов кнопка  **Использовать мастера** нажата. Если нет — "выделите" ее щелчком левой кнопки мыши. Нам понадобится работа построителя подчиненных форм.
3. Нажмите на панели элементов кнопку  **Подчиненная форма**. Наведите указатель мыши на то место первичной формы, где вы планируете поместить левый верхний угол подчиненной формы. Указатель мыши превратится в значок подчиненной формы с крестиком в левом верхнем углу.

4. Нажмите левую кнопку мыши и, удерживая ее в нажатом состоянии, переместите курсор по диагонали так, чтобы получилась рамка требуемого размера. Отпустите левую кнопку мыши. Автоматически запустится построитель подчиненной формы.
5. Первый шаг работы мастера подчиненных форм — определение данных, которые надо включить в подчиненную форму. "Утопите" переключатель  **Имеющиеся таблицы и запросы** и щелкните по кнопке **Далее**. Появится окно для выбора таблиц и полей. Выберите все поля из таблицы `tblAccount`.
6. Второй шаг — определение полей связи между главной и подчиненной формами. Сделайте щелчок по кнопке **Самостоятельное определение** и установите связь форм по полю `ACCOUNT`.
7. Шаг номер три — выбор имени для подчиненной формы. Под этим именем она появится в списке форм проекта Real EstateCS. Введите имя, например `SlaveAccount`, и щелкните по кнопке **Готово**.
8. Подвергнем работу мастера некоторому улучшению. Изменим четыре свойства формы. Они приведены в табл. 14.1.

Таблица 14.1. *Измененные свойства формы* `SlaveAccount`

Номер	Свойство	Вкладка	Значение
1	**Режим по умолчанию**	**Макет**	*Одиночная форма*
2	**Кнопки перехода**	**Макет**	*Нет*
3	**Область выделения**	**Макет**	*Нет*
4	**Полосы прокрутки**	**Макет**	*Отсутствуют*

9. Удалим все объекты, созданные мастером подчиненных форм, а также заголовок формы и ключевое поле `ACCOUNT`. Для удаления элемента сделайте по нему щелчок левой кнопкой мыши и нажмите клавишу <Delete>.
10. Закроем форму `Flats` и откроем в режиме конструктора подчиненную форму `SlaveAccount`. Создадим в ней пять необходимых полей: фамилия, имя, отчество, дата рождения и документ (рис. 14.19).
11. Отформатируем объекты формы. Для этого поместите указатель мыши в любую точку на границе выделенного элемента, отличную от маркеров изменения размеров. Нажмите левую кнопку мыши и, не отпуская ее, перетащите элемент на новое место. Более точно "выставить" элемент управления на форме можно при помощи клавиш-стрелок при нажатой клавише <Ctrl>, а поточнее изменить размеры элемента можно при помощи клавиш-стрелок при нажатой клавише <Shift>.

Проверим работу формы `Flats` в режиме добавления новой записи. На экране появится очередное препятствие (рис. 14.20). Это сработал триггер на добавление

записи в таблицу `tblFlats` (листинг 14.12). Триггер создан мастером преобразования и педантично исполняет свои обязанности. Действительно, мы добавляем запись в таблицу квартир, а лицевой счет по этой квартире не заведен. Выход прост — следует отказаться от услуг этого триггера. Удалим его из базы данных и получим правильно работающую форму.

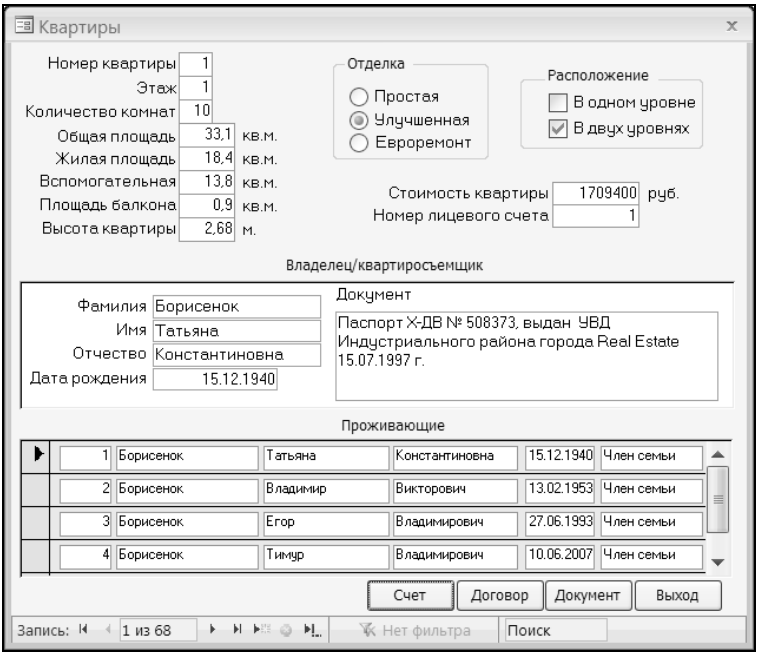


Рис. 14.19. Форма `Flats` после доработки

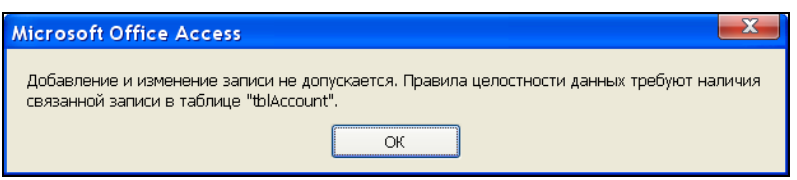


Рис. 14.20. Результат срабатывания триггера

Листинг 14.12. Текст триггера `T_tblFlats_ITrig`, срабатывающего при добавлении новой записи в таблицу `tblFlats`. Этот триггер необходимо удалить

```
ALTER TRIGGER T_tblFlats_ITrig
ON dbo.tblFlats FOR INSERT AS
SET NOCOUNT ON
```

```
/* * ЗАПРЕТ ВСТАВКИ БЕЗ СОВПАДАЮЩЕГО КЛЮЧА В 'tblAccount' */
IF (SELECT COUNT(*) FROM inserted) !=
   (SELECT COUNT(*) FROM tblAccount, inserted
          WHERE (tblAccount.ACCOUNT = inserted.ACCOUNT))
    BEGIN
        RAISERROR 44447 'Добавление и изменение записи
        не допускается. Правила целостности данных требуют
        наличия связанной записи в таблице "tblAccount".'
        ROLLBACK TRANSACTION
    END
```

Две последние проблемы, которые осталось решить:

- удаление связанной записи в таблице лицевых счетов `tblAccount` при удалении данных по квартире в таблице `tblFlats`;
- каскадное обновление связанной записи в таблице `tblAccount` при изменении данных по квартире в таблице `tbtFlats`.

В листингах 14.13 и 14.14 приведен текст, позволяющий создать триггеры для таблицы `tbtFlats` на удаление и модификацию записей.

Листинг 14.13. Текст триггера `T_tblFlats_DTrig`, срабатывающего при удалении записи в таблице `tblFlats`

```
CREATE TRIGGER [T_tblFlats_DTrig] ON [dbo].[tblFlats] FOR DELETE AS
SET NOCOUNT ON

/* * Удаление связанной записи в 'tblAccount' */
DELETE tblAccount
     FROM deleted, tblAccount
     WHERE deleted.ACCOUNT = tblAccount.ACCOUNT
```

Листинг 14.14. Текст триггера `T_tblFlats_UTrig`, срабатывающего при изменении номера лицевого счета в таблице `tblFlats`

```
CREATE TRIGGER [T_tblFlats_UTrig] ON [dbo].[tblFlats] FOR UPDATE AS
SET NOCOUNT ON

/* * Каскадные обновления в 'tblAccount' */
IF UPDATE(ACCOUNT)
    BEGIN
       UPDATE tblAccount
       SET tblAccount.ACCOUNT = inserted.ACCOUNT
       FROM tblAccount, deleted, inserted
       WHERE deleted.ACCOUNT = tblAccount.ACCOUNT
    END
```

14.5.3. Улучшенный вариант формы

Форма `Flats`, созданная в *части I* для работы с информацией по квартирам, расположенным в здании, потребовала минимум времени на ее создание. Однако форма `Flats`, выполненная в основном стиле MS Access, имеет ряд недостатков. Вот только несколько из них.

- Пользователь не видит списка всех квартир, расположенных в здании. В любой момент времени у него перед глазами полная информация только по одной из них.
- Линейка навигации по записям не дает возможности сразу перейти к требуемой квартире. Список следует "прокрутить" шаг за шагом, а ведь есть здания, в которых по сотне квартир и более.
- Все изменения, выполненные в форме, сразу попадают в базу данных. Работник лишен возможности отказаться от них.

Оставим форму `Flats` в составе программного комплекса и создадим ее лучший вариант — `PageFlats`, но не с "нуля", а с использованием уже имеющихся наработок. MS Access 2010 легко позволяет копировать свои объекты как в accdb-файл другой базы данных или adp-файл, так и дублировать их в имеющемся, но с другим именем. Для этого выполните следующие действия:

1. Откройте в среде MS Access 2010 файл базы данных, с которым предстоит работать.
2. Найдите в области навигации форму `Flats`. Сделайте по ее имени щелчок правой кнопкой мыши.
3. Появится контекстное меню. Выберите в нем пункт **Копировать.**
4. Щелкните правой кнопкой мыши в любом месте области переходов. Не бойтесь скопировать форму туда, где она не нужна. MS Access просто не позволит этого сделать.
5. Появится контекстное меню. Выберите в нем пункт **Вставить.**
6. На экране появится диалоговое окно, предлагающее ввести имя создаваемой формы (рис. 14.21). Введите имя `PageFlats` и щелкните по кнопке **OK.**
7. Копия формы `Flats` под именем `PageFlats` появится в разделе **Формы** области навигации.

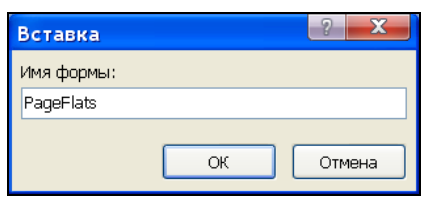


Рис. 14.21. Создание копии формы `Flats` для дальнейшей ее модификации

Примечание

Для копирования объекта в другой файл вместо пункта контекстного меню **Копировать** выберите пункт **Экспорт**, а в открывшемся диалоговом окне укажите файл accdb или adp, в котором требуется разместить копируемый объект.

Для дальнейшей доработки откроем форму `PageFlats` в режиме конструктора и выполним следующие шаги:

1. Выделим все объекты формы и переместим их в буфер обмена нажатием комбинации клавиш <Ctrl>+<X>.
2. Создадим в форме набор из двух вкладок. Для этого в разделе **Элементы управления** вкладки **Конструктор** выберем инструмент **Вкладка** и разместим его в форме.
3. В нужном месте активной области формы при помощи левой кнопки мыши отведем место этому объекту.
4. Перейдем на вторую вкладку. Щелчок правой кнопкой мыши по рамке, обрамляющей вкладку, откроет контекстное меню. Выберем в нем пункт **Вставить**. Все ранее "вырезанные" объекты появятся на второй вкладке.
5. Переименуем наши вкладки. Выберем их по очереди. В **Окне свойств** в разделе **Другие** изменим свойство **Имя** на *Page1* и *Page2*.
6. Перейдем на вкладку **Макет**. Изменим значения свойства **Подпись** на *Список квартир* и *Данные по квартире* (рис. 14.22).
7. Для второй вкладки значение свойства **Вывод на экран** изменим на *Нет*.
8. Дадим имя набору вкладок — *PageFrame*.

Форма `PageFlats` не может работать независимо от формы `PageBuilding`, которая сообщает ей координаты здания, поэтому код кнопки **Квартиры** придется изменить (листинг 14.15).

Листинг 14.15. Код кнопки *Квартиры* формы `PageBuilding`

```
Private Sub CommandFlats_Click()
' Кнопка Квартиры
On Error GoTo Err_CommandFlats_Click
    Dim stDocName As String
    Dim stLinkCriteria As String
    ' Номер улицы, на которой расположено здание
    SelectAddressStreet = Me!ComboStreet.Value
    ' Номер дома
    SelectAddressHouse = Me!txtHOUSE.Value
    stDocName = "PageFlats"
    DoCmd.OpenForm stDocName, , , stLinkCriteria
Exit_CommandFlats_Click:
    Exit Sub
```

```
Err_CommandFlats_Click:
    MsgBox Err.Description
    Resume Exit_CommandFlats_Click
End Sub
```

В нем появились две глобальные переменные: `SelectAddressStreet` и `SelectAddressHouse` (номер улицы и номер дома), объявление которых необходимо выполнить в разделе описаний модуля `ModuleMain`. В этом случае они будут "видимы" из любой формы проекта MS Access.

Для того чтобы предотвратить отображение в форме `PageFlats` ошибочной информации при ее автономном запуске, добавим в код события **Открытие** следующий текст (листинг 14.16).

Листинг 14.16. Открытие формы `PageFlats`

```
Private Sub Form_Open(Cancel As Integer)
' Открытие формы
If Not CurrentProject.AllForms("PageBuilding").IsLoaded Then
   MsgBox "Форма запущена автономно." _
   & " Запустите ее из формы PageBuilding!", _
   vbOKOnly + vbExclamation, "Внимание"
   Cancel = 1
End If
End Sub
```

Определимся с объектами, которые будут размещены на первой странице формы `PageFlats`. Это, прежде всего, адрес здания и количество квартир в нем, а также сам список целиком (рис. 14.22). Не будем в качестве источника данных этого списка использовать функцию MS SQL Server `qwrFlats`. Есть способ проще. Создадим обновляемый статический набор, добавив в событие **Загрузка** формы `PageFlats` код, представленный в листинге 14.17. Обратите внимание на глобальные переменные `SelectAddressStreet` и `SelectAddressHouse`, которые получили значения в листинге 14.15. В табл. 14.2 приведен список объектов первой страницы формы `PageFlats`.

Листинг 14.17. Загрузка формы `PageFlats`

```
Private Sub Form_Load()
ListBox.RowSource = "SELECT FLAT,STOREY,ROOMS,SQUAREFLAT, " & _
  "DWELL,BRANCH,BALCONY,HEIGHT,COST,ACCOUNT " & _
  "FROM tblFlats WHERE STREET = " & _
  SelectAddressStreet & " AND HOUSE = " & SelectAddressHouse & _
  " ORDER BY FLAT"
```

```
' Выделение первой записи объекта "Поле со списком"
ListBox.Selected(0) = True
CommandMain.Enabled = AccountWork
End Sub
```

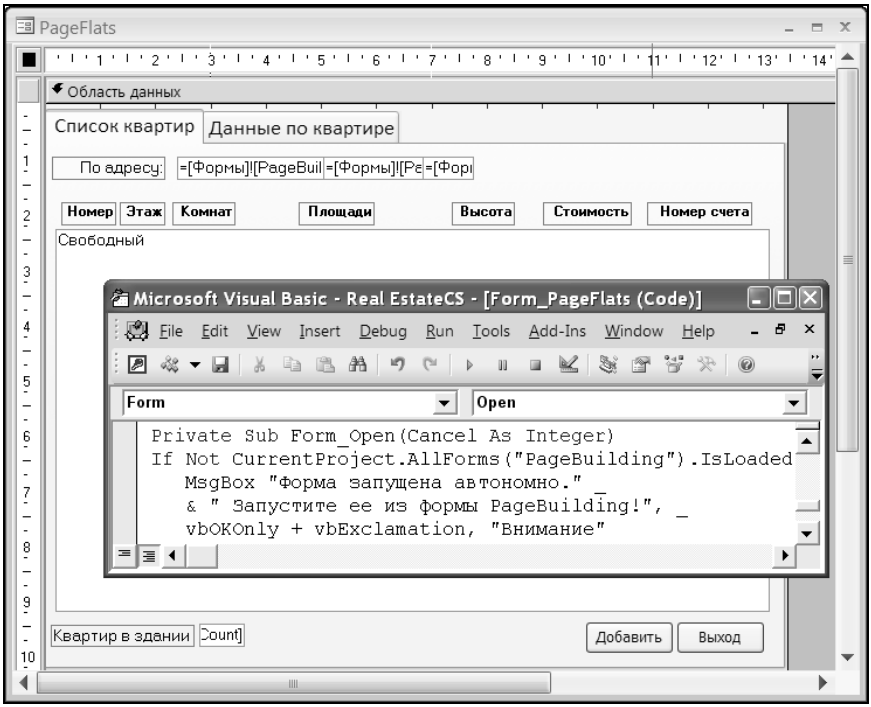


Рис. 14.22. Первая страница формы `PageFlats` в режиме конструктора

Таблица 14.2. *Список объектов первой страницы формы* `PageFlats`

Объект	Имя объекта	Пояснение	Данные
Поле	`txtStreet`	Название улицы	`=[Формы]![PageBuilding]![ComboStreet].[Column](1)`
Поле	`txtSign`	Признак адреса	`=[Формы]![PageBuilding]![ComboStreet].[Column](2)`
Поле	`txtHouse`	Номер дома	`=[Формы]![PageBuilding]![txtHouse].[Value]`
Список	`ListBox`	Список всех квартир в здании и их основные характеристики	Код VBA (см. листинг 14.14)
Поле	`txtCountFlat`	Количество квартир в здании	`=[ListBox].[ListCount]`

Таблица 14.2 *(окончание)*

Объект	Имя объекта	Пояснение	Данные
Кнопка	commandAdd	Занесение новой квартиры	Код VBA
Кнопка	commandExit	Завершение работы	Код VBA

В состав запроса для свойства ListBox.RowSource включены только те поля таблицы tblFlats, которые отражают основную информацию по квартирам. Остальные поля работник увидит, открыв двойным щелчком мыши по выбранной квартире вторую страницу формы (листинг 14.18).

Листинг 14.18. Двойной щелчок мышью по строчке списка ListBox с выбранной квартирой

```
Private Sub ListBox_DblClick(Cancel As Integer)
' Код события "Двойное нажатие кнопки"
Dim TxtSQL As String ' Строка запроса
IndFlats = 2         ' Режим корректировки включен

' Номер выбранной улицы: SelectAddressStreet
' Номер выбранного дома: SelectAddressHouse
' Номер выбранной квартиры
SelectAddressFlat = [ListBox].Column(0)
' Строка запроса
TxtSQL = "SELECT * FROM tblFlats WHERE STREET = " & _
    SelectAddressStreet & " AND HOUSE = " & SelectAddressHouse & _
    " AND FLAT = " & SelectAddressFlat
' Источник данных формы — теперь этот запрос
Me.RecordSource = TxtSQL
' Вторая вкладка видима и активна
Page2.Visible = True
Page2.SetFocus
' Номер квартиры изменению не подлежит
txtFLAT.Enabled = False
txtFLAT.Locked = True
' Номер лицевого счета не корректируется
txtACCOUNT.Enabled = False
txtACCOUNT.Locked = True
' Список проживающих доступен
Owners.Enabled = True
End Sub
```

На рис. 14.23 представлена первая страница формы PageFlats. Для второй страницы формы значение свойства **Вывод на экран** было изменено нами ранее на *Нет*, поэтому на рисунке она не видна.

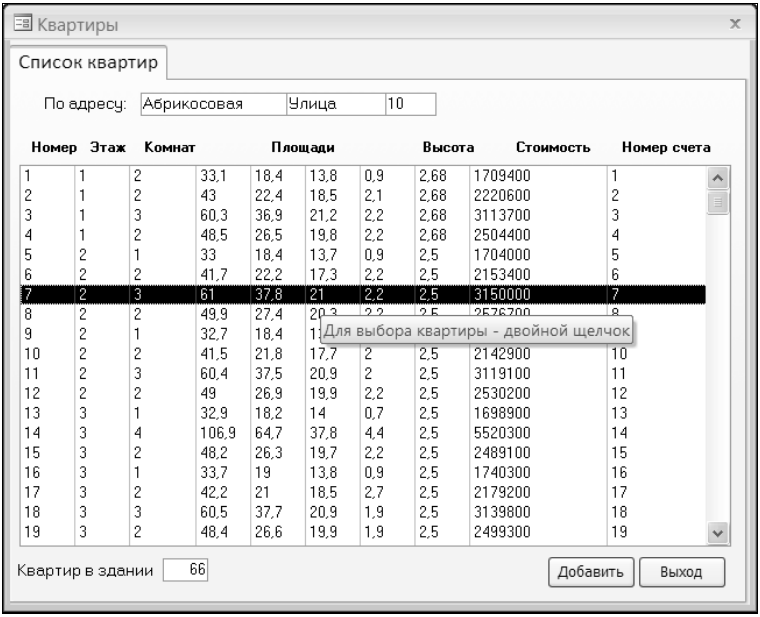


Рис. 14.23. Список квартир в здании (первая страницы формы `PageFlats`)

Для занесения новой квартиры предназначена кнопка **Добавить**. Код события **Нажатие кнопки** приведен в листинге 14.19.

Листинг 14.19. Щелчок по кнопке *Добавить*

```
Private Sub CommandAdd_Click()
' Кнопка Добавить
Dim TextSQL As String ' Строка запроса
IndFlats = 1          ' Режим добавления включен
' Вывод на экран второй вкладки
Page2.Visible = True
' Изменение заголовка второй вкладки
Page2.Caption = "Занесение новой квартиры"
' Кнопка Документ недоступна
CommandDoc.Enabled = False
' Кнопка Договор недоступна
CommandTreaty.Enabled = False
' Кнопка Счет недоступна
CommandAccount.Enabled = False
' Кнопка Удалить недоступна
CommandDel.Enabled = False
' Кнопка Владелец недоступна
CommandMain.Enabled = False
```

```
' Номер квартиры доступен
txtFLAT.Enabled = True
txtFLAT.Locked = False
' Номер лицевого счета доступен
txtACCOUNT.Enabled = True
txtACCOUNT.Locked = False
' Список проживающих недоступен
Owners.Enabled = False
' Добавление новой записи
DoCmd.GoToRecord , , acNewRec
' Передача управления полю с номером квартиры
txtFLAT.SetFocus
' Первая вкладка невидима
Page1.Visible = False
End Sub
```

В коде блокируется подчиненная форма проживающих в квартире `Owners` и открывается доступ к полям номера квартиры и лицевого счета. Для этого используется связка из двух операторов по каждому объекту, например для номера лицевого счета:

```
txtACCOUNT.Enabled = True
txtACCOUNT.Locked = False
```

Примечание

Если использовать только `Enabled` (`False`/`True`), то объект, лишенный доступа, будет выделен блеклым фоном. Применение `Locked` снимает выделение, оставляя объект недоступным.

Рассмотрим алгоритм, обеспечивающий работу конечного пользователя со второй страницей формы `PageFlats`, представленной на рис. 14.24.

Прежде всего, обратите внимание на то, что подчиненная форма для работы с данными по владельцу/квартиросъемщику `SlaveAccount` исчезла из основной формы. Она теперь запускается кнопкой **Владелец**. Это объясняется простой причиной. Лицевыми счетами (финансовый документ) всегда занимается специальный отдел — бухгалтерия, поэтому другим пользователям просмотр его запрещен. Именно эта кнопка для других категорий пользователей и будет погашена при загрузке формы `PageFlats` (см. листинг 14.17):

```
CommandMain.Enabled = AccountWork
```

Глобальная переменная `AccountWork` получает значение `False`/`True` при регистрации работника в самом начале работы программного комплекса.

Режим работы второй страницы формы `PageFlats` определяется значением переменной уровня модуля `IndFlats`. Переменная может принимать только два значения (1 — занесение новой квартиры, 2 — корректировка). Кнопка **Сохранить**

доступна при работе в любом из режимов, только действия выполняет разные (листинг 14.20).

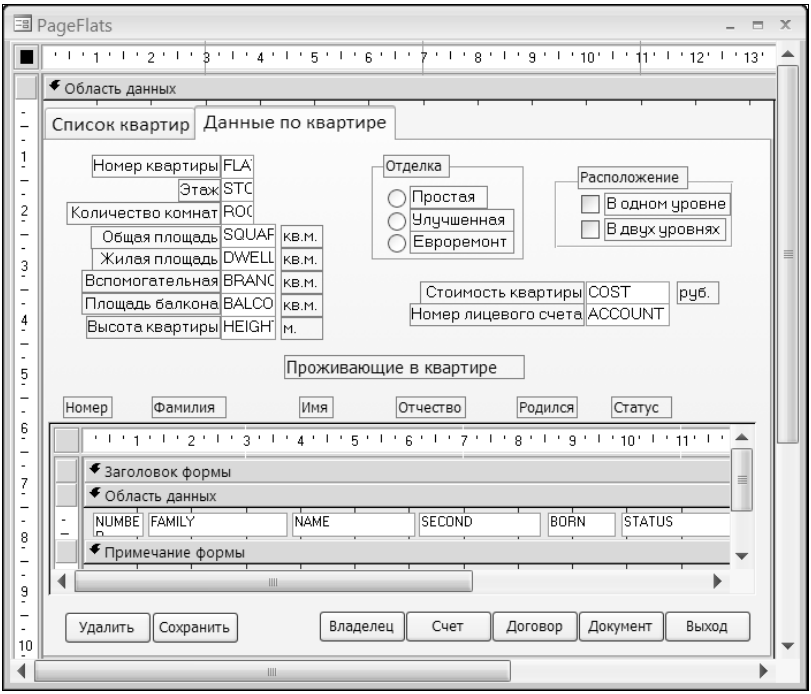


Рис. 14.24. Вид второй страницы формы `PageFlats` в конструкторе

Листинг 14.20. Код события *Нажатие кнопки* для `CommandSave`

```
Private Sub CommandSave_Click()
' Кнопка Сохранить
On Error GoTo SqlUpdateErr
If IsNull(txtFLAT.Value) Then
   ' Номер квартиры не введен
   MsgBox "Вы забыли номер квартиры.", _
       vbOKOnly + vbExclamation, "Внимание"
   ' Установка курсора в поле номера квартиры
   txtFLAT.SetFocus
   ' Возврат в форму
   Exit Sub
End If
If IsNull(txtSQUAREFLAT.Value) Then
   MsgBox "Вы забыли про общую площадь квартиры.", _
       vbOKOnly + vbExclamation, "Внимание"
```

```
   txtSQUAREFLAT.SetFocus
   Exit Sub
End If
If IsNull(txtDWELL.Value) Then
   MsgBox "Вы забыли про жилую площадь квартиры.", _
       vbOKOnly + vbExclamation, "Внимание"
   txtDWELL.SetFocus
   Exit Sub
End If
If IsNull(txtBRANCH.Value) Then
   MsgBox "Вы забыли про вспомогательную площадь квартиры.", _
       vbOKOnly + vbExclamation, "Внимание"
   txtBRANCH.SetFocus
   Exit Sub
End If
If IsNull(txtBALCONY.Value) Then
   MsgBox "Вы забыли про площадь балкона.", _
       vbOKOnly + vbExclamation, "Внимание"
   txtBALCONY.SetFocus
   Exit Sub
End If
If IsNull(txtACCOUNT.Value) Then
   ' Номер лицевого счета не введен
   MsgBox "Вы забыли номер лицевого счета.", _
       vbOKOnly + vbExclamation, "Внимание"
   ' Установка курсора в поле номера счета
   txtACCOUNT.SetFocus
   ' Возврат в форму
   Exit Sub
End If
If IndFlats = 1 Then
      ' Режим добавления
      ' Имеется ли такой номер квартиры в базе?
      If DCount("*", "tblFlats", _
         "STREET = " & SelectAddressStreet & _
         "AND HOUSE = " & SelectAddressHouse & _
         "AND FLAT = " & txtFLAT.Value) > 0 Then
         MsgBox "Квартира с таким номером уже есть!", _
                vbOKOnly + vbExclamation, "Внимание"
         ' Установка курсора в поле номера квартиры
         txtFLAT.SetFocus
         ' Возврат в форму
         Exit Sub
      End If
```

```
      If MsgBox("Добавить новую запись в базу данных? ", _
         vbInformation + vbYesNo, "Сохранение") = vbNo Then
         DoCmd.RunCommand acCmdUndo
      End If
      ' Сохранение записи без закрытия формы
      DoCmd.RunCommand acCmdSaveRecord
      ' Первая вкладка (Список квартир) видима
      Page1.Visible = True
      Page1.SetFocus
      ' Гашение второй вкладки
      Page2.Visible = False
      ' Изменение заголовка второй вкладки
      Page2.Caption = "Данные по квартире"
      ' Кнопка Удалить доступна
      CommandDel.Enabled = True
      ' Кнопка Документ доступна
      CommandDoc.Enabled = True
      ' Кнопка Договор доступна
      CommandTreaty.Enabled = True
      ' Кнопка Счет доступна
      CommandAccount.Enabled = True
      ' Кнопка Удалить доступна
      CommandDel.Enabled = True
      ' Кнопка Владелец доступна в соответствии с правами
      CommandMain.Enabled = AccountWork
      Page1.SetFocus
End If
If IndFlats = 2 Then
   If Me.Dirty Then
      ' Режим корректировки
      If MsgBox("Сохранить сделанные изменения " & _
         "по техническим характеристикам квартиры? ", _
         vbInformation + vbYesNo, "Сохранение") = vbYes Then
        ' Сохранение записи без закрытия формы
        DoCmd.RunCommand acCmdSaveRecord
      End If
   End If
   Page1.SetFocus
   ' Обновление данных в списке квартир
   ListBox.Requery
End If
Exit Sub

SqlUpdateErr:
  Select Case Err.NUMBER
```

```
    Case 2627
         ' Такой лицевой счет уже есть
         MsgBox "Лицевой с таким номером уже есть.", _
         vbOKOnly + vbExclamation, "Внимание"
         txtACCOUNT.SetFocus
         Exit Sub
    Case 50001
         ' Несоответствие площадей
         MsgBox "Несоответствие площади квартиры
                 и ее составляющих.", _
                 vbOKOnly + vbExclamation, "Внимание"
         Exit Sub
    Case Else
         ' Другие ошибки
          MsgBox Err.Description
  End Select
End Sub
```

Перед сохранением записи выполняется ряд проверок. Они просто необходимы. В режиме добавления новой записи проверяется наличие в базе данных квартиры с введенным номером. Если квартира имеется в базе, то данные по ней можно только редактировать. В коде демонстрируются два способа, позволяющие добиться корректной работы. Первый — предотвращение возникновения ошибки с использованием стандартной функции `DCount()`:

```
If DCount("*", "tblFlats", _
         "STREET = " & SelectAddressStreet & _
         "AND HOUSE = " & SelectAddressHouse & _
         "AND FLAT = " & txtFLAT.Value) > 0 Then
         MsgBox "Квартира с таким номером уже есть!", _
                vbOKOnly + vbExclamation, "Внимание"
         ' Установка курсора в поле номера квартиры
         txtFLAT.SetFocus
         ' Возврат в форму
         Exit Sub
End If
```

Второй — перехват уже свершившегося события (при дублировании номера лицевого счета) сработает ограничение внешнего ключа `FK_tblFlats_tvlAccount` таблицы `tblFlats`:

```
Select Case Err.NUMBER
    Case 2627
         ' Такой лицевой счет уже есть
         MsgBox "Лицевой с таким номером уже есть.", _
         vbOKOnly + vbExclamation, "Внимание"
         txtACCOUNT.SetFocus
         Exit Sub
```

Второй способ надежнее. За корректность данных в базе всегда должен отвечать сервер. Перекладывать эту ответственность на клиентское приложение — большая ошибка. Первый (упреждающий) способ приведен здесь только для примера. Если эту проверку не выполнить, то сработает ограничение первичного ключа `PK_tblFlats` таблицы `tblFlats`, и дублирующая запись все равно не попадет в базу.

В обоих режимах проверяется соответствие площадей квартиры:

```
[SQUAREFLAT]=[DWELL]+[BRANCH]+[BALCONY]
```

Эту проверку выполняет триггер таблицы `tblFlats`, текст которого приведен в листинге 14.2. Однако он не сработает, если хотя бы одно из этих четырех значений будет `Null`. Исправим эту ошибку:

1. Запустите SQL Server Management Studio.
2. Появится окно **Connect to Server**. Выберите требуемый сервер, режим аутентификации, имя пользователя и пароль. Нажмите кнопку **Connect**.
3. В окне обозревателя объектов **Object Explorer** откройте объект **Databases**, а затем узел требуемой базы данных.
4. Раскройте узел **Tables**. Выберите таблицу `tblFlats`. Сделайте щелчок правой кнопкой мыши. Появится контекстное меню.
5. Выберите в нем пункт **Modify**. Появится диалоговое окно (рис. 14.25). Снимите флажки в колонке **Allow Nulls** для полей `SQUAREFLAT`, `DWELL`, `BRANCH` и `BALCONY`.

Тем не менее, четыре проверки не будем убирать из теста программы — пусть пишут по-русски, т. к. англоязычные сообщения очень плохо принимаются конечным пользователем. Вот одна из них:

```
If IsNull(txtSQUAREFLAT.Value) Then
   MsgBox "Вы забыли про общую площадь квартиры.", _
       vbOKOnly + vbExclamation, "Внимание"
   txtSQUAREFLAT.SetFocus
   Exit Sub
End If
```

Для удаления данных по квартире служит кнопка `CommandDel`, код обработки нажатия которой представлен в листинге 14.21.

Листинг 14.21. Код события *Нажатие кнопки* для `CommandDel`

```
Private Sub CommandDel_Click()
' Кнопка Удалить
Dim TextSQL As String ' Строка запроса
```

```
' При ошибке перейти к метке: SqlUpdateErr
On Error GoTo SqlUpdateErr

' Номер выбранной улицы: SelectAddressStreet
' Номер выбранного дома: SelectAddressHouse
' Номер выбранной квартиры
SelectAddressFlat = [ListBox].Column(0)
TextSQL = "DELETE from tblFlats WHERE STREET = " & _
          SelectAddressStreet & _
          " AND HOUSE = " & SelectAddressHouse & _
          " AND FLAT = " & SelectAddressFlat
If MsgBox("Вы действительно хотите удалить эту квартиру? ", _
   vbInformation + vbYesNo, "Удаление") = vbYes Then
   ' Выполнение запроса
   DoCmd.RunSQL TextSQL
   Page1.SetFocus
   ListBox.Requery
   Exit Sub
End If
Exit Sub
SqlUpdateErr:
  Select Case Err.NUMBER
    Case 2501
      ' Отказ пользователя от удаления квартиры
      MsgBox "Удаление квартиры не выполнено.", _
      vbOKOnly + vbExclamation, "Внимание"
    Case 44445
      ' Ошибка срабатывания триггера удаления
      ' Смотри текст триггера T_tblFlats_DTrig
      MsgBox "Квартиру удалить нельзя. В ней есть проживающие!", _
          vbOKOnly + vbExclamation, "Внимание"
    Case Else
      ' Другие ошибки
      MsgBox Err.Description
  End Select
End Sub
```

На рис. 14.26 показан вид второй страницы формы `PageFlats` в режиме отображения информации по квартире. Кнопка **Владелец** для работника недоступна, т. к. лицевой счет — не его компетенция. Последнее событие, которое непременно следует отследить, это событие **До обновления** подчиненной формы `Owners` (листинг 14.22). Эта форма работает со своей таблицей, и если этого не сделать, то все изменения пользователя, связанные с проживающими в квартире, попадут в базу данных без предупреждения.

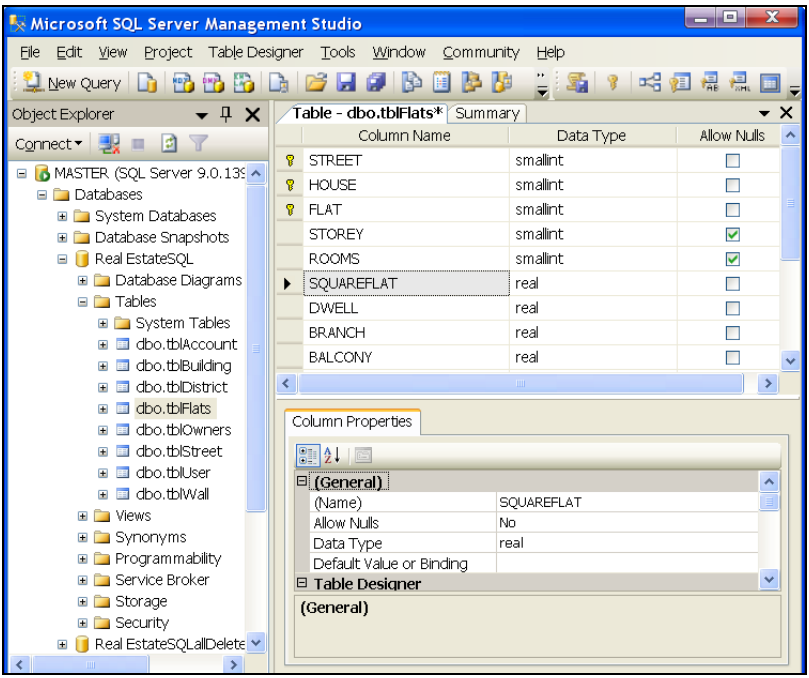


Рис. 14.25. Работа со свойством **Allow Nulls** полей таблицы `tblFlats`

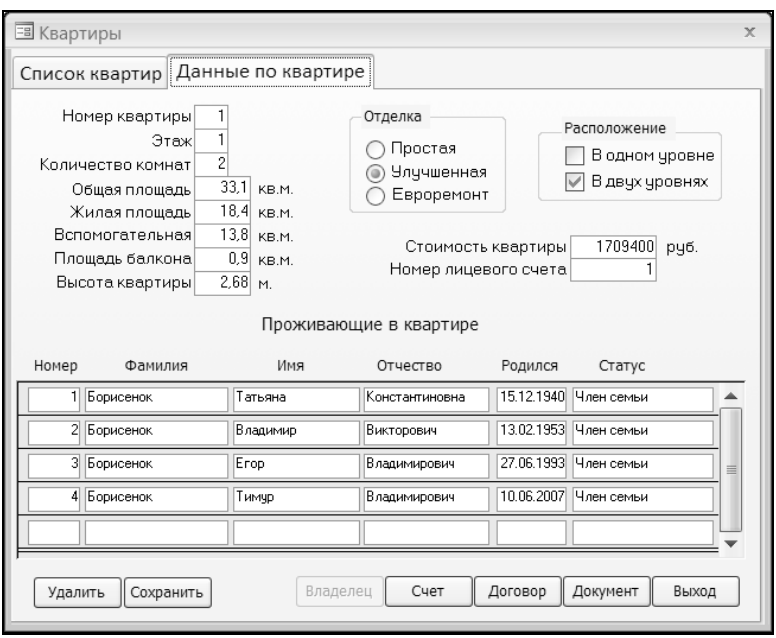


Рис. 14.26. Работа со второй страницей формы `PageFlats`

Листинг 14.22. Код события *До обновления* подчиненной формы `Owners`

```
Private Sub Form_BeforeUpdate(Cancel As Integer)
 If MsgBox("Сохранить сделанные изменения по проживающим? ", _
      vbInformation + vbYesNo, "Сохранение") = vbNo Then
      DoCmd.RunCommand acCmdUndo
    End If
End Sub
```

14.6. Доработка запросов

Мастер преобразования MS Access 2010, как и его предшественники, не особенно силен в преобразовании запросов, тем не менее следует отдать должное его разработчикам. Определенный прогресс в этой области имеется. Все меньше типов конструкций остается вне пределов его компетенции с выходом каждой новой версии продукта. Доведем до рабочего состояния конструкции двух запросов MS Access, не попавших в базу данных MS SQL Server 2008.

14.6.1. Доработка запросов с параметрами

В *главе 3* нами был создан запрос, на основе которого базируется отчет MS Access 2010. Для его создания используется информация из другого запроса и двух таблиц базы данных. Кроме этого, запрос содержит параметры и запускается из формы. Работу мастера усложнило присутствие трех полей, имеющих одинаковое имя `Name` (название улицы, имя владельца квартиры, имена проживающих). В листинге 14.23 приведен текст запроса MS Access. Листинг 14.24 содержит текст на Transact-SQL, сгенерированный мастером и признанный MS SQL-сервером непригодным для выполнения.

Листинг 14.23. Текст исходного запроса Access на языке SQL

```
SELECT street.NAME, street.SIGN, flats.HOUSE, flats.FLAT,
       flats.STOREY, flats.ROOMS, flats.SQUAREFLAT, flats.DWELL,
       flats.BRANCH, flats.BALCONY,
       flats.HEIGHT, flats.ACCOUNT, flats.FAMILY, flats.NAME,
       flats.SECOND, owners.NUMBER, owners.FAMILY, owners.NAME,
       owners.SECOND, owners.BORN, owners.STATUS,street.FIRST
  FROM street INNER JOIN (flats INNER JOIN owners ON
       (flats.HOUSE = owners.HOUSE)
       AND (flats.FLAT = owners.FLAT)
       AND (flats.STREET = owners.STREET))
           ON street.STREET = flats.STREET
  WHERE (((flats.FLAT)=[Forms]![Flats]![Flat]))
  ORDER BY owners.NUMBER
```

Листинг 14.24. Результат работы мастера преобразования (фрагмент файла-отчета, выданного в процессе конвертации)

```
Имя запроса:  qwrDocument
Ошибка преобразования. Попытка использования SQL:

CREATE FUNCTION qwrDocument
  (@Forms_Building_Street1 varchar (255),
   @Forms_Building_House2 varchar (255),
   @Forms___Flats___Flat3 varchar (255))
RETURNS TABLE
AS RETURN (SELECT TOP 100 PERCENT
tblStreet.NAME AS NAME2,tblStreet.SIGN,
qwrFlats.HOUSE, qwrFlats.FLAT,qwrFlats.STOREY,
qwrFlats.ROOMS,qwrFlats.SQUAREFLAT,
qwrFlats.DWELL,qwrFlats.BRANCH, qwrFlats.BALCONY,
qwrFlats.HEIGHT,qwrFlats.ACCOUNT AS "_qwrFlats.ACCOUNT_",
qwrFlats.FAMILY,qwrFlats.NAME,qwrFlats.SECOND,
tblOwners.NUMBER, tblOwners.FAMILY AS FAMILY1,
tblOwners.NAME AS NAME1,tblOwners.SECOND AS SECOND1,
tblOwners.BORN,tblOwners.STATUS,
FROM (qwrFlats INNER JOIN tblStreet ON
     (qwrFlats.STREET=tblStreet.STREET)) INNER
     JOIN tblOwners ON (qwrFlats.STREET=tblOwners.STREET)
     AND (qwrFlats.FLAT=tblOwners.FLAT) AND
    (qwrFlats.HOUSE=tblOwners.HOUSE)
WHERE (((qwrFlats.FLAT)=@Forms___Flats___Flat3))
```

Мастер попытался создать пользовательскую функцию `qwrDocument` типа **Table-valued Functions**, которая имеет три параметра:

```
(@Forms_Building_Street1 varchar (255), -- номер улицы
 @Forms_Building_House2 varchar (255),  -- номер дома
 @Forms___Flats___Flat3 varchar (255))  -- номер квартиры
```

и возвращает в точку вызова таблицу значений, которые и служат основой для создания отчета MS Access.

С совпадающими именами полей мастер прекрасно справился, но зачем-то поставил запятую перед служебным словом `FROM`, не закрыл все скобки в конце конструкции и в результате получил вердикт SQL-сервера:

```
Msg 216, Level 16, State 1, Procedure qwrDocument, Line 2
Parameters were not supplied for the function 'qwrFlats'.
```

Не переданы параметры в функцию `qwrFlats`, на основе которой создана функция `qwrDocument`, а это означает, что конвертировать запрос с использованием алго-

ритма, выбранного мастером и основанного на функции `qwrFlats`, при современном положении дел не представляется возможным.

Возьмем текст, сгенерированный мастером, за основу и создадим хранимую процедуру с тремя параметрами (номер улицы, номер дома и номер квартиры), которая возвращает выборку из четырех таблиц (улицы, квартиры, проживающие и лицевой счет). Для этого выполним следующие действия:

1. Запустим SQL Server Management Studio.
2. Появится окно **Connect to Server**. Выберем требуемый сервер, режим аутентификации, имя пользователя и пароль. Нажмем кнопку **Connect**.
3. В окне обозревателя объектов **Object Explorer** откроем объект **Databases**, а затем узел требуемой базы данных.
4. Откроем объект **Programmability**.
5. Сделаем щелчок правой кнопкой мыши по узлу **Stored Procedures**. Появится контекстное меню. Выберем в нем пункт **New Stored Procedure...**. Появится окно с заготовкой текста хранимой процедуры.
6. Введем ее текст, приведенный в листинге 14.25.

Листинг 14.25. Текст хранимой процедуры `dbo.Document`

```
SET ANSI_NULLS ON
SET QUOTED_IDENTIFIER ON
GO

CREATE PROCEDURE [dbo].[Document]
       (@StreetSql smallint, -- Номер улицы
        @HouseSql smallint,  -- Номер дома
        @FlatSql smallint)   -- Номер квартиры
AS SELECT dbo.tblFlats.STREET,
          dbo.tblStreet.NAME AS NameST,dbo.tblStreet.SIGN,
          dbo.tblFlats.HOUSE,dbo.tblFlats.FLAT,
          dbo.tblFlats.STOREY, dbo.tblFlats.ROOMS,
          dbo.tblFlats.SQUAREFLAT, dbo.tblFlats.DWELL,
          dbo.tblFlats.BRANCH,dbo.tblFlats.BALCONY,
          dbo.tblFlats.HEIGHT, dbo.tblFlats.ACCOUNT,
          dbo.tblAccount.FAMILY, dbo.tblAccount.NAME AS Expr1,
          dbo.tblAccount.SECOND, dbo.tblOwners.NUMBER,
          dbo.tblOwners.FAMILY AS Expr2,
          dbo.tblOwners.NAME AS Expr3,
          dbo.tblOwners.SECOND AS Expr4, dbo.tblOwners.BORN,
          dbo.tblOwners.STATUS,dbo.tblStreet.FIRST
FROM  dbo.tblStreet INNER JOIN dbo.tblFlats
      ON dbo.tblStreet.STREET = dbo.tblFlats.STREET
      INNER JOIN dbo.tblOwners ON
```

```
      dbo.tblFlats.STREET = dbo.tblOwners.STREET AND
      dbo.tblFlats.HOUSE = dbo.tblOwners.HOUSE AND
      dbo.tblFlats.FLAT = dbo.tblOwners.FLAT INNER JOIN
      dbo.tblAccount ON
      dbo.tblFlats.ACCOUNT = dbo.tblAccount.ACCOUNT
WHERE (dbo.tblFlats.STREET = @StreetSql) AND
      (dbo.tblFlats.HOUSE = @HouseSql) AND
      (dbo.tblFlats.FLAT = @FlatSql)
ORDER BY dbo.tblOwners.NUMBER
```

Для проверки правильности работы хранимой процедуры `dbo.Document` сделаем щелчок мышью по кнопке **New Query**, расположенной на панели инструментов. Появится окно. Наберем в нем текст:

```
EXECUTE dbo.Document @StreetSql=2,@HouseSql=10,@FlatSql=67
```

В этой строке: `EXECUTE` — служебное слово, `@StreetSQL=2` — номер улицы, на которой расположено здание, `@HouseSQL=10` — номер дома, `@FlatSql=67` — номер квартиры, для которой выдается справка. После щелчка по кнопке **! Execute** на экране появится окно с двумя вкладками (рис. 14.27). Хранимая процедура работает правильно.

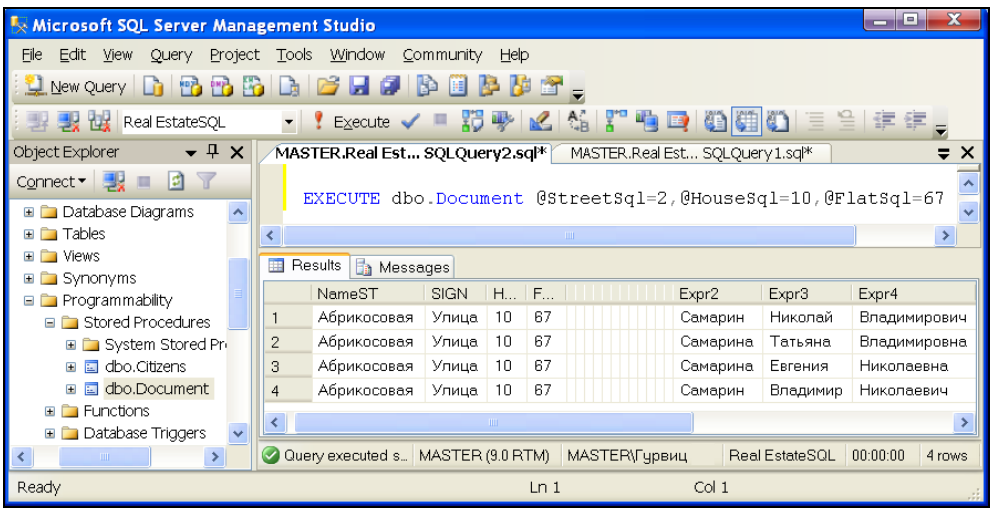


Рис. 14.27. Результаты работы хранимой процедуры `dbo.Document`

14.6.2. Доработка подчиненных запросов

В состав оператора `SELECT`, формирующего запрос к базе данных, могут входить подзапросы, результат выполнения которых используется для определения окон-

чательного результата внешнего запроса. Подзапросы могут входить в состав конструкций WHERE и HAVING внешнего запроса и иметь несколько уровней вложения.

В *главе 2* мы воспользовались этой возможностью языка Access SQL для поиска квартир в таблице flat, в которых нет проживающих. Текст запроса с добавлением приставки "tbl" перед именами таблиц Flats, Street и Owners приведен в листинге 14.26.

Листинг 14.26. Выборка пустующих квартир на языке Access SQL

```
SELECT tblStreet.NAME, tblStreet.SIGN, tblFlats.HOUSE,
       tblFlats.FLAT, tblFlats.ROOMS
FROM tblStreet INNER JOIN tblFlats ON
     tblStreet.STREET=tblFlats.STREET
WHERE NOT EXISTS
     (SELECT * FROM tblOwners
               WHERE STREET=tblFlats.STREET AND
                     HOUSE=tblFlats.HOUSE AND
                     FLAT=tblFlats.FLAT)
   ORDER BY NAME, HOUSE, FLAT;
```

Эта конструкция, так же как и предыдущая, преобразована мастером в пользовательскую функцию qwrNoOwners типа **Table-valued Functions** без параметров. Фрагмент отчета, содержащий ее текст, представлен в листинге 14.27.

Листинг 14.27. Результат работы мастера

```
Имя запроса: qwrNoOwners
Ошибка преобразования. Попытка использования SQL:

CREATE FUNCTION qwrNoOwners ()
RETURNS TABLE
AS RETURN (SELECT TOP 100 PERCENT tblStreet.NAME,
tblStreet.SIGN, tblFlats.HOUSE,
tblFlats.FLAT, tblFlats.ROOMS,NAME,HOUSE,FLAT
FROM tblStreet INNER JOIN tblFlats ON (tblStreet.STREET=tblFlats.STREET)
WHERE NOT EXISTS (SELECT * FROM tblOwners
WHERE STREET=tblFlats.STREET AND
HOUSE=tblFlats.HOUSE AND FLAT=tblFlats.FLAT)
ORDER BY NAME,HOUSE,FLAT)
```

MS SQL Server отказался выполнить конструкцию, преобразованную мастером, сославшись на неизвестные ему объекты: NAME, HOUSE и FLAT (в листинге 14.27 они выделены полужирным). Как они попали в состав запроса — остается только га-

дать! После их удаления функция `qwrNoOwners`, представленная в листинге 14.28, выдала правильный результат.

Листинг 14.28. Текст функции `qwrNoOwners` после исправления

```
CREATE FUNCTION [dbo].[qwrNoOwners]()
RETURNS TABLE
AS RETURN (SELECT TOP 100 PERCENT tblStreet.NAME,
           tblStreet.SIGN, tblFlats.HOUSE,
           tblFlats.FLAT, tblFlats.ROOMS
     FROM tblStreet INNER JOIN tblFlats ON
         (tblStreet.STREET=tblFlats.STREET)
     WHERE NOT EXISTS
           (SELECT * FROM tblOwners
                WHERE STREET=tblFlats.STREET AND
                      HOUSE=tblFlats.HOUSE AND
                      FLAT=tblFlats.FLAT)
                ORDER BY NAME,HOUSE,FLAT)
```

В отличие от хранимой процедуры, для запуска которой применяется служебное слово `EXECUTE`, для запуска функции `qwrNoOwners` необходимо воспользоваться конструкцией `SELECT`. После имени функции в скобках — пустой список параметров:

```
SELECT * FROM qwrNoOwners()
```

14.7. Исправление отчета

При запуске на выполнение отчета `Document` из формы `Flats`, перенесенного мастером преобразования в файл проекта Real EstateCS, на экране одна за другой появляются две ошибки. Первая сообщает об отсутствии значений параметров `StreetSql`, `HouseSql` и `FlatSql`, а вторая говорит о том, что функции не нравится столбец `tblStreet` (рис. 14.28).

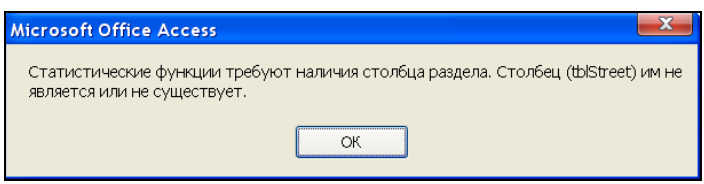


Рис. 14.28. Сообщение об ошибке при запуске отчета на выполнение

С первой ошибкой мы уже встречались. Объекту MS Access (в данном случае это отчет) требуется номер улицы, на которой расположено здание, а также номер

дома и квартиры. Исправить такую ошибку достаточно просто. Укажите в окне свойств отчета для свойства **Входные параметры** (рис. 14.29) строку:

```
@StreetSql SmallInt=[Forms]![Building]![Street],
@HouseSql SmallInt=[Forms]![Building]![House],
@FlatSql SmallInt=[Forms]![Flats]![Flat]
```

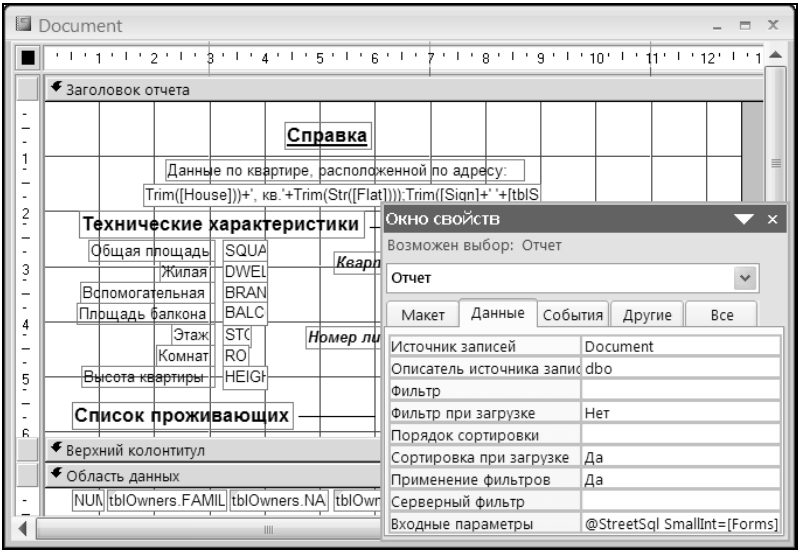


Рис. 14.29. Доработка отчета `Document` в режиме конструктора

О какой же функции заявляет вторая ошибка? Догадаться не трудно. В нашем отчете функций не так уж много. Виновником ошибки является вычисляемое поле `ADDRESS`, в свойстве **Данные** которого используется:

```
=IIf([First]=Истина;Trim([tblStreet.Name]+' '+[Sign]+
', дом '+(Trim([House]))+', кв.'+Trim(Str([Flat])));Trim([Sign]+
' '+[tblStreet.Name])+', дом '+(Trim([House]))+', кв.'+Trim(Str([Flat])))
```

Заменим `tblStreet.Name` на псевдоним `NameST`, который мы указали в тексте хранимой процедуры сервера `dbo.Document` (см. листинг 14.25):

```
=IIf([First]=Истина;Trim([NameST]+' '+[Sign]+', дом '+(Trim([House]))+
', кв.'+Trim(Str([Flat])));Trim([Sign]+' '+[NameST])+
', дом '+(Trim([House]))+', кв.'+Trim(Str([Flat])))
```

Выполним также замену значений в свойствах **Данные** для полей, отмеченных цветным треугольником в левом верхнем углу (рис. 14.29). Заменим `tblOwners.FAMILY`, `tblOwners.NAME` и `tblOwners.SECOND` на `Expr2`, `Expr3` и `Expr4` соответственно.

Третья ошибка проявит себя после запуска отчета на выполнение (рис. 14.30).

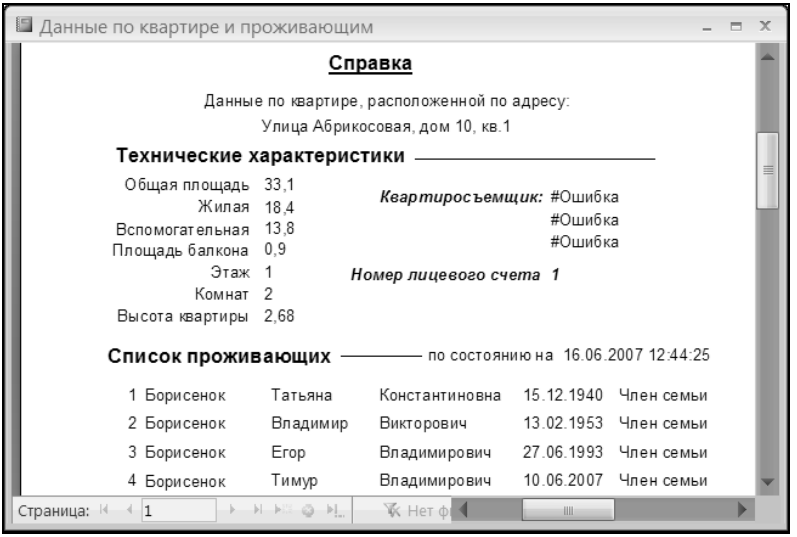


Рис. 14.30. Ошибка, возникшая во время выполнения отчета

Поля отчета ссылаются на объекты формы `Flats`, которые были при доработке перемещены в подчиненную форму `Account`:

```
=[Формы]![Flats]![FAMILY]
=[Формы]![Flats]![NAME]
=[Формы]![Flats]![SECOND]
```

Заменим эти три ссылки на `FAMILY`, `Expr1` и `SECOND`. Отчет готов к работе!

14.8. Включение в отчет суммы прописью

Многие финансовые документы, в соответствии с действующими стандартами, должны отображать величину суммы прописью. В связи с этим арсенал любого разработчика должен содержать инструмент, предоставляющий такую возможность. В листинге 14.29 приведен текст пользовательской функции, которая возвращает в точку вызова строку, соответствующую указанной сумме. Функция работает с суммами до триллиона рублей.

Листинг 14.29. Сумма прописью

```
Public Function LineNumeral(Chislo) As String
' Chislo — входной параметр
If Chislo > 999999999999# Then
   LineNumeral = "Слишком большая сумма"
   Exit Function
End If
```

```
Dim I, J, K, LK, NK, PART, PRS, FF, Z ' Рабочие переменные
Dim RES As String             ' Результат
Dim NN(1 To 4) As Integer     ' Составные части числа
Dim RSset As ADODB.Recordset  ' Набор данных из таблицы
Set RSset = New ADODB.Recordset
RSset.Open "tblNumeral", CurrentProject.Connection
' RSset(0)- первый столбец таблицы tblNumeral (прописью)
' RSset(1)- второй столбец таблицы tblNumeral (число)
Z = 0
If Int(Chislo) = 0 Then
   RES = "Ноль "
Else
   RES = ""
End If
PRS = Right(Space(12) + Str(Int(Chislo)), 12)  ' Строка из числа
NN(1) = Val(Mid(PRS, 1, 3))              ' Миллиарды
NN(2) = Val(Mid(PRS, 4, 3))              ' Миллионы
NN(3) = Val(Mid(PRS, 7, 3))              ' Тысячи
NN(4) = Val(Mid(PRS, 10, 3))             ' Рубли
NK = Round((Chislo - Int(Chislo)) * 100) ' Копейки

For J = 1 To 4
    ' Цикл по миллиардам, миллионам, тысячам и единицам
    For I = 0 To 35
        ' Цикл по числительным
        ' Переход к I-й записи для извлечения слов:
        ' девятьсот ... девяносто ... десять и т. д.
        RSset.Move I, adBookmarkFirst
        FF = RSset(0)
        If (NN(J) >= RSset(1)) Then
           If J = 3 Then
              ' Только для тысяч
              If NN(J) = 2 Then
                 PART = "две "
              Else
                 If NN(J) = 1 Then
                    PART = "одна "
                 Else
                    PART = RTrim(FF) + " "
                 End If
              End If
           Else
              PART = RTrim(FF) + " "
           End If
           RES = RES + PART
           NN(J) = NN(J) - RSset(1)
```

```
            ' RSset(1) — число из второго столбца таблицы
            Z = RSset(1)
        End If
    Next I
    If Z > 4 Then
                K = 1
             Else
                If Z > 1 Then
                   K = 2
                Else
                   K = 3
                End If
    End If
    ' Переход к записи с вычисленным номером
    ' для извлечения слов: миллиард, миллион и т. д.
    RSset.Move 35 + (K + 3 * (J — 1)), adBookmarkFirst
    FF = RSset(0)
    If Z <> 0 Then
           RES = RES + RTrim(FF) + " "
    End If
    If J = 3 Then
       Z = 5
    Else
       Z = 0
    End If
Next J
' Работа с копейками
LK = LTrim(Str(NK))
If (Len(LK) < 2) Then
   LK = "0" + LK
End If
Select Case NK
   Case 1       ' Одна копейка
        K = 3
   Case 2 To 4  ' Две, три, четыре копейки
        K = 2
   Case Is > 20 ' Двадцать, тридцать и т. д.
        Z = NK — Int(NK / 10) * 10
        ' Z — последняя цифра копеек
        Select Case Z
           Case 1         ' Одна копейка
                K = 3
           Case 2 To 4    ' Две, три, четыре копейки
                K = 2
           Case Else      ' Пять, шесть, ..., девять копеек
                K = 1
        End Select
```

```
   Case Else
        K = 1
End Select
' Переход к записи с вычисленным номером
' для извлечения слов: копейка, копеек и т. д.
RSset.Move 47 + K, adBookmarkFirst
FF = RSset(0)
RES = RES + LK + " " + RTrim(FF) + "."
RSset.Close
Set RSset = Nothing
LineNumeral = RES
End Function
```

Функция `LineNumeral` использует данные, расположенные в таблице числительных `tblNumeral` (рис. 14.31). Скопируйте эту таблицу в свою базу данных и забудьте о ней. Она не участвует в связях с другими таблицами, данные в ней не модифицируются, и поэтому она может вообще остаться в первой нормальной форме. Более того, не создавайте для нее первичный ключ. Его создание повлечет за собой создание кластерного индекса, а он обязательно изменит порядок следования записей в таблице, и функция `LineNumeral` будет работать неправильно.

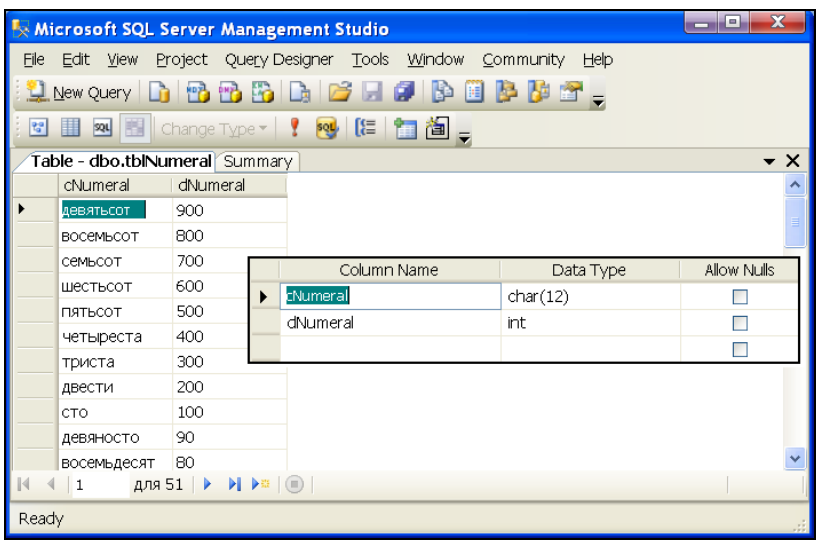


Рис. 14.31. Информация о таблице числительных

В таблице `tblNumeral` всего два столбца и 51 строка, которые содержат все служебные слова, необходимые для генерации любой суммы прописью. В современной литературе приведено множество вариантов, решающих пробле-

му суммы прописью, но вряд ли читатель найдет более короткий текст. К тому же, применяя таблицу числительных, хочу затронуть еще один очень важный аспект работы с таблицами MS SQL Server — перемещение по набору данных средствами ADO.

В тексте пользовательской функции в нужный момент времени из таблицы извлекается требуемая строчка. Для получения такой возможности сначала необходимо создать набор данных `Recordset` (имя этого набора — `RSset`):

```
Dim RSset As ADODB.Recordset  ' Набор данных из таблицы
Set RSset = New ADODB.Recordset
```

Затем извлечь в него записи из таблицы `tblNumeral`, используя соединение текущего проекта:

```
RSset.Open "tblNumeral", CurrentProject.Connection
```

А потом установить указатель на требуемую строку набора, применяя конструкцию:

```
RSset.Move <количество_записей>, <константа>
```

Для указания точки отсчета количества записей, на которое следует переместить указатель текущей записи, применяются три константы (табл. 14.3).

Таблица 14.3. *Значения констант метода* `Move`

Константа	Значение	Пояснение
adBookmarkCurrent	0	Точка отсчета — текущая запись
adBookmarkFirst	1	Точка отсчета — первая запись
adBookmarkLast	2	Точка отсчета — последняя запись

В тексте пользовательской функции `LineNumeral` всегда применяется вторая константа, и указатель "выставляется" от начала таблицы:

```
RSset.Move I, adBookmarkFirst
```

Если значение параметра количества записей больше нуля, то указатель смещается вперед, в противном случае — назад.

Предупреждение

Первая запись таблицы `tblNumeral` в наборе `RSset` станет нулевой, вторая — первой и т. д. Для того чтобы перейти к записи с номером 48, следует написать:

```
RSset.Move 47, adBookmarkFirst
```

Чтобы сослаться на значение поля `cNumeral`, используйте строчку:

```
RSset.Fields ("cNumeral").Value
```

В тексте пользовательской функции `LineNumeral` применяется более простой синтаксис:

```
<имя_переменной> = RSset(0) ' Для извлечения cNumeral
<имя_переменной> = RSset(1) ' Для извлечения dNumeral
```

Но, опять же, индекс первого поля (`cNumeral`) — ноль, второго (`dNumeral`) — один и т. д.

14.9. Работа с MS SQL Server 2008 средствами MS Access 2010

Для удобства разработки приложений компания Microsoft обеспечила интеграцию конструкторов MS Access 2010 с объектами MS SQL Server 2008. Разработчик клиентской части приложения может, не покидая MS Access, выполнить практически любые операции с серверными объектами, за исключением задач администрирования. Например, создать таблицу, изменить триггер, удалить представление, создать архивную копию базы данных или перенести ее на другой сервер и т. д.

В подавляющем числе случаев утилита администрирования SQL Server 2008 Management Studio разработчику не нужна.

14.9.1. Построение схемы данных

В MS Access схема данных — отправная точка работы над приложением. Работать со схемой средствами MS Access удобнее, чем с утилитой администрирования SQL Server 2008 Management Studio. Сказывается наглядность ленты MS Access 2010 и наличие дополнительных возможностей. Например, изменение связей между таблицами. SQL Server 2008 Management Studio предписывает сначала удалить старую и только потом создать новую, а MS Access предлагает корректировку существующей связи без каких-либо ограничений. Для построения схемы данных выполните следующие операции:

1. Откройте главное окно Microsoft Access 2010.
2. Выберите вкладку **Создание**. Перейдите в раздел **Макросы и код**.
3. В самом правом углу раздела расположен значок **Схема**. Щелкните по нему. Откроются окно построителя **Схема1** и окно добавления таблицы в схему данных.
4. Добавьте все таблицы, которые хотите видеть в схеме, и нажмите кнопку **Закрыть**.
5. Закройте окно **Схема1**. Появится запрос на ввод имени схемы.
6. Введите имя схемы и нажмите кнопку **OK**. Результаты работы конструктора представлены на рис. 14.32.

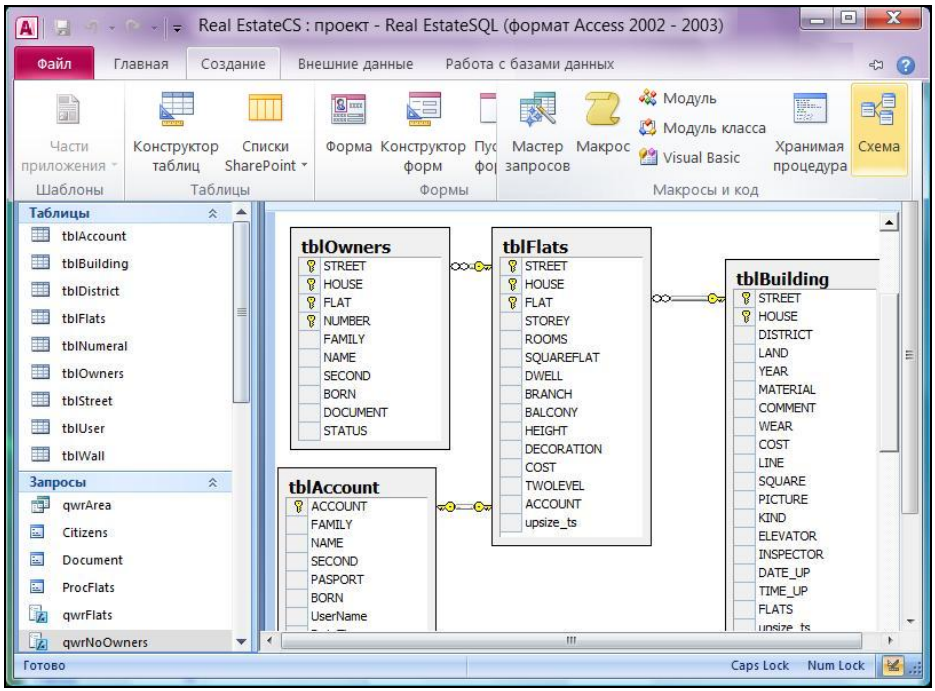


Рис. 14.32. Построение схемы данных средствами MS Access 2010

14.9.2. Таблицы, индексы, ключи и ссылочная целостность

Технология работы с таблицами MS SQL Server 2008 из MS Access выглядит так же, как и с его собственными. Сделайте щелчок правой кнопкой мыши по названию таблицы в области навигации, выберите пункт **Конструктор** и убедитесь в этом сами. Интерфейс работы с ключами и индексами также практически не изменился. При работе с этими объектами базы данных MS SQL Server всегда предлагается одно и то же диалоговое окно **Свойства**.

Вкладок у этого окна пять.

- **Таблицы**;
- **Отношения**;
- **Индексы и ключи**;
- **Ограничения проверки**;
- **Данные**.

На рис. 14.33 показана вкладка, отображающая отношения между таблицами `tblDistrict` и `tblBuilding`. Не удивляйтесь — декларативная ссылочная целост-

ность действительно не обеспечена, но ведь так и было задумано при конвертации базы MS Access в MS SQL Server (см. рис. 12.5). За "порядком" здесь следят триггеры этих таблиц (см. листинги 12.1—12.3), код которых также можно просмотреть из MS Access 2010. Найдите таблицу `tblBuilding` в области навигации. Пункт контекстного меню **Триггеры** приведет вас к цели. На рис. 14.33 окна свойств и триггеров для наглядности совмещены.

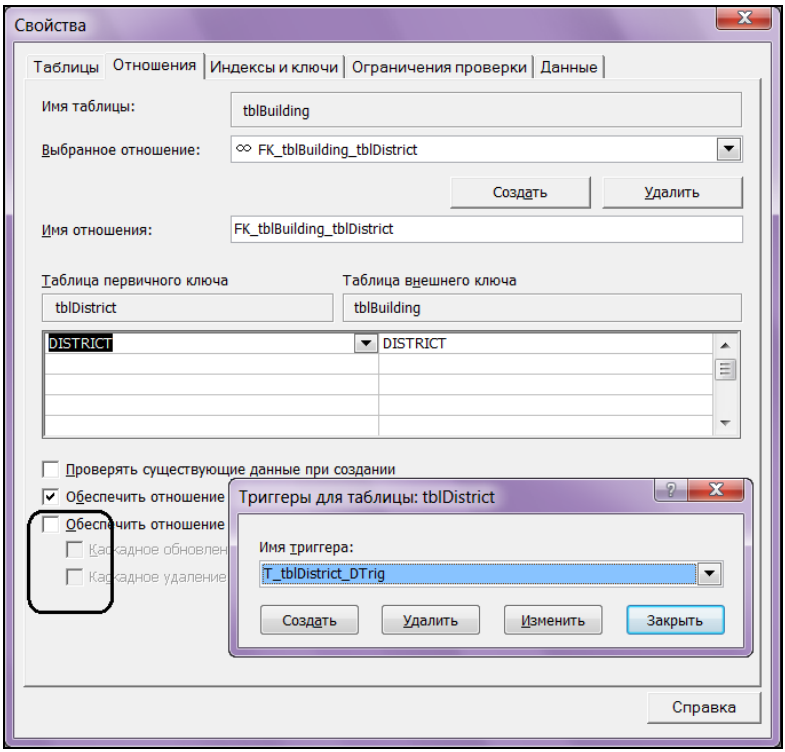


Рис. 14.33. Параметры связи между таблицами `tblDistrict` и `tblBuilding`

14.9.3. Конструктор пользовательской функции

Создать пользовательскую функцию типа **Table-valued Functions** средствами SQL Server 2008 Management Studio можно, вызвав соответствующий конструктор.

Для этого предназначена пиктограмма **Design Query in Editor**. Рассмотрим более простой способ. Откроем в конструкторе MS Access 2010 функцию `qwrFlats`, созданную мастером преобразования (см. рис. 14.6) и откорректированную нами в Management Studio. Она находится в разделе **Запросы** области переходов. Сразу запустится окно конструктора запросов (рис. 14.34). Вносите любые изменения.

Все они сразу будут отображены в нижней части окна, где представлен код Transact-SQL.

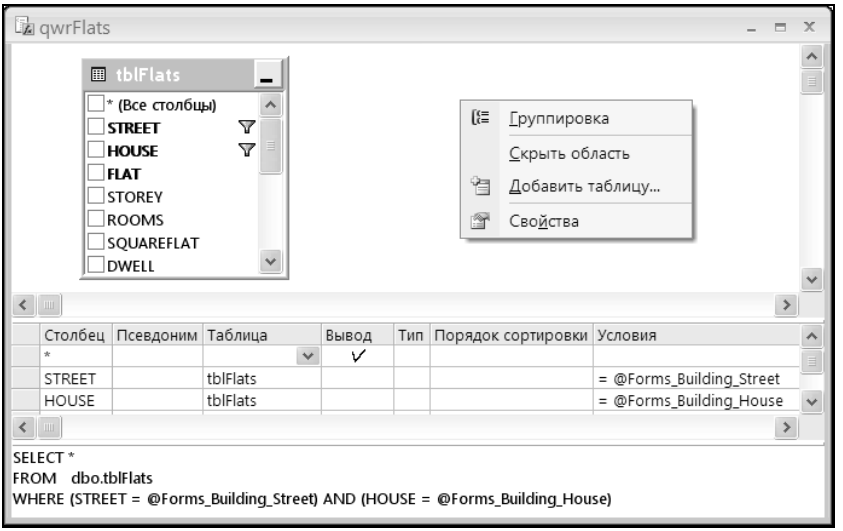


Рис. 14.34. Функция `qwrFlats` типа **Table-valued Functions** в конструкторе

Примечание

После запуска конструктора окно кода на экране отсутствует. Для его отображения выберите в разделе **Сервис** вкладки **Конструктор** пиктограмму **SQL**. После завершения процесса конструирования проверьте синтаксис SQL. Для этого предназначена одноименная пиктограмма, также находящаяся в разделе **Сервис**.

Для запуска конструктора встроенной функции выполните следующие действия:

1. Откройте главное окно MS Access 2010.
2. Выберите вкладку **Создание**. Перейдите в раздел **Макросы и код**.
3. Сделайте щелчок по пиктограмме **Мастер запросов**. Откроется диалоговое окно **Новый запрос**.
4. Выберите в нем пункт **Конструктор встроенной функции** и нажмите кнопку **OK**.
5. Откроются: окно конструктора **Функция1** и окно добавления таблицы.
6. После завершения процесса конструирования закройте окно **Функция1**. Появится запрос на ввод имени функции. Введите имя функции и нажмите кнопку **OK**.

14.9.4. Создание хранимой процедуры средствами MS Access 2010

В *разд. 14.6* мы создали хранимую процедуру `dbo.Document` при помощи кода, написанного на Transact-SQL. Выполним эту же работу при помощи конструктора MS Access:

1. Откройте главное окно Microsoft Access 2010.
2. Выберите вкладку **Создание**. Перейдите в раздел **Макросы и код**.
3. Сделайте щелчок по пиктограмме **Мастер запросов**. Откроется диалоговое окно **Новый запрос**.
4. Выберите в нем пункт **Конструктор хранимой процедуры** и нажмите кнопку **OK**.
5. Откроются окно конструктора **ХранимаяПроцедура1** и окно добавления таблицы.

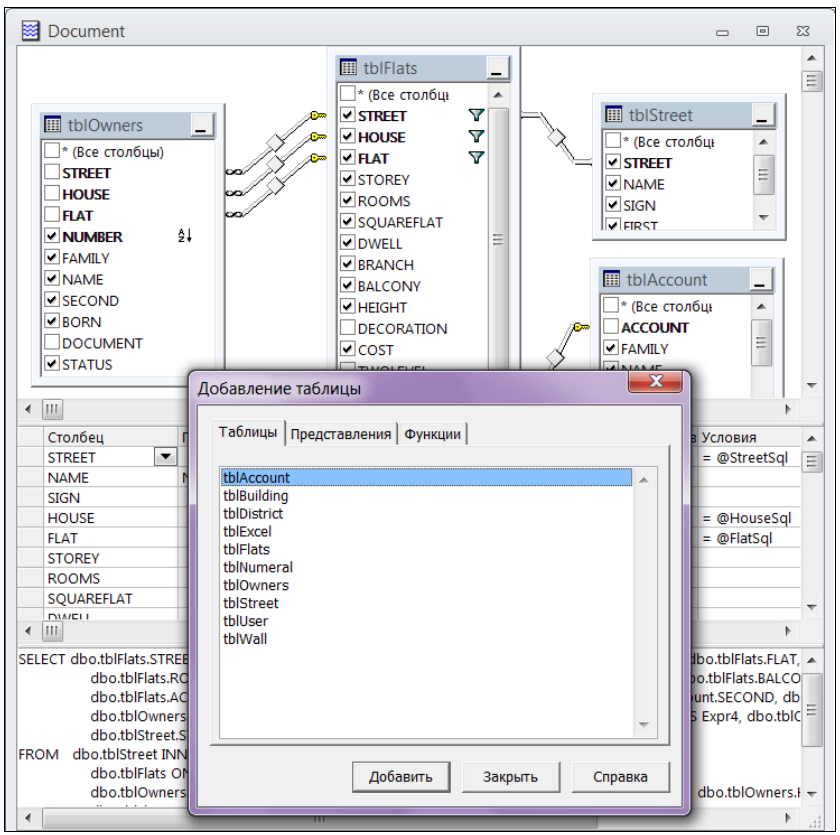


Рис. 14.35. Конструирование хранимой процедуры `Document`

При создании хранимой процедуры `Document` нам понадобится информация из таблиц `tblStreet`, `tblFlats`, `tblAccount` и `tblOwners`. После выбора требуемой таблицы и щелчка по кнопке **Добавить** окна добавления очередная таблица появится в верхней части окна конструктора (рис. 14.35). После завершения этого этапа работы получим все таблицы, участвующие в запросе, вместе со связями между ними.

Для полей `STREET`, `HOUSE` и `FLAT` в столбец **Условия** второй части окна конструктора занесите параметры будущей процедуры:

```
=@StreetSql
=@HouseSql
=@FlatSql
```

Сделайте щелчок по пиктограмме  **SQL**, расположенной в разделе **Сервис** вкладки **Конструктор** главной ленты MS Access 2010. Появится третья часть окна конструктора с текстом хранимой процедуры. Некоторые поля в выбранных таблицах имеют одинаковые названия, поэтому конструктор использует понятие *"псевдоним"* (`Expr1`, `Expr2`, `Expr3` и т. д.). Если у поля есть псевдоним, то именно его следует использовать в форме или отчете проекта Microsoft Access.

После завершения процесса конструирования проверьте синтаксис созданной конструкции. Для этого предназначена пиктограмма **Проверить синтаксис SQL**, находящаяся в разделе **Сервис**. В случае отсутствия ошибок на экране дисплея появится сообщение (рис. 14.36).

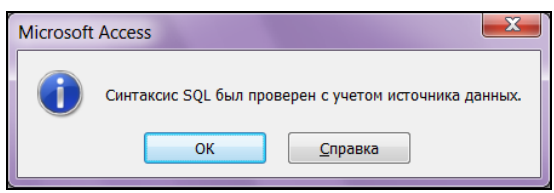


Рис. 14.36. Сообщение об успешной проверке синтаксиса хранимой процедуры

14.9.5. Создание резервной копии базы данных

Основной способ защиты от потери данных в результате отказа жесткого диска, на котором расположена база данных, — это наличие резервной копии этих данных. В процессе разработки приложения, когда изменяются объекты базы (триггеры, хранимые процедуры и т. д.), очень часто приходится возвращаться на шаг назад, отказавшись от внесенных изменений. В этом случае также не обойтись без резервной копии. В MS Access выполнить резервное копирование можно непосредственно из adp-файла.

1. Щелкните по кнопке **Файл**.
2. В появившемся меню выберите пункт **Сведения**. Появится окно **Сведения о Real EstateCS**. Щелкните по значку **Сервер**. Откроется меню.
3. Выберите в нем пункт **Создать резервную копию SQL-базы данных**. Откроется окно **Резервирование**.
4. Укажите файл, в который будет выгружена база данных SQL Server 2008, и его месторасположение. По умолчанию это папка **Документы**.
5. Резервная копия будет помещена в файл, имеющий расширение dat.

При необходимости всегда можно восстановить базу данных из имеющейся копии, выбрав в том же меню пункт **Восстановить SQL-базу данных**.

14.10. Последний штрих

При переносе базы данных на другой сервер или изменении местоположения клиента в сети проект MS Access не сможет установить связь с базой данных. Потратив на операцию соединения несколько секунд, система в большинстве случаев даже не сообщит об ошибке, а в заголовке окна проекта MS Access появится строчка "Real EstateCS нет соединения". В этом случае следует воспользоваться кнопкой **Файл**, пунктом меню **Сведения**, значком **Сервер** и строкой **Подключение** для вызова окна **Свойства канала передачи данных** (см. рис. 14.10).

Добавим интеллектуальности нашему проекту. Для этого в событие **Загрузка** стартовой формы `Start` поместим код, представленный в листинге 14.30.

Листинг 14.30. Код события *Загрузка* формы `Start`

```
Private Sub Form_Load()
If Not CurrentProject.IsConnected Then
   DoCmd.RunCommand acCmdConnection
End If
End Sub
```

Если соединение текущего проекта с базой данных не установлено, то VBA сразу запустит окно MS Access 2010 **Свойства канала передачи данных**.

Для запуска приложения в постоянную эксплуатацию преобразуйте файл проекта Real EstateCS.adp в файл без исходных текстов VBA — Real EstateCS.ade. Для этого воспользуйтесь кнопкой **Файл**, пунктом меню **Доступ** и значком **Создать ADE**. Растиражируйте клиентскую часть по рабочим станциям. Система готова к работе!

Если все пользователи информационной системы работают под одной учетной записью MS SQL-сервера, то разработчик может значительно облегчить себе жизнь при внесении изменений в действующий проект MS Access. Для этого следует разместить его не на каждой рабочей станции, а в сети и в случае модернизации обновлять всего одну копию.

Примечание

Для совместного доступа всех пользователей к одному файлу проекта необходимо указать ключ `/runtime`. Пропишите его в свойствах ярлыка на каждой рабочей станции. Например: `Y:\Real EstateCS.ade /runtime`. Проект MS Access будет запускаться с опцией "Только для чтения". Вот почему у всех пользователей должна быть одна учетная запись.

ПРИЛОЖЕНИЕ

Описание компакт-диска

Компакт-диск содержит все папки и файлы, которые были созданы в процессе работы с учебным примером Real Estate 2010 (рис. П1.1). Всего 5 папок и 9 файлов:

- BUILDING — папка с фотографиями зданий учебного примера;
- DESKTOP— папка с темами оформления окна приложения;
- HELP — папка с исходными файлами, которые использовались для создания файла-справки к приложению;
- ICO — папка со значками для оформления пользовательской ленты;
- PHOTO — папка с фотографиями персонажей программного комплекса;
- Readme.htm — файл, который необходимо прочитать в первую очередь;
- Real Estate Часть I.accdb — файл базы данных MS Access 2010, созданный в *части I*;
- Real Estate Часть II.accdb — файл базы данных MS Access 2010, созданный в *части II*;
- Real EstateCS.adp — файл проекта MS Access 2010, предназначенный для работы в качестве клиента базы MS SQL Server 2008;
- Real EstateCS.dat — резервная копия базы данных Real EstateSQL;
- RealEstate.chm — откомпилированный файл справки к приложению Real Estate;
- RibbonVBA.txt — файл с текстом первой пользовательской ленты;
- RibbonXML.txt — файл с текстом второй пользовательской ленты;
- варианты заданий.chm — откомпилированный файл с вариантами заданий на разработку приложений баз данных.

Для работы с вариантами приложения Real Estate скопируйте содержимое компакт-диска в папку, расположенную на вашем компьютере. Например, Real

Estate 2010. Не запускайте на выполнение файлы с CD. Полноценной работы в этом случае не получится!

Перед запуском на выполнение файлов:

- Real Estate Часть I.accdb;
- Real Estate Часть II.accdb;
- Real EstateCS.adp

измените настройки MS Access 2010, установленного на вашем компьютере. Измененных настроек будет две.

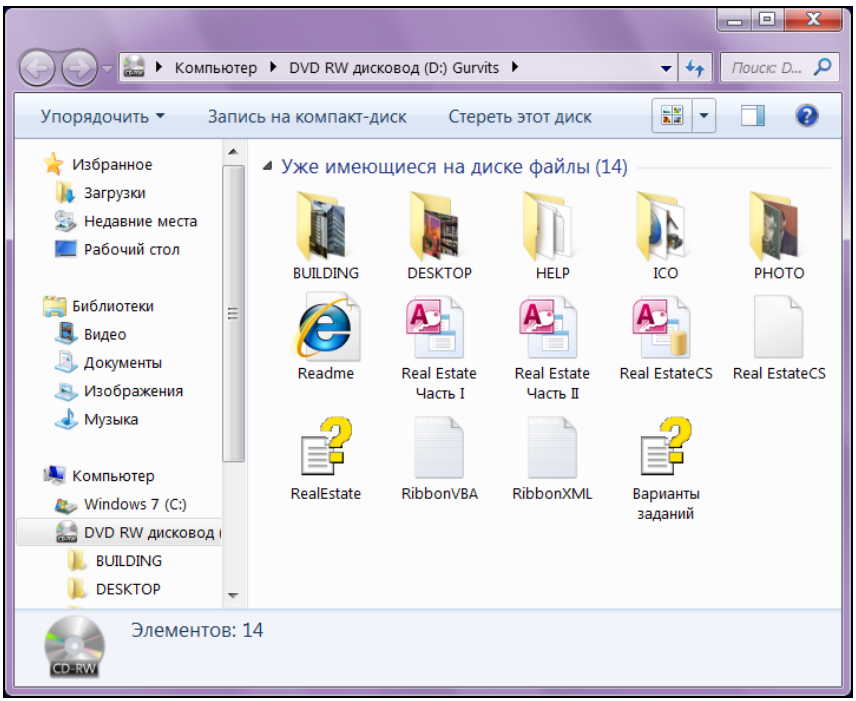


Рис. П1.1. Состав компакт-диска

Отключение функции блокирования макросов и процедур VBA:

1. Выберите в главном окне Microsoft Access 2010 кнопку **Файл**. Откроется вкладка.
2. Сделайте щелчок по кнопке **Параметры**. Она расположена в левом нижнем углу вкладки. Появится окно **Параметры Access.**
3. В левой части появившегося окна выберите пункт **Центр управления безопасностью**. Появится диалоговое окно.
4. Нажмите кнопку **Параметры центра управления безопасностью**.

5. В появившемся окне выберите пункт **Параметры макросов.**
6. В разделе параметров установите переключатель **Включить все макросы...**.

Включение отображения сообщений об ошибках пользовательского интерфейса надстроек:

1. Щелкните по кнопке **Файл**. Выберите пункт **Параметры**. Откроется диалоговое окно **Параметры Access.**
2. Выберите пункт **Параметры клиента**. Выполните прокрутку вниз до появления раздела **Общие.**
3. Установите флажок **Показывать ошибки интерфейса пользователя надстроек.**
4. Щелкните по кнопке **OK** для завершения процесса.

Для запуска на выполнение приложения "клиент-сервер" (файл Real EstateCS.adp) выполните дополнительно следующие действия:

1. Установите на своем компьютере MS SQL Server 2008.
2. Запустите SQL Server Management Studio.
3. Появится окно **Connect to Server** (Подключение к серверу). Выберите требуемый сервер, режим аутентификации, имя пользователя и пароль. Нажмите кнопку **Connect** (Подключиться).
4. В окне обозревателя объектов **Object Explorer** выделите объект **Databases** (Базы данных) и сделайте по нему щелчок правой кнопкой мыши. Появится контекстное меню.
5. Выберите в нем пункт **Restore Database** (Восстановить базу данных). Откроется одноименное окно (рис. П1.2).
6. Введите в поле **To database** название базы данных: **Real EstateSQL.**
7. Отметьте переключатель **From device.**
8. Щелкните по пиктограмме с тремя точками, которая расположена справа от этого переключателя. Откроется окно **Specify Backup.**
9. Сделайте щелчок по кнопке **Add**. Откроется окно Проводника.
10. Найдите в нем файл Real EstateCS.dat, расположенный на компакт-диске, и щелкните по кнопке **OK**. Имя этого файла появится в окне **Specify Backup.**
11. Щелкните по кнопке **OK**. Окно **Specify Backup** закроется, а информация о файле, из которого будет восстановлена база данных, появится в нижней части окна **Restore Database.**
12. Поставьте флажок в первой колонке окна восстановления **Select the backup sets to restore**, которая носит имя **Restore.**
13. Сделайте щелчок по кнопке **OK**. Появится сообщение об успешном восстановлении базы данных.

14. Окно **Restore Database** закроется, а база данных Real EstateSQL появится в окне обозревателя объектов **Object Explorer**.
15. Закройте окно **SQL Server Management Studio**. Для дальнейшей работы файл Real EstateCS.dat не нужен.
16. Запустите на выполнение файл Real EstateCS.adp.
17. Появится окно **Свойства связи с данными**.
18. Введите имя сервера, режим аутентификации и имя базы данных. Нажмите кнопку **OK**.

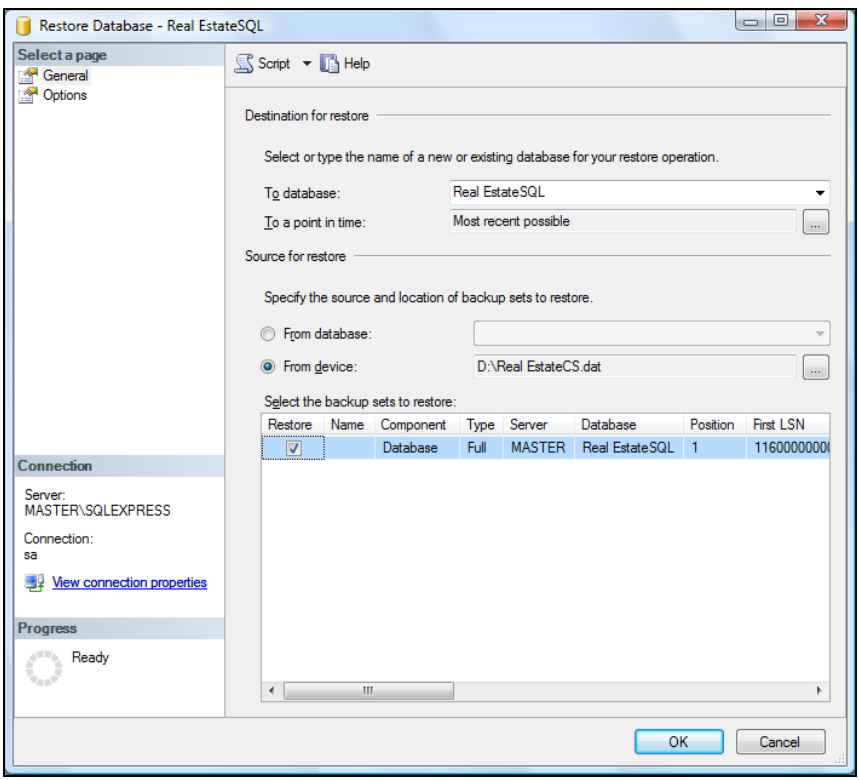


Рис. П1.2. Восстановление базы данных MS SQL Server 2008

Предметный указатель

А

Б

В

Г

Д

З

И

К

Л

М

Н

О

П

Р

С

Т

У

Ф

Х

Ц

Ш

Э

Я

Г. Гурвиц

MICROSOFT® ACCESS 2010

РАЗРАБОТКА ПРИЛОЖЕНИЙ НА РЕАЛЬНОМ ПРИМЕРЕ

ПРОФЕССИОНАЛЬНОЕ ПРОГРАММИРОВАНИЕ

На компакт-диске содержится реальное приложение в двух вариантах: локальном и в архитектуре «клиент-сервер», а также 50 вариантов заданий для курсового проекта на разработку прикладного программного обеспечения.

bhv®
Санкт-Петербург

БХВ-Петербург
Internet: www.bhv.ru
Тел.: (812) 251-42-44

MICROSOFT® ACCESS 2010

РАЗРАБОТКА ПРИЛОЖЕНИЙ НА РЕАЛЬНОМ ПРИМЕРЕ

Обучение на реальном примере — самая эффективная методика

Гурвиц Геннадий Александрович, кандидат технических наук, доцент кафедры «Информационные технологии и системы» Дальневосточного государственного университета путей сообщения. Автор множества статей по базам данных и программному обеспечению. Написал более 20 программных комплексов, обеспечивающих деятельность департаментов и отделов администрации Хабаровска. Является автором нескольких книг по разработке приложений, в том числе «Microsoft Access 2007. Разработка приложений на реальном примере».

Студенты и разработчики, вам предлагается подробное и доступное руководство по разработке приложений баз данных в файл-серверной и клиент-серверной архитектурах. На примере небольшой, но реальной базы данных рассмотрены все этапы жизненного цикла приложения: от постановки задачи до запуска его в постоянную эксплуатацию. Применяется предложенный автором ранее метод оформления интерфейса корпоративного приложения – метод пересекающихся каскадов. Рассмотрена стратегия защиты программного комплекса и базы данных от несанкционированного доступа.

КАТЕГОРИЯ:
БАЗЫ ДАННЫХ

ISBN 978-5-9775-0579-6

+ *На компакт-диске содержится реальное приложение в двух вариантах: локальном и в архитектуре «клиент-сервер», а также 50 вариантов заданий для курсового проекта на разработку прикладного программного обеспечения.*

БХВ-ПЕТЕРБУРГ, 190005,
Санкт-Петербург, Измайловский пр., 29
E-mail: mail@bhv.ru
Internet: www.bhv.ru
Тел.: (812) 251-42-44
Факс: (812) 320-01-79