Самоучитель

Алексей Жилинский

Microsoft® SQL Server 2005

Проектирование баз данных

Описание языка SQL

Программирование баз данных

Интеграция с .NET Framework

Описание служб MS SQL Server 2005

bhv®

Алексей Жилинский

Самоучитель Microsoft® SQL Server 2005

Санкт-Петербург
«БХВ-Петербург»
2007

УДК 681.3.06
ББК 32.973.26-018.1
Ж72

Жилинский А. А.

Ж72 Самоучитель Microsoft® SQL Server 2005. — СПб.: БХВ-Петербург, 2007. — 224 с.: ил.

ISBN 978-5-9775-0016-6

Рассмотрены основы работы с СУБД Microsoft SQL Server 2005, начиная с вопросов установки, создания и программирования баз данных и заканчивая описанием специальных возможностей SQL Server, включая интеграцию с .NET Framework, работу с XML и использование различных служб. Большое внимание уделено программированию на языке T-SQL. Описано создание и использование курсоров, хранимых процедур и триггеров; приведены табличные и скалярные функции. Уделено внимание вопросам безопасности и администрирования, в том числе с использованием службы SQL Server Agent. Материал книги сопровождается большим количеством примеров.

Для начинающих программистов

УДК 681.3.06
ББК 32.973.26-018.1

Группа подготовки издания:

Главный редактор	*Екатерина Кондукова*
Зам. главного редактора	*Игорь Шишигин*
Зав. редакцией	*Григорий Добин*
Редактор	*Елена Кашлакова*
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 28.02.07.
Формат 70×100 1/16. Печать офсетная. Усл. печ. л. 18,06.
Тираж 3000 экз. Заказ №
"БХВ-Петербург", 194354, Санкт-Петербург, ул. Есенина, 5Б.

Санитарно-эпидемиологическое заключение на продукцию
№ 77.99.02.953.Д.006421.11.04 от 11.11.2004 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 978-5-9775-0016-6

Оглавление

Введение

Система управления SQL Server 2005 представляет собой последнюю разработку фирмы Microsoft в области баз данных и анализа данных для быстрого создания масштабируемых решений электронной коммерции, бизнес-приложений и хранилищ данных. С помощью этой книги вы научитесь создавать эффективные базы данных на SQL Server.

В SQL Server 2005, по сравнению с предыдущей версией, было внесено множество изменений и улучшений, появилось много новых возможностей, например, новые типы данных, новая утилита управления, интеграция с .NET Framework, сервисы отчетов. Эти возможности значительно упрощают и увеличивают эффективность разработчиков баз данных.

Эта книга состоит из нескольких частей.

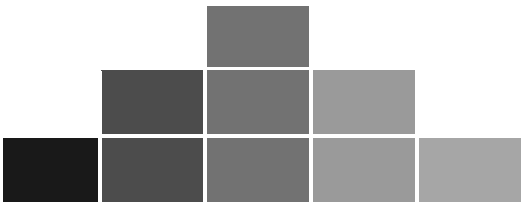
Из первой части вы узнаете, как проектировать и создавать базы данных, как управлять их поведением. Здесь же описывается, как извлекать и изменять данные в базе данных. Знания, описанные в этой части, пригодятся вам при работе с любой базой данных, а не только с SQL Server.

Во второй части книги описывается программирование для баз данных. Здесь вы узнаете, как включить в вашу базу данных более сложную логику, чем простая выборка данных, как можно просто искать текст в базе данных и какие возможности вам дает интеграция с .NET Framework и XML.

Третья часть посвящена администрированию сервера. В ней рассмотрена система безопасности SQL Server, а также конфигурирование и работа служб SQL Server.

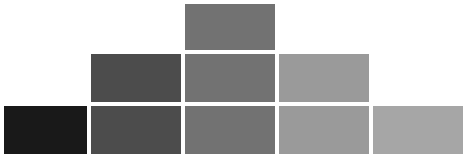
Книга снабжена множеством примеров для лучшего понимания, как работают изучаемые принципы. Также в книге описываются различия между возможностями SQL Server 2005 и его предыдущей версией, SQL Server 2000. Ведь еще некоторое время компании будут использовать SQL Server 2000, и вы должны знать различия между ними.

Автор надеется, что эта книга поможет вам стать профессиональным разработчиком баз данных для SQL Server.



ЧАСТЬ I

Установка SQL Server и проектирование баз данных



ГЛАВА 1

Установка SQL Server

Система управления SQL Server 2005 представляет собой последнюю разработку фирмы Microsoft в области баз данных и анализа данных для быстрого создания масштабируемых решений электронной коммерции, бизнес-приложений и хранилищ данных.

Для работы с материалом и примерами из книги вам потребуется одна из следующих версий SQL Server 2005:

- SQL Server 2005 Developer edition — это специальная версия, предназначенная для разработки баз данных;
- SQL Server 2005 Express edition — бесплатная версия SQL Server 2005, но с ограниченной функциональностью. Правда, вам будет достаточно этой версии для освоения материала книги. Вы можете скачать эту версию с сайта **http://download.microsoft.com**.

Установка SQL Server 2005

Здесь описывается сценарий установки SQL Server 2005 Developer edition.

1. Для запуска установки SQL Server 2005 вставьте DVD-диск в дисковод и дождитесь запуска мастера установки. Или запустите setup.exe.
2. После этого вам будет представлено лицензионное соглашение. Если вы согласитесь с ним, то мастер проверит наличие всех необходимых для установки SQL Server компонентов (например .NET Framework). Если необходимо, мастер их установит.

   В конце этого шага мастер установки будет выглядеть так, как на рис. 1.1.
3. Нажмите кнопку **Next**, и мастер начнет проверять систему на наличие в ней всех необходимых компонентов для работы SQL Server. Каждый проверенный пункт, если все в порядке, помечается зеленой галочкой. Пункты, которые могут вызвать проблемы, помечаются желтым знаком преду-

преждения. Установка при этом может продолжаться. Но если хоть один пункт помечен красным крестом, он должен быть исправлен до продолжения установки (рис. 1.2).

Рис. 1.1. Мастер установки SQL Server 2005 перед началом установки SQL Server

4. Если все системные требования выполнены, кнопка **Next** будет доступна для нажатия. Нажмите ее и введите регистрационную информацию на следующей странице мастера.
5. После этого вам необходимо выбрать компоненты SQL Server 2005 (рис. 1.3), которые вы хотите установить. Установите все флажки. При этом, если у вас на компьютере не установлен IIS, то опция **Reporting Services** будет недоступна. Нажмите кнопку **Next**.
6. Установите флажок **Default instance** на следующей странице мастера и нажмите **Next**.
7. После этого вам необходимо определить, под какой учетной записью будет работать SQL Server и службы, которые автоматически стартуют при загрузке системы. Установите все флажки так, как на рис. 1.4.
8. На следующей странице мастера вам необходимо задать, какой тип аутентификации будет использовать SQL Server 2005. Выберите **Windows authentication mode** и нажмите кнопку **Next**.

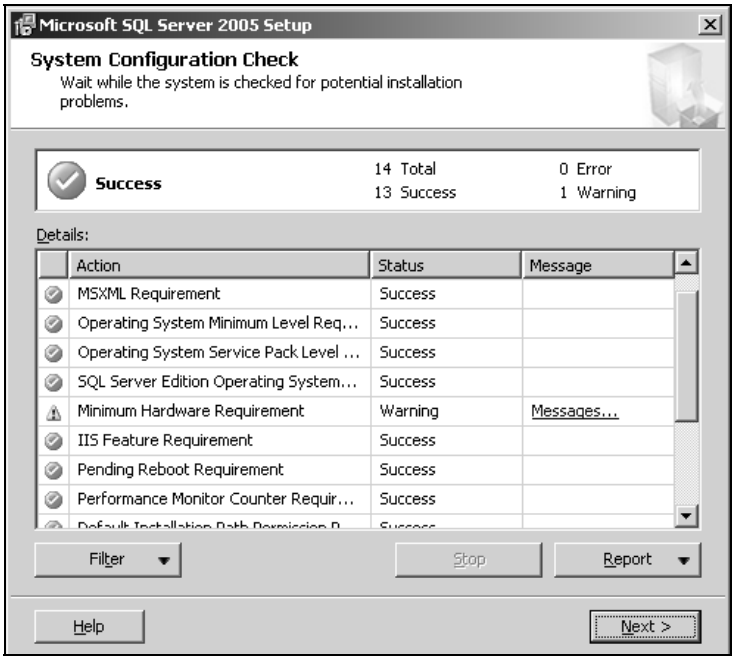


Рис. 1.2. Результат проверки системы для установки SQL Server 2005

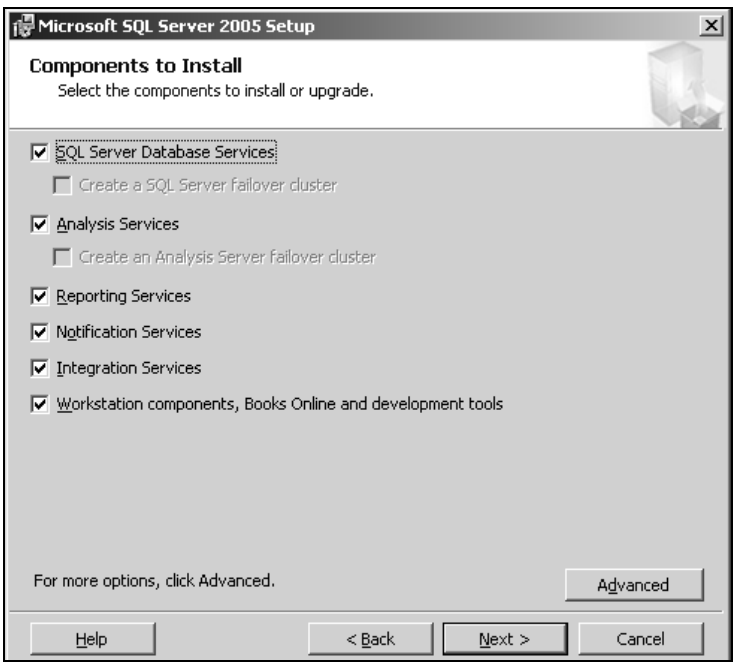


Рис. 1.3. Выбор устанавливаемых компонентов SQL Server 2005

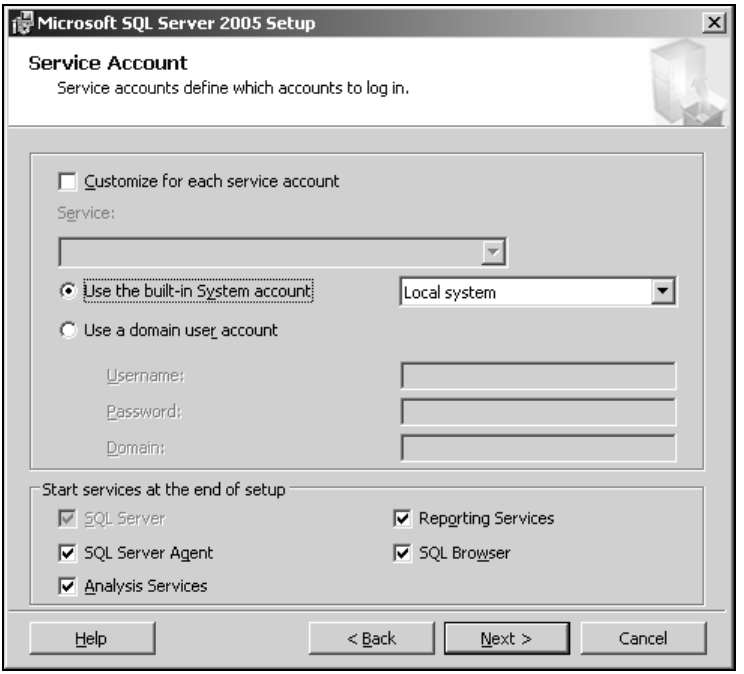


Рис. 1.4. Установка учетной записи для SQL Server 2005

9. Перед вами появится страница параметров сопоставления. Если вам не нужна совместимость с предыдущими версиями, установите параметры так, как на рис. 1.5. В противном случае выберите флажок **SQL Collation**, а после этого выберите нужный вам режим сопоставления.
10. После этого появится страница настройки параметров службы отчетов. Оставьте установку конфигурации по умолчанию и нажмите кнопку **Next**.
11. Появится страница настройки отправления сообщений об ошибках SQL Server 2005 в Microsoft. Эти сообщения позволяют сделать следующие версии SQL Server более стабильными и удобными в работе. Просто нажмите кнопку **Next**.
12. Теперь появится страница со списком всех компонентов, которые будут установлены. Нажмите кнопку **Install**, откроется страница мастера, на которой будет отображаться процесс установки различных компонентов SQL Server (рис. 1.6). По завершении установки нажмите кнопку **Finish**.

После установки SQL Server в меню **Пуск | Все программы | Microsoft SQL Server 2005** у вас окажется несколько новых значков, в частности, SQL Server Management Studio — утилита для создания и ведения баз данных и запросов.

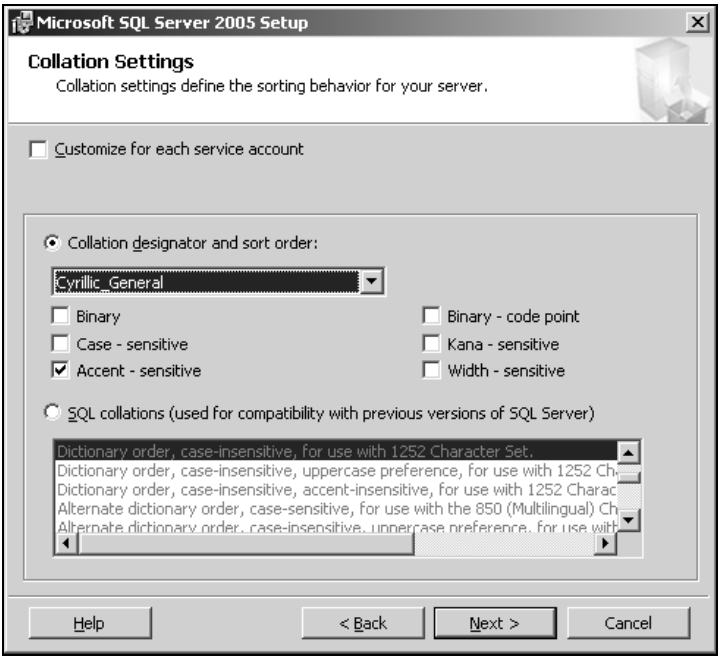


Рис. 1.5. Установка параметров сопоставления для SQL Server 2005

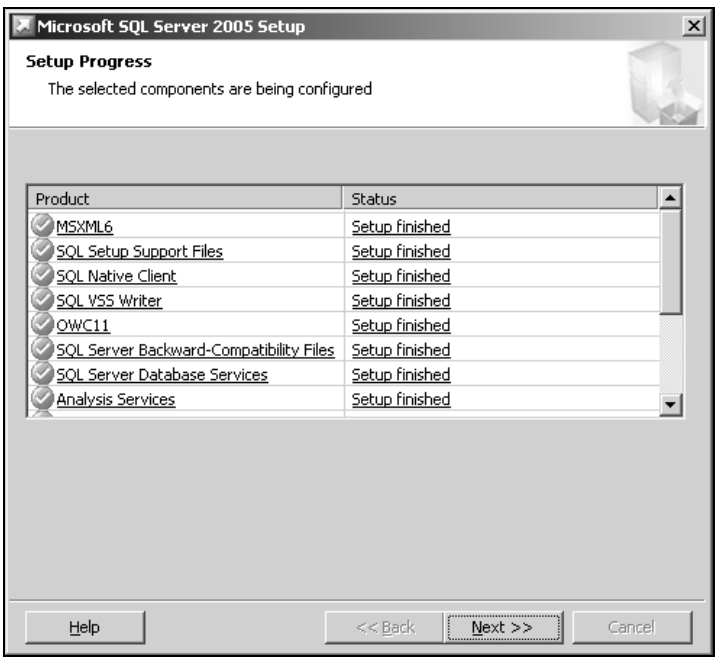
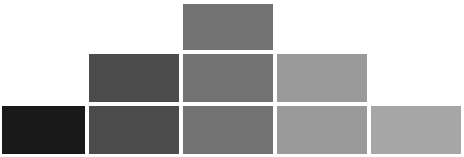


Рис. 1.6. Результат установки SQL Server 2005

ГЛАВА 2



Введение в проектирование баз данных

Основные сведения

Термин *реляционный* означает основанный на отношениях. Реляционная база данных состоит из сущностей (таблиц), находящихся в некотором отношении друг с другом. Название произошло от английского слова *relation* — отношение.

Проектирование базы данных состоит из двух основных фаз: логического и физического моделирования.

Во время логического моделирования вы собираете требования и разрабатываете модель базы данных, не зависящую от конкретной СУБД (системы управления реляционными базами данных). Это похоже на то, как если бы вы создавали чертежи для вашего дома. Вы могли бы продумать начертить все: где будет кухня, спальни, гостиная. Но это все на бумаге и в макетах.

Во время физического моделирования вы создаете модель, оптимизированную для конкретного приложения и СУБД. Именно эта модель реализуется на практике. Если вернуться к дому из предыдущего абзаца, на этом этапе вам придется строить где-нибудь дом — таскать бревна, кирпичи...

Процесс проектирования базы данных состоит из следующих этапов:

- ❒ сбор информации;
- ❒ определение сущностей;
- ❒ определение атрибутов для каждой сущности;
- ❒ определение связей между сущностями;
- ❒ нормализация;
- ❒ преобразование к физической модели;
- ❒ создание базы данных.

Первые 5 этапов образуют фазу логического проектирования, а остальные два представляют собой фазу физического моделирования.

Логическая фаза

Логическая фаза состоит из нескольких этапов. Далее они все рассмотрены.

Сбор требований

На этом этапе вам необходимо точно определить, как будет использоваться база данных и какие данные будут в ней храниться. Соберите как можно больше данных о том, что система должна делать и чего не должна.

Определение сущностей

На этом этапе вам необходимо определить сущности, как из которых будет состоять база данных.

Сущность — это объект в базе данных, в котором хранятся данные. Сущность может представлять собой нечто вещественное (дом, человек, предмет, место) или абстрактное (банковская операция, отдел компании, маршрут автобуса). В физической модели сущность называется таблицей.

Сущности состоят из атрибутов (столбцов таблицы) и записей (строк в таблице).

Обычно базы данных состоят из нескольких основных сущностей, связанных с большим количеством подчиненных сущностей. Основные сущности называются независимыми: они не зависят ни от какой-либо другой сущности. Подчиненные сущности называются зависимыми: для того чтобы она существовала, должна существовать связанная с ней основная таблица.

На диаграммах сущности обычно представляются в виде прямоугольников. Имя сущности указывается за границей прямоугольника (рис. 2.1).

Любая таблица имеет следующие характеристики:

- в ней нет одинаковых строк;
- все столбцы (атрибуты) в таблице должны иметь одинаковое имя;
- элементы в пределах одной колонки имеют одинаковый тип (строка, число, дата);
- порядок следования строк в таблице может быть произвольным.

На этом этапе вам необходимо выявить все категории информации (сущности), которые будут храниться в базе данных.

Пример базы данных с некоторыми сущностями показан на рис. 2.1.

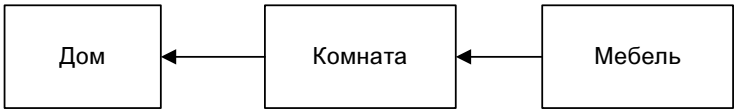


Рис. 2.1. База данных "Дома". Сущности

Определение атрибутов

Атрибут представляет свойство, описывающее сущность. Атрибуты часто бывают числом, датой или текстом. Все данные, хранящиеся в атрибуте, должны иметь одинаковый тип и обладать одинаковыми свойствами.

В физической модели атрибуты называют *колонками*.

После определения сущностей необходимо определить все атрибуты этих сущностей.

На диаграммах атрибуты обычно перечисляются внутри прямоугольника сущности. На рис. 2.2 вы найдете пример базы данных "Дома", только теперь для сущностей из этой базы определены некоторые атрибуты.

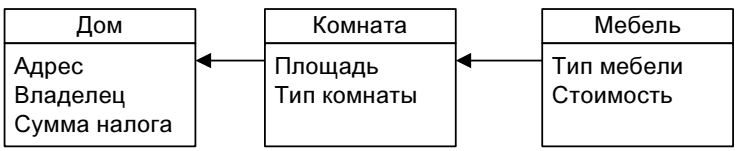


Рис. 2.2. База данных "Дома". Сущности и атрибуты

Для каждого атрибута определяется тип данных, их размер, допустимые значения и любые другие правила. К их числу относятся правила обязательности заполнения, изменяемости и уникальности.

Правило обязательности заполнения определяет, является ли атрибут обязательной частью сущности. Если атрибут является необязательной частью сущности, то он может принимать NULL-значение, иначе — нет.

Также вы должны определить, является ли атрибут изменяемым. Значения некоторых атрибутов не могут измениться после создания записи.

И, наконец, вам нужно определить, является ли атрибут уникальным. Если атрибут является уникальным, то значения атрибута не могут повторяться.

Ключи

Ключом (key) называется набор атрибутов, однозначно определяющий запись. Ключи делятся на два класса: простые и составные.

Простой ключ состоит только из одного атрибута. Например, в базе "Паспорта граждан Эфиопии" номер паспорта будет простым ключом: ведь не бывает двух паспортов с одинаковым номером.

Составной ключ состоит из нескольких атрибутов. В той же базе "Паспорта граждан страны" может быть составной ключ со следующими атрибутами: фамилия, имя, отчество, дата рождения. Это — как пример, т. к. этот составной ключ, теоретически, не обеспечивает гарантированной уникальности записи.

Также существует несколько типов ключей, о которых рассказано далее.

Возможный ключ

Возможный ключ представляет собой любой набор атрибутов, однозначно идентифицирующих запись в таблице. Возможный ключ может быть простым или составным.

Каждая сущность должна иметь, по крайней мере, один возможный ключ, хотя таких ключей может быть и несколько. Ни один из атрибутов первичного ключа не может принимать неопределенное (NULL) значение.

Возможный ключ называется также суррогатным.

Первичные ключи

Первичным ключом называется совокупность атрибутов, однозначно идентифицирующих запись в таблице (сущности). Один из возможных ключей становится первичным ключом. На диаграммах первичные ключи часто изображаются выше основного списка атрибутов или выделяются специальными символами. Сущность на рис. 2.3 имеет как ключевые, так и обычные атрибуты.

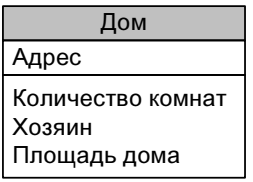


Рис. 2.3. Сущность с первичным ключом

Альтернативные ключи

Любой возможный ключ, не являющийся первичным, называется *альтернативным ключом*. Сущность может иметь несколько альтернативных ключей.

Внешние ключи

Внешним ключом называется совокупность атрибутов, ссылающихся на первичный или альтернативный ключ другой сущности. Если внешний ключ не связан с первичной сущностью, то он может содержать только неопределенные значения. Если при этом ключ является составным, то все атрибуты внешнего ключа должны быть неопределенными.

На диаграммах атрибуты, объединяемые во внешние ключи, обозначаются специальными символами. На рис. 2.4 изображены две связанные сущности (Дома и их Хозяева) и образованные ими внешние ключи (ведь один человек может владеть больше, чем одним домом).

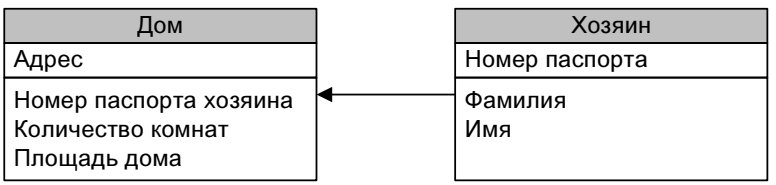


Рис. 2.4. Сущность с внешним ключом

Ключи являются логическими конструкциями, а не физическими объектами. В реляционных базах данных предусмотрены механизмы, обеспечивающие сохранение ключей.

Определение связей между сущностями

Реляционные базы данных позволяют объединять информацию, принадлежащую разным сущностям.

Отношение — это ситуация, при которой одна сущность ссылается на первичный ключ второй сущности. Как, например, сущности Дом и Хозяин на рис. 2.4.

Отношения определяются в процессе проектирования базы. Для этого следует проанализировать сущности и выявить логические связи, существующие между ними.

Тип отношения определяет количество записей сущности, связанных с записью другой сущности. Отношения делятся на три основных типа, о которых рассказано далее.

Один к одному

Каждой записи первой сущности соответствует только одна запись из второй сущности. А каждой записи второй сущности соответствует только одна

запись из первой сущности. Например, есть две сущности: Люди и Свидетельства о рождении. Например, у одного человека может быть только одно свидетельство о рождении.

Один ко многим

Каждой записи первой сущности могут соответствовать несколько записей из второй сущности. Однако каждой записи второй сущности соответствует только одна запись из первой сущности. Например, есть две сущности: Заказ и Позиция заказа. И в одном заказе может быть заказано много товаров.

Многие ко многим

Каждой записи первой сущности могут соответствовать несколько записей из второй сущности. Однако и каждой записи второй сущности может соответствовать несколько записей из первой сущности. Например, есть две сущности: Автор и Книга. Один автор может написать много книг. Но у книги может быть несколько авторов.

По критерию обязательности отношения делятся на обязательные и необязательные.

- Обязательное отношение означает, что для каждой записи из первой сущности непременно должны присутствовать связанные записи во второй сущности.
- Необязательное отношение означает, что для записи из первой сущности может и не существовать записи во второй сущности.

Нормализация

Нормализацией называется процесс удаления избыточных данных из базы данных. Каждый элемент данных должен храниться в базе в одном и только в одном экземпляре. Существует пять распространенных форм нормализации. Как правило, база данных приводится к третьей нормальной форме.

В процессе нормализации выполняются определенные действия по удалению избыточных данных. Нормализация повышает быстродействие, ускоряет сортировку и построение индекса, уменьшает количество индексов на сущность, ускоряет операции вставки и обновления.

Нормализованная база данных обычно отличается большей гибкостью. При модификации запросов или сохраняемых данных в нормализованную базу обычно приходится вносить меньше изменений, а внесение изменений имеет меньше последствий.

Первая нормальная форма

Чтобы преобразовать сущность в первую нормальную форму, следует исключить повторяющиеся группы значений и добиться того, чтобы каждый атрибут содержал только одно значение, списки значений не допускаются. Другими словами, каждый атрибут в сущности должен храниться только в одном экземпляре.

Например, на рис. 2.5 сущность Дом не нормализована. Она содержит несколько атрибутов для хранения данных о владельцах дома.

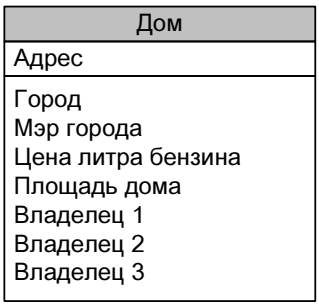


Рис. 2.5. Сущность Дом, не соответствующая первой нормальной форме

Для приведения сущности Дом в первую нормальную форму необходимо удалить повторяющиеся группы значений, т. е. удалить атрибуты Владелец 1—3, поместив их в отдельную сущность. Что получилось, смотрите на рис. 2.6.

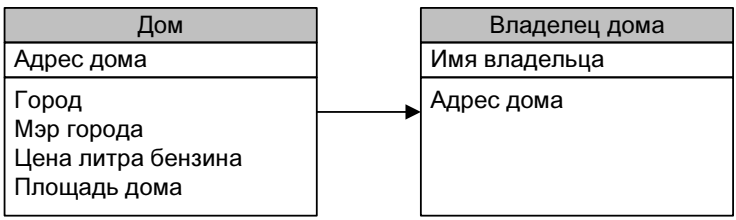


Рис. 2.6. Сущность Дом, приведенная к первой нормальной форме

Вторая нормальная форма

Таблица во второй нормальной форме содержит только те данные, которые к ней относятся. Значения неключевых атрибутов сущности зависят от первичного ключа. Если более точно, то атрибуты зависят от первичного ключа, от всего первичного ключа и только от первичного ключа.

Для соответствия второй нормальной форме сущности должны быть в первой нормальной форме.

Например, у сущности Дом на рис. 2.6 есть атрибут Цена литра бензина, который не имеет ничего общего с домами. Этот атрибут удаляется (или вы можете перенести его в другую сущность). А также мы переносим атрибут Мэр в отдельную сущность — этот атрибут зависит от города, где находится дом, а не от дома.

На рис. 2.7 изображена сущность Дом во второй нормальной форме.

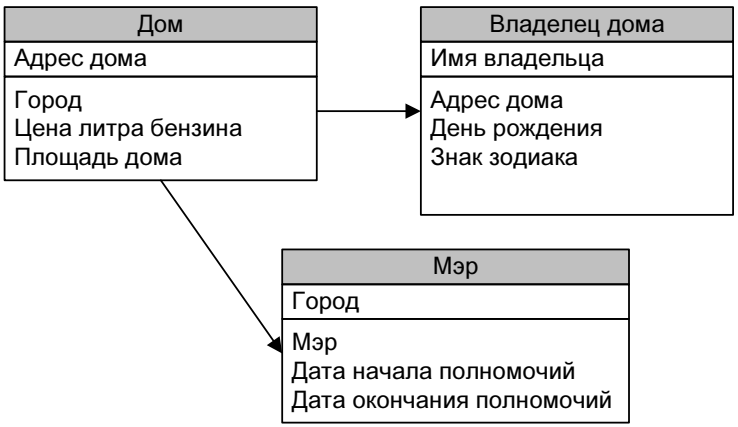


Рис. 2.7. Сущность Дом, приведенная ко второй нормальной форме

Третья нормальная форма

В третьей нормальной форме исключаются атрибуты, не зависящие от всего ключа. Любая сущность, находящаяся в третьей нормальной форме, находится также и во второй. Это самая распространенная форма базы данных.

В третьей нормальной форме каждый атрибут зависит от ключа, от всего ключа и ни от чего, кроме ключа.

Например, у сущности Владелец на рис. 2.8 есть атрибут Знак зодиака, который зависит от даты рождения владельца дома, а не от его имени (которое является ключом).

Для приведения сущности Владелец необходимо создать сущность Знаки зодиака и перенести туда атрибут Знак зодиака.

Ограничения

Ограничения (constrains) — это правила, за соблюдением которых следит система управления базы данных. Ограничения определяют множество значений, которые можно вводить в столбец или столбцы.

Например, вы не хотите, чтобы сумма заказа в вашем очень крутом магазине была бы меньше 500 рублей. Вы просто устанавливаете ограничение на колонку Сумма заказа.

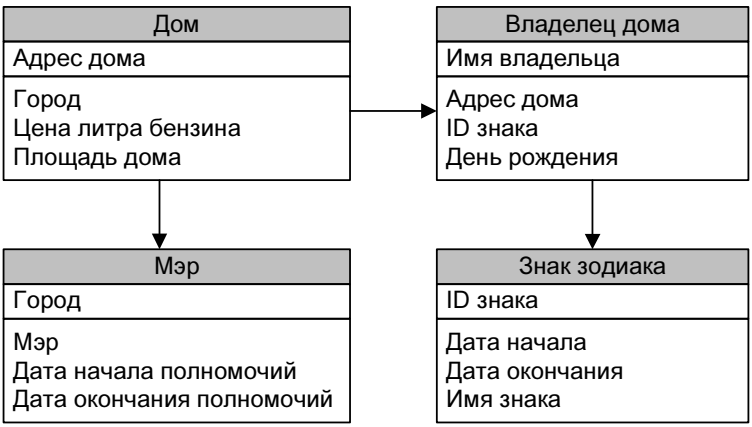


Рис. 2.8. Сущности Владелец, приведенная ко второй нормальной форме

Хранимые процедуры

Хранимые процедуры (stored procedures) — это предварительно откомпилированные процедуры, хранящиеся в базе данных. Хранимые процедуры можно использовать для определения деловых правил, с их помощью можно осуществлять более сложные вычисления, чем с помощью одних лишь ограничений.

Хранимые процедуры могут содержать логику хода выполнения программы, а также запросы к базе данных. Они могут принимать параметры и возвращать результаты в виде таблиц или одиночных значений.

Хранимые процедуры похожи на обычные процедуры или функции в любой программе.

Примечание

Хранимые процедуры хранятся в базе данных и выполняются на сервере базы данных. Как правило, они быстрее операторов SQL, поскольку хранятся в компилированном виде.

Целостность данных

Организовав данные в таблицы и определив связи между ними, можно считать, что была создана модель, правильным образом отражающая бизнес-среду. Теперь нужно обеспечить, чтобы данные, вводимые в базу, давали

правильное представление о состоянии дела. Иными словами, нужно обеспечить выполнение *деловых правил* и поддержку *целостности* (integrity) базы данных.

Например, ваша компания занимается доставкой книг. Вы вряд ли примете заказ от неизвестного клиента, ведь вы даже не сможете доставить заказ тогда. Отсюда бизнес-правило: заказы принимаются только от клиентов, информация о которых есть в базе данных.

Правильность данных в реляционных базах обеспечивается набором правил. Правила целостности данных делятся на четыре категории:

- Целостность сущностей — каждая запись сущности должна обладать уникальным идентификатором и содержать данные. Ведь надо же вам как-то различать все эти записи в базе данных.
- Целостность атрибутов — каждый атрибут принимает лишь допустимые значения. Например, сумма покупки определенно не может быть меньше нуля.
- Ссылочная целостность — набор правил, обеспечивающих логическую согласованность первичных и внешних ключей при вставке, обновлении и удалении записей. Ссылочная целостность обеспечивает, чтобы для каждого внешнего ключа существовал соответствующий первичный ключ. Возьмем предыдущий пример с сущностями Владелец и Дом. Допустим, вы Вася Иванов и владеете домом. Вы сменили фамилию на Сидоров и внесли соответствующие изменения в сущность Владелец. Определенно вы бы хотели, чтобы ваш дом продолжал числиться за вами под вашим новым именем, а не принадлежал некому Васе Иванову, которого уже не существует.
- Пользовательские правила целостности — любые правила целостности, не относящиеся ни к одной из перечисленных категорий.

Триггеры

Триггер — это аналог хранимой процедуры, который вызывается автоматически при изменении данных в таблице.

Триггеры являются мощным механизмом для поддержания целостности базы данных. Триггеры вызываются до или после изменения данных в таблице. С помощью триггеров вы можете не только отменить эти изменения, но и изменить данные в любой другой таблице.

Например, вы создаете интернет-форум и вам необходимо сделать так, чтобы в списке форумов показывалось последнее сообщение форума. Конечно, вы можете брать сообщение из сущности Сообщения форума, но это увеличит сложность вашего запроса и время его выполнения. Проще добавить триггер

к сущности Сообщения форума, который бы записывал последнее добавленное сообщение в сущность Форумы, в атрибут Последнее сообщение. Это сильно упростит запрос.

Деловые правила

Деловые правила определяют ограничения, накладываемые на данные в соответствии с требованиями бизнеса (тех, для кого вы создаете базу). Деловые правила могут состоять из набора шагов, необходимых для выполнения определенной задачи, или же они могут быть просто проверками, которые проверяют правильность введенных данных. Деловые правила могут включать правила целостности данных. В отличие от других правил, их главная цель — обеспечить правильное ведение деловых операций.

Например, в компании "Очень крутые парни" может быть так принято, что они закупают для служебных нужд только белые, синие и черные автомобили. Тогда деловое правило для атрибута Цвет автомобиля сущности Служебные автомобили будет гласить, что автомобиль может быть только белым, синим или черным.

Большинство СУБД предоставляют средства:

- для указания значений по умолчанию;
- для проверки данных перед занесением их в базу;
- для поддержания связей между таблицами;
- для обеспечения уникальности значений;
- для хранения хранимых процедур непосредственно в базе.

Все эти возможности можно применять для реализации деловых правил в базе данных.

Физическая модель

Следующим шагом, после создания логической модели, является построение физической модели. Физическая модель — это практическая реализация базы данных. Физическая модель определяет все объекты, которые вам предстоит реализовать.

При переходе от логической модели к физической сущности преобразовываются в таблицы, а атрибуты в столбцы.

Отношения между сущностями можно преобразовать в таблицы или оставить как внешние ключи.

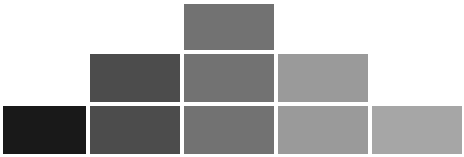
Первичные ключи преобразуются в ограничения первичных ключей. Возможные ключи преобразуются в ограничения уникальности.

Денормализация

Денормализация — это умышленное изменение структуры базы, нарушающее правила нормальных форм. Обычно это делается с целью улучшения производительности базы данных.

Теоретически, надо всегда стремиться к полностью нормализованной базе, однако на практике полная нормализация базы почти всегда означает падение производительности. Чрезмерная нормализация базы данных может привести к тому, что при каждом извлечении данных придется обращаться к нескольким таблицам. Обычно в запросе должны участвовать четыре таблицы или менее.

Стандартными приемами денормализации являются: объединение нескольких таблиц в одну, сохранение одинаковых атрибутов в нескольких таблицах, а также хранение в таблице сводных или вычисляемых данных.



ГЛАВА 3

Выборка данных

Хранение данных в базе данных принесет пользу лишь в том случае, если вы сможете получить эти данные. Для обращения к данным в Microsoft SQL Server используется язык SQL.

SQL (Structured Query Language, Язык Структурированных Запросов) — язык, используемый для запросов, обновления и управления реляционной базы данных. Диалекты языка SQL, используемые в различных реляционных базах данных, имеют некоторые различия, но в целом схожи. Диалект SQL, который используется в Microsoft SQL Server, называется Transact-SQL (T-SQL).

В SQL выборка данных осуществляется командой `SELECT`. С ее помощью вы можете получать строки из одной или нескольких таблиц базы данных.

Структура команды *SELECT*

Команда `SELECT` состоит из следующих основных частей:

- `список выборки`;
- `FROM` — определяет, откуда выбирать данные;
- `WHERE` — определяет условия выборки данных;
- `GROUP BY` — определяет условия группировки данных;
- `HAVING` — определяет условия выборки данных при использовании агрегатных функций;
- `ORDER BY` — определяет условия сортировки.

Все части команды, кроме списка выборки, являются необязательными. Эти части следуют в строго определенном порядке.

После выполнения команды `SELECT` вы получаете от SQL Server *итоговый набор* (таблицу с результатами запроса). Итоговый набор состоит из трех час-