БИБЛИОТЕКА ПРОГРАММИСТА

Виктор Гольцман

MySQL 5.0

Эта книга научит вас:

- работать с системой управления базами данных MySQL
- использовать SQL
- программировать доступ к базам данных из веб-приложений
- оптимизировать работу сервера MySQL

ПИТЕР

Виктор Гольцман
MySQL 5.0. Библиотека программиста

Текст предоставлен издательством
http://www.litres.ru/pages/biblio_book/?art=421862
MySQL 5.0. Библиотека программиста: Питер; Санкт-Петербург; 2010
ISBN 978-5-49807-135-0

Аннотация

Эта книга предназначена для всех, кто желает освоить СУБД MySQL. Для ее чтения вам не нужны никакие специальные знания – достаточно быть пользователем Windows. Вы узнаете, как установить и запустить MySQL, как создать собственную базу данных, как работать с данными при помощи команд SQL, как администрировать базу данных и оптимизировать ее работу. Разработчики веб-приложений на языках PHP, Perl и Java найдут в этой книге полезные сведения по использованию базы данных MySQL в соответствующих приложениях. Для всех операций, которые вам предстоит выполнить, приводятся подробные пошаговые инструкции, все основные действия поясняются на примере учебной базы данных.

Содержание

Виктор Гольцман MySQL 5.0. Библиотека программиста

Введение

MySQL – это система управления базами данных (СУБД) с открытым кодом. Это высокопроизводительная и масштабируемая СУБД с множеством программных интерфейсов. Она обладает огромными функциональными возможностями и подходит для решения самых разных задач.

Данная книга предназначена для всех, кто желает освоить MySQL. Чтобы начать работу, вам не потребуется никаких специальных знаний – достаточно быть пользователем Windows. Вы узнаете, как установить и запустить MySQL, как построить, администрировать собственную базу данных и оптимизировать ее работу. Вы также узнаете, как работать с данными с помощью команд языка SQL. Разработчики веб-приложений на языках PHP, Perl и Java найдут в этой книге руководство по использованию базы данных MySQL в соответствующих приложениях. В книге приводятся подробные пошаговые инструкции по выполнению всех операций. Кроме того, все основные действия поясняются на примере

учебной базы данных, содержащей информацию о клиентах, товарах и заказах торговой компании. Книга состоит из шести глав.

• **Глава 1. Знакомство.** Данная глава содержит общую информацию о СУБД MySQL, начальные сведения о реляционных базах данных и этапах проектирования базы данных. Кроме того, в главе подробно описываются установка, настройка и запуск сервера MySQL, а также подключение к нему клиентских приложений.

• **Глава 2. Управление базой данных с помощью SQL.** Глава посвящена SQL-командам, обеспечивающим работу с таблицами и их данными. Изучив эту главу, вы сможете управлять структурой таблиц, добавлять, редактировать и получать данные.

• **Глава 3. Операторы и функции языка SQL.** Данная глава дополняет предыдущую: в ней представлены сведения об операторах и функциях, позволяющих создавать условия отбора данных, обрабатывать результаты выполнения вложенных запросов, агрегировать содержащуюся в таблицах информацию и вычислять значения различных выражений.

• **Глава 4. Доступ к базе данных из веб-приложений.** Глава содержит три раздела, в которых рассматриваются интерфейсы MySQL с языками программирования PHP, Perl и Java. В каждом из разделов описываются функции подключения к базе данных, ввод и извлечение данных, обработка

ошибок взаимодействия с БД, а также примеры веб-приложений, использующих эти функции.

• **Глава 5. Администрирование и безопасность.** Глава описывает систему привилегий доступа пользователей MySQL к различным операциям с данными, а также процедуру резервного копирования и восстановления данных в случае сбоя.

• **Глава 6. Оптимизация.** В заключительной главе приводятся рекомендации по повышению производительности сервера MySQL.

Прочитав эту книгу, вы станете настоящим профессионалом и ценным сотрудником для коммерческих фирм, занятых разработкой веб-приложений различного назначения.

От главы коллектива авторов

Высказать замечания и пожелания, задать вопросы по этой книге вы можете по адресу AlexanderZhadaev@sigmaplus.mcdir.ru или посетив нашу домашнюю страничку www.sigmaplus.mcdir.ru (там вы найдете также дополнительные материалы по книге, сможете принять участие в форуме или пообщаться в чате).

Александр Жадаев

От издательства

Ваши замечания, предложения и вопросы отправляйте по адресу электронной почты gurski@minsk.piter.com (издательство «Питер», компьютерная редакция).

Мы будем рады узнать ваше мнение!

На сайте издательства http://www.piter.com вы найдете подробную информацию о наших книгах.

Глава 1
Знакомство

Эта глава содержит сведения о построении базы данных и о подготовительных этапах работы: проектировании БД, установке и запуске MySQL. Следующий раздел даст вам общее представление об этой программе.

1.1. Что такое MySQL

MySQL – это свободно распространяемая СУБД, разработанная компанией MySQL AB (www.mysql.com). MySQL имеет клиент-серверную архитектуру: к серверу MySQL могут обращаться различные клиентские приложения, в том числе с удаленных компьютеров. Рассмотрим важнейшие особенности MySQL, благодаря которым эта программа приобрела популярность.

• MySQL – это СУБД с открытым кодом. Любой желающий может бесплатно скачать программу на сайте разработчика (http://dev.mysql.com/downloads/) и при необходимости доработать ее. Существует множество приложений MySQL, созданных и свободно распространяемых сторонними разработчиками. Однако для применения MySQL в коммерческом приложении необходимо приобрести коммерческую лицензированную версию программы у компании MySQL AB.

• MySQL – кроссплатформенная система. Ее можно использовать практически во всех современных операционных системах, в том числе Windows, Linux, Mac OS, Solaris, HP-UX и др. В этой книге мы рассмотрим работу с MySQL только в ОС Windows.

• MySQL имеет множество программных интерфейсов (API), благодаря которым к базе данных MySQL могут под-

ключаться приложения, созданные с помощью C/C++, Eiffel, Java, Perl, PHP, Python, Tcl, ODBC, NET и Visual Studio. В главе 4 вы узнаете, как обращаться к базе данных MySQL из PHP-, Perl– и Java-приложений.

• MySQL имеет отличные технические характеристики: многопоточность, многопользовательский доступ, быстродействие, масштабируемость (компания-разработчик приводит пример MySQL-сервера, который работает с 60 тыс. таблиц, содержащими приблизительно 5 млрд строк).

• MySQL имеет развитую систему обеспечения безопасности и разграничения доступа на основе системы привилегий (гл. 5).

MySQL представляет собой реляционную СУБД, то есть систему управления реляционными базами данных. Поэтому для построения базы данных в MySQL нам потребуются базовые понятия теории реляционных баз данных. Этим понятиям посвящается следующий раздел.

1.2. Основные сведения о реляционных базах данных

Из этого раздела вы узнаете, как устроена реляционная база данных. Вначале мы рассмотрим таблицы, затем ключевые столбцы, связи между таблицами и, наконец, целостность данных в базе.

Таблицы

Реляционная база данных существует в виде таблиц, имеющих свои имена. На пересечении каждого столбца и каждой строки располагается одно значение.

Рассмотрим таблицу, содержащую сведения о клиентах компании (табл. 1.1).

Таблица 1.1. Customers (Клиенты)

id (идентификатор)	name (имя)	phone (телефон)	address (адрес)	rating (рейтинг)
533	ООО «Кускус»	313-48-48	ул. Смольная, д. 7	1000
534	Петров	7(929)112-14-15	ул. Рокотова, д. 8	1500
536	Крылов	444-78-90	Зеленый пр-т, д. 22	1000

Строки таблицы могут храниться в произвольной последовательности и не должны повторяться.

Каждый столбец таблицы имеет имя и *тип данных,* кото-

рому соответствуют все значения в столбце. Так, в нашем примере столбцы с именами id и rating – числовые, а с именами name, phone и address – символьные.

По существу, таблица реляционной базы данных представляет собой набор информации об однотипных объектах. При этом каждая строка содержит сведения об одном объекте, а каждый столбец – значения некоторого атрибута этих объектов. Например, строка с идентификационным номером 533 содержит информацию об объекте, у которого атрибут name (имя) имеет значение ООО «Кускус», атрибут phone (телефон) – значение 313-48-48 и т. д.

Далее мы рассмотрим специальные столбцы таблицы – первичный и внешний ключи.

Первичный ключ

Строки таблицы неупорядочены и не имеют номеров, поэтому различить их можно только по содержащимся значениям. В связи с этим возникает необходимость рассмотреть понятие первичного ключа (primary key).

Первичный ключ – это минимальный набор столбцов, совокупность значений которых однозначно определяет строку. Это означает, что в таблице не должно быть строк, у которых значения во всех столбцах первичного ключа совпадают, при этом ни один столбец нельзя исключить из первичного ключа, иначе это условие нарушится.

На практике первичным ключом служит специальный столбец, значения которого автоматически задает СУБД. Например, в таблице Customers (Клиенты) (см. табл 1.1) это столбец id (идентификатор). Использовать такой искусственный первичный ключ значительно проще, чем естественный (основанный на атрибутах объекта). Например, в таблице Customers столбец name (имя) не может быть первичным ключом, так как имена клиентов могут совпадать; а первичный ключ из столбцов name (имя) и phone (телефон) был бы слишком громоздким. Дополнительными преимуществами искусственного ключа являются гарантированная уникальность значений (ее обеспечивает СУБД), постоянство значений (может меняться значение атрибута, но не значение искусственного ключа), а также числовой тип данных (поиск по числовым значениям выполняется намного быстрее, чем по символьным).

Еще одна функция первичного ключа – организация *связей* между таблицами.

Связи между таблицами. Внешний ключ

Реляционная база данных – это не просто набор таблиц. Объединить разрозненные фрагменты информации в единую структуру данных позволяют *связи* между таблицами, посредством которых строка одной таблицы сопоставляется строке (строкам) другой таблицы. Благодаря связям мож-

но извлекать информацию одновременно из нескольких таблиц (например, выводить с помощью одного запроса и сведения о клиенте, и сведения о его заказах), избегать дублирования информации (не требуется в каждом заказе хранить адрес клиента), поддерживать полноту информации (не хранить сведения о заказанном товаре, если в базе данных отсутствует его описание) и многое другое.

Рассмотрим на примере, что такое связь между таблицами. Допустим, у нас есть таблицы *A* и *B,* и мы хотим их связать. Для этого в каждую строку таблицы *A* мы должны поместить некую информацию, позволяющую идентифицировать связанную с ней строку таблицы B. Эта информация называется *ссылкой,* а поля таблицы *A,* содержащие эту ссылку, – *внешними ключами.* Наверное, вы уже сами догадались, что в качестве ссылки используется первичный ключ таблицы *B,* поскольку именно его значения позволят однозначно идентифицировать нужную строку таблицы *B.* После того как мы во все строки таблицы *A* поместим ссылки на строки таблицы B, эти таблицы будут связаны. При этом таблица *A* будет называться *дочерней,* а таблица *B – родительской.*

Существует три типа связей, устанавливаемых между таблицами в базе данных.

• Связь «один ко многим».

Этот тип связи используется чаще всего. В этом случае одна или несколько строк таблицы *A* ссылаются на одну из строк таблицы *B.*

Для установки связи между таблицами в дочернюю таблицу добавляется *внешний ключ* (foreign key) – один или несколько столбцов, содержащих значения первичного ключа родительской таблицы (иными словами, во внешнем ключе хранятся *ссылки* на строки родительской таблицы).

Рассмотрим таблицу, которая содержит сведения о заказах, сделанных клиентами, и является дочерней по отношению к таблице Customers (Клиенты) (табл. 1.2).

Таблица 1.2. Orders (Заказы)

id (идентификатор)	date (дата)	product_id (товар)	qty (количество)	amount (сумма)	customer_id (клиент)
1012	12.12.2007	5	8	4500	533
1013	12.12.2007	2	14	22 000	536
1014	21.01.2008	5	12	5750	533

В таблице Orders внешним ключом является столбец customer_id (клиент), в котором содержатся номера клиентов из таблицы Customers (Клиенты). Таким образом, каждая строка таблицы Orders ссылается на одну из строк таблицы Customers. Например, строка с идентификационным номером 1012 содержит в столбце customer_id (клиент) значение 533: это означает, что заказ № 1012 сделан клиентом ООО «Кускус».

Столбец product_id таблицы Orders также является внешним ключом – он содержит номера товаров из столбца id (идентификатор) таблицы Products (Товары). Таким обра-

зом, таблица Orders является дочерней по отношению к таблицам Customers и Products.

• Связь «один к одному».

Такая связь между таблицами означает, что каждой строке одной таблицы соответствует одна строка другой таблицы, и наоборот. Например, если требуется хранить паспортные данные клиентов, можно создать таблицу Passports (Паспорта), связанную отношением «один к одному» с таблицей Customers (Клиенты).

Таблицы, соединенные связью «один к одному», можно объединить в одну. Две таблицы вместо одной используют по соображениям конфиденциальности (например, можно ограничить доступ пользователей к таблице Passports), для удобства (если в единой таблице слишком много столбцов), для экономии дискового пространства (в дополнительную таблицу выносят те столбцы, которые часто бывают пустыми, тогда дополнительная таблица содержит значительно меньше строк, чем основная, и обе они занимают меньше места, чем единая таблица).

Связь «один к одному» может быть организована так же, как связь «один ко многим», – с помощью первичного ключа родительской таблицы и внешнего ключа дочерней. Другой вариант – связь посредством первичных ключей обеих таблиц, при этом связанные строки имеют одинаковое значение первичного ключа.

• Связь «многие ко многим».

Этот тип связи в реляционной базе данных реализуется только с помощью вспомогательной таблицы. Например, если потребуется включить в заказ несколько наименований товаров, связь «многие ко многим» между таблицами Orders (Заказы) и Products (Товары) можно организовать с помощью вспомогательной таблицы Items (Позиции заказа), содержащей столбцы product_ id (номер товара из таблицы Products), qty (количество товаров данного наименования в заказе) и order_id (номер заказа из таблицы Orders). При этом столбцы product_id и qty из таблицы Orders исключаются. Таким образом, таблица Items будет дочерней по отношению к таблицам Orders и Products и каждая строка таблицы Items будет соответствовать одному наименованию товара в заказе.

Как видим, реляционная база данных представляет собой весьма запутанную структуру, в которой все части (то есть записи) ссылаются на другие самым произвольным образом. А раз структура сложная, то неизбежны ее нарушения, происходящие по различным причинам, включая сбои программы, ошибки оператора и др. Последствия такого нарушения могут быть просто катастрофическими: скажем, что будет, если таблица заказов будет неверно ссылаться на таблицу товаров? Вся деятельность фирмы будет дезорганизована – вместо заказанного товара, допустим лопат, заказчику доставят топоры, а то и вовсе ничего, если ссылка на заказанный товар указывает на несуществующую строку таблицы това-

ров.

Итак, важнейшим понятием теории реляционных баз данных является *целостность данных.*

Целостность данных

Целостностью данных, хранимых в СУБД, называется их корректность и непротиворечивость.

Базовыми требованиями целостности, которые должны выполняться в любой реляционной базе данных, являются *целостность сущностей* и *целостность связей* (ссылочная целостность).

Целостность сущностей означает, что в каждой таблице есть первичный ключ – уникальный идентификатор строки. Первичный ключ не должен содержать повторяющихся и неопределенных значений. Например, если в таблицу Customers (Клиенты) добавить еще одну строку с идентификатором 533 (притом что одна строка с таким идентификатором уже существует в таблице), то целостность сущностей будет нарушена и невозможно будет определить, кому из этих двух клиентов с одинаковыми идентификаторами принадлежат заказы №№ 1012 и 1014.

Целостность связей означает, что внешний ключ в дочерней таблице не содержит значения, отсутствующие в первичном ключе родительской таблицы. Иными словами, строка дочерней таблицы не должна ссылаться на несуществую-

щую строку родительской таблицы. В отличие от первичного, внешний ключ может содержать неопределенные значения (NULL), и в этом случае целостность не нарушится. Например, в таблицу Orders (Заказы) добавлена строка, содержащая в столбце customer_id значение 999. Здесь нарушится целостность связи между таблицами Customers и Orders: с одной стороны, заказ не является «ничьим», так как в этом случае в столбце customer_id было бы установлено значение NULL, с другой стороны, невозможно выяснить имя и адрес клиента, сделавшего этот заказ.

Как видно из приведенных примеров, если целостность данных нарушена, то с ними невозможно нормально работать. Поэтому поддержание целостности данных является одной из основных функций любой СУБД.

Для поддержания целостности сущностей СУБД проверяет корректность значения первичного ключа при добавлении и изменении строк. Механизм поддержания ссылочной целостности более сложный. Помимо проверки корректности значения внешнего ключа при добавлении и изменении строк дочерней таблицы, необходимо также предотвратить нарушение ссылочной целостности при удалении и изменении строк родительской таблицы. Для этого существует несколько способов.

• Запрет (RESTRICT): если на строку родительской таблицы ссылается хотя бы одна строка дочерней таблицы, то удаление родительской строки и изменение значения пер-

вичного ключа в такой строке запрещаются. Например, не допускается удаление информации о клиенте из таблицы Customers (Клиенты), если у этого клиента есть зарегистрированные заказы, то есть строки в таблице Orders (Заказы), которые ссылаются на строку со сведениями об этом клиенте.

• Каскадное удаление/обновление (CASCADE): при удалении строки из родительской таблицы автоматически удаляются все ссылающиеся на нее строки дочерней таблицы; при изменении значения первичного ключа в строке родительской таблицы автоматически обновляется значение внешнего ключа в ссылающихся на нее строках дочерней таблицы.

Например, при удалении записи о клиенте из таблицы Customers (Клиенты) автоматически удаляются сведения о заказах этого клиента, то есть соответствующие строки в таблице Orders (Заказы).

• Обнуление (SET NULL): при удалении строки и при изменении значения первичного ключа в строке значение внешнего ключа во всех строках, ссылающихся на данную, автоматически становится неопределенным (NULL). Например, при удалении записи о клиенте из таблицы Customers (Клиенты) заказы этого клиента автоматически становятся «ничьими», то есть в соответствующих строках таблицы Orders (Заказы) в столбце customer_id (клиент) устанавливается значение NULL.

В СУБД MySQL способ поддержания целостности связи указывается при создании или изменении структуры дочерней таблицы.

С понятием целостности данных тесно связано понятие *транзакции*. Транзакцией называется группа связанных операций, которые должны быть либо все выполнены, либо все отменены. Если при выполнении одной из операций происходит ошибка или сбой, то транзакция отменяется. При этом все уже внесенные другими операциями изменения автоматически аннулируются и восстанавливается исходное состояние базы данных. Важнейшее применение транзакций – это объединение тех операций, которые, будучи выполнены по отдельности, могут нарушить целостность данных. Например, рассмотренная выше операция каскадного удаления выполняется как единая транзакция: строка родительской таблицы должна быть удалена вместе со всеми ссылающимися на нее строками дочерней таблицы, а если по каким-либо причинам одну из этих строк удалить невозможно, то не будет удалена ни одна из строк.

Теперь, когда вы ознакомились с основными понятиями теории реляционных баз данных, можно приступить к разработке собственной базы.

1.3. Проектирование базы данных

Построение базы данных (как и любой информационной системы, любого программного продукта) начинается с проектирования. В процессе его мы определяем задачи, для решения которых предназначена база данных, и создаем представление о данных и связях между ними.

Проектирование включает в себя следующие основные этапы.

• Определение требований к базе данных.

В первую очередь, необходимо составить перечень требований, которым должна соответствовать проектируемая база данных. В этом разделе мы рассматриваем только функциональные требования. Другие требования (производительность, масштабируемость, надежность) также нужно учитывать, однако их выполнение во многом зависит от используемой СУБД.

Например, при проектировании базы данных для торговой компании может выясниться, что отделу по работе с клиентами необходимо знать номера телефонов всех клиентов, отделу доставки нужен отчет, содержащий адрес клиента и список заказанных им товаров, отделу логистики – информация о том, какие товары в каком количестве были заказаны в прошлом месяце, и т. п. Эти требования и будут поло-

жены в основу проекта базы данных.

• Создание модели данных, соответствующей всем предъявленным требованиям. Для разработки модели данных на основе сформулированных требований можно использовать одну из двух противоположных стратегий.

• Проектирование «снизу вверх», от элемента к структуре: вначале определяется, какие именно атрибуты должны храниться в базе данных, затем группы атрибутов объединяются в объекты. Этот метод годится для небольших баз данных, в которых количество атрибутов невелико.

• Проектирование «сверху вниз» начинается с выделения высокоуровневых объектов и связей между ними, затем осуществляется декомпозиция объектов и последовательная детализация модели до уровня атрибутов. Для сложных баз данных с большим количеством атрибутов такой метод более эффективен, чем метод «снизу вверх».

В результате мы получим предварительную структуру базы данных: список объектов – таблиц и список атрибутов каждого объекта – столбцов таблицы. Например, на основе требований, приведенных в п. 1, можно построить модель данных, содержащую сведения о таких объектах, как клиенты, заказы и товары.

• Для клиентов: идентификатор, имя (или название организации), номер контактного телефона, адрес, а также рейтинг, используемый для расчета скидки.

• Для товаров: идентификатор, наименование, описание, название склада, где хранится этот вид товара, и адрес склада.

• Для заказов: дату заказа, идентификатор заказанного товара, количество товаров этого наименования, общую стоимость заказа с учетом скидки, идентификатор клиента, сделавшего заказ, и адрес клиента, куда нужно доставить заказ (здесь мы предполагаем, что каждый заказ может включать только одно наименование товара).

• Нормализация.

Нормализация базы данных заключается в минимизации избыточности данных. Нормализация позволяет уменьшить объем БД и устранить потенциальную противоречивость данных (например, если в базе данных одна и та же информация дублируется в нескольких местах, то при ее обновлении есть риск появления разночтений).

Результатом нормализации является приведение таблиц базы данных к одной из *нормальных форм.* На практике чаще всего используются три нормальные формы.

• Таблица находится в первой нормальной форме, если все атрибуты атомарны, то есть на пересечении любого столбца и строки находится значение, части которого не будут использоваться по отдельности.

Ответ на вопрос, является ли атрибут атомарным, зависит от функциональных требований к базе данных. Рассмотрим,

например, столбец address (адрес) из таблицы Customers (Клиенты) (см. табл. 1.1). Если адрес клиента будет использоваться только целиком, то этот столбец является атомарным. Если же потребуется получать из базы отдельно название города, улицы и т. п., то для приведения таблицы Customers к первой нормальной форме столбец address следует разбить на столбцы city (город), street (улица), building (здание) и т. д.

• Таблица находится во второй нормальной форме, если она находится в первой нормальной форме и ни один из ее неключевых атрибутов не находится в функциональной зависимости от части первичного ключа.

Это означает, что в таблице, в которой есть составной первичный ключ, значения остальных столбцов таблицы должны зависеть от значений *всех* столбцов первичного ключа. Если же есть столбцы, которые зависят только от *некоторых* столбцов первичного ключа, то для приведения таблицы во вторую нормальную форму необходимо перенести все эти столбцы в другую таблицу.

Например, в нашей модели, построенной в п. 1, в таблице заказов первичным ключом может служить набор столбцов, содержащих дату заказа, идентификатор товара и идентификатор клиента (если мы допустим, что клиент не может сделать повторный заказ того же товара в тот же день, а может только изменить ранее сделанный заказ). Таким образом, для приведения таблицы заказов ко второй нормаль-

ной форме нужно исключить из таблицы адрес клиента, так как он зависит от идентификатора клиента, который является частью возможного первичного ключа. В противном случае адрес клиента будет повторяться в каждом заказе, что может привести к несогласованности данных. В частности, при изменении адреса клиента потребуется изменить адрес во всех заказах этого клиента. Если при выполнении такого массового обновления данных произойдет ошибка, то возможна ситуация, когда в некоторых заказах адрес будет изменен, а в некоторых останется прежним, и будет неясно, какой из адресов правильный. Нормализация таблицы позволяет избежать такой несогласованности.

> **Примечание**
>
> Атрибут A функционально зависит от группы атрибутов B, если значение атрибута A однозначно определяется набором значений группы атрибутов B, иными словами, в строках с одинаковым набором значений атрибутов группы B значение атрибута A также одинаково.

• Таблица находится в *третьей нормальной форме,* если она находится во второй нормальной форме и любой неключевой атрибут функционально зависит *только* от первичного ключа.

Например, в модели данных из п. 1 таблица, содержащая сведения о товарах, не находится в третьей нормальной форме, поскольку в ней имеется функциональная зависимость

адреса склада от его названия. Таким образом, вам придется всякий раз при упоминании склада писать и его адрес, что приведет к многократному дублированию данных. Чтобы привести таблицу к третьей нормальной форме, все данные о складе нужно вынести в отдельную таблицу, которая будет родительской по отношению к таблице товаров.

Когда все таблицы базы данных приведены в третью нормальную форму, мы можем считать, что наша база данных нормализована, а информация о каждом факте хранится только в одном месте.

Итак, мы разработали логическую структуру базы данных, и можно переходить к созданию базы данных в СУБД MySQL. Если программа MySQL еще не установлена на вашем компьютере, из следующего раздела вы узнаете, как это сделать.

1.4. Установка и настройка MySQL

В этом разделе вы узнаете, как установить сервер MySQL и выполнить его начальную настройку. Начнем с нескольких советов по загрузке программы.

Загрузка MySQL

Как упоминалось ранее, дистрибутив MySQL можно бесплатно скачать с сайта компании-разработчика. Windows-версию MySQL вы найдете на веб-странице http://dev.mysql.com/downloads/mysql/5.0.html в одном следующих разделов:

• Windows downloads – здесь находятся ссылки на дистрибутивы MySQL для 32-разрядных операционных систем: Windows Vista/Server 2003/XP/Millennium Edition/2000/98/95;

• Windows x64 downloads – здесь располагаются ссылки на дистрибутивы MySQL для 64-разрядных операционных систем: Windows Vista x64/Server 2003 x64/ XP x64.

В каждом из этих разделов представлены три варианта дистрибутива:

• Essentials – базовый вариант без опциональных компонентов. Включает мастер настройки (Configuration Wizard);

• ZIP/Setup.EXE – полный вариант, включающий опцио-

нальные утилиты, документацию и др., а также мастер настройки;

• Without installer – полный вариант, который не включает мастер настройки и требует ручной установки и настройки.

Рекомендуется использовать первый или второй вариант. Возможностей, предоставляемых первым вариантом, в большинстве случаев достаточно. В этой книге будет описана настройка MySQL только с помощью мастера настройки.

Если вы предпочитаете работать в графическом интерфейсе, а не в командной строке, то можете также скачать пакет графических утилит MySQL GUI Tools (http://dev.mysql.com/downloads/gui-tools/5.0.html), включающий три программы:

• MySQL Administrator – инструмент администрирования, конфигурирования, мониторинга, запуска/остановки сервера MySQL, управления пользователями и соединениями;

• MySQL Query Browser – инструмент создания, выполнения и оптимизации запросов в графической среде;

• MySQL Migration Toolkit – инструмент для переноса данных в MySQL из других реляционных баз данных (Oracle, Microsoft SQL Server, Microsoft Access, Sybase и др.).

Примечание

В этой книге будет кратко рассказано, как пользоваться утилитами MySQL Administrator и

MySQL Query Browser, а также описано выполнение операций с базой данных в командной строке.

Вы скачали необходимые дистрибутивы. Теперь приступим к установке СУБД MySQL 5.0.

Установка сервера MySQL

Чтобы установить на компьютере Windows-версию программы MySQL, выполните следующие действия.

1. Запустите мастер установки (Setup Wizard):

• если вы скачали базовый вариант дистрибутива MySQL, то дважды щелкните на значке файла mysql-essential-5.0.xxx-win32.msi.;

• если вы скачали полный вариант дистрибутива MySQL, то извлеките из архива файл Setup.exe и запустите его, дважды щелкнув на значке файла.

2. В начальном окне мастера установки (рис. 1.1) нажмите кнопку Next (Далее).

Рис. 1.1. Начальное окно мастера установки

3. Выберите тип установки (рис. 1.2):

• Typical (Обычная) – будут установлены только основные компоненты MySQL: сервер и утилиты командной строки;

• Complete (Полная) – будут установлены все компоненты MySQL, в том числе библиотеки и заголовочные файлы;

• Custom (Выборочная) – вы сможете указать путь к каталогу, в котором будет установлена программа MySQL, и выбрать компоненты, которые требуется установить.

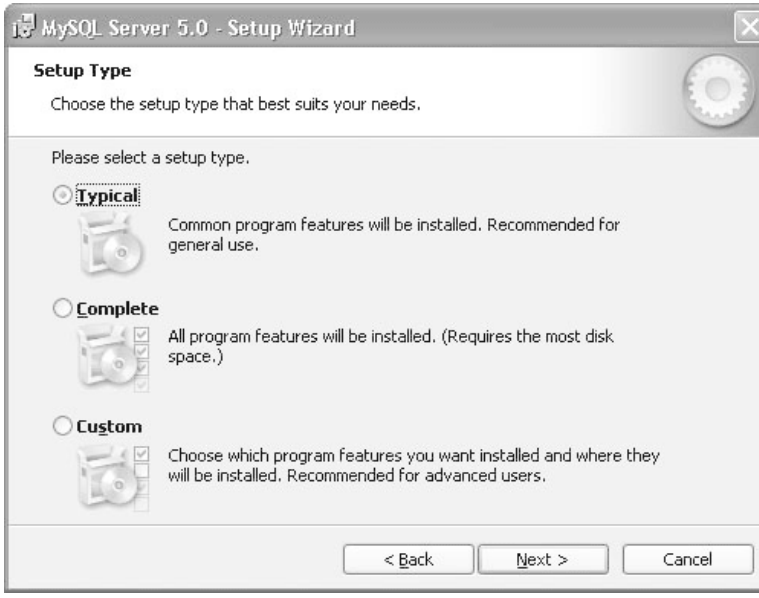

Рис. 1.2. Выбор типа установки

Рекомендуется выбрать вариант Custom (Выборочная), иначе выбор каталога установки будет недоступен. Нажмите кнопку Next (Далее).

Если вы выбрали тип установки Typical (Обычная) или Complete (Полная), то следующий пункт пропустите.

Если вы выбрали тип установки Custom (Выборочная), то укажите параметры установки (рис. 1.3).

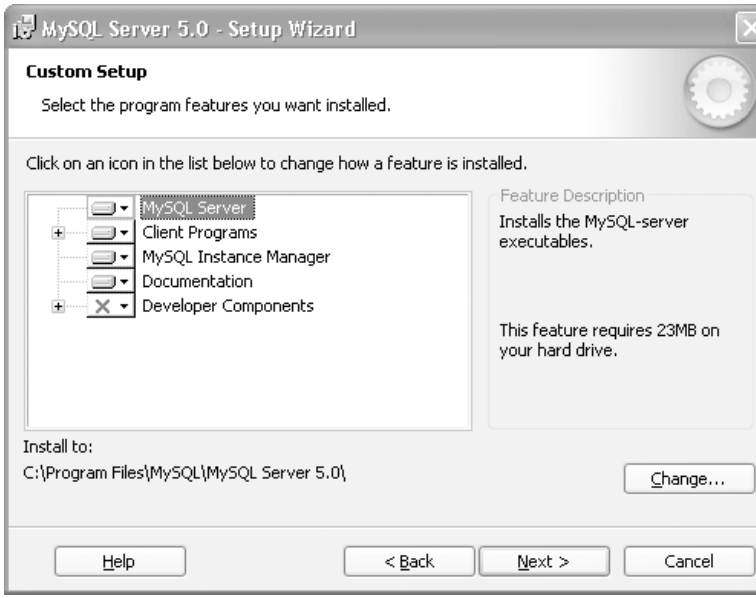

Рис. 1.3. Выбор параметров установки

• Если необходимо включить в установку или исключить из нее какой-либо компонент, найдите его в дереве компонентов, щелкните на значке слева от названия компонента и в контекстном меню выберите нужное действие (

– компонент будет установлен,

– компонент не будет установлен). Если вы пока не знаете точно, какие компоненты вам потребуются, рекомендую оставить набор компонентов неизменным.

• Если необходимо изменить каталог установки, нажмите кнопку Change (Изменить). В появившемся окне (рис. 1.4) введите путь к каталогу в поле Folder name (Имя каталога) либо в поле Look in (Смотреть в) и выберите из списка нужный диск, а затем последовательно раскройте вложенные папки, пока не дойдете до нужной. Нажмите кнопку OK.

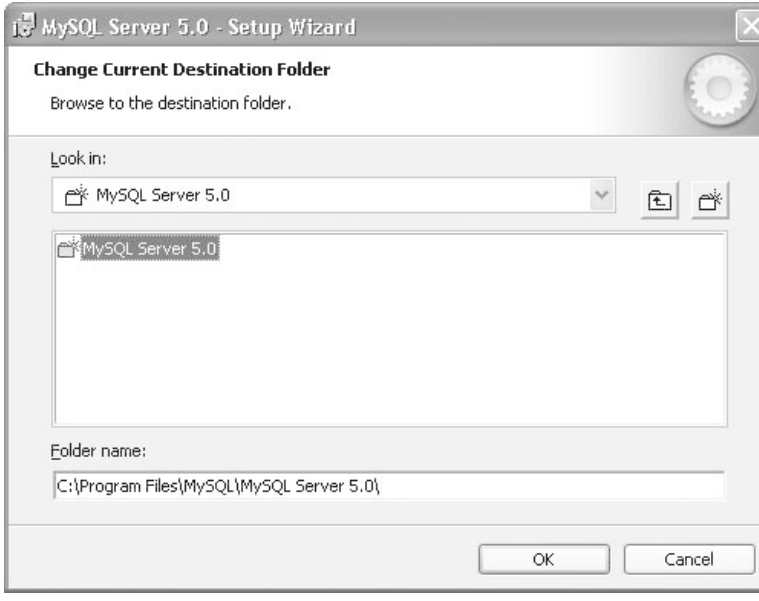

Рис. 1.4. Выбор каталога установки

Завершив настройку параметров установки, нажмите кнопку Next (Далее).

4. Проверьте правильность выбранного типа и каталога установки (рис. 1.5). Если параметры необходимо изменить, нажмите кнопку Back (Назад). Если параметры указаны верно, для запуска установки нажмите кнопку Install (Установить).

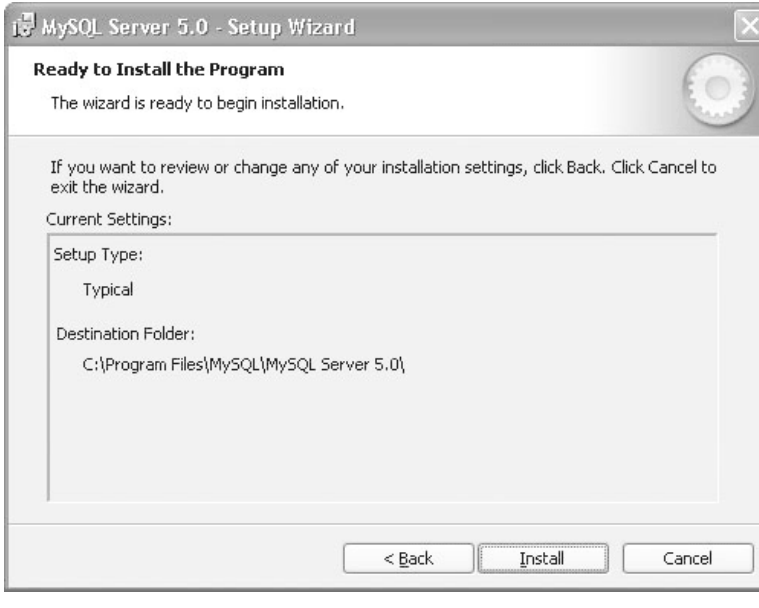

Рис. 1.5. Подтверждение параметров

5. После окончания установки на экране появится информационное окно (рис. 1.6). В этом и последующих аналогич-

ных окнах просто нажмите кнопку Next (Далее).

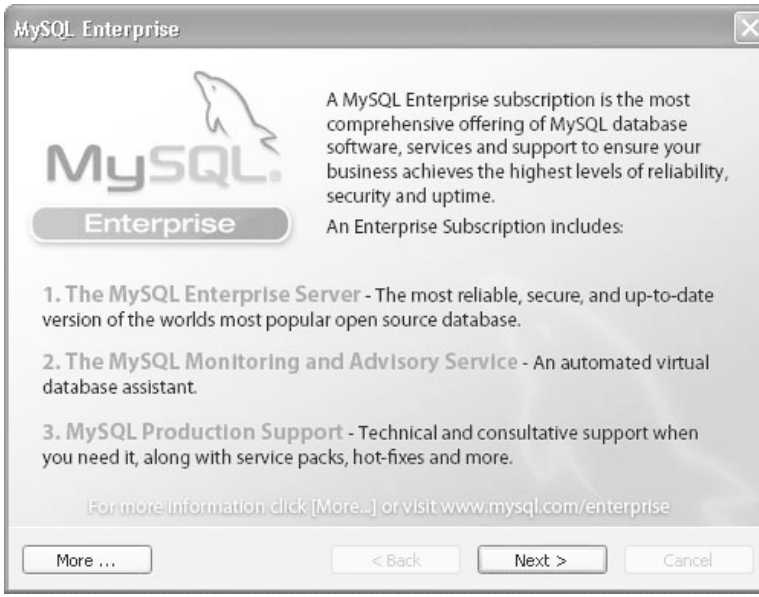

Рис. 1.6. Информационное окно

6. Укажите необходимость запуска мастера настройки, установив флажок Configure the MySQL Server now (Конфигурировать сервер MySQL сейчас). Нажмите кнопку Finish (Готово) (рис. 1.7).

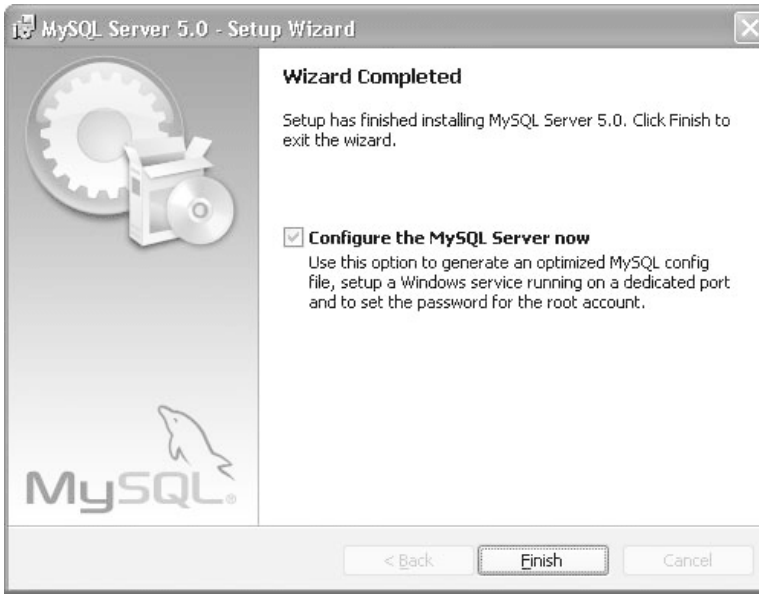

Рис. 1.7. Завершение установки MySQL После установки необходимо выполнить настройку MySQL.

Настройка сервера MySQL

Для задания конфигурационных параметров сервера MySQL удобно использовать мастер настройки (Configuration Wizard). Он автоматически запускается, если вы установили флажок Configure the MySQL Server now (Конфигурировать сервер MySQL сейчас). Вы также можете запустить мастер настройки, нажав кнопку Пуск и выбрав

последовательно пункты меню Все программы → MySQL → MySQL Server 5.0 → MySQL Server Instance Config Wizard.

Для настройки сервера MySQL с помощью мастера настройки выполните следующие действия.

1. В начальном окне мастера настройки (рис. 1.8) нажмите кнопку Next (Далее).

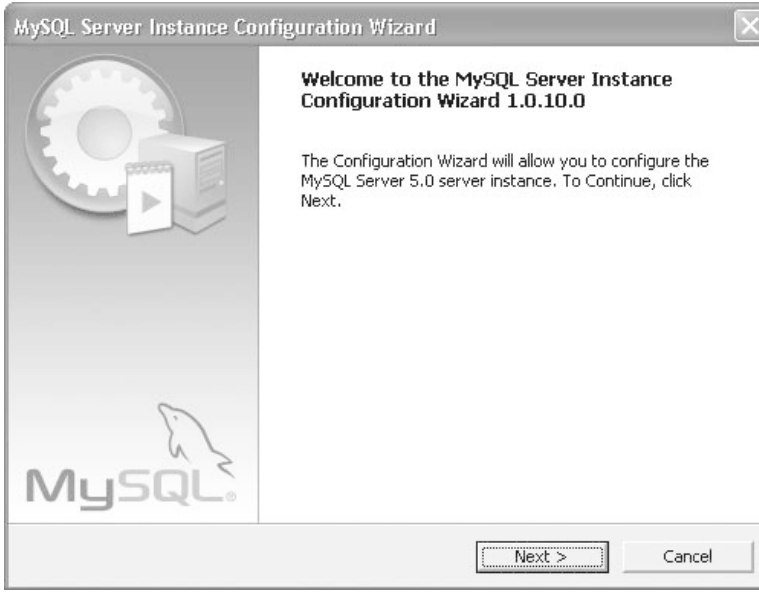

Рис. 1.8. Начальное окно мастера настройки

2. Выберите режим настройки сервера (рис. 1.9):

• Detailed Configuration (Настройка в подробном режиме) – этот вариант настройки ориентирован на опытных

пользователей и предоставляет возможность выбрать конфигурацию сервера и указать множество конфигурационных параметров;

• Standard Configuration (Настройка в стандартном режиме) – этот вариант потребует от вас минимум усилий, поскольку большая часть параметров будет задана автоматически.

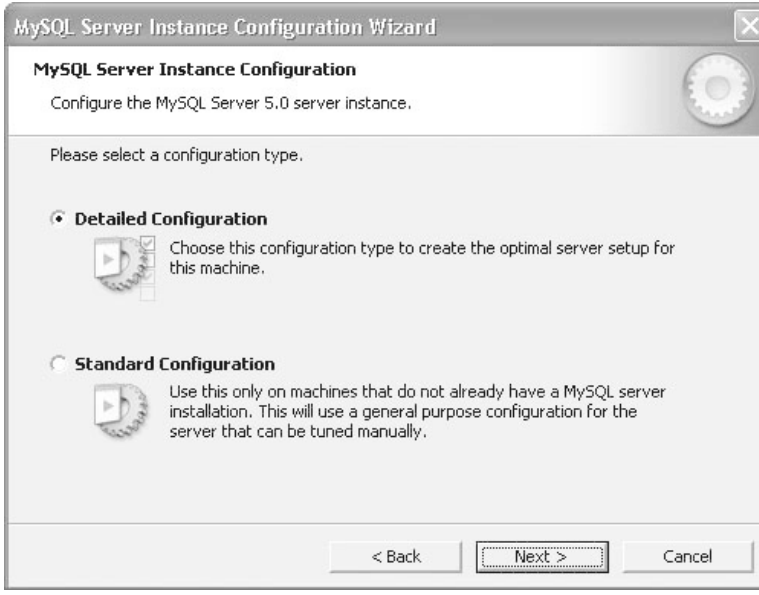

Рис. 1.9. Выбор режима настройки

Рекомендуется выбрать вариант Detailed Configuration (Настройка в подробном режиме), так как он позволит вам

указать ряд важных параметров работы сервера.

Нажмите кнопку Next (Далее).

Если вы выбрали вариант Standard Configuration (Настройка в стандартном режиме), то пункты 3–8 пропустите.

3. Выберите конфигурацию MySQL, которая зависит от того, на каком компьютере будет функционировать программа (рис. 1.10):

• Developer Machine (Компьютер разработчика) – данная конфигурация подходит для персонального компьютера, где одновременно с MySQL будет запускаться множество других программ. Конфигурация требует минимального количества системных ресурсов (оперативной памяти, дискового пространства, загрузки процессора);

• Server Machine (Сервер) – данная конфигурация MySQL подходит для сервера, где одновременно работает несколько серверных приложений (например, для веб-сервера). Требует среднего количества системных ресурсов;

• Dedicated MySQL Server Machine (Выделенный MySQL-сервер) – данная конфигурация походит для выделенного сервера, где будет работать только MySQL. Требует максимального количества ресурсов.

Рис. 1.10. Выбор конфигурации сервера

Если вы установили программу MySQL на своем персональном компьютере, рекомендуется выбрать вариант Developer Machine (Компьютер разработчика). Нажмите кнопку Next (Далее).

4. Выберите тип базы данных (рис. 1.11) в зависимости от того, какие типы таблиц вы преимущественно планируете использовать. Основными типами таблиц в MySQL являются InnoDB и MylSAM. Таблицы InnoDB обеспечивают высокую эффективность операций изменения данных в много-

пользовательском режиме благодаря поддержке транзакций (понятие транзакции мы рассматривали в подразделе «Целостность данных») и блокировок отдельных строк. Таблицы MylSAM не поддерживают обработку транзакций, зато обеспечивают отличную производительность операций поиска и чтения данных.

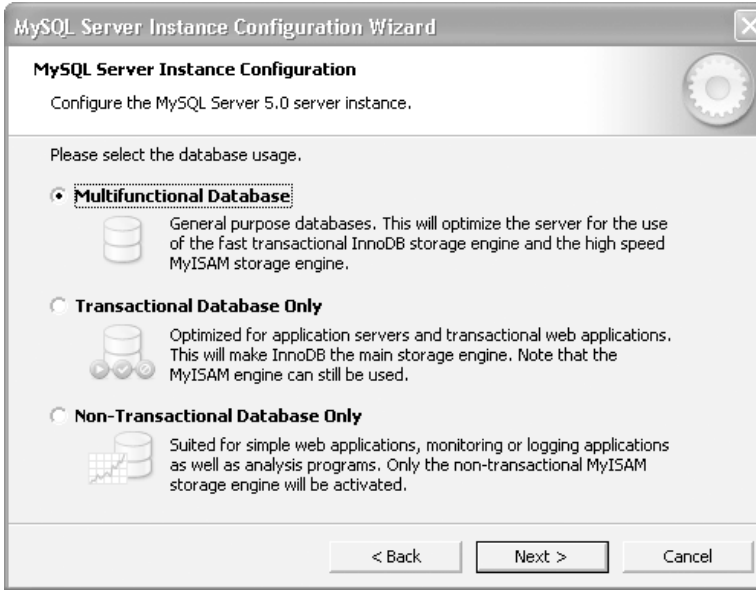

Рис. 1.11. Выбор типа базы данных Вы можете выбрать одно из следующих значений:

• Multifunctional Database (Многофункциональная база данных) – база данных общего назначения, оптимизирован-

ная для работы как с таблицами InnoDB, так и с MylSAM;

• Transactional Database Only (Транзакционная база данных) – база данных, оптимизированная главным образом для работы с таблицами InnoDB (однако другие типы таблиц также используются). Этот тип подходит для применения в корпоративных информационных системах;

• Non-Transactional Database Only (Нетранзакционная база данных) – база данных, включающая только таблицы без поддержки транзакций (MylSAM и др.). Подходит для использования в веб-приложениях и системах анализа данных.

Если вы пока не знаете, потребуется ли вам поддержка транзакций, то выберите значение Multifunctional Database. Нажмите кнопку Next (Далее).

Если вы выбрали вариант Non-Transactional Database Only (Нетранзакционная база данных), то следующий пункт пропустите.

5. Если вы выбрали вариант Multifunctional Database (Многофункциональная база данных) или Transactional Database Only (Транзакционная база данных), то при необходимости можно изменить каталог файловой системы, где будут храниться файлы табличной области InnoDB (рис. 1.12). Размещение табличной области на отдельном физическом носителе может использоваться для повышения емкости или производительности базы данных.

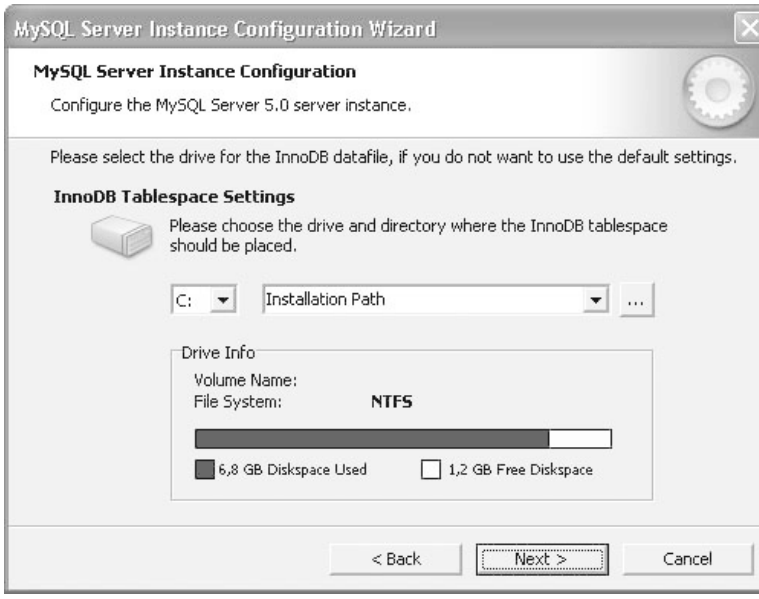

Рис. 1.12. Выбор каталога для размещения табличной области InnoDB

Рекомендуется использовать каталог по умолчанию. В этом случае просто нажмите кнопку Next (Далее).

Если требуется изменить каталог, выберите из списка нужный файловый носитель (диск). В нижней части окна отобразится информация об объеме свободного и занятого пространства на этом носителе. Затем введите путь к каталогу либо нажмите кнопку

и выберите папку в стандартном окне Windows для открытия файла. В завершение нажмите кнопку Next (Далее).

6. Укажите предполагаемое количество пользователей/приложений, одновременно подключенных к серверу (рис. 1.13):

• Decision Support (DSS)/OLAP (Система поддержки принятия решений/аналитической обработки данных) – прогнозируемое количество одновременных соединений составляет в среднем 20; допускается не более 100 одновременных соединений;

• Online Transaction Processing (OLTP) (Система оперативной обработки транзакций/массового ввода и модификации данных) – допускается не более 500 одновременных соединений;

• Manual Setting (Ручная установка) – выберите из списка или введите максимальное количество одновременных соединений.

Нажмите кнопку Next (Далее).

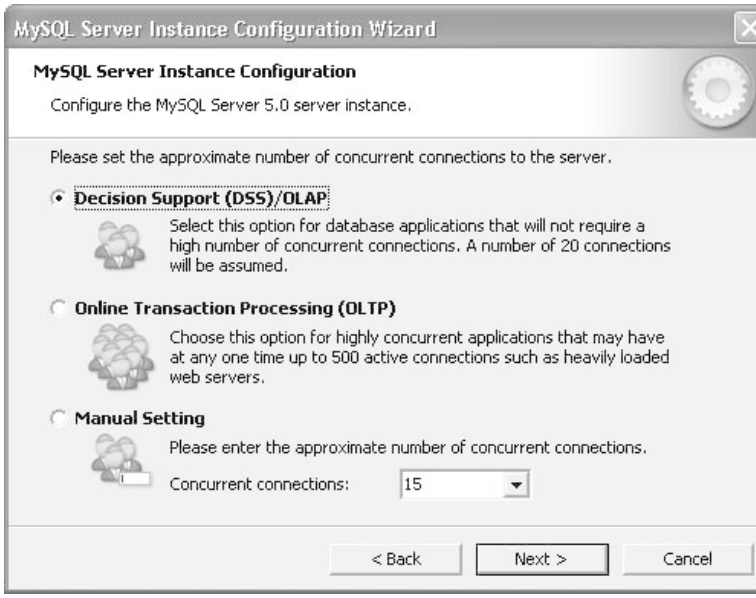

Рис. 1.13. Задание максимального количества одновременных соединений

7. Определите необходимость разрешить удаленные подключения и возможность использования строгого режима (рис. 1.14).

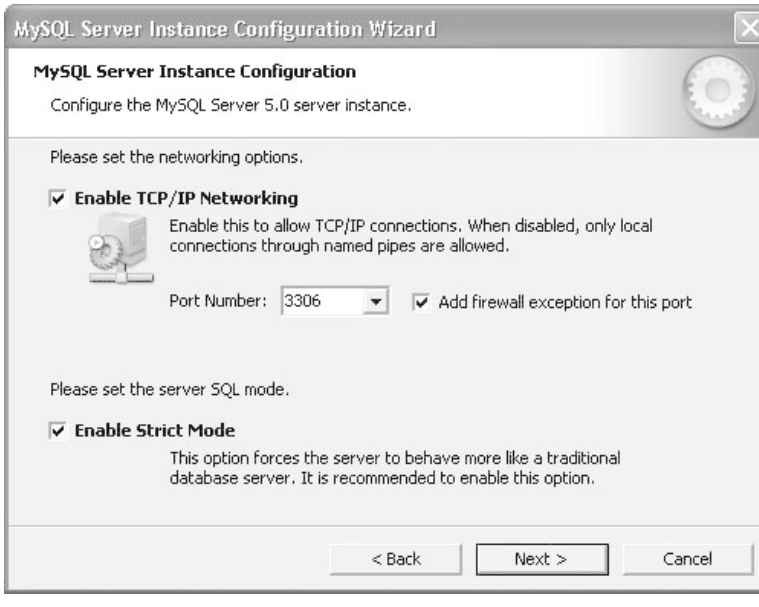

Рис. 1.14. Включение необходимых режимов работы

• Если вы предполагаете разрешить пользователям подключаться к серверу MySQL с удаленных компьютеров, то должен быть установлен флажок Enable TCP/IP Networking (Разрешить TCP/IP-соединения).

В этом случае введите в поле Port Number (Номер порта) номер порта (по умолчанию используется 3306). Установите флажок Add firewall exception for this port (Добавить исключение брандмауэра для этого порта) для автоматического открытия этого порта в брандмауэре Windows. Также вы

можете открыть этот порт вручную (Пуск → Панель управления → Брандмауэр Windows → Исключения → Добавить порт). Рекомендуется не менять номер порта по умолчанию и установить флажок Add firewall exception for this port (Добавить исключение брандмауэра для этого порта).

• Если необходимо разрешить использование строгого режима, установите флажок Enable Strict Mode (Разрешить строгий режим). В строгом режиме при попытке ввода в таблицу некорректного значения операция отменяется и выдается сообщение об ошибке (в обычном режиме некорректное значение заменяется подходящим и выдается предупреждение). Рекомендуется установить этот флажок.

• Если требуется разрешить подключение к серверу MySQL с удаленных компьютеров, то должен быть установлен флажок Enable TCP/IP Networking (Разрешить TCP/IP-соединения). В этом случае введем в поле Port Number (Номер порта) номер порта (по умолчанию используется 3306). Установите флажок Add firewall exception for this port (Добавить исключение брандмауэра для этого порта) для автоматического открытия этого порта в брандмауэре Windows. Вы можете открыть этот порт и вручную (Пуск → Панель управления → Брандмауэр Windows → Исключения → Добавить порт).

Нажмите кнопку Next (Далее).

8. Выберите кодировку (кодовую страницу), используемую по умолчанию для данных в базе (рис. 1.15).

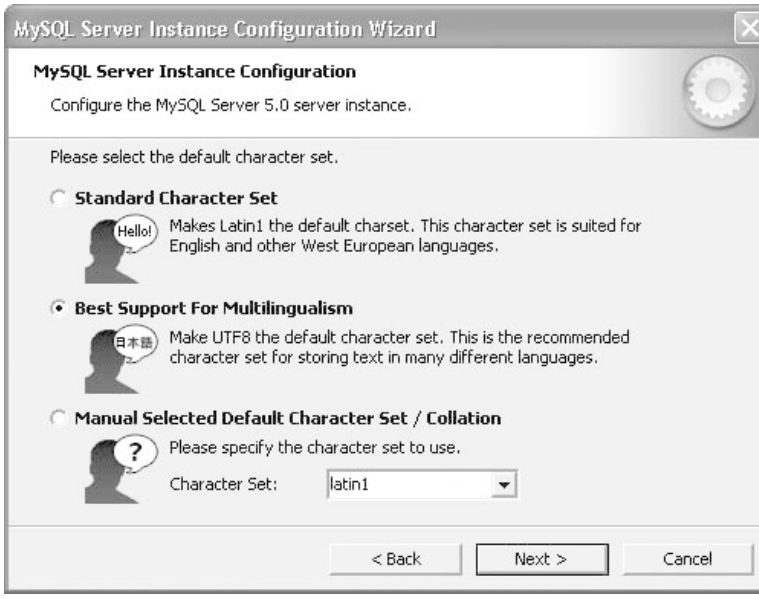

Рис. 1.15. Выбор кодировки по умолчанию

Кодировка – это таблица соответствия между символами (в частности, символами национальных алфавитов), которые отображаются на экране, и числовыми кодами символов, с которыми работает программа. Для кодирования (представления) текстов на русском языке традиционно применяются несколько различных кодировок, такие как KOI8-R, CP-866 (кодировка DOS), CP-1251 (кодировка Windows), UTF-8 (Unicode) и др. Итак, выберите одно из следующих значений:

• Standard Character Set – кодировка Latin1;

• Best Support For Multilingualism (Наилучшая поддержка мультиязычности) – кодировка Unicode (UTF-8);

• Manual Selected Default Character Set / Collation (Ручная установка кодировки и правил сортировки по умолчанию) – выберите кодировку из списка поддерживаемых кодировок.

Если вы еще не знаете, какая кодировка для вашей базы данных предпочтительнее, рекомендуется выбрать значение Best Support For Multilingualism (Наилучшая поддержка мультиязычности). Впоследствии вы сможете, независимо от выбранной кодировки по умолчанию, назначать другие кодировки для отдельных таблиц и даже столбцов.

Нажмите кнопку Next (Далее).

9. Задайте параметры Windows, которые будут использоваться программой MySQL (рис. 1.16).

Рис. 1.16. Выбор параметров Windows

• Если необходимо сконфигурировать MySQL как сервис Windows, установите флажок Install As Windows Service (Установить как сервис Windows). При этом вы можете изменить имя сервиса и указать его автоматический запуск при запуске Windows, установив флажок Launch the MySQL Server automatically (Автоматически запускать сервер MySQL).

• Установите флажок Include Bin Directory in Windows PATH (Включить каталог bin в переменную Windows PATH),

чтобы при запуске сервера и утилит из командной строки не надо было указывать полный путь к ним (поскольку путь к подкаталогу bin будет добавлен в значение системной переменной Path).

> **Внимание!**
>
> Если вы сняли флажок Install As Windows Service (Установить как сервис Windows), следующий пункт пропустите. В этом случае для обеспечения безопасности необходимо вручную установить пароль пользователя root при первом запуске сервера MySQL.

Рекомендуется в данном окне установить все три флажка.

Нажмите кнопку Next (Далее).

10. Настройте параметры безопасности MySQL (рис. 1.17).

Рис. 1.17. Настройка параметров безопасности

• Введите в поля New root password (Новый пароль root) и Confirm (Подтверждение) пароль пользователя root (этот пользователь обладает правами для проведения любых действий в MySQL).

• Вы можете разрешить пользователю root подключаться к серверу с удаленных компьютеров. Для этого установите флажок Enable root access from remote machines (Разрешить пользователю root доступ с удаленных компьютеров). С точки зрения безопасности предпочтительнее запретить поль-

зователю root доступ с удаленных компьютеров.

• Если необходимо разрешить пользователям анонимный доступ, установите флажок Create An Anonymous Account (Создать анонимного пользователя). Это делать не рекомендуется, так как снижается защищенность базы данных.

Нажмите кнопку Next (Далее).

11. Для запуска процесса конфигурирования нажмите кнопку Execute (Выполнить) (рис. 1.18).

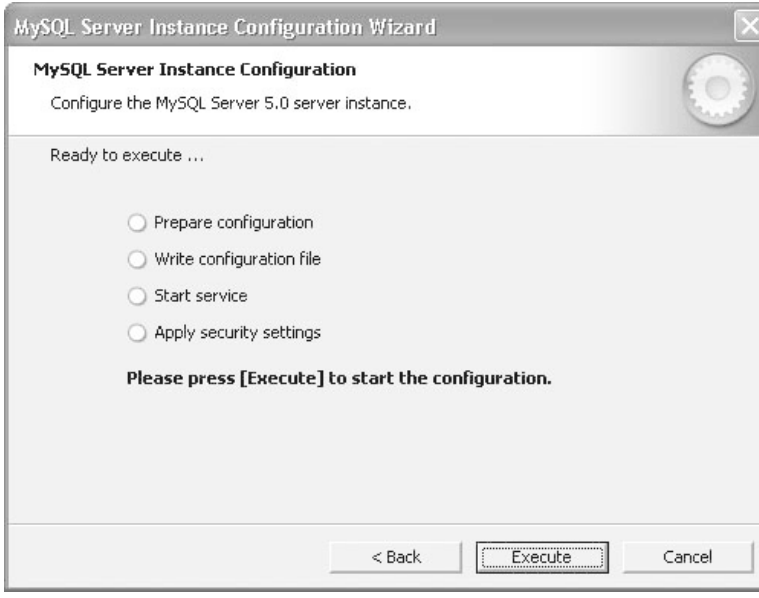

Рис. 1.18. Конфигурирование MySQL

12. По окончании конфигурирования нажмите кнопку

Finish (Готово).

Выполненные настройки можно посмотреть в файле my.ini, расположенном в каталоге, где установлена программа MySQL.

Если вы указали необходимость сконфигурировать MySQL как сервис Windows, мастер настройки создаст и запустит этот сервис. В противном случае нужно запустить сервер вручную. После запуска вы можете подключиться к серверу как пользователь root с паролем, который вы ввели при настройке параметров безопасности (или с пустым паролем, если вы не вводили пароль при настройке). Об этом пойдет речь в разделе «Начало работы в MySQL».

Далее мы рассмотрим установку графических утилит MySQL (о которых было сказано в подразделе «Загрузка MySQL»). Если вы решили использовать только командную строку, то можете перейти к следующему разделу.

Установка MySQL GUI Tools

Чтобы установить графические утилиты MySQL, выполним следующие действия.

1. Запустите мастер установки MySQL GUI Tools (Setup Wizard), дважды щелкнув на значке файла mysql-gui-tools-5.0xxx-win32.msi.

2. В начальном окне мастера установки (рис. 1.19) нажмите кнопку Next (Далее).

Рис. 1.19. Начальное окно мастера установки

3. Подтвердите согласие с лицензионным соглашением, выбрав значение переключателя I accept the terms in the license agreement (Я принимаю условия лицензионного соглашения) (рис. 1.20). Нажмите кнопку Next (Далее).

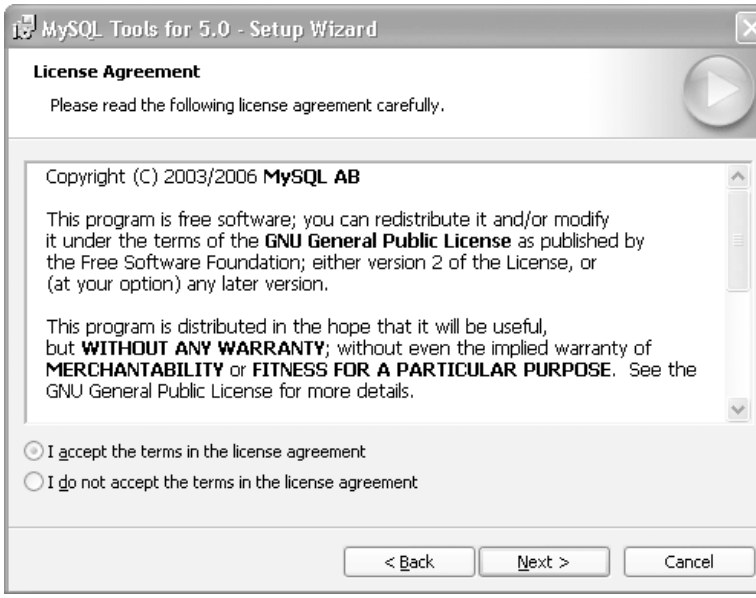

Рис. 1.20. Лицензионное соглашение

4. Если необходимо изменить каталог установки утилит (рис. 1.21), нажмите кнопку Change (Изменить).

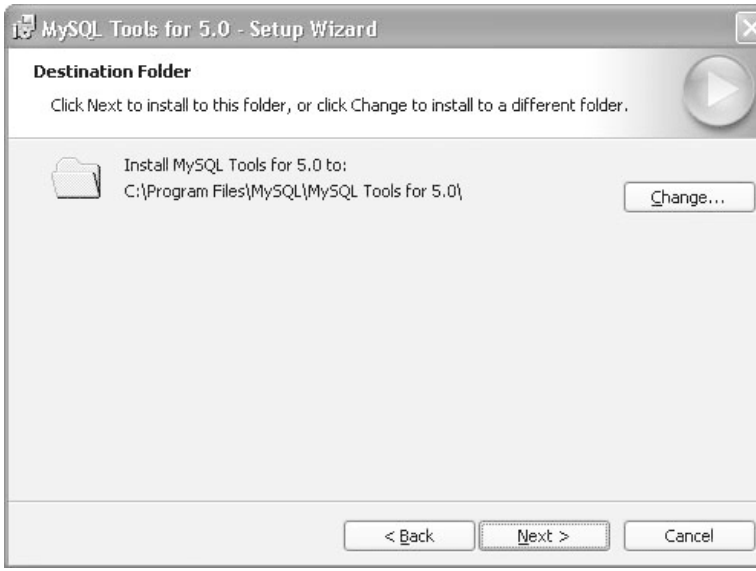

Рис. 1.21. Настройка каталога установки

В появившемся окне для выбора каталога установки (см. рис. 1.4) введите нужный путь к каталогу в поле Folder name (Имя каталога) либо в поле Look in (Смотреть в) выберите из списка нужный диск, а затем последовательно раскройте вложенные папки, пока не дойдете до нужной. Нажмите кнопку OK.

Для продолжения установки нажмите кнопку Next (Далее).

5. Выберите тип установки (рис. 1.22):

• Complete (Полная) – установка всех графических утилит

пакета;

• Custom (Выборочная) – установка отдельных компонентов.

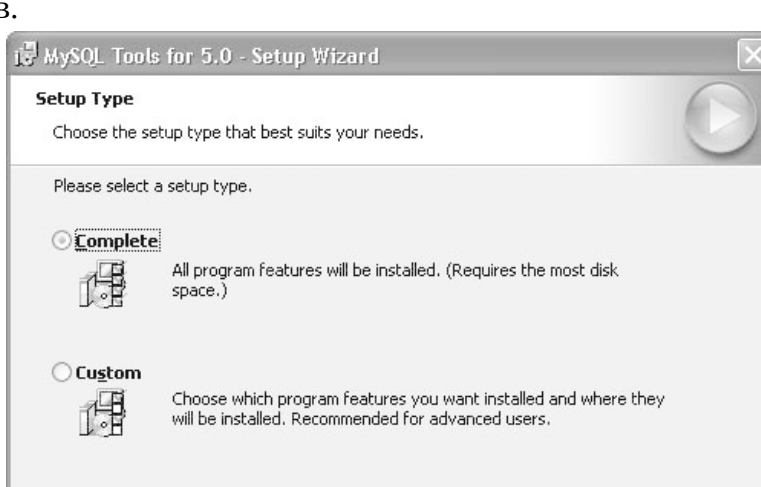

Рис. 1.22. Выбор типа установки

Если вы хотите установить все три графические утилиты, выберите вариант Complete (Полная). Если же вы не планируете использовать какую-либо из этих утилит, выберите вариант Custom (Выборочная). Нажмите кнопку Next (Далее).

Если вы выбрали тип установки Complete (Полная), следующий пункт пропустите.

6. Если вы выбрали тип установки Custom (Выборочная),

то укажите набор устанавливаемых компонентов (рис. 1.23). Если необходимо включить в установку или исключить из нее какой-либо компонент, найдите его в дереве компонентов, щелкните на значке слева от названия компонента и в контекстном меню выберите нужное действие (

– компонент будет установлен,

– компонент не будет установлен).

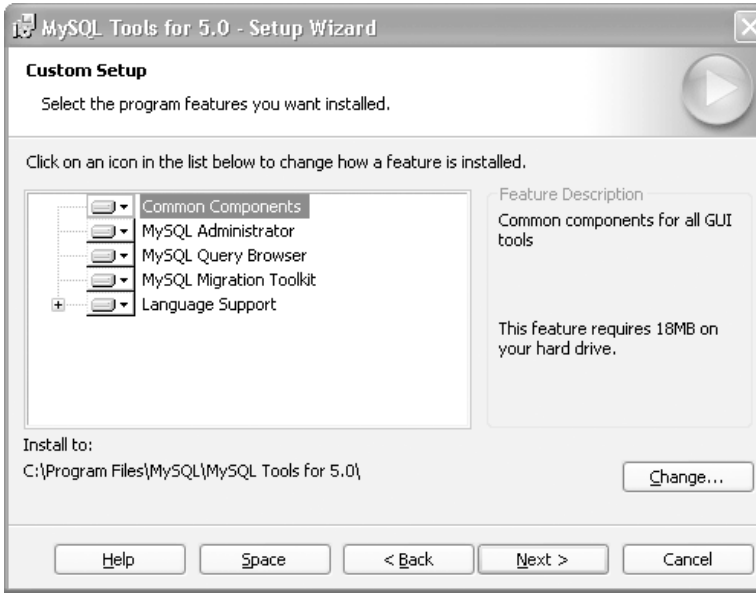

Рис. 1.23. Выбор устанавливаемых компонентов

Определив набор устанавливаемых компонентов, нажмите кнопку Next (Далее).

7. Проверьте правильность выбранного типа установки и каталога установки (рис. 1.24). Если параметры необходимо изменить, нажмите кнопку Back (Назад). Если параметры указаны верно, для запуска установки нажмите кнопку Install (Установить).

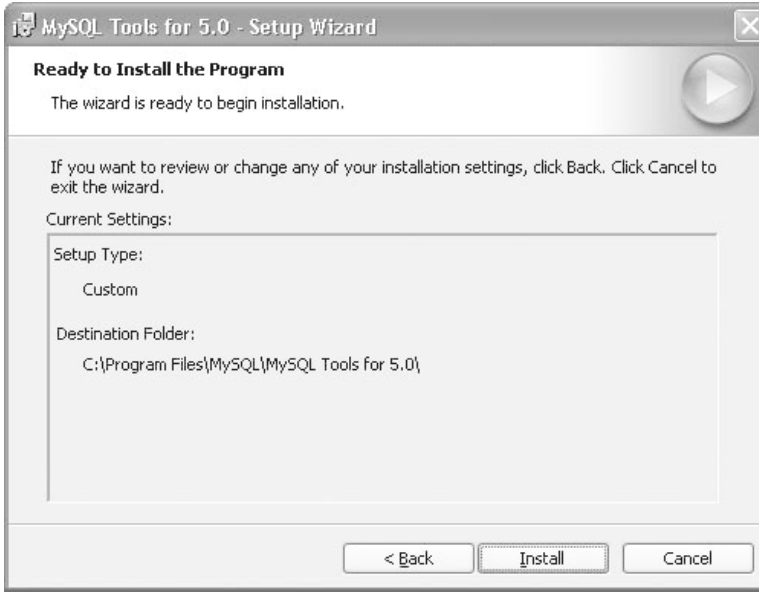

Рис. 1.24. Подтверждение параметров

8. После окончания установки на экране появится информационное окно (см. рис. 1.6). В этом и последующих аналогичных окнах просто нажмите кнопку Next (Далее). В последнем окне нажмите кнопку Finish (Готово).

Итак, установка MySQL завершена. Следующий этап – запуск сервера MySQL и подключение к нему. Об этом пойдет речь в следующем разделе.

1.5. Начало работы в MySQL

Чтобы работать с базой данных, вначале необходимо запустить сервер MySQL и подключиться к нему. Если при настройке сервер MySQL был сконфигурирован как сервис Windows, то он был автоматически запущен по окончании настройки. В противном случае сервер нужно запустить из командной строки (см. подраздел «Запуск и остановка сервера MySQL из командной строки») или с помощью графической утилиты MySQL Administrator (см. подраздел «Запуск и остановка сервера MySQL с помощью MySQL Administrator»).

Подключиться к работающему серверу можно из командной строки (см. подраздел «Подключение к серверу из командной строки») или с помощью графической утилиты MySQL Query Browser (см. подраздел «Подключение к серверу с помощью MySQL Query Browser»).

Запуск и остановка сервера MySQL из командной строки

Запустить сервер MySQL вручную можно одним из двух способов:

• Дважды щелкните на значке файла mysqld-nt.exe, расположенного в подкаталоге bin каталога, где установлена про-

грамма MySQL.

• Откройте окно командной строки Windows. Для этого нажмите кнопку Пуск, в меню выберите пункт Выполнить, в появившемся окне Запуск программы в поле Открыть введите команду cmd и нажмите кнопку OK. На экране появится окно командной строки (рис. 1.25).

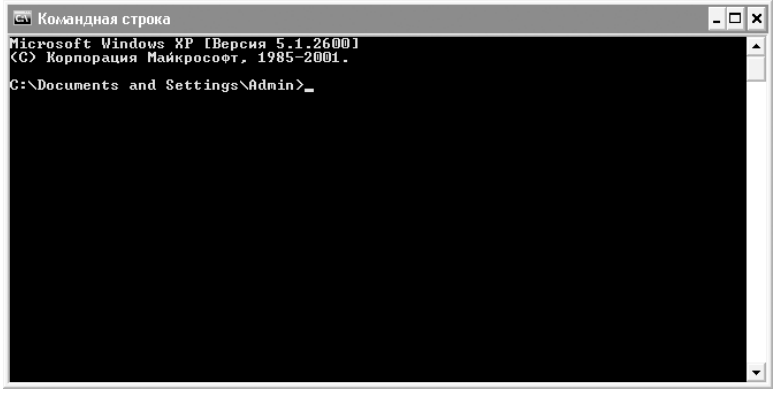

Рис. 1.25. Окно командной строки

В командной строке введите команду

mysqld-nt

и нажмите клавишу Enter. Сервер MySQL будет запущен.

Если при настройке сервера путь к подкаталогу bin не был добавлен в значение системной переменной Path, то для запуска сервера необходимо ввести не только имя файла, но и полный путь к нему, например:

C: \Program Files\MySQL\MySQL Server 5.0\bin\mysqld-nt

Если вы хотите просматривать в окне командной строки диагностические сообщения о работе сервера, вместо mysqld-nt введите

mysqld-nt –console

> **Внимание!**
>
> Если при настройке сервера MySQL вы не указывали пароль пользователя root, то необходимо установить пароль при первом запуске сервера (иначе кто угодно сможет управлять сервером под именем root без пароля).

Чтобы установить пароль root, откройте новое окно командной строки и введите следующую команду:

mysqladmin -u root password <пароль>

(или C: \Program Files\MySQL\MySQL Server 5.0\bin \mysqladmin -u root password <пароль>, если путь к подкаталогу bin не был добавлен в значение системной переменной Path при настройке сервера) и нажмите клавишу Enter.

В дальнейшем, если потребуется сменить пароль пользователя root, выполните такую же команду, только с использованием опции – p:

mysqladmin -u root -p password <новый пароль>

После появления приглашения Enter password (Введите пароль) укажите прежний пароль и нажмите клавишу Enter.

Наконец, если необходимо остановить сервер MySQL, выполните команду

mysqladmin -u root -p shutdown

и в ответ на приглашение Enter password (Введите пароль) введите пароль пользователя root. Нажмите клавишу Enter. Сервер MySQL будет остановлен.

Для запуска и остановки сервера MySQL можно также использовать графическую утилиту MySQL Administrator.

Запуск и остановка сервера MySQL с помощью MySQL Administrator

Чтобы запустить сервер MySQL с помощью графической утилиты MySQL Administrator, выполните следующие действия.

1. Запустите программу MySQL Administrator (Пуск → Все программы → MySQL → MySQL Administrator). На экране появится окно соединения с сервером (рис. 1.26).

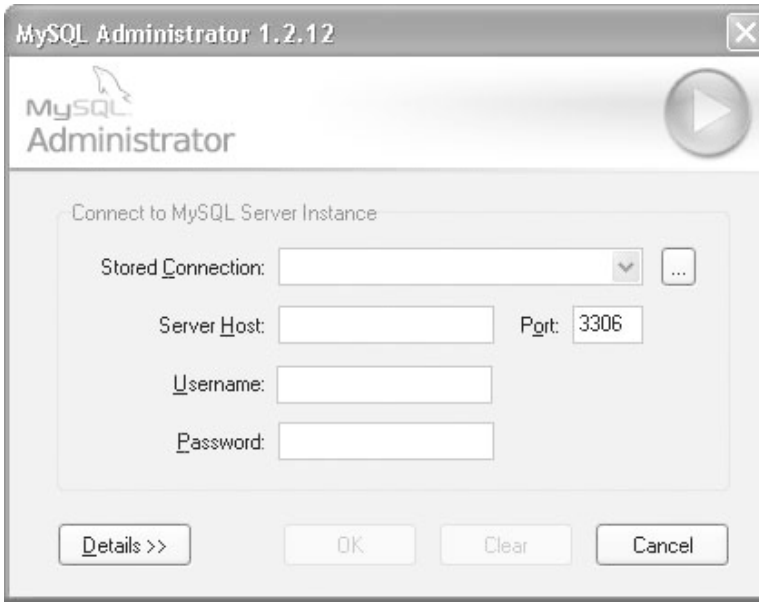

Рис. 1.26. Окно соединения с сервером MySQL

2. Нажмите клавишу Ctrl и, удерживая ее, щелкните на кнопку Skip (Пропустить), появившуюся в правом нижнем углу окна вместо кнопки Cancel (Отмена). На экране появится главное окно MySQL Administrator (рис. 1.27).

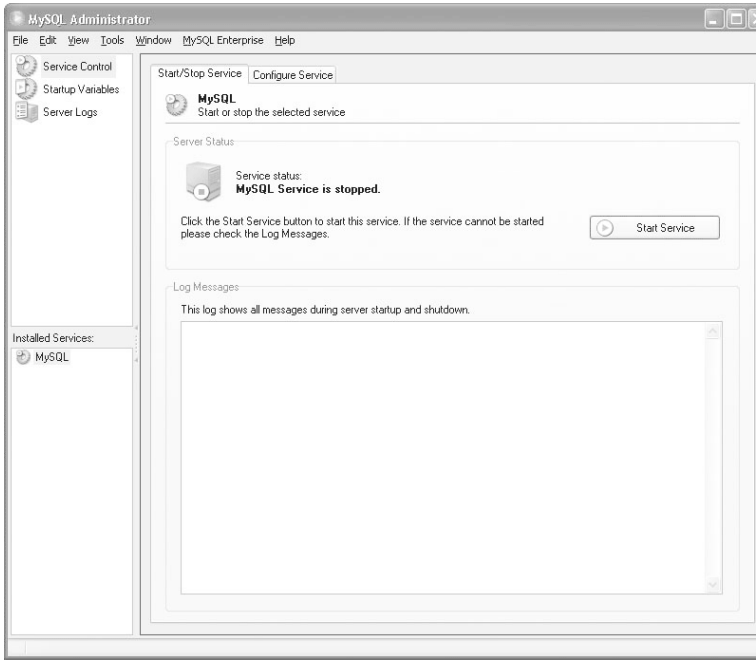

Рис. 1.27. Главное окно MySQL Administrator

3. В главном окне MySQL Administrator в левой области щелкните пункт Service Control (Управление сервисом).

4. Если сервер MySQL не был сконфигурирован как сервис Windows, то кнопка Start Service (Запустить сервис), расположенная в правой области окна, недоступна. Необходимо выполнить следующие предварительные действия:

1) перейдите на вкладку Configure Service (Настройка

сервиса). Найдите внизу вкладки кнопку Install new Service (Установить новый сервис) и нажмите ее;

2) в появившейся диалоговой панели укажите название сервиса и нажмите кнопку OK;

3) в поле Config Filename (Имя конфигурационного файла) введите путь к конфигурационному файлу my.ini (рис. 1.28), например C: \Program Files\ MySQL\MySQL Server 5.0\my.ini. Красный цвет шрифта означает, что файл не найден; если цвет сменился на обычный, то путь указан верно;

4) в поле Path to binary (Путь к исполняемому файлу) введите путь к файлу mysqld-nt.exe, например C: \Program Files \MySQL\MySQL Server 5.0\bin\mysqld-nt;

5) нажмите кнопку Apply Changes (Сохранить изменения);

6) вернитесь на вкладку Start/Stop Service (Запуск/остановка сервиса).

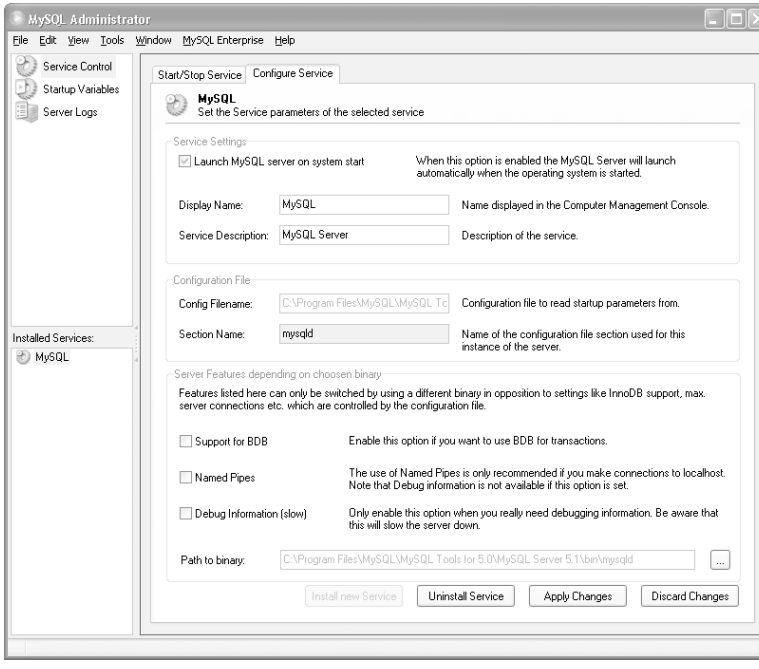

Рис. 1.28. Закладка Configure Service 5. Нажмите кнопку Start Service (Запустить сервис). Сервер MySQL будет запущен.

Внимание!

Если при настройке сервера MySQL вы не указывали пароль пользователя root, то необходимо установить его при первом запуске сервера (иначе кто угодно может управлять сервером под именем root без пароля). В текущей версии MySQL Administrator

установка пароля root недоступна, и для выполнения этой операции нужно использовать утилиту командной строки mysqladmin (см. подраздел «Запуск и остановка сервера MySQL из командной строки»).

Чтобы остановить сервер MySQL с помощью MySQL Administrator, выполните следующие действия.

1. Запустите программу MySQL Administrator (Пуск → Все программы → MySQL → MySQL Administrator). На экране появится окно соединения с сервером (см. рис. 1.26).

2. В поля окна соединения с сервером введите параметры соединения:

• Server Host (Имя хоста) – значение localhost (локальный компьютер);

• Port (Порт) – номер порта, выбранный при настройке сервера (по умолчанию – 3306);

• Usename (Имя пользователя) – значение root;

• Password (Пароль) – пароль пользователя root. Нажмите кнопку OK.

3. В главном окне MySQL Administrator в левой области щелкните пункт Service Control (Управление сервисом).

4. В правой области окна нажмите кнопку Stop Service (Остановить сервис). Сервер MySQL будет остановлен.

В следующем подразделе вы узнаете, как запустить сервер MySQL с помощью средств администрирования Windows.

Запуск и остановка сервера MySQL с панели управления

Если сервер MySQL был сконфигурирован как сервис Windows с помощью мастера настройки (см. подраздел «Настройка сервера MySQL») или с помощью утилиты MySQL Administrator (см. подраздел «Запуск и остановка сервера MySQL с помощью MySQL Administrator»), то запускать и останавливать его можно с помощью компонента Службы панели управления.

Чтобы вызвать компонент Службы, нажмите кнопку Пуск, в меню выберите пункт Панель управления, затем в панели управления дважды щелкните на значке Администрирование и, наконец, в окне средств администрирования дважды щелкните на значке Службы. На экране появится окно Службы (рис. 1.29) со списком всех локальных служб.

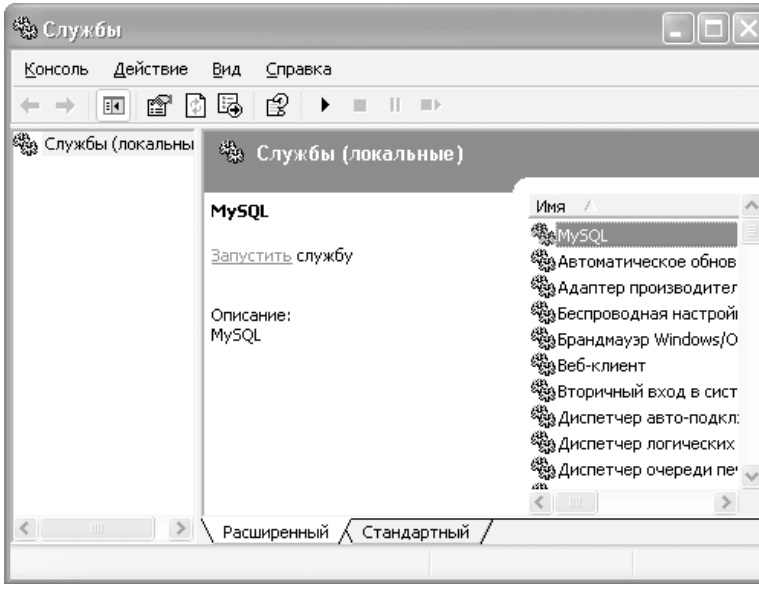

Рис. 1.29. Сервис MySQL в панели управления

В окне Службы щелкните на названии сервиса MySQL (название определяется при создании сервиса в мастере настройки или в MySQL Administrator). Затем щелкните на нужную ссылку под названием сервиса: Запустить службу, Остановить службу или Перезапустить службу.

После того как сервер MySQL запущен, к нему можно подключиться. В следующих подразделах вы узнаете, как это сделать.

Подключение к серверу из командной строки

Чтобы подключиться к серверу MySQL из командной строки, выполните следующие действия.

1. Откройте окно командной строки Windows. Для этого нажмите кнопку Пуск, в меню выберите пункт Выполнить, в появившемся окне Запуск программы введите в поле Открыть команду cmd и нажмите кнопку OK.

2. В командной строке (см. рис. 1.25) введите команду

mysql -h <Имя компьютера> -u <Имя пользователя> -p

(где <Имя компьютера> – это имя компьютера, на котором работает сервер) и нажмите клавишу Enter. После появления приглашения Enter password (Введите пароль) введите пароль пользователя.

Если требуется подключиться к серверу MySQL, работающему на этом же компьютере, имя компьютера (localhost) можно не указывать, например

mysql -u root -p

После подключения к серверу приглашение командной строки изменится на mysql> (рис. 1.30). Теперь можно приступать к работе с базой данных: добавлять таблицы, вводить и запрашивать данные, регистрировать новых пользователей и др.

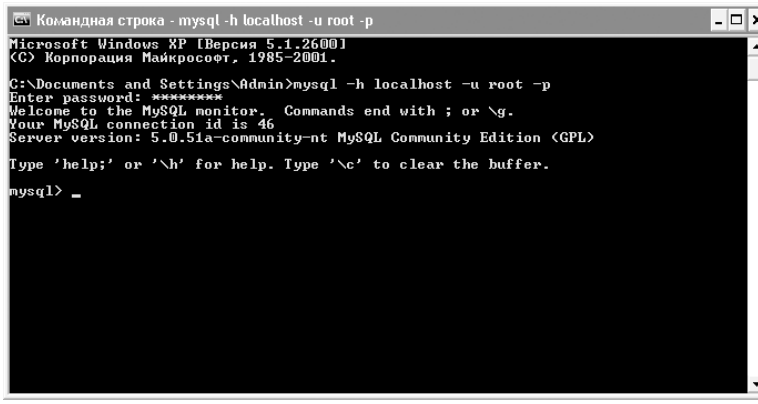

Рис. 1.30. Соединение с сервером MySQL из командной строки

Чтобы отключиться от сервера, просто наберите в командной строке команду

exit

и нажмите клавишу Enter.

Альтернативный способ подключения к серверу MySQL предоставляет графическая утилита MySQL Query Browser.

Подключение к серверу с помощью MySQL Query Browser

Утилита MySQL Query Browser – интерфейс для создания, редактирования и выполнения инструкций SQL. Она

удобнее, чем командная строка. Если вы решили использовать для работы с базой данных MySQL Query Browser, то для подключения к серверу выполните следующие действия.

1. Запустите программу MySQL Query Browser (Пуск → Все программы → MySQL → MySQL Query Browser). На экране появится окно соединения с сервером (рис. 1.31).

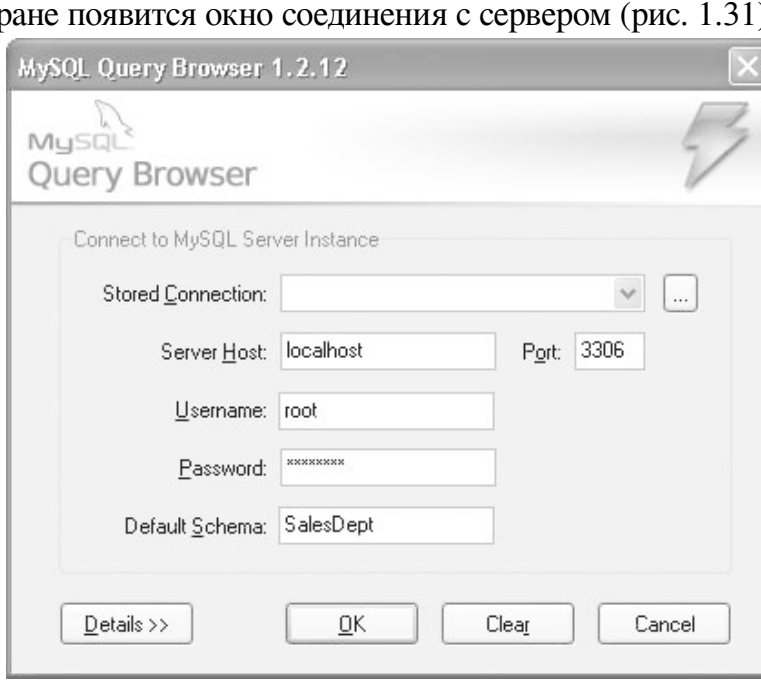

Рис. 1.31. Окно соединения с сервером MySQL

2. В поля окна соединения с сервером введите параметры соединения:

• Server Host (Имя хоста) – имя компьютера, на котором работает сервер MySQL;

• Port (Порт) – номер порта, выбранный при настройке сервера (по умолчанию – 330 6);

• Username (Имя пользователя) – имя пользователя;

• Password (Пароль) – пароль пользователя;

• Default Schema (Схема по умолчанию) – имя базы данных, с которой вы будете работать (это может быть как существующая, так и новая база данных).

3. Нажмите кнопку OK. Если вы ввели имя новой базы данных, то в появившейся диалоговой панели нажмите кнопку Yes (Да) для создания этой базы данных.

После подключения к серверу на экране появится главное окно MySQL Query Browser (рис. 1.32). В нем вы можете выполнять любые операции с базой данных: добавлять таблицы, вводить и запрашивать данные, регистрировать новых пользователей и др.

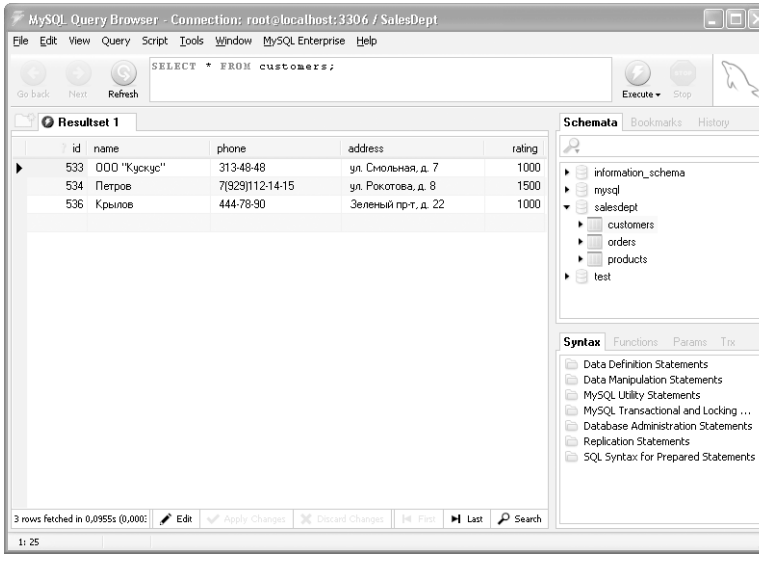

Рис. 1.32. Главное окно MySQL Query Browser

Внимание!

Шрифт, который по умолчанию используется в MySQL Query Browser для отображения SQL-запросов, не поддерживает русские буквы. Чтобы вводить русские буквы в текстах запросов, необходимо выбрать другой шрифт (например, Arial или Book Antiqua). Для этого в главном окне MySQL Query Browser откройте меню Tools (Сервис) и выберите пункт Options (Параметры). В появившемся окне Options (Параметры) в левой области щелкните пункт General Options (Общие параметры) и в правой области в поле Code Font

(Шрифт кода) выберите из списка нужный шрифт. Нажмите кнопку Apply (Сохранить).

Чтобы отключиться от сервера, просто закройте окно MySQL Query Browser.

На этом мы заканчиваем знакомство с MySQL и переходим к подведению итогов.

1.6. Резюме

В этой главе были рассмотрены СУБД MySQL и графические утилиты MySQL Administrator и MySQL Query Browser. Вы освоили достаточно сложную процедуру установки и настройки сервера MySQL, научились управлять сервером и подключаться к нему. Вы также узнали, как устроена реляционная база данных и как спроектировать собственную БД.

Итак, следующим этапом является построение базы данных в MySQL. Этому посвящена вторая глава. В ней будет рассказано, как создавать таблицы, вносить в них информацию и находить нужные сведения в базе данных.

Глава 2
Управление базой данных с помощью SQL

Из этой главы вы узнаете, как работать с данными в СУБД MySQL, как определять их структуру, а также как добавлять, изменять и удалять данные. Эти операции выполняет SQL – универсальный язык структурированных запросов, являющийся стандартным средством доступа к реляционным базам данных.

Для выполнения SQL-команд вы можете использовать любое из многочисленных клиентских приложений сервера MySQL. В этой главе не будут рассматриваться приложения сторонних разработчиков. Вы познакомитесь только с приложениями, созданными компанией MySQL AB: утилитой командной строки mysql и графической утилитой MySQL Query Browser.

В обеих утилитах доступны все операции с данными. В MySQL Query Browser удобно работать с базой данных: ее компоненты наглядно представлены, можно непосредственно редактировать данные (без использования SQL-оператора UPDATE), работать с запросами, например строить их с помощью специального инструмента (при этом названия таблиц и столбцов вводить вручную не нужно), сохранять за-

просы в файле, экспортировать результаты запросов и многое другое. Вы можете узнать о всех возможностях MySQL Query Browser, обратившись к документации на русском языке, найти которую можно по ссылке http://dev.mysql.com/doc/query-browser/ru/index.html.

Вначале вы узнаете, как выполнять SQL-команды в MySQL Query Browser и в командной строке, а в дальнейшем будет рассмотрен только синтаксис SQL-команд.

2.1. Выполнение SQL-команд

Прежде чем выполнять SQL-команды, необходимо подключиться к работающему серверу MySQL (как это сделать, рассказывалось в главе 1). В этом разделе вы узнаете, как создавать SQL-команды и передавать их серверу для выполнения.

Если вы используете командную строку, то для выполнения SQL-команды введем ее текст в окне командной строки и нажмем клавишу Enter для отправки команды на сервер. Чтобы избежать проблем с кодировкой русскоязычных данных, перед началом работы с данными выполните команду

SET NAMES cp866;

Результат выполнения этой команды вы видите на рис. 2.1.

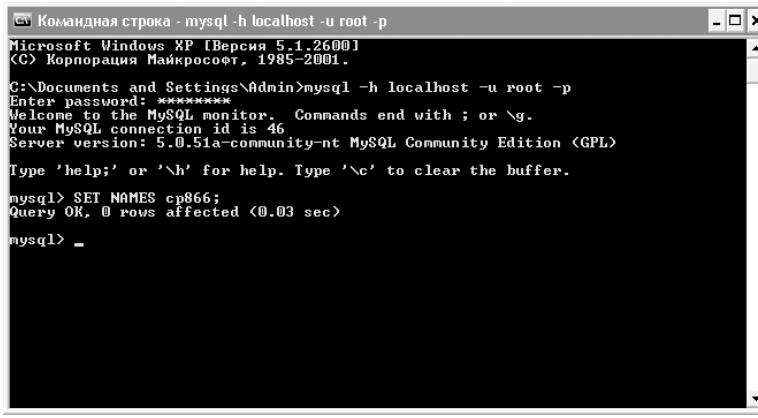

Рис. 2.1. Установка кодировки в командной строке

Команду SET NAMES необходимо повторять при *каждом* подключении к серверу с помощью командной строки. Эта команда указывает серверу, что данное клиентское приложение (утилита mysql) использует кодировку CP-866 (это кодировка командной строки Windows), и сервер будет автоматически выполнять преобразование кодировок при обмене данными с клиентским приложением.

После смены кодировки вы можете вводить в командной строке любые SQL-команды. Сообщение о результате выполнения команды, а также запрошенные данные выводятся непосредственно в окне командной строки (рис. 2.2).

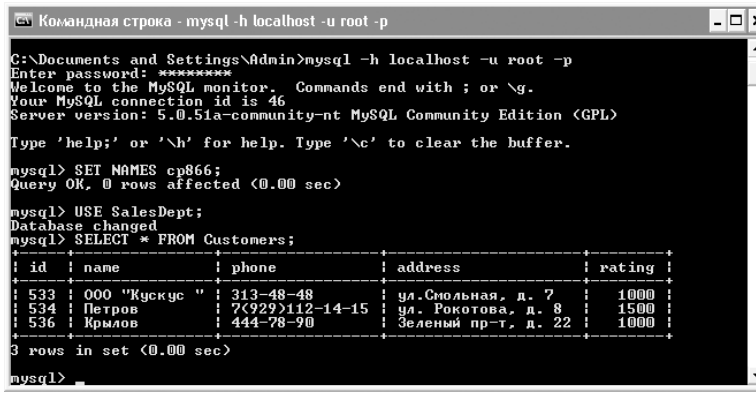

Рис. 2.2. Выполнение SQL-запроса в командной строке

Утилита mysql позволяет вводить и многострочные команды (на рис. 2.3 таким образом введена команда SHOW DATABASES). Если не введена точка с запятой – признак конца команды, то при нажатии клавиши Enter утилита не отправляет команду на сервер, а предлагает продолжить ввод команды. Если вы хотите отменить ввод многострочной команды, наберите \c (рис. 2.3).

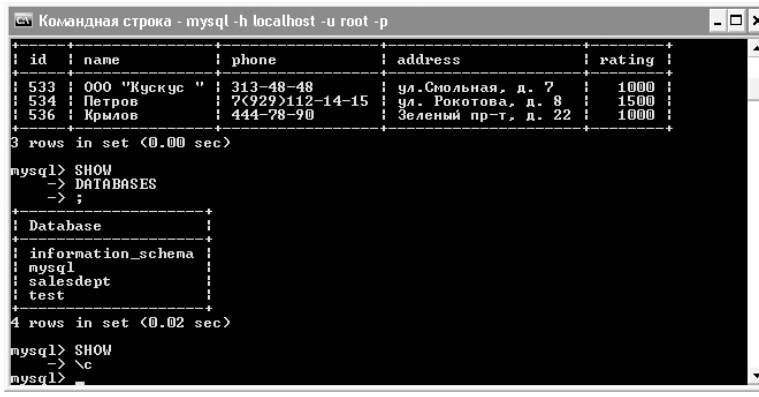

Рис. 2.3. Многострочная команда

Если вы используете MySQL Query Browser, то кодировку устанавливать не нужно – эта программа работает в кодировке UTF-8 и сама сообщает об этом серверу. Однако в MySQL Query Browser существует проблема отображения русских букв в области запросов (области, куда вводится текст SQL-команд, рис. 2.4). Для решения этой проблемы необходимо изменить шрифт, используемый в области запросов (как это сделать, рассказывалось в конце предыдущей главы). Выполнить смену шрифта достаточно один раз.

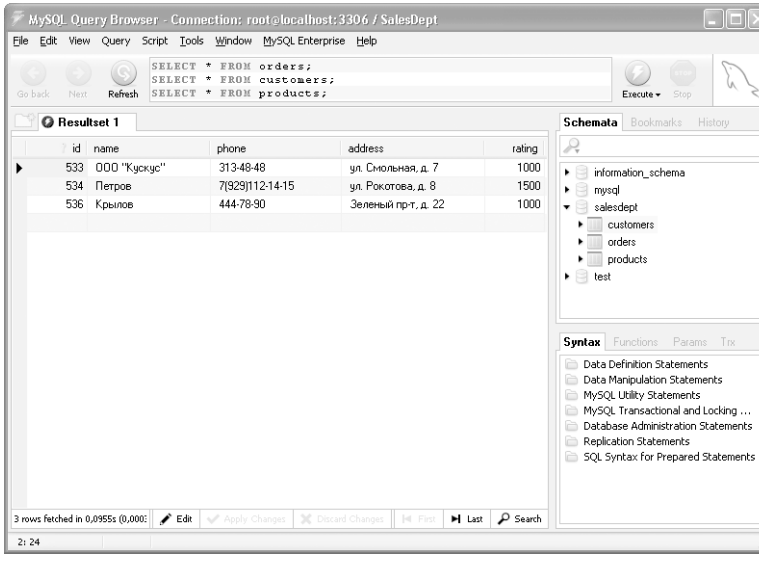

Рис. 2.4. Выполнение SQL-запроса в MySQL Query Browser

В области запросов вы можете ввести сразу несколько SQL-команд, как показано на рис. 2.4. Текущая команда (на одной из ее строк установлен курсор) выделена белым цветом фона, остальные команды отображены на светло-сером фоне. Чтобы выполнить текущую команду, вы можете нажать либо кнопку Execute, расположенную справа от области запросов, либо комбинацию клавиш Ctrl+Enter. После выполнения команды запрошенные данные выводятся в области результатов, а сообщение о результате выполнения ко-

манды – в нижней части этой области.

Теперь, когда вы научились вводить SQL-команды, приступим к управлению данными с помощью этих команд. В первую очередь мы рассмотрим команды, предназначенные для работы с базой данных в целом.

2.2. Создание базы данных

В этом разделе вы узнаете, как создать и удалить базу данных, изменить для нее кодировку по умолчанию, выбрать текущую БД, а также просмотреть список всех баз на данном сервере MySQL.

Чтобы создать базу данных, выполним команду

CREATE DATABASE <Имя базы данных>;

Например, команда

CREATE DATABASE SalesDept;

создает базу данных с именем SalesDept (Отдел продаж).

Если вам по каким-либо причинам нужно для новой базы данных установить кодировку по умолчанию, отличную от кодировки, указанной при настройке MySQL, то при создании базы данных вы можете указать нужную кодировку (character set) и/или правило сравнения (сортировки) символьных значений:

CREATE DATABASE <Имя базы данных>
CHARACTER SET <Имя кодировки>
COLLATE <Имя правила сравнения>;

Например, если вы будете в новую базу импортировать данные, которые находятся в кодировке CP-1251, то укажем эту кодировку при создании базы данных таким образом:

CREATE DATABASE SalesDept
CHARACTER SET cp1251 COLLATE cp1251_general_ci;

Совет

Чтобы просмотреть список используемых в MySQL кодировок, выполним команду SHOW CHARACTER SET; а чтобы увидеть список правил сравнения символьных значений – команду SHOW COLLATION;. При этом можно использовать оператор LIKE: например, чтобы увидеть все правила сравнения для кодировки CP-1251, выполним команду SHOW COLLATION LIKE %1251 %;. Окончание «_ci» (case insensitive) в названии правил сравнения означает, что при сравнении и сортировке регистр символов не учитывается, окончание «_cs» (case sensitive) – регистр учитывается, окончание «_bin» (binary) – сравнение и сортировка выполняются по числовым кодам символов. Для большинства правил сравнения вы можете найти описание (то есть порядок следования символов, в соответствии с которым будут упорядочиваться текстовые значения) на веб-странице http://www.collation-charts.org/mysql60/.

Кодировка, указанная при создании базы данных, будет по умолчанию использоваться для таблиц этой базы, однако вы можете задать и другую кодировку.

Изменить кодировку и/или правило сравнения символьных значений для базы данных вы можете с помощью команды

ALTER DATABASE <Имя базы данных>
CHARACTER SET <Имя кодировки>

COLLATE <Имя правила сравнения>;

При этом кодировка, используемая в уже существующих таблицах базы данных, остается прежней; меняется только кодировка, назначаемая по умолчанию для вновь создаваемых таблиц.

Чтобы удалить ненужную или ошибочно созданную базу данных, выполните команду

DROP DATABASE <Имя базы данных>;

> **Внимание!**
>
> Удаление базы данных – очень ответственная операция, поскольку она приводит к удалению всех таблиц этой базы и данных, хранившихся в таблицах. Перед удалением рекомендуется создать резервную копию базы данных.

Одну из баз, созданных на данном сервере MySQL, вы можете выбрать в качестве текущей базы данных с помощью команды

USE <Имя базы данных>;

Например,:

USE SalesDept;

После этого вы можете выполнять операции с таблицами этой базы данных, не добавляя имя базы в виде префикса к имени таблицы. Например, для обращения к таблице Customers (Клиенты) базы данных SalesDept (Отдел продаж) можно вместо SalesDept.Customers писать просто Customers. Указав текущую базу, вы можете обращаться и к таблицам

других баз данных, однако использование имени базы данных в виде префикса при этом обязательно. Выбор текущей базы сохраняется до момента отсоединения от сервера или до выбора другой текущей базы данных.

Чтобы увидеть список всех баз, существующих на данном сервере MySQL, выполните команду

SHOW DATABASES;

Даже если вы еще не создали ни одной базы данных, в полученном списке вы увидите три системных базы данных.

• INFORMATION_SCHEMA – информационная база данных, из которой вы можете получить сведения о всех остальных базах, о структуре данных в них и о всевозможных объектах: таблицах, столбцах, первичных и внешних ключах, правах доступа, хранимых процедурах, кодировках и др. Эта база данных доступна только для чтения и является виртуальной, то есть она не хранится в виде каталога на диске: вся информация, запрашиваемая из этой БД, предоставляется динамически сервером MySQL.

• mysql – служебная база данных, которую использует сервер MySQL. В ней хранятся сведения о зарегистрированных пользователях и их правах доступа, справочная информация и др.

• test – пустая база данных, которую можно использовать для «пробы пера» или просто удалить.

Итак, вы освоили основные операции, выполняемые с базой данных как единым целым: команды CREATE

DATABASE (создание), ALTER DATABASE (изменение), DROP DATABASE (удаление), USE (выбор текущей базы данных) и SHOW DATABASES (просмотр списка баз данных). Далее мы рассмотрим операции с таблицами. При этом будем считать, что вы выбрали какую-либо базу данных в качестве текущей и работаете с ее таблицами.

2.3. Работа с таблицами

В этом разделе вы узнаете, как создать, изменить и удалить таблицу, как просмотреть информацию о ней и список всех таблиц в текущей базе данных. Начнем с наиболее сложной команды – создания таблицы.

Создание таблицы

Чтобы создать таблицу, выполните команду, представленную в листинге 2.1.

Листинг 2.1. Команда создания таблицы

```
CREATE TABLE <Имя таблицы>
(<Имя столбца 1> <Тип столбца 1> [<Свойства столбца 1>],
<Имя столбца 2> <Тип столбца 2> [<Свойства столбца 2>],
…
[<Информация о ключевых столбцах и индексах>])
[<Опциональные свойства таблицы>];
```

Как вы видите, команда создания таблицы может включать множество параметров, однако многие из них задавать

необязательно (в листинге 2.1 такие параметры заключены в квадратные скобки). В действительности для создания таблицы достаточно указать ее имя, а также имена и типы всех столбцов; остальные параметры используются в случае необходимости.

Рассмотрим вначале несколько примеров, которые помогут вам освоить команду CREATE TABLE и сразу же, не изучая ее многочисленных параметров, начать создавать собственные (простые по структуре) таблицы.

Предположим, что мы строим базу данных, которую спроектировали в главе 1. Используя команды из предыдущего раздела, мы создали пустую базу данных SalesDept (Отдел продаж) и выбрали ее в качестве текущей. Теперь создадим три таблицы: Customers (Клиенты), Products (Товары) и Orders (Заказы). В листинге 2.2 представлена команда создания таблицы Customers.

Листинг 2.2. Команда создания таблицы Customers

```
CREATE TABLE Customers
(id SERIAL,
name VARCHAR(100),
phone VARCHAR(20),
address VARCHAR(150),
rating INT,
```

PRIMARY KEY (id))
ENGINE InnoDB CHARACTER SET utf8;

В этой команде использовались параметры: во-первых, название таблицы и, во-вторых, названия и типы столбцов, из которых будет состоять таблица (см. также табл. 1.1 в главе 1).

• id – идентификатор записи. Этому столбцу вы назначили тип SERIAL, позволяющий автоматически нумеровать строки таблицы. Ключевое слово SERIAL расшифровывается как BIGINT UNSIGNED NOT NULL AUTO_INCREMENT UNIQUE. Это означает, что в столбец можно вводить большие целые (BIGINT) положительные (UNSIGNED) числа, при этом автоматически контролируется отсутствие неопределенных и повторяющихся значений (NOT NULL UNIQUE). Если при добавлении строки в таблицу вы не укажете значение для этого столбца, то программа MySQL внесет в этот столбец очередной порядковый номер (AUTO_INCREMENT).

> **Примечание**
>
> NULL – это константа, указывающая на отсутствие значения. Если в столбце находится значение NULL, то считается, что никакого определенного значения для этого столбца не задано (поэтому мы также называем NULL неопределенным значением). Не следует путать NULL с пустой строкой («») или числом 0. Значения NULL обрабатываются особым образом: большинство

функций и операторов возвращают NULL, если один из аргументов равен NULL. Например, результат сравнения 1 = 1 – истинное значение (TRUE), а результат сравнения NULL = NULL – неопределенное значение (NULL), то есть два неопределенных значения не считаются равными.

• Nam – имя клиента, phone – номер телефона и address – адрес. Вы присвоили этим столбцам тип VARCHAR, поскольку они будут содержать символьные значения. В скобках указывается максимально допустимое количество символов в значении столбца.

• Rating – рейтинг. Тип INT означает, что столбец будет содержать обычные целые числа.

В-третьих, вы указали, что столбец id будет первичным ключом таблицы, включив в команду создания таблицы определение PRIMARY KEY (id).

В-четвертых, вы задали для этой таблицы два опциональных параметра. Параметр ENGINE определяет тип таблицы. Таблице Customers вы присвоили тип InnoDB, так как только этот тип обеспечивает поддержание целостности связей между таблицами (более подробно о типах таблиц будет рассказано в пункте «Опциональные свойства таблицы»). Параметр CHARACTER SET определяет кодировку по умолчанию для данных в таблице. Поскольку вы не задали кодировку отдельно для столбцов name, phone и address, данные в этих столбцах будут храниться в кодировке UTF-8, которая

назначена в качестве кодировки по умолчанию для таблицы Customers.

Следующий пример, который мы рассмотрим, – команда создания таблицы Products (Товары), представленная в листинге 2.3.

Листинг 2.3. Команда создания таблицы Products

```
CREATE TABLE Products
(id SERIAL,
description VARCHAR(100),
details TEXT,
price DECIMAL(8,2),
PRIMARY KEY (id))
ENGINE InnoDB CHARACTER SET utf8;
```

Эта команда очень похожа на команду создания таблицы Customers и отличается от нее только названием таблицы и набором столбцов. Столбцы id (номер товара) и description (наименование товара) таблицы Products имеют уже знакомые нам типы. Столбец details (описание) имеет тип TEXT. Этот тип удобно использовать вместо типа VARCHAR, если столбец будет содержать длинные значения: суммарная длина значений всех столбцов с типом VARCHAR ограничена 65 535 байтами для каждой таблицы, а на общую длину

столбцов с типом TEXT ограничений нет. Недостатком типа TEXT является невозможность включать такие столбцы во внешний ключ таблицы, то есть создавать связь между таблицами на основе этих столбцов.

Столбец price (цена) имеет тип DECIMAL, предназначенный для хранения денежных сумм и других значений, для которых важно избежать ошибок округления. В скобках мы указали два числа: первое из них определяет максимальное количество цифр в значении столбца, второе – максимальное количество цифр после десятичного разделителя. Другими словами, цена товара может содержать до шести цифр в целой части (6 = 8–2) и до двух цифр в дробной части.

И, наконец, последний пример – команда создания таблицы Orders (Заказы), представленная в листинге 2.4.

Листинг 2.4. Команда создания таблицы Orders

```
CREATE TABLE Orders
(id SERIAL,
date DATE,
product_id BIGINT UNSIGNED NOT NULL,
qty INT UNSIGNED,
amount DECIMAL(10,2),
customer_id BIGINT UNSIGNED,
PRIMARY KEY (id),
FOREIGN KEY (product_id) REFERENCES Products (id)
```

ON DELETE RESTRICT ON UPDATE CASCADE,
FOREIGN KEY (customer_id) REFERENCES Customers (id)
ON DELETE RESTRICT ON UPDATE CASCADE)
ENGINE InnoDB CHARACTER SET utf8;

Особенностью таблицы Orders является наличие внешних ключей: столбец product_id (товар) содержит номера товаров из таблицы Products, а столбец customer_id (клиент) – номера клиентов из таблицы Customers (см. также табл. 1.2 в главе 1). Поскольку номера товаров и клиентов являются большими целыми положительными числами, столбцам product_id и customer_id мы назначили тип BIGINT UNSIGNED.

Далее, чтобы обеспечить автоматическое поддержание целостности связей (о целостности мы рассказывали в главе 1), мы сообщили программе MySQL, какому первичному ключу соответствует каждый внешний ключ. Так, конструкция FOREIGN KEY (customer_id) REFERENCES Customers (id) означает, что в столбце customer_id могут содержаться только значения из столбца id таблицы Customers и неопределенные значения (NULL), а остальные значения запрещены. Для столбца product_id мы задали аналогичное ограничение и присвоили этому столбцу свойство NOT NULL, чтобы запретить регистрировать заказы с неопределенным товаром. Дополнительно мы указали для каждой из связей правила поддержания целостности (их мы также рассматривали в главе 1). Правило ON DELETE RESTRICT означает, что

нельзя удалить запись о клиенте, если у этого клиента есть зарегистрированный заказ, и нельзя удалить запись о товаре, если этот товар был кем-то заказан. Правило ON UPDATE CASCADE означает, что при изменении номера клиента в таблице Customers или номера товара в таблице Products соответствующие изменения вносятся и в таблицу Orders.

> **Примечание**
>
> Обратите внимание, что таблицу Orders мы создали в последнюю очередь, так как первичные ключи в таблицах Customers и Products должны быть созданы раньше, чем ссылающиеся на них внешние ключи в таблице Orders. Впрочем, можно было бы создать таблицы без внешних ключей в любой последовательности, а затем добавить внешние ключи с помощью команды ALTER TABLE, которую мы рассмотрим в подразделе «Изменение структуры таблицы».

В наших примерах мы рассмотрели лишь некоторые параметры команды создания таблицы. Теперь мы перечислим все основные параметры, которые могут вам пригодиться при создании таблиц. В пункте «Типы данных в MySQL» речь пойдет о типах столбцов, в пункте «Свойства столбцов» – о настройке ключевых столбцов и, наконец, в пункте «Ключевые столбцы и индексы» – об опциональных свойствах таблицы.

Типы данных в MySQL

Как вы уже знаете, при создании таблицы нужно указать тип данных для каждого столбца. В MySQL предусмотрено множество типов данных для хранения чисел, даты/времени и символьных строк (текстов). Кроме того, существуют типы данных для хранения пространственных (spatial) объектов, которые в этой книге рассматриваться не будут.

Рассмотрим числовые типы данных.

• BIT[(<Количествобитов>)].

Битовое число, содержащее заданное количество битов. Если количество битов не указано, число состоит из одного бита.

• TINYINT.

Целое число в диапазоне либо от -128 до 127, либо (если указано свойство UNSIGNED) от 0 до 255.

• BOOL или BOOLEAN.

Являются синонимами к типу данных TINYINT(1) (число в скобках – это количество отображаемых цифр, см. примечание ниже). При этом ненулевое значение рассматривается как истинное (TRUE), нулевое – как ложное (FALSE).

• SMALLINT.

Целое число в диапазоне либо от -32 768 до 32 767, либо (если указано свойство UNSIGNED) от 0 до 65 535.

• MEDIUMINT.

Целое число в диапазоне либо от -8 388 608 до 8 388 607, либо (если указано свойство UNSIGNED) от 0 до 16 777 215.

• INT или INTEGER.

Целое число в диапазоне либо от -2 147 483 648 до 2 147 483 647, либо (если указано свойство UNSIGNED) от 0 до 4 294 967 295.

• BIGINT.

Целое число в диапазоне либо от -9 223 372 036 854 775 808 до 9 223 372 036 854 775 807, либо (если указано свойство UNSIGNED) от 0 до 18 446 744 073 70 9 551 615.

• SERIAL.

Синоним выражения BIGINT UNSIGNED NOT NULL AUTO_INCREMENT UNIQUE (большое целое число без знака, принимающее автоматически увеличиваемые уникальные значения; значения NULL запрещены). Используется для автоматической генерации уникальных значений в столбце первичного ключа. Описание свойств UNSIGNED и AUTO_INCREMENT вы найдете в этом подразделе, а свойств NOT NULL и UNIQUE – в пункте «Свойства столбцов».

Примечание

Для всех целочисленных типов данных, кроме BOOL (BOOLEAN) и SERIAL, можно в скобках указать количество отображаемых цифр, которое используется совместно с параметром ZEROFILL: если число содержит меньшее количество цифр, то при выводе

оно дополняется слева нулями. Например, если столбец таблицы определен как INT(5) ZEROFILL, то значения «1234567» и «12345» отображаются «как есть», а значение «123» – как «00123». Для типа данных BIT в скобках указывается размер числа, то есть максимальное количество хранимых битов.

• FLOAT.

Число с плавающей точкой в диапазоне от $-3{,}402823466^{38}$ до $-1{,}175494351^{-38}$ и от $1{,}175494351^{-38}$ до $3{,}402823466^{38}$ (а также значение 0) с точностью около 7 значащих цифр (точность зависит от возможностей вашего компьютера).

• DOUBLE, DOUBLE PRECISION или REAL.

Число с плавающей точкой в диапазоне от $-1{,}7976931348623157^{308}$ до $-2{,}2250738585072014^{-308}$ и от $2{,}2250738585072014^{-308}$ до $1{,}7976931348623157^{38}$ (а также значение 0) с точностью около 15 значащих цифр (точность зависит от возможностей вашего компьютера).

• FLOAT(<Точность>).

При значении точности от 0 до 24 этот тип данных эквивалентен типу FLOAT, при значении от 25 до 53 – типу DOUBLE.

• DECIMAL, DEC, NUMERIC или FIXED.

Точное (неокругляемое) число с фиксированной точкой. Может содержать до 65 значащих цифр и до 30 цифр после десятичного разделителя (по умолчанию – 10 значащих цифр и 0 после десятичного разделителя).

Примечание

Для всех десятичных (нецелочисленных) типов данных, кроме FLOAT(<Точность>), можно в скобках указать точность и шкалу, то есть максимальное количество хранимых значащих цифр и максимальное количество хранимых цифр после десятичного разделителя. Например, если для столбца задан тип данных FLOAT(7,5), это означает, что в столбец нельзя добавить значение с более чем двумя (2 = 7–5) цифрами в целой части и все введенные значения будут округляться до 5 знаков после десятичного разделителя. Для чисел с плавающей точкой можно указать точность до 255 и шкалу до 30, однако указывать слишком большую точность и шкалу не имеет смысла, так как в базе данных сохраняются приближенные значения, которые совпадают с реальными лишь в первых 7 (для типа FLOAT) или 15 (для типа DOUBLE) значащих цифрах, последующие цифры при сохранении могут быть искажены. Для чисел с фиксированной точкой можно указать точность до 65 и шкалу до 30. Если точность и шкала не указаны, то они равны, соответственно, 10 и 0. При сохранении чисел с фиксированной точкой искажений не происходит.

Завершая рассмотрение числовых типов данных, обсудим три свойства, которые можно указать для числовых столбцов:

- UNSIGNED – данное свойство означает, что в столбце

запрещены отрицательные (со знаком «-») значения. Указывать это свойство можно для любых столбцов с числовым типом данных, кроме BIT, BOOL (BOOLEAN) и SERIAL. Для целочисленных столбцов при добавлении свойства UNSIGNED максимально допустимое значение столбца увеличивается вдвое.

• ZEROFILL – данное свойство означает, что значения при отображении будут дополнены нулями. Целые числа дополняются нулями слева в соответствии с указанным количеством отображаемых цифр, десятичные – слева и справа в соответствии с указанными точностью и шкалой. Например, если столбец определен как DOUBLE(10,5) ZEROFILL, то значение «12.23» отображается как «0012.23000». Кроме того, данное свойство запрещает отрицательные значения, как и свойство UNSIGNED. Указывать свойство ZEROFILL можно для любых столбцов с числовым типом данных, кроме BIT, BOOL (BOOLEAN) и SERIAL.

• AUTO_INCREMENT – данное свойство обеспечивает автоматическую нумерацию строк таблицы. Это означает, что при добавлении в столбец неопределенного (NULL) или нулевого значения оно автоматически заменяется следующим номером, на единицу больше предыдущего (нумерация по умолчанию начинается с единицы, установить другой начальный номер можно с помощью соответствующего свойства таблицы). Указывать это свойство можно для любых столбцов с числовым типом данных, кроме BIT и DECIMAL

(DEC, NUMERIC, FIXED). В таблице может быть только один столбец с таким свойством, и для него должен быть создан ключ или индекс (об этом вы узнаете в пункте «Ключевые столбцы и индексы»).

Далее рассмотрим типы данных, используемые при хранении даты и времени.

Для столбца, который будет содержать дату и/или время, вы можете использовать один из следующих типов данных.

• DATE.

Дата в формате «YYYY-MM-DD», в диапазоне от «0000-01-01» до «9999-12-31».

• DATETIME.

Дата и время в формате «YYYY-MM-DD HH:MM:SS» в диапазоне от «0000–0101 00:00:00» до «9999-12-31 23:59:59».

• TIMESTAMP.

Отметка времени в формате «YYYY-MM-DD HH:MM:SS» в диапазоне от «1970-01-01 00:00:00» до некоторой даты в 2038 г. При добавлении или изменении строки таблицы в столбце с типом TIMESTAMP автоматически устанавливается дата и время выполнения операции (если значение этого столбца не указано явно или указано неопределенное значение). Если нужно, чтобы отметка времени проставлялась только при добавлении строки, после слова TIMESTAMP добавим свойство DEFAULT CURRENT_TIMESTAMP.

Если в таблице есть несколько столбцов с типом TIMESTAMP, отметка времени автоматически проставляется только в первом из них. Если необходимо также вносить отметку времени в какой-либо из последующих столбцов с типом TIMESTAMP, то при добавлении/изменении строки укажем для этого столбца значение NULL, которое будет автоматически заменено текущей датой.

• TIME.

Время в формате «HH:MM:SS» в диапазоне от «-838:59:59» до «838:59:59».

• YEAR, YEAR(2), YEAR(4).

Год в формате «YYYY» или «YY» (если количество цифр не указано, используется формат «YYYY»). Диапазон значений – от 1901 до 2155, если используется формат «YYYY», или от 70 (соответствует 1970 г.) до 69 (соответствует 2069 г.), если используется формат «YY».

Отмечу, что MySQL воспринимает даты не только в указанном выше формате. Вы можете ввести дату с любым знаком препинания в качестве разделителя, например 2007@12@31 23%59%59, или без разделителя, например 20071231235959. Более того, если в столбец с типом даты или времени вносится символьное или числовое значение в одном из таких форматов, MySQL автоматически преобразует это значение в дату и/или время.

Завершая изучение типов данных, рассмотрим символьные типы.

Столбцам, которые будут содержать текст, можно присвоить один из следующих типов данных.

• CHAR(<Количество символов>) или NATIONAL CHAR(<Количество символов>).

Символьная строка фиксированной длины. В таком столбце всегда хранится указанное количество символов, при необходимости значение дополняется справа пробелами. Вы можете задать количество символов от 0 до 255. Если количество символов не задано, используется длина строки по умолчанию – 1 символ.

Тип данных NATIONAL CHAR отличается от CHAR тем, что для столбцов с типом NATIONAL CHAR используется кодировка UTF-8, в то время как для столбцов с типом CHAR можно указать любую кодировку, поддерживаемую MySQL.

• VARCHAR(<Максимальное количество символов>) или NATIONAL VARCHAR(<Максимальное количество символов>).

Символьная строка переменной длины, содержащая не более указанного количества символов. Вы можете указать максимальное количество символов от 0 до 65 535, но не более 65 535 байтов в сумме для всех столбцов таблицы с типом CHAR, VARCHAR, BINARY или VARBINARY. Таким образом, если во всей таблице вы используете однобайтовую кодировку (где каждому символу соответствует 1 байт, например кодировку KOI8-R, CP-866 или CP-1251), то сум-

марное количество символов, указанное при описании этих столбцов, не должно превышать 65 535. Если же вы используете кодировку UTF-8 (для которой сервер MySQL выделяет до 3 байтов на символ), то суммарное количество символов, указанное при описании этих столбцов, не должно превышать 21 844 (в три раза меньше, чем для однобайтовых кодировок).

Тип данных NATIONAL VARCHAR отличается от VARCHAR тем, что для столбцов с типом NATIONAL VARCHAR используется кодировка UTF-8, в то время как для столбцов с типом VARCHAR можно указать любую кодировку, поддерживаемую MySQL.

• BINARY(<Количество байтов>)

Байтовая (бинарная) строка фиксированной длины. Этот тип аналогичен типу CHAR, только строка содержит не символы, а байты, и значение меньшей длины дополняется справа не пробелами, а нулевыми байтами.

• VARBINARY(<Максимальное количество байтов>)

Байтовая (бинарная) строка переменной длины. Этот тип аналогичен типу VARCHAR, только строка содержит не символы, а байты.

• TINYBLOB

Байтовая (бинарная) строка переменной длины. Максимальная длина – 255 байтов.

• TINYTEXT

Символьная строка переменной длины. Максимальная

длина – 255 байтов (не символов!).

Примечание

Обратите внимание, что для типов данных TINYTEXT, TEXT, MEDIUMTEXT или LONGTEXT длина значения ограничена максимальным количеством байтов, а не символов. Для однобайтовых кодировок (таких как KOI8-R, CP-866 или CP-1251) длина значения в байтах и в символах одинакова. Однако для многобайтовых кодировок реальное количество символов в значении может быть меньше, чем количество байтов. Так, в кодировке UTF-8 для кодирования символов английского алфавита используется 1 байт на символ, для русского алфавита – 2 байта на символ, поэтому максимальное количество символов русского алфавита, которое можно ввести в такой столбец, приблизительно в два раза меньше, чем максимальное допустимое количество байтов для этого столбца.

• BLOB[(<Максимальное количество байтов>)].

Байтовая (бинарная) строка переменной длины. Если количество байтов не указано, то значение столбца ограничено 65 535 байтами. Если количество байтов указано, то создается столбец с типом данных TINYBLOB, BLOB, MEDIUMBLOB или LONGBLOB: выбирается тип данных с наименьшим размером, достаточным для хранения этого количества байтов.

• TEXT[(<Максимальное количество символов>)].

Символьная строка переменной длины. Если количество символов не указано, то значение столбца ограничено 65 535 байтами. Если количество символов указано, то создается столбец с типом данных TINYTEXT, TEXT, MEDIUMTEXT или LONGTEXT: выбирается тип данных с наименьшим размером, достаточным для хранения этого количества символов.

• MEDIUMBLOB.

Байтовая (бинарная) строка переменной длины. Максимальная длина – 16 777 215 байтов.

• MEDIUMTEXT.

Символьная строка переменной длины. Максимальная длина – 16 777 215 байтов.

• LONGBLOB.

Байтовая (бинарная) строка переменной длины. Максимальная длина – не более 4 294 967 295 байтов (4 Гбайт), в зависимости от используемого протокола взаимодействия с сервером MySQL и доступных системных ресурсов.

• LONGTEXT.

Символьная строка переменной длины. Максимальная длина – не более 4 294 967 295 байтов (4 Гбайт), в зависимости от используемого протокола взаимодействия с сервером MySQL и доступных системных ресурсов.

• ENUM('<Значение 1>', '<Значение 2>',…).

Строка, содержащая ровно один элемент из заданного списка. Например, если столбец определен как

ENUM('a','b'), то допустимыми значениями этого столбца являются значения a, b и NULL (а также пустая строка «», которая может появиться при попытке вставки некорректного значения в данный столбец; о добавлении строк в таблицу и о возможных вариантах обработки некорректных значений пойдет речь в подразделе «Вставка отдельных строк»). В список вы можете включить до 65 535 элементов.

• SET('<Значение 1>', '<Значение 2>',…).

Строка, содержащая любой набор элементов из заданного списка (в том числе пустой). Например, если столбец определен как SET('a','b'), то он может содержать значения «» (пустая строка), a, b, a,b и NULL. В список вы можете включить до 64 элементов. Элементы списка не должны содержать запятых. Каждый из элементов может присутствовать в значении столбца только один раз, причем элементы могут следовать только в том порядке, в котором они перечислены в списке. Например, при вставке значений a,b,a,b и b,a они автоматически преобразуются в значение a,b.

В заключение отметим, что в MySQL вы можете указать кодировку отдельно для каждого символьного столбца. А именно, для столбцов с типом CHAR, VARCHAR, TINYTEXT, TEXT, MEDIUMTEXT, LONGTEXT, ENUM и SET вы можете задать свойство CHARACTER SET <Имя кодировки> и/или COLLATE <Имя правила сравнения> (подробнее о кодировках и правилах сравнения символьных значений говорилось в разделе «Создание базы данных»).

Например, чтобы имена клиентов хранились в кодировке CP-1251, тогда как кодировкой по умолчанию для таблицы Customers (Клиенты) является UTF-8, столбец name (имя) можно определить следующим образом:

name VARCHAR(100)
CHARACTER SET cp1251 COLLATE cp1251_general_ci

Если кодировка для столбца не задана, то используется кодировка, заданная для таблицы в целом (об этом вы узнаете из подраздела «Другие команды для работы с таблицами»). Если не задана кодировка и для таблицы, то используется кодировка, установленная для базы данных (см. раздел «Создание базы данных»). Наконец, если и для базы данных не была указана кодировка, то используется кодировка, установленная по умолчанию при настройке MySQL.

Итак, мы рассмотрели типы данных, которые вы можете назначать столбцам таблицы, а также свойства, специфичные для отдельных типов столбцов: свойства UNSIGNED, ZEROFILL и AUTO_INCREMENT для числовых столбцов и свойства CHARACTER SET и COLLATE – для символьных. Перейдем теперь к свойствам, используемым независимо от типа столбцов.

Свойства столбцов

При создании или изменении таблицы вы можете указать следующие свойства столбцов.

• NOT NULL.

Это свойство указывает, что в данном столбце не допускаются неопределенные значения (NULL).

В качестве примера рассмотрим столбец product_id (товар) таблицы Orders (Заказы) (см. листинг 2.4), который мы определили как

product_id BIGINT UNSIGNED NOT NULL

Тем самым мы запретили неопределенные номера товаров, поскольку регистрировать заказ с неизвестным товаром не имеет смысла.

Если для столбца задано свойство NOT NULL, то, в частности, NULL не может использоваться в качестве значения по умолчанию для этого столбца. Значение по умолчанию, отличное от NULL, вы можете задать с помощью свойства DEFAULT <Значение>, которое описано ниже. Если же вы задали для столбца свойство NOT NULL, но не задали значение по умолчанию и не указали значение для этого столбца при вставке строки в таблицу, то поведение программы MySQL зависит от того, в каком режиме вы работаете (об этом будет подробно рассказано в подразделе «Вставка отдельных строк»).

• NULL.

Данное свойство указывает, что в столбце разрешены неопределенные значения (NULL). Задавать это свойство

имеет смысл только для столбцов с типом TIMESTAMP, которые по умолчанию не допускают неопределенных значений. Остальные типы столбцов допускают неопределенные значения, если только для них не задано свойство NOT NULL.

• DEFAULT <Значение>.

Данное свойство определяет значение по умолчанию для столбца, которое используется, если при вставке строки в таблицу значение столбца не задано явно. Значением по умолчанию может быть только константа; исключение составляют столбцы с типом TIMESTAMP, для которых в качестве значения по умолчанию можно задать переменную величину CURRENT_TIMESTAMP (текущую дату и время). Нельзя установить значение по умолчанию для столбцов с типом TINYBLOB, TINYTEXT, BLOB, TEXT, MEDIUMBLOB, MEDIUM-TEXT, LONGBLOB и LONGTEXT, а также для числовых столбцов, для которых задано свойство AUTO_INCREMENT. Кроме того, нельзя использовать неопределенное значение по умолчанию (NULL), если для столбца задано свойство NOT NULL.

Например, чтобы задать для поля phone (телефон) таблицы Customers (Клиенты) значение по умолчанию, равное пустой строке, можно определить это поле следующим образом:

phone VARCHAR(20) DEFAULT ''

• COMMENT 'Текст комментария'.

Произвольное текстовое описание столбца длиной до 255 символов. Например, описание для поля rating (рейтинг) таблицы Customers (Клиенты) можно задать следующим образом:

rating INT COMMENT 'Рейтинг клиента'

Помимо перечисленных свойств, для столбца можно также задать свойства UNIQUE и PRIMARY KEY, однако соответствующие настройки ключевых столбцов и индексов можно указать и после определения всех столбцов таблицы. Мы будем рассматривать только второй вариант создания ключевых столбцов и индексов. Об этом и пойдет речь в следующем пункте.

Ключевые столбцы и индексы

После того как определены все столбцы таблицы, можно перечислить через запятую ключевые столбцы и индексы (см. листинги 2.1–2.4). Вы можете использовать следующие конструкции:

• [CONSTRAINT <Имя ключа>] PRIMARY KEY (<Список столбцов>).

Определяет первичный ключ таблицы (о первичных клю-

чах было рассказано в главе 1). В таблице может быть только один первичный ключ, состоящий из одного или нескольких столбцов. Столбцам, входящим в первичный ключ, автоматически присваивается свойство NOT NULL. Ключевое слово CONSTRAINT и имя ключа можно опустить, так как для первичного ключа заданное имя игнорируется и используется имя PRIMARY.

Если в состав первичного ключа входят столбцы с типом TINYBLOB, TINYTEXT, BLOB, TEXT, MEDIUMBLOB, MEDIUMTEXT, LONGBLOB и LONGTEXT, необходимо указать количество символов в начале значения столбца; при этом первичный ключ содержит не полные значения столбца, а только начальные подстроки значений. Пример определения первичного ключа:

PRIMARY KEY (id)

Именно так мы создали первичный ключ для таблиц Customers (Клиенты), Orders (Заказы) и Products (Товары) (см. листинги 2.2–2.4). Если бы мы решили не использовать дополнительный столбец id в таблице Products, а образовать первичный ключ из столбцов description (название) и details (описание), то в команду создания таблицы Products нужно было бы включить следующее определение:

PRIMARY KEY (description,details(10))

В этом случае в первичный ключ вошли бы столбец description и начальные подстроки значений столбца details длиной 10 символов.

• INDEX [<Имя индекса>] (<Список столбцов>).

Создает индекс для указанных столбцов. Индекс – это вспомогательный объект, позволяющий значительно повысить производительность запросов с условием на значение столбцов, включенных в индекс (подробнее об индексах мы поговорим в главе 6). Например, чтобы создать индекс для быстрого поиска по именам клиентов, в команду создания таблицы Customers (Клиенты) (см. листинг 2.2) можно включить определение

INDEX (name)

Аналогично первичному ключу, при создании индекса для столбцов с типом TINYBLOB, TINYTEXT, BLOB, TEXT, MEDIUMBLOB, MEDIUMTEXT, LONGBLOB и LONGTEXT необходимо указать количество символов в начале значения столбца, по которым будет проведено индексирование.

Имя индекса указывать не обязательно. Если вы не зададите имя индекса, оно сгенерируется автоматически.

Вместо ключевого слова INDEX можно использовать его синоним – слово KEY.

[CONSTRAINT <Имя ограничения>] UNIQUE [<Имя индекса>] (<Список столбцов>)

Создает уникальный индекс для указанных столбцов. Уникальный индекс отличается от обычного наличием дополнительного ограничения: наборы значений в столбцах, включенных в уникальный индекс, должны быть различны. Иными словами, в таблице не должно быть строк, у которых значения во всех этих столбцах совпадают. Исключение составляют неопределенные значения (NULL): индекс может содержать два (и более) одинаковых набора значений, если хотя бы одно из значений в этих наборах – NULL. Например, ограничение

UNIQUE (address.phone)

запрещает добавлять в таблицу Customers (Клиенты) две строки, в которых и адрес, и номер телефона определены и совпадают, но разрешает добавлять строки, в которых адрес совпадает, а номер телефона не определен (то есть столбец phone в обеих строках содержит значение NULL), а также строки, в которых и адрес, и номер телефона не определены.

Для столбцов с типом TINYBLOB, TINYTEXT, BLOB, TEXT, MEDIUMBLOB, MEDIUMTEXT, LONGBLOB и LONGTEXT необходимо указать количество символов в на-

чале значения столбца, по которым будет проведено индексирование.

Имя ограничения и имя индекса указывать не обязательно. Если ни имя ограничения, ни имя индекса не указаны, имя индекса присваивается программой автоматически.

Вместо ключевого слова UNIQUE можно использовать его синонимы – выражения UNIQUE INDEX или UNIQUE KEY.

• FULLTEXT [<Имя индекса>] (<Список столбцов>).

Создает полнотекстовый индекс для указанных столбцов. Полнотекстовый индекс обеспечивает ускоренный поиск по значениям символьных столбцов (типы CHAR, VARCHAR, TINYTEXT, TEXT, MEDIUMTEXT и LONGTEXT) независимо от длины значений. Такой индекс подобен предметному указателю в книге: он представляет собой список всех слов, встречающихся в значениях столбцов, со ссылками на те значения, в которых каждое слово содержится.

Полнотекстовый индекс можно создать только в таблицах с типом MyISAM (см. пункт «Опциональные свойства таблицы»). Для поиска с использованием полнотекстового индекса предназначен оператор MATCH… AGAINST, о котором будет идти речь в главе 3.

Имя индекса указывать не обязательно. Если вы не зададите имя индекса, оно сгенерируется автоматически.

Вместо ключевого слова FULLTEXT можно использовать его синонимы – выражения FULLTEXT INDEX или

FULLTEXT KEY.

• SPATIAL [<Имя индекса>] (<Список столбцов>).

Создает индекс для поиска по пространственным и географическим значениям, которые остаются за рамками нашего рассмотрения.

• [CONSTRAINT <Имя внешнего ключа>].

FOREIGN KEY [<Имя индекса>] (<Список столбцов>)
REFERENCES <Имя родительской таблицы>
(<Список столбцов первичного ключа родительской таблицы>)
[<Правила поддержания целостности связи>]

Определяет внешний ключ таблицы (внешние ключи мы рассматривали в главе 1). Настроив внешний ключ, мы тем самым создадим связь между данной (дочерней) таблицей и родительской таблицей. Внешние ключи поддерживаются только для таблиц с типом InnoDB (причем и дочерняя, и родительская таблица должны иметь тип InnoDB), для остальных типов таблиц выражение FOREIGN KEY игнорируется.

Столбцы, составляющие внешний ключ, должны иметь типы, аналогичные типам столбцов первичного ключа в родительской таблице. Для числовых столбцов должен совпадать размер и знак, для символьных – кодировка и правило сравнения значений. Столбцы с типом TINYBLOB, TINYTEXT, BLOB, TEXT, MEDIUMBLOB,

MEDIUMTEXT, LONGBLOB и LONGTEXT не могут входить во внешний ключ.

Имя внешнего ключа и имя индекса указывать не обязательно. Если вы не зададите эти имена, они будут автоматически сгенерированы. Вы можете также указать, какие именно правила поддержания целостности связи необходимо использовать для операций удаления и для операций изменения строк родительской таблицы (все эти правила мы обсуждали в подразделе «Целостность данных» главы 1). Для операций удаления вы можете указать одно из следующих выражений:

• ON DELETE CASCADE – каскадное удаление строк дочерней таблицы (строка родительской таблицы удаляется вместе со всеми ссылающимися на нее строками дочерней таблицы);

• ON DELETE SET NULL – обнуление значений внешнего ключа в соответствующих строках дочерней таблицы;

• ON DELETE RECTRICT или ON DELETE NO ACTION (в MySQL эти выражения являются синонимами) – запрет удаления строк родительской таблицы при наличии ссылающихся на них строк дочерней таблицы.

Если вы не задали правило поддержания целостности для операций удаления, по умолчанию используется правило ON DELETE RECTRICT.

Для операций изменения строк родительской таблицы вы можете указать одно из следующих выражений:

• ON UPDATE CASCADE – каскадное обновление значений внешнего ключа дочерней таблицы (вместе со значением первичного ключа в строке родительской таблицы изменяется значение внешнего ключа во всех ссылающихся на нее строках дочерней таблицы);

• ON UPDATE SET NULL – обнуление значений внешнего ключа в соответствующих строках дочерней таблицы;

• ON UPDATE RECTRICT или ON UPDATE NO ACTION (в MySQL эти выражения являются синонимами) – запрет изменения значений первичного ключа в строках родительской таблицы при наличии ссылающихся на них строк дочерней таблицы.

Если вы не задали правило поддержания целостности для операций изменения, по умолчанию используется правило ON UPDATE RECTRICT.

Для столбцов внешнего ключа автоматически создается индекс, поэтому проверки значений внешних ключей в ходе контроля целостности связи выполняются быстро.

Конец ознакомительного фрагмента.