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

***CREATE TABLE*** <имя таблицы>

*(<Имя столбца 1> <Тип столбца 1> [<Свойство 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 необхо-

димости.

Предположим, что мы строим базу данных, которую спроектировали в главе 1. Исполь-

зуя команды из предыдущего раздела, мы создали пустую базу данных SalesDept (Отдел

продаж) и выбрали ее в качестве текущей. Теперь создадим три таблицы: Customers (Кли-

енты), 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;

В этой команде использовались параметры: во-первых, название таблицы и, во-вторых,

названия и типы столбцов, из которых будет состоять таблица.

• ***id*** – идентификатор записи. Этому столбцу вы назначили тип SERIAL, позволяю-

щий автоматически нумеровать строки таблицы. Ключевое слово SERIAL расшифровыва-

ется 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 значе-

ний (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).

• ***name*** – 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»).

**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;

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 столб-

цов с типом TEXT ограничений нет. Недостатком типа TEXT является невозможность вклю-

чать 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.

**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 Custmores(id) ON DELETE RESTRICT ON UPDATE CASCADE) ENGINE InnoDB;

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, 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. 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.

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».