После создания таблиц можно приступать к наполнению их данными. В данном раз-

деле вы узнаете о двух операциях, с помощью которых можно добавить строки в таблицу.

Вначале мы рассмотрим загрузку данных из текстового файла, затем – вставку отдельных

строк.

**Загрузка данных из файла**

Если требуется добавить в таблицу большой массив данных, удобно использовать

для этого команду загрузки данных из файла. Загрузка из файла выполняется программой

MySQL значительно быстрее, чем вставка строк с помощью команды INSERT.

Например, чтобы загрузить данные в таблицу Customers (Клиенты), команда создания

которой показана в листинге 2.2, выполните следующие действия.

1. Запустите стандартную программу Windows Блокнот (Пуск → Все программы →

Стандартные → Блокнот).

2. В окне программы Блокнот введите данные, используя для отделения значений друг

от друга клавишу Tab, а для перехода на следующую строку – клавишу Enter (рис. 2.5).

3. Для сохранения файла с данными нажмите комбинацию клавиш Ctrl+S. В стандарт-

ном окне Windows Сохранить как выберите папку, в которую нужно поместить файл (напри-

мер, C: \data). Введите имя файла (например, Customers. txt) и нажмите кнопку Сохранить.

(По дефолту функция загрузки из файла отключена в целях безопасности. Для того чтобы ее включить, надо в файле ***my.ini*** вписать две строчки: ***loose-local-infile=1*** в колонку ***[client]***, и

***local-infile = 1*** в колонку ***[mysqld]***).

4. Для загрузки данных из созданного файла выполните команду

***LOAD DATA LOCAL INFILE*** *'C:/data/Customers.txt'*

***INTO TABLE*** *Customers*

***CHARACTER SET*** *cp1251;*

Обратите внимание, что в пути к файлу необходимо использовать прямую косую черту,

а не обратную.

Однако если вам нужно загрузить в таблицу данные из текстового файла, который был

создан в другом формате (например, выгружен из другой базы данных), могут потребоваться

и другие параметры. Полностью команда LOAD DATA имеет следующий вид:

***LOAD DATA*** *[LOCAL]* ***INFILE*** *'Путь и имя файла'*

*[REPLACE или IGNORE]*

***INTO TABLE*** *<Имя таблицы>*

***CHARACTER SET*** *<Имя кодировки>*

*[*

***FIELDS***

*[****TERMINATED BY*** *<Разделитель значений в строке>]*

*[[****OPTIONALLY****]*

***ENCLOSED BY*** *<Символ, в который заключены значения>]*

*[****ESCAPED BY*** *<Экранирующий символ>]*

*][*

***LINES***

*[****STARTING BY*** *<Префикс строки>]*

***[TERMINATED BY*** *<Разделитель строк>]*

*]*

*[****IGNORE*** *<Количество строк в начале файла> LINES]*

*[(<Список столбцов>)]*

*[****SET*** *<Имя столбца> = <Выражение>,...];*

В этой команде вы можете использовать следующие параметры.

• LOCAL – укажите этот параметр, если файл с данными находится на клиентском

компьютере (то есть на том компьютере, где работает клиентское приложение, в котором вы

и вводите эту команду). Если файл расположен на компьютере, где работает сервер MySQL,

параметр LOCAL указывать не нужно.

• 'Путь и имя файла' – введите полный путь к файлу, например C:/Data/ mytable.txt

(необходимо использовать прямую косую черту вместо обратной, принятой в Windows).

• REPLACE или IGNORE – укажите один из этих параметров, чтобы сообщить про-

грамме MySQL, как нужно обрабатывать загружаемую строку, если в таблице уже есть

строка с таким же значением первичного ключа или уникального индекса. Если указан пара-

метр REPLACE, то существующая в таблице строка заменяется новой. Если указан параметр

IGNORE, новая строка в таблицу не загружается.

• CHARACTER SET <Имя кодировки> – укажите кодировку данных в файле. Этот

параметр используется для корректного преобразования кодировок. Предполагается, что все

данные в файле имеют одну и ту же кодировку.

**Внимание!**

Загрузка данных в кодировке UTF-8 может работать некорректно

вследствие переменного количества байтов на символ в этой кодировке.

Рекомендуем файл с данными в этой кодировке перед загрузкой

преобразовать в какую-либо однобайтовую кодировку. Например, откроем

файл с помощью программы Блокнот, в меню Файл выберем команду

Сохранить как, а затем в стандартном окне Windows Сохранить как в поле

Кодировка выберем из списка значение ANSI и нажмем кнопку Сохранить.

После этого загрузим этот файл, указав в команде LOAD DATA параметр

CHARACTER SET cp1251.

• FIELDS – укажите этот параметр, чтобы сообщить программе MySQL, в каком фор-

мате заданы значения в файле:

– TERMINATED BY <Разделитель значений в строке> – укажем символ, разделяющий

значения в строке файла. Например, если значения разделены запятыми, укажем параметр

TERMINATED BY, если значения разделены символами табуляции – TERMINATED BY \t',

если значения разделены косой чертой – TERMINATED BY /;

– ENCLOSED BY <Символ, в который заключены значения> – укажем символ, кото-

рым обрамляются значения. Например, если все значения заключены в одинарные кавычки,

укажем ENCLOSED BY \, если в одинарные кавычки заключены только символьные зна-

чения, укажем OPTIONALLY ENCLOSED BY \, а если никакие значения не обрамляются

никакими символами, укажем ENCLOSED BY или вообще опустим этот параметр;

– ESCAPED BY <Экранирующий символ> – укажем экранирующий символ (его также

называют escape-символом). Этот символ сообщает программе MySQL, что следующий за

ним символ нужно интерпретировать особым образом. А именно, обычный символ, следу-

ющий после экранирующего, будет рассматриваться как специальный символ, а специаль-

ный символ, наоборот, – как обычный символ.

Чаще всего экранирующим символом служит обратная косая черта, и в этом случае

зададим значение ESCAPED BY \\. Тогда, например, записанное в файле значение «\N» будет

прочитано и загружено в базу данных как NULL. Другой пример: если значения в файле

разделены запятыми, то экранирующий символ помещается перед запятой, которая должна

восприниматься как часть значения, а не как разделитель, то есть последовательность сим-

волов «\,» интерпретируется как символ запятой в значении.

Если параметр FIELDS не указан, программа MySQL считает, что значения в файле

разделяются табуляцией и не заключаются ни в какие кавычки, а в качестве экранирующего

символа используется обратная косая черта.

• LINES – укажите этот параметр, чтобы сообщить программе MySQL, в каком формате

заданы строки в файле:

– STARTING BY <Префикс строки> – укажем последовательность символов в начале

каждой строки, которая должна игнорироваться программой вместе со всеми предшеству-

ющими символами. После префикса должны начинаться значения;

– TERMINATED BY <Разделитель строк> – укажем символ, которым заканчиваются

строки. Например, если строки заканчиваются символом перевода строки, укажем параметр

TERMINATED BY \n', если символом возврата каретки – укажем TERMINATED BY \r', если

сочетанием этих символов – укажем TERMINATED BY \r\n', если нулевым байтом – укажем

TERMINATED BY \0.

Если параметр LINES не указан, программа MySQL считает, что строки в файле не

имеют префикса и заканчиваются символом перевода строки «\n».

• IGNORE <Количество строк в начале файла> LINES – укажите этот параметр, если

первые несколько строк в файле не содержат значений (иными словами, являются заголов-

ком) и при загрузке их нужно пропустить.

• (<Список столбцов>) – перечислите столбцы таблицы, в которые будут загружаться

данные. Это необходимо, если файл содержит данные не для всех столбцов таблицы или

порядок следования значений в файле отличается от порядка столбцов в таблице.

• SET <Имя столбца> = <Выражение> – вы можете записывать в столбцы не только

значения из файла, но и значения, вычисленные с помощью выражений. Для этого создадим

одну или несколько переменных, присвоим им считанные из файла значения и запишем в

столбец значение выражения, использующего эти переменные. Пусть, например, имеется

таблица t1 с числовым столбцом c1 и столбцом с2 с типом TIMESTAMP. В столбец c1 нужно

загрузить значение из файла, если это значение не превосходит 1000, либо NULL, если зна-

чение в файле больше 1000, а в столбец c2 при этом нужно записать текущую дату и время.

Это можно сделать с помощью команды

*LOAD DATA INFILE 'С:/DATA/t1.txt'*

*INTO TABLE t1 (@var1)*

*SET c1 = IF(@var1< = 1000,@var1,NULL), c2 = CURRENT\_TIMESTAMP;*

(о функции IF и о других функциях, используемых в выражениях, рассказывается в

главе 3).

Далее рассмотрим команду INSERT, с помощью которой также можно добавлять

строки в таблицу.

**Вставка отдельных строк**

Для добавления одной или нескольких строк в таблицу можно использовать команду

***INSERT*** *[****INTO****] <Имя таблицы>*

*[(<Список столбцов>)]*

***VALUES***

*(<Список значений 1>),*

*(<Список значений 2>),*

*...*

*(<Список значений N>);*

В команде ***INSERT*** используются следующие основные параметры.

• Имя таблицы, в которую добавляются строки.

• Список имен столбцов, для которых будут заданы значения. Если значения будут

заданы для всех столбцов таблицы, то приводить список столбцов необязательно.

**Примечание**

Если столбец таблицы не включен в список, то в этом столбце

при добавлении строки будет автоматически установлено значение по

умолчанию.

• Значения, которые нужно добавить в таблицу. Значения могут указываться в одном

из следующих форматов:

– набор значений для каждой добавляемой строки заключается в скобки. Набор зна-

чений внутри каждой пары скобок должен соответствовать указанному списку столбцов, а

если список столбцов не указан, то упорядоченному списку всех столбцов, составляющих

таблицу.

Значения внутри набора, а также сами наборы отделяются друг от друга запятыми;

– символьные значения, а также значения даты и времени приводятся в одинарных

кавычках. Для числовых значений кавычки необязательны. Десятичным разделителем для

чисел с дробной частью служит точка. Время и даты вводятся, соответственно, в формате

«YYYY-MM-DD» и «HH:MM:SS»;

– чтобы ввести в столбец неопределенное значение, то необходимо указать вместо зна-

чения ключевое слово NULL *без кавычек* (слово в кавычках рассматривается как обычная

символьная строка);

– вместо значения можно указать ключевое слово DEFAULT *без кавычек,* тогда в стол-

бец будет введено значение по умолчанию (если оно задано для этого столбца).

Например, добавьте в таблицу Products (Товары) сведения о продукции компании с

помощью команды, представленной в листинге 2.5.

**Листинг 2.5. Команда добавления строк в таблицу Products**

***INSERT INTO*** *Products (description, details, price)*

***VALUES***

*('Обогреватель Мосбытприбор ВГД 121R',*

*'Инфракрасный обогреватель. 3 режима нагрева:*

*400 Вт, 800 Вт, 1200 Вт','1145.00'),*

*('Гриль Мосбытприбор СТ-14',*

*'Мощность 1440 Вт. Быстрый нагрев. Термостат.*

*Цветовой индикатор работы','2115.00'),*

*('Кофеварка Мосбытприбор ЕКЛ-1032',*

*'Цвет: черный. Мощность: 450 Вт.*

*Вместительность: 2 чашки','710.00'),*

*('Чайник Мосбытприбор МН',*

*'Цвет: белый. Мощность: 2200 Вт. Объем: 2 л','925.00'),*

*('Утюг Мосбытприбор c паром АБ 200',*

*'Цвет: фиолетовый. Мощность: 1400 вт','518.00');*

Эта команда добавляет значения в столбцы description (наименование), details (описа-

ние) и price (цена) таблицы Products. При этом в столбец id (идентификатор) автоматически

вносятся порядковые номера строк, поскольку этот столбец имеет тип данных SERIAL (см.

листинг 2.3).

Напомню, что каждая строка таблицы Orders ссы-

лается на строку таблицы Customers и строку таблицы Products, и в момент добавления

строки в таблицу Orders соответствующие строки в таблицах Customers и Products должны

уже существовать.

**Листинг 2.6. Команда добавления строк в таблицу Orders**

***INSERT INTO*** *Orders*

***VALUES***

*(1012,'2007-12-12',5,8,'4500',533),*

*(1013,'2007-12-12',2,14,'22000',536),*

*(1014,'2008-01-21',5,12,'5750',533);*

Если вы пытаетесь добавить в таблицу некорректное значение, то результат выполне-

ния команды INSERT зависит от того, в каком режиме ваше клиентское приложение взаи-

модействует с сервером MySQL. Остановимся на этом подробнее.

Узнать, в каком режиме вы в данный момент работаете, можно с помощью команды

***SHOW VARIABLES LIKE*** *'sql\_mode';*

Эта команда показывает значение переменной sql\_mode. Если в значении нет клю-

чевых слов STRICT\_TRANS\_TABLES и STRICT\_ALL\_TABLES, это означает, что сервер

работает в нестрогом режиме.

В нестрогом режиме вставляемое некорректное значение преобразуется в допустимое,

в частности

• некорректная дата заменяется нулевой датой (0000-00-00 00:00:00);

• «лишние» символы в слишком длинном символьном значении отбрасываются (так,

значение abc, вставляемое в столбец с типом CHAR (2), сокращается до ab);

• слишком большое число заменяется максимально возможным значением для данного

типа столбца;

• при внесении в числовой столбец символьного значения все символы, начиная с пер-

вой буквы, отбрасываются, и в таблицу вносится начальная числовая часть значения, а если

первый символ в значении буква, а не цифра, то значение заменяется нулем (даже если

задано отличное от нуля значение по умолчанию);

• если вы установили для столбца свойство NOT NULL, но не задали значение по умол-

чанию, а при вставке строки не указали значение для этого столбца, то в столбец будет доба-

влено следующее значение:

– для числовых столбцов – 0, а если задано свойство AUTO\_INCREMENT, то очеред-

ной порядковый номер;

– для столбцов с типом даты и времени – нулевая дата и/или время (0000–0000

00:00:00), а для первого в таблице столбца с типом TIMESTAMP – текущая дата и время;

– для символьных столбцов – пустая строка, а для столбца с типом ENUM – первый

из элементов списка.

При этом операция добавления завершается успешно и генерируется предупреждение,

которое можно просмотреть, выполнив после команды INSERT команду

***SHOW WARNINGS****;*

**Внимание!**

В любом из режимов попытка добавить повторяющееся значение в

столбец первичного ключа или уникального индекса вызывает ошибку и

прекращение выполнения операции. Также в любом из режимов вызывает

ошибку и отмену операции попытка добавить во внешний ключ таблицы с

типом InnoDB значение, отсутствующее в первичном ключе родительской

таблицы.

Если в значении переменной sql\_mode содержится ключевое слово STRICT\_

TRANS\_TABLES или STRICT\_ALL\_TABLES, значит, сервер работает в строгом режиме.

Для таблиц с поддержкой транзакций (то есть таблиц с типом InnoDB) эти два режима экви-

валентны и действуют одинаково. При попытке вставки некорректного значения операция

INSERT полностью отменяется (то есть в таблицу не загружается ни одна из добавляемых

строк, даже если некорректное значение обнаружено только в одной из них) и выдается сооб-

щение об ошибке.

Для таблиц, не поддерживающих транзакции (то есть таблиц с типом MyISAM и др.),

режимы STRICT\_TRANS\_TABLES или STRICT\_ALL\_TABLES функционируют следую-

щим образом:

• если некорректное значение обнаружено в первой из добавляемых строк, операция

INSERT полностью отменяется и выдается сообщение об ошибке (то есть результат выпол-

нения такой же, как для таблиц InnoDB);

• если установлен режим STRICT\_TRANS\_TABLES, а некорректное значение обнару-

жено в одной из последующих добавляемых строк, то некорректное значение преобразуется

в допустимое и генерируется предупреждение (как в нестрогом режиме);

• если установлен режим STRICT\_ALL\_TABLES, а некорректное значение обнару-

жено в одной из последующих добавляемых строк, то операция частично отменяется:

строки, предшествующие ошибочной строке, добавляются в таблицу, а ошибочная и следу-

ющие за ней – игнорируются. В связи с этим в таблицы, не поддерживающие транзакции,

рекомендуется добавлять по одной строке в каждой команде INSERT.

**Внимание!**

В любом из режимов числа с дробной частью, добавляемые в

целочисленный столбец, округляются до целого, причем такая операция не

вызывает ни ошибок, ни предупреждений.

Если вы задали для столбца таблицы свойство NOT NULL (тем самым запретив

использовать значение NULL в качестве значения по умолчанию), но не задали отличное от

NULL значение по умолчанию (свойство DEFAULT), и если вы удалили значение столбца по

умолчанию с помощью команды ALTER TABLE, то в строгом режиме считается, что у такого

столбца нет значения по умолчанию. Поэтому при добавлении строки в таблицу необходимо

явно задать значение для такого столбца, в противном случае операция INSERT полностью

отменяется и выдается сообщение об ошибке. Исключение составляют столбцы с типом

TIMESTAMP и ENUM, а также числовые столбцы со свойством AUTO\_INCREMENT: для

них в случае отсутствия явно заданного значения используются такие же значения, что и в

нестрогом режиме.

Изменить режим взаимодействия вашего клиентского приложения с сервером вы

можете с помощью команды

***SET SQL\_MODE*** *= '<Режим>';*

Например, команда

***SET SQL\_MODE*** *= '';*

устанавливает нестрогий режим, а команды

***SET SQL\_MODE*** *= 'STRICT\_TRANS\_TABLES';*

и

***SET SQL\_MODE*** *= 'STRICT\_ALL\_TABLES';*

устанавливают соответствующий строгий режим.

Команда ***SET SQL\_MODE*** изменяет режим взаимодействия с сервером *только для*

*текущего соединения* и не влияет на взаимодействие сервера с другими клиентскими при-

ложениями. Новый режим вступает в силу немедленно после выполнения команды и сохра-

няется только до момента отключения от сервера.

Если же вы хотите, чтобы новый режим действовал глобально (то есть для всех кли-

ентских приложений), необходимо включить в команду изменения режима ключевое слово

GLOBAL:

***SET GLOBAL SQL\_MODE*** *= '<Режим>';*

Режим, установленный глобально, применяется для всех вновь подключаемых к

серверу клиентских приложений; ранее подключенные приложения продолжают работать в

прежнем режиме. Чтобы новый режим вступил в силу для вашего приложения, необходимо

отключиться от сервера и затем снова подключиться к нему. Глобальный режим сохраняется

до момента перезапуска сервера MySQL.

**Внимание!**

В отличие от других клиентских приложений, графическая утилита

MySQL Query Browser всегда взаимодействует с сервером в глобальном

режиме. Установить для нее режим, отличный от установленного глобально,

невозможно: команда SET SQL\_MODE, выполненная в MySQL Query

Browser, игнорируется. Таким образом, чтобы изменить режим для MySQL

Query Browser, нужно изменить режим глобально с помощью команды SET

GLOBAL SQL\_MODE. При изменении глобального режима новый режим

применяется к этой утилите сразу же, без переподключения к серверу.