Для получения информации из таблиц базы данных используются *запросы* – SQL-

команды, начинающиеся с ключевого слова SELECT. В этом разделе вы познакомитесь со

структурой запросов.

**Простые запросы**

Знакомство с запросами начнем с наиболее простой команды, которая выводит все дан-

ные, содержащиеся в таблице:

***SELECT*** *\** ***FROM*** *<Имя таблицы>;*

Например, в результате выполнения запроса

***SELECT*** *\** ***FROM*** *Customers;*

вы получите всю информацию о клиентах.

Вместо звездочки можно указать список столбцов таблицы, из которых нужно полу-

чить информацию. Например, чтобы вывести только имя, телефон и рейтинг каждого кли-

ента, выполните запрос

***SELECT*** *name, phone, rating* ***FROM*** *Customers;*

Получать с помощью запроса можно не только значения столбцов, но и значения, вычи-

сленные с помощью выражений.

Например, запрос

***SELECT*** *name, phone, rating/1000* ***FROM*** *CUSTOMERS;*

возвращает результат, аналогичный предыдущему, только значения рейтинга разде-

лены на 1000.

С помощью запросов можно также вычислять значения без обращения к какой-либо

таблице.

Например, запрос

***SELECT*** *2\*2;*

возвращает результат 4

Результат запроса может содержать повторяющиеся строки. Например, клиенты могут

иметь одинаковые рейтинги, поэтому запрос

***SELECT*** *rating* ***FROM*** *Customers;*

выдает результат, в котором есть одинаковые строки.

Чтобы исключить повторения из результата запроса, добавьте в текст запроса ключевое

слово DISTINCT. Например, чтобы просмотреть список рейтингов клиентов, не содержащий

дубликатов, выполните запрос

***SELECT DISTINCT*** *rating* ***FROM*** *Customers;*

Чтобы упорядочить строки, выведенные запросом, по значениям одного из столбцов,

добавьте в текст запроса выражение

***ORDER BY*** *<Имя столбца> [ASC или DESC]*

Ключевое слово ASC означает, что сортировка выполняется по возрастанию, DESC –

по убыванию значений. Если ни то, ни другое слово не указано, выполняется сортировка по

возрастанию. Кроме того, для сортировки можно использовать сразу несколько столбцов,

тогда строки будут отсортированы по значениям первого из столбцов, строки с одинаковым

значением в первом столбце будут отсортированы по значениям второго из столбцов и т. д.

Например, запрос

***SELECT*** *name, phone, rating* ***FROM*** *Customers*

***ORDER BY*** *rating* ***DESC****, name;*

Вместо имен столбцов в выражении ORDER BY можно использовать их порядковые

имена в результате запроса. Например, приведенный выше запрос аналогичен запросу

***SELECT*** *name, phone, rating* ***FROM*** *Customers*

***ORDER BY*** *3* ***DESC****, 1;*

**Условия отбора**

Чтобы выбрать из таблицы строки, удовлетворяющие какому-либо критерию, добавьте

в текст запроса выражение

***WHERE*** *<Условие отбора>*

Например, запрос

***SELECT*** *name, phone, rating* ***FROM*** *Customers* ***WHERE*** *rating = 1000;*

возвращает только те строки, в которых значение рейтинга равно 1000.

В условии отбора можно использовать любые операторы и функции языка SQL

(подробно о них вы узнаете в главе 3), в том числе логические операторы AND и OR для

создания составных условий отбора.

Например, запрос

***SELECT*** *name, phone, rating* ***FROM*** *Customers*

***WHERE*** *name* ***LIKE*** *'ООО%'* ***OR*** *rating>1000*

***ORDER BY*** *rating* ***DESC****;*

выводит информацию о тех клиентах, чье имя начинается с «ООО», а также о тех,

чей рейтинг превосходит 1000, упорядочивая строки в порядке убывания значения рейтинга.

**Объединение таблиц**

Получить информацию из нескольких таблиц вы можете, указав в запросе список

столбцов и список таблиц, из которых нужно получить информацию:

***SELECT*** *<Список столбцов>* ***FROM*** *<Список таблиц>*

***WHERE*** *<Условие отбора>;*

Например, если требуется вывести информацию о всех заказанных товарах за опреде-

ленную дату с указанием имен и адресов заказчиков, выполните команду

***SELECT*** *name, address, product\_id, qty*

***FROM*** *Customers, Orders*

***WHERE*** *Customers.id = customer\_id* ***AND*** *date = '2007-12-12';*

С помощью этого запроса мы получили данные из столбцов name (имя) и address

(адрес) таблицы Customers (Клиенты) и столбцов product\_id (товар) и qty (кол-во) таблицы

Orders (Заказы). Указав условие WHERE Customers. id = customer\_id, мы сообщили про-

грамме MySQL, что для каждого клиента должны выводиться сведения только о заказах

этого клиента. Иначе мы получили бы бессмысленный набор всевозможных комбинаций

данных из таблицы Customers с данными из таблицы Orders. Обратите внимание, что стол-

бец с именем id есть и в таблице Customers, и в таблице Orders, поэтому мы добавили имя

таблицы Customers в виде префикса к имени столбца.

По такому же принципу можно объединять в запросе и более двух таблиц, и даже

таблицу с самой собой. Объединение таблицы с собой можно представить себе как объеди-

нение нескольких идентичных таблиц. Чтобы различать эти таблицы, им присваиваются

разные псевдонимы. В качестве примера объединения таблицы с самой собой рассмотрим

запрос, который выводит всевозможные пары клиентов с одинаковым рейтингом:

***SELECT*** *L. name,R.name* ***FROM*** *Customers L, Customers R*

***WHERE*** *L. rating = R. rating;*

Создавая этот запрос, мы присвоили «первому экземпляру» таблицы Customers псевдо-

ним L, «второму экземпляру» – псевдоним R. В результате объединения «таблиц» мы полу-

чили всевозможные пары клиентов: первый клиент в каждой паре – это строка из «таблицы»

L, второй – строка из «таблицы» R. С помощью условия WHERE L.rating = R.rating мы

выбрали те пары, в которых рейтинг клиента из таблицы L (L.rating) равен рейтингу клиента

из таблицы R (R.rating). Как и в предыдущем примере, к именам столбцов мы добавили в

виде префикса имена «таблиц» (в данном случае – псевдонимы), чтобы указать, к какому из

экземпляров таблицы относится каждый из столбцов.

Таким образом, запрос выводит следующие пары имен (табл. 2.12).

Поскольку наборы строк в «таблицах» L и R одинаковые, в результате запроса появи-

лось много лишних данных: пары одинаковых имен (они возникли при сравнении строки

«таблицы» L с точной копией этой строки в «таблице» R), а также одна и та же пара имен сна-

чала в прямом, затем в обратном порядке. Чтобы избавиться от повторений, введите допол-

нительное условие отбора.

***SELECT*** *L. name,R.name* ***FROM*** *Customers L, Customers R*

***WHERE*** *L. rating = R. rating* ***AND*** *L. name<R.name;*

Поскольку в действительности одинаковый рейтинг имеют только клиенты Крылов и

ООО «Кускус», результатом этого запроса является единственная строка.

Запросы, объединяющие таблицу с самой собой, можно использовать, в частности, для

поиска ошибок дублирования данных в таблице, например для поиска клиентов с разными

идентификаторами, но одинаковыми именами, адресами и телефонами.

**Вложенные запросы**

Результатом запроса является массив данных в виде таблицы, поэтому вы можете

использовать результат одного запроса в другом запросе. Во многих случаях вложенными

запросами можно заменить объединение таблиц. Например, получить список имен клиен-

тов, когда-либо заказывавших товар № 5, можно с помощью вложенного запроса:

***SELECT*** *name* ***FROM*** *Customers*

***WHERE*** *id* ***IN***

*(****SELECT******DISTINCT*** *customer\_id* ***FROM*** *Orders*

***WHERE*** *product\_id = 5);*

Здесь вложенный запрос получают из таблицы Orders (Заказы) номера клиентов, зака-

завших товар № 5. Для обработки результатов подзапроса мы применили оператор IN, кото-

рый возвращает истинное значение (TRUE), если элемент слева от оператора совпадает с

одним из элементов списка справа от оператора. В данном случае оператор IN проверяет,

содержится ли номер клиента (значение столбца id) в списке номеров, выданных подзапро-

сом. Таким образом, внешний запрос выводит имена тех клиентов, номера которых полу-

чены в результате подзапроса.

Такой же результат можно получить и с использованием объединения таблиц:

***SELECT DISTINCT*** *name* ***FROM*** *Customers, Orders*

***WHERE*** *Customers.id = customer\_id* ***AND*** *product\_id = 5;*

Однако не всегда вложенные запросы и объединения таблиц взаимозаменяемы. В част-

ности, запросы с объединениями могут выводить данные из всех участвующих в запросе

таблиц, а запросы с вложенными запросами, – только из таблиц, участвующих во внешнем

запросе. А с помощью запросов, использующих групповые (агрегатные) функции в подза-

просах, можно получить результат, не достижимый другими способами. Например, вывести

заказ с наибольшей суммой можно только с помощью вложенного запроса, подсчитываю-

щего максимальную сумму заказа:

***SELECT*** *\** ***FROM*** *Orders*

***WHERE*** *amount = (****SELECT MAX****(amount)* ***FROM*** *Orders);*

Во вложенном запросе групповая функция MAX возвращает наибольшее из значений

столбца amount (сумма) таблицы Orders (Заказы) – в данном случае 22 000. Внешний запрос,

в свою очередь, выводит те строки таблицы Orders, в которых значение столбца amount равно

значению, выданному подзапросом, то есть 22 000.

**Объединение результатов запросов**

Чтобы объединить несколько запросов в одну SQL-команду и, соответственно, объ-

единить результаты запросов, используется ключевое слово UNION. Запросы, объединя-

емые с помощью UNION, должны выводить одинаковое количество столбцов, и типы

данных столбцов должны быть совместимы. При объединении результатов автоматически

удаляются повторяющиеся строки; чтобы запретить удаление повторяющихся строк, вместо

слова UNION нужно использовать выражение UNION ALL. Наконец, строки объединенного

запроса можно упорядочить с помощью выражения ORDER BY. В качестве примера рассмо-

трим запрос, выводящий информацию о заказах с наибольшей и наименьшей суммой заказа:

***SELECT*** *\** ***FROM*** *Orders*

***WHERE*** *amount = (****SELECT******MAX****(amount)* ***FROM*** *Orders)*

***UNION***

***SELECT*** *\** ***FROM*** *Orders*

***WHERE*** *amount = (****SELECT MIN****(amount)* ***FROM*** *Orders)*

***ORDER BY*** *1;*

Первый запрос возвращает строку таблицы Orders, в которой значение поля amount

максимально (это строка с id = 1013), второй – строку, в которой значение поля amount мини-

мально (это строка с id = 1012), и при упорядочении по значению столбца id строки меня-

ются местами.

**Выгрузка данных в файл**

(по умолчанию стоит путь C:\ProgramData\MySQL\MySQL Server 8.0\Uploads, его можно сменить в файле my.ini, поле #secure-file-priv=)

Чтобы результат запроса был сохранен в файл, добавьте в команду SELECT выражение

***INTO OUTFILE*** *'Путь и имя файла' [FILEDS ...] [LINES ...]*

При необходимости вы можете также задать параметры FILEDS и LINES, которые

имеют тот же смысл, что и параметры FILEDS и LINES команды LOAD DATA (см. подраз-

дел «Загрузка данных из файла»). Если впоследствии файл будет загружаться в базу данных

MySQL с помощью команды LOAD DATA, то в команде LOAD DATA нужно будет указать

те же самые значения параметров FILEDS и LINES, которые использовались при выгрузке.

Команды SELECT… INTO OUTFILE и LOAD DATA можно использовать для резерв-

ного копирования таблиц или для переноса данных на другой сервер MySQL. Например,

данные из таблицы Customers (Клиенты), сохраненные в файл с помощью команды

***SELECT*** *\* from Customers* ***INTO OUTFILE*** *'C:/data/Customers.txt';*

можно загрузить в таблицу Customers\_copy (имеющую такую же структуру, что и

таблица Customers) с помощью команды

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

***INTO TABLE*** *Customers\_copy;*