Лекция 3. Азбука запросов

Блоки запроса

Выражение *select* могут образовывать несколько компонентов, или ***блоков***. Хотя при работе с MySQL обязательным является только один из них (блок *select*), обычно в запрос включаются по крайней мере два-три из шести доступных блоков. В табл. 1 показаны разные блоки и их назначение.

Таблица 1 – Блоки запроса

Блок	**Назначение**
Select	Определяет столбцы, которые должны быть включены в результирующий набор запроса
From	Указывает таблицы, из которых должны быть извлечены данные, и то, как эти таблицы должны быть соединены
Where	Ограничивает число строк в окончательном результирующем наборе
Group by	Используется для группировки строк по одинаковым значениям столбцов
Having	Ограничивает число строк в окончательном результирующем наборе с помощью группировки данных
Order by	Сортирует строки окончательного результирующего набора по одному или более столбцам

Все показанные в таблице 1 блоки включены в спецификацию ANSI.

Блок *select*

Даже несмотря на то, что блок *select* является первым в выражении *select*, сервер БД обрабатывает его одним из последних. Причина в том, что, прежде чем можно будет определить, что включать в окончательный результирующий набор, необходимо знать все столбцы, которые *могли бы* быть включены в этот набор. Поэтому, чтобы полностью понять роль блока *select*, надо немного разобраться с блоком *from*. Вот запрос для начала:

```
SELECT *
FROM department;
```

```
+---------+----------------+
| dept_id | name           |
+---------+----------------+
|       1 | Operations     |
|       2 | Loans          |
|       3 | Administration |
+---------+----------------+
```

В данном запросе в блоке *from* указана всего одна таблица (**department**), и блок *select* показывает, что в результирующий набор должны быть включены *все* столбцы (обозначено символом «*») таблицы **department**.

Выбрать все столбцы можно не только с помощью символа звездочки, но и явно указав имена интересующих столбцов:

```
SELECT dept_id, name
FROM department;
```

Результаты аналогичны первому запросу, поскольку в блоке *select* указаны все столбцы таблицы **department** (*dept_id* и *name*). Можно также выбрать только некоторые из столбцов таблицы **department**, например:

```
SELECT name
FROM department;
```

```
+----------------+
| name           |
+----------------+
| Operations     |
| Loans          |
| Administration |
+----------------+
```

Таким образом, задача блока *select* заключается в следующем: *Блок **select** определяет, какие из всех возможных столбцов должны быть включены в результирующий набор запроса.*

Псевдонимы столбцов

Хотя инструмент *mysql* и генерирует имена для столбцов, возвращаемых в результате запроса, вы можете задавать эти имена самостоятельно. Кроме того, что при желании можно дать другое имя столбцу из таблицы (если у него «плохое» или неоднозначное имя), практически наверняка вы захотите по-своему назвать те столбцы результирующего набора, которые будут сформированы в результате выполнения выражения или встроенной функции. Сделать это можно добавлением ***псевдонима столбца*** после каждого элемента блока *select*.

```
SELECT emp_id,
  'ACTIVE' status,
  emp_id * 3.14159 empid_x_pi,
  UPPER(lname) last_name_upper
FROM employee;
```

emp_id	status	empid_x_pi	last_name_upper
1	ACTIVE	3.14159	SMITH
2	ACTIVE	6.28318	BARKER
3	ACTIVE	9.42477	TYLER
4	ACTIVE	12.56636	HAWTHORNE
5	ACTIVE	15.70795	GOODING

Добавить псевдонимы для столбцов также можно с использованием ключевого слова **as** перед названием псевдонима столбца:

```
SELECT emp_id,
  'ACTIVE' AS status,
  emp_id * 3.14159 AS empid_x_pi,
  UPPER(lname) AS last_name_upper
FROM employee;
```

Удаление дупликатов

В некоторых случаях запрос может возвратить **дублирующие** строки данных. Например, при выборе ID всех клиентов, имеющих счета, было бы представлено следующее:

```
SELECT cust_id
FROM account;
```

cust_id
1
1
1
2
2
3
3

Поскольку некоторые клиенты имеют несколько счетов, один и тот же ID клиента будет выведен столько раз, сколько счетов имеет клиент. Однако, очевидно, целью данного запроса является *выбор* клиентов, имеющих счета, а не получение ID клиента для каждой строки таблицы **account**. Добиться этого можно, поместив ключевое слово ***distinct*** непосредственно после ключевого слова *select*:

```
SELECT DISTINCT cust_id
FROM account;
```

cust_id
1
2
3

*Блок **from** определяет таблицы, используемые запросом, а также средства связывания таблиц.*

Типы таблиц:

1) Постоянные таблицы (созданные с помощью выражения *create table*)
2) Временные таблицы (строки, возвращаемые подзапросом)
3) Виртуальные таблицы (представления) (созданные с помощью выражения *create view*).

Каждый из этих типов таблиц может быть включен в блок *from* запроса.

Таблицы, формируемые подзапросом

Подзапрос (*subquery*) – это запрос, содержащийся в другом запросе. Подзапросы заключаются в круглые скобки и могут располагаться в различных частях выражения *select*. Однако в рамках блока *from* подзапрос выполняет функцию формирования временной таблицы, видимой для всех остальных блоков запроса и способной взаимодействовать с другими таблицами, указанными в блоке *from*. Например,

```
SELECT e.emp_id, e.fname, e.lname
FROM (SELECT emp_id, fname, lname, start_date, title
      FROM employee) e;
```

emp_id	fname	lname
1	Michael	Smith
2	Susan	Barker
3	Robert	Tyler

Здесь подзапрос к таблице **employee** возвращает пять столбцов, а *основной запрос* ссылается на три из пяти доступных столбцов. Запрос ссылается на подзапрос посредством псевдонима, в данном случае ***e***.

Представления

Представление (*view*) – это запрос, хранящийся в словаре данных. Оно выглядит и работает как таблица, но с представлением не ассоциированы

никакие данные (вот почему его называют *виртуальной таблицей*). При выполнении запроса к представлению запрос соединяется с описанием представления и создается окончательный запрос, который и будет выполнен.

Рассмотрим представление, запрашивающее таблицу **employee** и включающее вызов встроенной функции:

```
CREATE VIEW employee_vw AS
SELECT emp_id, fname, lname,
   YEAR(start_date) start_year
FROM employee;
```

После создания представления никакие дополнительные данные не создаются: сервер просто сохраняет выражение *select* для дальнейшего использования. Теперь, когда представление существует, можно делать запросы к нему:

```
SELECT emp_id, start_year
FROM employee_vw;
```

Представления создаются по следующим причинам:

- скрыть столбцы от пользователей;
- упростить сложно устроенные БД.

Связи таблиц

Если в блоке *from* присутствует более одной таблицы, обязательно должны быть включены и условия, используемые для ***связывания*** таблиц. Это не требование определенного сервера БД, а утвержденный ANSI метод соединения нескольких таблиц, которые является наиболее переносимым между серверами БД.

```
SELECT employee.emp_id, employee.fname,
  employee.lname, department.name dept_name
FROM employee INNER JOIN department
  ON employee.dept_id = department.dept_id;
```

emp_id	fname	lname	dept_name
1	Michael	Smith	Administration
2	Susan	Barker	Administration
3	Robert	Tyler	Administration
4	Susan	Hawthorne	Operations

Определение псевдонимов таблиц

При соединении нескольких таблиц в одном запросе вам понадобится идентифицировать таблицу, на которую делается ссылка при указании столбцов в блоках *select*, *where*, *group by*, *having* и *order by*. Дать ссылку на таблицу вне блока *from* можно одним из двух способов:

- Использовать полное имя таблицы, например *employee.emp_id*.
- Присвоить каждой таблице псевдоним и использовать его в запросе.

Рассмотрим предыдущий запрос с применением ***псевдонимов таблиц***:

```
SELECT e.emp_id, e.fname, e.lname, d.name dept_name
FROM employee e INNER JOIN department d
  ON e.dept_id = d.dept_id;
```

Если внимательнее посмотреть на блок *from*, видно, что таблица **employee** получила псевдоним ***e***, а таблица **department** – псевдоним ***d***. Затем эти псевдонимы используются в блоке *on* при описании условия соединения, а также в блоке *select* при задании столбцов, которые должны быть включены в результирующий набор.

Блок *where*

До сих пор все приводимые запросы осуществляли выбор всех строк из имеющихся таблиц. Однако чаще всего извлекать *все* строки таблицы не требуется, и нужен способ, позволяющий отфильтровывать строки, не представляющие интереса. Это работа для блока *where*.

*Блок **where** – это механизм отсеивания нежелательных строк из результирующего набора.*

Например, требуется извлечь из таблицы **employee** данные, но только для сотрудников, нанятых в качестве *старших операционистов* (head tellers). Это делается следующим образом:

```
SELECT emp_id, fname, lname, start_date, title
FROM employee
WHERE title = 'Head Teller';
```

Этот блок *where* содержит всего одно *условие фильтрации*, но этих условий может быть столько, сколько потребуется. Условия разделяются с помощью таких операторов, как ***and***, ***or*** и ***not***.

Блок *order by*

В общем случае строки результирующего набора запроса возвращаются в произвольном порядке. Если требуется упорядочить результирующий набор определенным образом, необходимо предписать серверу сортировать результаты с помощью блока *order by*.

*Блок **order by** – это механизм сортировки результирующего набора на основе данных столбцов, или выражений, использующих данные столбцов.*

Если требуется проанализировать данные каждого сотрудника, полезно было бы отсортировать результаты по столбцу *open_emp_id*. Для этого просто добавляем этот столбец в блок *order by*:

```
SELECT open_emp_id, product_cd
FROM account
ORDER BY open_emp_id;
```

Сортировку данных также можно осуществлять по нескольким столбцам. Например,

```
ORDER BY open_emp_id, product_cd;
```

При сортировке можно задать порядок *по возрастанию* (*ascending*) или *по убыванию* (*descending*) с помощью ключевых слов ***asc*** и ***desc***. По умолчанию выполняется сортировка по возрастанию, поэтому добавлять придется только ключевое слово ***desc*** – если требуется сортировка по убыванию.

```
ORDER BY open_emp_id DESC;
```

При сортировке с использованием столбцов, перечисленных в блоке *select*, можно ссылаться на столбцы не по имени, а по их *порядковому номеру*.

```
SELECT emp_id, title, start_date, fname, lname
FROM employee
ORDER BY 2, 5;
```

Обновление данных

При первичном вводе информации в таблицу о Вильяме Тернере в выражение ***insert*** не были включены данные для различных столбцов адреса. Следующее выражение показывает, как заполнить эти столбцы с помощью выражения ***update***:

```
UPDATE person
SET street = '1225 Tremont St.',
   city = 'Boston',
   country = 'USA',
   postal_code = '02138'
WHERE person_id = 1;
```

Как видите, одним выражением *update* можно изменять несколько столбцов. Одним выражением также можно изменять несколько строк в зависимости от условий блока *where*:

```
WHERE person_id < 10
```

Если опустить блок *where* совсем, выражение *update* обновит все строки таблицы.

Удаление данных

Предположим, что в базе данных содержатся сведения о 2 людях. Похоже, Вильям и Сьюзен не вполне ладят друг с другом, поэтому один из них должен уйти. Поскольку Вильям был первым, Сьюзен будет вежливо «выставлена» выражением ***delete***:

```
DELETE FROM person
WHERE person_id = 2;
```

Опять же для выделения интересующей строки используется первичный ключ, поэтому из таблицы удаляется всего одна строка. Как и в случае выражения *update*, можно удалить и несколько строк. Все зависит от условий, заданных в блоке *where*. Если блок *where* опущен, будут удалены все строки.

Литература:

1. Алан Бьюли. Изучаем SQL: пер. с англ. – СПб-М.: Символ, O'Reilly, 2007. – 310 с.