Лекция 5. Запросы к нескольким таблицам

Данная лекция посвящена наиболее распространенному типу соединений нескольких таблиц – *внутреннему соединению* (*inner join*).

Что такое соединение?

Запросы к одной таблице, конечно, не редкость, но большинство запросов обращены к двум, трем или даже более таблицам. Для иллюстрации давайте рассмотрим описания таблиц **employee** и **department** и затем определим запрос, извлекающий данные из обеих:

```
mysql> DESC employee;
+--------------------+----------------------+------+-----+---------+
| Field              | Type                 | Null | Key | Default |
+--------------------+----------------------+------+-----+---------+
| emp_id             | smallint(5) unsigned | NO   | PRI | NULL    |
| fname              | varchar(20)          | NO   |     | NULL    |
| lname              | varchar(20)          | NO   |     | NULL    |
| start_date         | date                 | NO   |     | NULL    |
| end_date           | date                 | YES  |     | NULL    |
| superior_emp_id    | smallint(5) unsigned | YES  | MUL | NULL    |
| dept_id            | smallint(5) unsigned | YES  | MUL | NULL    |
| title              | varchar(20)          | YES  |     | NULL    |
| assigned_branch_id | smallint(5) unsigned | YES  | MUL | NULL    |
+--------------------+----------------------+------+-----+---------+
9 rows in set (0.11 sec)

 mysql> DESC department;
+---------+----------------------+------+-----+---------+
| Field   | Type                 | Null | Key | Default |
+---------+----------------------+------+-----+---------+
| dept_id | smallint(5) unsigned | No   | PRI | NULL    |
| name    | varchar(20)          | No   |     | NULL    |
+---------+----------------------+------+-----+---------+
2 rows in set (0.03 sec)
```

Предположим, что нам требуется выбрать имя и фамилию каждого сотрудника, а также название отдела, в котором он работает. Тогда запрос должен будет извлекать столбцы *employee.fname*, *employee.lname* и *department.name*. Но как можно получить данные двух таблиц одним запросом?

Ответ кроется в столбце *employee.dept_id*, в котором хранится ID отдела каждого сотрудника (столбец *employee.dept_id* является *внешним ключом*, ссылающимся на таблицу **department**). Запрос указывает серверу использовать столбец *employee.dept_id* как *мост* между таблицами **employee** и **department**, обеспечивая таким образом возможность включения столбцов

обеих таблиц в результирующий набор запроса. Такой тип операции называется ***соединением***.

Декартово произведение

Начнем с самого простого: поместим таблицы **employee** и **department** в блок *from* запроса и посмотрим, что произойдет.

```
SELECT e.fname, e.lname, d.name
FROM employee e JOIN department d;
```

```
+----------+-----------+----------------+
| fname    | lname     | name           |
+----------+-----------+----------------+
| Michael  | Smith     | Operations     |
| Michael  | Smith     | Loans          |
| Michael  | Smith     | Administration |
| Susan    | Barker    | Operations     |
| Susan    | Barker    | Loans          |
| Susan    | Barker    | Administration |
| Robert   | Tyler     | Operations     |
| Robert   | Tyler     | Loans          |
| Robert   | Tyler     | Administration |
…
```

54 строки

У нас имеется 18 сотрудников и 3 разных отдела. Но как же получилось, что в результирующем наборе оказалось 54 строки?

Поскольку запрос не определил, *как* должны быть соединены эти две таблицы, сервер БД сгенерировал ***Декартово произведение***, т. е. *все возможные* перестановки двух таблиц (18 сотрудников × 3 отдела = 54 перестановки). Такой тип соединения называют ***перекрестным соединением*** (***cross join***).

Внутренние соединения

Чтобы изменить предыдущий запрос и получить результирующий набор, включающий только 18 строк (по одной для каждого сотрудника), понадобится описать взаимосвязь двух таблиц. Как упоминалось ранее, связью между двумя таблицами служит столбец *employee.dept_id*, осталось только добавить эту информацию в подблок *on* блока *from*:

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

fname	lname	name
Michael	Smith	Administration
Susan	Barker	Administration
Robert	Tyler	Administration
Susan	Hawthorne	Operations
John	Gooding	Loans
Helen	Fleming	Operations

…

18 строк

Теперь благодаря добавлению подблока *on*, предписывающего сервер соединять таблицы **employee** и **department**, прослеживается связь от одно таблицы к другой по столбцу *dept_id*.

Если определенное значение столбца *dept_id* присутствует в одной таблице, но его *нет* в другой, соединение строк не происходит, и они не включаются в результирующий набор. Такой тип соединения называют ***внутренним соединением*** (***inner join***), и он является наиболее широко используемым.

Если требуется включить все строки той или иной таблицы независимо от наличия соответствия, можно воспользоваться ***внешним соединением*** (***outer join***).

В предыдущем примере в блоке *from* не был указан тип используемого соединения. Однако если требуется соединить две таблицы путем внутреннего соединения, это следует явно указать в блоке *from*:

```
FROM employee e INNER JOIN department d
```

Замечание: Если тип соединения не задан, сервер по умолчанию проведет внутреннее соединение.

Если имена столбцов, используемых для соединения двух таблиц, совпадают, можно вместо подблока *on* применить подблок ***using***:

```
SELECT e.fname, e.lname, d.name
FROM employee e INNER JOIN department d
   USING (dept_id);
```

Нотация, используемая здесь для соединения таблиц, была введена в версии SQL92 стандарта **ANSI SQL**. Во всех основных СУБД (Oracle Database, Microsoft SQL Server, MySQL, IBM DB2 Universal Database, Sybase Adaptive Server) принят синтаксис соединения SQL92.

Соединение трех и более таблиц

Соединение **трех** таблиц аналогично соединению двух. При соединении **двух** таблиц имеются *две* таблицы и *один* тип соединения в блоке ***from***, а также единственный подблок ***on***, определяющий, как соединяются таблицы. При соединении **трех** таблиц присутствуют *три* таблицы и *два* типа соединения в блоке ***from***, а также *два* подблока ***on***. Например,

```
SELECT a.account_id, c.fed_id, e.fname, e.lname
FROM account a INNER JOIN customer c
   ON a.cust_id = c.cust_id
   INNER JOIN employee e
   ON a.open_emp_id = e.emp_id
WHERE c.cust_type_cd = 'B';
```

Запрос, использующий три или более таблиц, можно представить как снежный ком, катящийся с горы. Первые две таблицы запускают этот ком, а каждая последующая таблица вносит в него свою лепту. Снежный ком можно рассматривать как ***промежуточный результирующий набор***, который подбирает все больше и больше столбцов по мере соединения с последующими таблицами. Поэтому таблица **employee** в действительности была соединена не с таблицей **account**, а с промежуточным результирующим набором, созданным при соединении таблиц **customer** и **account**.

Применение подзапросов в качестве таблиц

Выше мы изучили пример запроса с тремя таблицами, но хочется особо отметить одну из разновидностей таких запросов. Что делать, если некоторые таблицы формируются подзапросами? Рассмотрим запрос, который выбирает все счета, открытые опытными операционистами, в настоящее время работающими в отделении Woburn:

```
SELECT a.account_id, a.cust_id, a.open_date, a.product_cd
FROM account a INNER JOIN
   (SELECT emp_id, assigned_branch_id
    FROM employee
    WHERE start_date <= '2003-01-01'
       AND (title = 'Teller' OR title = 'Head Teller')) e
   ON a.open_emp_id = e.emp_id
   INNER JOIN
    (SELECT branch_id
     FROM branch
```

```
        WHERE name = 'Woburn Branch') b
    ON e.assigned_branch_id = b.branch_id;
```

- Первый подзапрос, имеющий псевдоним ***e***, находит всех опытных операционистов.
- Второй подзапрос, имеющий псевдоним ***b***, выбирает ID отделения Woburn.

Сначала таблица **account** соединяется с подзапросом по опытным операционистам с помощью ID сотрудников, а затем результирующая таблица соединяется с подзапросом по отделению *Woburn* на основе ID филиала.

Повторное использование таблицы

При соединении нескольких таблиц может обнаружиться, что требуется неоднократно соединять одну и ту же таблицу. В нашем примере БД внешние ключи к таблице **branch** присутствуют как в таблице **account** (отделение, в котором был открыт счет), так и в таблице **employee** (отделение, в котором работает сотрудник).

Если в результирующий набор должны войти *оба* отделения, таблицу **branch** можно включить в блок *from* дважды, в первый раз соединив ее с таблицей **employee**, а второй раз – с таблицей **account**. Это сработает, если каждому экземпляру таблицы **branch** присвоить свой псевдоним, чтобы сервер знал, на какой экземпляр делается ссылка:

```
SELECT a.account_id, e.emp_id,
   b_a.name open_branch, b_e.name emp_branch
FROM account a INNER JOIN branch b_a
   ON a.open_branch_id = b_a.branch_id
   INNER JOIN employee e
   ON a.open_emp_id = e.emp_id
   INNER JOIN branch b_e
   ON e.assigned_branch_id = b_e.branch_id
WHERE a.product_cd = 'CHK';
```

account_id	emp_id	open_branch	emp_branch
10	1	Headquarters	Headquarters
14	1	Headquarters	Headquarters
21	1	Headquarters	Headquarters
1	10	Woburn Branch	Woburn Branch
4	10	Woburn Branch	Woburn Branch
7	13	Quincy Branch	Quincy Branch
13	16	So. NH Branch	So. NH Branch
18	16	So. NH Branch	So. NH Branch
24	16	So. NH Branch	So. NH Branch
28	16	So. NH Branch	So. NH Branch

Этот запрос показывает, кто открыл каждый текущий счет, в каком отделении это произошло и к какому отделению приписан в настоящее время сотрудник, открывший счет. Таблица **branch** включена дважды под псевдонимами ***b_a*** и ***b_e***. Благодаря присваиванию разных псевдонимов каждому экземпляру таблицы **branch**, сервер сможет определить экземпляр, на который делается ссылка, – соединенный с таблицей **account** или с таблицей **employee**.

Таким образом, имеем пример запроса, в котором использование псевдонимов таблиц является ***обязательным***.

Рекурсивные соединения

Можно не только несколько раз включать одну и ту же таблицу в один запрос, фактически можно соединить таблицу с самой собой. В таблице **employee**, например, есть ***рекурсивный внешний ключ*** (*self-referencing foreign key*). Это означает, что она включает столбец (*superior_emp_id*), указывающий на первичный ключ в рамках той же таблицы. Этот столбец указывает на начальника сотрудника. С помощью ***рекурсивного соединения*** (*self-join*) можно создать запрос, в результате выполнения которого выводится список всех сотрудников с указанием имен их начальников:

```
SELECT e.fname, e.lname,
   e_mgr.fname mgr_fname, e_mgr.lname mgr_lname
FROM employee e INNER JOIN employee e_mgr
   ON e.superior_emp_id = e_mgr.emp_id;
```

fname	lname	mgr_fname	mgr_lname
Susan	Barker	Michael	Smith
Robert	Tyler	Michael	Smith
Susan	Hawthorne	Robert	Tyler
John	Gooding	Susan	Hawthorne
Helen	Fleming	Susan	Hawthorne
Chris	Tucker	Helen	Fleming
Sarah	Parker	Helen	Fleming
Jane	Grossman	Helen	Fleming

Этот запрос включает два экземпляра таблицы **employee**: из одного (под псевдонимом ***e***) извлекаются имена сотрудников, а из другого (под псевдонимом ***e_mgr***) – имена начальников. Подблок *on* использует эти псевдонимы для соединения таблицы **employee** с самой собой посредством внешнего ключа *superior_emp_id*.

Это еще один пример запроса, для которого псевдонимы таблиц являются ***обязательными***.

Сравнение эквисоединений с неэквисоединениями

Все запросы к нескольким таблицам, показанные до сих пор, использовали ***эквисоединения*** (*equi-joins*). Это означает, что для обеспечения успешности соединения значения двух таблиц должны совпадать. Эквисоединение всегда использует знак равенства, например:

```
ON e.assigned_branch_id = b.branch_id
```

Подавляющее большинство запросов использует эквисоединения, но можно также соединять таблицы посредством диапазонов значений, называемых ***неэквисоединениями*** (*non-equi-joins*). Вот пример запроса, осуществляющего соединение по диапазону значений:

```
SELECT e.emp_id, e.fname, e.lname, e.start_date
   FROM employee e INNER JOIN product p
      ON e.start_date >= p.date_offered
        AND e.start_date <= p.date_retired
WHERE p.name = 'no_fee checking';
```

Этот запрос соединяет две таблицы, между которыми нет взаимосвязей по внешним ключам. Задача – найти всех сотрудников, принятых в банк в то время, когда предлагалась услуга беспроцентного текущего вклада. Таким

образом, дата начала работы сотрудника должна находиться между датами начала и конца этой акции.

Литература:

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