Лекция 9. Подзапросы и их типы

Подзапросы – мощный инструмент, который можно использовать во всех четырех SQL-выражениях для работы с данными. В данной лекции мы подробно рассмотрим различные варианты применения подзапроса.

Что такое подзапрос?

Подзапрос (*subquery*) – это запрос, содержащийся в другом SQL-выражении (*содержащем выражении*). Подзапрос всегда заключен в круглые скобки и обычно выполняется до содержащего выражения. Как и любой другой запрос, подзапрос возвращает таблицу, которая может состоять из:

- Одной строки и одного столбца;
- Нескольких строк и одного столбца;
- Нескольких строк и столбцов.

Подзапрос действует как временная таблица, областью видимости которой является выражение (т. е. после завершения выполнения выражения сервер высвобождает всю память, отведенную под результаты подзапроса).

Рассмотрим простой пример подзапроса:

```
SELECT account_id, product_cd, cust_id, avail_balance
FROM account
WHERE account_id = (SELECT MAX(account_id) FROM account);
```

```
+------------+------------+---------+---------------+
| account_id | product_cd | cust_id | avail_balance |
+------------+------------+---------+---------------+
|         24 | SBL        |      13 |      50000.00 |
+------------+------------+---------+---------------+
```

В этом примере подзапрос возвращает максимальное значение столбца *account_id* таблицы **account**. Затем содержащее выражение возвращает данные по этому счету. Если возникают какие-нибудь вопросы по поводу того, что делает подзапрос, можно выполнить его отдельно и посмотреть, что он возвращает. Так как подзапрос возвращает одну строку и один столбец, то его можно использовать как одно из выражений в условии равенства.

Типы подзапросов

Наряду с различиями, отмеченными ранее относительно типа возвращаемой подзапросом таблицы (одна строка/один столбец, одна строка/несколько столбцов или несколько столбцов), подзапросы делят на следующие два типа:

1) ***Несвязанные подзапросы*** (могут выполняться отдельно от основного запроса);

2) ***Связанные подзапросы*** (ссылаются на столбцы содержащего выражения).

Несвязанные подзапросы

Приведенный ранее в лекции пример является несвязанным подзапросом. Он может выполняться самостоятельно и не использует ничего из содержащего выражения. Большинство подзапросов являются несвязанными. Упомянутый пример не только является несвязанным, но и возвращает таблицу, состоящую всего из одной строки и одного столбца. Такой тип подзапроса называется ***скалярным подзапросом***, и его можно помещать в любую часть условия, использующего обычные операторы (=, <>, <, >, <=, >=).

Следующий пример показывает применение скалярного подзапроса в условии неравенства:

```
SELECT account_id, product_cd, cust_id, avail_balance
FROM account
WHERE open_emp_id <> (SELECT e.emp_id
   FROM employee e INNER JOIN branch b
      ON e.assigned_branch_id = b.branch_id
   WHERE e.title = 'Head Teller' AND b.city = 'Woburn');
```

```
+------------+------------+---------+--------------+
| account_id | product_cd | cust_id | avail_balance |
+------------+------------+---------+--------------+
|          6 | CHK        |       3 |      1057.75 |
|          7 | MM         |       3 |      2212.50 |
|          8 | CHK        |       4 |       534.12 |
|          9 | SAV        |       4 |       767.77 |
|         10 | MM         |       4 |      5487.09 |
…
|         23 | CHK        |      12 |     38552.05 |
|         24 | SBL        |      13 |     50000.00 |
+------------+------------+---------+--------------+
```

17 строк

Этот запрос возвращает данные по всем счетам, которые не были открыты старшим операционистом отделения Woburn.

Подзапросы могут использовать любые из всех доступных блоков запроса (*select*, *from*, *where*, *group by*, *having*, *order by*).

Если при использовании в условии равенства подзапрос возвращает более одной строки, будет сформирована **ошибка**.

Подзапросы, возвращающие несколько строк и один столбец

Существует 4 оператора, позволяющие строить условия с подзапросами этих типов.

1. Операторы *in* и *not in*

Нельзя *приравнять* одно значение набору значений, но можно проверить *наличие* этого значения в наборе. Для этого используется оператор ***in***. Следующий запрос использует оператор *in* и подзапрос в правой части условия фильтрации для того, чтобы выявить руководящий состав банка:

```
SELECT emp_id, fname, lname, title
FROM employee
WHERE emp_id IN (SELECT superior_emp_id
   FROM employee);
```

```
+--------+---------+-----------+--------------------+
| emp_id | fname   | lname     | title              |
+--------+---------+-----------+--------------------+
|      1 | Michael | Smith     | President          |
|      3 | Robert  | Tyler     | Treasurer          |
|      4 | Susan   | Hawthorne | Operations Manager |
|      6 | Helen   | Fleming   | Head Teller        |
|     10 | Paula   | Roberts   | Head Teller        |
|     13 | John    | Blake     | Head Teller        |
|     16 | Theresa | Markham   | Head Teller        |
+--------+---------+-----------+--------------------+
```

Подзапрос возвращает ID всех сотрудников, имеющих кого-то в подчинении, а основной запрос извлекает для этих сотрудников четыре столбца таблицы **employee**.

Можно проверять не только наличие значения в наборе значений, но и его *отсутствие*. Делается это с помощью оператора ***not in***. Вот другой вариант предыдущего запроса с оператором *not in* вместо *in*:

```
SELECT emp_id, fname, lname, title
FROM employee
WHERE emp_id NOT IN (SELECT superior_emp_id
   FROM employee
   WHERE superior_emp_id IS NOT NULL);
```

emp_id	fname	lname	title
2	Susan	Barker	Vice President
5	John	Gooding	Loan Manager
7	Chris	Tucker	Teller
8	Sarah	Parker	Teller
9	Jane	Grossman	Teller
11	Thomas	Ziegler	Teller
12	Samantha	Jameson	Teller
14	Cindy	Mason	Teller
15	Frank	Portman	Teller
17	Beth	Fowler	Teller
18	Rick	Tulman	Teller

Этот запрос находит всех сотрудников, которые никем *не* руководят. Здесь потребовалось добавить в подзапрос также условие фильтрации, чтобы гарантировать отсутствие значений ***null*** в возвращаемой подзапросом таблице.

2. Оператор *all*

Оператор ***in*** используется для поиска выражения в наборе выражений, а оператор ***all*** позволяет проводить сравнение одиночного значения с каждым значением набора. Для построения такого условия также понадобится использовать один из операторов сравнения (=, <>, <, > и т.д.). Следующий запрос находит всех сотрудников, ID которых не равен ни одному из ID руководителей:

```
SELECT emp_id, fname, lname, title
FROM employee
WHERE emp_id <> ALL (SELECT superior_emp_id
   FROM employee
   WHERE superior_emp_id IS NOT NULL);
```

emp_id	fname	lname	title
2	Susan	Barker	Vice President
5	John	Gooding	Loan Manager
7	Chris	Tucker	Teller
8	Sarah	Parker	Teller
9	Jane	Grossman	Teller
11	Thomas	Ziegler	Teller
12	Samantha	Jameson	Teller
14	Cindy	Mason	Teller
15	Frank	Portman	Teller
17	Beth	Fowler	Teller
18	Rick	Tulman	Teller

Результаты этого запроса полностью аналогичны результатам последнего примера с оператором ***not in***.

! Сравнивать значения с набором значений с помощью операторов ***not in*** или ***<>all*** нужно аккуратно, убедившись, что в наборе нет значения ***null***. Сервер приравнивает значение из левой части выражения к каждому члену набора, и любая попытка приравнять значение к ***null*** дает в результате *unknown*. Например,

```
SELECT emp_id, fname, lname, title
FROM employee
WHERE emp_id NOT IN (1, 2, NULL);
Empty set (0.00 sec)
```

3. Оператор *any*

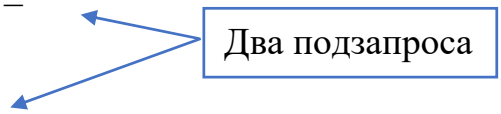
Как и оператор ***all***, оператор ***any*** обеспечивает возможность сравнивать значение с элементами набора значений. Однако, в отличие от ***all***, условие, использующее оператор ***any***, истинно (*true*), если есть хотя бы одно совпадение, тогда как при использовании оператора ***all*** требуется, чтобы условие выполнялось для всех элементов набора.

Операторы ***=any*** и ***in*** эквиваленты.

Подзапросы, возвращающие несколько столбцов

В определенных ситуациях требуется использовать подзапросы, возвращающие два или более столбцов. Рассмотрим для начала пример использования нескольких подзапросов, возвращающих один столбец:

```
SELECT account_id, product_cd, cust_id
FROM account
WHERE open_branch_id = (SELECT branch_id
   FROM branch
   WHERE name = 'Woburn Branch')
  AND open_emp_id IN (SELECT emp_id
   FROM employee
   WHERE title = 'Teller' OR title = 'Head Teller');
```



account_id	product_cd	cust_id
1	CHK	1
2	SAV	1
3	CD	1
4	CHK	2
5	SAV	2
14	CD	7
22	BUS	11

Чтобы выявить ID отделения Woburn и ID всех банковских операционистов, этот запрос использует два подзапроса.

Те же результаты можно получить и с единственным подзапросом к таблицам **employee** и **branch** путем сравнения столбцов *account.open_branch_id* и *account.open_emp_id*, так как в таблице **employee** есть информация об отделении, в котором числится каждый сотрудник. Для этого в условии фильтрации следует указать в круглых скобках имена обоих столбцов таблицы **account** в том же порядке, в каком они возвращаются подзапросом:

```
SELECT account_id, product_cd, cust_id
FROM account
WHERE (open_branch_id, open_emp_id) IN
   (SELECT b.branch_id, e.emp_id
    FROM branch b INNER JOIN employee e
       ON b.branch_id = e.assigned_branch_id
    WHERE b.name = 'Woburn Branch'
    AND (e.title = 'Teller' OR e.title = 'Head Teller'));
```

Связанные подзапросы

Связанный подзапрос зависит от содержащего выражения, из которого он ссылается на один или более столбцов. В отличие от *несвязанного подзапроса*, который выполняется непосредственно перед выполнением содержащего выражения, связанный подзапрос выполняется по разу для каждой строки-кандидата (это строки, которые предположительно могут быть включены в окончательные результаты).

Например, следующий запрос использует связанный подзапрос для подсчета количества счетов у каждого клиента. Затем основной запрос выбирает тех клиентов, у которых ровно по два счета:

```
SELECT c.cust_id, c.cust_type_cd, c.city
FROM customer c
WHERE 2 = (SELECT COUNT(*)
   FROM account a
   WHERE a.cust_id = c.cust_id);
```

```
+---------+--------------+---------+
| cust_id | cust_type_cd | city    |
+---------+--------------+---------+
|       2 | I            | Woburn  |
|       3 | I            | Quincy  |
|       6 | I            | Waltham |
|       8 | I            | Salem   |
|      10 | B            | Salem   |
+---------+--------------+---------+
```

Связанный подзапрос выполняется 13 раз (один раз для каждого клиента).

Ссылка на *c.cust_id* в самом конце подзапроса – это то, что делает этот подзапрос связанным. Если подзапрос возвращает значение 2, условие фильтрации выполняется, и строка добавляется в результирующий набор.

Помимо *условий равенства* связанные подзапросы можно применять в условиях других типов, таких как *условие вхождения в диапазон*:

```
SELECT c.cust_id, c.cust_type_cd, c.city
FROM customer c
WHERE (SELECT SUM(a.avail_balance)
      FROM account a
      WHERE a.cust_id = c.cust_id)
   BETWEEN 5000 AND 10000;
```

Этот вариант приведенного ранее запроса находит всех клиентов, чей общий доступный остаток по всем счетам находится в диапазоне от 5000 до 10 000 долларов. И снова связанный подзапрос выполняется 13 раз (по разу для каждой строки), и каждое выполнение подзапроса возвращает общий остаток по счетам данного клиента.

Оператор *exists*

Связанные подзапросы часто используются в условиях равенства и вхождения в диапазон, но самый распространенный оператор, применяемый в условиях со связанными подзапросами, – это оператор ***exists***. Оператор ***exists*** применяется, если требуется показать, что связь есть, а количество связей при этом не имеет значения. Например, следующий запрос находит все счета, для которых транзакция была выполнена в определенный день, без учета количества транзакций:

```
SELECT a.account_id, a.product_cd, a.cust_id, a.avail_balance
FROM account a
WHERE EXISTS (SELECT 1
   FROM transaction t
   WHERE t.account_id = a.account_id
      AND t.txn_date = '2005-01-22');
```

При использовании оператора ***exists*** подзапрос может возвращать ни одной, одну или много строк, а условие просто проверяет, возвращены ли в результате выполнения подзапроса строки (все равно сколько).

Если взглянуть на блок *select* подзапроса, можно увидеть, что он состоит из единственного литерала (1); для условия основного запроса имеет значение только число возвращенных строк, а что именно было возвращено подзапросом – не важно. В принципе, подзапрос может возвращать все, что вам вздумается; однако при использовании ***exists*** принято задавать *select 1* или *select **.

Для поиска подзапросов, не возвращающих строки, используется оператор ***not exists***:

```
SELECT a.account_id, a.product_cd, a.cust_id
FROM account a
WHERE NOT EXISTS (SELECT 1
   FROM business b
   WHERE b.cust_id = a.cust_id);
```

account_id	product_cd	cust_id
1	CHK	1
2	SAV	1
3	CD	1
4	CHK	2
5	SAV	2
6	CHK	3
7	MM	3
8	CHK	4
9	SAV	4
10	MM	4
11	CHK	5
12	CHK	6
13	CD	6
14	CD	7
15	CHK	8
16	SAV	8
17	CHK	9
18	MM	9
19	CD	9

Этот запрос выявляет всех клиентов, ID которых нет в таблице **business** (т.е. выбираются только клиенты-физические лица).

Манипулирование данными с помощью связанных подзапросов

Кроме выражения *select*, подзапросы также широко задействуются в выражениях *update*, *delete* и *insert*, а связанные подзапросы часто применяются в выражениях *update* и *delete*. Вот пример связанного подзапроса, с помощью которого изменяется столбец *last_activity_date* таблицы **account**:

```
UPDATE account a
SET a.last_activity_date =
   (SELECT MAX(t.txn_date)
    FROM transaction t
    WHERE t.account_id = a.account_id);
```

Это выражение корректирует все строки таблицы **account** (поскольку блока *where* нет), выбирая дату последней операции для каждого счета.

Вот другой вариант выражения ***update***, на этот раз использующий блок *where* со вторым связанным подзапросом:

```
UPDATE account a
SET a.last_activity_date =
   (SELECT MAX(t.txn_date)
    FROM transaction t
    WHERE t.account_id = a.account_id)
WHERE EXISTS (SELECT 1
   FROM transaction t
   WHERE t.account_id = a.account_id);
```

Эти два связанных подзапроса идентичны, за исключением блоков *select*. Однако подзапрос блока *set* выполняется, только если условие блока *where* выражения *update* истинно (*true*) (т.е. для счета была найдена, по крайней мере, одна операция). Таким образом данные столбца *last_activity_date* защищены от перезаписи значением ***null***.

Связанные подзапросы часто используются и в выражениях ***delete***. Например, в конце каждого месяца запускается сценарий, уничтожающий ненужные данные. Этот сценарий может включать следующее выражение, которое удаляет из таблицы **department** данные, не имеющие дочерних строк в таблице **employee**:

```
DELETE FROM department
WHERE NOT EXISTS (SELECT 1
   FROM employee
   WHERE employee.dept_id = department.dept_id);
```

При использовании связанных подзапросов в выражениях *delete* в MySQL необходимо помнить, что *псевдонимы* таблиц **не допускаются**. Для большинства других серверов БД можно было бы снабдить таблицы **department** и **employee** псевдонимами:

```
DELETE FROM department d
WHERE NOT EXISTS (SELECT 1
   FROM employee e
   WHERE e.dept_id = d.dept_id);
```

Литература:

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