Лекция 4. Фильтрация

В данной лекции мы изучим различные типы условий фильтрации, которые могут применяться в блоках *where* выражений *select*, *update* и *delete*.

Оценка условия

Блок *where* может содержать одно или более условий, разделенных операторами ***and*** и ***or***. При использовании только оператора ***and*** строка будет включена в результирующий набор в случае истинности (*True*) всех условий для нее. Рассмотрим следующий блок *where*:

```
WHERE title = 'Teller' AND start_date < '2019-01-01'
```

Исходя из этих двух условий, из рассмотрения будет исключен любой сотрудник, не являющийся операционистом или работающий в банке, начиная с 2019 года.

Если все условия в блоке *where* разделены оператором ***or***, то чтобы строка вошла в результирующий набор, должно выполниться (принять значение *true*) хотя бы одно из них. Например,

```
WHERE title = 'Teller' OR start_date < '2019-01-01'
```

Ниже показаны возможные результаты вычисления блока *where*, содержащего два условия, разделенных оператором ***or***.

Промежуточный результат	**Конечный результат**
WHERE true OR true	True
WHERE true OR false	True
WHERE false OR true	True
WHERE false OR false	False

Скобки

Если блок включает три или больше условий с использованием как оператора ***and***, так и ***or***, следует применять круглые скобки. Это сделает намерения запроса понятными и для сервера БД, и для всех, кто будет читать код. Например,

```
WHERE end_date IS NULL
    AND (title = 'Teller' OR start_date < '2019-01-01')
```

Теперь имеем три условия. Чтобы строка попала в конечный результирующий набор, первое условие для нее должно быть истинным (*true*), а также истинным должно быть второе *или* третье условие (или оба).

Оператор *not*

Рассмотрим следующую модификацию предыдущего примера:

```
WHERE end_date IS NULL
    AND NOT (title = 'Teller' OR start_date < '2019-01-01')
```

Здесь выбираются неуволенные сотрудники, которые или не являются операционистами, или начали работу в банке в 2019 и позже.

Сервер легко обработает такое выражение, однако человеку оценить блок, включающий оператор ***not***, обычно трудно. Поэтому мы можем переписать данное выражение *where*, не используя оператор ***not***:

```
WHERE end_date IS NULL
    AND title != 'Teller' AND start_date >= '2019-01-01'
```

Создание условия

Условие образуют одно или более *выражений*, попарно объединенных одним или более операторами. **Выражением** может быть любое из следующего:

- Число
- Столбец таблицы или представления
- Строковый литерал, например *'Teller'*
- Встроенная функция, например *CONCAT('Learning', ' ', 'SQL')*
- Подзапрос
- Список выражений, например (*'Teller', 'Head Teller', 'Operations Manager'*)

К **операторам**, используемым в условиях, относятся:

- Операторы сравнения, такие как =, !=, <, >, <>, LIKE, IN и BETWEEN
- Арифметические операторы, такие как +, –, * и /.

Типы условий

1) Условия равенства

Многие из создаваемых или существующих условий фильтрации имеют форму ***'столбец = выражение'***:

```
title = 'Teller'
amount = 375.25
dept_id = (SELECT dept_id FROM department WHERE name = 'Loans')
```

Такие условия называются ***условиями равенства***, потому что они проверяют равенство одного выражения другому. Следующий запрос использует два условия равенства, одно в блоке *on* (условие соединения) и второе в блоке *where* (условие фильтрации):

```
SELECT pt.name product_type, p.name product
FROM product p INNER JOIN product_type pt
    ON p.product_type_cd = pt.product_type_cd
WHERE pt.name = 'Customer Accounts';
```

```
+------------------+------------------------+
| product_type     | product                |
+------------------+------------------------+
| Customer Accounts | certificate of deposit |
| Customer Accounts | checking account       |
| Customer Accounts | money market account   |
| Customer Accounts | savings account        |
+------------------+------------------------+
```

Условия неравенства

Другой достаточно распространенный тип условия – ***условие неравенства***, которое определяет, что два выражения **не** равны.

```
SELECT pt.name product_type, p.name product
FROM product p INNER JOIN product_type pt
    ON p.product_type_cd = pt.product_type_cd
WHERE pt.name <> 'Customer Accounts';
```

!= или <>

```
+-----------------------------+------------------------+
| product_type                | product                |
+-----------------------------+------------------------+
| Individual and Business Loans | auto loan              |
| Individual and Business Loans | business line of credit |
| Individual and Business Loans | home mortgage          |
| Individual and Business Loans | small business loan    |
```

В результате этого запроса выводятся все счета, *не* являющиеся лицевыми счетами.

Условия равенства/неравенства обычно используются при изменении данных. Например, в банке принято уничтожать записи о старых счетах раз в год. Задача состоит в удалении из таблицы **account** строк с данными о счетах, закрытых в 2019 году. Вот одно из возможных решений:

```
DELETE FROM account
    WHERE status = 'CLOSED' AND YEAR(close_date) = 2019;
```

2) Условия вхождения в диапазон

Кроме проверки равенства (или неравенства) одного выражения другому, можно создать условия, проверяющие, попадает ли выражение в *определенный диапазон*. Этот тип условия широко используется при работе с числовыми или временными данными. Рассмотрим следующий запрос, который выявляет всех сотрудников, нанятых до 2007 года:

```
SELECT emp_id, fname, lname, start_date
FROM employee
WHERE start_date < '2007-01-01';
```

```
+--------+---------+-----------+------------+
| emp_id | fname   | lname     | start_date |
+--------+---------+-----------+------------+
|      1 | Michael | Smith     | 2005-06-22 |
|      2 | Susan   | Barker    | 2006-09-12 |
|      3 | Robert  | Tyler     | 2005-02-09 |
|      4 | Susan   | Hawthorne | 2006-04-24 |
|      8 | Sarah   | Parker    | 2006-12-02 |
```

Кроме верхней границы даты начала работы, можно задать и нижнюю границу:

```
SELECT emp_id, fname, lname, start_date
FROM employee
WHERE start_date < '2007-01-01'
    AND start_date >= '2005-01-01';
```

Оператор *between*

Если имеются верхняя и нижняя границы диапазона, вместо двух разных условий можно использовать одно, использующее оператор ***between***:

```
SELECT emp_id, fname, lname, start_date
FROM employee
WHERE start_date BETWEEN '2005-01-01' AND '2006-12-31';
```

При работе с оператором ***between*** необходимо помнить два правила:

- Первой всегда должна задаваться нижняя граница диапазона, а потом (после ***and***) верхняя граница.

- Верхняя и нижняя границы включаются в диапазон.

Как и для дат, можно создавать условия, определяющие диапазон для **чисел**. Например,

```
SELECT account_id, product_cd, cust_id, avail_balance
FROM account
WHERE avail_balance BETWEEN 3000 AND 5000;
```

```
+------------+------------+---------+---------------+
| account_id | product_cd | cust_id | avail_balance |
+------------+------------+---------+---------------+
|          3 | CD         |       1 |       3000.00 |
|         17 | CD         |       7 |       5000.00 |
|         18 | CHK        |       8 |       3487.19 |
+------------+------------+---------+---------------+
```

Здесь выбираются все счета, доступный остаток которых составляет от 3000 до 5000 долларов.

Строковые диапазоны

Можно также создавать условия для поиска диапазона **строк**. Например, требуется найти всех клиентов, номер социальной страховки которых находится между '500-00-0000' и '999-99-9999':

```
SELECT cust_id, fed_id
FROM customer
WHERE cust_type_cd = 'I'
    AND fed_id BETWEEN '500-00-0000' AND '999-99-9999';
```

```
+---------+-------------+
| cust_id | fed_id      |
+---------+-------------+
|       5 | 555-55-5555 |
|       6 | 666-66-6666 |
```

Для работы со строковыми диапазонами необходимо знать порядок символов в наборе символов.

3) Условия членства

В некоторых случаях выражение ограничивается не одним значением или диапазоном значений, а конечным набором значений. Например, требуется выбрать все счета, кодом типа которых является 'CHK', 'SAV', 'CD' или 'MM':

```
SELECT account_id, product_cd, cust_id, avail_balance
FROM account
WHERE product_cd = 'CHK' OR product_cd = 'SAV'
    OR product_cd = 'CD' OR product_cd = 'MM';
```

account_id	product_cd	cust_id	avail_balance
1	CHK	1	1057.75
2	SAV	1	500.00
3	CD	1	3000.00
4	CHK	2	2258.02
5	SAV	2	200.00
7	CHK	3	1057.75
8	MM	3	2212.50

Если же набор выражений содержит большое количество элементов, то в таких ситуациях можно использовать оператор ***in***. Например,

```
SELECT account_id, product_cd, cust_id, avail_balance
FROM account
WHERE product_cd IN ('CHK','SAV','CD','MM');
```

Иногда требуется проверить, присутствует ли определенное выражение в наборе выражений, а иногда нужно удостовериться в его *отсутствии*. В таких ситуациях можно использовать оператор ***not in***:

```
SELECT account_id, product_cd, cust_id, avail_balance
FROM account
WHERE product_cd NOT IN ('CHK','SAV','CD','MM');
```

account_id	product_cd	cust_id	avail_balance
25	BUS	10	0.00
27	BUS	11	9345.55
29	SBL	13	50000.00

4) Условия соответствия

Данный тип условий касается частичного соответствия строк. Например, требуется найти всех сотрудников, фамилия которых начинается с

«*T*». Получить первую букву значения столбца *lname* можно с помощью встроенной функции:

```
SELECT emp_id, fname, lname
FROM employee
WHERE LEFT(lname, 1) = 'T';
```

```
+--------+--------+--------+
| emp_id | fname  | lname  |
+--------+--------+--------+
|      3 | Robert | Tyler  |
|      7 | Chris  | Tucker |
|     18 | Rick   | Tulman |
+--------+--------+--------+
```

Хотя встроенная функция ***left()*** выполняет то, что требуется, она не обеспечивает особой гибкости. Вместо нее в выражениях поиска можно использовать **символы маски**. Они применяются для следующих случаев:

- Строки, начинающиеся/заканчивающиеся определенным символом
- Строки, начинающиеся/заканчивающиеся подстрокой
- Строки, содержащие определенный символ в любом месте строки
- Строки, содержащие подстроку в любом месте строки
- Строки определенного формата, независимо от входящих в них отдельных символов

Символ маски	**Соответствие**
_	Точно один символ
%	Любое число символов (в том числе ни одного)

При построении условий, использующих выражения поиска, применяется оператор ***like***. Например,

```
SELECT lname
FROM employee
WHERE lname LIKE '_a%e%';
```

```
+-----------+
| lname     |
+-----------+
| Barker    |
| Hawthorne |
| Parker    |
| Jameson   |
```

В таблице ниже показано еще несколько выражений поиска и их интерпретации.

Выражение поиска	**Интерпретация**
F%	Строки, начинающиеся с «F»
%t	Строки, заканчивающиеся на «t»
%bas%	Строки, содержащие подстроку «bas»
_ _t_	Строки, состоящие из четырех символов с «t» в третьей позиции
_ _ _ - _ _ - _ _ _ _	Строки из 11 символов, где четвертый и седьмой символы – дефисы

Символы маски также можно использовать для построения множественных выражений поиска:

```
SELECT emp_id, fname, lname
FROM employee
WHERE lname LIKE 'F%' OR lname LIKE 'G%';
```

```
+--------+-------+----------+
| emp_id | fname | lname    |
+--------+-------+----------+
|      5 | John  | Gooding  |
|      6 | Helen | Fleming  |
|      9 | Jane  | Grossman |
|     17 | Beth  | Fowler   |
+--------+-------+----------+
```

Значение null

Null – это отсутствие значения. Например, пока сотрудник не уволен, в его столбце *end_date* таблицы **employee** должно быть записано null.

При работе с null необходимо **помнить**:

• Выражение может *быть* нулевым (null), но оно никогда не может быть *равным* нулю.
• Два null никогда не равны друг другу.

Проверить выражение на значение null можно с помощью оператора **is null**. Например, следующий запрос возвращает всех сотрудников, у которых нет начальника (superior):

```
SELECT emp_id, fname, lname, superior_emp_id
FROM employee
WHERE superior_emp_id IS NULL;
```

= NULL – ошибка!

```
+--------+---------+-------+-----------------+
| emp_id | fname   | lname | superior_emp_id |
+--------+---------+-------+-----------------+
|      1 | Michael | Smith |            NULL |
+--------+---------+-------+-----------------+
```

Проверка наличия значения в столбце осуществляется с помощью оператора **is not null**.

Литература:

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