Содержание

УРОК 7

Развитая фильтрация данных

На этом уроке демонстрируется сочетание предложений `WHERE` *для создания эффективных и изощренных условий поиска, а также применение условных операций* `NOT` *и* `IN`.

Сочетание предложений `WHERE`

Все предложения `WHERE`, представленные на уроке 6, позволяют отфильтровать данные по единому критерию. Для более полного контроля над фильтрацией в Oracle допускается указывать несколько предложений `WHERE`. Этими предложениями можно пользоваться двумя способами: сочетая их с логической операцией `AND` или `OR`.

> *Операция* — это специальное ключевое слово, применяемое для сочетания или изменения выражений, употребляемых в предложении **`WHERE`**. Называется также *логической операцией*.

Применение логической операции `AND`

Чтобы отфильтровать извлекаемые данные по нескольким столбцам, достаточно воспользоваться логической операцией `AND`, присоединив условия фильтрации к предложению `WHERE`. В следующем примере кода показано, как это делается:

Ввод ▼

```
SELECT prod_id, prod_price, prod_name
FROM products
WHERE vend_id = 1003 AND prod_price <= 10;
```

Анализ ▼

В приведенном выше запросе SQL извлекаются наименования и цены на все товары, произведенные поставщиком с идентификатором `1003` по цене `10` или еще ниже. В операторе `SELECT` из этого запроса предложение `WHERE` состоит из двух условий, соединяемых логической операцией `AND`, которая предписывает СУБД возвратить только те строки, которые удовлетворяют обоим заданным условиям. Так, если товар, произведенный поставщиком с идентификатором `1003`, стоит больше `10`, то сведения о нем не извлекаются из таблицы. Аналогично из таблицы не извлекаются товары стоимостью меньше `10`, произведенные другим, а не указанным в запросе поставщиком. Результаты, выводимые по данному запросу SQL, выглядят следующим образом:

Вывод ▼

```
+---------+------------+----------------+
| prod_id | prod_price | prod_name      |
+---------+------------+----------------+
| FB      |         10 | Bird seed      |
| FC      |        2.5 | Carrots        |
| SLING   |       4.49 | Sling          |
| TNT1    |        2.5 | TNT (1 stick)  |
| TNT2    |         10 | TNT (5 sticks) |
+---------+------------+----------------+
```

> **`AND`** — это ключевое слово, употребляемое в предложении **`WHERE`** для указания на то, что из таблицы будут извлечены только те строки, которые удовлетворяют всем заданным в запросе условиям.

В приведенном выше примере употреблена единственная логическая операция `AND`, а следовательно, она состоит из двух условий фильтрации. В предложении `WHERE` можно указать и больше условий фильтрации, разделив их ключевым словом `AND`.

> ПРИМЕЧАНИЕ: **запросы SQL без предложения ORDER BY**
>
> В приведенных выше примерах отсутствует предложение **ORDER BY**. Но при желании вы можете ввести его в свой запрос SQL.

Применение логической операции OR

По своему действию логическая операция OR совершенно противоположна логической операции AND. В частности, логическая операция OR предписывает Oracle извлечь строки, удовлетворяющие любому из двух заданных условий.

Рассмотрим в качестве примера следующий запрос SQL с оператором SELECT:

Ввод ▼

```
SELECT prod_name, prod_price
FROM products
WHERE vend_id = 1002 OR vend_id = 1003;
```

Анализ ▼

По приведенному выше запросу SQL из таблицы извлекаются наименования и цены любых товаров, произведенных указанными поставщиками. Логическая операция OR предписывает СУБД обнаружить соответствие любому из двух заданных условий, а не обоим вместе. Если бы в данном запросе была употреблена логическая операция AND, то никаких данных не было бы возвращено, поскольку в этом случае было бы составлено такое предложение WHERE, которому нельзя было бы вообще удовлетворить. Результаты, выводимые по данному запросу SQL, выглядят следующим образом:

Вывод ▼

```
+----------------+------------+
| prod_name      | prod_price |
+----------------+------------+
| Detonator      |         13 |
| Bird seed      |         10 |
| Carrots        |        2.5 |
| Fuses          |       3.42 |
| Oil can        |       8.99 |
| Safe           |         50 |
| Sling          |       4.49 |
| TNT (1 stick)  |        2.5 |
| TNT (5 sticks) |         10 |
+----------------+------------+
```

> **OR** — это ключевое слово, употребляемое в предложении **WHERE** для указания на то, что из таблицы будут извлечены строки, совпадающие с любым из заданных в запросе условий.

Общее представление о порядке вычисления

Предложения WHERE могут состоять из любого количества логических операций AND и OR. Благодаря сочетанию обоих этих операций можно организовать довольно изощренную фильтрацию.

Но сочетание логических операций AND и OR представляет собой интересную проблему. Продемонстрируем ее решение на конкретном примере. Допустим, требуется получить список всех товаров стоимостью **10** или выше, произведенных поставщиками с идентификаторами **1002** и **1003**. В следующем запросе SQL с оператором SELECT логические операции сочетаются в предложении WHERE для извлечения товаров по указанным выше условиям:

Ввод ▼

```
SELECT prod_name, prod_price
FROM products
WHERE vend_id = 1002 OR vend_id = 1003 AND prod_price >= 10;
```

Вывод ▼

```
+----------------+------------+
| prod_name      | prod_price |
+----------------+------------+
| Detonator      |         13 |
| Bird seed      |         10 |
| Fuses          |       3.42 |
| Oil can        |       8.99 |
| Safe           |         50 |
| TNT (5 sticks) |         10 |
+----------------+------------+
```

Анализ ▼

Проанализируем приведенные выше результаты. В двух из всех возвращаемых строк содержатся цены ниже **10**, а следовательно, эти строки отфильтрованы не так, как требуется. Почему так произошло? Ответ на этот вопрос кроется в порядке вычисления. В языке SQL, как и в большинстве других языков, логические операции `AND` выполняются прежде логических операций `OR`. Если в запросе SQL обнаруживается предшествующее предложение `WHERE`, то из таблицы извлекаются товары, произведенные поставщиком с идентификатором **1002**, независимо от их цены, а также любые товары стоимостью **10** или выше, произведенные поставщиком с идентификатором **1003**. Иными словами, логическая операция `AND` имеет более высокий порядок вычисления, и поэтому в данном запросе сочетаются вместе неверные логические операции.

Решение описанной выше проблемы состоит в использовании круглых скобок для явного группирования связанных вместе логических операций. Рассмотрим в качестве примера следующий запрос SQL с оператором `SELECT` и проанализируем результаты его выполнения:

Ввод ▼

```
SELECT prod_name, prod_price
FROM products
WHERE (vend_id = 1002 OR vend_id = 1003) AND prod_price >= 10;
```

Вывод ▼

```
+----------------+------------+
| prod_name      | prod_price |
+----------------+------------+
| Detonator      |         13 |
| Bird seed      |         10 |
| Safe           |         50 |
| TNT (5 sticks) |         10 |
+----------------+------------+
```

Анализ ▼

Единственное отличие этого запроса SQL от предыдущего заключается в том, что первые два условия в предложении `WHERE` заключены в круглые скобки. А поскольку круглые скобки имеют более высокий порядок вычисления, чем логические операции `AND` или `OR`, то СУБД отфильтровывает сначала данные по условию `OR` в этих круглых скобках, а затем по данному запросу SQL извлекаются любые товары, произведенные поставщиком с идентификатором **`1002`** или **`1003`** стоимостью **`10`** или выше, что, собственно, и требуется.

> СОВЕТ: **употребление круглых скобок в предложениях `WHERE`**
>
> Всякий раз, когда вы составляете предложение **`WHERE`**, в котором употребляются логические операции **`AND`** и **`OR`**, заключите группу логических операций в круглые скобки явным образом. Не полагайтесь только на соблюдаемый по умолчанию порядок вычисления, даже если это именно то, что вам требуется. Использование круглых скобок не таит в себе никаких недостатков, напротив — они позволяют избежать любой неоднозначности в коде.

Применение операции `IN`

Круглые скобки находят еще одно, но совсем другое применение в предложениях `WHERE`. В частности, для указания ряда

условий, каждое из которых может быть удовлетворено, служит условная операция IN. Эта операция принимает разделяемый запятой список достоверных значений, причем все они заключены в круглые скобки, как демонстрируется в следующем примере:

Ввод ▼

```
SELECT prod_name, prod_price
FROM products
WHERE vend_id IN (1002,1003)
ORDER BY prod_name;
```

Вывод ▼

```
+----------------+------------+
| prod_name      | prod_price |
+----------------+------------+
| Bird seed      |         10 |
| Carrots        |        2.5 |
| Detonator      |         13 |
| Fuses          |       3.42 |
| Oil can        |       8.99 |
| Safe           |         50 |
| Sling          |       4.49 |
| TNT (1 stick)  |        2.5 |
| TNT (5 sticks) |         10 |
+----------------+------------+
```

Анализ ▼

По приведенному выше запросу SQL оператором SELECT извлекаются все товары, произведенные поставщиками с идентификаторами **1002** и **1003**. Разделяемый запятой список достоверных значений следует после операции IN, и поэтому он должен быть заключен в круглые скобки.

Если вы считаете, что условная операция IN служит той же цели, что и логическая операция OR, то совершенно правы. В приведенном ниже примере по запросу SQL выполняются те же действия, что и в предыдущем примере.

Ввод ▼

```
SELECT prod_name, prod_price
FROM products
WHERE vend_id = 1002 OR vend_id = 1003
ORDER BY prod_name;
```

Вывод ▼

```
+---------------+------------+
| prod_name     | prod_price |
+---------------+------------+
| Bird seed     |         10 |
| Carrots       |        2.5 |
| Detonator     |         13 |
| Fuses         |       3.42 |
| Oil can       |       8.99 |
| Safe          |         50 |
| Sling         |       4.49 |
| TNT (1 stick) |        2.5 |
| TNT (5 sticks)|         10 |
+---------------+------------+
```

Зачем вообще употреблять операцию `IN`? Она дает следующие преимущества.

- Синтаксис операции `IN` более ясен и прост для работы с длинными списками достоверных значений.
- Порядок вычисления проще контролировать при использовании операции `IN`, поскольку в этом случае употребляется меньше операций.
- Условные операции `IN` почти всегда выполняются быстрее, чем списки логических операций `OR`, хотя обработка таких коротких списков, как в приведенных выше примерах запросов, практически никак не сказывается на производительности.
- Самое главное преимущество операции `IN` заключается в том, что она может содержать другой оператор `SELECT`, что дает возможность составлять весьма динамичные предложения `WHERE`. Подробнее об этом см. урок 14.

IN — это ключевое слово, употребляемое в предложении **WHERE** для указания списков значений, совпадающих при сравнении по логической операции **OR**.

Применение операции NOT

Условная операция NOT, употребляемая в предложении WHERE, выполняет одну и только одну функцию. Она отрицает любое следующее далее условие.

NOT — это ключевое слово, употребляемое в предложении **WHERE** для отрицания заданного условия.

В приведенном ниже примере демонстрируется применение операции NOT. Чтобы перечислить товары, произведенные всеми поставщиками, кроме обозначенных идентификаторами **1002** и **1003**, можно сделать следующий запрос SQL:

Ввод ▼

```
SELECT prod_name, prod_price
FROM products
WHERE vend_id NOT IN (1002,1003)
ORDER BY prod_name;
```

Вывод ▼

```
+--------------+------------+
| prod_name    | prod_price |
+--------------+------------+
| .5 ton anvil |       5.99 |
| 1 ton anvil  |       9.99 |
| 2 ton anvil  |      14.99 |
| JetPack 1000 |         35 |
| JetPack 2000 |         55 |
+--------------+------------+
```

Анализ ▼

В данном примере операция NOT служит для отрицания указанного после нее условия. Таким образом, вместо совпадения со

значением **1002** или **1003** в Oracle обнаруживается совпадение с любыми значениями в столбце `vend_id`, кроме **1002** или **1003**.

Так зачем же нужна условная операция `NOT`? Для составления простых предложений `WHERE` ее применение действительно не дает никаких преимуществ. Условная операция `NOT` оказывается более полезной для составления сложных предложений. Например, сочетая условные операции `NOT` и `IN`, можно упростить поиск всех строк, не соответствующих заданным критериям.

Резюме

На этом уроке было продолжено рассмотрение материала предыдущего урока. В частности, вы научились сочетать предложения `WHERE` с логическими операциями `AND` и `OR`, а также управлять явным образом порядком вычисления и пользоваться условными операциями `IN` и `NOT`.