

## Операции в SQL.

Операции представляются зарезервированными словами или символами. В SQL операции используются в выражениях ключевого слова WHERE.

- Операции сравнения
- Логические операции
- Операция отрицания
- Арифметические операции

Операции сравнения

**Операции сравнения** используются в операторах SQL для сравнения отдельных значений и представляются знаками =, <, > и <=. Эти операции предназначены соответственно для проверки равенства и неравенства значений, проверки выполнения отношений "меньше" и "больше" между ними.

**Операция проверки равенства** в операторе SQL выясняет равенство одного значения другому. Для этого используется знак равенства (=). При выяснении равенства сравниваемые значения должны совпадать в точности, иначе запрос к базе данных не вернет никаких данных. Если сравниваемые значения равны, соответствующее выражение получает значение TRUE (Истина), иначе — FALSE (Ложь). Это логическое значение (TRUE/FALSE) используется системой для того, чтобы выяснить, должны ли соответствующие данные включаться в ответ запроса.

Пример

Значение

WHERE SALARY = '20000'

Зарплата равна 20000

**Ввод:**

**SELECT \* FROM PRODUCTTBL WHERE PROD\_ID = '2345';**

**Вывод:**

PROD_ID	PROD_DESC	COST
2345	ПОЛОЧКА ИЗ ДУБА	59.99

## Неравенство

В противоположность равенству существует неравенство. В SQL для неравенства используется знак  $\neq$  (комбинация знаков "меньше" и "больше"). В этом случае условие возвращает TRUE, если обнаруживается неравенство значений, и FALSE — если равенство.

Во многих из основных реализаций SQL эквивалентом знака операции  $\neq$  является комбинация  $\neq$

Пример

Значение

WHERE SALARY  $\neq$  '20000'

Зарплата не равна 20000

Знаки < ("меньше") и > ("больше") можно использовать по отдельности, и в комбинации с другими операциями.

*Логические операции в SQL* задаются ключевыми словами, а не символами.

Ниже мы рассмотрим следующие логические операции.

- IS NULL
- EXISTS
- BETWEEN
- UNIQUE
- IN
- ALL И ANY
- LIKE

IS NULL

Ключевое слово `is NULL` используется для проверки равенства данного значения значению `NULL`

<u>Пример</u>	<u>Значение</u>
<code>WHERE SALARY is NULL</code>	Для зарплаты не задано значение

Вот пример, в котором значение `NULL` не будет найдено.

<u>Пример</u>	<u>Значение</u>
<code>WHERE SALARY = NULL</code>	Зарплата имеет значение, равное строке символов <code>N-U-L-L</code>

Ввод:

```
SELECT EMP_ID, LAST_NAME, FIRST_NAME, PAGER  
FROM EMPLOYEE_TBL  
WHERE PAGER IS NULL;
```

Ввод:

```
SELECT EMP_ID, LAST_NAME, FIRST_NAME, PAGER  
FROM EMPLOYEE_TBL  
WHERE PAGER = NULL;
```

# BETWEEN

Ключевое слово BETWEEN используется для поиска значений, попадающих в диапазон, заданный некоторыми минимальным и максимальным значениями. Эти минимальное и максимальное значения включаются в соответствующее условие.

<u>Пример</u>	<u>Значение</u>
WHERE SALARY BETWEEN '20000' AND '30000'	Зарплата должна быть от 20000 до 30000, включая их

```
Ввод:  
SELECT *  
FROM PRODUCTS_TBL  
WHERE BETWEEN 5.95 AND 14.5;
```

## IN

Ключевое слово IN используется для сравнения значения с заданным списком буквальных значений. Чтобы возвратилось TRUE, сравниваемое значение должно совпадать хотя бы с одним значением из списка.

### Пример

### Значение

WHERE SALARY IN ('20000',  
'30000', '40000')

Зарплата должна равняться  
20000, 30000 или 40000

### **Ввод:**

**SELECT \***

**FROM PRODUCTS\_TBL**

**WHERE PROD\_ID IN (<'13','9','87','119');**

То же самое можно получить, комбинируя условия с помощью ключевого слова OR, но с помощью IN результат получается быстрее.

# LIKE

Ключевое слово LIKE используется для нахождения значений, похожих на заданное. В данном случае предполагается использование следующих двух знаков подстановки: • знак процента (%); • знак подчеркивания (\_).

Знак процента представляет ноль, один или несколько символов. Знак подчеркивания представляет один символ или число. Знаки подстановки могут использоваться в комбинации.

## Пример

## Значение

WHERE SALARY LIKE '200%'	Любое значение, начинающееся с 200
WHERE SALARY LIKE '%200%'	Любое значение, имеющее 200 в любой позиции
WHERE SALARY LIKE '_00%'	Любое значение, имеющее 00 во второй и третьей позициях
WHERE SALARY LIKE '2_%_'	Любое значение, начинающееся с 2 и состоящее как минимум из трех символов
WHERE SALARY LIKE '%2'	Любое значение, заканчивающееся 2
WHERE SALARY LIKE '_2%3'	Любое значение, имеющее 2 во второй позиции и заканчивающееся 3
WHERE SALARY LIKE '2__3'	Любое значение длиной 5 символов, начинающееся с 2 и заканчивающееся 3

В следующем примере выбираются описания для тех товаров, описания которых заканчиваются на "Ы".

**Ввод:**

```
SELECT PROD_DESC  
FROM PRODUCTS_TBL  
WHERE PROD_DESC LIKE '%Ы';
```

В следующем примере выбираются описания товаров с буквой "Ы" во второй позиции.

**Ввод:**

```
SELECT PROD_DESC  
FROM PRODUCTS_TBL  
WHERE PROD_DESC LIKE '_Ы%';
```

# EXISTS

Ключевое слово EXISTS используется для поиска в таблице строк, удовлетворяющих заданным критериям.

Пример

Значение

WHERE EXISTS (SELECT EMP_ID	Проверка наличия EMP_ID со
FROM EMPLOYEE_TBL	значением 333333333 в
таблице	WHERE EMPLOYEE_ID = '333333333')
EMPLOYEE_TBL	

В следующем примере в операторе используется подчиненный запрос

Ввод:

```
SELECT COST
FROM PRODUCTS_TBL
WHERE EXISTS ( SELECT COST
FROM PRODUCTS_TBL
WHERE COST > 100 )
```

## UNIQUE

Ключевое слово UNIQUE используется для проверки строк заданной таблицы на уникальность (т. е. отсутствие повторений).

### Пример

```
WHERE UNIQUE (SELECT SALARY  
FROM EMPLOYEE_TBL  
WHERE EMPLOYEE_ID = '333333333')
```

### Значение

Проверка SALARY на наличие повторений

## ALL И ANY

Ключевое слово ALL используется для сравнения заданного значения со всеми значениями из некоторой другой выборки значений.

### Пример

```
WHERE SALARY > ALL (SELECT SALARY  
FROM EMPLOYEE_TBL  
WHERE CITY = 'PSKOV')
```

Пскова

### Значение

Проверка значения FROM SALARY на предмет превышения им всех значений зарплаты служащих из

**Ввод:**

```
SELECT *  
FROM PRODUCTS_TBL  
WHERE COST > ALL ( SELECT COST  
FROM PRODUCTS_TBL  
WHERE COST < 10 );
```

Ключевое слово ANY используется для сравнения заданного значения с любым из значений некоторой другой выборки значений.

<u>Пример</u>	<u>Значение</u>
<pre><b>WHERE SALARY &gt; ANY (SELECT SALARY FROM EMPLOYEE_TBL WHERE CITY = 'PSKOV')</b></pre>	Проверка значения SALARY на Предмет превышения им какого-нибудь из значений платы для
— зар — — служащих из Пскова	—

**Ввод:**

**SELECT \***

**FROM PRODUCTS\_TBL**

**WHERE COST > ANY ( SELECT COST**

**FROM PRODUCTS\_TBL**

**WHERE COST < 10 );**

## Операции конъюнкции и дизъюнкции

Как быть, если необходимо использовать несколько условий, чтобы сузить набор возвращаемых запросом данных? Нужно скомбинировать условия с помощью *операций конъюнкции и дизъюнкции*. Эти операции задаются с помощью ключевых слов AND и OR.

### AND

Ключевое слово AND позволяет связать логическим умножением два условия в выражении ключевого слова WHERE. Чтобы оператор SQL, представляющий транзакцию или запрос, выполнил заданное действие, оба связанные ключевым словом AND условия должны возвратить TRUE.

#### Пример

<u>Пример</u>	<u>Значение</u>
WHERE EMPLOYEE_ID = '333333333' AND SALARY = '20000'	Значение EMPLOYEE_ID должно быть равным 333333333, а значение SALARY должно быть равным 20000

**Ввод:**

```
SELECT *  
FROM PROOCTS_TBL  
WHERE COST > 10  
AND COST < 30;
```

**Ввод:**

```
SELECT *  
FROM PRODUCTS_TBL  
WHERE PROD_ID = '7725'  
AND PROD_ID = '2345';
```

**OR**

Ключевое слово **OR** позволяет связать логическим сложением условия в выражении ключевого слова **WHERE**. Чтобы оператор **SQL**, представляющий транзакцию или запрос, выполнил заданное действие, хотя бы одно из связанных ключевым словом **AND** условий должно вернуть **TRUE**.

Пример

Значение

<b>WHERE SALARY = ' 20000 '</b> <b>OR SALARY = '30000'</b>	Значение <b>SALARY</b> должно быть равным либо 20000, либо 30000
---	---

Операции сравнения и логические операции в выражениях могут использоваться самостоятельно или в комбинации с другими подобными операциями.

**SELECT \***

**FROM PRODUCTS\_TBL**

**WHERE PROD\_ID = '7725'**

**OR PROD\_ID = '2345';**

## Отрицание условий с помощью операции отрицания

Для всех выше рассмотренных типов операций можно построить их отрицания, чтобы таким образом рассмотреть противоположные условия.

Ключевое слово NOT обращает смысл операции, с которой оно используется. Ключевое слово NOT используется с операциями следующим образом.

- NOT BETWEEN
- IS NOT NULL
- NOT IN
- NOT EXISTS
- NOT LIKE
- NOT UNIQUE

Пример

Значение

WHERE SALARY <> '20000'

Зарплата не равна 20000

WHERE SALARY != '20000'

Зарплата не равна 20000

## Арифметические операции

Арифметические операции используются в SQL точно так же, как и в большинстве других языков. Таких операций четыре:

- + (сложение);
- \* (умножение);
- - (вычитание);
- / (деление).

Сложение - знак "+".

Пример

Значение

SELECT SALARY + BONUS  
FROM EMPLOYEE\_PAY\_TBL;      Значение SALARY складывается  
со значением BONUS для  
каждой -

SELECT SALARY FROM EMPLOYEE\_PAY\_TBL      строки данных  
WHERE SALARY + BONUS > '40000'; Выбор строк, для которых  
сумма SALARY И BONUS

превышает 40000

Вычитание - знак "-".

<u>Пример</u>	<u>Значение</u>
SELECT SALARY - BONUS FROM EMPLOYEE_PAY_TBL;	Значение BONUS вычитается из значения SALARY
SELECT SALARY FROM EMPLOYEE_PAY_TBL BONUS > '40000';	Выбор строк, для WHERE SALARY > '40000';
— разность SALARY и BONUS Превышает 40000	—

Умножение - знак "\*".

<u>Пример</u>	<u>Значение</u>
SELECT SALARY * 10 FROM EMPLOYEE_PAY_TBL;	Значение SALARY умножается на 10
SELECT SALARY FROM EMPLOYEE_PAY_TBL WHERE SALARY * 10 > '40000';	Выбор строк, для которых значение SALARY * 10 > '40000';
значение	умноженное на 10, превышает 40000

## Деление

Деление представлено знаком "/" (косой чертой).

### Пример

### Значение

```
SELECT SALARY /10  
FROM EMPLOYEE_PAY_TBL;
```

Значение SALARY делится на 10

```
SELECT SALARY FROM EMPLOYEE_PAY_TBL  
WHERE SALARY / 10 > '40000';
```

Выбор строк,

для которых значение SALARY, деленное на 10, превышает 40000

## Комбинирование арифметических операций

Арифметические операции можно комбинировать. Сначала выполняются операции умножения и деления, а затем — операции сложения и вычитания. Пользователь может управлять порядком выполнения операций в выражении только с помощью скобок. Заключенное в скобки выражение означает необходимость рассматривать выражение как единый блок.

*Порядок выполнения операций (приоритет операций)* задает порядок, в котором обрабатываются выражения в математических выражениях или встроенных функциях SQL.

```
SELECT SALARY * 10 + 1000  
FROM EMPLOYEE_PAY_TBL  
WHERE SALARY > 20000;
```

```
SELECT SALARY / 52 + BONUS  
FROM EMPLOYEE_PAY_TBL;
```

```
SELECT (SALARY - 1000 + BONUS) / 52 * 1.1  
FROM EMPLOYEE_PAY_TBL;
```

При использовании в выражении нескольких арифметических операций учитывайте порядок выполнения арифметических операций, поскольку неправильно расставленные скобки обычно приводят к неправильным результатам