# Лабораторная работа 18. Скалярные операторы MySQL

Цель: изучить скалярные операторы MySQL.

Ссылка: https://phpclub.ru/mysql/doc/logical-operators.html

#### Задание. Выполнить в MySQL все последующие запросы, по возможности используя как аргумент СВОЙ НОМЕР ПО ВАРИАНТУ.

#### Операторы сравнения

Операторы сравнения дают в результате величину 1 (истина, TRUE), 0 (ложь, FALSE) или NULL. Эти функции работают как с числами, так и со строками. Строки при необходимости автоматически преобразуются в числа, а числа - в строки (как в Perl).

Операции сравнения в MySQL выполняются по следующим правилам:

* Если один или оба аргумента - NULL, то и результат сравнения будет NULL. Справедливо для всех операторов кроме <=>.
* Если оба аргумента в операторе сравнения являются строками, то они сравниваются как строки.
* Если оба аргумента - целые числа, то они сравниваются как целые числа.
* Шестнадцатеричные величины, если они не сравниваются с числом, трактуются как строки с двоичными данными.
* Если один из аргументов представляет собой столбец типа TIMESTAMP или DATETIME, а второй аргумент - константа, то константа перед выполнением сравнения преобразуется к типуTIMESTAMP. Это сделано для лучшей совместимости с ODBC.
* Во всех других случаях аргументы сравниваются как действительные числа с плавающей точкой.

По умолчанию сравнение строк производится без учета регистра символов с использованием текущего набора символов (по умолчанию ISO-8859-1 Latin1, который, к тому же, прекрасно подходит для английского языка).

Ниже приведены примеры, иллюстрирующие преобразование строк в числа для операторов сравнения:

SELECT 1 > '6x';

0

SELECT 7 > '6x';

1

SELECT 0 > 'x6';

0

SELECT 0 = 'x6';

1

* =

Равно:

SELECT 1 = 0;

0

SELECT '0' = 0;

1

SELECT '0.0' = 0;

1

SELECT '0.01' = 0;

0

SELECT '.01' = 0.01;

1

* <> , !=

Не равно:

SELECT '.01' <> '0.01';

1

SELECT .01 <> '0.01';

0

SELECT 'zapp' <> 'zappp';

1

* <=

Меньше или равно:

SELECT 0.1 <= 2;

1

* <

Меньше чем:

SELECT 2 < 2;

0

* >=

Больше или равно:

SELECT 2 >= 2;

1

* >

Больше чем:

SELECT 2 > 2;

0

* <=>

NULL-безопасное сравнение (равно):

SELECT 1 <=> 1, NULL <=> NULL, 1 <=> NULL;

1 1 0

* IS NULL , IS NOT NULL

Тест для определения, является величина равной NULL или нет:

SELECT 1 IS NULL, 0 IS NULL, NULL IS NULL;

0 0 1

SELECT 1 IS NOT NULL, 0 IS NOT NULL, NULL IS NOT NULL;

1 1 0

Для того, чтобы MySQL хорошо работал с другими программами, обеспечивается поддержка следующих дополнительных возможностей для функции IS NULL:

* + Можно найти последнюю вставленную строку, используя выражение:
  + SELECT \* FROM tbl\_name WHERE auto\_col IS NULL

Это свойство можно блокировать установкой SQL\_AUTO\_IS\_NULL=0.

* + Для данных типа NOT NULL DATE и столбцов DATETIME можно найти особую дату 0000-00-00, используя выражение:
  + SELECT \* FROM tbl\_name WHERE date\_column IS NULL

Это необходимо для работы некоторых приложений ODBC (так как ODBC не поддерживает значение даты 0000-00-00).

* expr BETWEEN min AND max

Если величина выражения expr больше или равна заданному значению min и меньше или равна заданному значению max, то функция BETWEEN возвращает 1, в противном случае - 0. Это эквивалентно выражению (min <= expr AND expr <= max), в котором все аргументы представлены одним и тем же типом данных. В противном случае имеет место быть преобразование типов так, как сказано выше, но применительно ко всем трем аргументами.

SELECT 1 BETWEEN 2 AND 3;

0

SELECT 'b' BETWEEN 'a' AND 'c';

1

SELECT 2 BETWEEN 2 AND '3';

1

SELECT 2 BETWEEN 2 AND 'x-3';

0

* expr NOT BETWEEN min AND max

То же справедливо и для функции NOT (expr BETWEEN min AND max).

* expr IN (value,...)

Возвращает 1, если выражение expr равно любой величине из списка IN, иначе - 0. Если все величины - константы, то они оцениваются в соответствии с типом выражения expr и сортируются. Поиск элемента в этом случае производится методом логического поиска. Это означает, что функция IN является очень быстрой, если список значений IN состоит полностью из констант. Если expr является зависимым от регистра строковым выражением, то сравнение строк производится с учетом регистра:

SELECT 2 IN (0,3,5,'wefwf');

0

SELECT 'wefwf' IN (0,3,5,'wefwf');

1

Начиная с 4.1 (в соответствии со стандартом SQL-99), IN возвращает NULL не только если выражение в левой части является NULL, но также если не найдено соответствия в списке и одно из выражений в списке является величиной NULL.

* expr NOT IN (value,...)

То же справедливо и для функции NOT (expr IN (value,...)).

* ISNULL(expr)

Если expr равно NULL, то ISNULL() возвращает 1, в противном случае - 0:

SELECT ISNULL(1+1);

0

SELECT ISNULL(1/0);

1

Обратите внимание: при сравнении величин NULL с использованием оператора = всегда будет возвращаться значение FALSE!

* COALESCE(list)

Возвращает первый в списке элемент со значением, не равным NULL:

SELECT COALESCE(NULL,1);

1

SELECT COALESCE(NULL,NULL,NULL);

NULL

* INTERVAL(N,N1,N2,N3,...)

Возвращает 0, если N < N1, и 1, если N < N2, и так далее. Все аргументы трактуются как целые числа. Для корректной работы этой функции необходимо условие N1 < N2 < N3 < ... < Nn. Это обусловлено тем, что используется логический поиск (очень быстрый):

SELECT INTERVAL(23, 1, 15, 17, 30, 44, 200);

3

SELECT INTERVAL(10, 1, 10, 100, 1000);

2

SELECT INTERVAL(22, 23, 30, 44, 200);

0

Если регистронезависимая строка сравнивается с помощью любого стандартного оператора (=, <>, ..., но не LIKE), то конечные пустые символы (т.е. пробелы, табуляторы и переводы строк) игнорируются: игнорируется.

SELECT "a" ="A \n";

1

#### Арифметические операции

В MySQL можно применять обычные арифметические операторы. Следует иметь в виду, что если оба аргумента являются целыми числами, то при использовании операторов '-', '+' и '\*' результат вычисляется с точностью BIGINT (64 бита). Если один из аргументов - беззнаковое целое число, а второй аргумент - также целое число, то результат будет беззнаковым целым числом.

* +

Сложение:

SELECT 3+5;

8

* -

Вычитание:

SELECT 3-5;

-2

* \*

Умножение:

SELECT 3\*5;

15

SELECT 18014398509481984\*18014398509481984.0;

324518553658426726783156020576256.0

SELECT 18014398509481984\*18014398509481984;

0

В последнем выражении получим неверный результат, так как произведение умножения целых чисел выходит за границы 64-битового диапазона для вычислений с точностью BIGINT.

* /

Деление:

SELECT 3/5;

0.60

Деление на ноль приводит к результату NULL:

SELECT 102/(1-1);

NULL

Деление будет выполняться по правилам BIGINT-арифметики только в случае, если эта операция представлена в контексте, где ее результат преобразуется в INTEGER!

#### Логические операторы

В SQL, все логические операторы возвращают TRUE (ИСТИНА), FALSE (ЛОЖЬ) или NULL (UNKNOWN, неизвестно). В MySQL это реализовано как 1 (TRUE, ИСТИНА), 0 (FALSE, ЛОЖЬ) или NULL. Это справедливо для большинства SQL СУБД, однако некоторые возвращают любое положительное значение как значение TRUE.

* NOT , !

Логическое НЕ. Возвращает 1, если операнд равен 0, 0 если операнд - ненулевая величина, и NOT NULL возвращает NULL.

SELECT NOT 10;

0

SELECT NOT 0;

1

SELECT NOT NULL;

NULL

SELECT ! (1+1);

0

SELECT ! 1+1;

1

Последний пример дает 1, поскольку данное выражение вычисляется тем же способом, что и (!1)+1.

* AND , &&

Логическое И. Дает 1 если все операнды ненулевые и не NULL, 0 если один или более операндов равны 0, или NULL в остальных случаях.

SELECT 1 && 1;

1

SELECT 1 && 0;

0

SELECT 1 && NULL;

NULL

SELECT 0 && NULL;

0

SELECT NULL && 0;

0

* OR , ||

Логическое ИЛИ. Возвращает 1, если любой из операндов не 0, NULL если один из операндов NULL, в остальных случаях возвращает 0.

SELECT 1 || 1;

1

SELECT 1 || 0;

1

SELECT 0 || 0;

0

SELECT 0 || NULL;

NULL

SELECT 1 || NULL;

1

* XOR

Логический XOR (побитовое сложение по модулю 2) Возвращает NULL если любой из операндов - NULL. Для не-NULL операндов, возвращает 1 если нечетное количество операндов - не 0.

SELECT 1 XOR 1;

0

SELECT 1 XOR 0;

1

SELECT 1 XOR NULL;

NULL

SELECT 1 XOR 1 XOR 1;

1

a XOR b математически эквалиентно (a AND (NOT b)) OR ((NOT a) and b).

**Контрольные вопросы.**

1. Использование арифметических операций.

2. Использование логических операций.

3. Использование операций сравнения.