Стандартные булевы опе­раторы, распознаваемые SQL, -это AND, OR, NОТ.

• AND берет два булевых выражения (в виде А AND В) в качестве аргументов и дает в результате истину, если они оба истинны.

• OR два булевых выражения (в виде А OR В) в качестве аргументов и оценивает результат как истину, если хотя бы один из них истинен.

• NOT берет единственное булево выражение (в виде NOT А) в качестве аргу­мента и изменяет его значение с истинного на ложное или с ложного на истин­ное.

Пример: выбрать всё из customers где city равно 'San Jose' или raiting не больше 200(меньше 200)

SELECT \* FROM customers WHERE city='San Jose' OR NOT raiting>200;

+------+----------+----------+---------+------+ РАЗБОР :

| cnum | cname | city | raiting | snum |

+------+----------+----------+---------+------+

| 2001 | Hoffman | London | 100 | 1001 |city!='San Jose' no raiting <200 poetomy vivodim

| 2002 | Giovanni | Rome | 200 | 1003 |city!='San Jose' no raiting <200 poetomy vivodim

| 2003 | Liu | San Jose | 200 | 1002 |city=='San Jose' poetomy vivodim

| 2006 | Clements | London | 100 | 1001 |city!='San Jose' no raiting <200 poetomy vivodim

| 2008 | Cisneros | San Jose | 300 | 1007 |city=='San Jose' poetomy vivodim

| 2007 | Perejra | Rome | 100 | 1004 |city!='San Jose' no raiting <200 poetomy vivodim

+------+----------+----------+---------+------+

+------+----------+----------+---------+------+

| 2004 | Grass | Berlin | 300 | 1002 |city!='San Jose'i raiting>200 poetomy ne vivodim

+------+----------+----------+---------+------+

ОПЕРАТОР "**IN**" **используется в условии WHERE.**

Полностью определяет множество (то,что в скобках), которому данное значение может принадлежать или не принадлежать.т.е. выводит в результат то, что в скобках (то, что в in);

**SELECT \* FROM salespeople WHERE city='BARSELONA' OR city ='London';** <--простой запрос

тоже самле но с "IN"

**SELECT \* FROM salespeople WHERE city IN ('BARSELONA','London');** т.е. выведи все строки из salespeople где city = всё-то, что в скобках.

IN определяет множество, элементы которого точно перечисляются в круглых скобках и разделяются запятыми. Если в поле, имя которого ука­зано слева от IN, есть одно из перечисленных в списке значений (требуется точное совпадение), то предикат считается истинным. Если элементы множества имеют чи­словой, а не символьный тип, то одиночные кавычки непосредственно слева и справа от значения необходимо опустить.

**SELECT \* FROM customers WHERE snum IN (1001, 1007, 1004);**

—--------------------------------------------------------------------------------------------------------------------------------------------

Оператор **BETWEEN используется в условии WHERE.**

сходен с IN. Вместо перечисления элементов множества, как это делается в IN, BETWEEN ЗАДАЁТ ГРАНИЦЫ, в которые должно попадать значение, чтобы предикат был истинным. Используется ключевое слово BETWEEN, за которым следуют начальное значение, ключевое слово AND и конечное значение. т.е. BETWEEN начальное\_значение and канечное\_значение

пример: **SELECT \* FROM salsepeople WHERE comm BETWEEN 10 AND 12;**

Оператор BETWEEN является включающим, т.е. граничные значения (в данном примере это 10 и 12) включаются в предикат.

интересный пример:

**SELECT \***

**FROM salespeopl**

**WHERE (comm BETWEEN 10 AND 12) AND NOT comm IN (10, 12);**

первое булево выражение (comm BETWEEN 10 AND 12);второе булево выражение AND NOT comm IN (10, 12);В первом, сравниваются условие и если (comm=10 или 11 или 12 то вернётся true) дальше второе условие и если сomm =11. То ето строка выводится. т.е условие предиката (если сomm=10,11,12 и comm не равенн 10, 12 то эту строку выводим)<--ето равносильно(если comm=11 выводим на экран);

BETWEEN действует на символьных полях, представленных в двоичном (ASCII) эквиваленте, т.е. для выборки можно вос­пользоваться алфавитным порядком. Следующий запрос выбирает всех покупателей имена которых попадают в заданный алфавитный диапазон:

**SELECT •**

**FROM Customers**

**WHERE cname BEТWEEN 'A' AND 'G';**

Пример: Выбрать названия и количества тех книг, количество которых от 5 до 14 включительно.

**SELECT title, amount**

**FROM book**

**WHERE amount BETWEEN 5 AND 14;**

| title | amount

| Белая гвардия | 5

| Идиот | 10

**LIKE**

Применим только к полям типа CHAR или VARCHAR, поскольку он исполь­зуется для поиска подстрок. Другими словами, он осуществляет просмотр строки для выяснения: входит ли заданная подстрока в указанное поле. С этой же целью используются шаблоны, специальные символы, которые могут обозначать все, что угодно. Существует два типа шаблонов, используемых с LIКE:

--Символ "подчеркивание" (\_) заменяет один любой символ. Например, образцу 'Ь\_t' соответствуют 'Ьаt' или 'Ьit', но не соответствует 'Ьrat'. Так как в brat между b и t (b \_ \_ T) 2-а символа.

Внимание: Строчные и прописные буквы в строках эквивалентны(при построении запроса не надо парится верхний там регистр или нижний).

--Символ "процент" (%) заменяет последовательность символов произвольной длины, в том числе и нулевой. Например, образцу '%p%t' соответствуют 'put', 'posit', 'opt', но не 'spite'.

**SELECT что-либо**

**FROM откуда-либо**

**WHERE last\_name LIKE 'Ber%'**; <-вернёт все имена которые наченаются на Ber и заканчиваются на любые символы с любой длинной. Что-то типа:Bers,Berman, Bertran...

Для того чтобы вывести названия, состоящие из любого количества символов после LIKE можно использовать шаблон "%", с помощью которого отбираются строки, состоящие из любого количества символов, в том числе и "пустые", поскольку % заменяет любое количество символов, в том числе и нулевое.

Чтобы указать, что в названии должен быть хотя бы один символ, можно использовать один из эквивалентных шаблонов:

"**\_%**" - сначала идет символ, а за ним любое количество символов;

"**%\_**" - сначала идет любое количество символов, а затем обязательный символ;

"**%\_%**" - сначала идет любое количество символов, потом обязательный символ, а за ним любое количество символов.

**SELECT что-либо**

**FROM откуда-либо**

**WHERE last\_name LIKE '%ns%';**  <-вернёт имена которые содержат ns. Что-то тип: Venskii, Ansin, Konski…

Пример: Вывести названия книг, которые состоят ровно из одного слова, если считать, что слова в названии отделяются друг от друга пробелами

**SELECT title FROM book**

**WHERE title NOT LIKE "% %";**

Отсутствие пробела в названии означает, что оно состоит из одного слова. Чтобы это проверить используется оператор NOT LIKE, который в данном случае отберет все названия, в которых нет пробелов.

**SELECT что-либо**

**FROM откуда-либо**

**WHERE last\_name LIKE 'Ber\_ard';** <-вернёт 'Bernard', 'Berzard', 'Bermard', 'Bersard'...

**SELECT что-либо**

**FROM откуда-либо**

**WHERE last\_name LIKE '12345\_';** <-вернёт 123456,123454, 123453,123452,123451…

Пример: Вывести название книг, состоящих ровно из 5 букв.

**SELECT title FROM book WHERE title LIKE "\_ \_ \_ \_ \_"**

Пояснение- Для обозначения одного любого символа используется "\_", следовательно для обозначения 5 символов используется 5 подряд символов подчеркивания.

Пример:

Вывести название и автора тех книг, название которых состоит из двух и более слов, а инициалы автора содержат букву «С». Считать, что в названии слова отделяются друг от друга пробелами и не содержат знаков препинания, между фамилией автора и инициалами обязателен пробел, инициалы записываются без пробела в формате: буква, точка, буква, точка. Информацию отсортировать по названию книги в алфавитном порядке.

**SELECT title, author**

**FROM book**

**WHERE title LIKE "\_% %" AND author LIKE "%С.%"**

**ORDER BY title**

**SELECT что-либо**

**FROM откуда-либо**

**WHERE last\_name NOT LIKE 'R%'**; ←Помещая оператора NOT перед условием LIKE, получаем элементы которые не начинаются на 'R'.

Пример: Вывести из таблицы trip информацию о командировках тех сотрудников, фамилия которых заканчивается на букву «а», в отсортированном по убыванию даты последнего дня командировки виде. В результат включить столбцы name, city, per\_diem, date\_first, date\_last.

**SELECT name, city, per\_diem, date\_first, date\_last**

**FROM trip**

**WHERE name like "%a %"**

**ORDER BY date\_last DESC**

Предположим, вы хотели найти символы % или \_ в условии LIKE. Вы можете сделать это, используя символ Escape.Обратите внимание что вы можете определить только escape-символ как один символ(длина=1).

**SELECT \***

**FROM suppliers**

**WHERE supplier\_name LIKE 'B\%';**<-- вернёт элементы с именем B%. По умолчанию Escape символ это '\' соответственно символ следующий после \ будет рассматриватся как литерал и проверятся на наличие или совпадение в искомом элементе.

Также можно переопределить Escape символ используя key-word Escape.

**SELECT \***

**FROM suppliers**

**WHERE supplier\_name LIKE 'Br!%' ESCAPE '!';**<--вернёт элементы сименем 'Br%'

ELECT \* FROM suppliers WHERE supplier\_name LIKE 'H%\%';<--вернёт злементы с именем

'Hello%', 'Hi%', 'Hachapyri%'...

SELECT \* FROM suppliers WHERE supplier\_name LIKE 'H%\\_';<--вернёт элементы с именем

'Hello\_', 'Hi\_', Hachapyri\_'...

Как я это вижу: SELECT что-либо FROM откуда-либо WHERE LIKE 'H%\\_';

выведи строку похожую на 'H%\\_';

можно использовать NOT BETWEEN и NOT LIKE, NOT IN,

**SOME, ANY и ALL** -операторы КОТОРЫЕ РАБОТАЮТ С **ПОДЗАПРОСАМИ**. они берут все значения выведенные подзапросом и обрабатывают их как модуль(как одно целое) .

**any** - (хоть что либо), это когда хоть что либо соответствует условию-поподает в результат.

Например:

SELECT \*

FROM Salespeople

WHERE city = **ANY**

(SELECT city FROM Customers ); <-- Оператор ANY берет все значения выведенные подзапросом, ( для этого случая - это все значения city в таблице Заказчиков ), и оценивает их как верные если любой (ANY) из их равняется значению города текущей строки внешнего запроса. Это означает, что подзапрос должен выбирать значения такого же типа как и те, которые сравниваются в основном предикате.

ВАЖНО: операторы работают с любыми операторами сравнения:!=, =, <, >, <=, >=.

**ALL** –> compares a value of the first table with all values of the second table and returns the row if there is a match with all values.

В примерах ниже используется 10,12 как результат подзапроса, т.е. в коде на этом месте должен быть подзапрос, который бы вернул эти значения.

amount > ALL (10, 12) эквивалентно amount > 12

amount < ALL (10, 12) эквивалентно amount < 10

amount = ALL (10, 12) не вернет ни одной записи, так как эквивалентно (amount = 10) AND (amount = 12)

amount <> ALL (10, 12) вернет все записи кроме тех, в которых amount равно 10 или 12

SELECT \*

FROM Teachers

WHERE age > ALL (SELECT age FROM Students);

**LIMIT**

Для ограничения вывода записей в SQL используется оператор LIMIT, после которого указывается количество строк. Результирующая таблица будет иметь количество строк не более указанного после LIMIT. LIMIT размещается после раздела ORDER BY.

Как правило, этот оператор используется, чтобы отобрать заданное количество отсортированных строк результата запроса.

Пример: Вывести информацию о первой командировке из таблицы trip. "Первой" считать командировку с самой ранней датой начала.

**SELECT \***

**FROM trip**

**ORDER BY date\_first**

**LIMIT 1;**

Пример: Вывести два города, в которых чаще всего были в командировках сотрудники. Вычисляемый столбец назвать Количество.

**SELECT city, COUNT(city) AS 'Количество'**

**FROM trip**

**GROUP BY city**

**ORDER BY Количество DESC**

**LIMIT 2**