**Базовые SQL запросы. Скалярные функции. Операторы CASE, COALESCE, NULLIF и др. Работа с текстом, датой и временем.**

### **Синтаксис базового SQL запроса. Отбор данных.**

**Базовый синтаксис:**

select <столбцы>

from <таблица>

**Выбрать все столбцы:**

select \*

from <таблица>

**Примечание:** “\*” - в боевых запросах лучше избегать, используя перечисление столбцов.

**Алиасы**

Алиас – короткое имя столбца/таблицы. Назначается с помощью ключевого слова AS (может быть опущено):

select e.first\_name, e.last\_name

from employees AS e

или

select e.first\_name, e.last\_name

from employees e

**Вычисляемый столбец**

Вычисляемые столбцы — результат преобразований одного или нескольких исходных столбцов. Например:

select name, unit\_price \* quantity \* 100

from sales

Чаще всего вычисляемому столбцу нужно дать имя (алиас):

select name, unit\_price \* quantity \* 100 my\_col

from sales

### **Оператор WHERE. Фильтрация**

Оператор WHERE необходим для фильтрации строк. С его помощью можно задавать условия для выбора строк из таблицы.

**Базовый синтаксис:**

select <столбцы>

from <таблица>

where <условие>

Пример: вывести всю информацию о товарах из таблицы products, у которых цена больше 20:

select \*

from products

where unit\_price > 20

Принцип работы операторы: для каждой строки проверяется выполнение заданного после оператора WHERE условия. Выводятся только те строки, для которых условие выполняется.

**Примечание:** в операторе WHERE нельзя использовать алиасы столбцов, так как WHERE выполняется раньше, чем присваивается алиас.

**Комбинация условий**

С помощью оператора **AND** (логический оператор И) можно пересекать несколько условий фильтраций. То есть будут выбраны те строки, для которых выполняются все перечисленные условия.

Пример: найти товары из категории “Косметика”, стоимость которых больше 20 и меньше или равна 40:

select \*

from products

where category = “Косметика”

and unit\_price > 20

and unit\_price <= 40

**Примечание:** для фильтрации строк в некотором диапазоне, например, дат, цены или количества удобно использовать оператор **BETWEEN**. Тогда запрос выше можно записать компактнее:

select \*

from products

where category = “Косметика”

and unit\_price between 20 and 40

**Важно!** Но в этом случае левая и правая граница диапазона будут включены в выборку.

С помощью оператора **OR** (логический оператор ИЛИ) можно объединять несколько условий. То есть будут выбраны те строки, для которых выполняется хотя бы одно из перечисленных условий.

Пример: найти все товары из категории “Косметика”, товары, стоимость которых больше 100 и товары, количество которых меньше или равно 40:

select \*

from products

where category = “Косметика”

or unit\_price > 100

or quantity <= 40

Операторы AND и OR можно группировать.

**Примечание:** приоритетоператора AND выше приоритета оператора OR. Т.е. если встречаются рядом OR и AND, то сначала вычисляется выражение с AND.

Пример:

select \*

from products

where category = “Косметика” and unit\_price > 100

or quantity <= 40 and unit\_price < 50

Сначала проверяется условие **category = “Косметика” and unit\_price > 100**. Оно в свою очередь состоит из двух условий, соединенных оператором AND. Это значит, что составное условие будет выполнено только если оба условия **category = “Косметика”** и **unit\_price > 100** будут выполнены. Затем аналогично проверяется второе составное условие **quantity <= 40 and unit\_price < 50**. Если для строки выполняется хотя бы одно из составных условий **category = “Косметика” and unit\_price > 100** или **quantity <= 40 and unit\_price < 50**, тогда она будет выведена.

**Примечание:** чтобы задать нужный порядок вычисления условий, лучше брать выражения в скобки. Это также повышает читаемость кода.

Чтобы вывести строки, которые не удовлетворяют условию, нужно добавить перед условием оператор **NOT.**

Пример: вывести все товары, кроме тех, что принадлежат категории “Косметика”:

select \*

from products

where not category = “Косметика”

**NULL** - отсутствие значения.

**IS NULL** - отобрать строки, где есть пустые ячейки

**IS NOT NULL** - отобрать строки, где нет пустых ячеек

Пример: вывести все товары, для которых задана категория:

select \*

from products

where category is not null

**IN**  - оператор для указания множества значений для фильтрации.

Пример: вывести все товары из категорий “Косметика”, “Одежда”, “Аксессуары”:

select \*

from products

where category in (“Косметика”, “Одежда”, “Аксессуары”)

Оператор **DISTINCT** - отобрать уникальные значения.Можно применять только на одном столбце.

Пример: вывести уникальные id товаров, которые продавались в количество больше 40 штук

select distinct id

from products

where quantity > 40

Оператор **EXISTS** - показывает, существуют ли строки, для которых выполняется некоторое условие.

**Базовый синтаксис:**

select <столбцы>

from <таблица>

where exists (select <столбцы> from <таблица> … )

### **Оператор ORDER BY. Сортировка**

**Базовый синтаксис:**

select <столбцы>

from <таблица>

order by <столбцы> <тип: desc/asc >

desc - по убыванию.

asc - по возрастанию (по умолчанию).

Пример:

select product\_name

from products

order by quantity desc

**Примечание:** Если нужно отсортировать запрос по нескольким столбцам, то нужно указать через запятую наименование столбцов по которым будет проводиться сортировка.

Для сортировки можно указывать порядковые номера столбцов, а не их названия или даже вычисляемые значения. **Но это нежелательно!**

### **Операторы LIKE, ILIKE, SIMILAR TO**

**LIKE** - сравнивает с шаблоном, указанном в кавычках.

Шаблон может включать буквы, цифры, пробелы и специальные знаки:

% - специальный символ, который указывает, что вместо него может идти любая последовательность символов.

\_ - специальный символ, который указывает, что вместо него может стоять только один символ (любой).

\ - символ экранирование. Нужен для того чтобы указать, что написанный символ является простым символом, а не специальным. Например, ‘\%’ показывает, что нужно найти именно символ процента в базе, а не указать произвольное количество символов или \\_% - значит что пред произвольным количеством символов доолжно стоять нижнее подчеркивание.

Пример: шаблон, в котором первая буква - Д, потом идет один любой символ: ‘Д\_’.

Пример: шаблон всех имен, которые начинаются на букву А: ‘А%’.

Пример: шаблон всех слов и фраз, в которых встречается сочетание букв “on”: ‘%on%’.

**Важно:** LIKE - делает регистрозависимый поиск. То есть заглавные буквы и строчные считаются разными символами. Например, по шаблону ‘S%’ мы не найдем слово ‘sql’’.

**ILIKE** - делает все то же самое, что и оператор LIKE. Только ILIKE выполняет регистронезависимый поиск. То есть заглавные и строчные буквы рассматриваются как одинаковые символы.

**SIMILAR TO** - поиск по регулярным выражениям (специальный набор комбинаций символов).

D|B - вертикальная черта - специальный символ, который указывает, что должна идти буква D или B

\* - специальный символ, который указывает, что последовательность перед ним может повторяться произвольное количество раз

SIMILAR TO (D|B)\_\* - это поиск слова, начинающегося с D или B, далее после этой буквы идет любой один символ произвольное количество раз.

SIMILAR TO (D|B)\_{10} - если необходимо ограничить число символов, то в {} необходимо отметить количество символов.

Если в ILIKE подставить (D|B)\_{10}, то оператор будет воспринимать это не как шаблон, а как символы, которые нужно искать в чистом виде.

**Урок 5**

### **Оператор CASE**

**CASE** - аналог операторов SWITCH CASE и IF в других языках программирования. Определяет некоторые условия, в зависимости от которых для строки присваивается определенное значение.

Пример: Если цена продукта больше 100, то в столбец price\_group добавляется значение `A` , если больше 50, тогда `B`, в других случаях - `C`.

select \*,

case

when unit\_price > 100 then `A`

when unit\_price > 50 then `B`

else `C`

end price\_group

from products

**Примечание:** если какое-то условие выполняется, то дальше проверка условий не идет. То есть в нашем примере, если цена больше 100, то записывается значение `A` и следующие проверки делаться не будет, поэтому второе условие не сработает.

### 

### **Операторы COALESCE и NULLIF**

**COALESCE** - вычисляет аргументы по порядку и возвращает текущее значение первого выражения, изначально не вычисленного как NULL.

Например, SELECT COALESCE(NULL, NULL, 'third\_value', 'fourth\_value'); возвращает третье значение, так как это первое значение, не равное NULL.

**NULLIF**(expression1, expression2) - получает два аргумента, сравнивает их между собой. Если expression1 и expression2 равны, функция возвращает NULL. В противном случае он возвращает первое выражение expression1.

Например: вывести название товаров и их количество на складе. Если количество на складе равно 0, тогда путь ячейка остается пустой (NULL).

select product\_name,

nullif(units\_in\_stocks, 0)

from products p.

### **Преобразование типов**

Неявное преобразование скрыто от пользователя (пользователь специально не указывает тип данных). СУБД автоматически преобразует данные из одного типа в другой. Неявное преобразование типов - происходит при операции над элементами с приводимыми типами данных.

Типы называются приводимыми, если из одного типа можно получить другой. Например, из целого число (integer) можно получить число с плавающей точкой (numeric).

Например, если к numeric прибавить integer, то СУБД неявно преобразует integer в numeric и результат будет иметь тип numeric.

select (1 + 4.0)

Явное преобразование - пользователь сам меняет тип данных.

**CAST** - это оператор, который преобразует значение (локальную переменную, столбец или выражение) из одного типа данных в другой (применим только к приводимым типам данных).

Пример:

select cast(1 as numeric) – преобразовали число integer в numeric

select cast(units\_in\_stocks numeric)

from products

**Примечание:** вместо функцииcast можно использовать оператор (::), после которого указывается тип, в который нужно преобразовать данные. Но этот оператор есть только в PosgreSQL. В других СУБД могут быть другие операторы.

select units\_in\_stocks::numeric

### **Функции для работы с текстом**

**||** - оператор конкатенации строк (сложение).

select ‘Привет’|| ‘Как тебя зовут?’

**CONCAT()** - функция для конкатенации строк, которая принимает неограниченное количество аргументов.

select concat(company\_name,’:’,contact\_name,’:’,contact\_title) as info

from customers c

**Примечание:** Чтобы не писать каждый раз разделитель, можно использовать следующую функцию.

**CONCAT\_WS()** - первый аргумент разделитель, далее - строки, которые нужно сложить.

select concat\_ws(’:’,company\_name, contact\_name,contact\_title)

from customers c

**LENGTH()** - функция, которая определяет длину строки. Результат - целое число.

**LOWER()** - функция, которая переводит все в нижний регистр

**UPPER()** - функция, которая переводит все в верхний регистр

**INITCAP()** - функция, которая делает все первые буквы каждого слова заглавными, а остальные - строчными.

select lower(company\_name),

upper(contact\_name),

initcap(contact\_title)

from customers c

**POSITION(<string1> in <string2>)** - функция, которая осуществляет поиск подстроки в строке, возвращает позицию ее первого символа

select position(‘Москва’ in ‘город Москва’)

-- 7

**SUBSTRING(<string> [from <start\_position>] [for <quantity>] )** - выделение подстроки

select substring('Меня зовут Вася' from 1 for 4)

-- Меня

**TRIM()** - функция, которая удаляет указанные символы в начале и конце строки. По умолчанию - убирает пробелы.

select trim(' Привет ')

-- Привет

Чтобы убрать другие символы, нужно указать ключевое слово **BOTH**, тогда уберется и в начале, и в конце, **LEADING** - только в начале, **TRAILING** - только в конце.

select trim(both 'xyz ' from 'xy Привет z ')

-- Привет

select trim(leading 'xyz ' from 'xy Привет z ')

-- Привет z

select trim(trailing 'xyz ' from 'xy Привет z ')

-- xy Привет

**LEFT(string, N)** - отбор N левых символов

**RIGHT(string, N)** - отбор N правых символов

select left('aaa bbb',2)

-- aa

select right('aaa bbb',2)

-- bb

### **Функции для работы с датой и временем**

**CURRENT\_DATE** - текущая дата

**CURRENT\_TIME** - текущее время

**EXTRACT(<part> from <date>)** - дает возможность выделять часть даты или времени в формате числа

select extract(year from order\_date)

from orders o

select extract(month from order\_date)

from orders o

**MAKE\_DATE(year, month, day)** - функция, которая позволяет создать дату из чисел

select make\_date(2021, 3, 31)

-- 2021-03-31 00:00:00

**INTERVAL** - оператор, который указывает, что за ним идет некоторый интервал.

select make\_date(2021, 3, 31) + interval '3 days 5 hours 31 minutes'

-- 2021-03-31 05:31:00

**TO\_DATE (string, pattern)** - преобразование строки в дату на основании заданного паттерна (вид, в котором записана дата в строке)

select to\_date(‘10 Apr 2021’, ‘DD Mon YYYY’)

-- 2021-04-10

select to\_date(‘10 04 2021’, ‘DD MM YYYY’)

-- 2021-04-10

**TO\_TIMESTAMP (string, pattern) -** преобразование строки в дату со временем на основании заданного паттерна (вид, в котором записана дата в строке)

to\_char - преобразование даты и времени в строку

select to\_timedtamp(‘10 Apr 2021’, ‘DD Mon YYYY’)

-- 2021-04-10 00:00:00

select to\_timedtamp(‘10 Apr 2021 21 05 31’, ‘DD Mon YYYY HH24 MI SS’)

-- 2021-04-10 21:05:31

**TO\_CHAR(date, pattern)** - преобразование даты и времени в строку в указанном формате

select to\_char(2021-04-10, ‘DD.MM.YYYY’)

-- 10.04.2021

**DATE\_TRUNC(text, timestamp/interval)** - отсекает компоненты даты до заданной точности

select date\_trunc('hour', timestamp '2001-02-16 20:38:40')

-- 2001-02-16 20:00:00

select date\_trunc('day', interval '2 days 3 hours 40 minutes')

-- 2 days 00:00:00