

ИСПОЛЬЗОВАНИЕ WHERE И ORDER BY В SQL-ЗАПРОСАХ

| ВВЕДЕНИЕ

На этом уроке мы

- Продолжим работу с SQL-запросами
- Освоим применение оператора WHERE для фильтрации таблиц данных, а также оператора ORDER BY для упорядочивания данных в нужной последовательности по заданным вами критериям
- Освоим различные способы отбора данных, с использованием для этого логических операторов и функций языка SQL

| WHERE

Следующим по порядку после SELECT и FROM элементом SQL-запроса является конструкция WHERE. Она позволяет добавить в наш запрос фильтрацию таблицы данных по выбранных критериям

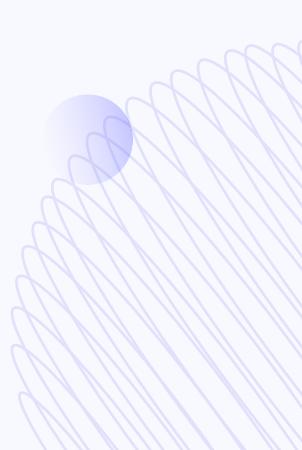
Критерии фильтрации могут формулироваться различными способами:

01 При помощи операторов: «==» либо «<>». При нескольких условиях можно их соединять при помощи логических операторов – **AND (и)** либо **OR (или)**

02 Оператор LIKE позволяющий задать поиск заданного паттерна в тексте. Противоположное действие – **NOT LIKE**

03 При помощи проверки нахождения в списке – оператор **IN**, либо не нахождения в списке – **NOT IN**

Сразу же разберем механизм использования фильтрации



| ПРИМЕРЫ

01

Выбрать из таблицы Customers (клиенты) данные о клиентах, проживающих в Лондоне:

о о о

```
SELECT *
FROM public.Customers
WHERE "City" = 'London'
```

Обратите внимание, что текст в запросе указывается в одинарных кавычках, что отличает текст от указания названия столбца

02

Выбрать из таблицы Customers (клиенты) все данные о клиентах

Проживающих в Лондоне и не являющихся торговыми представителями:

о о о

```
SELECT *
FROM public.customers
WHERE "City" = 'London' AND "ContactTitle" = 'Sales Representative'
```

Если мы хотим поменять условия на обратное, то есть получить данные о клиентах, не являющихся торговыми представителями, то необходимо применить оператор неравенства:

о о о

```
WHERE "City" = 'London' AND "ContactTitle" <> 'Sales Representative'
```

03

Выбрать из таблицы Customers (клиенты) все данные о клиентах, имя которых начинается с буквы А:

о о о

```
SELECT *
FROM public.customers
WHERE "ContactName" LIKE 'A%'
```

После оператора LIKE внутри одинарных кавычек мы можем использовать текстовое выражение, которое является «маской поиска», или паттерном распознавания текста. Символ «%» в данной маске обозначает любое количество любых символов, которые могут идти после первой буквы «A»

Если мы хотим немного дополнить условие: «Отобрать всех клиентов, имя которых начинается на «A» и оканчивается на «r», то мы можем видоизменить маску следующим образом:

```
○○○  
SELECT *  
FROM public.customers  
WHERE "ContactName" LIKE 'A%r'
```

Как видим, после данного изменения в таблице остались только строки, соответствующие заданной маске текста в столбце ContactName:

| Contact Name | ✎ |
|--------------------|---|
| Alexander Feuer | |
| Art Braunschweiger | |

04

Выбрать данные о клиентах, проживающих в Лондоне, Мадриде или Берлине

Для решения этой задачи мы можем использовать новый элемент SQД-запроса – список. Список – это объект, представляющий собой последовательность из нескольких значений, объединенных в одну коллекцию

Использование списков в коде позволяет осуществить проверку условия наличия либо отсутствия любого заданного значения в коллекции

```
○○○  
SELECT *  
FROM public.customers  
WHERE "City" IN ('London', 'Madrid', 'Berlin')
```

Для противоположного условия необходимо заменить IN на NOT IN:

о о о

```
SELECT *
FROM public.customers
WHERE "City" NOT IN ('London', 'Madrid', 'Berlin')
```

Другой способ решения задачи – использование нескольких последовательных условий неравенства через операторы AND либо OR

Для выбора только клиентов, проживающих в заданных трех городах, можно сформулировать условия отбора следующим образом:

о о о

```
SELECT *
FROM public.customers
WHERE "City" = 'London' OR "City" = 'Madrid' OR "City" = 'Berlin'
```

Для выбора только клиентов, не проживающих в заданных трех городах, можно использовать следующий код:

о о о

```
SELECT *
FROM public.customers
WHERE "City" <> 'London' AND "City" <> 'Madrid' AND "City" <> 'Berlin'
```

Этот запрос хорошо иллюстрирует взаимозаменяемость различных способов отбора данных. В данном случае – мы рассмотрели два способа создания одного и того же запроса: при помощи проверки условия наличия значения в списке и при помощи последовательной проверки выполнения нескольких условия равенства / неравенства и логических операторов OR и AND

Для данного конкретного примера наиболее подходящим представляется вариант с использованием списка. Он позволяет получить тот же результат с использованием меньшего объёма кода

Часто на практике встречаются задачи, где нужно отобрать данные по дате за требуемый период. Рассмотрим примеры фильтрации по дате, а также использования функций для извлечения отдельных атрибутов из даты.

05

Из таблицы orders выбрать только строки с датой заказа в 1997 году

о о о

```
SELECT *
FROM Orders
WHERE EXTRACT(YEAR FROM "OrderDate") = 1997
```

06

Из таблицы orders выбрать только строки с датой заказа в 1997-1999 годах и отсортировать по столбцу RequiredDate от самого позднего к самому раннему

Для этого мы можем использовать ещё одну конструкцию, которую стоит запомнить: это конструкция BETWEEN ... AND ...

о о о

```
SELECT *
FROM Orders
WHERE EXTRACT(YEAR FROM "OrderDate") BETWEEN 1997 AND 1999
ORDER BY "RequiredDate" DESC
```

I ИТОГ

⌚ Вы научились применять различные способы фильтрации таблиц данных в SQL-запросе при помощи инструкции WHERE и проверки условий различными методами:

- Проверка равенства/неравенство
- Оператор LIKE с использованием паттерна текста
- Проверка наличия либо отсутствия элемента в списке при помощи оператора IN либо NOT IN

⌚ Также мы познакомились со вспомогательными функциями языка SQL, позволяющими осуществить выборку с фильтрацией по датам или отдельным элементам дат

