

Основы баз данных

Алексей Кузьмин

Директор разработки, Data Scientist «ДомКлик»



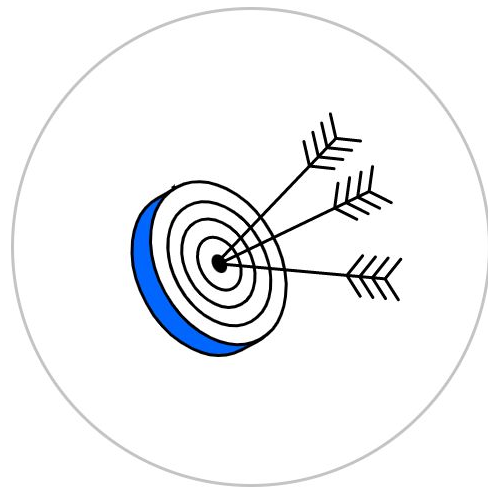
Алексей Кузьмин

Директор разработки,
Data Scientist в «ДомКлик»



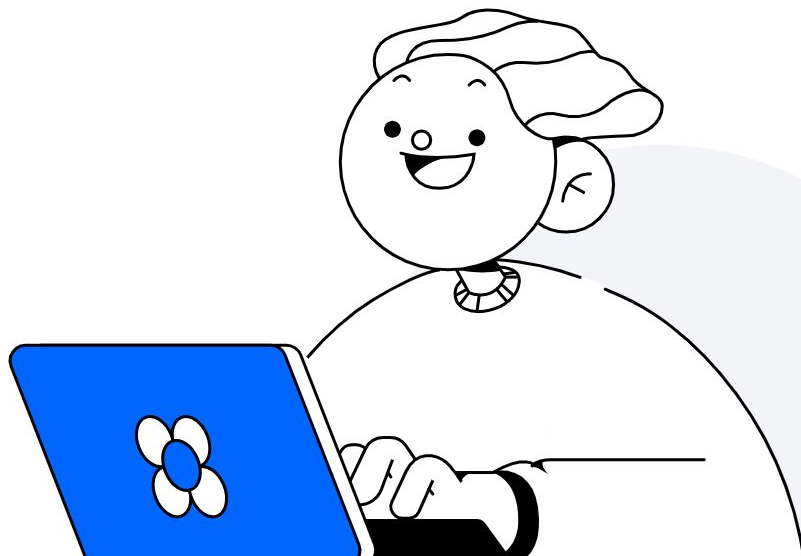
Цели занятия

- Разобраться, как устроены запросы и результаты запросов в БД
- Познакомиться с основными типами данных
- Узнать об операциях с различными типами данных
- Выяснить, как выполняется фильтрация данных в строках

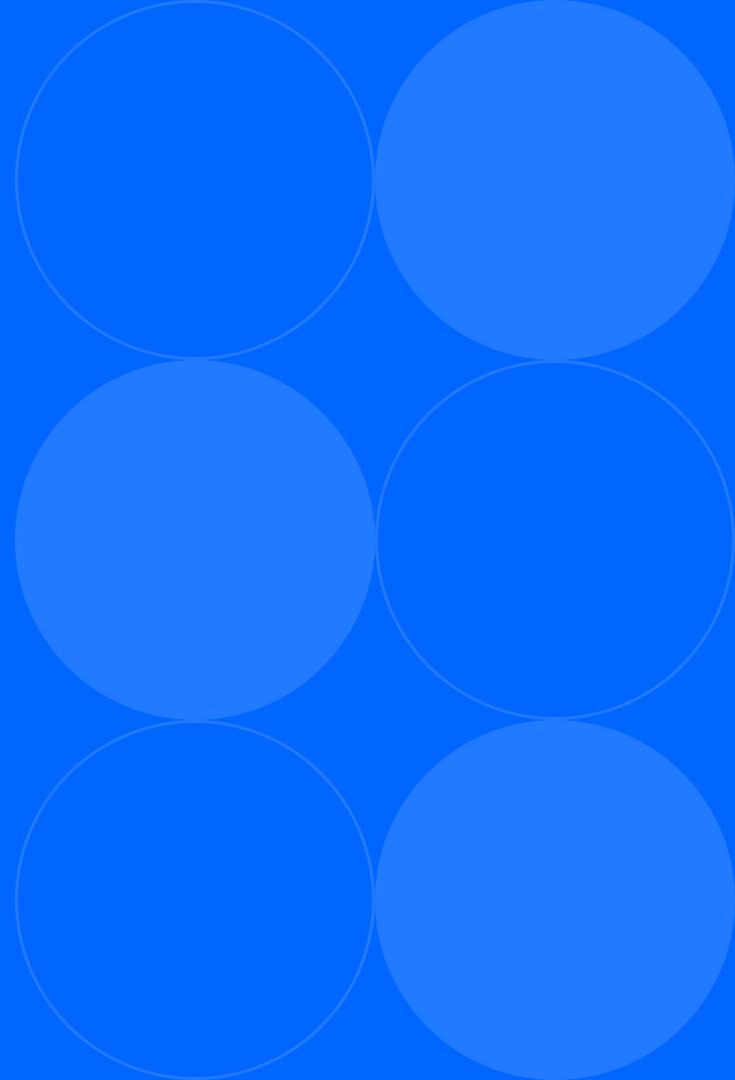


План занятия

- 1 Запросы
- 2 Основные типы данных
- 3 Работа с разными типами данных
- 4 Фильтрация

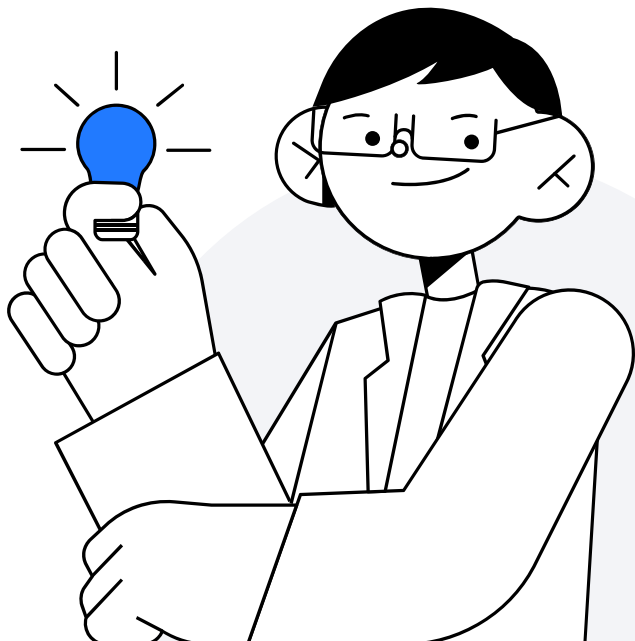


Запросы



Вы узнаете

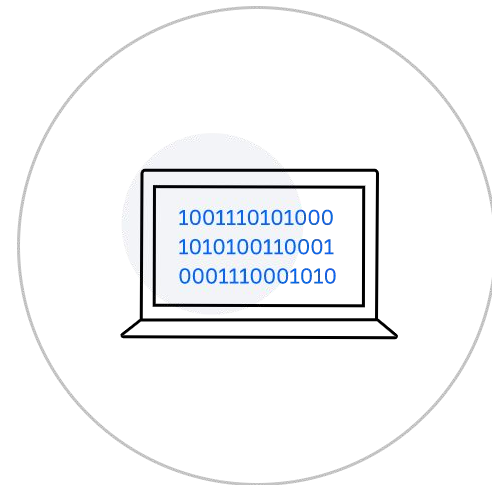
- 1 Как с помощью языка запросов SQL получить из БД нужные данные
- 2 Какие варианты запросов есть в PostgreSQL
- 3 Как БД представляет результат запроса



Как получить данные из БД

Чтобы извлечь данные из БД, применяют команду SELECT. В упрощённом виде она имеет следующий синтаксис:

```
SELECT список_столбцов FROM имя_таблицы;
```

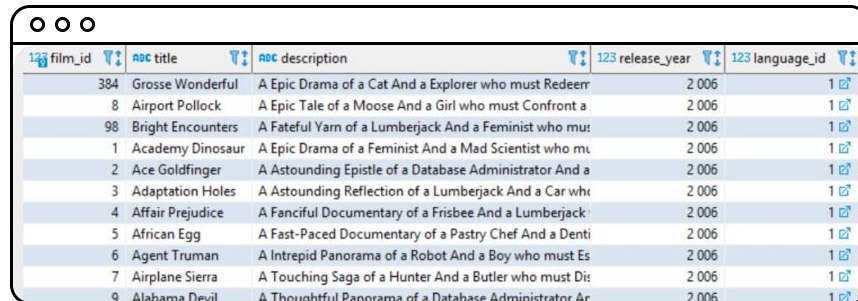


Получение данных по всем столбцам

Чтобы получить все объекты из таблицы film базы dvd-rental, используют команду:

```
SELECT * FROM film;
```

Символ * указывает, что надо получить все столбцы



film_id	title	description	release_year	language_id
384	Grosse Wonderful	A Epic Drama of a Cat And a Explorer who must Redeem	2 006	1
8	Airport Pollock	A Epic Tale of a Moose And a Girl who must Confront a	2 006	1
98	Bright Encounters	A Fateful Yarn of a Lumberjack And a Feminist who mus	2 006	1
1	Academy Dinosaur	A Epic Drama of a Feminist And a Mad Scientist who mu	2 006	1
2	Ace Goldfinger	A Astounding Epistle of a Database Administrator And a	2 006	1
3	Adaptation Holes	A Astounding Reflection of a Lumberjack And a Car wh	2 006	1
4	Affair Prejudice	A Fanciful Documentary of a Frisbee And a Lumberjack	2 006	1
5	African Egg	A Fast-Paced Documentary of a Pastry Chef and a Denti	2 006	1
6	Agent Truman	A Intrepid Panorama of a Robot And a Boy who must Es	2 006	1
7	Airplane Sierra	A Touching Saga of a Hunter And a Butler who must Dis	2 006	1
9	Alabama Devil	A Thoughtful Panorama of a Database Administrator Ar	2 006	1

Получение данных по конкретным столбцам

Использование символа * считается не очень хорошей практикой, потому что не все столбцы бывают нужны.

Если надо получить данные не по всем, а по каким-то конкретным столбцам, все эти спецификации столбцов перечисляют через запятую после SELECT:

```
SELECT title, description FROM film;
```

Исключения:

- ⚡ нужно получить данные по абсолютно всем столбцам таблицы
- ⚡ ситуация, когда точно не известны названия столбцов

Использование выражений в запросах

Спецификация столбца не обязательно должна представлять его название. Это может быть любое выражение, например, результат арифметической операции:

```
SELECT title, description, rental_rate / rental_duration FROM film;
```

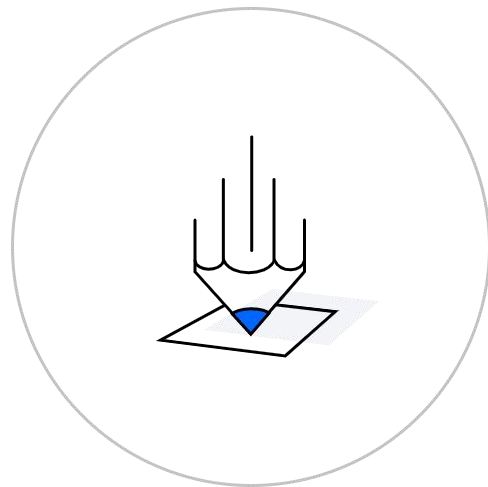
Здесь при выборке будут создаваться три столбца.

Третий столбец представляет значение столбца `rental_rate`, разделённое на значение столбца `rental_duration`, то есть стоимость аренды за день

Название колонок в выдаче БД

С помощью оператора **AS** можно изменить название выходного столбца или определить его псевдоним:

```
SELECT title AS "Название фильма", description,  
rental_rate / rental_duration AS cost_per_day  
FROM film;
```



Использование кавычек в запросах

- Двойные кавычки предназначены для имён таблиц, полей или псевдонимов
Брать названия на английском языке в двойные кавычки не обязательно, но это необходимо, если нужно соблюдать регистр и если используются другие языки или пробелы
- Одинарные кавычки предназначены для строковых констант

```
SELECT "first_name" AS "Имя пользователя"  
FROM "customer"  
WHERE first_name = 'Иван Иванович';
```



Запросы данных из разных схем

Схема — это способ логически разделять таблицы в рамках одной БД.

→ **Пример:** БД по продажам на сайте разделена на схемы — продажи через сайт и продажи через приложение.

Если нужно в одном запросе получать данные из разных схем или таблиц, необходимо явно указывать, откуда будут взяты данные:

```
SELECT название_таблицы.название_столбца  
FROM название_схемы.название_таблицы
```

Сокращение надписей в запросах

С помощью оператора AS также можно задавать псевдонимы названия таблиц для сокращения надписей:

```
SELECT a.col_A, b.col_B  
FROM schema_A.table_A AS a  
JOIN schema_B.table_B AS b ON условие_соединения
```

Выводы

1

Шаблон запроса на получение данных состоит из обязательных элементов SELECT и FROM, между ними указывают столбцы, после FROM — имя таблицы

2

Данные из БД можно получать так:

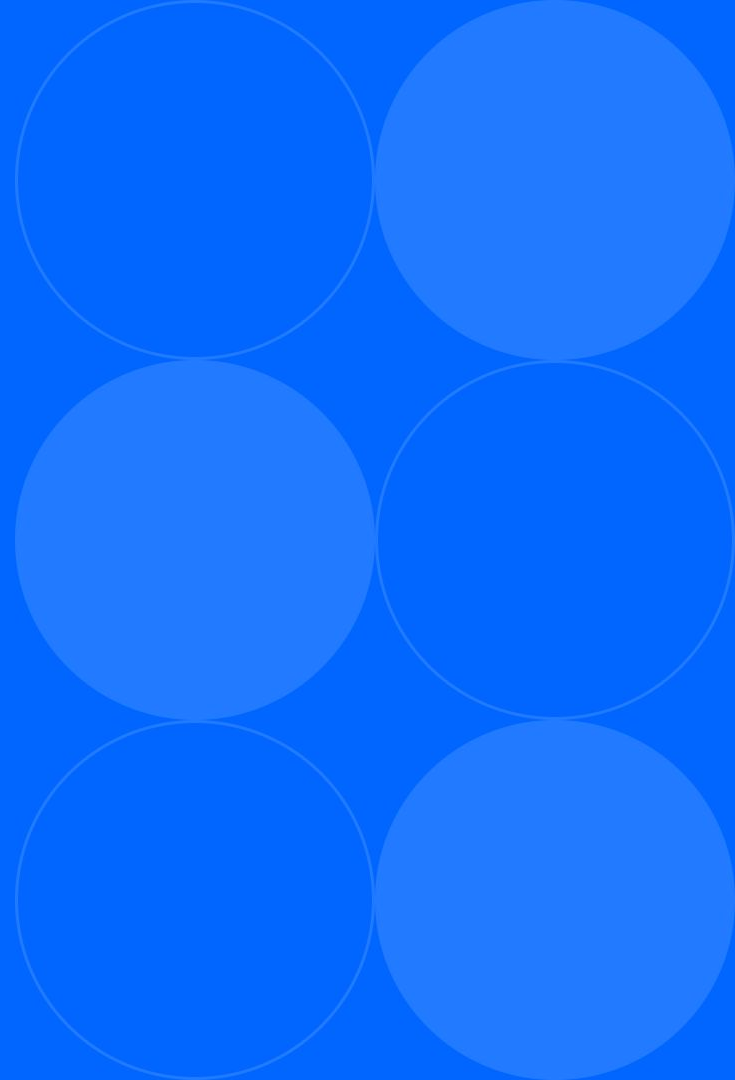
- используя символ *, чтобы получить все данные
- чтобы получить данные из конкретных столбцов, нужно перечислить их через запятую после SELECT
- используя выражения, если необходимо сделать вычисление

3

БД представляет результат запроса в виде таблицы. Чтобы управлять названием колонок и других сущностей в БД, используют оператор AS

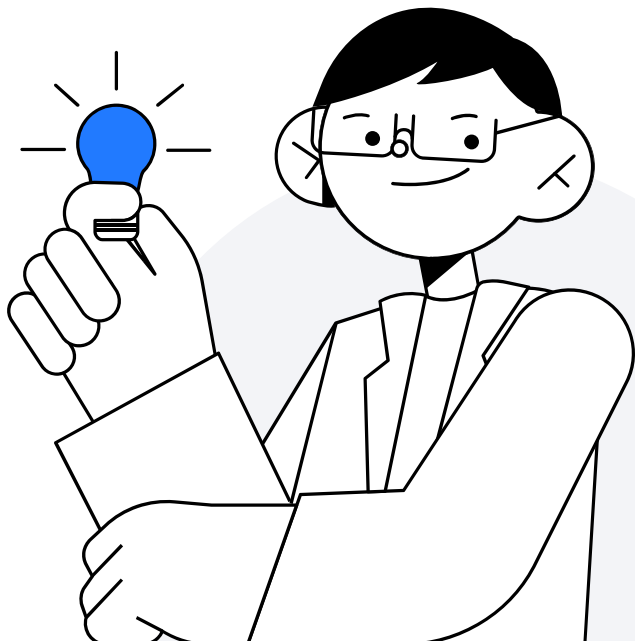


Основные типы данных



Вы узнаете

- 1 Какие типы данных бывают в PostgreSQL
- 2 Как преобразовать данные



Числовые типы данных



integer

Хранит числа от -2147483648 до +2147483647.
Занимает 4 байта. Имеет псевдонимы int и int4



bigint

Хранит числа от -9223372036854775808 до +9223372036854775807.
Занимает 8 байт. Имеет псевдоним int8



real

Хранит числа с плавающей точкой из диапазона от $1E-37$ до $1E+37$.
Занимает 4 байта. Имеет псевдоним float4



double precision

Хранит числа с плавающей точкой из диапазона от $1E-307$ до $1E+308$.
Занимает 8 байт. Имеет псевдоним float8

Символьные типы



character(n)

Строка из фиксированного количества символов.
С помощью параметра задаётся количество символов в строке. Имеет псевдоним `char(n)`



character varying(n)

Строка из нефиксированного количества символов.
С помощью параметра задаётся максимальное количество символов в строке. Имеет псевдоним `varchar(n)`



text

Представляет текст произвольной длины

Типы для работы с датами и временем

→ timestamp

Хранит дату и время. Занимает 8 байт. Для дат нижнее значение — 4713 г. до н. э., верхнее значение — 294276 г. н. э.

→ date

Представляет дату от 4 713 г. до н. э. до 5 874 897 г. н. э. Занимает 4 байта

→ time with time zone

Хранит время с точностью до 1 микросекунды с указанием часового пояса. Принимает значения от 00:00:00 + 1459 до 24:00:00 – 1459. Занимает 12 байт

→ timestamp with time zone

То же самое, что и timestamp, только добавляет данные о часовом поясе

→ time

Хранит время с точностью до 1 микросекунды без указания часового пояса. Принимает значения от 00:00:00 до 24:00:00. Занимает 8 байт

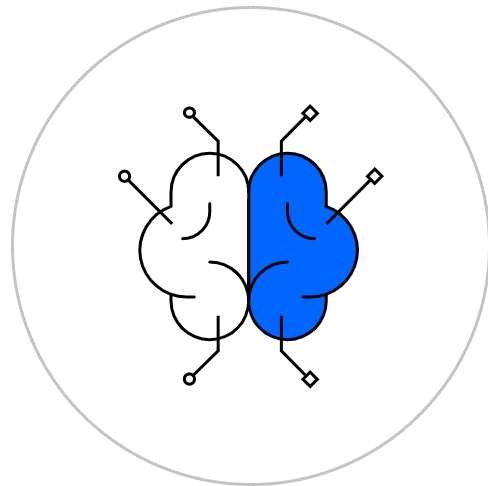
→ interval

Представляет временной интервал. Занимает 16 байт

Логический тип

Тип **boolean** описывает результаты операций сравнения:

- может принимать одно из двух значений:
true или false
- вместо true можно указывать следующие значения:
TRUE, 't', 'true', 'y', 'yes', 'on', '1'
- вместо false можно указывать следующие значения:
FALSE, 'f', 'false', 'n', 'no', 'off', '0'



Специальные типы данных



json

хранит данные JSON в текстовом виде



jsonb

хранит данные JSON в бинарном формате



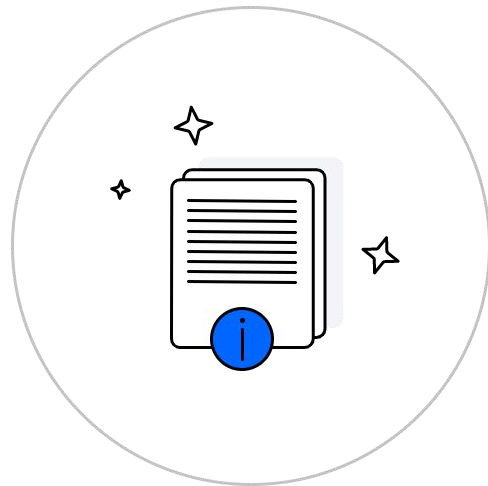
xml

хранит данные в формате XML



массивы

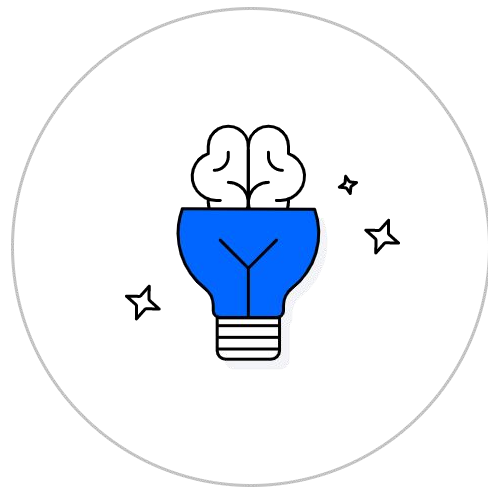
позволяют хранить несколько различных значений в рамках одной ячейки



Специальный тип данных NULL

Означает, что значение отсутствует

- Для сравнения с **NULL** используют конструкцию:
WHERE column **IS NULL**
- Чтобы указать, что значение не является **NULL**,
используют конструкцию: **WHERE** column **IS NOT NULL**



Операции над типами данных

1

Математические

2

Строковые

3

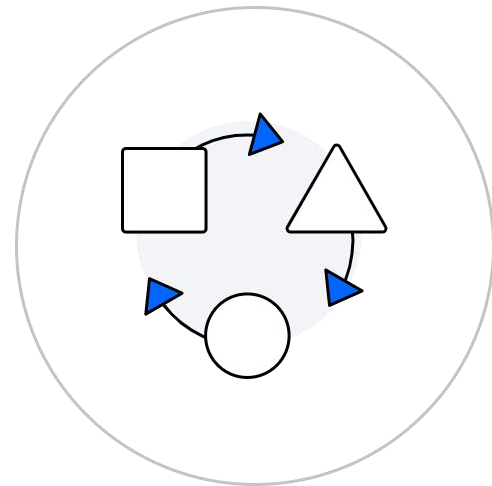
Логические

Преобразование типов данных

Чтобы преобразовать один тип данных к другому, используют операторы **CAST** или **::**

```
SELECT CAST(100 AS text) — преобразует число к тексту
```

```
SELECT '01/01/2021'::date — преобразует строку к дате
```



Выводы

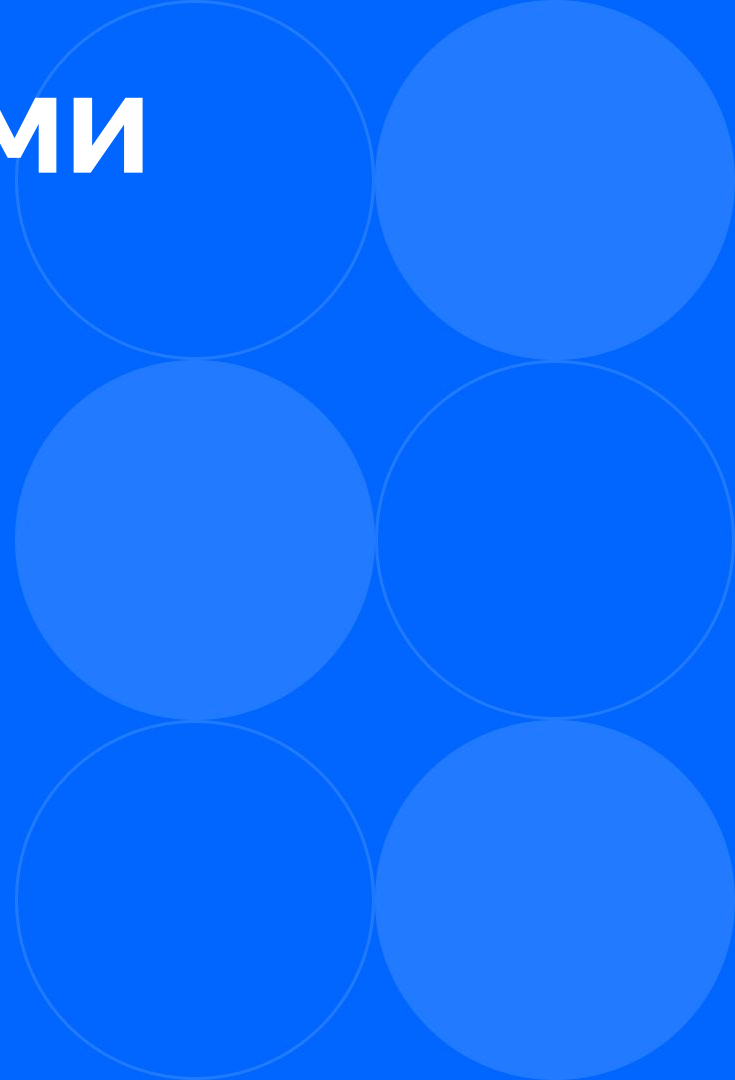
1 В PostgreSQL есть множество типов данных, основные из них:

- числовые
- символьные
- типы для работы с датами и временем
- логический тип
- специальные типы

2 Чтобы преобразовать один тип данных к другому, используют операторы **CAST** или **::**

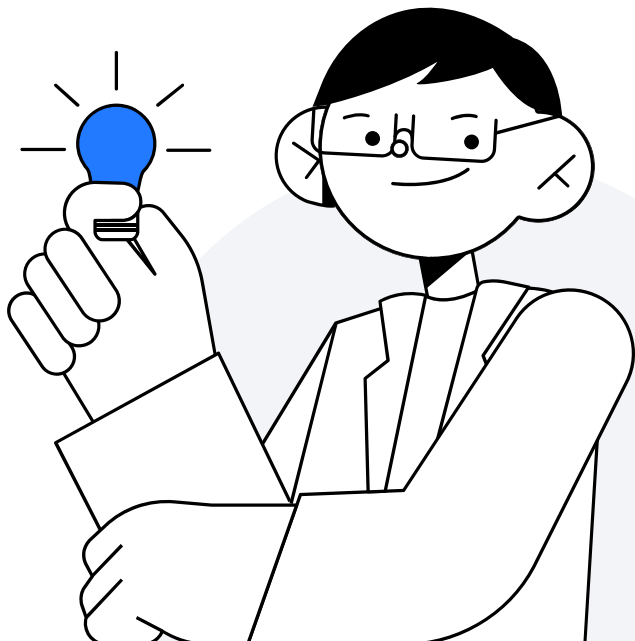


Работа с разными типами данных



Вы узнаете

- 1 Какие операции возможны в работе со строками
- 2 Какие функции работы с датами бывают
- 3 Как создавать более сложные запросы



Работа со строками. LIKE

- Выражение **LIKE** возвращает true, если строка соответствует заданному шаблону.
- Выражение **NOT LIKE** возвращает false, когда **LIKE** возвращает true, и наоборот.
- Если шаблон не содержит знаков процента и подчёркиваний, он в точности представляет строку, и **LIKE** работает как оператор сравнения.
- Подчёркивание () в шаблоне заменяет любой символ.
- Знак процента (%) заменяет любую (в том числе и пустую) последовательность символов:

```
'abc' LIKE 'abc' — true  
'abc' LIKE 'a%' — true  
'abc' LIKE '_b_' — true  
'abc' LIKE '_c_' — false
```

Работа со строками. LIKE/ILIKE

→ **LIKE** — регистрозависимый поиск

→ **ILIKE** — регистронеزависимый поиск

```
'abc' LIKE 'abc' — true
```

```
'abc' LIKE 'a%' — true
```

```
'abc' LIKE '_b_' — true
```

```
'abc' LIKE 'c' — false
```

```
'abc' ILIKE 'ABC' — true
```

```
'aBC' ILIKE '%a%' — true
```

```
'abc' ILIKE '_b' — false
```

Работа со строками. Функции

Возвращает положение указанной подстроки:

```
SELECT STRPOS('Hello world!', 'world') – 7  
SELECT POSITION('world' IN 'Hello world!') – 7
```

Возвращает длину строки:

```
SELECT CHARACTER_LENGTH('Hello world!') – 12
```

Заменяет подстроку:

```
SELECT OVERLAY('Hello world!' PLACING 'Max' FROM 7 FOR 5) – Hello Max!
```

Извлекает подстроку:

```
SELECT SUBSTRING('Hello world!' FROM 7 FOR 5) – world
```

Работа со строками. Конкатенация

Используется, чтобы соединить несколько подстрок в одну. Можно использовать операторы **CONCAT** или **||**, которые отличаются работой со значением **NULL**:

```
SELECT 'Hello' || ' ' || 'world!' – Hello world!
SELECT 'Hello' || ' ' || NULL – null

SELECT CONCAT('Hello', ' ', 'world!') – Hello world!
SELECT CONCAT('Hello', ' ', NULL) – Hello
```


Функции для работы с датами

Чтобы проверить, входит ли дата в какой-то промежуток диапазона, используют операторы сравнения или оператор **BETWEEN**.

Два запроса ниже идентичны:

```
SELECT last_name, first_name, create_date
FROM customer
WHERE create_date BETWEEN '01-02-2006' AND '01-03-2006';
```

```
SELECT last_name, first_name, create_date
FROM customer
WHERE create_date >= '01-02-2006' AND create_date <= '01-03-2006';
```

Работа с датами. Дополнительные функции

Получить год:

```
SELECT EXTRACT(YEAR FROM '28.02.2020'::DATE) — 2020
```

Получить месяц:

```
SELECT EXTRACT(MONTH FROM '28.02.2020'::DATE) — 2
```

Получить день:

```
SELECT EXTRACT(DAY FROM '28.02.2020'::DATE)  
— 28
```

Получить месяц с учётом года:

```
SELECT DATE_TRUNC('MONTH', '28.02.2020'::DATE)  
— 2020-02-01 00:00:00
```

Получить день с учётом месяца и года:

```
SELECT DATE_TRUNC('DAY', '28.02.2020'::DATE)  
— 2020-02-28 00:00:00
```

Получить текущие дата и время:

```
SELECT NOW() — текущие дата и время
```

Сортировка данных

Оператор **ORDER BY** позволяет отсортировать значения по определённому столбцу. Упорядочим выборку из таблицы film по столбцу title:

```
SELECT * FROM film  
ORDER BY title;
```

По умолчанию данные сортируются по возрастанию, однако с помощью оператора **DESC** можно задать сортировку по убыванию. Явно задать сортировку по возрастанию можно, указав оператор **ASC**:

```
SELECT * FROM film  
ORDER BY title DESC;
```

Выборка уникальных значений

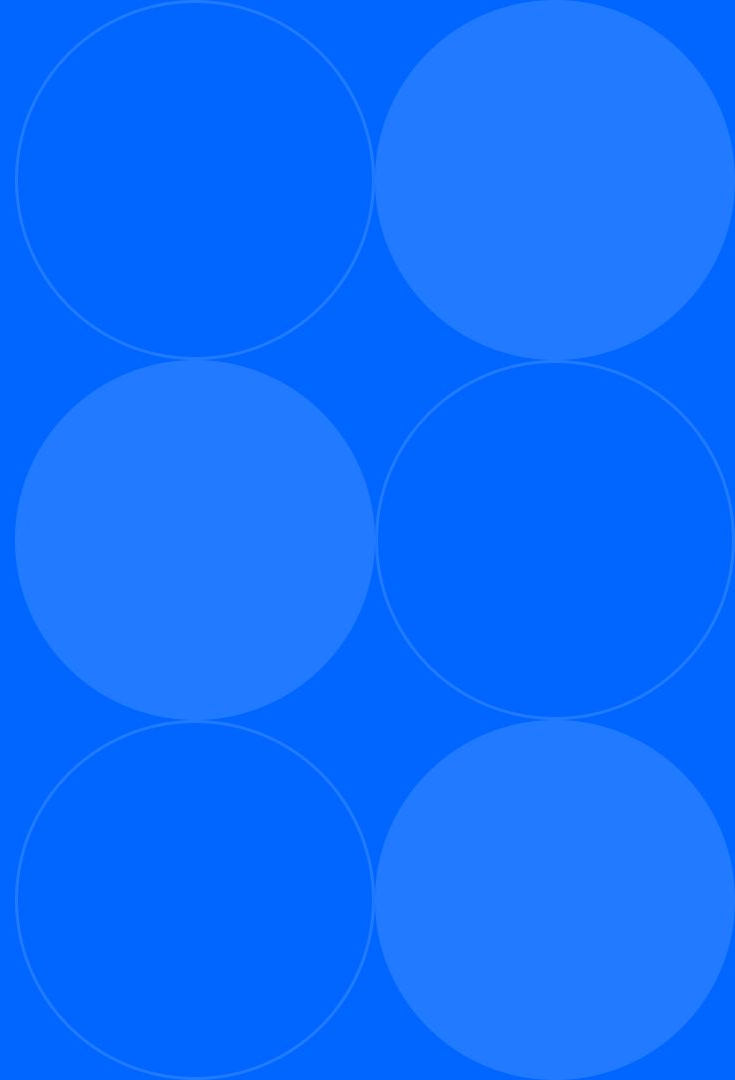
Оператор **DISTINCT** позволяет выбрать уникальные данные по определённым столбцам:

```
SELECT DISTINCT title FROM film;
```

Выводы

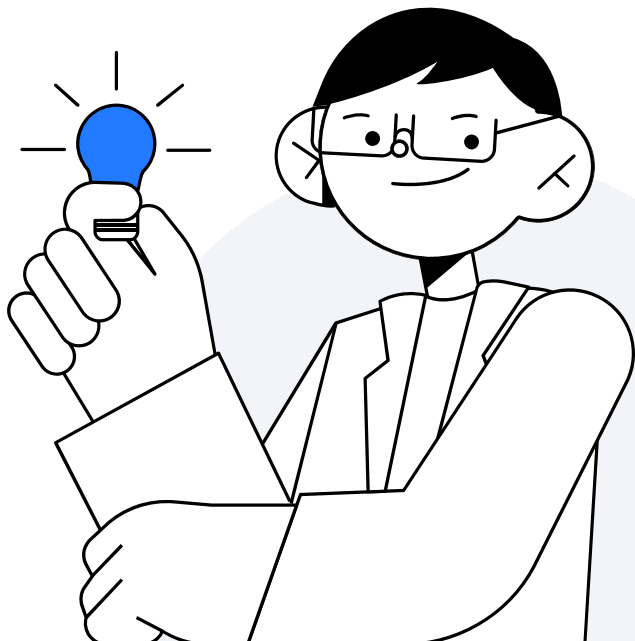
- 1 В работе со строками используются:
 - операторы LIKE и ILIKE для проверки соответствия строки шаблону. Их отличие в том, что LIKE требует соблюдения регистра букв
 - функции для работы с подстроками и длиной строки
 - конкатенация для соединения нескольких подстрок в одну
- 2 Для работы с датами используются:
 - оператор BETWEEN и операторы сравнения, чтобы проверить, входит ли дата в какой-то промежуток диапазона
 - функции EXTRACT и DATE_TRUNC, позволяющие получить год, месяц, день
- 3 Более сложные запросы:
 - оператор ORDER BY отсортирует значения по определённому столбцу
 - оператор DISTINCT выберет уникальные данные по определённым столбцам

Фильтрация



Вы узнаете

- 1 Как использовать фильтрацию данных в строках
- 2 Как группировать условия в запросах



Фильтрация

Для фильтрации данных применяют оператор **WHERE**, после него указывают условие, на основании которого производится фильтрация.

Если условие истинно, строка попадает в результирующую выборку.

В качестве условия можно использовать операции сравнения. Они сравнивают два выражения



Операции сравнения в PostgreSQL

- = сравнение на равенство
- <> или != сравнение на неравенство
- < меньше чем
- > больше чем
- !< не меньше чем
- !> не больше чем
- <= меньше чем или равно
- >= больше чем или равно

Пример:

найдем информацию о фильме с названием Ace Goldfinger. Для этого используем запрос:

```
SELECT * FROM film
WHERE title = 'Ace Goldfinger';
```

Группировка условий

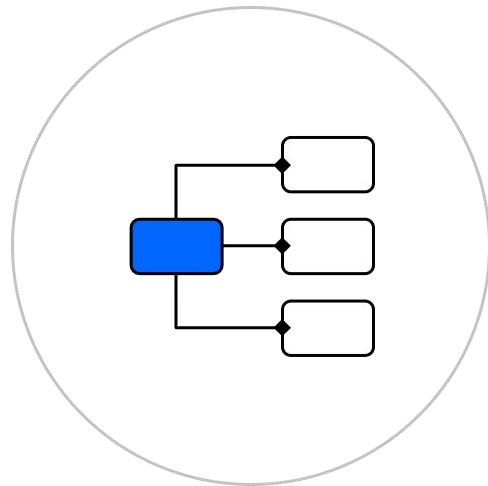
Чтобы объединить несколько условий в одно, в PostgreSQL можно использовать логические операторы:

- **AND** — операция логического И
Она объединяет два выражения. Только если оба этих выражения одновременно истинны, общее условие оператора AND также будет истинно. То есть если и первое условие истинно, и второе
- **OR** — операция логического ИЛИ
Она также объединяет два выражения. Если хотя бы одно из этих выражений истинно, общее условие оператора OR тоже будет истинно. То есть если или первое условие истинно, или второе
- **NOT** — операция логического отрицания
Если выражение в этой операции ложно, общее условие истинно

Пример группировки условий

Выберем все фильмы, у которых продолжительность более 100 минут и стоимость аренды равна 4,99:

```
SELECT * FROM film  
WHERE length > 100 AND rental_rate = 4.99;
```



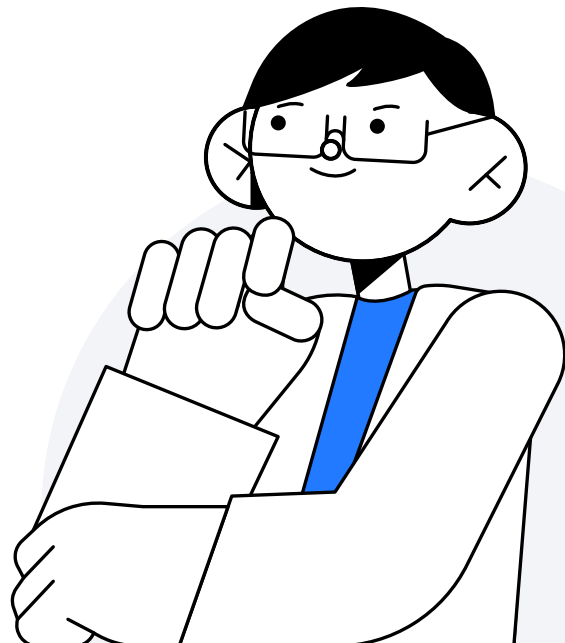
Выводы

- 1 Для фильтрации данных в строках применяют оператор WHERE и указывают условие, на основании которого производится фильтрация
- 2 Чтобы объединить несколько условий в одно, в PostgreSQL можно использовать логические операторы AND, OR, NOT



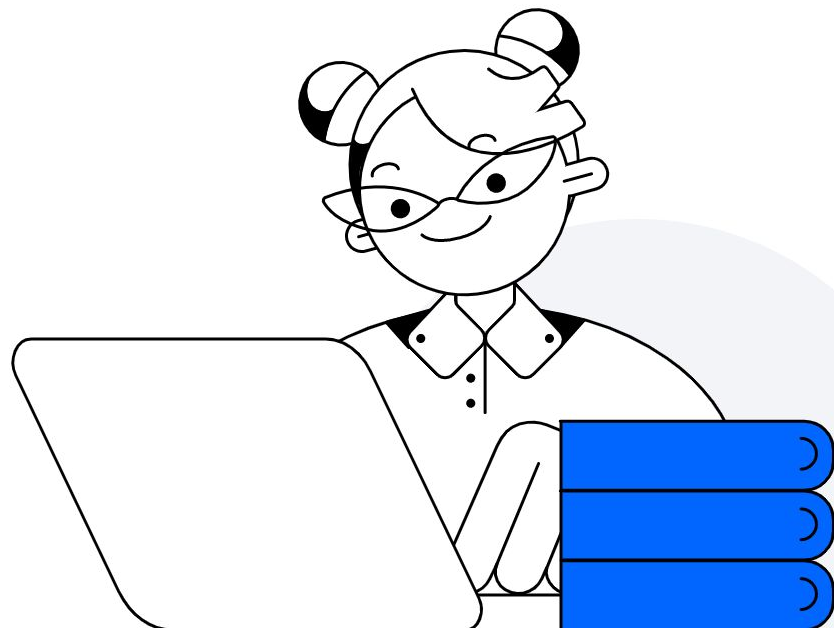
Итоги

- 1 Разобрались, как устроены запросы и результаты запросов в БД
- 2 Познакомились с основными типами данных
- 3 Узнали об операциях с различными типами данных
- 4 Выяснили, как выполняется фильтрация данных в строках



Дополнительные материалы

- [Документация PostgreSQL](#). Типы данных (англ.)
- [Документация PostgreSQL](#). Функции и операторы (англ.)
- [Описание базы данных](#)



Основы баз данных

Алексей Кузьмин

Директор разработки, Data Scientist «ДомКлик»

