Работа с базами данных

Занятие 1.2

Алексей Кузьмин
Директор разработки, Data Scientist ДомКлик

Алексей Кузьмин

Директор разработки, Data Scientist
ДомКлик

О чём поговорим и что сделаем

1. Запросы
2. Основные типы данных
3. Работа с разными типами данных
4. Усложняем выборку данных
5. Фильтрация

Запросы

1

Для извлечения данных из БД применяется команда SELECT.
В упрощённом виде она имеет следующий синтаксис:

```
SELECT список_столбцов FROM имя_таблицы;
```

Рассмотрим получение данных из таблицы film базы dvd-rental.

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 who	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 An	2 006	1

Получить все объекты из этой таблицы можно командой

```
SELECT * FROM film;
```

Символ * указывает, что нам надо получить все столбцы.

Однако использование символа звёздочки считается не очень хорошей практикой, потому что не все столбцы бывают нужны. И более оптимальный подход заключается в указании всех необходимых столбцов после слова SELECT.

Исключение составляет случай, когда надо получить данные по абсолютно всем столбцам таблицы. Также использование символа может быть предпочтительно в ситуациях, когда в точности не известны названия столбцов.

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

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

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

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

Здесь при выборке будут создаваться три столбца. Причём третий столбец представляет значение столбца rental_rate, разделённое на значение столбца rental_duration, то есть стоимость аренды за день

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;
```

Двойные кавычки предназначены для имён таблиц, полей или псевдонимов, при этом брать названия в двойные кавычки не обязательно. Но это необходимо для соблюдения регистра или кириллицы.

Одинарные кавычки предназначены для строковых констант

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

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

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

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

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

Время практики

Практика 1. Работаем с базой dvdrental

- Получите из таблицы film атрибуты: id фильма, название, описание, год релиза. Переименуйте поля так, чтобы все они начинались со слова Film (FilmTitle вместо title и т. п.)

- В таблице dvdrental есть два атрибута: **rental_duration** — длина периода аренды в днях и **rental_rate** — стоимость аренды фильма на этот промежуток времени. Для каждого фильма из таблицы film получите стоимость его аренды в день

Практика 1. Решение

```
SELECT film_id AS "Filmfilm_id", title AS "Filmtitle",
description AS "Filmdescriprion", release_year AS "Filmrelease_year"
FROM film
```

```
SELECT title, rental_rate/rental_duration AS cost_per_day
FROM film
```

Основные типы данных

2

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): представляет строку из фиксированного количества символов. С помощью параметра задаётся количество символов в строке. Имеет псевдоним char(n)

- character varying(n): представляет строку 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 самое нижнее значение — 4713 г. до н. э., самое верхнее значение — 294276 г. н. э.
- timestamp with time zone: то же самое, что и timestamp, только добавляет данные о часовом поясе
- date: представляет дату от 4713 г. до н. э. до 5874897 г. н. э. Занимает 4 байта
- 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
- 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
- interval: представляет временной интервал. Занимает 16 байт

Логический тип

- Тип boolean может хранить 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

Операции над типами данных

- Математические https://www.postgresql.org/docs/9.2/functions-math.html
- Строковые https://www.postgresql.org/docs/9.2/functions-string.html
- Логические https://www.postgresql.org/docs/9.2/functions-logical.html

NULL

- **NULL** — это значение, которое означает, что значение отсутствует

- Для сравнения с **NULL** используется конструкция:
  WHERE column **IS NULL**

- Для сравнения, что значение не является **NULL**, используется конструкция:
  WHERE column **IS NOT NULL**

Для преобразования одного типа данных к другому используются операторы **CAST** или **::**

```
SELECT CAST(100 AS text) — преобразует число к тексту
SELECT '01/01/2021'::date — преобразует строку к дате
```

Работа с разными типами данных

3

Работа со строками. LIKE / ILIKE

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, тогда шаблон представляет в точности строку и **LIKE** работает как оператор сравнения.
Подчёркивание (_) в **шаблоне** подменяет (вместо него подходит) любой символ.
Знак процента (**%**) подменяет 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 — регистронезависимый 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

Чтобы соединить несколько подстрок в одну используется конкатенация. Для этого используются 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
```

Работа с датами

Для того, чтобы сравнить дату между двумя датами, используются 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. 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

Усложняем выборку данных

4

Сортировка

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 ;
```

Фильтрация

5

Фильтрация

Для фильтрации данных применяется оператор **WHERE**, после которого указывается условие, на основании которого производится фильтрация. Если условие истинно, то строка попадает в результирующую выборку.
В качестве можно использовать операции сравнения. Эти операции сравнивают два выражения. В PostgreSQL можно применять следующие операции сравнения:

- = 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

Найдём информацию по фильму с названием Ace Goldfinger:

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

Чтобы объединить нескольких условий в одно, в PostgreSQL можно использовать логические операторы

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

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;
```

Время практики

Практика 2

- Отсортировать список фильмов по убыванию стоимости за день аренды (второе задание первой практики)
- Вывести все уникальные годы выпуска фильмов
- Вывести весь список фильмов имеющий рейтинг 'PG-13'

Практика 2. Решения

```
SELECT title, rental_rate/rental_duration AS cost_per_day
FROM film
ORDER BY cost_per_day DESC
```

```
SELECT DISTINCT release_year
FROM film
```

```
SELECT CONCAT (title, ' - ', release_year)
FROM film
WHERE rating = 'PG-13'
```

Полезные материалы

Полезные материалы

- https://github.com/ustu/lectures.db/blob/master/docs/databases/rdb.rst
- https://www.postgresql.org/docs/12/datatype.html
- https://www.postgresql.org/docs/12/functions.html
- Описание базы данных: https://dev.mysql.com/doc/sakila/en/sakila-structure-tables.html

Спасибо за внимание!

Алексей Кузьмин
Директор разработки, Data Scientist ДомКлик