

Базы данных

Лекция 3. Основы SQL. DML.

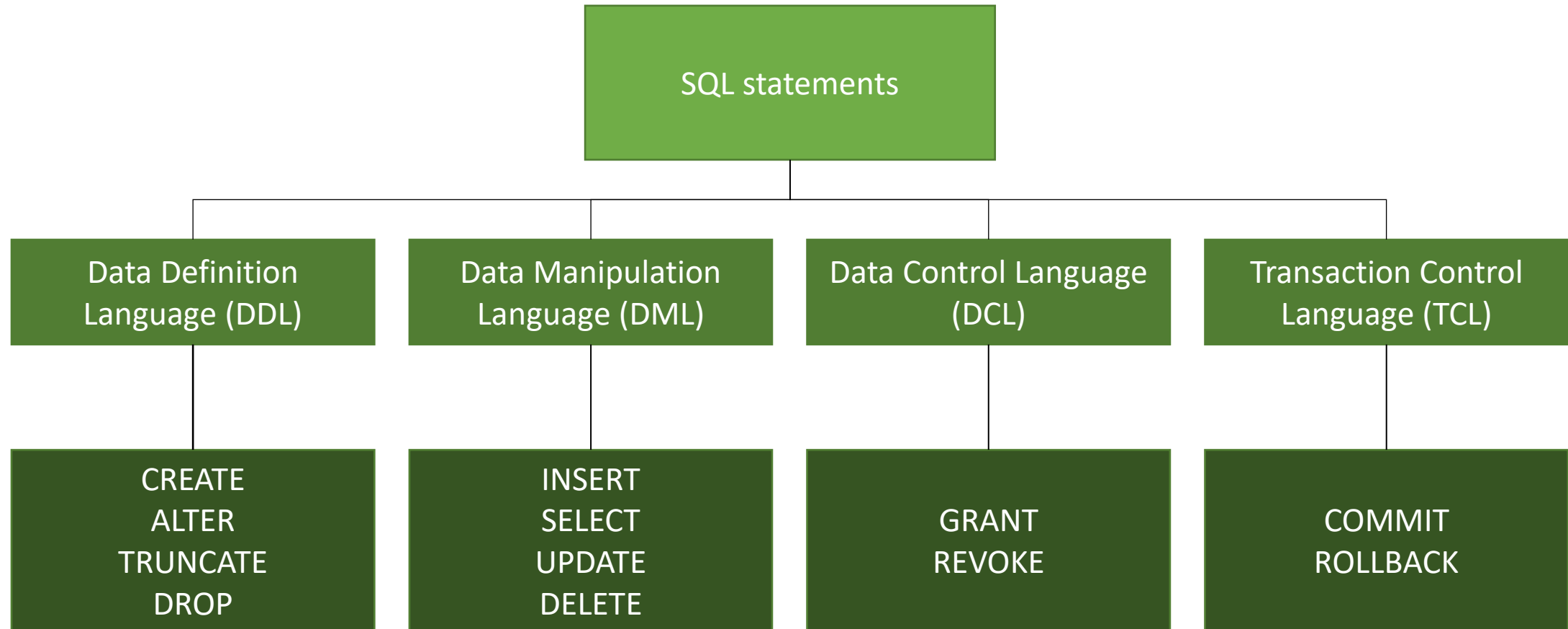
МФТИ, 2024

Игорь Шевченко

@igorshvch

I. SQL - DML

Группы операторов SQL



Команды DML

- ▶ INSERT
- ▶ UPDATE
- ▶ DELETE
- ▶ SELECT

Что читать?

- Официальная англоязычная документация:
 - [PostgreSQL: Documentation: 16: Chapter 6. Data Manipulation](#)
 - [PostgreSQL: Documentation: 16: Chapter 7. Queries](#)
 - [PostgreSQL: Documentation: 16: Chapter 9. Functions and Operators](#)
 - [PostgreSQL: Documentation: 16: INSERT](#)
 - [PostgreSQL: Documentation: 16: SELECT](#)
 - [PostgreSQL: Documentation: 16: UPDATE](#)
 - [PostgreSQL: Documentation: 16: DELETE](#)
- Официальная русскоязычная документация:
 - [PostgreSQL : Документация: 16: Глава 6. Модификация данных : Компания Postgres Professional](#)
 - [PostgreSQL : Документация: 16: Глава 7. Запросы : Компания Postgres Professional](#)
 - [PostgreSQL : Документация: 16: Глава 9. Функции и операторы : Компания Postgres Professional](#)
 - [PostgreSQL : Документация: 16: INSERT : Компания Postgres Professional](#)
 - [PostgreSQL : Документация: 16: SELECT : Компания Postgres Professional](#)
 - [PostgreSQL : Документация: 16: UPDATE : Компания Postgres Professional](#)
 - [PostgreSQL : Документация: 16: DELETE : Компания Postgres Professional](#)

INSERT: базовый синтаксис

```
INSERT INTO имя_таблицы [ ( имя_столбца [, ...] ) ]  
    { DEFAULT VALUES | VALUES ( { выражение | DEFAULT } [, ...] ) [, ...] |  
запрос }  
    [ RETURNING * | выражение_результата [ [ AS ] имя_результата ] [,  
...]
```

INSERT: элементы синтаксиса

- **имя_таблицы** - имя существующей таблицы (возможно, дополненное схемой)
- **имя_столбца** - это имя столбца при необходимости может быть дополнено именем вложенного поля или индексом в массиве
- **выражение** - выражение или значение, которое будет присвоено соответствующему столбцу
- **запрос** - запрос (оператор SELECT), который выдаст строки для добавления в таблицу
- **выражение_результата** - выражение, которое будет вычисляться и возвращаться командой INSERT после добавления или изменения каждой строки. В этом выражении можно использовать имена любых столбцов таблицы **имя_таблицы**. Чтобы получить все столбцы, достаточно написать *
- **имя_результата** - имя, назначаемое возвращаемому столбцу
- Имена столбцов можно располагать в произвольном порядке. Имена столбцов со значениями по умолчанию можно опустить при перечислении либо явно указать параметр DEFAULT. Если не указать список столбцов, но привести во вставку не все значения для столбцов, то вставка будет происходить последовательно во все «ячейки» строки, в связи с чем система выдаст ошибку либо из-за несовпадения типа данных, либо из-за недостаточности добавляемых в строку данных

INSERT: примеры

Для начала создадим тренировочную таблицу:

```
CREATE TABLE example_table (  
    id SERIAL PRIMARY KEY,  
    name TEXT DEFAULT 'Unknown',  
    age INTEGER DEFAULT 18,  
    email TEXT DEFAULT 'example@example.com'  
);  
SELECT * FROM example_table;
```

id	name	age	email
----	------	-----	-------

INSERT: добавление произвольных значений

Добавим значения в нашу таблицу:

- 1) INSERT INTO example_table (name) VALUES ('John');
- 2) INSERT INTO example_table (name, age) VALUES ('Alice', DEFAULT);
- 3) INSERT INTO example_table DEFAULT VALUES;
- 4) INSERT INTO example_table (name, age, email) VALUES ('Alice', 25, 'alice@example.com'), ('Bob', 30, 'bob@example.com'), ('Charlie', DEFAULT, 'charlie@example.com');
- 5) INSERT INTO example_table (age, name) VALUES (12+7+2, 'Pete');

SELECT * FROM example_table;

id	name	age	email
1	John	18	example@example.com
2	Alice	18	example@example.com
3	Unknown	18	example@example.com
4	Alice	25	alice@example.com
5	Bob	30	bob@example.com
6	Charlie	18	charlie@example.com
7	Pete	21	example@example.com

INSERT: добавление произвольных значений

Добавим значения в нашу таблицу:

1) INSERT INTO example_table DEFAULT VALUES
RETURNING *;

id	name	age	email
8	Unknown	18	example@example.com

2) INSERT INTO example_table (age) VALUES (DEFAULT),
(DEFAULT), (DEFAULT)
RETURNING *;

id	name	age	email
9	Unknown	18	example@example.com
10	Unknown	18	example@example.com
11	Unknown	18	example@example.com

INSERT: использование SELECT для вставки

- Добавим значения в нашу таблицу:

```
INSERT INTO products (product_no, name, price)
    SELECT product_no, name, price FROM new_products
    WHERE release_date = 'today';
```

- Здесь в таблицу products будут добавлены выбранные по условию значения из таблицы new_products

INSERT: использование SELECT для построчного копирования таблиц

- Скопируем таблицу example_table в таблицу example_table_2 построчно:

```
CREATE TABLE example_table_2 (  
  id SERIAL PRIMARY KEY,  
  name TEXT DEFAULT 'Unknown',  
  age INTEGER DEFAULT 18,  
  email TEXT DEFAULT 'example@example.com'  
);
```

```
INSERT INTO example_table_2 SELECT * FROM example_table;
```

id	name	age	email
1	John	18	example@example.com
2	Alice	18	example@example.com
3	Unknown	18	example@example.com
4	Alice	25	alice@example.com
5	Bob	30	bob@example.com
6	Charlie	18	charlie@example.com
7	Pete	21	example@example.com
8	Unknown	18	example@example.com
9	Unknown	18	example@example.com
10	Unknown	18	example@example.com
11	Unknown	18	example@example.com

UPDATE: базовый синтаксис

UPDATE имя_таблицы [*]

SET { имя_столбца = { выражение | DEFAULT } |

(имя_столбца [, ...]) = (вложенный_SELECT)

} [, ...]

[FROM элемент_FROM [, ...]]

[WHERE условие]

[RETURNING * | выражение_результата [[AS] имя_результата] [, ...]]

UPDATE: элементы синтаксиса

- **вложенный_SELECT** - подзапрос SELECT, выдающий столько выходных столбцов, сколько перечислено в предшествующем ему списке столбцов в скобках. При выполнении этого подзапроса должна быть получена максимум одна строка. Если он выдаёт одну строку, значения столбцов в нём присваиваются целевым столбцам; если же он не возвращает строку, целевым столбцам присваивается NULL. Этот подзапрос может обращаться к предыдущим значениям текущей изменяемой строки в таблице
- **элемент_FROM** - табличное выражение, позволяющее обращаться в условии WHERE и выражениях новых данных к столбцам других таблиц. В нём используется тот же синтаксис, что и в предложении FROM оператора SELECT
- **условие** - выражение, возвращающее значение типа boolean. Изменены будут только те строки, для которых это выражение возвращает true.

UPDATE: примеры

```
UPDATE example_table  
SET age = 25  
WHERE id = 1;
```

id	name	age	email
1	John	18	example@example.com
2	Alice	18	example@example.com
3	Unknown	18	example@example.com
4	Alice	25	alice@example.com
5	Bob	30	bob@example.com
6	Charlie	18	charlie@example.com
7	Pete	21	example@example.com
8	Unknown	18	example@example.com
9	Unknown	18	example@example.com
10	Unknown	18	example@example.com
11	Unknown	18	example@example.com


id	name	age	email
1	John	25	example@example.com
2	Alice	18	example@example.com
3	Unknown	18	example@example.com
4	Alice	25	alice@example.com
5	Bob	30	bob@example.com
6	Charlie	18	charlie@example.com
7	Pete	21	example@example.com
8	Unknown	18	example@example.com
9	Unknown	18	example@example.com
10	Unknown	18	example@example.com
11	Unknown	18	example@example.com

UPDATE: примеры

```
UPDATE example_table  
  SET (age, name, email) = (20, 'Will', 'will@mail.com')  
 WHERE id = 8;
```

id	name	age	email
1	John	18	example@example.com
2	Alice	18	example@example.com
3	Unknown	18	example@example.com
4	Alice	25	alice@example.com
5	Bob	30	bob@example.com
6	Charlie	18	charlie@example.com
7	Pete	21	example@example.com
8	Unknown	18	example@example.com
9	Unknown	18	example@example.com
10	Unknown	18	example@example.com
11	Unknown	18	example@example.com

id	name	age	email
1	John	25	example@example.com
2	Alice	18	example@example.com
3	Unknown	18	example@example.com
4	Alice	25	alice@example.com
5	Bob	30	bob@example.com
6	Charlie	18	charlie@example.com
7	Pete	21	example@example.com
8	Will	20	will@mail.com
9	Unknown	18	example@example.com
10	Unknown	18	example@example.com
11	Unknown	18	example@example.com




UPDATE: примеры

```
UPDATE example_table
  SET (age, name, email) = (
    SELECT age+3, name, email
      FROM example_table
     WHERE id = 4
  )
 WHERE id = 11;
```

id	name	age	email
1	John	18	example@example.com
2	Alice	18	example@example.com
3	Unknown	18	example@example.com
4	Alice	25	alice@example.com
5	Bob	30	bob@example.com
6	Charlie	18	charlie@example.com
7	Pete	21	example@example.com
8	Unknown	18	example@example.com
9	Unknown	18	example@example.com
10	Unknown	18	example@example.com
11	Unknown	18	example@example.com

id	name	age	email
1	John	25	example@example.com
2	Alice	18	example@example.com
3	Unknown	18	example@example.com
4	Alice	25	alice@example.com
5	Bob	30	bob@example.com
6	Charlie	18	charlie@example.com
7	Pete	21	example@example.com
8	Will	20	will@mail.com
9	Unknown	18	example@example.com
10	Unknown	18	example@example.com
11	Alice	28	alice@example.com



UPDATE: примеры

- Пример с FROM... WHERE: изменение имени контакта в таблице счетов (это должно быть имя назначенного менеджера по продажам):

```
UPDATE accounts SET
```

```
    contact_first_name = first_name,
```

```
    contact_last_name = last_name
```

```
    FROM employees
```

```
    WHERE employees.id = accounts.sales_person;
```

DELETE: базовый синтаксис

UPDATE имя_таблицы [*]

SET { имя_столбца = { выражение | DEFAULT } |

(имя_столбца [, ...]) = (вложенный_SELECT)

} [, ...]

[FROM элемент_FROM [, ...]]

[WHERE условие]

[RETURNING * | выражение_результата [[AS] имя_результата] [, ...]]

DELETE: примеры

DELETE FROM example_table_2

WHERE id = 11

RETURNING *



id	name	age	email
11	Alice	28	alice@example.com

id	name	age	email
1	John	25	example@example.com
2	Alice	18	example@example.com
3	Unknown	18	example@example.com
4	Alice	25	alice@example.com
5	Bob	30	bob@example.com
6	Charlie	18	charlie@example.com
7	Pete	21	example@example.com
8	Will	20	will@mail.com
9	Unknown	18	example@example.com
10	Unknown	18	example@example.com
11	Alice	28	alice@example.com

id	name	age	email
1	John	25	example@example.com
2	Alice	18	example@example.com
3	Unknown	18	example@example.com
4	Alice	25	alice@example.com
5	Bob	30	bob@example.com
6	Charlie	18	charlie@example.com
7	Pete	21	example@example.com
8	Will	20	will@mail.com
9	Unknown	18	example@example.com
10	Unknown	18	example@example.com

DELETE vs TRUNCATE vs DROP

DELETE	TRUNCATE	DROP
Построчное удаление строк	Удаление всех строк разом	Удаление всей таблицы
Можно задавать условия выборки удаляемого содержимого	Задать условия нельзя	Задать условия нельзя
Можно откатить	Нельзя откатить	Нельзя откатить
Физически строки не удаляются, нужно запускать команду VACUUM	Физическое удаление данных	Физическое удаление данных

III. SELECT

SELECT: базовый синтаксис

- Наиболее общий синтаксис SELECT выглядит следующим образом:

SELECT *список_выборки* **FROM** *табличное_выражение* [*определение_сортировки*]

- Спецификация команды SELECT выглядит следующим образом:

SELECT [**ALL** | **DISTINCT** [**ON** (*выражение* [, ...])]]
[***** | *выражение* [**AS** *имя_результата*] [, ...]] } ← **Список выборки**
[**FROM** *элемент_FROM* [, ...]]
[**WHERE** *условие*]
[**GROUP BY** *элемент_группирования* [, ...]]
[**HAVING** *условие*] } ← **Табличное выражение**
[{ **UNION** | **INTERSECT** | **EXCEPT** } *выборка*]
[**ORDER BY** *выражение* [**ASC** | [, ...]]] ← **Определение сортировки**
[**LIMIT** { *число* | **ALL** }]
[**OFFSET** *начало* [**ROW** | **ROWS**]]

SELECT: элементы синтаксиса (СПИСОК ВЫБОРКИ)

- SELECT возвращает набор, состоящий по крайней мере из одного значения. Для простоты на данном этапе можно считать, что SELECT всегда возвращает таблицу, в которой есть по крайней мере одна строка с по крайней мере одной ячейкой. В последнем случае также говорят, что SELECT возвращает **скалярное значение**.
- **Список выборки** задает столбцы, которые появятся в результирующем отношении. Допустимо также задать операцию над каждым значением столбца. Синтаксис ссылок на столбцы вариативен:

SELECT a, b, c FROM ...

SELECT tbl1.a, tbl2.a, tbl1.b FROM ...

SELECT tbl1.*, tbl2.a FROM ...

SELECT: элементы синтаксиса (СПИСОК ВЫБОРКИ)

- В **списке выборки** можно назначить «псевдонимы» для столбцов результирующей таблицы:

SELECT a AS value, b + c AS sum FROM ...

- Слово AS обычно можно опустить, но в некоторых случаях, когда желаемое имя столбца совпадает с ключевым словом PostgreSQL, нужно написать AS или заключить имя столбца в кавычки во избежание неоднозначности.
- **Аналогичное правило действует для таблиц, перечисляемых после предложения FROM, но:**
 - псевдоним становится новым именем таблицы в рамках текущего запроса, т. е. после назначения псевдонима использовать исходное имя таблицы в другом месте запроса нельзя
 - они бывают необходимы, когда таблица соединяется сама с собой

SELECT: элементы синтаксиса (ТАБЛИЧНОЕ ВЫРАЖЕНИЕ)

- Результат вычисления **табличного выражения** – это (виртуальная) таблица, которая также может состоять всего из одной ячейки (скаляра)
- **Табличное выражение** можно опустить и использовать SELECT для возврата результата вычисления выражения: `SELECT 3*4; SELECT 5!=4;`
- Предложение FROM образует таблицу из одной или нескольких ссылок на таблицы, разделённых запятыми:

`FROM table1 [, table_2, ..., table_N]`

- Табличной ссылкой (table_N в примере) может быть:
 - имя таблицы (возможно, с именем схемы),
 - производная таблица, например:
 - подзапрос,
 - соединение таблиц
 - сложная комбинация этих вариантов.
- Если в предложении FROM перечисляются несколько ссылок, для них применяется перекрестное соединение в виде результата декартова произведения отношений

SELECT: элементы синтаксиса: соединение таблиц (JOIN)

- Базовый синтаксис соединения таблиц выглядит следующим образом:

T1 тип_соединения T2 [условие_соединения]

- Соединения любых типов могут вкладываются друг в друга или объединяться: и T1, и T2 могут быть результатами соединения
- Типы соединений в PostgreSQL:
 - CROSS JOIN (перекрестное соединение)
 - Соединения с сопоставлением строк:
 - INNER JOIN
 - LEFT OUTER JOIN
 - RIGHT OUTER JOIN
 - FULL OUTER JOIN

SELECT: элементы синтаксиса: соединение таблиц (JOIN)

- Слова INNER и OUTER необязательны во всех формах. По умолчанию подразумевается INNER (внутреннее соединение), а при указании LEFT, RIGHT и FULL — внешнее соединение.
- Условие соединения указывается в предложении ON или USING, либо неявно задаётся ключевым словом NATURAL. Это условие определяет, какие строки двух исходных таблиц считаются «соответствующими» друг другу (это подробно рассматривается ниже).

T1 { [INNER] | { LEFT | RIGHT | FULL } [OUTER] } JOIN T2

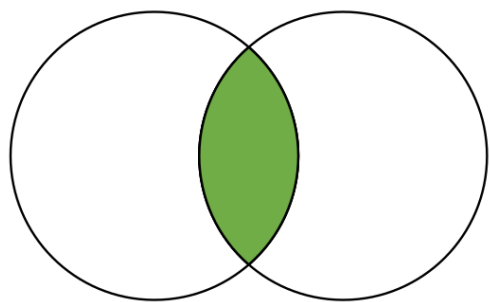
ON логическое_выражение

T1 { [INNER] | { LEFT | RIGHT | FULL } [OUTER] } JOIN T2

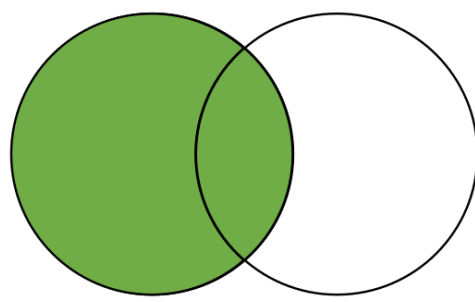
USING (список столбцов соединения)

T1 **NATURAL** { [INNER] | { LEFT | RIGHT | FULL } [OUTER] } JOIN T2

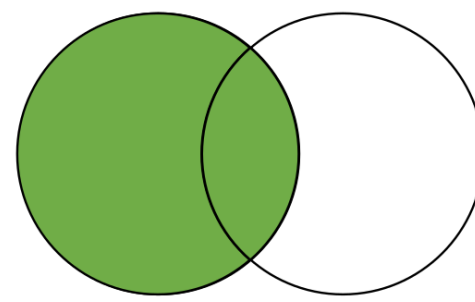
SELECT: элементы синтаксиса: соединение таблиц (JOIN)



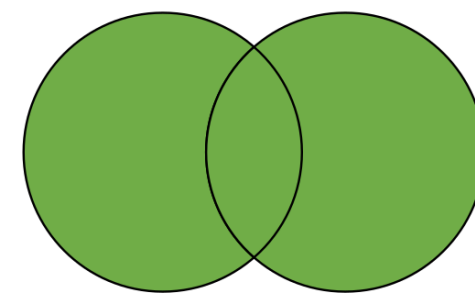
INNER JOIN



LEFT JOIN



RIGHT JOIN



FULL JOIN

SELECT: элементы синтаксиса: CROSS JOIN

- `SELECT * FROM employees
CROSS JOIN departments;`

employee_id	employee_name	department_id
1	John Doe	1
2	Jane Smith	2
3	Alice Johnson	1

department_id	department_name
1	Engineering
2	Sales



employee_id	employee_name	department_id	department_id-2	department_name
1	John Doe	1	1	Engineering
1	John Doe	1	2	Sales
2	Jane Smith	2	1	Engineering
2	Jane Smith	2	2	Sales
3	Alice Johnson	1	1	Engineering
3	Alice Johnson	1	2	Sales

SELECT: элементы синтаксиса: CROSS JOIN

SELECT * FROM employees
CROSS JOIN departments;



SELECT * FROM employees
INNER JOIN departments **ON**
TRUE;

employee_id	employee_name	department_id
1	John Doe	1
2	Jane Smith	2
3	Alice Johnson	1

department_id	department_name
1	Engineering
2	Sales



employee_id	employee_name	department_id	department_id-2	department_name
1	John Doe	1	1	Engineering
1	John Doe	1	2	Sales
2	Jane Smith	2	1	Engineering
2	Jane Smith	2	2	Sales
3	Alice Johnson	1	1	Engineering
3	Alice Johnson	1	2	Sales

SELECT: элементы синтаксиса: INNER JOIN

```
SELECT *  
FROM employees e  
    INNER JOIN departments d  
    ON e.department_id = d.department_id;
```

employee_id	employee_name	department_id
1	John Doe	1
2	Jane Smith	2
3	Alice Johnson	1

department_id	department_name
1	Engineering
2	Sales



employee_id	employee_name	department_id	department_id-2	department_name
1	John Doe	1	1	Engineering
2	Jane Smith	2	2	Sales
3	Alice Johnson	1	1	Engineering

SELECT: элементы синтаксиса: INNER JOIN

```
SELECT *  
FROM employees e  
    INNER JOIN departments d  
    ON e.department_id = 1;
```

employee_id	employee_name	department_id
1	John Doe	1
2	Jane Smith	2
3	Alice Johnson	1

department_id	department_name
1	Engineering
2	Sales



SELECT: элементы синтаксиса: INNER JOIN

```
SELECT *  
FROM employees e  
    INNER JOIN departments d  
    ON e.department_id = 1;
```

employee_id	employee_name	department_id
1	John Doe	1
2	Jane Smith	2
3	Alice Johnson	1

department_id	department_name
1	Engineering
2	Sales



employee_id	employee_name	department_id	department_id-2	department_name
1	John Doe	1	1	Engineering
3	Alice Johnson	1	1	Engineering
1	John Doe	1	2	Sales
3	Alice Johnson	1	2	Sales

SELECT: элементы синтаксиса: INNER JOIN

```
SELECT *  
FROM employees e  
    INNER JOIN departments d  
    ON e.department_id = d.department_id  
    AND e.department_id = 1;
```

employee_id	employee_name	department_id
1	John Doe	1
2	Jane Smith	2
3	Alice Johnson	1

department_id	department_name
1	Engineering
2	Sales



employee_id	employee_name	department_id	department_id-2	department_name
1	John Doe	1	1	Engineering
3	Alice Johnson	1	1	Engineering

SELECT: элементы синтаксиса: LEFT JOIN

```
SELECT * FROM employees  
LEFT JOIN departments  
ON employees.department_id =  
departments.department_id;
```

employee_id	employee_name	department_id
1	John Doe	1
2	Jane Smith	1
3	Alice Johnson	2
4	Emily Brown	2
5	David Wilson	4

department_id	department_name
1	Engineering
2	Sales
3	Marketing



SELECT: элементы синтаксиса: LEFT JOIN

```
SELECT * FROM employees  
LEFT JOIN departments  
ON employees.department_id =  
departments.department_id;
```

employee_id	employee_name	department_id
1	John Doe	1
2	Jane Smith	1
3	Alice Johnson	2
4	Emily Brown	2
5	David Wilson	4

department_id	department_name
1	Engineering
2	Sales
3	Marketing



employee_id	employee_name	department_id	department_id-2	department_name
1	John Doe	1	1	Engineering
2	Jane Smith	1	1	Engineering
3	Alice Johnson	2	2	Sales
4	Emily Brown	2	2	Sales
5	David Wilson	4		

SELECT: элементы синтаксиса: RIGHT JOIN

```
SELECT * FROM employees  
RIGHT JOIN departments  
ON employees.department_id =  
departments.department_id;
```

employee_id	employee_name	department_id
1	John Doe	1
2	Jane Smith	1
3	Alice Johnson	2
4	Emily Brown	2
5	David Wilson	4

department_id	department_name
1	Engineering
2	Sales
3	Marketing



SELECT: элементы синтаксиса: RIGHT JOIN

```
SELECT * FROM employees  
  RIGHT JOIN departments  
ON employees.department_id =  
  departments.department_id;
```

employee_id	employee_name	department_id
1	John Doe	1
2	Jane Smith	1
3	Alice Johnson	2
4	Emily Brown	2
5	David Wilson	4

department_id	department_name
1	Engineering
2	Sales
3	Marketing



employee_id	employee_name	department_id	department_id-2	department_name
2	Jane Smith	1	1	Engineering
1	John Doe	1	1	Engineering
4	Emily Brown	2	2	Sales
3	Alice Johnson	2	2	Sales
			3	Marketing

SELECT: элементы синтаксиса: FULL JOIN

```
SELECT * FROM  
  employees  
FULL JOIN departments  
ON employees.department_id =  
   departments.department_id;
```

employee_id	employee_name	department_id
1	John Doe	1
2	Jane Smith	1
3	Alice Johnson	2
4	Emily Brown	2
5	David Wilson	4

department_id	department_name
1	Engineering
2	Sales
3	Marketing



SELECT: элементы синтаксиса: FULL JOIN

```
SELECT * FROM  
  employees  
FULL JOIN departments  
ON employees.department_id =  
   departments.department_id;
```

employee_id	employee_name	department_id
1	John Doe	1
2	Jane Smith	1
3	Alice Johnson	2
4	Emily Brown	2
5	David Wilson	4

department_id	department_name
1	Engineering
2	Sales
3	Marketing



employee_id	employee_name	department_id	department_id-2	department_name
1	John Doe	1	1	Engineering
2	Jane Smith	1	1	Engineering
3	Alice Johnson	2	2	Sales
4	Emily Brown	2	2	Sales
5	David Wilson	4		
			3	Marketing

SELECT: элементы синтаксиса (ТАБЛИЧНОЕ ВЫРАЖЕНИЕ, WHERE)

- Синтаксис:

WHERE *условие_ограничения*

- *условие_ограничения* - любое выражение значения, выдающее результат типа boolean

SELECT ... FROM fdt WHERE c1 > 5

SELECT ... FROM fdt WHERE c1 IN (1, 2, 3)

SELECT ... FROM fdt WHERE c1 IN (SELECT c1 FROM t2)

SELECT: элементы синтаксиса (ТАБЛИЧНОЕ ВЫРАЖЕНИЕ, GROUP BY и HAVING)

- Синтаксис:

SELECT список_выборки

FROM ...

[WHERE ...]

GROUP BY *элемент_группирования* [, *элемент_группирования*]...

HAVING логическое_выражение

- В стандарте SQL GROUP BY может группировать только по столбцам исходной таблицы, но расширение PostgreSQL позволяет использовать GROUP BY более гибко. Предложение GROUP BY собирает в одну строку все выбранные строки, выдающие одинаковые значения для выражений группировки. В качестве выражения внутри *элемента_группирования* может выступать имя входного столбца, либо имя или порядковый номер выходного столбца (из списка элементов SELECT), либо произвольное значение, вычисляемое по значениям входных столбцов. В случае неоднозначности имя в GROUP BY будет восприниматься как имя входного, а не выходного столбца
- **Агрегатные функции**, если они используются, вычисляются по всем строкам, составляющим каждую группу, и в итоге выдают отдельное значение для каждой группы
- Когда в запросе присутствует предложение GROUP BY или какая-либо агрегатная функция, выражения в списке SELECT, по общему правилу, не могут обращаться к негруппируемым столбцам, так как иначе в негруппируемом столбце нужно было бы вернуть более одного возможного значения
- Если таблица была сгруппирована с помощью GROUP BY, но интерес представляют только некоторые группы, отфильтровать их можно с помощью предложения HAVING, действующего подобно WHERE

SELECT: элементы синтаксиса (ТАБЛИЧНОЕ ВЫРАЖЕНИЕ, GROUP BY и HAVING)

SELECT department FROM
new_employees GROUP BY
department;

SELECT salary FROM
new_employees GROUP BY
salary;

employee_id	name	age	date_of_employment	salary	department	position
1	Иван Иванов	35	15.01.2020	50000.00	ИТ	программист
2	Екатерина Смирнова	28	20.05.2018	60000.00	Маркетинг	ведущий маркетолог
3	Михаил Иванов	42	10.11.2015	70000.00	Финансы	руководитель отдела
4	Анна Сергеева	30	03.09.2019	50000.00	Кадры	специалист
5	Дмитрий Петров	45	22.07.2012	70000.00	ИТ	руководитель отдела
6	Ольга Кузнецова	38	12.03.2017	60000.00	Маркетинг	руководитель отдела
7	Александр Сидоров	33	18.08.2021	50000.00	Финансы	бухгалтер
8	Наталья Иванова	31	25.04.2016	50000.00	Кадры	специалист
9	Павел Морозов	40	30.10.2014	50000.00	ИТ	программист
10	Анастасия Лебедева	29	01.02.2022	50000.00	Маркетинг	маркетолог



SELECT: элементы синтаксиса (ТАБЛИЧНОЕ ВЫРАЖЕНИЕ, GROUP BY и HAVING)

SELECT department FROM
new_employees GROUP BY
department;

SELECT salary FROM
new_employees GROUP BY
salary;

employee_id	name	age	date_of_employment	salary	department	position
1	Иван Иванов	35	15.01.2020	50000.00	ИТ	программист
2	Екатерина Смирнова	28	20.05.2018	60000.00	Маркетинг	ведущий маркетолог
3	Михаил Иванов	42	10.11.2015	70000.00	Финансы	руководитель отдела
4	Анна Сергеева	30	03.09.2019	50000.00	Кадры	специалист
5	Дмитрий Петров	45	22.07.2012	70000.00	ИТ	руководитель отдела
6	Ольга Кузнецова	38	12.03.2017	60000.00	Маркетинг	руководитель отдела
7	Александр Сидоров	33	18.08.2021	50000.00	Финансы	бухгалтер
8	Наталья Иванова	31	25.04.2016	50000.00	Кадры	специалист
9	Павел Морозов	40	30.10.2014	50000.00	ИТ	программист
10	Анастасия Лебедева	29	01.02.2022	50000.00	Маркетинг	маркетолог



salary
60000.00
50000.00
70000.00

department
Кадры
Маркетинг
Финансы
ИТ

SELECT: элементы синтаксиса (ТАБЛИЧНОЕ ВЫРАЖЕНИЕ, GROUP BY и HAVING)

```
SELECT department, salary
FROM new_employees
GROUP BY
    salary,
    department;
```

employee_id	name	age	date_of_employment	salary	department	position
1	Иван Иванов	35	15.01.2020	50000.00	ИТ	программист
2	Екатерина Смирнова	28	20.05.2018	60000.00	Маркетинг	ведущий маркетолог
3	Михаил Иванов	42	10.11.2015	70000.00	Финансы	руководитель отдела
4	Анна Сергеева	30	03.09.2019	50000.00	Кадры	специалист
5	Дмитрий Петров	45	22.07.2012	70000.00	ИТ	руководитель отдела
6	Ольга Кузнецова	38	12.03.2017	60000.00	Маркетинг	руководитель отдела
7	Александр Сидоров	33	18.08.2021	50000.00	Финансы	бухгалтер
8	Наталья Иванова	31	25.04.2016	50000.00	Кадры	специалист
9	Павел Морозов	40	30.10.2014	50000.00	ИТ	программист
10	Анастасия Лебедева	29	01.02.2022	50000.00	Маркетинг	маркетолог



SELECT: элементы синтаксиса (ТАБЛИЧНОЕ ВЫРАЖЕНИЕ, GROUP BY и HAVING)

```
SELECT department, salary
FROM new_employees
GROUP BY
    salary,
    department;
```

- Порядок перечисления столбцов не влияет на результат!

employee_id	name	age	date_of_employment	salary	department	position
1	Иван Иванов	35	15.01.2020	50000.00	ИТ	программист
2	Екатерина Смирнова	28	20.05.2018	60000.00	Маркетинг	ведущий маркетолог
3	Михаил Иванов	42	10.11.2015	70000.00	Финансы	руководитель отдела
4	Анна Сергеева	30	03.09.2019	50000.00	Кадры	специалист
5	Дмитрий Петров	45	22.07.2012	70000.00	ИТ	руководитель отдела
6	Ольга Кузнецова	38	12.03.2017	60000.00	Маркетинг	руководитель отдела
7	Александр Сидоров	33	18.08.2021	50000.00	Финансы	бухгалтер
8	Наталья Иванова	31	25.04.2016	50000.00	Кадры	специалист
9	Павел Морозов	40	30.10.2014	50000.00	ИТ	программист
10	Анастасия Лебедева	29	01.02.2022	50000.00	Маркетинг	маркетолог



department	salary
Финансы	70000.00
Маркетинг	60000.00
Финансы	50000.00
ИТ	70000.00
ИТ	50000.00
Кадры	50000.00
Маркетинг	50000.00

SELECT: элементы синтаксиса (ТАБЛИЧНОЕ ВЫРАЖЕНИЕ, GROUP BY и HAVING)

```
SELECT department, salary
FROM new_employees
GROUP BY
    salary,
    department
HAVING salary > 60000;
```

employee_id	name	age	date_of_employment	salary	department	position
1	Иван Иванов	35	15.01.2020	50000.00	ИТ	программист
2	Екатерина Смирнова	28	20.05.2018	60000.00	Маркетинг	ведущий маркетолог
3	Михаил Иванов	42	10.11.2015	70000.00	Финансы	руководитель отдела
4	Анна Сергеева	30	03.09.2019	50000.00	Кадры	специалист
5	Дмитрий Петров	45	22.07.2012	70000.00	ИТ	руководитель отдела
6	Ольга Кузнецова	38	12.03.2017	60000.00	Маркетинг	руководитель отдела
7	Александр Сидоров	33	18.08.2021	50000.00	Финансы	бухгалтер
8	Наталья Иванова	31	25.04.2016	50000.00	Кадры	специалист
9	Павел Морозов	40	30.10.2014	50000.00	ИТ	программист
10	Анастасия Лебедева	29	01.02.2022	50000.00	Маркетинг	маркетолог



SELECT: элементы синтаксиса (ТАБЛИЧНОЕ ВЫРАЖЕНИЕ, GROUP BY и HAVING)

```
SELECT department, salary
FROM new_employees
GROUP BY
    salary,
    department
HAVING salary > 60000;
```

employee_id	name	age	date_of_employment	salary	department	position
1	Иван Иванов	35	15.01.2020	50000.00	ИТ	программист
2	Екатерина Смирнова	28	20.05.2018	60000.00	Маркетинг	ведущий маркетолог
3	Михаил Иванов	42	10.11.2015	70000.00	Финансы	руководитель отдела
4	Анна Сергеева	30	03.09.2019	50000.00	Кадры	специалист
5	Дмитрий Петров	45	22.07.2012	70000.00	ИТ	руководитель отдела
6	Ольга Кузнецова	38	12.03.2017	60000.00	Маркетинг	руководитель отдела
7	Александр Сидоров	33	18.08.2021	50000.00	Финансы	бухгалтер
8	Наталья Иванова	31	25.04.2016	50000.00	Кадры	специалист
9	Павел Морозов	40	30.10.2014	50000.00	ИТ	программист
10	Анастасия Лебедева	29	01.02.2022	50000.00	Маркетинг	маркетолог



department	salary
ИТ	70000.00
Финансы	70000.00

SELECT: базовый синтаксис (СОПТИРОВКА)

- Синтаксис выражения сортировки выглядит следующим образом:

SELECT список_выборки

FROM табличное_выражение

ORDER BY выражение_сортировки1 [ASC | DESC] [NULLS { FIRST | LAST }]

[, выражение_сортировки2 [ASC | DESC] [NULLS { FIRST | LAST }]] ...]

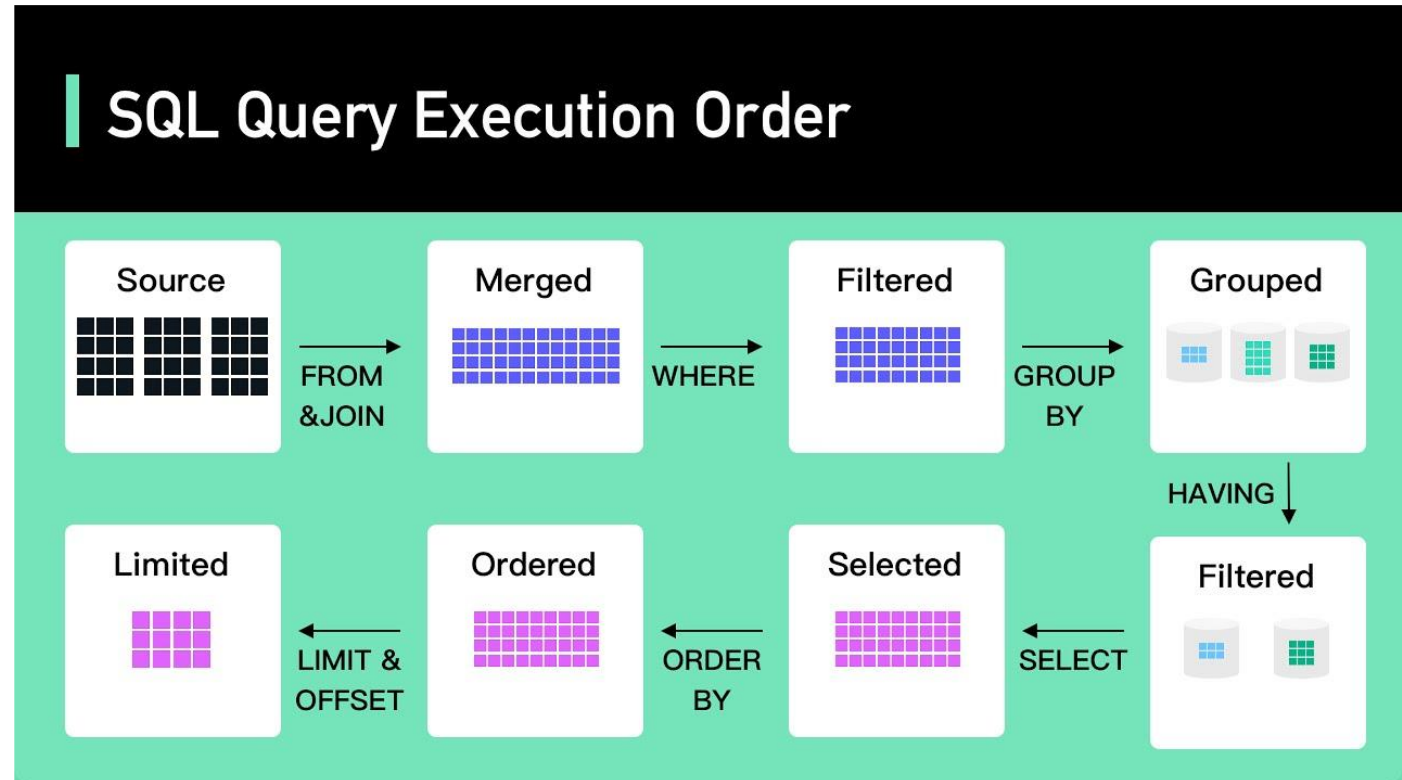
- Выражениями сортировки могут быть любые выражения, допустимые в списке выборки запроса
- Когда указывается несколько выражений, последующие значения позволяют отсортировать строки, в которых совпали все предыдущие значения. Каждое выражение можно дополнить ключевыми словами ASC или DESC, которые выбирают сортировку соответственно по возрастанию или убыванию. По умолчанию принят порядок по возрастанию (ASC)
- Для определения места значений NULL можно использовать указания NULLS FIRST и NULLS LAST, которые помещают значения NULL соответственно до или после значений не NULL. По умолчанию значения NULL считаются больше любых других, то есть подразумевается NULLS FIRST для порядка DESC и NULLS LAST в противном случае

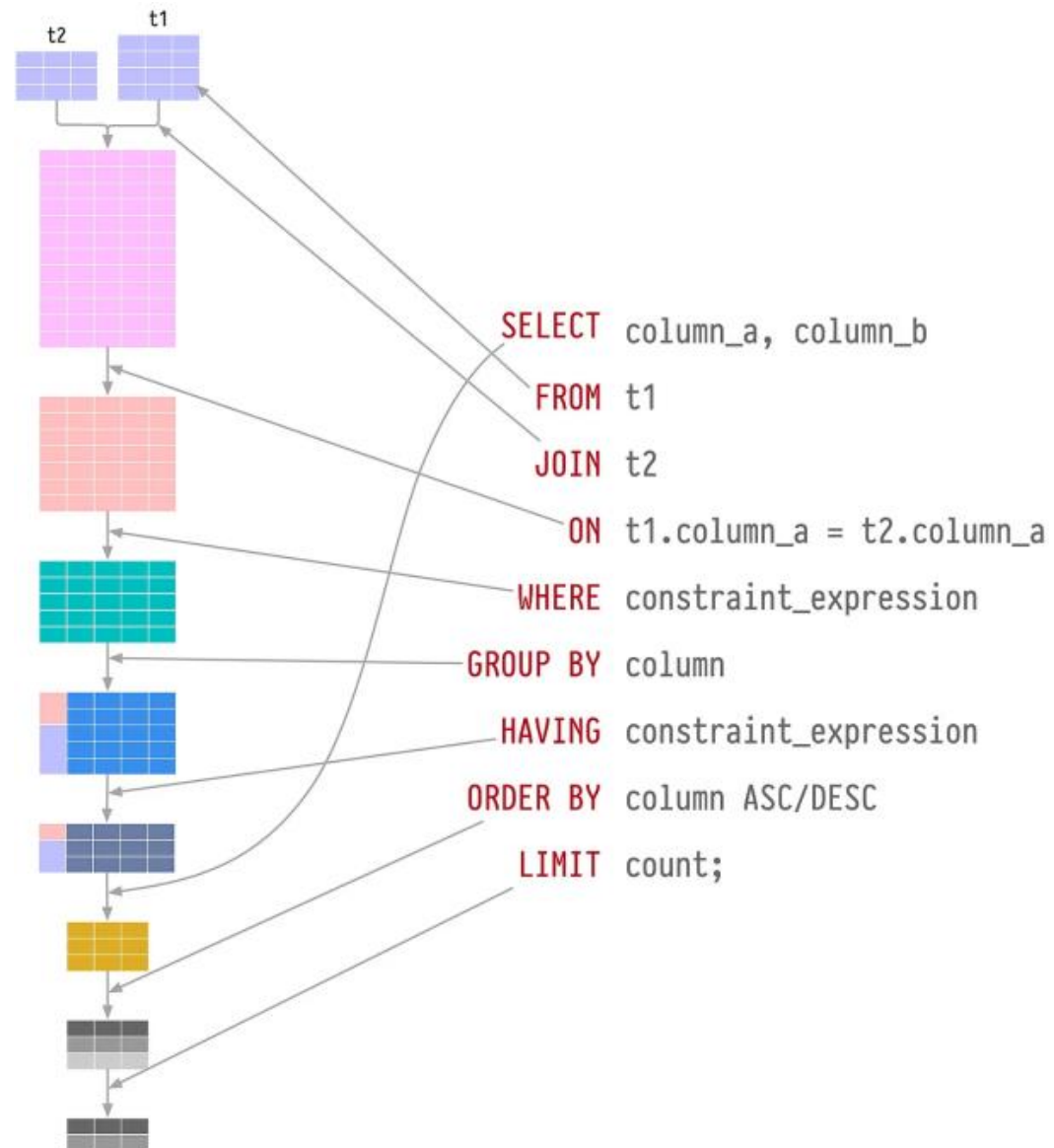
SQL: порядок выполнения запросов

- SQL – декларативный язык, в нем вы описываете то, что вы хотите получить, а не то, как это сделать
- SQL – семантически избыточный язык!
- Хотя разработчики языка пытались максимально снизить «порог входа» для начинающих работать с SQL, запросы на SQL не тривиальны
- **Процесс или команда получения данных из базы данных называется запросом.** В SQL запросы формулируются с помощью команды SELECT
- Разберемся, как работает SELECT-запрос

Этапы выполнения запроса SELECT

1. FROM & JOIN
2. WHERE
3. GROUP BY
4. HAVING
5. SELECT
6. ORDER BY
7. LIMIT & OFFSET





Пример работы запроса SELECT

- Мы хотим знать название только двух городов, кроме Сан-Бруно, в которых проживает два или более жителей.
- Мы также хотим получить результат, упорядоченный по алфавиту

CITIZEN	
Name	char(20)
City_id	integer



CITY	
City_id	integer
City_name	char(50)

CITIZEN	
Name	City_id
Andre	3
Rachel	2
Jenny	1
Dough	3
Kevin	1
Sarah	2
Trevor	3
Al	1
Yung	1

CITY	
City_id	City_name
1	Palo Alto
2	Sunnyvale
3	San Bruno

Пример работы запроса SELECT

```
SELECT city.city_name AS "City"  
FROM citizen  
JOIN city  
ON citizen.city_id = city.city_id  
WHERE city.city_name != 'San Bruno'  
GROUP BY city.city_name  
HAVING COUNT(*) >= 2  
ORDER BY city.city_name ASC  
LIMIT 2
```

CITIZEN	
Name	City_id
Andre	3
Rachel	2
Jenny	1
Dough	3
Kevin	1
Sarah	2
Trevor	3
Al	1
Yung	1

CITY	
City_id	City_name
1	Palo Alto
2	Sunnyvale
3	San Bruno

Пример работы запроса SELECT

```
SELECT city.city_name AS "City"  
FROM citizen  
JOIN city  
ON citizen.city_id = city.city_id  
WHERE city.city_name != 'San Bruno'  
GROUP BY city.city_name  
HAVING COUNT(*) >= 2  
ORDER BY city.city_name ASC  
LIMIT 2
```

Name	City_id	City_id	City_name
Andre	3	1	Palo Alto
Andre	3	2	Sunnyvale
Andre	3	3	San Bruno
Rachel	2	1	Palo Alto
Rachel	2	2	Sunnyvale
Rachel	2	3	San Bruno
Jenny	1	1	Palo Alto
Jenny	1	2	Sunnyvale
Jenny	1	3	San Bruno
Dough	3	1	Palo Alto
Dough	3	2	Sunnyvale
Dough	3	3	San Bruno
Kevin	1	1	Palo Alto
Kevin	1	2	Sunnyvale
Kevin	1	3	San Bruno
Sarah	2	1	Palo Alto
Sarah	2	2	Sunnyvale
Sarah	2	3	San Bruno
Trevor	3	1	Palo Alto
Trevor	3	2	Sunnyvale
Trevor	3	3	San Bruno
Al	1	1	Palo Alto
Al	1	2	Sunnyvale
Al	1	3	San Bruno
Yung	1	1	Palo Alto
Yung	1	2	Sunnyvale
Yung	1	3	San Bruno

Пример работы запроса SELECT

```
SELECT city.city_name AS "City"  
FROM citizen  
JOIN city  
ON citizen.city_id = city.city_id  
WHERE city.city_name != 'San Bruno'  
GROUP BY city.city_name  
HAVING COUNT(*) >= 2  
ORDER BY city.city_name ASC  
LIMIT 2
```

Name	City_id	City_name
Andre	3	San Bruno
Rachel	2	Sunnyvale
Jenny	1	Palo Alto
Dough	3	San Bruno
Kevin	1	Palo Alto
Sarah	2	Sunnyvale
Trevor	3	San Bruno
Al	1	Palo Alto
Yung	1	Palo Alto

Пример работы запроса SELECT

```
SELECT city.city_name AS "City"  
FROM citizen  
JOIN city  
ON citizen.city_id = city.city_id  
WHERE city.city_name != 'San Bruno'  
GROUP BY city.city_name  
HAVING COUNT(*) >= 2  
ORDER BY city.city_name ASC  
LIMIT 2
```

Name	City_id	City_name
Rachel	2	Sunnyvale
Jenny	1	Palo Alto
Kevin	1	Palo Alto
Sarah	2	Sunnyvale
Al	1	Palo Alto
Yung	1	Palo Alto

Пример работы запроса SELECT

```
SELECT city.city_name AS "City"  
FROM citizen  
JOIN city  
ON citizen.city_id = city.city_id  
WHERE city.city_name != 'San Bruno'  
GROUP BY city.city_name  
HAVING COUNT(*) >= 2  
ORDER BY city.city_name ASC  
LIMIT 2
```

Name	City_id	City_name
Rachel	2	Sunnyvale
Sarah	2	
Jenny	1	Palo Alto
Kevin	1	
Al	1	
Yung	1	

Пример работы запроса SELECT

```
SELECT city.city_name AS "City"  
FROM citizen  
JOIN city  
ON citizen.city_id = city.city_id  
WHERE city.city_name != 'San Bruno'  
GROUP BY city.city_name  
HAVING COUNT(*) >= 2  
ORDER BY city.city_name ASC  
LIMIT 2
```

Name	City_id	City_name
Rachel	2	Sunnyvale
Sarah	2	
Jenny	1	Palo Alto
Kevin	1	
Al	1	
Yung	1	

Пример работы запроса SELECT

```
SELECT city.city_name AS "City"  
FROM citizen  
JOIN city  
ON citizen.city_id = city.city_id  
WHERE city.city_name != 'San Bruno'  
GROUP BY city.city_name  
HAVING COUNT(*) >= 2  
ORDER BY city.city_name ASC  
LIMIT 2
```

City
Sunnyvale
Palo Alto

Пример работы запроса SELECT

```
SELECT city.city_name AS "City"  
FROM citizen  
JOIN city  
ON citizen.city_id = city.city_id  
WHERE city.city_name != 'San Bruno'  
GROUP BY city.city_name  
HAVING COUNT(*) >= 2  
ORDER BY city.city_name ASC  
LIMIT 2
```

City
Palo Alto
Sunnyvale

SELECT: использование агрегирующих(агрегатных) функций

Функций довольно много, мы рассмотрим только несколько часто употребляемых:

№	Функция	Описание
1	count (*) → bigint	Выдаёт количество входных строк
2	sum (<args...>) → numeric	Вычисляет сумму всех входных значений, отличных от NULL
3	max (<args...>) → тот же тип, что на входе	Вычисляет максимальное из всех значений, отличных от NULL
4	min (<args...>) → тот же тип, что на входе	Вычисляет минимальное из всех значений, отличных от NULL
5	avg (<args...>) → numeric	Вычисляет арифметическое среднее для всех входных значений, отличных от NULL

SELECT: использование агрегирующих(агрегатных) функций

```
SELECT count(*) as  
total_rows  
FROM new_employees;
```

employee_id	name	age	date_of_employment	salary	department	position
1	Иван Иванов	35	15.01.2020	50000.00	ИТ	программист
2	Екатерина Смирнова	28	20.05.2018	60000.00	Маркетинг	ведущий маркетолог
3	Михаил Иванов	42	10.11.2015	70000.00	Финансы	руководитель отдела
4	Анна Сергеева	30	03.09.2019	50000.00	Кадры	специалист
5	Дмитрий Петров	45	22.07.2012	70000.00	ИТ	руководитель отдела
6	Ольга Кузнецова	38	12.03.2017	60000.00	Маркетинг	руководитель отдела
7	Александр Сидоров	33	18.08.2021	50000.00	Финансы	бухгалтер
8	Наталья Иванова	31	25.04.2016	50000.00	Кадры	специалист
9	Павел Морозов	40	30.10.2014	50000.00	ИТ	программист
10	Анастасия Лебедева	29	01.02.2022	50000.00	Маркетинг	маркетолог



total_rows
10

SELECT: использование агрегирующих(агрегатных) функций

```
SELECT sum(salary) as  
total_payments  
FROM new_employees;
```

employee_id	name	age	date_of_employment	salary	department	position
1	Иван Иванов	35	15.01.2020	50000.00	ИТ	программист
2	Екатерина Смирнова	28	20.05.2018	60000.00	Маркетинг	ведущий маркетолог
3	Михаил Иванов	42	10.11.2015	70000.00	Финансы	руководитель отдела
4	Анна Сергеева	30	03.09.2019	50000.00	Кадры	специалист
5	Дмитрий Петров	45	22.07.2012	70000.00	ИТ	руководитель отдела
6	Ольга Кузнецова	38	12.03.2017	60000.00	Маркетинг	руководитель отдела
7	Александр Сидоров	33	18.08.2021	50000.00	Финансы	бухгалтер
8	Наталья Иванова	31	25.04.2016	50000.00	Кадры	специалист
9	Павел Морозов	40	30.10.2014	50000.00	ИТ	программист
10	Анастасия Лебедева	29	01.02.2022	50000.00	Маркетинг	маркетолог



total_payments
560000.00

SELECT: использование агрегирующих(агрегатных) функций

```
SELECT  
max(date_of_employment)  
AS last_employee  
FROM new_employees;
```

employee_id	name	age	date_of_employment	salary	department	position
1	Иван Иванов	35	15.01.2020	50000.00	ИТ	программист
2	Екатерина Смирнова	28	20.05.2018	60000.00	Маркетинг	ведущий маркетолог
3	Михаил Иванов	42	10.11.2015	70000.00	Финансы	руководитель отдела
4	Анна Сергеева	30	03.09.2019	50000.00	Кадры	специалист
5	Дмитрий Петров	45	22.07.2012	70000.00	ИТ	руководитель отдела
6	Ольга Кузнецова	38	12.03.2017	60000.00	Маркетинг	руководитель отдела
7	Александр Сидоров	33	18.08.2021	50000.00	Финансы	бухгалтер
8	Наталья Иванова	31	25.04.2016	50000.00	Кадры	специалист
9	Павел Морозов	40	30.10.2014	50000.00	ИТ	программист
10	Анастасия Лебедева	29	01.02.2022	50000.00	Маркетинг	маркетолог



last_employee
01.02.2022

SELECT: использование агрегирующих(агрегатных) функций

```
SELECT  
min(date_of_employment)  
AS first_employee  
FROM new_employees;
```

employee_id	name	age	date_of_employment	salary	department	position
1	Иван Иванов	35	15.01.2020	50000.00	ИТ	программист
2	Екатерина Смирнова	28	20.05.2018	60000.00	Маркетинг	ведущий маркетолог
3	Михаил Иванов	42	10.11.2015	70000.00	Финансы	руководитель отдела
4	Анна Сергеева	30	03.09.2019	50000.00	Кадры	специалист
5	Дмитрий Петров	45	22.07.2012	70000.00	ИТ	руководитель отдела
6	Ольга Кузнецова	38	12.03.2017	60000.00	Маркетинг	руководитель отдела
7	Александр Сидоров	33	18.08.2021	50000.00	Финансы	бухгалтер
8	Наталья Иванова	31	25.04.2016	50000.00	Кадры	специалист
9	Павел Морозов	40	30.10.2014	50000.00	ИТ	программист
10	Анастасия Лебедева	29	01.02.2022	50000.00	Маркетинг	маркетолог



first_employee
22.07.2012

SELECT: использование агрегирующих(агрегатных) функций

```
SELECT avg(salary) as  
salary_average  
FROM new_employees;
```

employee_id	name	age	date_of_empoyment	salary	department	position
1	Иван Иванов	35	15.01.2020	50000.00	ИТ	программист
2	Екатерина Смирнова	28	20.05.2018	60000.00	Маркетинг	ведущий маркетолог
3	Михаил Иванов	42	10.11.2015	70000.00	Финансы	руководитель отдела
4	Анна Сергеева	30	03.09.2019	50000.00	Кадры	специалист
5	Дмитрий Петров	45	22.07.2012	70000.00	ИТ	руководитель отдела
6	Ольга Кузнецова	38	12.03.2017	60000.00	Маркетинг	руководитель отдела
7	Александр Сидоров	33	18.08.2021	50000.00	Финансы	бухгалтер
8	Наталья Иванова	31	25.04.2016	50000.00	Кадры	специалист
9	Павел Морозов	40	30.10.2014	50000.00	ИТ	программист
10	Анастасия Лебедева	29	01.02.2022	50000.00	Маркетинг	маркетолог



salary_avrage
56000.0000000000