КСиТ

# 4. Язык манипулирования данными

## Базы данных

**Хисамутдинов М.А.**
кафедра №12 НИЯУ МИФИ
2026

# План занятия

1. SQL
2. Insert, Delete, Update
3. Select
4. Join’ы
5. Примеры запросов с таблицами

# SQL



SQL (Structured Query Language) — это язык запросов к реляционным базам данных.

С его помощью можно:

- извлекать данные (SELECT)
- добавлять новые (INSERT)
- изменять существующие (UPDATE)
- удалять данные (DELETE)
- управлять структурой таблиц и правами доступа

SQL - это декларативный язык. Описываем что хотим получить, а не как именно база данных должна это сделать.

# INSERT

INSERT - добавление данных

```
INSERT INTO имя_таблицы (колонка1, ...)
VALUES (значение1, ...)

INSERT INTO имя_таблицы (колонка1, ...)
(подзапрос)
```



# INSERT

```sql
INSERT INTO users (id, name, email)
VALUES (1, 'Ivan', 'ivan@example.com');
```

```sql
INSERT INTO users (name, email)
VALUES
  ('Alice', 'alice@mail.com'),
  ('Bob', 'bob@mail.com'),
  ('Charlie', 'charlie@mail.com');
```

```sql
INSERT INTO archive_users (id, name, email)
SELECT id, name, email
FROM users
WHERE deleted_at IS NOT NULL;
```



# UPDATE

UPDATE - изменение данных

```
UPDATE имя таблицы
SET колонка1 = выражение, ...
WHERE условие_отбора_строк;
```

⚠️ Без WHERE обновятся все строки в таблице

# UPDATE

```sql
UPDATE users
SET email = 'new@mail.com'
WHERE id = 1;
```

```sql
UPDATE users
SET
  last_login = NOW(),
  is_active = true
WHERE email IS NOT NULL;
```

```sql
UPDATE accounts
SET balance = balance - 100
WHERE user_id = 42;
```

# UPDATE

```sql
UPDATE orders
SET status = 'paid'
WHERE id IN (
  SELECT order_id
  FROM payments
  WHERE success = true
);
```

# DELETE

DELETE - удаление данных

```
DELETE FROM имя_таблицы
WHERE условие_отбора_строк
```

⚠️ Без WHERE удалятся все строки в таблице

# DELETE

```sql
DELETE FROM users
WHERE id = 10;
```

```sql
DELETE FROM sessions
WHERE expires_at < NOW();
```

```sql
DELETE FROM orders
WHERE user_id IN (
  SELECT id
  FROM users
  WHERE banned = true
);
```

# SELECT

SELECT - извлечение данных

```
SELECT [DISTINCT] список_вывода
FROM источники
[INNER | LEFT | RIGHT | FULL] JOIN таблица_соединения
  ON условие_соединение
WHERE условие_отбора_строк
GROUP BY список_для_группирования
HAVING условие_отбора_групп
ORDER BY список_для_упорядочитвания
LIMIT количество_строк OFFSET смещение;
```

- Каждому столбцу можно выдать новое имя (alias).
- Для каждой используемой таблицы можно указать короткий alias.
- Описываем, что хотим получить
- WHERE фильтрует строки до группировки
- HAVING фильтрует группы после GROUP BY

# Логический порядок выполнения

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

# UNION

UNION - оператор SQL, который объединяет результаты нескольких SELECT-запросов в один набор строк.

Важно:

- объединяются результаты, а не таблицы
- запросы выполняются независимо
- результат выглядит как один SELECT

```
SELECT ...
UNION
SELECT ...
```

# UNION

Требования к UNION:

1. одинаковое количество колонок
2. совместимые типы данных
3. порядок колонок важен
4. имена колонок берутся из первого SELECT

Виды UNION:

1. UNION (по умолчанию): внутри имеется DISTINCT и удаляет дубликаты строк
2. UNION ALL - не удаляет дубликаты, работает быстрее
3. UNION + ORDER BY - с сортировкой данных, но ORDER BY можно писать только один раз и в конце



# UNION

**users**

| id | name |
|---|---|
| 1 | Alex |
| 2 | Alice |

**admins**

| id | name |
|---|---|
| 10 | Alice |
| 11 | Bob |

```
SELECT name FROM users
UNION
SELECT name FROM admins;
```

| name |
|---|
| Alex |
| Alice |
| Bob |

# UNION

**users**

| id | name |
|---|---|
| 1 | Alex |
| 2 | Alice |

**admins**

| id | name |
|---|---|
| 10 | Alice |
| 11 | Bob |

```
SELECT name FROM users
UNION ALL
SELECT name FROM admins;
```

| name |
|---|
| Alex |
| Alice |
| Alice |
| Bob |

# UNION с разными колонками

Соединить можно, если указать явно NULL

**users**

| id | name |
|---|---|
| 1 | Alex |
| 2 | Alice |

**admins**

| id | name |
|---|---|
| 10 | Alice |
| 11 | Bob |

```sql
SELECT id, name, NULL AS role
FROM users
UNION ALL
SELECT id, name, 'admin' AS role
FROM admins;
```

# UNION с разными колонками

| id | name | role |
|---|---|---|
| 1 | Alex | NULL |
| 2 | Alice | NULL |
| 10 | Alice | admin |
| 11 | Bob | admin |

# EXCEPT

EXCEPT - оператор SQL, который возвращает строки из первого SELECT, которых НЕТ во втором SELECT.

Требования аналогичны UNION.

```
SELECT ...
EXCEPT
SELECT ...;
```

Виды:

- EXCEPT - удаляет дубликаты
- EXCEPT ALL - учитывает количеств повторений

# EXCEPT



**users**

| name |
| --- |
| Alex |
| Alice |
| Alice |
| Bob |

**banned_users**

| name |
| --- |
| Alice |

```
SELECT name FROM users
EXCEPT
SELECT name FROM banned_users;
```

| name |
| --- |
| Alex |
| Bob |

# EXCEPT ALL

**users**

| name |
|---|
| Alex |
| Alice |
| Alice |
| Bob |

**banned_users**

| name |
|---|
| Alice |

```
SELECT name FROM users
EXCEPT ALL
SELECT name FROM banned_users;
```

| name |
|---|
| Alex |
| Alice |
| Bob |



# INTERSECT

INTERSECT возвращает только те строки, которые есть одновременно в обоих SELECT.

Требования аналогичны UNION, EXCEPT

```
SELECT ...
INTERSECT
SELECT ...;
```

Виды:

- INTERSECT - возвращает уникальные значения
- INTERSECT ALL - учитывает количеств повторений



# INTERSECT



**users**

| name |
| --- |
| Alex |
| Alice |
| Alice |
| Bob |

**admins**

| name |
| --- |
| Alice |
| Alice |
| Bob |
| Bob |

```
SELECT name FROM users
INTERSECT
SELECT name FROM admins;
```

| name |
| --- |
| Alice |
| Bob |

# INTERSECT ALL

**users**

| name |
|---|
| Alex |
| Alice |
| Alice |
| Bob |

**admins**

| name |
|---|
| Alice |
| Alice |
| Bob |
| Bob |

```
SELECT name FROM users
INTERSECT ALL
SELECT name FROM admins;
```

| name |
|---|
| Alice |
| Alice |
| Bob |

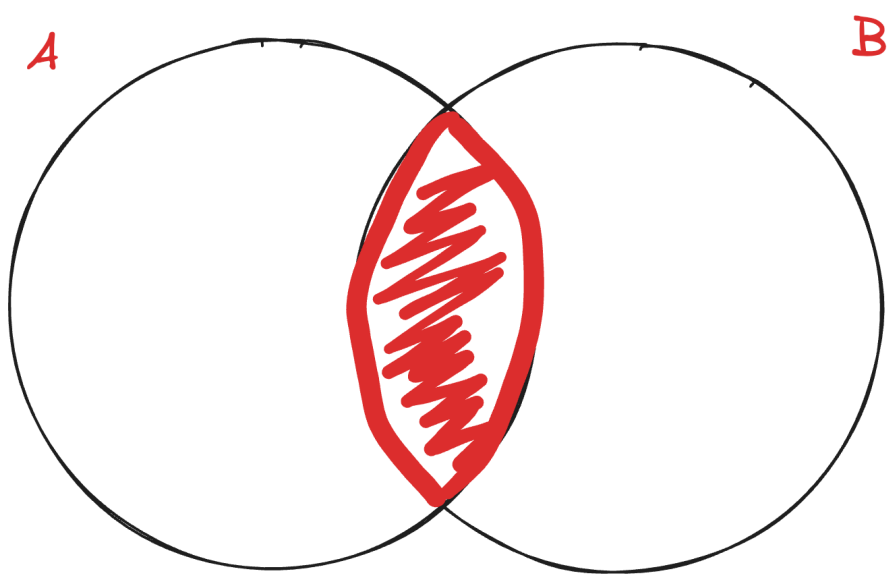


# INNER JOIN



JOIN - это операция, которая соединяет строки из двух (или более) таблиц по какому-то условию. Если не указывать вид соединения таблиц, то по умолчанию будет выполняться INNER JOIN.

INNER JOIN - возвращает только те строки, где нашлось совпадение в обеих таблицах.



# INNER JOIN



**users**

| id | name |
|---|---|
| 1 | Alex |
| 2 | Alice |
| 3 | Bob |
| 4 | Dana |

**orders**

| id | user_id | amount |
|---|---|---|
| 101 | 1 | 120 |
| 102 | 1 | 60 |
| 103 | 2 | 200 |
| 104 | 5 | 999 |

```
SELECT
  u.id   AS user_id,
  u.name,
  o.id   AS order_id,
  o.amount
FROM users u
INNER JOIN orders o
  ON o.user_id = u.id
ORDER BY u.id, o.id;
```

# INNER JOIN

**Результат**

| user_id | name | order_id | amount |
|---|---|---|---|
| 1 | Alex | 101 | 120 |
| 1 | Alex | 102 | 60 |
| 2 | Alice | 103 | 200 |

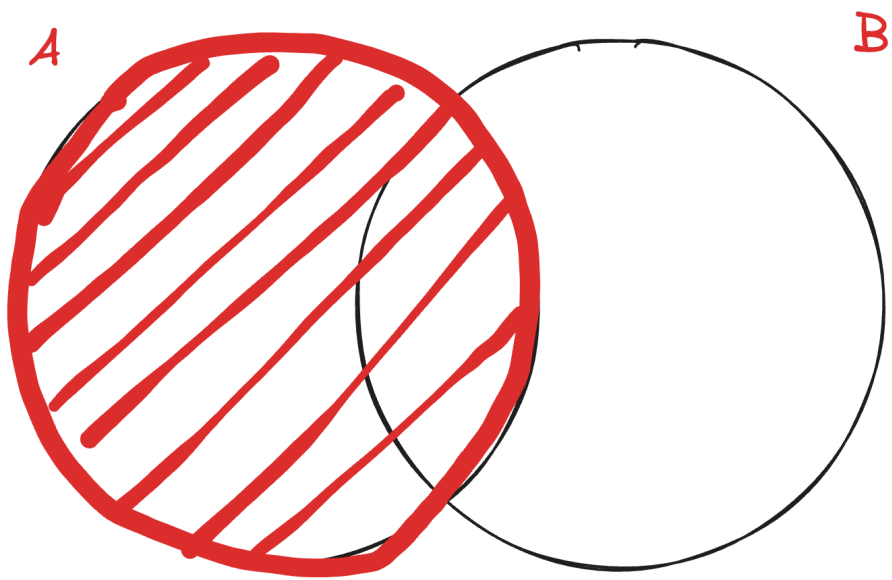
Результат будет таким же, если уберем inner

```
SELECT
  u.id   AS user_id,
  u.name,
  o.id   AS order_id,
  o.amount
FROM users u
JOIN orders o ON o.user_id = u.id
ORDER BY u.id, o.id;
```

# LEFT JOIN

LEFT JOIN возвращает:

- Все строки из левой таблицы
- Совпадения из правой
- Если совпадения нет → NULL в колонка правой таблицы



# LEFT JOIN



**users**

| id | name |
|---|---|
| 1 | Alex |
| 2 | Alice |
| 3 | Bob |
| 4 | Dana |

**orders**

| id | user_id | amount |
|---|---|---|
| 101 | 1 | 120 |
| 102 | 1 | 60 |
| 103 | 2 | 200 |
| 104 | 5 | 999 |

```
SELECT
  u.id   AS user_id,
  u.name,
  o.id   AS order_id,
  o.amount
FROM users u
LEFT JOIN orders o
  ON o.user_id = u.id
ORDER BY u.id, o.id;
```

# LEFT JOIN

| user_id | name | order_id | amount |
|---|---|---|---|
| 1 | Alex | 101 | 120 |
| 1 | Alex | 102 | 60 |
| 2 | Alice | 103 | 200 |
| 3 | Bob | NULL | NULL |
| 4 | Dana | NULL | NULL |

# LEFT ANTI JOIN

Иногда требуется найти пользователей без заказов. Или сделать LEFT ANTI JOIN.

```
SELECT u.id, u.name
FROM users u
LEFT JOIN orders o ON o.user_id = u.id
WHERE o.id IS NULL;
```

| id | name |
|---|---|
| 3 | Bob |
| 4 | Dana |

# Ловушка LEFT JOIN

```sql
SELECT u.id, u.name
FROM users u
LEFT JOIN orders o ON o.user_id = u.id
WHERE o.amount > 100;
```

Какой будет результат?

**users**

| id | name |
|---|---|
| 1 | Alex |
| 2 | Alice |
| 3 | Bob |
| 4 | Dana |

**orders**

| id | user_id | amount |
|---|---|---|
| 101 | 1 | 120 |
| 102 | 1 | 60 |
| 103 | 2 | 200 |
| 104 | 5 | 999 |



# Ловушка LEFT JOIN

```
SELECT u.id, u.name, o.amount
FROM users u
LEFT JOIN orders o ON o.user_id = u.id
WHERE o.amount > 100;
```

| id | name | amount |
|---|---|---|
| 1 | Alex | 120 |
| 2 | Alice | 200 |

Как сделать так, чтобы остались все наши пользователи?



# Ловушка LEFT JOIN

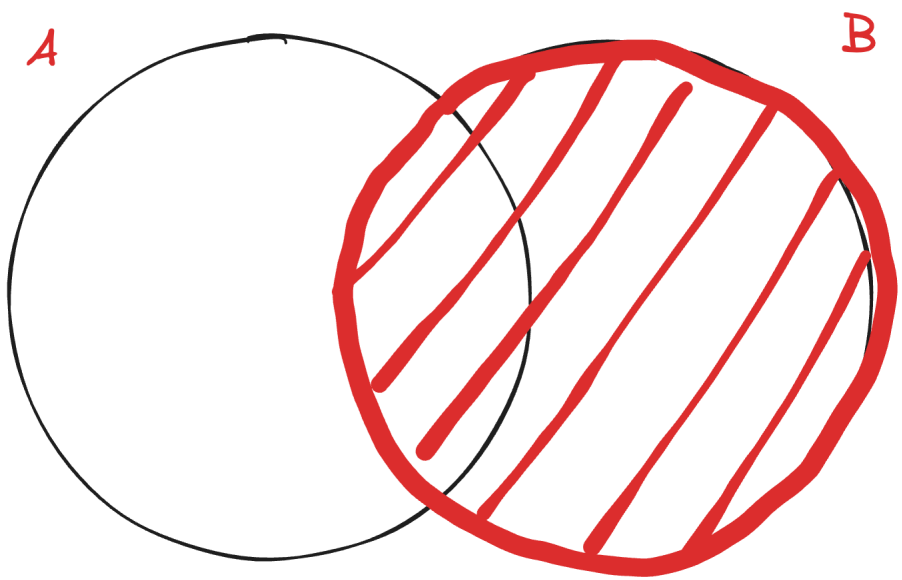
```
SELECT u.id, u.name
FROM users u
LEFT JOIN orders o
  ON o.user_id = u.id
 AND o.amount > 100;
```

| id | name | amount |
|---|---|---|
| 1 | Alex | 120 |
| 2 | Alice | 200 |
| 3 | Bob | NULL |
| 4 | Dana | NULL |

# RIGHT JOIN

RIGHT JOIN возвращает:

- Все строки из правой таблицы
- Совпадения из левой
- Если совпадения нет → NULL в колонка левой таблицы



# RIGHT JOIN



**users**

| id | name |
|---|---|
| 1 | Alex |
| 2 | Alice |
| 3 | Bob |
| 4 | Dana |

**orders**

| id | user_id | amount |
|---|---|---|
| 101 | 1 | 120 |
| 102 | 1 | 60 |
| 103 | 2 | 200 |
| 104 | 5 | 999 |

```
SELECT
  u.id   AS user_id,
  u.name,
  o.id   AS order_id,
  o.amount
FROM users u
RIGHT JOIN orders o
  ON o.user_id = u.id
ORDER BY o.id;
```

# RIGHT JOIN

| user_id | name | order_id | amount |
|---|---|---|---|
| 1 | Alex | 101 | 120 |
| 1 | Alex | 102 | 60 |
| 2 | Alice | 103 | 200 |
| NULL | NULL | 104 | 999 |

RIGHT JOIN почти всегда можно переписать как LEFT JOIN, просто поменяв таблицы местами.

```
SELECT
  u.id,
  u.name,
  o.id,
  o.amount
FROM orders o
LEFT JOIN users u ON o.user_id = u.id;
```



# RIGHT ANTI JOIN

Иногда требуется найти заказы без пользователей. Или сделать `RIGHT ANTI JOIN`.

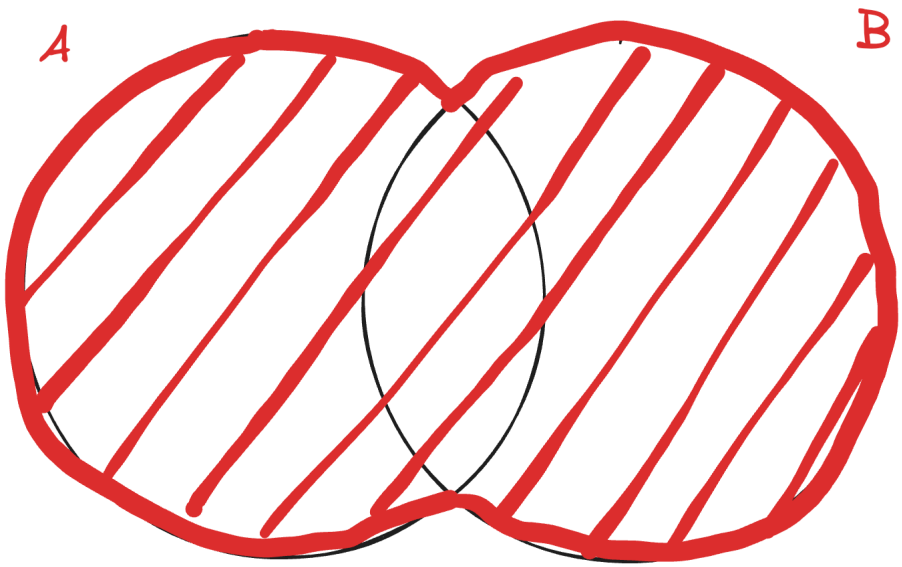
```
SELECT o.id, o.user_id
FROM users u
RIGHT JOIN orders o ON o.user_id = u.id
WHERE u.id IS NULL;
```

| id | user_id |
|---|---|
| 104 | 5 |

# FULL JOIN

FULL JOIN возвращает:

- Все строки из левой таблицы
- Все строки из правой таблицы
- Совпадения - в одну строку
- Если совпадения нет → NULL с противоположной стороны



# FULL JOIN

```sql
SELECT
  u.id      AS user_id,
  u.name,
  o.user_id AS order_user_id,
  o.id      AS order_id,
  o.amount
FROM users u
FULL JOIN orders o
  ON o.user_id = u.id
ORDER BY u.id, o.id;
```



# FULL JOIN



| user_id | name | order_user_id | order_id | amount |
|---|---|---|---|---|
| 1 | Alex | 1 | 101 | 120 |
| 1 | Alex | 1 | 102 | 60 |
| 2 | Alice | 2 | 103 | 200 |
| 3 | Bob | NULL | NULL | NULL |
| 4 | Dana | NULL | NULL | NULL |
| NULL | NULL | 5 | 104 | 999 |

```sql
SELECT u.user_id, u.name, o.user_id as order_user_id, o.order_id, o.amount
FROM users u
LEFT JOIN orders o ON o.user_id = u.id
UNION ALL
SELECT u.user_id, u.name, o.user_id as order_user_id, o.order_id, o.amount
FROM users u
RIGHT JOIN orders o ON o.user_id = u.id
WHERE u.id IS NULL;
```

# CROSS JOIN



CROSS JOIN делает декартово произведение - к аждая строка из левой таблицы соединяется с каждой строкой из правой.

Количество строк в результате = rows_left x rows_right

# CROSS JOIN

**users**

| id | name |
|---|---|
| 1 | Alex |
| 2 | Alice |
| 3 | Bob |

**orders**

| type |
|---|
| small |
| big |

```sql
SELECT
  u.id,
  u.name,
  o.type
FROM users u
CROSS JOIN orders o
ORDER BY u.id, o.type;
```

# CROSS JOIN

| id | name | type |
|---|---|---|
| 1 | Alex | small |
| 1 | Alex | big |
| 2 | Alice | small |
| 2 | Alice | big |
| 3 | Bob | small |
| 3 | Bob | big |

# CROSS JOIN

Можно сделать CROSS JOIN без слова JOIN

```
SELECT *
FROM users u, orders o;
```

Но читаемость хуже

# INNER JOIN через WHERE

```sql
SELECT
  u.id   AS user_id,
  u.name,
  o.id   AS order_id,
  o.amount
FROM users u, orders o
WHERE o.user_id = u.id
ORDER BY u.id, o.id;
```

Что происходит на самом деле?

1. Выполняется CROSS JOIN
2. Фильтруются данные согласно условию WHERE



# Почему визуализация с кругами плохая?

```
SELECT *
FROM table1
INNER JOIN table2 on table1.id = table2.id
```

**table1**

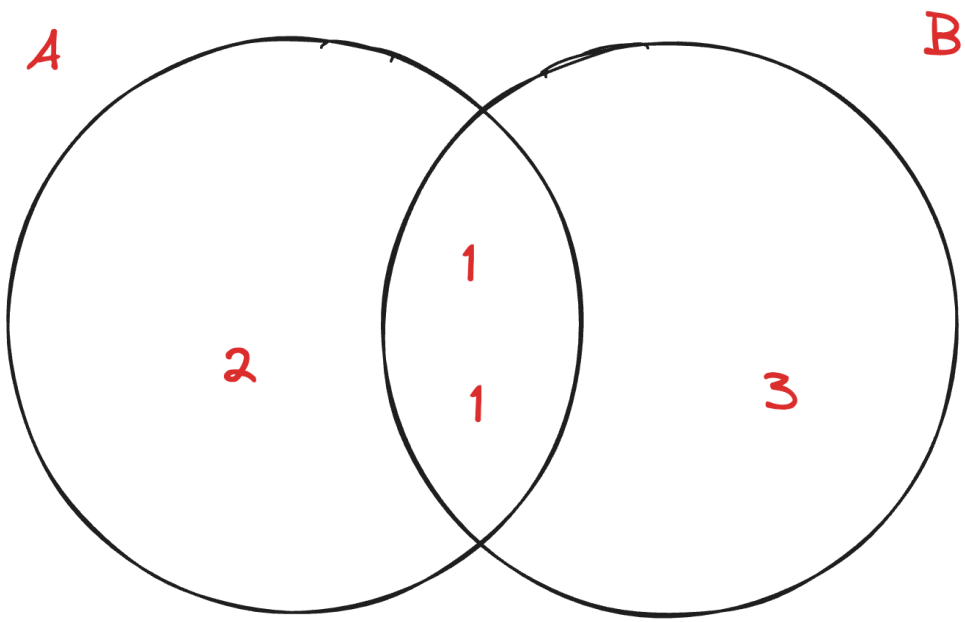
| id |
|---|
| 1 |
| 1 |
| 3 |

**table2**

| id |
|---|
| 1 |
| 1 |
| 2 |



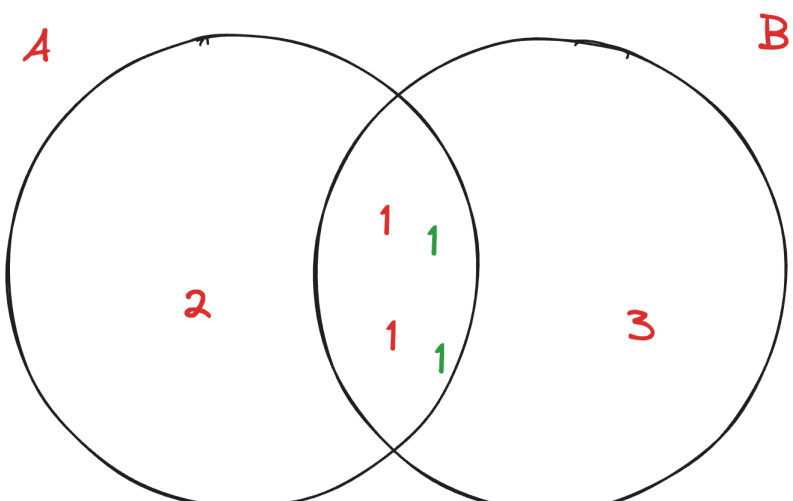
# Почему визуализация с кругами плохая?



# Почему визуализация с кругами плохая?

Но на самом деле результат будет такой

| id | id |
|---|---|
| 1 | 1 |
| 1 | 1 |
| 1 | 1 |
| 1 | 1 |



# Почему визуализация с кругами плохая?

Вспомним inner join через cross join

```sql
SELECT
  u.id   AS user_id,
  u.name,
  o.id   AS order_id,
  o.amount
FROM users u, orders o
WHERE o.user_id = u.id
ORDER BY u.id, o.id;
```



# Почему визуализация с кругами плохая?

Вспомним inner join через cross join

```
SELECT
  u.id   AS user_id,
  u.name,
  o.id   AS order_id,
  o.amount
FROM users u, orders o
WHERE o.user_id = u.id
ORDER BY u.id, o.id;
```

Но СУБД не обязательно будет идти по пути CROSS JOIN + WHERE



# Почему визуализация с кругами плохая?

```
SELECT *
FROM table1
LEFT JOIN table2 on table1.id = table2.id
```

**table1**

| id |
|---|
| 1 |
| 1 |
| 3 |

**table2**

| id |
|---|
| 1 |
| 1 |
| 4 |
| 5 |

Сколько строк будет в результате?



# Почему визуализация с кругами плохая?

| id | id |
|---|---|
| 1 | 1 |
| 1 | 1 |
| 1 | 1 |
| 1 | 1 |
| 3 | |

# Почему визуализация с кругами плохая?

| id | id |
|---|---|
| 1 | 1 |
| 1 | 1 |
| 1 | 1 |
| 1 | 1 |
| 3 | |

LEFT JOIN - это тоже самое что и INNER JOIN, и плюс еще записи из левой таблицы, для которых в правой по этому фильтру ничего не совпало.

# Почему визуализация с кругами плохая?



```
SELECT *
FROM t1
CROSS JOIN t2
WHERE t1.id = t2.id

UNION ALL

SELECT t1.id, null
FROM t1
WHERE NOT EXISTS (
  SELECT id
  FROM t2
  WHERE t2.id = t1.id
)
```

# Но как удобнее визуализировать?

Пусть есть две таблицы

**table1**

| id |
|---|
| 1 |
| 1 |
| 6 |
| 5 |

**table2**

| id |
|---|
| 1 |
| 1 |
| 2 |
| 3 |
| 5 |



# Но как удобнее визуализировать?

Пусть есть две таблицы

**table1**

| id |
|---|
| 1 |
| 1 |
| 6 |
| 5 |

**table2**

| id |
|---|
| 1 |
| 1 |
| 2 |
| 3 |
| 5 |

```
SELECT t1.id, t2.id
FROM t1
CROSS JOIN t2
```

# Но как удобнее визуализировать?

| | **1** | **1** | **2** | **3** | **5** |
|---|---|---|---|---|---|
| **1** | 1,1 | 1,1 | 1,2 | 1,3 | 1,5 |
| **1** | 1,1 | 1,1 | 1,2 | 1,3 | 1,5 |
| **6** | 6,1 | 6,1 | 6,2 | 6,3 | 6,5 |
| **5** | 5,1 | 5,1 | 5,2 | 5,3 | 5,5 |

# Но как удобнее визуализировать?

| | 1 | 1 | 2 | 3 | 5 |
|---|---|---|---|---|---|
| **1** | 1,1 | 1,1 | 1,2 | 1,3 | 1,5 |
| **1** | 1,1 | 1,1 | 1,2 | 1,3 | 1,5 |
| **6** | 6,1 | 6,1 | 6,2 | 6,3 | 6,5 |
| **5** | 5,1 | 5,1 | 5,2 | 5,3 | 5,5 |

# А как быть с INNER JOIN?

```
SELECT t1.id, t2.id
FROM t1
INNER JOIN t2 ON t1.id = t2.id
```

| | **1** | **1** | **2** | **3** | **5** |
|---|---|---|---|---|---|
| **1** | 1,1 | 1,1 | 1,2 | 1,3 | 1,5 |
| **1** | 1,1 | 1,1 | 1,2 | 1,3 | 1,5 |
| **6** | 6,1 | 6,1 | 6,2 | 6,3 | 6,5 |
| **5** | 5,1 | 5,1 | 5,2 | 5,3 | 5,5 |

# А как быть с LEFT JOIN?

```
SELECT t1.id, t2.id
FROM t1
LEFT JOIN t2 ON t1.id = t2.id
```

| | **1** | **1** | **2** | **3** | **5** | |
|---|---|---|---|---|---|---|
| **1** | 1,1 | 1,1 | 1,2 | 1,3 | 1,5 | |
| **1** | 1,1 | 1,1 | 1,2 | 1,3 | 1,5 | |
| **6** | 6,1 | 6,1 | 6,2 | 6,3 | 6,5 | 6, null |
| **5** | 5,1 | 5,1 | 5,2 | 5,3 | 5,5 | |

# А как быть с RIGHT JOIN?

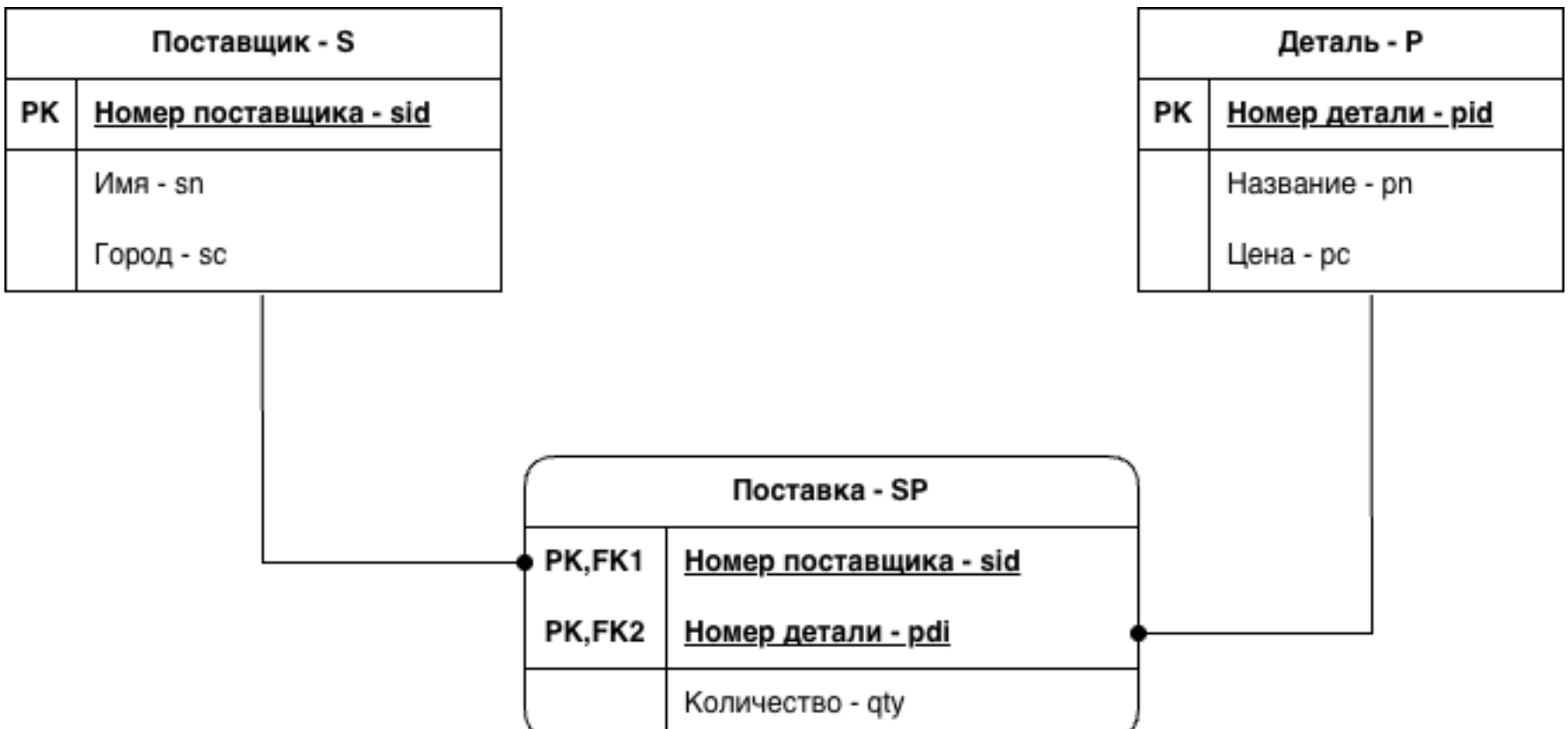
```
SELECT t1.id, t2.id
FROM t1
RIGHT JOIN t2 ON t1.id = t2.id
```

| | 1 | 1 | 2 | 3 | 5 |
|---|---|---|---|---|---|
| **1** | 1,1 | 1,1 | 1,2 | 1,3 | 1,5 |
| **1** | 1,1 | 1,1 | 1,2 | 1,3 | 1,5 |
| **6** | 6,1 | 6,1 | 6,2 | 6,3 | 6,5 |
| **5** | 5,1 | 5,1 | 5,2 | 5,3 | 5,5 |
| | | | null, 2 | null, 3 | |



# Примеры запросов





# Примеры запросов



1 Получить имена поставщиков, поставляющих деталь с номером P1

**S**

| SId | SN | SC |
|---|---|---|
| S1 | Smith | London |
| S2 | Jones | Paris |
| S3 | Clark | Paris |
| S4 | Adams | London |
| S5 | Black | Athens |

**SP**

| SId | PId | Qty |
|---|---|---|
| S1 | P1 | 100 |
| S1 | P2 | 150 |
| S2 | P1 | 200 |
| S3 | P1 | 110 |
| S3 | P2 | 20 |
| S3 | P3 | 180 |
| S5 | P2 | 140 |

# Примеры запросов



1 Получить имена поставщиков, поставляющих деталь с номером P1

**S**

| SId | SN | SC |
|---|---|---|
| S1 | Smith | London |
| S2 | Jones | Paris |
| S3 | Clark | Paris |
| S4 | Adams | London |
| S5 | Black | Athens |

**SP**

| SId | PId | Qty |
|---|---|---|
| S1 | P1 | 100 |
| S1 | P2 | 150 |
| S2 | P1 | 200 |
| S3 | P1 | 110 |
| S3 | P2 | 20 |
| S3 | P3 | 180 |
| S5 | P2 | 140 |

# Примеры запросов

```
SELECT SN as "Имя поставщика"
FROM S
JOIN SP ON S.SId = SP.SId
WHERE PId = 'P1'
```

| Имя поставщика |
| --- |
| Smith |
| Jones |
| Clark |

# Примеры запросов



2 Получить номера и имена поставщиков, не поставляющих деталь с номеров P1

**S**

| SId | SN | SC |
|---|---|---|
| S1 | Smith | London |
| S2 | Jones | Paris |
| S3 | Clark | Paris |
| S4 | Adams | London |
| S5 | Black | Athens |

**SP**

| SId | PId | Qty |
|---|---|---|
| S1 | P1 | 100 |
| S1 | P2 | 150 |
| S2 | P1 | 200 |
| S3 | P1 | 110 |
| S3 | P2 | 20 |
| S3 | P3 | 180 |
| S5 | P2 | 140 |

# Примеры запросов

```
SELECT SId, SN
FROM S
WHERE SId NOT IN (
  SELECT SId
  FROM SP
  WHERE PId = 'P1'
)
```

| SId |
|---|
| S1 |
| S2 |
| S3 |

| SId | SN |
|---|---|
| S4 | Adams |
| S5 | Black |

# Примеры запросов

```
SELECT S.SId, S.SN
FROM S
JOIN (
  SELECT SId
  FROM S
  EXCEPT
  SELECT SId
  FROM SP
  WHERE PId = 'P1'
) as T
ON S.SId = T.SId
```



# Примеры запросов



3 Получить имена поставщиков, поставляющих только деталь с номером P1

**S**

| SId | SN | SC |
|---|---|---|
| S1 | Smith | London |
| S2 | Jones | Paris |
| S3 | Clark | Paris |
| S4 | Adams | London |
| S5 | Black | Athens |

**SP**

| SId | PId | Qty |
|---|---|---|
| S1 | P1 | 100 |
| S1 | P2 | 150 |
| S2 | P1 | 200 |
| S3 | P1 | 110 |
| S3 | P2 | 20 |
| S3 | P3 | 180 |
| S5 | P2 | 140 |

# Примеры запросов

```
SELECT SId, SN
FROM S
JOIN SP ON S.SId = SP.SId
WHERE PId = 'P1'
AND S.SId NOT IN (
  SELECT SId
  FROM SP
  WHERE PId != 'P1'
)
```

| SId |
|---|
| S1 |
| S2 |
| S3 |

| SId |
|---|
| S1 |
| S3 |
| S5 |

| SId | SN |
|---|---|
| S2 | Jones |

# Примеры запросов

4 Получить имена поставщиков, поставляющих все детали

**S**

| SId | SN | SC |
|---|---|---|
| S1 | Smith | London |
| S2 | Jones | Paris |
| S3 | Clark | Paris |
| S4 | Adams | London |
| S5 | Black | Athens |

**SP**

| SId | PId | Qty |
|---|---|---|
| S1 | P1 | 100 |
| S1 | P2 | 150 |
| S2 | P1 | 200 |
| S3 | P1 | 110 |
| S3 | P2 | 20 |
| S3 | P3 | 180 |
| S5 | P2 | 140 |

**P**

| PId | PN | PC |
|---|---|---|
| P1 | Nut | 10 |
| P2 | Bolt | 20 |
| P3 | Cam | 33 |

# Примеры запросов

```
SELECT S.SN
FROM S
WHERE NOT EXISTS (
  SELECT PId
  FROM P
  WHERE NOT EXISTS (
    SELECT SId
    FROM SP
    WHERE P.PId = SP.PId AND S.SId = SP.SId
  )
)
```

# Примеры запросов

```
SELECT S.SN
FROM S
WHERE NOT EXISTS (
  SELECT PId
  FROM P
  WHERE NOT EXISTS (
    SELECT SId
    FROM SP
    WHERE P.PId = SP.PId AND S.SId = SP.SId
  )
)
```

Для текущего P и S в таблице SP ищется запись. Если запись есть -> NOT EXISTS FALSE. Если записей нет, то NOT EXISTS TRUE.

# Примеры запросов

```
SELECT S.SN
FROM S
WHERE NOT EXISTS (
  SELECT PId
  FROM P
  WHERE NOT EXISTS (
    SELECT SId
    FROM SP
    WHERE P.PId = SP.PId AND S.SId = SP.SId
  )
)
```

Ищем детали, которые поставщик не поставляет. Если список существует, то NOT EXISTS FALSE. Если таких деталей нет, то TRUE.

# Примеры запросов

```
SELECT s.SN
FROM S s
WHERE NOT EXISTS (
  (SELECT p.PId FROM P p)
  EXCEPT
  (SELECT sp.PId FROM SP sp WHERE sp.SId = s.SId)
);
```

# Примеры запросов



**5** Получить имена поставщиков, поставляющих максимальное количество деталей

**S**

| SId | SN | SC |
|---|---|---|
| S1 | Smith | London |
| S2 | Jones | Paris |
| S3 | Clark | Paris |
| S4 | Adams | London |
| S5 | Black | Athens |

**SP**

| SId | PId | Qty |
|---|---|---|
| S1 | P1 | 100 |
| S1 | P2 | 150 |
| S2 | P1 | 200 |
| S3 | P1 | 110 |
| S3 | P2 | 20 |
| S3 | P3 | 180 |
| S5 | P2 | 140 |

# Примеры запросов

```
SELECT T1.Qty
FROM SP T1, SP T2
WHERE T1.Qty < T2.Qty
```

**T1 = SP**

| SId | PId | Qty |
|---|---|---|
| S1 | P1 | 100 |
| S1 | P2 | 150 |
| S2 | P1 | 200 |
| S3 | P1 | 110 |
| S3 | P2 | 20 |
| S3 | P3 | 180 |
| S5 | P2 | 140 |

**T2 = SP**

| SId | PId | Qty |
|---|---|---|
| S1 | P1 | 100 |
| S1 | P2 | 150 |
| S2 | P1 | 200 |
| S3 | P1 | 110 |
| S3 | P2 | 20 |
| S3 | P3 | 180 |
| S5 | P2 | 140 |

# Примеры запросов

```
SELECT DISTINCT T1.Qty
FROM SP T1, SP T2
WHERE T1.Qty < T2.Qty
```

**T2 = SP**

| T1.Qty |
|---|
| 100 |
| 150 |
| 110 |
| 20 |
| 180 |
| 140 |

# Примеры запросов

```
SELECT DISTINCT SN, Qty
FROM S JOIN SP on S.SId = SP.SId
WHERE Qty NOT IN (
  SELECT DISTINCT T1.Qty
  FROM SP T1, SP T2
  WHERE T1.Qty < T2.Qty
)
```

| SN | Qty |
|---|---|
| Jones | 200 |