



САМАРСКИЙ УНИВЕРСИТЕТ  
SAMARA UNIVERSITY

# Базы данных

## Лекция 6 SQL. Подзапросы. Соединения

Агафонов Антон Александрович  
к.т.н., доцент кафедры ГИИБ

Самара



- Подзапросы
  - Типы подзапросов
  - Использование подзапросов в **SELECT**, **INSERT**, **UPDATE**, **DELETE**
  - Использование предикатов **ALL**, **ANY**
- Соединения таблиц
  - **INNER JOIN**
  - **LEFT** | **RIGHT JOIN**
  - **UNION**





Инструкция **SELECT** допускает вложение в запрос неограниченного количества подзапросов, благодаря которым можно создавать весьма сложные многоярусные конструкции выборки данных.

- Внутренний подзапрос представляет собой также оператор **SELECT**, а кодирование его предложений подчиняется тем же правилам, что и основного оператора **SELECT**.
- Внешний оператор **SELECT** использует результат выполнения внутреннего оператора для определения содержания окончательного результата всей операции.
- Внутренние запросы могут быть помещены непосредственно после оператора сравнения (=, <, >, <=, >=, <>) в предложения **WHERE** и **HAVING** внешнего оператора **SELECT**.
- Внутренние операторы **SELECT** могут применяться также в операторах **INSERT**, **UPDATE** и **DELETE**.





Подзапрос – это инструмент создания временной таблицы, содержимое которой извлекается и обрабатывается внешним оператором.

К подзапросам применяются следующие правила и ограничения:

- **ORDER BY** не используется, хотя и может присутствовать во внешнем подзапросе;
- список в предложении **SELECT** состоит из имен отдельных столбцов или составленных из них выражений – за исключением случая, когда в подзапросе присутствует ключевое слово **EXISTS**;
- по умолчанию имена столбцов в подзапросе относятся к таблице, имя которой указано в предложении **FROM**. Однако допускается ссылка и на столбцы таблицы, указанной во фразе **FROM** внешнего запроса, для чего применяются квалифицированные имена столбцов (т.е. с указанием таблицы);
- если подзапрос является одним из двух операндов, участвующих в операции сравнения, то запрос должен указываться в правой части этой операции.





По типу результирующего набора данных можно выделить два типа подзапросов:

- *Скалярный подзапрос* возвращает единственное значение. Запрос может использоваться везде, где требуется указать единственное значение.
- *Табличный подзапрос* возвращает множество значений, т.е. значения одного или нескольких столбцов таблицы, размещенные в более чем одной строке. Запрос может использоваться везде, где допускается наличие таблицы.





Использование подзапросов:

- В выражении **SELECT**:
  - В качестве спецификации столбца в выражении **SELECT**
  - В качестве таблицы для выборки в выражении **FROM**
  - В условии в выражении **WHERE**
  - В условии в выражении **HAVING**
- В выражении **INSERT**
  - Для определения значения, которое вставляется в один из столбцов таблицы
- В выражении **UPDATE**
  - В качестве устанавливаемого значения после оператора **SET**
  - Как часть условия в выражении **WHERE**
- В выражении **DELETE**
  - Как часть условия в выражении **WHERE**





Таблица товаров:

```
CREATE TABLE products (  
    id INT NOT NULL AUTO_INCREMENT,  
    product_name VARCHAR(45) NOT NULL,  
    manufacturer VARCHAR(45) NOT NULL,  
    product_count INT DEFAULT 0,  
    price DECIMAL(10, 2) NOT NULL,  
    PRIMARY KEY (id)  
);
```

Таблица заказов:

```
CREATE TABLE orders(  
    id INT AUTO_INCREMENT PRIMARY KEY,  
    product_id INT NOT NULL,  
    product_count INT DEFAULT 1,  
    created_at DATE NOT NULL,  
    FOREIGN KEY (product_id) REFERENCES products(id) ON DELETE CASCADE  
);
```





```
INSERT INTO products (product_name, manufacturer, product_count, price)  
VALUES
```

```
( 'iPhone 13', 'Apple', 5, 115000 ),  
( 'iPhone 12', 'Apple', 7, 79900 ),  
( 'Galaxy S22', 'Samsung', 4, 88000 ),  
( 'Galaxy S21', 'Samsung', 3, 67000 ),  
( 'P50', 'Huawei', 1, 115000 ),  
( '12 Pro', 'Xiaomi', 3, 115000 );
```

Данные таблицы products

	id	product_name	manufacturer	product_count	price
	1	iPhone 13	Apple	5	115000.00
	2	iPhone 12	Apple	7	79900.00
	3	Galaxy S22	Samsung	4	88000.00
	4	Galaxy S21	Samsung	3	67000.00
	5	P50	Huawei	1	115000.00
	6	12 Pro	Xiaomi	3	115000.00







```
INSERT INTO orders (product_id, created_at, product_count)
VALUES
( (SELECT id FROM products WHERE product_name='Galaxy S22'), '2022-05-21', 2 ),
( (SELECT id FROM products WHERE product_name='iPhone 13'), '2022-05-23', 1 ),
( (SELECT id FROM Products WHERE product_name='P50'), '2022-05-21', 1 );
```

Данные таблицы orders

	id	product_id	product_count	created_at
	1	3	2	2022-05-21
	2	1	1	2022-05-23
	3	5	1	2022-05-21





Выборка товаров, имеющих максимальную цену

```
SELECT * FROM products  
WHERE price = (SELECT MAX(price) FROM products);
```

Результат запроса

	id	product_name	manufacturer	product_count	price
	1	iPhone 13	Apple	5	115000.00
	5	P50	Huawei	1	115000.00
	6	12 Pro	Xiaomi	3	115000.00



Выборка товаров, имеющих цену ниже средней

```
SELECT * FROM products  
WHERE price < (SELECT AVG(price) FROM products)  
ORDER BY price;
```

Результат запроса

	id	product_name	manufacturer	product_count	price
	4	Galaxy S21	Samsung	3	67000.00
	2	iPhone 12	Apple	7	79900.00
	3	Galaxy S22	Samsung	4	88000.00



Выборка заказов самого дорогого товара

```
SELECT * FROM orders
WHERE product_id IN (
    SELECT id FROM products
    WHERE price = (
        SELECT MAX(price) FROM products
    )
);
```

Результат запроса

	id	product_id	product_count	created_at
	2	1	1	2022-05-23
	3	5	1	2022-05-21



## Вложенные запросы. Оператор EXISTS

Оператор **EXISTS** проверяется, возвращает ли подзапрос какое-либо значение. Поскольку возвращение набора строк не происходит, то подзапросы с подобным оператором выполняются довольно быстро.

**WHERE [NOT] EXISTS** (<подзапрос>)

Выборка товаров, для которых существуют заказы

```
SELECT * FROM products
WHERE EXISTS (
    SELECT * FROM orders
    WHERE orders.product_id = products.id
);
```

Результат запроса

	id	product_name	manufacturer	product_count	price
	1	iPhone 13	Apple	3	115000.00
	3	Galaxy S22	Samsung	4	88000.00
	5	P50	Huawei	1	115000.00





Выборка товаров, для которых существуют заказы

```
SELECT * FROM products
WHERE id IN (
    SELECT product_id FROM orders
);
```

Результат запроса

	id	product_name	manufacturer	product_count	price
	1	iPhone 13	Apple	3	115000.00
	3	Galaxy S22	Samsung	4	88000.00
	5	P50	Huawei	1	115000.00



При выполнении подзапроса **IN** проверяется равенство некоторого значения, обрабатываемого основным запросом какому-то из значений, содержащихся в столбце результатов вложенного запроса. В SQL предусмотрены операторы, осуществляющие более сложные правила сравнения: предикаты **ALL** и **ANY** | **SOME**.

Предикат **ALL**

<сравниваемое значение> <оператор отношения> **ALL** (<подзапрос>)

Проверяемое значение поочередно сравнивается с каждым элементом, возвращаемым подзапросом:

- если все сравнения дают **TRUE**, то и вся проверка **ALL** возвратит **TRUE**
- если хотя бы один результат сравнения равен **FALSE**, то общим результатом также станет **FALSE**

Особенность проверки **ALL** заключается в том, что если вложенный подзапрос вернет пустой набор данных, то предикат **ALL** возвратит **TRUE**.



Выборка товаров, цена которых меньше чем у любого товары компании Apple

```
SELECT * FROM products
WHERE price < ALL (
    SELECT price FROM products
    WHERE manufacturer = 'Apple'
);
```

Результат запроса

	id	product_name	manufacturer	product_count	price
	4	Galaxy S21	Samsung	3	67000.00

Примеры работы предиката **ALL**:

- $x > \text{ALL}(1, 2)$  эквивалентно  $x > 2$
- $x < \text{ALL}(1, 2)$  эквивалентно  $x < 1$
- $x = \text{ALL}(1, 2)$  эквивалентно  $(x = 1) \text{ AND } (x = 2)$
- $x \neq \text{ALL}(1, 2)$  эквивалентно  $x \text{ NOT IN } (1, 2)$





Выборка товаров, цена которых меньше чем у любого товары компании Apple

```
SELECT * FROM products
WHERE price < (
    SELECT MIN(price) FROM products
    WHERE manufacturer = 'Apple'
);
```

Результат запроса

	id	product_name	manufacturer	product_count	price
	4	Galaxy S21	Samsung	3	67000.00



Предикат **ANY** ( **SOME** ):

<сравниваемое значение> <оператор отношения> **ANY** | **SOME** (<подзапрос>)

Проверяемое значение поочередно сравнивается с каждым элементом, возвращаемым подзапросом:

- если хотя бы одно из сравнений даст результат **TRUE**, то проверка также завершится с результатом **TRUE**
- иначе результат проверки – **FALSE**

Если вложенный подзапрос вообще не возвратит данные, то сравнение **ANY** (**SOME**) заканчивается значением **FALSE**.



Выборка товаров, цена которых меньше самого дорогого товара компании Apple

```
SELECT * FROM products
WHERE price < ANY (
    SELECT price FROM products
    WHERE manufacturer = 'Apple'
);
```

Результат запроса

	id	product_name	manufacturer	product_count	price
	2	iPhone 12	Apple	7	79900.00
	3	Galaxy S22	Samsung	4	88000.00
	4	Galaxy S21	Samsung	3	67000.00

Примеры работы предиката **ANY (SOME)** :

- $x > \text{ANY}(1, 2)$  эквивалентно  $x > 1$
- $x < \text{ANY}(1, 2)$  эквивалентно  $x < 2$
- $x = \text{ANY}(1, 2)$  эквивалентно  $x \text{ IN } (1, 2)$
- $x \neq \text{ANY}(1, 2)$  эквивалентно  $(x \neq 1) \text{ OR } (x \neq 2)$





По типу подзапросов:

- *Некоррелирующий подзапрос* не зависит от основного запроса. Некоррелирующий подзапрос выполняется один раз для всего внешнего запроса.
- *Коррелирующий подзапрос* (correlated subquery) – подзапрос, который содержит ссылку на столбцы из включающего его основного запроса. Коррелирующий подзапрос будет выполняться для каждой строки основного запроса, так как значения столбцов основного запроса будут меняться.



Выборка заказов с указанием информации о товаре

```
SELECT created_at, product_count,  
      (SELECT product_name  
       FROM products  
       WHERE products.id = orders.product_id) AS product  
FROM orders;
```

Результат запроса

	created_at	product_count	product
	2022-05-21	2	Galaxy S22
	2022-05-23	1	iPhone 13



Выборка товаров, стоимость которых выше средней цены товаров данного производителя

```
SELECT product_name,  
        manufacturer,  
        price,  
        (SELECT AVG(price) FROM products AS sub_products  
         WHERE sub_products.manufacturer = prods.manufacturer) AS avg_price  
FROM products AS prods  
WHERE price >  
        (SELECT AVG(price) FROM products AS sub_products  
         WHERE sub_products.manufacturer = prods.manufacturer);
```

Результат запроса

	product_name	manufacturer	price	avg_price
	iPhone 13	Apple	115000.00	97450.000000
	Galaxy S22	Samsung	88000.00	77500.000000



Использование подзапроса как часть условия WHERE

```
UPDATE orders
SET product_count = product_count + 2
WHERE product_id IN (
    SELECT id FROM products
    WHERE manufacturer = 'Apple'
);
```

Использование подзапроса после оператора SET

```
UPDATE products
SET product_count = product_count - (
    SELECT SUM(product_count) FROM orders
    WHERE orders.product_id = products.id
)
WHERE id = 1;
```





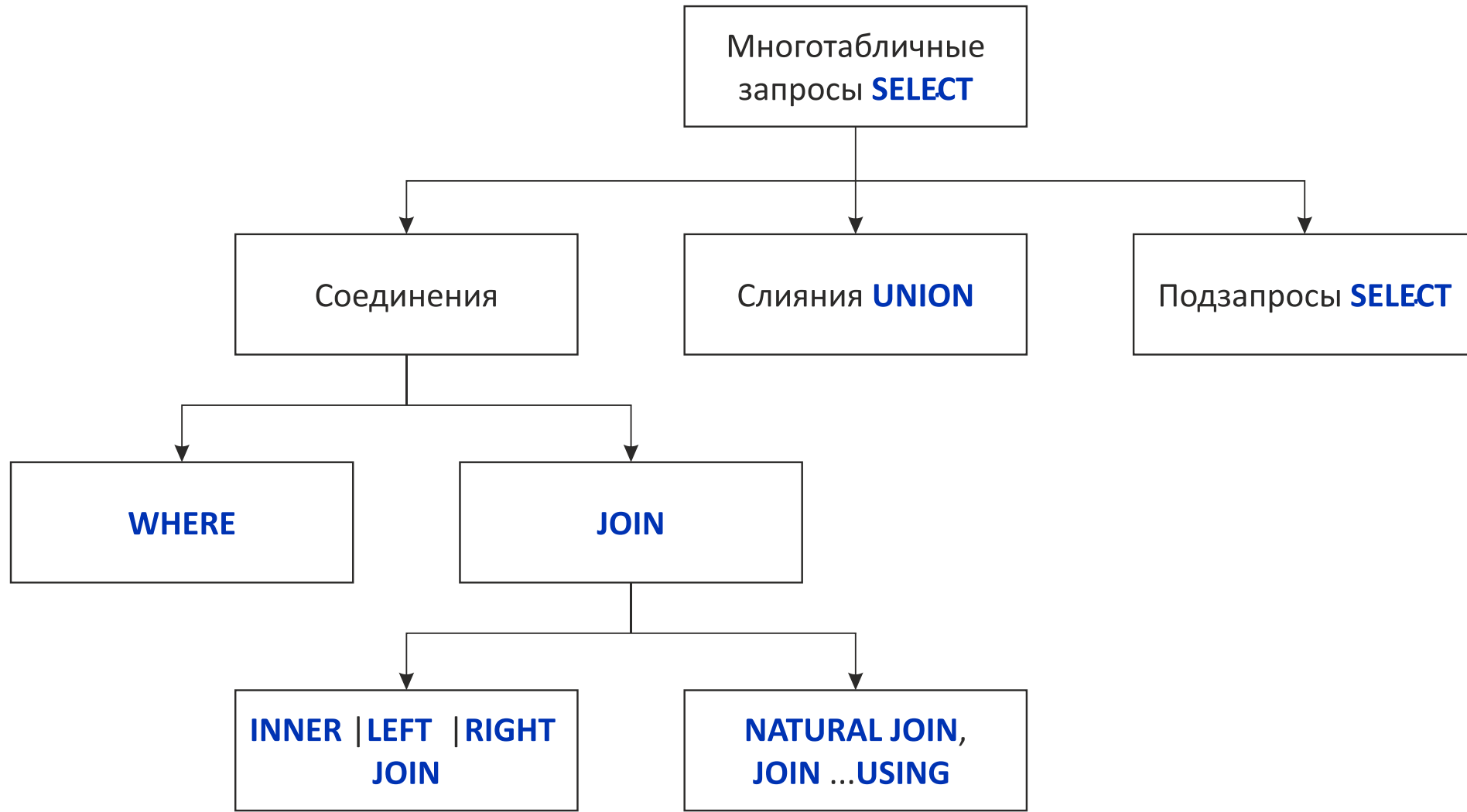
Использование подзапроса как часть условия WHERE

```
DELETE FROM orders
WHERE product_id = (
    SELECT id FROM products
    WHERE product_name = 'Galaxy S22'
);
```





# Классификация многотабличных запросов





## Запросы к нескольким таблицам

Таблица товаров:

```
CREATE TABLE products (  
    product_id INT NOT NULL AUTO_INCREMENT,  
    product_name VARCHAR(45) NOT NULL,  
    manufacturer VARCHAR(45) NOT NULL,  
    product_count INT DEFAULT 0,  
    price DECIMAL(10, 2) NOT NULL,  
    PRIMARY KEY (product_id)  
);
```

Таблица заказов:

```
CREATE TABLE orders(  
    id INT AUTO_INCREMENT PRIMARY KEY,  
    product_id INT NOT NULL,  
    customer_id INT NOT NULL,  
    product_count INT DEFAULT 1,  
    created_at DATE NOT NULL,  
    FOREIGN KEY (product_id) REFERENCES products(product_id) ON DELETE CASCADE,  
    FOREIGN KEY (customer_id) REFERENCES customers(customer_id) ON DELETE CASCADE  
);
```

Таблица клиентов:

```
CREATE TABLE customers (  
    customer_id INT NOT NULL  
        AUTO_INCREMENT,  
    first_name VARCHAR(30) NOT NULL,  
    PRIMARY KEY (customer_id)  
);
```





```
INSERT INTO products (product_name, manufacturer, product_count, price)
VALUES
( 'iPhone 13', 'Apple', 1, 115000 ),
( 'iPhone 12', 'Apple', 7, 79900 ),
( 'Galaxy S22', 'Samsung', 4, 88000 ),
( 'Galaxy S21', 'Samsung', 3, 67000 ),
( 'P50', 'Huawei', 1, 115000 ),
( '12 Pro', 'Xiaomi', 3, 115000 );
```





```
INSERT INTO customers (first_name)
VALUES
( 'Tom' ),
( 'Bob' ),
( 'Sam' );
```





```
INSERT INTO orders (product_id, customer_id, created_at, product_count)
VALUES
(
  (SELECT product_id FROM products WHERE product_name='Galaxy S22'),
  (SELECT customer_id FROM customers WHERE first_name='Tom'),
  '2022-05-21',
  2
),
(
  (SELECT product_id FROM products WHERE product_name='iPhone 13'),
  (SELECT customer_id FROM customers WHERE first_name='Tom'),
  '2022-05-23',
  1
),
(
  (SELECT product_id FROM Products WHERE product_name='iPhone 13'),
  (SELECT customer_id FROM customers WHERE first_name='Bob'),
  '2022-05-21',
  1
);
```





Данные таблицы products

	product_id	product_name	manufacturer	product_count	price
	1	iPhone 13	Apple	1	115000.00
	2	iPhone 12	Apple	7	79900.00
	3	Galaxy S22	Samsung	4	88000.00
	4	Galaxy S21	Samsung	3	67000.00
	5	P50	Huawei	1	115000.00
	6	12 Pro	Xiaomi	3	115000.00

Данные таблицы orders

	id	product_id	customer_id	product_count	created_at
	1	3	1	2	2022-05-21
	2	1	1	1	2022-05-23
	3	1	2	1	2022-05-21

Данные таблицы customers

	customer_id	first_name
	1	Tom
	2	Bob
	3	Sam



## Декартово произведение таблиц

Выборка данных из двух таблиц

```
SELECT *  
FROM orders, customers;
```

Результат запроса

	Id	product_id	customer_id	product_count	created_at	customer_id	first_name
	3	1	2	1	2022-05-21	1	Tom
	2	1	1	1	2022-05-23	1	Tom
	1	3	1	2	2022-05-21	1	Tom
	3	1	2	1	2022-05-21	2	Bob
	2	1	1	1	2022-05-23	2	Bob
	1	3	1	2	2022-05-21	2	Bob
	3	1	2	1	2022-05-21	3	Sam
	2	1	1	1	2022-05-23	3	Sam
	1	3	1	2	2022-05-21	3	Sam



Выборка данных из двух таблиц с фильтрацией в условии WHERE

```
SELECT *  
FROM orders, customers  
WHERE orders.customer_id = customers.customer_id;
```

Результат запроса

	Id	product_id	customer_id	product_count	created_at	customer_id	first_name
	1	3	1	2	2022-05-21	1	Tom
	2	1	1	1	2022-05-23	1	Tom
	3	1	2	1	2022-05-21	2	Bob





Выборка данных из трех таблиц с фильтрацией в условии WHERE

```
SELECT customers.first_name, products.product_name, orders.created_at
FROM orders, customers, products
WHERE orders.customer_id = customers.customer_id
      AND orders.product_id = products.product_id;
```

Результат запроса

	first_name	product_name	created_at
	Bob	iPhone 13	2022-05-21
	Tom	iPhone 13	2022-05-23
	Tom	Galaxy S22	2022-05-21



Использование псевдонимов таблиц

```
SELECT c.first_name, p.product_name, o.created_at
FROM orders AS o, customers c, products p
WHERE o.customer_id = c.customer_id
      AND o.product_id = p.product_id;
```

Результат запроса

	first_name	product_name	created_at
	Bob	iPhone 13	2022-05-21
	Tom	iPhone 13	2022-05-23
	Tom	Galaxy S22	2022-05-21



Выборка всех столбцов определенной таблицы

```
SELECT c.first_name, p.product_name, o.*  
FROM orders AS o, customers c, products p  
WHERE o.customer_id = c.customer_id  
      AND o.product_id = p.product_id;
```

Результат запроса

	first_name	product_name	id	product_id	customer_id	product_count	created_at
	Tom	Galaxy S22	1	3	1	2	2022-05-21
	Tom	iPhone 13	2	1	1	1	2022-05-23
	Bob	iPhone 13	3	1	2	1	2022-05-21



Внутреннее соединение:

```
SELECT имя_столбца [,...]  
FROM <левая_таблица>  
[INNER] JOIN <правая_таблица> [ON <правило_соединения>]  
[WHERE условия_отбора]  
...
```

Внешнее соединение:

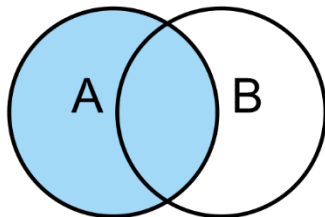
```
SELECT имя_столбца [,...]  
FROM <левая_таблица>  
{LEFT | RIGHT} JOIN <правая_таблица> [ON <правило_соединения>]  
[WHERE условия_отбора]  
...
```



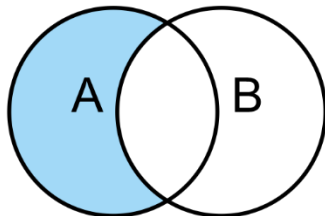
# Соединение JOIN

## SQL JOINS

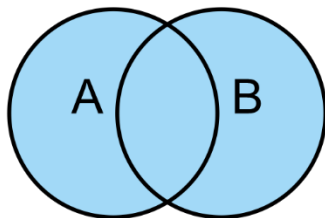
```
SELECT <select_list>  
FROM TableA A  
LEFT JOIN TableB B  
    ON A.Key = B.Key
```



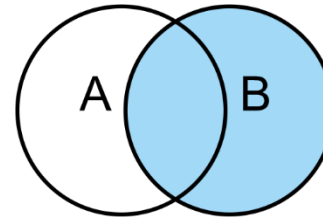
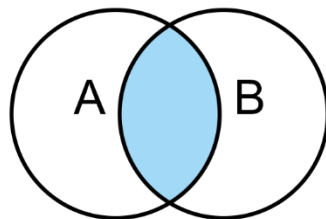
```
SELECT <select_list>  
FROM TableA A  
LEFT JOIN TableB B  
    ON A.Key = B.Key  
WHERE B.Key IS NULL
```



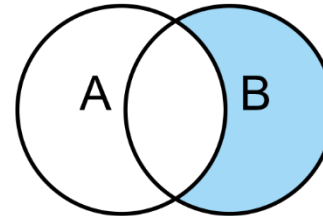
```
SELECT <select_list>  
FROM TableA A  
FULL OUTER JOIN TableB B  
    ON A.Key = B.Key
```



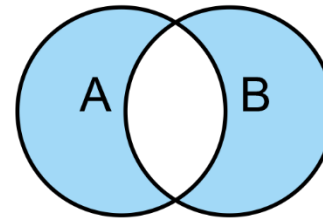
```
SELECT <select_list>  
FROM TableA A  
INNER JOIN TableB B  
    ON A.Key = B.Key
```



```
SELECT <select_list>  
FROM TableA A  
RIGHT JOIN TableB B  
    ON A.Key = B.Key
```



```
SELECT <select_list>  
FROM TableA A  
RIGHT JOIN TableB B  
    ON A.Key = B.Key  
WHERE A.Key IS NULL
```



```
SELECT <select_list>  
FROM TableA A  
FULL OUTER JOIN TableB B  
    ON A.Key = B.Key  
WHERE A.Key IS NULL  
    OR B.Key IS NULL
```





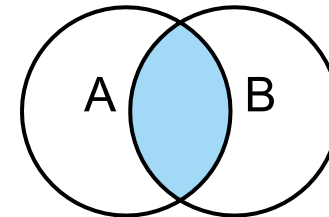
## Внутреннее соединение. INNER JOIN

Выборка заказов с информацией о товаре

```
SELECT orders.created_at, orders.product_count, products.product_name
FROM orders
JOIN products ON products.product_id = orders.product_id;
```

Результат запроса

	created_at	product_count	product_name
	2022-05-21	2	Galaxy S22
	2022-05-23	1	iPhone 13
	2022-05-21	1	iPhone 13





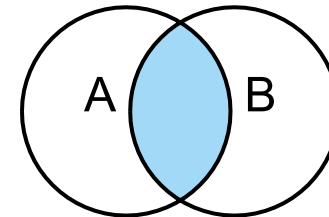
## Внутреннее соединение. INNER JOIN

Выборка заказов с информацией о товаре

```
SELECT o.created_at, o.product_count, products.product_name  
FROM orders o  
JOIN products p ON p.product_id = o.product_id;
```

Результат запроса

	created_at	product_count	product_name
	2022-05-21	2	Galaxy S22
	2022-05-23	1	iPhone 13
	2022-05-21	1	iPhone 13



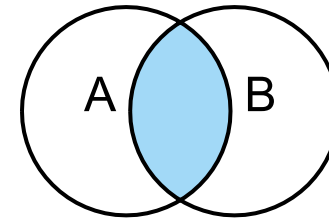


Выборка заказов с информацией о товаре

```
SELECT orders.created_at, orders.product_count, products.product_name
FROM orders
NATURAL JOIN products;
```

Результат запроса

	created_at	product_count	product_name
	2022-05-23	1	iPhone 13
	2022-05-21	1	iPhone 13





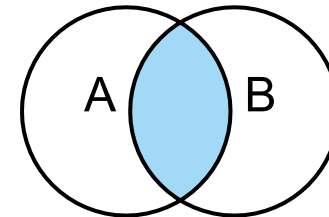


Выборка заказов с информацией о товаре

```
SELECT orders.created_at, orders.product_count, products.product_name
FROM orders
JOIN products USING (product_id);
```

Результат запроса

	created_at	product_count	product_name
	2022-05-21	2	Galaxy S22
	2022-05-23	1	iPhone 13
	2022-05-21	1	iPhone 13





Выборка заказов с информацией о товаре и клиенте

```
SELECT customers.first_name, products.product_name,  
        products.price, orders.created_at  
FROM products  
JOIN orders USING (product_id)  
JOIN customers ON customers.customer_id = orders.customer_id  
WHERE products.price > 90000  
ORDER BY orders.created_at DESC;
```

Результат запроса

	first_name	product_name	price	created_at
	Tom	iPhone 13	115000.00	2022-05-23
	Bob	iPhone 13	115000.00	2022-05-21

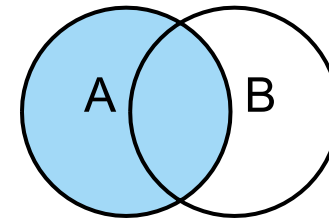


Выборка информации о **всех клиентах** и их заказах

```
SELECT first_name, created_at, product_count  
FROM customers  
LEFT JOIN orders ON orders.customer_id = customers.customer_id;
```

Результат запроса

	first_name	created_at	product_count
	Tom	2022-05-21	2
	Tom	2022-05-23	1
	Bob	2022-05-21	1
	Sam	NULL	NULL



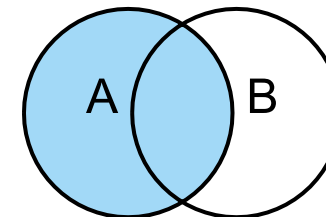


Выборка информации о **всех клиентах** и их заказах и товарах

```
SELECT first_name, created_at, o.product_count, product_name
FROM customers c
LEFT JOIN orders o USING (customer_id)
LEFT JOIN products p USING (product_id);
```

Результат запроса

	first_name	created_at	product_count	product_name
	Tom	2022-05-21	2	Galaxy S22
	Tom	2022-05-23	1	iPhone 13
	Bob	2022-05-21	1	iPhone 13
	Sam	NULL	NULL	NULL



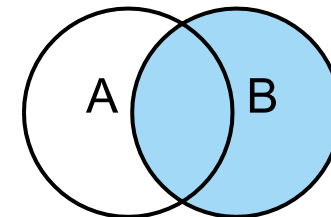


Выборка информации о **всех товарах** и заказах

```
SELECT orders.created_at, orders.product_count,  
       products.price, products.product_name  
FROM orders  
RIGHT JOIN products ON products.product_id = orders.product_id;
```

Результат запроса

	created_at	product_count	price	product_name
	2022-05-21	1	115000.00	iPhone 13
	2022-05-23	1	115000.00	iPhone 13
	NULL	NULL	79900.00	iPhone 12
	2022-05-21	2	88000.00	Galaxy S22
	NULL	NULL	67000.00	Galaxy S21
	NULL	NULL	115000.00	P50
	NULL	NULL	115000.00	12 Pro





## Объединение UNION

Оператор **UNION** предназначен для объединения результатов нескольких запросов

```
SELECT выражение_1  
UNION [ALL]  
SELECT выражение_2  
[UNION [ALL] SELECT выражение_N]
```



Таблица сотрудников:

```
CREATE TABLE employees (  
    id INT NOT NULL AUTO_INCREMENT,  
    first_name VARCHAR(30) NOT NULL,  
    PRIMARY KEY (id)  
);
```

```
INSERT INTO employees (first_name)  
VALUES  
( 'Tom' ),  
( 'Homer' ),  
( 'Rick' );
```





Выборка информации о всех сотрудниках и клиентах (без повторений)

```
SELECT first_name FROM customers  
UNION  
SELECT first_name FROM employees;
```

Результат запроса

	first_name
	Tom
	Bob
	Sam
	Homer
	Rick







Выборка информации о всех сотрудниках и клиентах

```
SELECT first_name FROM customers  
UNION ALL  
SELECT first_name FROM employees;
```

Результат запроса

	first_name
	Tom
	Bob
	Sam
	Tom
	Homer
	Rick





Выборка информации о всех сотрудниках и клиентах

```
SELECT customer_id, first_name FROM customers  
UNION ALL  
SELECT first_name FROM employees;
```

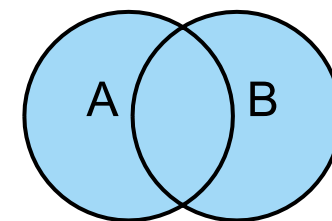




В ряде СУБД поддерживается оператор **FULL [OUTER] JOIN**, который осуществляет операцию полного внешнего соединения, возвращая все строки из правой и левой таблиц соединения.

В MySQL отдельного оператора для полного внешнего соединения не предусмотрено, однако оно может быть получено посредством объединения результатов левого и правого внешних соединений:

```
SELECT * FROM table_1  
LEFT JOIN table_2 ON table_1.id = table_2.id  
UNION ALL  
SELECT * FROM table_1  
RIGHT JOIN table_2 ON table_1.id = table_2.id;
```





**САМАРСКИЙ** УНИВЕРСИТЕТ  
SAMARA UNIVERSITY

**БЛАГОДАРЮ  
ЗА ВНИМАНИЕ**

Агафонов А.А.  
к.т.н., доцент кафедры ГИИБ