САМАРСКИЙ УНИВЕРСИТЕТ
SAMARA UNIVERSITY

Базы данных

Лекция 5
SQL. Операции манипулирования данными

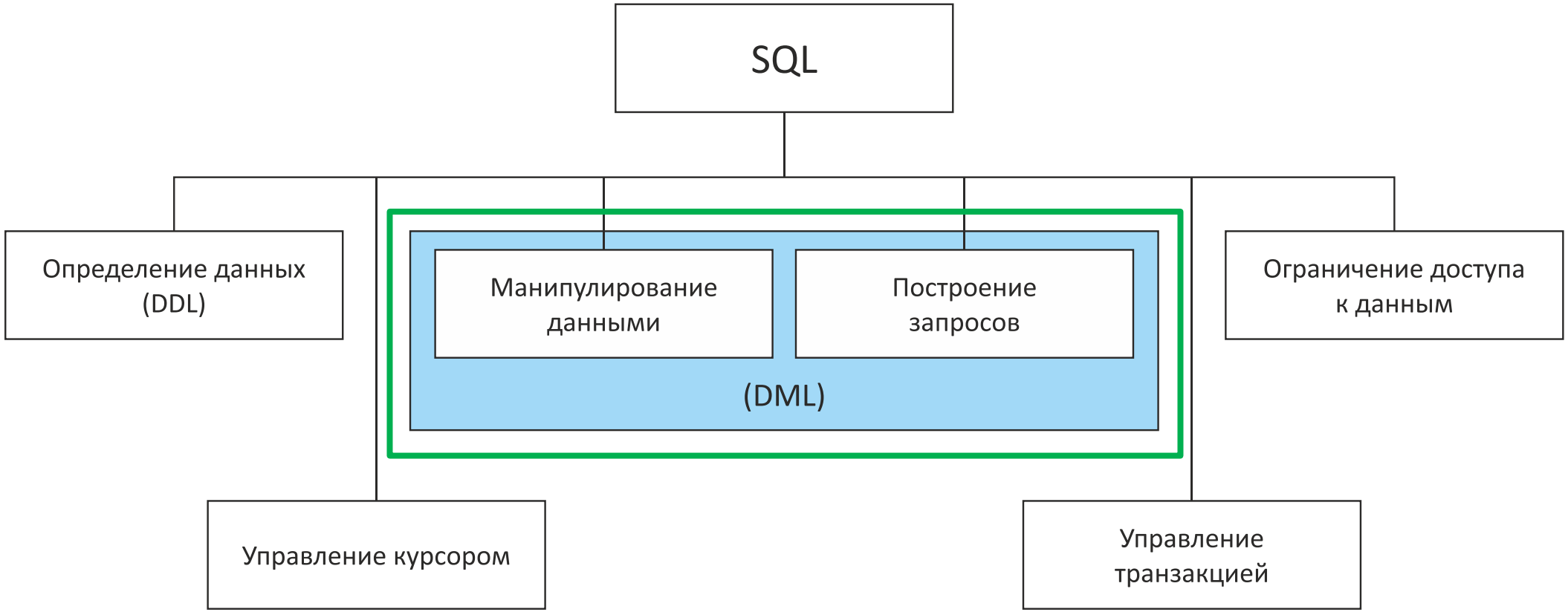
Агафонов Антон Александрович
д.т.н., доцент кафедры ГИиИБ

Самара

План лекции

- Операторы манипулирования данными (DML):
  - Выборка данных – **SELECT**
  - Добавление данных – **INSERT**
  - Изменение данных – **UPDATE**
  - Удаление данных – **DELETE**

Возможности SQL



Инструкция INSERT

Назначение инструкции **INSERT** – вставка в таблицу одной или нескольких строк.

```
INSERT INTO {имя_таблицы | имя_представления}
{имя_столбца [,…]}
{VALUES (значение_1 [,…]) | инструкция SELECT}
```

Инструкция INSERT. Примеры

```
CREATE DATABASE productsdb;
USE productsdb;

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)
);
```

Инструкция INSERT. Примеры

Вставка новой строки с помощью INSERT…INTO

```
INSERT INTO products
(product_name, manufacturer, product_count, price)
VALUES
('iPhone 13', 'Apple', 5, 94990);
```

Данные таблицы products

	id	product_name	manufacturer	product_count	price
	1	iPhone 13	Apple	5	94990.00

Инструкция INSERT. Примеры

Вставка новой строки с помощью INSERT…INTO

```
INSERT INTO products
(product_name, manufacturer, price)
VALUES
('iPhone 12', 'Apple', 79900);
```

Данные таблицы products

	id	product_name	manufacturer	product_count	price
	1	iPhone 13	Apple	5	94990.00
	2	iPhone 12	Apple	0	79900.00

Инструкция INSERT. Примеры

Вставка новой строки с помощью INSERT…INTO

```
INSERT INTO products (product_name, manufacturer, product_count, price)
VALUES ('Galaxy S22', 'Samsung', DEFAULT, 88000);
```

```
INSERT INTO products (product_name, manufacturer, price, product_count)
VALUES ('Galaxy S22', 'Samsung', 88000, NULL);
```

Данные таблицы products

	id	product_name	manufacturer	product_count	price
	1	iPhone 13	Apple	5	94990.00
	2	iPhone 12	Apple	0	79900.00
	3	Galaxy S22	Samsung	NULL	88000.00

Инструкция INSERT. Множественное добавление

Вставка нескольких строк с помощью INSERT…INTO

```
INSERT INTO products (product_name, manufacturer, product_count, price)
VALUES
('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	94990.00
	2	iPhone 12	Apple	0	79900.00
	3	Galaxy S22	Samsung	NULL	88000.00
	4	Galaxy S21	Samsung	3	67000.00
	5	P50	Huawei	1	115000.00
	6	12 Pro	Xiaomi	3	115000.00

Инструкция INSERT. Множественное добавление

Вставка записей с помощью INSERT…SELECT

```
INSERT INTO products
(product_name, manufacturer, price, product_count)
SELECT product_name, manufacturer, price, product_count
FROM products_2;
```

Назначение инструкции **SELECT** – выборка данных из одной или нескольких таблиц или представлений.

```
SELECT [DISTINCT|ALL]
 { имя_столбца [AS псевдоним] [,…]
 |функция_агрегирования [AS псевдоним] [,…]
 |выражение_для_вычисления_значения [AS псевдоним] [,…]
 |спецификатор.* }
FROM
 {имя_таблицы [AS псевдоним] [,…]
 |имя_представления [AS псевдоним][,…]}
[WHERE условия_отбора]
[GROUP BY имя_столбца [,…]] [HAVING условие]
[ORDER BY имя_столбца [ASC | DESC] [,…]]
```

```
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)
);
```

Данные таблицы products

	id	product_name	manufacturer	product_count	price
	1	iPhone 13	Apple	5	94990.00
	2	iPhone 12	Apple	0	79900.00
	3	Galaxy S22	Samsung	NULL	88000.00
	4	Galaxy S21	Samsung	3	67000.00
	5	P50	Huawei	1	115000.00
	6	12 Pro	Xiaomi	3	115000.00

Выборка всех данных из таблицы

```
SELECT * FROM products;
```

Результат запроса

	id	product_name	manufacturer	product_count	price
	1	iPhone 13	Apple	5	94990.00
	2	iPhone 12	Apple	0	79900.00
	3	Galaxy S22	Samsung	NULL	88000.00
	4	Galaxy S21	Samsung	3	67000.00
	5	P50	Huawei	1	115000.00
	6	12 Pro	Xiaomi	3	115000.00

Инструкция SELECT. Примеры

Выборка отдельного столбца из таблицы

```
SELECT product_id FROM products;
```

Выборка нескольких столбцов из таблицы

```
SELECT product_name, manufacturer FROM products;
```

Результат запроса

	product_name	manufacturer
	iPhone 13	Apple
	iPhone 12	Apple
	Galaxy S22	Samsung
	Galaxy S21	Samsung
	P50	Huawei
	12 Pro	Xiaomi

Исключение дубликатов строк

```
SELECT manufacturer FROM products;

SELECT DISTINCT manufacturer FROM products;
```

Результат запроса без DISTINCT

	manufacturer
	Apple
	Apple
	Samsung
	Samsung
	Huawei
	Xiaomi

Результат запроса с DISTINCT

	manufacturer
	Apple
	Samsung
	Huawei
	Xiaomi

Запрос без указания таблицы

```
SELECT 2 + 2;
```

Выборка результатов операций

```
SELECT
    id,
    CONCAT('Товар: ', product_name, '. Производитель: ', manufacturer),
    product_count * price
FROM products;
```

Результат запроса

	id	CONCAT('Товар: ', product_name, '. Производитель: ', manufacturer)	product_count * price
	1	Товар: iPhone 13. Производитель: Apple	474950.00
	2	Товар: iPhone 12. Производитель: Apple	0.00
	3	Товар: Galaxy S22. Производитель: Samsung	NULL
	4	Товар: Galaxy S21. Производитель: Samsung	201000.00
	5	Товар: P50. Производитель: Huawei	115000.00
	6	Товар: 12 Pro. Производитель: Xiaomi	345000.00

Использование псевдонимов

```
SELECT
    id AS ProductID,
    CONCAT('Товар: ', product_name, '. Производитель: ', manufacturer) AS Title,
    product_count * price AS TotalSum
FROM products;
```

Результат запроса

	id	Title	TotalSum
	1	Товар: iPhone 13. Производитель: Apple	474950.00
	2	Товар: iPhone 12. Производитель: Apple	0.00
	3	Товар: Galaxy S22. Производитель: Samsung	NULL
	4	Товар: Galaxy S21. Производитель: Samsung	201000.00
	5	Товар: P50. Производитель: Huawei	115000.00
	6	Товар: 12 Pro. Производитель: Xiaomi	345000.00

Сортировка таблицы по одному столбцу

```
SELECT * 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	0	79900.00
	3	Galaxy S22	Samsung	NULL	88000.00
	1	iPhone 13	Apple	5	94990.00
	5	P50	Huawei	1	115000.00
	6	12 Pro	Xiaomi	3	115000.00

Сортировка таблицы с использованием вычисляемых выражений

```
SELECT product_name, product_count * price AS TotalSum
FROM products
ORDER BY TotalSum DESC;

SELECT product_name
FROM products
ORDER BY product_count * price DESC;
```

Результат запроса (1)

	product_name	TotalSum
	iPhone 13	474950.00
	12 Pro	345000.00
	Galaxy S21	201000.00
	P50	115000.00
	iPhone 12	0.00
	Galaxy S22	NULL

Результат запроса (2)

	product_name
	iPhone 13
	12 Pro
	Galaxy S21
	P50
	iPhone 12
	Galaxy S22

Сортировка таблицы по нескольким столбцам

```
SELECT product_name, manufacturer, price
FROM products
ORDER BY manufacturer ASC, product_name DESC;
```

Результат запроса

	product_name	manufacturer	price
	iPhone 13	Apple	94990.00
	iPhone 12	Apple	79900.00
	P50	Huawei	115000.00
	Galaxy S22	Samsung	88000.00
	Galaxy S21	Samsung	67000.00
	12 Pro	Xiaomi	115000.00

Инструкция SELECT. Условие отбора данных WHERE

Отбор строк данных выполняется с использованием предложения **WHERE**:

```
[WHERE условия_отбора]
```

Применяемые ограничения:

- Сравнение. Проводится сравнение результатов вычисления одного выражения с другим («<», «>», «<=», «>=», «=», «!=, <>»). Результат сравнения может принимать значения: TRUE, FALSE и UNKNOWN.
- Попадание в диапазон. Проверяется, попадает ли результат вычисления выражения в определенный диапазон значений.
- Соответствие шаблону. Проверяется, соответствует ли некоторое строковое значение заданному шаблону.
- Неопределенность. Содержит ли поле неопределенное значение NULL.
- Проверка существования. Проверяется факт существования значения в выходных результатах вложенных подзапросов к другим отношениям БД.

Инструкция SELECT. Сравнение

Выборка с использованием операции сравнения «=»

```
SELECT * FROM products
WHERE manufacturer = 'Samsung';
```

	id	product_name	manufacturer	product_count	price
	3	Galaxy S22	Samsung	NULL	88000.00
	4	Galaxy S21	Samsung	3	67000.00

Выборка с использованием операции сравнения «<»

```
SELECT * FROM products
WHERE product_count < 3;
```

	id	product_name	manufacturer	product_count	price
	2	iPhone 12	Apple	0	79900.00
	5	P50	Huawei	1	115000.00

Выборка данных из таблицы по составному условию

```
SELECT * FROM products
WHERE manufacturer = 'Samsung' OR (price > 70000 AND price < 100000)
ORDER BY price DESC;
```

Результат запроса

	id	product_name	manufacturer	product_count	price
	1	iPhone 13	Apple	5	94990.00
	3	Galaxy S22	Samsung	NULL	88000.00
	2	iPhone 12	Apple	0	79900.00
	4	Galaxy S21	Samsung	3	67000.00

Приоритет операций

Ключ	Описание
-, ~	Унарный минус, битовая инверсия.
^	Побитовое исключающее ИЛИ.
*, /, DIV, %, MOD	Умножение, деление, остаток от деления, целочисленное деление.
+, -	Сложение, вычитание.
<<, >>	Побитовые сдвиги.
&	Побитовое И.
|	Побитовое ИЛИ.
=, <>, <=,>=	Операции сравнения.
NOT	Логическое отрицание.
AND	Логическое И.
XOR	Логическое исключающее ИЛИ.
OR	Логическое ИЛИ.

Инструкция SELECT. Оператор IN

Оператор **IN** производит проверку, входит ли результат вычисления в заданное множество. Множество может описываться как поэлементно, так и быть результатом другого, вложенного подзапроса.

```
<значение> [NOT] IN (подзапрос SELECT | элементы_множества)
```

Выборка по вхождению элемента в множество

```
SELECT * FROM products
WHERE manufacturer IN ('Samsung', 'HTC', 'Huawei');
```

Результат запроса

	id	product_name	manufacturer	product_count	price
	3	Galaxy S22	Samsung	NULL	88000.00
	4	Galaxy S21	Samsung	3	67000.00
	5	P50	Huawei	1	115000.00

Инструкция SELECT. Оператор BETWEEN

Оператор **BETWEEN** производит проверку принадлежности диапазону значений.

```
<значение> [NOT] BETWEEN <начало_диапазона> AND <конец_диапазона>
```

Выборка по вхождению элемента в диапазон

```
SELECT * FROM products
WHERE price BETWEEN 70000 AND 100000;
```

Результат запроса

	id	product_name	manufacturer	product_count	price
	1	iPhone 13	Apple	5	94990.00
	2	iPhone 12	Apple	0	79900.00
	3	Galaxy S22	Samsung	NULL	88000.00

Инструкция SELECT. Оператор BETWEEN

Выборка по нахождению элементов вне диапазона

```
SELECT *, price * product_count AS totalSum FROM products
WHERE price * product_count NOT BETWEEN 200000 AND 500000;
```

Результат запроса

	Id	product_name	manufacturer	product_count	price	totalSum
	2	iPhone 12	Apple	0	79900.00	0.00
	5	P50	Huawei	1	115000.00	115000.00

Оператор **LIKE** проверяет соответствие строки шаблону.

```
<значение> [NOT] LIKE <шаблон_подстроки>
```

Для определения шаблона могут применяться ряд специальных символов подстановки:

- символ подчеркивания «_» используется вместо любого одиночного символа в выражении;
- символ процента «%» заменяет последовательность любых символов.

Выборка по шаблону

```
SELECT * FROM products
WHERE product_name LIKE 'i_h_ne%';
```

Результат запроса

	id	product_name	manufacturer	product_count	price
	1	iPhone 13	Apple	5	94990.00
	2	iPhone 12	Apple	0	79900.00

Выборка строк с неопределенным значением в столбце

```
SELECT * FROM products
WHERE product_count IS NULL;
```

Результат запроса

	id	product_name	manufacturer	product_count	price
	3	Galaxy S22	Samsung	NULL	88000.00

Выборка строк с определенными значениями в столбце

```
SELECT * FROM products
WHERE product_count IS NOT NULL;
```

Инструкция SELECT. Проверка на неопределенность

Выборка строк с неопределенным значением проверкой X = NULL

```
SELECT * FROM products
WHERE product_count = NULL;
```

	id	product_name	Manufacturer	product_count	price
	NULL	NULL	NULL	NULL	NULL

Выборка строк с определенными значениями проверкой X != NULL

```
SELECT * FROM products
WHERE product_count <> NULL;
```

	id	product_name	Manufacturer	product_count	price
	NULL	NULL	NULL	NULL	NULL

```
SELECT 1 = NULL, 1 <> NULL;
```

	1 = NULL	1 <> NULL
	NULL	NULL

Инструкция SELECT. Получение диапазона строк

Оператор **LIMIT** позволяет извлечь определенное количество строк:

```
MySQL: LIMIT {[offset,] row_count | row_count OFFSET offset}
PostgreSQL: LIMIT {row_count | ALL} [ OFFSET offset ]
```

Выборка 2-х записей, начиная с 4-й

```
SELECT * FROM products ORDER BY id
LIMIT 2 OFFSET 3;
```

```
SELECT * FROM products ORDER BY id
LIMIT 3, 2;
```

Результат запроса

	id	product_name	manufacturer	product_count	price
	4	Galaxy S21	Samsung	3	67000.00
	5	P50	Huawei	1	115000.00

Агрегатные функции

Ключ	Описание
COUNT	Возвращает количество строк. Тип возвращаемого значения – целое число.
AVG	Вычисляет среднее арифметическое для указанных элементов. Используется только для полей цифровых типов данных. Тип возвращаемого значения – вещественное число.
SUM	Вычисляет сумму значений. Применяется только для цифровых типов данных.
MAX	Возвращает наибольшее из всех значений.
MIN	Возвращает наименьшее из всех значений.

Выборка среднего значения в столбце

```
SELECT AVG(price) AS avg_price
FROM products;
```

Результат запроса

	avg_price
	93315.000000

Выборка среднего значения выражения с фильтрацией строк

```
SELECT AVG(price * product_count) AS avg_total_price
FROM products
WHERE manufacturer = 'Samsung';
```

Результат запроса

	avg_total_price
	201000.000000

Выборка количества строк в выборке

```
SELECT COUNT(*)
FROM products;
```

Результат запроса

	COUNT(*)
	6

Выборка количества строк по определенному столбцу

```
SELECT COUNT(product_count)
FROM products;
```

Результат запроса

	COUNT(product_count)
	5

Выборка минимального, максимального значения и суммы

```
SELECT MIN(price), MAX(price), SUM(price * product_count)
FROM products;
```

Результат запроса

	MIN(price)	MAX(price)	SUM(price * product_count)
	67000.00	115000.00	1135950.00

Выборка количества строк без повторений

```
SELECT COUNT(DISTINCT manufacturer) AS count
FROM products;
```

Результат запроса

	COUNT(*)
	4

Выборка количества всех строк

```
SELECT COUNT(ALL manufacturer) AS count
FROM products;
```

Результат запроса

	COUNT(*)
	6

Инструкция SELECT. Группировка данных GROUP BY

Предложение **GROUP BY** предназначено для осуществления группировки выходных строк.

GROUP BY имя_столбца_1 [, имя_столбца_2, …]

- В группирующих запросах часто применяются агрегатные функции
- В результате выполнения группирующего запроса для каждой отдельной группы создается единственная группирующая строка
- Все имена столбца, приведенные в списке группирующего запроса **SELECT**, должны присутствовать и во фразе **GROUP BY**. Единственное исключение делается для полей, обрабатываемых агрегатной функцией

Инструкция SELECT. Группировка данных GROUP BY

Выборка с группировкой

```
SELECT manufacturer, COUNT(*) AS count
FROM products
GROUP BY manufacturer;
```

Результат запроса

	manufacturer	count
	Apple	2
	Samsung	2
	Huawei	1
	Xiaomi	1

Выборка с группировкой

```
SELECT manufacturer, COUNT(*) AS count
FROM products;
```

Выборка с группировкой

```
SELECT manufacturer, AVG(price) AS avg_price
FROM products
WHERE price < 100000
GROUP BY manufacturer
ORDER BY avg_price ASC;
```

Результат запроса

	manufacturer	avg_price
	Samsung	77500.000000
	Apple	87445.000000

Фильтрация групп выполняется с использованием предложения **HAVING**:

```
[HAVING условия_отбора]
```

Выборка с группировкой и фильтрацией

```
SELECT manufacturer, COUNT(*) AS models_count
FROM products
WHERE price * product_count < 500000
GROUP BY manufacturer
HAVING COUNT(*) > 1;
```

Результат запроса

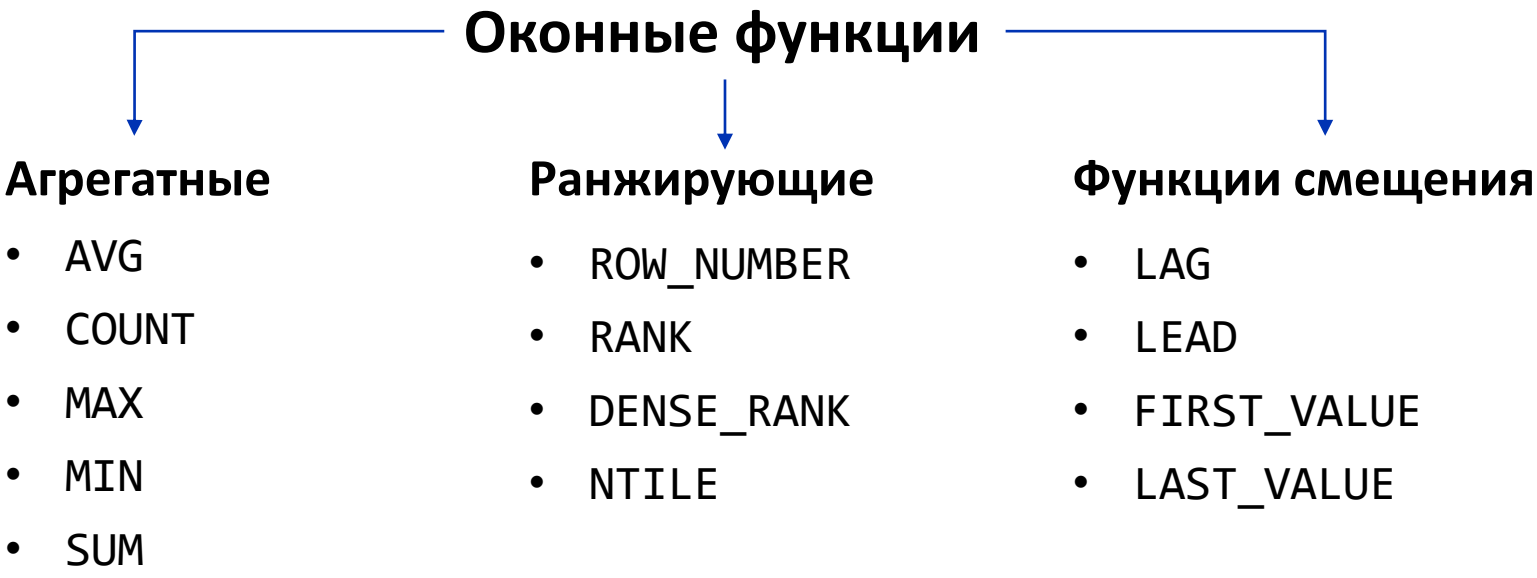
	manufacturer	models_count
	Apple	2

Инструкция SELECT. Оконные функции

Оконные функции используются для выполнения агрегатных вычислений по строкам, определенным в окне. В отличии от группировки с использованием **GROUP BY**, оконные функции **сохраняют все исходные строки в результирующем наборе данных**, добавляя к ним дополнительные вычисленные столбцы.

```
<оконная функция> (<столбец таблицы>)
OVER (
    [PARTITION BY <столбцы для разделения>]
    [ORDER BY <столбцы для сортировки>]
    [ROWS | RANGE <определение диапазона строк>]
)
```

- `<оконная функция> (<столбец таблицы>)` – используемая оконная функция
- **OVER** определяет окно (группу строк), которое будет передаваться в оконную функцию.
- **PARTITION BY** определяет, как строки будут разделены на группы (разделы, партиции) перед применением функции.
- **ORDER BY** устанавливает порядок строк внутри окна, особо важную роль играет в оконных функциях ранжирования.
- **ROWS | RANGE** формируют диапазоны строк (рамки окна). С помощью этого параметра можно указать, сколько строк до и после текущей брать в окно.



- **Агрегатные** функции выполняют арифметические вычисления на наборе данных и возвращают итоговое значение.
- **Ранжирующие** функции ранжируют значение для каждой строки в окне.
- Функции **смещения** позволяют перемещаться и обращаться к разным строкам в окне относительно текущей строки, а также обращаться к значениям в начале или в конце окна.

Оконные агрегатные функции

Агрегация с помощью **GROUP BY**

```
SELECT manufacturer, SUM(price) AS sum,
       COUNT(price) AS count,
       AVG(price) AS avg
FROM products
GROUP BY manufacturer;
```

Результат запроса

	manufacturer	sum	count	avg
	Apple	174890.00	2	87445.00
	Samsung	155000.00	2	77500.00
	Huawei	115000.00	1	115000.00
	Xiaomi	115000.00	1	115000.00

Агрегация с помощью **оконных функций**

```
SELECT product_name, manufacturer, price,
    SUM(price) OVER (PARTITION BY manufacturer) AS sum,
    COUNT(price) OVER (PARTITION BY manufacturer) AS count,
    AVG(price) OVER (PARTITION BY manufacturer) AS avg
FROM products;
```

Результат запроса

	product_name	manufacturer	price	sum	count	avg
	iPhone 13	Apple	94990.00	174890.00	2	87445.00
	iPhone 12	Apple	79900.00	174890.00	2	87445.00
	P50	Huawei	115000.00	115000.00	1	115000.00
	Galaxy S22	Samsung	88000.00	155000.00	2	77500.00
	Galaxy S21	Samsung	67000.00	155000.00	2	77500.00
	12 Pro	Xiaomi	115000.00	115000.00	1	115000.00

Оконные агрегатные функции

Кумулятивная агрегация с помощью **оконных функций**

```
SELECT product_name, manufacturer, price,
    SUM(price) OVER (ORDER BY product_name) AS sum,
    COUNT(price) OVER (ORDER BY product_name) AS count,
    AVG(price) OVER (ORDER BY product_name) AS avg
FROM products;
```

	product_name	manufacturer	price	sum	count	avg
	12 Pro	Xiaomi	115000.00	115000.00	1	115000.00
	Galaxy S21	Samsung	67000.00	182000.00	2	91000.00
	Galaxy S22	Samsung	88000.00	270000.00	3	90000.00
	iPhone 12	Apple	79900.00	349900.00	4	87475.00
	iPhone 13	Apple	94990.00	444890.00	5	88978.00
	P50	Huawei	115000.00	559890.00	6	93315.00

```
SELECT product_name, manufacturer, price,
    SUM(price) OVER (PARTITION BY manufacturer ORDER BY product_name) AS sum,
    COUNT(price) OVER (PARTITION BY manufacturer ORDER BY product_name) AS count,
    AVG(price) OVER (PARTITION BY manufacturer ORDER BY product_name) AS avg
FROM products;
```

Оконные ранжирующие функции

Функция	Описание
ROW_NUMBER	Присваивает уникальный номер каждой строке в пределах каждого окна. Номера присваиваются на основании указанного порядка строк.
RANK	Присваивает ранг каждой строке в пределах окна, при этом одинаковые значения получают одинаковый ранг, но с пропуском рангов для следующих строк.
DENSE_RANK	Работает аналогично функции RANK(), но без пропуска рангов.
NTILE(n)	Делит строки на заданное количество групп (n) и присваивает каждой строке номер группы.

Применение оконных ранжирующих функций

```
SELECT product_name, manufacturer,
    ROW_NUMBER() OVER (PARTITION BY manufacturer ORDER BY product_name) AS row_num,
    RANK() OVER (PARTITION BY manufacturer ORDER BY product_name) AS row_rank,
    DENSE_RANK() OVER (PARTITION BY manufacturer ORDER BY product_name) AS row_dense_rank
FROM products;
```

Результат запроса

	product_name	manufacturer	row_num	row_rank	row_dense_rank
	iPhone 12	Apple	1	1	1
	iPhone 13	Apple	2	2	2
	iPhone 13	Apple	3	2	2
	iPhone 14	Apple	4	4	3
	P50	Huawei	1	1	1
	Galaxy S21	Samsung	1	1	1
	Galaxy S22	Samsung	2	2	2
	12 Pro	Xiaomi	1	1	1

Оконные функции смещения

Функция	Описание
LAG (<столбец>, <смещение>, <значение по умолчанию>)	Обращается к данным из предыдущих строк окна. <столбец> - столбец, значение которого необходимо вернуть <смещение> - количество строк для смещения (по умолчанию 1) <значение по умолчанию> - значение, которое необходимо вернуть, если после смещения возвращается значение NULL.
LEAD (<столбец>, <смещение>, <значение по умолчанию>)	Обращается к данным из следующих строк
FIRST_VALUE(<столбец>)	Возвращает значение первой строки в окне.
LAST_VALUE(<столбец>)	Возвращает значение последней строки в окне.

Применение оконных функций смещения

```
SELECT product_name, manufacturer, price,
    LAG(price) OVER (PARTITION BY manufacturer ORDER BY product_name) AS row_lag,
    LEAD(price, 2, -1) OVER (PARTITION BY manufacturer ORDER BY product_name) AS row_lead,
    FIRST_VALUE(price) OVER (PARTITION BY manufacturer) AS row_first,
    LAST_VALUE(product_name) OVER (PARTITION BY manufacturer) AS row_last
FROM products;
```

	product_name	manufacturer	price	row_lag	row_lead	row_first	row_last
	iPhone 12	Apple	79900.00	NULL	90990.00	79900.00	iPhone 14
	iPhone 13	Apple	94990.00	79900.00	99990.00	79900.00	iPhone 14
	iPhone 13	Apple	90990.00	94990.00	-1.00	79900.00	iPhone 14
	iPhone 14	Apple	99990.00	90990.00	-1.00	79900.00	iPhone 14
	P50	Huawei	115000.00	NULL	-1.00	115000.00	P50
	Galaxy S21	Samsung	67000.00	NULL	-1.00	67000.00	Galaxy S22
	Galaxy S22	Samsung	88000.00	67000.00	-1.00	67000.00	Galaxy S22
	12 Pro	Xiaomi	115000.00	NULL	-1.00	115000.00	12 Pro

Оконные функции. Рамки окна

В контексте оконных функций SQL, «окно» определяет подмножество строк, которые рассматриваются SQL-функцией при выполнении вычислений.

Группа строк (партиция, `PARTITION BY`) – подмножество строк с одинаковыми значениями в одном или нескольких столбцах.
Окно определяет, какие конкретные строки в каждой группе будут использоваться для вычисления оконной функции для каждой строки.

Возможные определения границ окна

- `UNBOUNDED PRECEDING` – все строки, предшествующие текущей
- `N PRECEDING` – N строк до текущей строки
- `CURRENT ROW` – текущая строка
- `N FOLLOWING` – N строк после текущей строки
- `UNBOUNDED FOLLOWING` – все последующие строки

ROWS | RANGE

- `ROWS` – определение окна основывается на физическом положении строк относительно текущей строки. Окно точно ограничено количеством указанных строк.
- `RANGE` – определение окна основывается на значениях столбцов. Границы могут варьироваться в зависимости от данных, что делает окно гибким, но потенциально менее предсказуемым.

Рамки окна с использованием **ROWS**

```
SELECT product_name, price,
    SUM(price) OVER (
        ORDER BY product_name
        ROWS BETWEEN 2 PRECEDING
                 AND CURRENT ROW) AS sum
FROM products;
```

	product_name	price	sum
	12 Pro	115000.00	115000.00
	Galaxy S21	67000.00	182000.00
	Galaxy S22	88000.00	270000.00
	iPhone 12	79900.00	234900.00
	iPhone 13	94990.00	262890.00
	iPhone 13	90990.00	265880.00
	iPhone 14	99990.00	285970.00
	P50	115000.00	305980.00

Рамки окна с использованием **RANGE**

```
SELECT product_name, price,
    SUM(price) OVER (
        ORDER BY price
        RANGE BETWEEN CURRENT ROW
                  AND 10000 FOLLOWING) AS sum
FROM products;
```

	product_name	price	sum
	Galaxy S21	67000.00	67000.00
	iPhone 12	79900.00	167900.00
	Galaxy S22	88000.00	273980.00
	iPhone 13	90990.00	285970.00
	iPhone 13	94990.00	194980.00
	iPhone 14	99990.00	99990.00
	P50	115000.00	230000.00
	12 Pro	115000.00	230000.00

В окно попадают строки
`current price <= price <= current price + 10000`

Инструкция UPDATE

Назначение инструкции **UPDATE** – редактирование данных в таблице.

```
UPDATE {имя_таблицы | имя_представления}
SET {{имя_столбца
        = выражение | значение | NULL | DEFAULT | ARRAY | ROW=строка}[,…]}
[WHERE условие_отбора | WHERE CURRENT OF имя_курсора]
```

Упрощенный синтаксис:

```
UPDATE имя_таблицы
SET имя_столбца_1 = значение_1, имя_столбца_2 = значение_2, …,
    имя_столбца_N = значение_N
[WHERE условие_отбора]
```

Инструкция UPDATE. Примеры

Модификация значения столбца для всех строк таблицы

```
UPDATE products
SET price = price + 3000;
```

Модификация значения столбца по условию

```
UPDATE products
SET manufacturer = 'Samsung Inc.'
WHERE manufacturer = 'Samsung';
```

Инструкция UPDATE. Примеры

Модификация значений нескольких столбцов таблицы

```
UPDATE products
SET manufacturer = 'Samsung', product_count = product_count + 3
WHERE manufacturer = 'Samsung Inc.';
```

Модификация значения столбца с использованием значения по умолчанию

```
UPDATE products
SET product_count = DEFAULT
WHERE manufacturer = 'Samsung';
```

Инструкция DELETE

Назначение инструкции **DELETE** – удаление данных из таблиц.

```
DELETE FROM имя_таблицы
[WHERE условие_отбора | WHERE CURRENT OF имя_курсора]
```


Инструкция DELETE. Примеры

Удаление строк по условию

```
DELETE FROM products
WHERE manufacturer = 'Huawei';
```

Удаление строк по условию

```
DELETE FROM products
WHERE manufacturer = 'Apple' AND price < 90000;
```

Инструкция DELETE. Примеры

Удаление все строк таблицы

```
DELETE FROM products;
```

Удаление всех строк таблицы

```
TRUNCATE TABLE products;
```

САМАРСКИЙ УНИВЕРСИТЕТ
SAMARA UNIVERSITY

БЛАГОДАРЮ ЗА ВНИМАНИЕ

Агафонов А.А.
д.т.н., доцент кафедры ГИиИБ