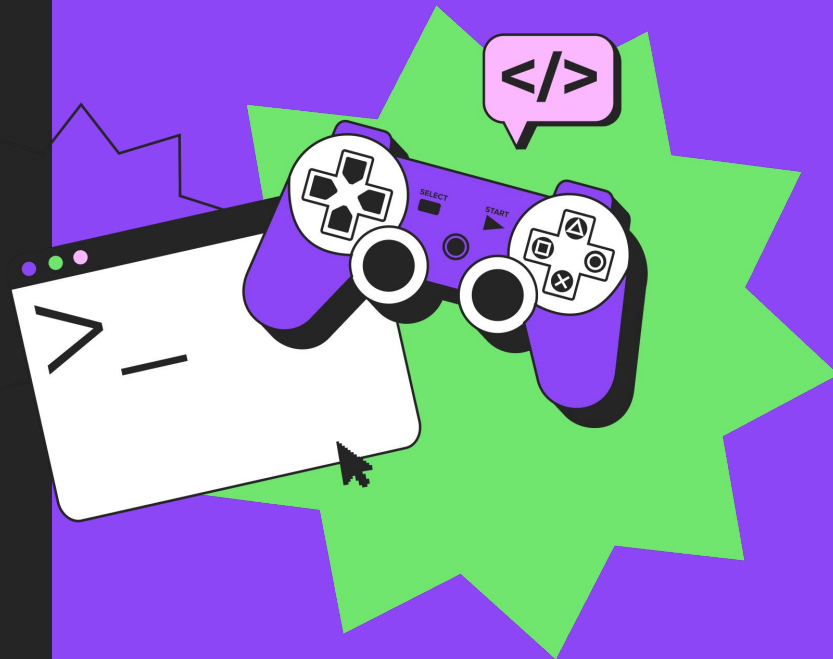


Базы данных и SQL

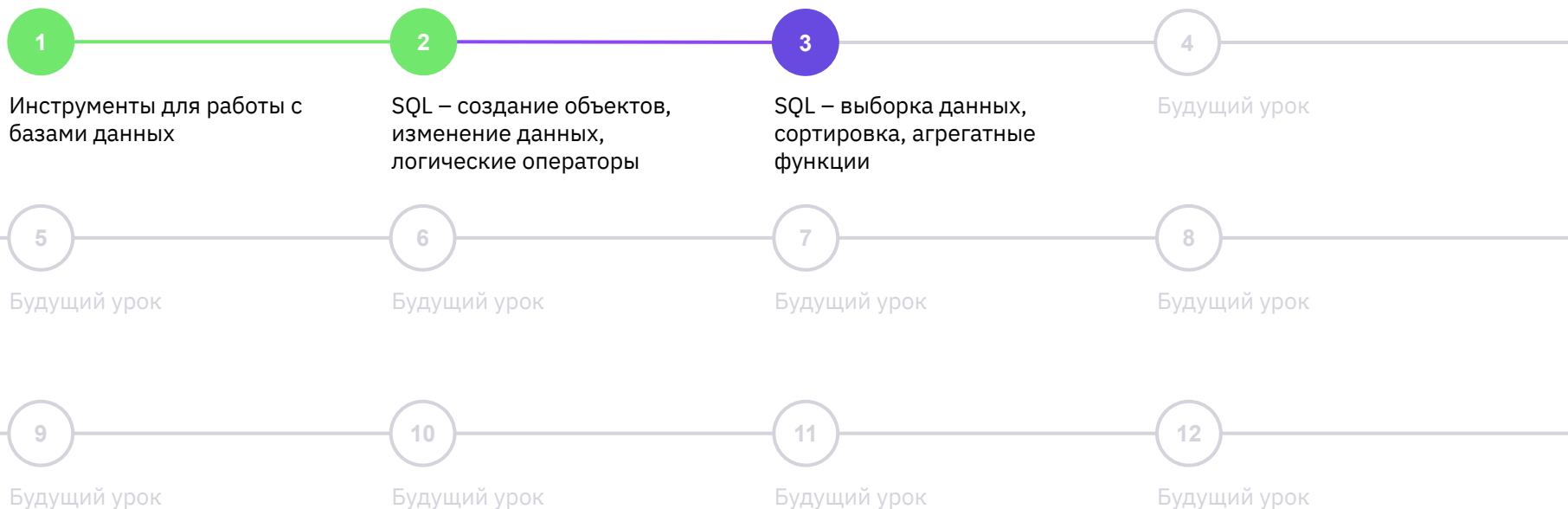
Урок 3

SQL – выборка данных, сортировка, агрегатные функции





План курса












Содержание урока



Что будет на уроке сегодня

-  Сортировка результатов запроса, сортировка по условиям
-  Ограничение выборки (top, limit, fetch),
-  Уникальные значения - distinct
-  Группировка — GROUP BY
-  Агрегатные функции — count, sum, avg, обработка Null
-  Использование Having
-  Приоритет операций



Сортировка результатов запроса: оператор ORDER BY

```
1 SELECT expressions
2 FROM tables
3 [WHERE conditions]
4 ORDER BY expression [ ASC | DESC ];
```



Просмотр созданных баз данных



expressions

Столбцы или расчеты, которые вы хотите получить



tables

Таблицы, из которых вы хотите получить записи



WHERE conditions

Необязательный. Условия, которые должны быть выполнены для записей, которые будут выбраны



ASC или DESC

Необязательный. Сортировка по возрастанию или по убыванию



Пример на сортировку с помощью ORDER BY:

```
1 SELECT * FROM Products
2 ORDER BY Price;
```

The screenshot shows a database management interface. At the top, a SQL query is entered in a text area:

```
1 • USE productsdb;
2
3 • SELECT * FROM Products
4   ORDER BY Price;
```

Below the query, a toolbar includes a 'Result Grid' button, a 'Filter Rows' input field, and an 'Edit' button. The 'Result Grid' is active, displaying a table of product data sorted by price in ascending order.

	Id	ProductName	Manufacturer	ProductCount	Price
	5	Honor 10	Huawei	3	26000
	4	Galaxy S8	Samsung	2	46000
	2	iPhone 8	Apple	3	56000
	3	Galaxy S9	Samsung	6	56000
	1	iPhone X	Apple	3	71000
	NULL	NULL	NULL	NULL	NULL



Использование псевдонима в запросе с ORDER BY:

```
1 SELECT ProductName, ProductCount * Price AS TotalSum
2 FROM Products
3 ORDER BY TotalSum;
```

The screenshot shows a SQL query editor with the following code:

```
1 USE productsdb;
2
3 SELECT ProductName, ProductCount * Price AS TotalSum
4 FROM Products
5 ORDER BY TotalSum;
```

Below the editor is a results grid with the following data:

ProductName	TotalSum
Honor 10	78000
Galaxy S8	92000
iPhone 8	168000
iPhone X	213000
Galaxy S9	336000



Использование сложного выражения в качестве критерия сортировки

```
1 SELECT Product Name, Price, Product Count
2 FROM Products
3 ORDER BY Product Count * Price;
```



Ограничение выборки в различных СУБД

Ключевое слово	Система баз данных
TOP	SQL Server, MS Access
LIMIT	MySQL, PostgreSQL, SQLite
FETCH FIRST	Oracle



Ограничение выборки: top

```
1 SELECT поля_выборки  
2 FROM список_таблиц  
3 LIMIT [количество_пропущенных_записей,] количество_записей_для_вывода;
```

Оператор LIMIT позволяет извлечь определенное количество строк и имеет следующий синтаксис:

```
1 LIMIT [offset,] rowcount
```



Пример использования: top

Например, выберем первые три строки:

```
1 SELECT * FROM Products
2 LIMIT 3;
```

Или, что то же самое:

```
1 SELECT * FROM Products
2 LIMIT 2, 3
```



Пример использования: top

```
1 • USE productsdb;
2
3 • SELECT * FROM Products
4   LIMIT 3;
```

<

Result Grid | Filter Rows: | Edit:

	Id	ProductName	Manufacturer	ProductCount	Price
	1	iPhone X	Apple	3	71000
	2	iPhone 8	Apple	3	56000
	3	Galaxy S9	Samsung	6	56000
	NULL	NULL	NULL	NULL	NULL

```
1 • USE productsdb;
2
3 • SELECT * FROM Products
4   LIMIT 2, 3;
```

<

Result Grid | Filter Rows: | Edit:

	Id	ProductName	Manufacturer	ProductCount	Price
	3	Galaxy S9	Samsung	6	56000
	4	Galaxy S8	Samsung	2	46000
	5	Honor 10	Huawei	3	26000
	NULL	NULL	NULL	NULL	NULL



Аналоги: извлечение диапазона строк в MS SQL Server

В примерах работаем с базой данных "Недвижимость" и её таблицей "Объект" (OBJECT).

Obj_ID	Type	District	Rooms
1	flat	Центр	2
2	flat	Центр	2
3	house	Волжский	4
4	flat	Центр	2
5	house	Волжский	5
6	flat	Пашино	2
7	flat	Центр	3
8	house	Сосновка	3



Аналоги: извлечение диапазона строк в MS SQL Server

Например, выберем первые три строки:

```
1 SELECT TOP 2 *  
2 FROM Object
```

При помощи применённого ограничения диапазона будет выведена следующая таблица:

Obj_ID	Type	District	Rooms
1	flat	Центр	2
2	flat	Центр	2



Ограничение выборки: fetch

Синтаксис оператора FETCH в MySQL:

```
1 SELECT ColumnNames FROM TableName ORDER BY ColumnName OFFSET  
   rows_to_be_skipped FETCH NEXT n ROWS ONLY;
```

Пример: будут извлекаться строки только от $(m+1)$ до $(m+1+p)$.

```
1 SELECT ColumnNames FROM TableName ORDER BY ColumnNames OFFSET m ROWS  
   FETCH NEXT p ROWS ONLY;
```




Уникальные значения - distinct

Пусть имеется таблица “Продукты”:

```
1 USE productsdb;
2
3 CREATE TABLE Products
4 (
5     Id INT AUTO_INCREMENT PRIMARY KEY,
6     ProductName VARCHAR(30) NOT NULL,
7     Manufacturer VARCHAR(20) NOT NULL,
8     ProductCount INT DEFAULT 0,
9     Price DECIMAL NOT NULL
10 );
11 INSERT INTO Products (ProductName, Manufacturer, ProductCount, Price)
12 VALUES
13 ('iPhone X', 'Apple', 3, 71000),
14 ('iPhone 8', 'Apple', 3, 56000),
15 ('Galaxy S9', 'Samsung', 6, 56000),
16 ('Galaxy S8', 'Samsung', 2, 46000),
17 ('Honor 10', 'Huawei', 3, 26000);
```



Уникальные значения - distinct: вывод уникальных производителей

Применим оператор DISTINCT для выборки уникальных значений - уникальных производителей:

```
1 SELECT DISTINCT Manufacturer FROM Products;
```

The screenshot shows a SQL IDE interface. The top pane contains the following SQL code:

```
1 • USE productsdb;  
2  
3 • SELECT DISTINCT Manufacturer FROM Products;
```

Below the code editor is a toolbar with icons for 'Result Grid', 'Filter Rows', 'Export', and 'Wrap'. The 'Result Grid' icon is active. Below the toolbar, a table displays the results of the query:

	Manufacturer
	Apple
	Samsung
	Huawei







Уникальные значения - distinct: вывод уникальных производителей по нескольким столбцам

```
1 SELECT DISTINCT Manufacturer, ProductCount FROM Products;
```

	Id	ProductName	Manufacturer	ProductCount	Price
▶	1	iPhone X	Apple	3	71000
	2	iPhone 8	Apple	3	56000
	3	Galaxy S9	Samsung	6	56000
	4	Galaxy S8	Samsung	2	46000
	5	Honor 10	Huawei	3	26000
•	NULL	NULL	NULL	NULL	NULL

1	•	USE productsdb;
2		
3	•	SELECT DISTINCT Manufacturer, ProductCount FROM Products;

<	
Result Grid	  Filter Rows: <input type="text"/>
Export: 	Wrap Cell Content: 

Manufacturer	ProductCount
Apple	3
Samsung	6
Samsung	2
Huawei	3



Группировка — GROUP BY

```
1 SELECT столбцы
2 FROM таблица
3 [WHERE условие_фильтрации_строк]
4 [GROUP BY столбцы_для_группировки]
5 [HAVING условие_фильтрации_групп]
6 [ORDER BY столбцы_для_сортировки]
```



Группировка — GROUP BY: пример

```
1 SELECT Manufacturer, COUNT(*) AS Models Count
2 FROM Products
3 GROUP BY Manufacturer
```

```
20 • SELECT Manufacturer, COUNT(*) AS ModelsCount
21     FROM Products
22     GROUP BY Manufacturer;
```

Result Grid			Filter Rows:	Export:	Wrap Cell Con
	Manufacturer	ModelsCount			
▶	Apple	2			
	Samsung	2			
	Huawei	1			



Агрегатные функции

В MySQL есть следующие агрегатные функции:

AVG: вычисляет среднее значение

SUM: вычисляет сумму значений

MIN: вычисляет наименьшее значение

MAX: вычисляет наибольшее значение

COUNT: вычисляет количество строк в запросе



Агрегатные функции: AVG

```
1 CREATE TABLE Products
2 (
3     Id INT AUTO_INCREMENT PRIMARY KEY,
4     ProductName VARCHAR(30) NOT NULL,
5     Manufacturer VARCHAR(20) NOT NULL,
6     ProductCount INT DEFAULT 0,
7     Price DECIMAL NOT NULL
8 );
9
10 INSERT INTO Products(ProductName, Manufacturer, ProductCount, Price)
11 VALUES
12 ('iPhone X', 'Apple', 3, 76000),
13 ('iPhone 8', 'Apple', 2, 51000),
14 ('iPhone 7', 'Apple', 5, 32000),
15 ('Galaxy S9', 'Samsung', 2, 56000),
16 ('Galaxy S8', 'Samsung', 1, 46000),
17 ('Honor 10', 'Huawei', 5, 28000),
18 ('Nokia 8', 'HMD Global', 6, 38000)
```



Агрегатные функции: AVG

Найдем среднюю цену товаров из базы данных:

```
1 SELECT AVG(Price) AS Average_Price FROM Products
```

```
1 • USE productsdb;
2
3 • SELECT AVG(Price) AS Average_Price FROM Products
```

<

Result Grid | | Filter Rows: | Export: | Wrap Cell Content:

Average_Price
46714.2857



Агрегатные функции: AVG с использованием фильтрации

```
1 --На этапе выборки можно применять фильтрацию. Например, найдем среднюю
  --цену для товаров определенного производителя:
2
3 SELECT AVG(Price) FROM Products
4 WHERE Manufacturer='Apple';
```

```
20 • SELECT AVG(Price) FROM Products
21 WHERE Manufacturer='Apple';
```

Result Grid			Filter Rows: <input type="text"/>
	AVG(Price)		
▶	63500.0000		



Агрегатные функции: COUNT

```
1 SELECT COUNT(*) FROM Products;
```

1	•	USE productsdb;
2		
3	•	SELECT COUNT(*) FROM Products

<

Result Grid Filter Rows:

	COUNT(*)
	7



Агрегатные функции: Min и Max

```
1 SELECT MIN(Price), MAX(Price) FROM Products;
```

The screenshot shows a SQL query editor with three lines of code. Line 1 is 'USE productsdb;', line 2 is empty, and line 3 is 'SELECT MIN(Price), MAX(Price) FROM Products'. Below the editor is a results window with a toolbar containing 'Result Grid', a grid icon, a refresh icon, 'Filter Rows:' with an input field, 'Export:' with a document icon, and 'Wrap'. The results window displays a table with two columns: 'MIN(Price)' and 'MAX(Price)'. The first row contains the values '28000' and '76000'.

	MIN(Price)	MAX(Price)
	28000	76000



Использования HAVING

```
1 SELECT expression1, expression2, ... expression_n,  
2     aggregate_function (expression)  
3 FROM tables  
4 [WHERE conditions]  
5 GROUP BY expression1, expression2, ... expression_n
```

`aggregate_function` - функция, такая как функции `SUM`, `COUNT`, `MIN`, `MAX` или `AVG`.

`expression1, expression2, ... expression_n` - выражения, которые не заключены в агрегированную функцию и должны быть включены в предложение `GROUP BY`.

`WHERE conditions` - необязательный. Это условия для выбора записей.

`HAVING condition` - Это дополнительное условие применяется только к агрегированным результатам для ограничения групп возвращаемых строк.



Использования HAVING

Например, найдем все группы товаров по производителям, для которых определено более 1 модели:

```
1 SELECT Manufacturer, COUNT(*) AS ModelsCount
2 FROM Products
3 GROUP BY Manufacturer
4 HAVING COUNT(*) > 1
```

The screenshot shows a SQL IDE interface. The top pane contains the following SQL code:

```
1 • USE productsdb;
2
3 • SELECT Manufacturer, COUNT(*) AS ModelsCount
4 FROM Products
5 GROUP BY Manufacturer
6 HAVING COUNT(*) > 1
```

Below the code editor is a toolbar with icons for 'Result Grid', 'Filter Rows', 'Export', and 'Wrap Cell'. The 'Result Grid' icon is active. Below the toolbar is a table with the following data:

	Manufacturer	ModelsCount
	Apple	3
	Samsung	2



Использования HAVING

```
1 SELECT Manufacturer, COUNT(*) AS ModelsCount
2 FROM Products
3 WHERE Price * ProductCount > 80000
4 GROUP BY Manufacturer
5 HAVING COUNT(*) > 1;
```

В данном случае сначала фильтруются строки: выбираются те товары, общая стоимость которых больше 80000. Затем выбранные товары группируются по производителям. И далее фильтруются сами группы - выбираются те группы, которые содержат больше 1 модели.



Использования HAVING: таблица для примера

```
1 CREATE TABLE Products
2 (
3     Id INT AUTO_INCREMENT PRIMARY KEY,
4     ProductName VARCHAR(30) NOT NULL,
5     Manufacturer VARCHAR(20) NOT NULL,
6     ProductCount INT DEFAULT 0,
7     Price DECIMAL NOT NULL
8 );
9 INSERT INTO Products(ProductName, Manufacturer, ProductCount, Price)
10 VALUES
11 ('iPhone X', 'Apple', 3, 76000),
12 ('iPhone 8', 'Apple', 2, 51000),
13 ('iPhone 7', 'Apple', 5, 32000),
14 ('Galaxy S9', 'Samsung', 2, 56000),
15 ('Galaxy S8', 'Samsung', 1, 46000),
16 ('Honor 10', 'Huawei', 5, 28000),
17 ('Nokia 8', 'HMD Global', 6, 38000);
```



Использования HAVING:

```
1 • USE productsdb;
2
3 • SELECT Manufacturer, COUNT(*) AS Models, SUM(ProductCount) AS Units
4 FROM Products
5 WHERE Price * ProductCount > 80000
6 GROUP BY Manufacturer
7 HAVING SUM(ProductCount) > 2
8 ORDER BY Units DESC;
```

<

Result Grid | | Filter Rows: | Export: | Wrap Cell Content:

	Manufacturer	Models	Units
	Apple	3	10
	HMD Global	1	6
	Huawei	1	5



Задание для самопроверки



Исходная таблица:

```
1 CREATE TABLE Products
2 (
3     Id INT AUTO_INCREMENT PRIMARY KEY,
4     ProductName VARCHAR(30) NOT NULL,
5     Manufacturer VARCHAR(20) NOT NULL,
6     ProductCount INT DEFAULT 0,
7     Price DECIMAL NOT NULL
8 );
9 INSERT INTO Products(ProductName, Manufacturer, ProductCount, Price)
10 VALUES
11 ('iPhone X', 'Apple', 3, 76000),
12 ('iPhone 8', 'Apple', 2, 51000),
13 ('iPhone 7', 'Apple', 5, 32000),
14 ('Galaxy S9', 'Samsung', 2, 56000),
15 ('Galaxy S8', 'Samsung', 1, 46000),
16 ('Honor 10', 'Huawei', 5, 28000),
17 ('Nokia 8', 'HMD Global', 6, 38000);
```



Что будет результатом запроса?

```
1 SELECT Manufacturer, COUNT(*) AS Models, ProductCount
2 FROM main
3 WHERE Price > 40000
4 GROUP BY Manufacturer;
```



Результат:

```
21 • SELECT Manufacturer, COUNT(*) AS Models, ProductCount
22 FROM main
23 WHERE Price > 40000
24 GROUP BY Manufacturer;
```

Result Grid



Filter Rows:

Export:



Wrap Cell Content:



	Manufacturer	Models	ProductCount
▶	Apple	2	3
	Samsung	2	2











Приоритет операций

1. FROM, включая JOINS
2. WHERE
3. GROUP BY
4. HAVING
5. Функции WINDOW
6. SELECT
7. DISTINCT
8. UNION
9. ORDER BY
10. LIMIT и OFFSET



Итоги занятия:

-  Изучили способ сортировка результатов запроса по конкретному полю
-  Узнали про ограничения выборки (top, limit, fetch)
-  Научились выбирать уникальные значения
-  Группировка с помощью GROUP BY
-  Научились применять агрегатные функции
-  Использование HAVING
-  Приоритет операций
-  Подготовились к дальнейшей работе по курсу.



Спасибо за внимание!