



SQL

SELECT – устанавливается, какие столбцы должны присутствовать в выходных данных;

DISTINCT – отбрасываются дублирующие записи и выполняется сортировка;

FROM – определяются имена используемых таблиц;

WHERE – выполняется фильтрация строк объекта в соответствии с заданными условиями;

GROUP BY – образуются группы строк, имеющие одно и то же значение в указанном столбце;

HAVING – фильтруются группы строк объекта в соответствии с указанным условием;

ORDER BY – определяется упорядоченность результатов выполнения операторов.

DISTINCT

```
SELECT DISTINCT Фамилия  
FROM Владельцы;
```

```
SELECT DISTINCT Фамилия, Код_улицы  
FROM Владельцы;
```

WHERE

Существует пять основных типов условий поиска (или предикатов):

- 1) сравнение,
- 2) диапазон,
- 3) принадлежность множеству,
- 4) соответствие шаблону,
- 5) значение NULL.

WHERE

- 1) сравнение - сравниваются результаты вычисления одного выражения с результатами вычисления другого

Операторы сравнения:

= равенство;

< меньше;

> больше;

<= меньше или равно;

>= больше или равно;

<> не равно.

```
SELECT *  
FROM Владельцы  
WHERE  
Номер_дома>100;
```

WHERE

Более сложные запросы могут быть построены с помощью логических операторов AND, OR или NOT, а также скобок, используемых для определения порядка вычисления выражения.

```
SELECT *  
FROM Владельцы  
WHERE  
Номер_дома>100  
AND  
Номер_дома<110;
```

WHERE

- 2) диапазон - проверяется, попадает ли результат вычисления выражения в заданный диапазон значений

Оператор
BETWEEN
используется для
поиска значения
внутри некоторого
интервала

```
SELECT *  
FROM Владельцы  
WHERE  
Номер_дома  
NOT BETWEEN  
100 AND 110;
```

WHERE

- 3) принадлежность множеству - проверяется, принадлежит ли результат вычислений выражения заданному множеству значений.

Оператор IN используется для сравнения некоторого значения со списком заданных значений

```
SELECT *  
FROM Владельцы  
WHERE Фамилия IN  
("Чернышов",  
"Медведев",  
"Гаспарян",  
"Слободской");
```


WHERE

- 4) соответствие шаблону - проверяется, отвечает ли некоторое строковое значение заданному шаблону.

С помощью оператора LIKE можно выполнять сравнение выражения с заданным шаблоном, в котором допускается использование символов-заменителей:

% (*) любое количество символов.

_ (?) один символ строки.

[] один из возможных символов, указанный в этих ограничителях.

[^] все символы, кроме указанных в ограничителях.

WHERE

```
SELECT *  
FROM Владельцы  
WHERE Телефон Like "6*";
```

```
SELECT *  
FROM Владельцы  
WHERE Телефон Like "6?????0";
```

```
SELECT *  
FROM Владельцы  
WHERE Телефон Like "[6,4]?????0";
```

WHERE

- 5) Значение NULL: проверяется, содержит ли данный столбец определитель NULL (неизвестное значение).

Оператор IS NULL используется для сравнения текущего значения со значением NULL:

```
SELECT *  
FROM Владельцы  
WHERE Телефон IS  
NULL;
```

ORDER BY

ORDER BY сортирует данные выходного набора в заданной последовательности. Сортировка по возрастанию задается ключевым словом **ASC**. Сортировка в обратной последовательности задается ключевым словом **DESC**.

```
SELECT *  
FROM Владельцы  
ORDER BY Фамилия, Имя DESC;
```


Агрегирующие функции

Count (Выражение) - определяет количество записей в выходном наборе SQL-запроса;

Min/Max (Выражение) - определяют наименьшее и наибольшее из множества значений в некотором поле запроса;

Avg (Выражение) - эта функция позволяет рассчитать среднее значение множества значений, хранящихся в определенном поле отобранных запросом записей. Оно является арифметическим средним значением, т.е. суммой значений, деленной на их количество.

Sum (Выражение) - вычисляет сумму множества значений, содержащихся в определенном поле отобранных запросом записей.



SELECT COUNT(*) AS COUNT
FROM Владельцы;

SELECT MAX(Номер_дома) AS
Максимальный_номер_дома
FROM Владельцы;

SELECT AVG(Номер_дома) AS
Среднее_значение_номера_дома
FROM Владельцы;

GROUP BY

GROUP BY без агрегирующих функций:

Аналогичен предложению DISTINCT

```
SELECT Код_улицы  
FROM Владельцы  
GROUP BY Код_улицы;
```

Аналогичен

```
SELECT DISTINCT Код_улицы  
FROM Владельцы;
```

GROUP BY

GROUP BY с агрегирующими функциями:

```
SELECT Код_улицы, AVG(Цена)  
FROM Владельцы  
GROUP BY Код_улицы;
```

```
SELECT Фамилия, SUM(Цена)  
FROM Владельцы  
GROUP BY Фамилия;
```


GROUP BY

GROUP BY с агрегирующими функциями:

При использовании GROUP BY
Часто возникают ошибки,
например нельзя писать так:

```
SELECT Фамилия, Цена  
FROM Владельцы  
GROUP BY Фамилия;
```

GROUP BY

GROUP BY с предложением WHERE:

```
SELECT Код_улицы, AVG(Цена)
FROM Владельцы
WHERE Цена>1500000
GROUP BY Код_улицы;
```

```
SELECT Код, Фамилия, SUM(Цена)
FROM Владельцы
WHERE Дробная_часть_номера IS NOT NULL
GROUP BY Фамилия;
```

HAVING

HAVING аналогичен WHERE, но:

- 1) HAVING используется только при группировке (использовании GROUP BY);
- 2) WHERE выполняется до группировки, HAVING – после;
- 3) в HAVING можно использовать агрегирующие функции, в WHERE – нельзя;
- 4) элементы предложения HAVING должны включаться в список выбора, на WHERE это правило не распространяется.

HAVING

```
SELECT Фамилия, SUM(Цена)  
FROM Владельцы  
GROUP BY Фамилия  
HAVING SUM(Цена)>1000000;
```

```
SELECT Фамилия, SUM(Цена)  
FROM Владельцы  
WHERE Цена>1500000  
GROUP BY Фамилия  
HAVING SUM(Цена)>1000000;
```

Создание БД

```
CREATE DATABASE имя_БД;
```

Выбор БД

```
USE имя_БД;
```

или

```
DATABASE имя_БД;
```

или

```
CONNECT имя_БД;
```

Удаление БД

```
DROP DATABASE имя_БД;
```

Удаление таблицы

`DROP TABLE имя_таблицы;`

Обработка данных в таблице

`INSERT` – добавление новых строк,
`UPDATE` – изменение строк,
`DELETE` – удаление строк.

INSERT

```
INSERT INTO имя_таблицы  
[(столбец1 [, столбец2][1,...,n])]  
VALUES  
(константа1 [, константа2][1,...,n]);
```

Указание столбцов необходимо для:

- 1) добавления данных в той последовательности, в какой перечислены столбцы;
- 2) добавления строк с пустыми полями.

INSERT

```
INSERT INTO Справочник_улиц  
VALUES (13, 'Вновьдобавленная улица');
```

```
INSERT INTO Справочник_улиц  
( Наименование )  
VALUES ('Университетская');
```


INSERT

Использование оператора SELECT
в команде INSERT

```
INSERT INTO имя_таблицы  
[(столбец1 [, столбец2][1,...,n])]  
SELECT список_столбцов  
FROM список_таблиц  
WHERE условия
```

INSERT

Использование оператора SELECT
в команде INSERT

```
INSERT INTO Справочник_улиц  
( Наименование )  
SELECT Фамилия FROM  
Владельцы  
WHERE Код_улицы = 11;
```

UPDATE

```
UPDATE имя_таблицы  
SET имя_столбца = выражение  
[WHERE условие];
```

```
UPDATE Владельцы  
SET Город = 'Томск';
```

```
UPDATE Владельцы  
SET Город = 'Новосибирск'  
WHERE Фамилия = "Гаспарян";
```

DELETE

```
DELETE FROM имя_таблицы  
WHERE условие;
```

```
DELETE FROM Справочник_улиц  
WHERE Код = 17;
```