**HAVING** применяется для фильтрации строк после группировки, фильтрует группы по значениям результата агрегатных функций.

**Агрегатной функцией** в языке SQL называется функция, возвращающая какую-либо одну строку по набору из входных строк.

**Конкатенация строк** в SQL выполняется с помощью функции CONCAT(), принимает от 2 до 254 строковых параметров. Неявно преобразует все аргументы к строковому типу. Если передать все значения типа NULL, то возвращается пустая строка.

Конкатенация строк может выполняться и с помощью оператора +, но в таком случае необходимы явные приведения типов.

Функция CAST преобразует выражение из одного типа данных в другой.

CAST ( expression AS data\_type [ ( length ) ] )

*length*  
Указываемое дополнительно целое число, обозначающее длину целевого типа данных, который допускает указание длины пользователем. Значение по умолчанию — 30.

Значение NULL хранится в БД следующим образом: для столбцов фиксированной длины, значение NULL занимает длину столбца плюс 1 байт, чтобы отметить является ли оно NULL или нет. Если столбец имеет переменную длину, то NULL занимает 0 байт плюс 1 байт чтобы отметить является ли оно NULL или нет.

**Вычисляемые поля –** виртуальный столбец, физически не хранящийся в таблице (если не установлен признак PERSISTED). Они автоматически рассчитываются при выполнении запроса.

**Права доступа**

"GRANT SELECT, INSERT ON таблица TO пользователь1, пользователь2;

дать права на SELECT и INSERT в таблице пользователю1 и пользователю2

GRANT ALL ON таблица TO пользователь1;

дать все права в таблице пользователю1

GRANT UPDATE (столбец1, столбец2) ON таблица TO пользователь1;

дать права на UPDATE столбец1 и столбец2 в таблице пользователю1

GRANT SELECT ON таблица TO PUBLIC;

дать права на SELECT в таблице всем

GRANT INSERT ON таблица TO пользователь1 WITH GRANT OPTION;

дать права с возможностью передачи прав

REVOKE INSERT ON таблица FROM пользователь1;

забрать права на INSERT в таблице у пользователя1

Запрет действий DENY SELECT ON таблица TO пользователь1;

**Добавление в таблицу значений**

INSERT INTO имя таблицы (столбец 1, столбец 2) VALUES (“значение 1’, “значение 2”);

NOT NULL столбец не будет иметь NULL значение

DEFAULT задает значение по умолчанию

CREATE TABLE user (name VARCHAR (10) NOT NULL DEFAULT serg, group INTEGER (10));

Создали таблицу с двумя столбцами первый из которых не может иметь значение NULL и по умолчанию если не задать значение, то будет serg

**Выборка данных**

SELECT \* FROM user; – просмотреть все данные из таблицы user

SELECT выбрать столбцы FROM из таблицы WHERE по критерию столбец=значение

SELECT \* FROM user WHERE name =’ivan’;

В таблице user будет искать по столбцу name значение ivan и выводим все строки, в которых есть ivan

SELECT name, group FROM user WHERE name =’ivan’;

По столбцам name, group в таблице user ищем по столбцу name значение ivan и выводим все строки из выбранных столбцов в которых есть ivan

SELECT name FROM user;

Вывести из таблицы user столбец name

**Ключевое слово AND – и**

SELECT name, group FROM user WHERE name =’ivan’ AND group = 1;

Выведет всех Иванов из 1 группы

**OR (или) объединение двух условий**

SELECT name, group FROM user WHERE name =’ivan’ OR name =’serg’;

Если истинными должны быть все условия, используйте AND. Если истинным должно быть хотя бы одно из условий, используйте 0R.

**Для выборки и поиска значение NULL используется IS NULL**

SELECT name FROM user WHERE name IS NULL;

**ключевое слово LIKE** (иногда называют wildcards), которое в сочетании со специальными символами ищет часть текстовой строки и возвращает совпадения.

SELECT \* FROM name WHERE user LIKE '%ivan’;

**%** — обозначает любое количество произвольных символов

**— знак подчеркивания** (\_) — представляет ровно один произвольный символ.

**ключевое слово BETWEEN** для проверки вхождения значений в диапазон (включительно с границами) можно воспользоваться

SELECT group FROM user WHERE group BETWEEN 1 AND 5;

Выведет все строки с 1-5 столбца group

**ключевого слова IN**

Вместо того чтобы строить длинные цепочки ОR, мы можем упростить запрос при помощи ключевого слова IN. После IN следует набор значений в круглых скобках. Если значение столбца совпадает с одним из значений набора, то запись или заданное подмножество столбцов включаются в результат запроса

SELECT group FROM user WHERE user IN (‘ivan ‘, serg ');

Данная запись выведет нам всех ivan, serg со всех group

**С конструкцией NOT IN** в выборку включаются записи, у которых значение столбца не входит в заданный набор.

SELECT group FROM user WHERE user NOT IN (‘ivan ‘, serg ');

Данная запись выведет всех кроме inav, serg

При использовании с AND и OR ключевое слово NOT ставится после них:

SELECT \* FROM user WHERE NOT name = 'ivan' AND NOT name = ‘serg’;

**Удаление и изменение данных в таблицы**

**Команда DELETE**

DELETE FROM user WHERE name = ‘serg’;

Удалит из таблицы user столбца name значение serg

* Команда delete не позволяет удалить значение одного столбца или группы столбцов.
* Команда delete удаляет из таблицы одну или несколько записей
* Также возможно удаление сразу нескольких записей. Для этого критерий выбора удаляемых записей определяется при помощи условия WHERE.
* следующая команда удаляет из таблицы все записи: DELETE FROM user

**Команда UPDATE**

UPDATE user SET name =’ivan’ WHERE name = ‘serg’;

Из таблицы user, столбца name значение serg измениться на ivan

UPDATE user SET name =’ivan’ WHERE group = 1;

Из таблицы user, столбца group со значением 1 все значение name поменяются на ivan

UPDATE user SET group = group + 1 WHERE name = ‘ivan’;

Из таблицы user столбца name со значением ivan измениться значение столбца group на +1, т.е если ivan был в 1 группе, то теперь он во 2

* Команда UPDATE может использоваться для изменения значения одного столбца или группы столбцов.

Включите дополнительные пары столбец = значение в условие SET и поставьте запятую после каждой пары: UPDATE your\_table SET первый\_столбец= 'новое\_значение', второй\_столбец = 'старое\_значение';

* Команда UPDATE может изменять одну запись или несколько записей в зависимости от условия where.

**Изменение таблицы**

**команда ALTER**

ALTER TABLE user

ADD COLUMN id INT NOT NULL AUTO\_INCREMENT FIRST

ADD PRIMARY KEY(id);

Добавляем в таблицу user новый столбец с помочью ADD COLUMN и присваиваем ему первое место в таблице с помощью FIRST

**Ключевое слова AFTER.**

задать местонахождение нового столбца в таблице после указанного столбца

**Ключевое слово BEFORE**

задать местонахождение нового столбца в таблице перед указанным столбцом

**Ключевое слово LAST**

задать местонахождение нового столбца в таблице в конце

**Ключевое слово FIRST**

задать местонахождение нового столбца в таблице в начале

**Ключевое слово SECOND**

задать местонахождение нового столбца в таблице после первого столбца

ALTER TABLE user ADD COLUMN family VARCHAR (20) AFTER name;

В таблице user создать новый столбец family после столбца name

CHANGE - изменение имени и типа данных столбцов

MODIDY- изменение типа данных или позиции столбцов

ADD - добавление столбцов в таблицу

DROP- удаление столбцов из таблицы

**Команда RENAME**

Позволяет переименовать таблицу

ALTER TABLE user RENAME TO children;

Переименовали таблицу user в children

**Конструкция CHANGE COLUMN**

Позволять менять имя и тип данных

ALTER TABLE user

CHANGE COLUMN group people VARCHAR (20);

Изменит столбец user на people и поменяет тип данных на символьный

**Команда MODIFY**

Изменяет тип данных

ALTER TABLE user

MODIFY COLUMN group VARCHAR (20);

В столбце user поменяет тип данных на символьный

**Команда DROP**

Используется для удаления архитектуры

ALTER TABLE user DROP COLUMN name;

Удаляет из таблицы user столбец name

**Функции RIGHT () и LEFT ()**

выделения заданного количества символов в столбце

SELECT RIGHT(name,4) FROM user;

Выделит с правого бока 4 символа в столбе name

**Функция SUBSTRING\_INDEX ()**

находит все символы текстового значения, предшествующие заданному символу или подстроке.

SELECT SUBSTRING\_INDEX (name, ‘,’ ,1) FROM user;

Возвращает все символы до 1 запятой

**Функция SUBSTRING (текст, начало, длина)**

возвращает часть строкового значения текст, начиная с буквы в позиции начало. Параметр длина определяет длину возвращаемой строки.

SELECT SUBSTRING {'Сан-Антонио, ТХ', 5, 3);

**Функции UPPER (текст) и LOWER (текст)** преобразуют все символы строки к верхнему или нижнему регистру

SELECT UPPER('сШа'); SELECT LOWER('спаГЕтти');

**Функция REVERSE (текст)**

переставляет символы строки в обратном порядке.

SELECT REVERSE('спаГЕтти');

**функции LTRIM(TeKCT) и RTRlM(TeKCT)**

возвращают строку, полученную удалением лишних пробелов в начале (у левого края) или в конце (у правого края) строки.

SELECT LTRIMC (‘собака ');

SELECT RTRIMC (‘кошка ');

**функция LENGTH (текст)**

возвращает количество символов в строке.

SELECT LENGTH ('Сан-Антонио, ТХ ');

**Объедение функций и команд**

UPDATE user SET family = RIGHT (name, 2);

В таблице user в столбец family и заполнит его выделенными значениями из столбца name

**Выражение CASE**

объединяет множество команд UPDATE, проверяя значение существующего столбца по условию. Если условие выполняется, то новый столбец заполняется заданным значением

UPDATE user

SET новый столбец =

CASE

WHEN столбец 1 = значение 1 //если выполнится это условие

THEN новое значение 1 // то новому столбцу присвоиться новое знач. 1

WHEN столбец 2 = значение 2 // если выполняется другое условие

THEN новое значение 2 // то новому столбцу присвоиться новое знач. 2

ELSE новое значение 3 // если не одно условие не выполниться то новому столбцу присвоится новое значение 3

END; //завершает выполнение CASE

### Основные команды

**DESCRIBE** позволяет посмотреть какая архитектура у таблицы

**SHOW CREATE\_TABLE** возвращает команду CREATE TABLE, которая была использована для создания таблицы

### Функции

**Ключевые слова ORDER BY**

Упорядочивает записи

SELECT name FROM user WHERE name ORDER BY name;

Упорядочивает записи в столбце name по алфавиту

**Ключевое слово DESC**

Указывается после имени столбца изменяя порядок по убыванию

SELECT name FROM user WHERE name ORDER BY name DESC;

**Ключевое слово ASC**

Указывается после имени столбца изменяя порядок по возрастанию

SELECT name FROM user WHERE name ORDER BY name ASC ;

**Функция SUM**

суммирует значения столбца

SELECT SUM (group) FROM user;

Суммирует значение таблицы group

**функцию AVG**.

суммирует значения столбца и находит среднее значение

**функция MIN**

выводит минимальное значение

**функция MAX**

выводит максимальное значение

**условие GROUP BY**

группирует записи по какому либо полю, для последующего применения агрегатной функции к каждой группе.

**функция COUNT**

возвращает количество записей в столбце

**Ключевое слово DISTINT**

возвращает записи в столбце без дубликатов

SELECT DISTINT name FROM user;

Выведет все значение из столбца name без дубликатов

**Ключевое слово LIMIT**

позволяет указать количество записей, возвращаемых запросом из итогового набора

так же есть LIMIT 0,2 позволяет вернуть записи от 1 до 2

### Многотабличные базы данных

### Перекрестное соединение

**Алиасы**

Временные имена для таблиц или столбцов. Используются чтобы сделать имена более удобочитабельными, либо чтобы дважды сделать выборку из одной таблицы в одном запросе.

**Псевдонимы столбцов**

указывается после первого использования имени столбца в запросе с другим ключевым словом AS

**Ключевое слово AS**

заполняет новую таблицу результатами SELECT

CREATE TABLE user(

id INT(10) NOT NULL AUTO\_INCREMENT PRIMARY KEY,

user VARCHAR(20) )

AS

SELECT user AS usr FROM baz\_dan

GROUP BY usr

ORDER BY usr;

В таблица user создали стобец id и столбец user пометивего псевдонимом usr с помощью команды AS

**Перекрестное соединение (CROSS JOIN)** возвращает комбинации каждой записи первой таблицы с каждой записью второй таблицы.

SELECT учителя.имя, ученики.номер\_учителя FROM учителя CROSS JOIN ученики;

**Внутренним соединением (INNER JOIN)** называется перекрестное соединение, из результатов которого часть записей исключается по условию запроса.

**Эквивалентное соединение** **=** внутреннее соединение с проверкой равенства.

**Неэквивалентное соединение <>** возвращает записи, у которых заданные значения столбцов не равны

**Естественные соединения** **(NATURAL JOIN)** возможны только в том случае, если столбец, по которому выполняется соединение, имеет одинаковые имена в обеих таблицах

**LEFT OUTER JOIN**

В левом внешнем соединении для каждой записи левой таблицы ищется соответствие среди записей правой таблицы.

внешние соединения в большей степени зависят от отношений между двумя таблицами, чем все рассмотренные ранее типы соединений.

Левое внешнее соединение **(LEFT OUTER JOIN**) перебирает все записи левой таблицы и ищет для каждой соответствие среди записей правой таблицы.

В левом внешнем соединении таблица, следующая

после FROM, но ДО JOIN, считается ≪левой≫, а таблица, следующая ПОСЛЕ JOIN, считается ≪правой≫.

внешнее соединение возвращает запись независимо от того, есть у нее совпадение в другой таблице или нет.

Значение NULL в результатах левого

внешнего соединения означает, что правая таблица не содержит значений, соответствующих левой таблице.

**RIGHT OUTER JOIN**

Правое внешнее соединение ищет в левой таблице соответствия для правой таблицы

Одна и та же таблица может использоваться в качестве левой и правой таблицы соединения.

### **Подзапросы**

Вложенный запрос – запрос внутри другого запроса, который используется внутри инструкции SELECT, INSERT, UPDATE или DELETE или внутри другого вложенного запроса. Количество вложенных запросов не ограничено.

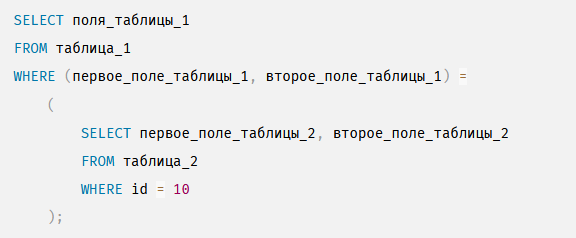
Подзапрос может содержать все стандартные инструкции, разрешённые для использования в обычном SQL-запросе: DISTINCT, GROUP BY, LIMIT, ORDER BY, объединения таблиц, запросов и т.д.

Подзапрос может возвращать скаляр (одно значение), одну строку, один столбец или таблицу (одну или несколько строк из одного или нескольких столбцов). Они называются скалярными, столбцовыми, строковыми и табличными подзапросами.

ANY — ключевое слово, которое должно следовать за операцией сравнения (>, <, <>, = и т.д.), возвращающее TRUE, если хотя бы одно из значений столбца подзапроса удовлетворяет обозначенному условию.

ALL – все значения столбца подзапроса удовлетворяют условию.

Строковый подзапрос — это подзапрос, возвращающий единственную строку с более чем одной колонкой.



**Связанным подзапросом** является подзапрос, который содержит ссылку на таблицу, которая была объявлена во внешнем запросе.

**Производная таблица** — выражение, которое генерирует временную таблицу в предложении FROM, которая работает так же, как и обычные таблицы, которые вы указываете через запятую.



**РЕФЛЕКСИВНЫЙ внешний ключ**

Первичный ключ таблицы, используемый в той же таблицедля других целей.

При само соединении запрос к одной таблице строится так, как если бы она была двумя таблицами, содержащими одинаковую информацию

**ключевое слово UNION**.

Союз объединяет в одну таблицу результаты двух и более запросов на основании того, что указано в запросе SELECT. Союзы можно трактовать как ≪пересекающиеся≫ значения всех запросов.

При использовании UNION допускается только одно условие ORDER BY в концекоманды *.* Это объясняется тем, что союз объединяет и группирует результаты нескольких команд SELECT

Количество столбцов в командах SELECT должно быть одинаковым.Нельзя выбрать два столбца одной командой и еще один столбец другой. Команды SELECT должны содержать одинаковые выражения и агрегатные функции. Команды SELECT могут следовать в любом порядке; на результаты это не влияет.

По умолчанию SQL исключает дубликаты из результатов союзов Типы данных в столбцах должны либо совпадать, либо быть совместимыми. Если вы по какой-то причине хотите получить список со всеми дубликатами**,** используйте **оператор UNION ALL .**Он возвращает все совпадения, не только уникальные.

UNION ALL работает точно так же, как UNION, если не считать того, что он возвращает все значения из столбцов — вместо одного экземпляра из каждой группы дубликатов.

**Конструкции INTERSECT и EXCEPT**, в отличие от UNION, используются для поиска перекрывающихся результатов запросов. INTERSECT возвращает данные из результатов первого запроса, которые также присутствуют во втором запросе.

EXCEPT возвращает только те значения, которые присутствуют в результатах первого запроса, но не встречаются в результатах второго.