**Порядок выполнения  SQL запроса на выборку на СЕРВЕРЕ**:

1. FROM
2. WHERE
3. GROUP BY
4. HAVING
5. SELECT
6. ORDER BY

Сначала определяется таблица, из которой выбираются данные (FROM), затем из этой таблицы отбираются записи в соответствии с условием  WHERE, выбранные данные агрегируются (GROUP BY),  из агрегированных записей выбираются те, которые удовлетворяют условию после HAVING. Потом формируются данные результирующей выборки, как это указано после SELECT ( вычисляются выражения, присваиваются имена и пр. ). Результирующая выборка сортируется, как указано после ORDER BY.

**Таблица команд по SQL**

|  |  |  |
| --- | --- | --- |
| Описание команды | Команда | Пример |
| **Тип данных** | INT - Целое число  DECIMAL - Вещественное число  DATE - Дата в формате ГГГГ-ММ-ДД  VARCHAR - Строка не более 255 символов | -587  12.56  2022-10-14  Пример |
| **Создание и корректировка таблицы** |  |  |
| Создание таблицы | CREATE TABLE имя таблицы(  название поля и его описание,  название поля и его описание,  …  ); | CREATE TABLE genre(  genre\_id INT PRIMARY KEY AUTO\_INCREMENT,  name\_genre VARCHAR(30),  price DECIMAL(8,2),  amount INT  ); |
| Создание таблицы с внешними ключами | FOREIGN KEY (связанное\_поле\_зависимой\_таблицы)  REFERENCES главная\_таблица (связанное\_поле\_главной\_таблицы) | CREATE TABLE book (  book\_id INT PRIMARY KEY AUTO\_INCREMENT,  title VARCHAR(50),  author\_id INT NOT NULL,  price DECIMAL(8,2),  amount INT,  **FOREIGN KEY (author\_id) REFERENCES author (author\_id)**  ); |
| Действия при удалении записи главной таблицы | С помощью выраженияON DELETE можно установить действия, которые выполняются для записей подчиненной таблицы при удалении связанной строки из главной таблицы. При удалении можно установить следующие опции:   * CASCADE: автоматически удаляет строки из зависимой таблицы при удалении  связанных строк в главной таблице. * SET NULL: при удалении  связанной строки из главной таблицы устанавливает для столбца внешнего ключа значение **NULL**. (В этом случае столбец внешнего ключа должен поддерживать установку **NULL**). * SET DEFAULT похоже на SET NULL за тем исключением, что значение  внешнего ключа устанавливается не в NULL, а в значение по умолчанию для данного столбца. * RESTRICT: отклоняет удаление строк в главной таблице при наличии связанных строк в зависимой таблице.   **Важно!** Если для столбца установлена опция SET NULL, то при его описании нельзя задать ограничение на пустое значение. | CREATE TABLE book (  book\_id INT PRIMARY KEY AUTO\_INCREMENT,  title VARCHAR(50),  author\_id INT NOT NULL,  price DECIMAL(8,2),  amount INT,  FOREIGN KEY (author\_id) REFERENCES author (author\_id) **ON DELETE CASCADE**  ); |
| Занесение новой записи в таблицу | INSERT INTO таблица(поле1, поле2)  VALUES (значение1, значение2); | INSERT INTO book (title, author, price, amount)  VALUES  ('Мастер и Маргарита', 'Булгаков', 670.99, NULL),  ('Идиот', 'Достоевский', 460.00, 10),  ('Братья Карамазовы', 'Достоевский', 799.01, 2); |
| Добавление записей из другой таблицы | С помощью запроса на добавление можно добавить записи из другой таблицы, отобранные с помощью запроса на выборку.  В этом случае вместо раздела VALUES записывается запрос на выборку, начинающийся с SELECT.  В нем можно использовать WHERE, GROUP BY, ORDER BY. | INSERT INTO book (title, author, price, amount)  SELECT title, author, price, amount  FROM supply  WHERE author NOT IN( 'Булгаков М.А.' , 'Достоевский Ф.М.'); |
| Добавление записей, вложенные запросы | В запросах на добавление можно использовать вложенные запросы. | INSERT INTO book (title, author, price, amount)  SELECT title, author, price, amount  FROM supply  WHERE title NOT IN (  SELECT title  FROM book  );  SELECT \* FROM book; |
| Запросы на обновление | UPDATE таблица SET поле = выражение  С помощью запросов на обновление можно изменять не все записи в таблице, а только часть из них. Для этого в запрос включается ключевое слово WHERE, после которого указывается условие отбора строк для изменения. | UPDATE book  SET price = 0.7 \* price  WHERE amount < 5; |
| Запросы на обновление нескольких столбцов | UPDATE таблица SET поле1 = выражение1, поле2 = выражение2 | UPDATE book  SET buy = IF(buy >= amount, amount, buy),  price = IF(buy = 0, 0.9 \* price, price); |
| Запросы на обновление нескольких таблиц | В запросах на обновление нескольких таблиц должны соблюдаться следующие требования:   * для столбцов, имеющих одинаковые имена, необходимо указывать имя таблицы, к которой они относятся, например, **book.price**; * все таблицы, используемые в запросе, нужно перечислить после ключевого слова UPDATE; * в запросе обязательно условие WHERE, в котором указывается условие, при котором обновляются данные. | UPDATE book, supply  SET book.amount = book.amount + supply.amount  WHERE book.title = supply.title AND book.author = supply.author; |
| Запросы на удаление | Запросы корректировки данных позволяют удалить одну или несколько записей из таблицы.  DELETE FROM таблица;  Запрос на удаление позволяет удалить не все записи таблицы, а только те, которые удовлетворяют условию, указанному после ключевого слова WHERE:  DELETE FROM таблица  WHERE условие; | DELETE FROM supply;  DELETE FROM supply  WHERE title IN (  SELECT title  FROM book  ); |
| Запросы на создание новой таблицы на основе данных из другой таблицы. | Для этого используется запрос SELECT, результирующая таблица которого и будет новой таблицей. При этом имена столбцов запроса становятся именами столбцов новой таблицы.  CREATE TABLE имя\_таблицы AS  SELECT ... | CREATE TABLE ordering AS  SELECT author, title, 5 AS amount  FROM book  WHERE amount < 4; |
| **Запросы на выборку** |  |  |
| Выбрать все записи таблицы | SELECT \* FROM имя таблицы; | SELECT \* FROM book; |
| Выборка отдельных столбцов | SELECT имя столбца, имя столбца... FROM имя таблицы; | SELECT title, amount FROM book; |
| Выборка столбцов и присвоение им новых имен | SELECT имя столбца AS псевдоним, имя столбца...  FROM имя таблицы; | SELECT title AS Название, amount  FROM book; |
| Выборка с созданием вычисляемого столбца | SELECT выражение AS псевдоним, имя столбца...  FROM имя таблицы; | SELECT title, author, price, amount,  price \* amount AS total  FROM book; |
| **Математические функции** |  |  |
| округляет до целого числа в большую сторону от х | CEILING(x) | CEILING(4.2)=5 CEILING(-5.8)=-5 |
| округляет значение **x** до **k** знаков после запятой, если **k** не указано – **x** округляется до целого | ROUND(x, k) | ROUND(4.361)=4 ROUND(5.86592,1)=5.9 |
| округляет до  целого числа в меньшую сторону от х | FLOOR(x) | FLOOR(4.2)=4 FLOOR(-5.8)=-6 |
| возведение **x** в степень **y** | POWER(x, y) | POWER(3,4)=81.0 |
| квадратный корень из **x** | SQRT(x) | SQRT(4)=2.0 SQRT(2)=1.41... |
| модуль числа **x** | ABS(x) | ABS(-1) = 1 ABS(1) = 1 |
| Логические функции | IF(логическое\_выражение, выражение\_1 (истина), выражение\_2 (ложь)) | SELECT title, amount, price,  ROUND(IF(amount<4, price\*0.5, price\*0.7),2) AS sale  FROM book; |
| Выборка данных по условию | SELECT имя столбца, имя столбца... FROM имя таблицы  WHERE логическое выражение (**=**, **<>**, **>**, **<**, **>=**, **<=**); | SELECT title, price  FROM book  WHERE price < 600; |
| Логические операции | И «AND», ИЛИ «OR», НЕ «NOT»  Приоритеты операций:   1. круглые скобки 2. умножение  (\*),  деление (/) 3. сложение  (+), вычитание (-) 4. =, >, <, >=, <=, <> 5. NOT 6. AND 7. OR | SELECT title, author, price  FROM book  WHERE price > 600 AND author = 'Булгаков М.А.'; |
| Выборка данных из интервала, включая его границы | BETWEEN левая граница AND правая граница | SELECT title, amount  FROM book  WHERE amount BETWEEN 5 AND 14; |
| Выборка данных из значений из списка | IN (значение1, значение2… ) | SELECT title, price  FROM book  WHERE author IN ('Булгаков', 'Достоевский'); |
| Сортировка  ASC (по возрастанию), DESC (по убыванию) | ORDER BY   * название столбца; * номер столбца; * псевдоним столбца. | SELECT title, author, amount AS Количество  FROM book  ORDER BY title; /\*ИЛИ \*/  ORDER BY 1, 3, Количество DESC; |
| Сравнение с шаблоном | LIKE  % - любая строка, содержащая ноль или более символов,  \_ - любой одиночный символ | SELECT title  FROM book  WHERE title LIKE 'Б%'; /\*ИЛИ 'б%' \*/  SELECT title FROM book  WHERE title LIKE "\_% и \_%" /\*отбирает слово И внутри названия \*/  OR title LIKE "и \_%" /\*отбирает слово И в начале названия \*/  OR title LIKE "\_% и" /\*отбирает слово И в конце названия \*/  OR title LIKE "и" /\* отбирает название, состоящее из одного слова И \*/ |
| Выбор уникальных элементов столбца | DISTINCT имя столбца  или  GROUP BY имя столбца | SELECT DISTINCT author  FROM book;  /\* ИЛИ \*/  SELECT author  FROM book  GROUP BY author; |
| Групповые функции | SUM() – подсчет суммы значений столбца  COUNT() – подсчет кол-ва записей в столбце   * COUNT(\*) —  подсчитывает  все записи в группе, в том числе NULL; * COUNT(имя\_столбца) — возвращает количество записей столбца (только NOT NULL)в группе.   MIN()-вычисляют минимальное  MAX()- максимальное  AVG()- среднее значение эл-в столбца, относящихся к группе. | SELECT author, sum(amount), count(amount)  FROM book  GROUP BY author;  SELECT author, COUNT(amount), COUNT(\*)  FROM book  GROUP BY author;  SELECT author, MIN(price) AS Минимальная\_цена,  MAX(price) AS Максимальная\_цена, AVG(price) AS Средняя\_цена  FROM book  GROUP BY author; |
| Групповые функции c вычислением |  | SELECT author, SUM(price \* amount) AS Стоимость  FROM book  GROUP BY author;  SELECT author, ROUND(AVG(price),2) AS Средняя\_цена  FROM book  GROUP BY author; |
| Вычисления по таблице целиком | Результатом таких запросов является единственная строка с вычисленными по таблице значениями. | SELECT MIN(price), MAX(price), ROUND(AVG(DISTINCT price), 2)  FROM book; |
| Групповые функции с выборкой по условию | В запросах с групповыми функциями вместо WHERE используется ключевое слово HAVING , указываемое после GROUP BY. | SELECT author,  MIN(price) AS Минимальная\_цена,  FROM book  GROUP BY author  HAVING SUM(price \* amount) > 5000; |
| WHERE и HAVING могут использоваться в одном запросе | Вывести максимальную и минимальную цену книг каждого автора, кроме Есенина, количество экземпляров книг которого больше 10. | SELECT author,  MIN(price) AS Минимальная\_цена,  MAX(price) AS Максимальная\_цена  FROM book  WHERE author <> 'Есенин С.А.'  GROUP BY author  HAVING SUM(amount) > 10; |
| **Вложенные запросы** |  |  |
| Вложенный запрос, возвращающий **одно** значение | Вложенный запрос, возвращающий одно значение, может использоваться в условии отбора записей WHERE как обычное значение совместно с операциями =, <>, >=, <=, >, <. | SELECT title, author, price, amount  FROM book  WHERE price = (  SELECT MIN(price)  FROM book  ); |
| Использование вложенного запроса **в выражении** | Вложенный запрос, возвращающий одно значение, может использоваться в выражениях как обычный операнд, например, к нему можно что-то прибавить, вычесть и пр. | SELECT title, author, amount  FROM book  WHERE ABS(amount - (SELECT AVG(amount) FROM book)) >3; |
| Вложенный запрос, возвращающий **несколько** значений | Если вложенный запрос возвращает несколько значений одного столбца, его можно использовать в разделе WHERE совместно с оператором IN.  WHERE имя\_столбца IN (вложенный запрос, возвращающий один столбец) | SELECT title, author, amount, price  FROM book  WHERE author IN (  SELECT author  FROM book  GROUP BY author  HAVING SUM(amount) >= 12  ); |
| Вложенный запрос, **операторы ANY и ALL** | Вложенный запрос, возвращающий несколько значений одного столбца, можно использовать для отбора записей с помощью операторов ANY и ALL совместно с операциями отношения (=, <>, <=, >=, <, >). | SELECT title, author, amount, price  FROM book  WHERE amount < ALL (  SELECT AVG(amount)  FROM book  GROUP BY author  ); |
| Вложенный запрос **после** SELECT | Если вложенный запрос располагается после SELECT, то результат выполнения запроса выводится в отдельном столбце результирующей таблицы. При этом результатом запроса может быть только одно значение, тогда оно будет повторяться во всех строках. Также вложенный запрос может использоваться в выражениях. | SELECT title, author, amount,  (  SELECT AVG(amount)  FROM book  ) AS Среднее\_количество  FROM book  WHERE abs(amount - (SELECT AVG(amount) FROM book)) >3; |
| **Запросы на выборку** |  |  |
| Ограничение выборки и отбрасывание записей | LIMIT 3  OFFSET 3 | SELECT author,  FROM book  OFFSET 3  LIMIT 3; |
| Вычитания двух дат | **DATEDIFF(дата\_1, дата\_2)**  результатом является количество дней между **дата\_1** и **дата\_2**.  Увеличьте разницу на 1, чтобы включить первый день | DATEDIFF('2020-04-01', '2020-03-28')=4  DATEDIFF('2020-05-09','2020-05-01')=8  DATEDIFF(date\_last, date\_first) + 1 |
| Выделить номер месяца из даты | **MONTH(дата)** | **MONTH('2020-04-12') = 4**  **MONTH(date\_first)** |
| Выделить название месяца из даты | **MONTHNAME(дата)** | **MONTHNAME('2020-04-12')='April'** |
| Выделить часть даты | День **DAY()**,  месяц **MONTH()**,  год **YEAR()** | DAY('2020-02-01') = 1  MONTH('2020-02-01') = 2  YEAR('2020-02-01') = 2020 |
| Генерации случайных чисел в интервале от 0 до 1 (не включительно) | Функция RAND()  Если эту функцию умножить на 365, то она будет генерировать вещественные числа от 0 до 365 (не включительно). Таким образом, можно получить случайное число от 0 до 365. | FLOOR(RAND() \* 365) |
| Сложение  даты с числом | DATE\_ADD(дата, INTERVAL число единица\_измерения),  Где  единица\_измерения (использовать прописные буквы) – это день (DAY), месяц(MONTH), неделя(WEEK) и пр.,  число – целое число,  дата – значение даты или даты и времени. | DATE\_ADD('2020-02-02', INTERVAL 45 DAY) возвращает 18 марта 2020 года  DATE\_ADD('2020-02-02', INTERVAL 6 MONTH) возвращает 2 августа 2020 года |
| Использование временного имени таблицы | FROM fine AS f, traffic\_violation AS tv  После присвоения таблице алиаса, он используется во всех разделах запроса, в котором алиас задан. Например:  WHERE f.violation = tv.violation | UPDATE fine AS f, traffic\_violation AS tv  SET f.sum\_fine = tv.sum\_fine  WHERE f.sum\_fine IS NULL AND  f.violation = tv.violation; |
| Группировка данных по нескольким столбцам | В разделе GROUP BY можно указывать несколько столбцов, разделяя их запятыми. Тогда к одной группе будут относиться записи, у которых значения столбцов, входящих в группу, равны.  **Важно!** В разделе **GROUP BY** нужно перечислять все НЕАГРЕГИРОВАННЫЕ столбцы (к которым не применяются групповые функции) из **SELECT**. | SELECT name, number\_plate, violation, count(\*)  FROM fine  GROUP BY name, number\_plate, violation; |
| можно использовать вложенный запрос как отдельную таблицу, записанную после ключевого слова UPDATE, при этом вложенному запросу необходимо присвоить имя, например **query\_in** | UPDATE fine,  (  SELECT ...  ) query\_in  SET ...  WHERE | UPDATE fine,  (  SELECT name, number\_plate, violation  FROM fine  GROUP BY name, number\_plate, violation  HAVING COUNT(\*) >= 2  ORDER BY name, number\_plate, violation  ) query\_in  SET sum\_fine = sum\_fine \* 2  WHERE date\_payment IS NULL AND  fine.name=query\_in.name AND  fine.number\_plate=query\_in.number\_plate  AND fine.violation=query\_in.violation; |
| **Соединение таблиц** |  |  |
| Соединение INNER JOIN | Оператор внутреннего соединения INNER JOIN соединяет две таблицы. Порядок таблиц для оператора неважен, поскольку оператор является симметричным.  SELECT  ...  FROM  таблица\_1 INNER JOIN таблица\_2  ON условие | SELECT title, name\_author  FROM  author INNER JOIN book  ON author.author\_id = book.author\_id; |
| Внешнее соединение LEFT и RIGHT OUTER JOIN | Оператор внешнего соединения LEFT OUTER JOIN (OUTER можно опустить) соединяет две таблицы. Порядок таблиц для оператора важен, поскольку оператор не является симметричным.  SELECT  ...  FROM  таблица\_1 LEFT JOIN таблица\_2  ON условие  ... | SELECT name\_author, title  FROM author LEFT JOIN book  ON author.author\_id = book.author\_id  ORDER BY name\_author; |
| Перекрестное**соединение CROSS JOIN** | Оператор перекрёстного соединения, или декартова произведения CROSS соединяет две таблицы. Порядок таблиц для оператора неважен, поскольку оператор является симметричным.  SELECT  ...  FROM  таблица\_1 CROSS JOIN таблица\_2  ...  или  SELECT  ...  FROM  таблица\_1, таблица\_2  ...  Результат запроса формируется так: каждая строка одной таблицы соединяется с каждой строкой другой таблицы, формируя в результате все возможные сочетания строк двух таблиц. | SELECT name\_author, name\_genre  FROM  author, genre; |
| Запросы на выборку из нескольких таблиц | Запрос на выборку может выбирать данные из двух и более таблиц базы данных. При этом таблицы должны быть логически связаны между собой. Для каждой пары таблиц, включаемых в запрос, необходимо указать свой оператор соединения. Наиболее распространенным является внутреннее соединение INNER JOIN  Пусть таблицы связаны между собой следующим образом:  тогда запрос на выборку для этих таблиц будет иметь вид ----->>>  Если же таблицы связаны так:  то запрос на выборку выглядит следующим образом --->>> | SELECT  ...  FROM  first  INNER JOIN second ON first.first\_id = second.first\_id  INNER JOIN third ON second.second\_id = third.second\_id  ...  SELECT  ...  FROM  first  INNER JOIN third ON first.first\_id = third.first\_id  INNER JOIN second ON third.second\_id = second.second\_id  ... |
| Операция соединение, использование USING() | При описании соединения таблиц с помощью **JOIN** в некоторых случаях вместо **ON** и следующего за ним условия можно использовать оператор **USING()**.  **USING** позволяет указать набор столбцов, которые есть в обеих объединяемых таблицах. Если база данных хорошо спроектирована, а каждый внешний ключ имеет такое же имя, как и соответствующий первичный ключ (например, **genre.genre\_id = book.genre\_id**), тогда можно использовать предложение **USING** для реализации операции **JOIN**.  При этом после **SELECT**,при использовании столбцов из **USING()**, необязательно указывать, из какой именно таблицы берется столбец. | Вариант с **ON**  SELECT title, name\_author, author.author\_id /\* явно указать таблицу - обязательно \*/  FROM  author INNER JOIN book  ON author.author\_id = book.author\_id;  Вариант с **USING**  SELECT title, name\_author, author\_id /\* имя таблицы, из которой берется author\_id, указывать не обязательно\*/  FROM  author INNER JOIN book  USING(author\_id); |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |