[**http://shpargalka.my1.ru/**](http://shpargalka.my1.ru/)

* **create database** *имя\_базы\_данных*;
* **use** *имя\_базы\_данных*;
* **create table** *имя\_таблицы (имя\_первого\_столбца тип, имя\_второго\_столбца тип, ..., имя\_последнего\_столбца тип )*;
* **show** *databases;* - показать все имеющиеся БД.
* **show** *tables;* - показать список таблиц текущей БД (предварительно ее надо выбрать с помощью оператора use).
* **describe** *имя\_таблицы;* - показать описание столбцов указанной таблицы
* **drop database** *имя\_базы данных;* - удалить БД.
* **drop** **table** *имя\_таблицы;* -удалить таблицу.
* **AUTO\_INCREMENT** - высчитывает максимальное значение этого столбца, полученное значение увеличивает на 1 и заносит его в столбец.
* **PRIMARY KEY ()**
* **FOREIGN KEY** *(имя\_столбца\_которое\_является\_внешним\_ключом)* **REFERENCES** *имя\_таблицы\_родителя (имя\_столбца\_родителя);*
* **INSERT INTO** *имя\_таблицы* **VALUES** *('значение\_первого\_столбца', 'значение\_второго\_столбца', ..., 'значение\_последнего\_столбца')*;
* **INSERT INTO** *имя\_таблицы ('имя\_столбца', 'имя\_столбца')* **VALUES** *('значение\_первого\_столбца','значение\_второго\_столбца');*
* **SELECT** *что\_выбрать* **FROM** *откуда\_выбрать;*
* **SELECT** *\** **FROM** *откуда\_выбрать;* - выбрать все столбцы таблицы
* **SELECT** *имя\_столбца* **FROM** *имя\_таблицы* **ORDER BY** *имя\_столбца\_сортировки;* - сортировка
* **SELECT** *имя\_столбца* **FROM** *имя\_таблицы* **ORDER BY** *имя\_столбца\_сортировки;* - сортировка **DESC**;
* **SELECT** *имя\_столбца* **FROM** *имя\_таблицы* **WHERE** *условие;*
* **Необычные операторы:**
* **BETWEEN***меньшее\_число* **AND**  *большее\_число* **-** отбираются значения, находящиеся между указанными.
* **IS NOT NULL (IS NULL) -** отбираются строки, (не) имеющие значения в указанном поле.
* **IN (NOT IN)** - отбираются значения, (кроме) соответствующие указанным
* **LIKE (NOT LIKE) -** отбираются значения, (не) соответствующие образцу. Самый распространенный метасимвол - %. Он означает любые символы. Например, если нам надо найти слова, начинающиеся с букв "вел", то мы напишем LIKE 'вел%', а если мы хотим найти слова, которые содержат символы "клуб", то мы напишем LIKE '%клуб%'.
* **SELECT** *имя\_столбца* **FROM** *имя\_таблицы* **WHERE** *часть условия* **IN** *(***SELECT** *имя\_столбца* **FROM** *имя\_таблицы* **WHERE** *часть условия* **IN** *(***SELECT** *имя\_столбца* **FROM** *имя\_таблицы* **WHERE** *условие) )* **; -** подзапросы
* **SELECT** *имена\_столбцов\_таблицы\_1, имена\_столбцов\_таблицы\_2* **FROM** *имя\_таблицы\_1, имя\_таблицы\_2;* - объединение
* **SELECT** *имя\_таблицы\_1.имя\_столбца1\_таблицы\_1, имя\_таблицы\_1.имя\_столбца2\_таблицы\_1, имя\_таблицы\_2.имя\_столбца1\_таблицы\_2, имя\_таблицы\_2.имя\_столбца2\_таблицы\_2* **FROM** *имя\_таблицы\_1, имя\_таблицы\_2* **WHERE** *имя\_таблицы\_1.имя\_столбца\_по\_которому\_объединяем = имя\_таблицы\_2.имя\_столбца\_по\_которому\_объединяем;*
* **SELECT** *имя\_таблицы\_1.имя\_столбца, имя\_таблицы\_2.имя\_столбца* **FROM** *имя\_таблицы\_1* **ТИП ОБЪЕДИНЕНИЯ** *имя\_таблицы\_2* **ON** *условие\_объединения;*- где **ТИП ОБЪЕДИНЕНИЯ** - либо **LEFT OUTER JOIN**, либо **RIGHT OUTER JOIN.** Чтобы взять все строки с таблицы, а не только полные.
* **COUNT() –** подсчет количества строк в таблице
* **SELECT COUNT*(****имя\_столбца****)* FROM***имя\_таблицы;*
* **SELECT** *имя\_столбца* **COUNT***(имя\_столбца)* **FROM** *имя\_таблицы* **GROUP BY** *имя\_столбца;*
* **SELECT** *имя\_столбца* **COUNT***(имя\_столбца)* **FROM** *имя\_таблицы* **GROUP BY** *имя\_столбца* **HAVING COUNT** *условие;* - **HAVING** исполняет функции **WHERE.**
* **ALTER TABLE** *имя\_таблицы* **ADD COLUMN** *имя\_столбца тип;* - добавление столбца.
* **FIRST** *-* новый столбец будет первым, и **AFTER**- указывает после какого столбца поместить новый.
* **UPDATE** *имя\_таблицы* **SET** *имя\_столбца=значение\_столбца* **WHERE** *условие* - для обновления уже существующих данных.
* **ALTER****TABLE** *имя\_таблицы* **CHANGE** *старое\_имя\_столбца новое\_имя\_столбца тип;* - измение названия столбца.
* **ALTER TABLE** *имя\_таблицы* **MODIFY** *имя\_столбца новый\_тип;* - изминение типа данных столбца.
* **DELETE FROM** *имя\_таблицы* **WHERE** *условие;* - удаление строк из столбца.
* **AVG()** - Функция возвращает среднее значение столбца.
* **COUNT()** - Функция возвращает число строк в столбце.
* **MAX()** - Функция возвращает самое большое значение в столбце.
* **MIN()** - Функция возвращает самое маленькое значение в столбце.
* **SUM()** - Функция возвращает сумму значений столбца.
* **SELECT** *имя\_таблицы\_1.имя\_столбца\* имя\_таблицы\_2.имя\_столбца* **AS** *имя \_вычисляемого\_столбца* **FROM** *имя\_таблицы\_1, имя\_таблицы\_2* – создание дополнительного столбца для вывода данных. Ее нельзя использовать, так как она не находиться в какой-либо таблице. Для таких случаев существуют Представления.
* **CREATE VIEW** *имя\_представления* **AS** *запрос*; - создание представления.
* **CONCAT(***str1,str2...***)**Возвращает строку, созданную путем объединения аргументов (аргументы указываются в скобках - str1,str2...). Добавляем пробел " ", как аргумент, для читабельности.
* **SELECT CONCAT\_WS(***' ', имя\_столбца1, имя\_столбца2***) FROM** *имя\_таблицы*; - если аргументов много, используем этот синтаксис для рациональности. Первым аргументом ставим разделитель.
* **INSERT(***str, pos, len, new\_str***)** Возвращает строку str, в которой подстрока, начинающаяся с позиции pos и имеющая длину len символов, заменена подстрокой new\_str.
* **LPAD(***str, len, dop\_str***)** Возвращает строку str, дополненную слева строкой dop\_str до длины len.
* **RPAD(***str, len, dop\_str***)** Возвращает строку str, дополненную справа строкой dop\_str до длины len.
* **LTRIM(***str***)** Возвращает строку str, в которой удалены все начальные пробелы. Эта строковая функция удобна для корректного отображения информации в случаях, когда при вводе данных допускаются случайные пробелы.
* **RTRIM(***str***)** Возвращает строку str, в которой удалены все конечные пробелы.
* **TRIM(***str***)** Возвращает строку str, в которой удалены все начальные и конечные пробелы.
* **LOWER(***str***)** Возвращает строку str, в которой все символы переведены в нижний регистр. С русскими буквами работает некорректно, поэтому лучше не применять.
* **UPPER(***str***)** Возвращает строку str, в которой все символы переведены в верхний регистр. С русскими буквами так же лучше не применять.
* **LENGTH(***str***)** Возвращает длину строки str.
* **LEFT(***str, len***)** Возвращает len левых символов строки str.
* **RIGHT(***str, len***)** Возвращает len правых символов строки str.
* **REPEAT(***str, n***)** Возвращает строку str n-количество раз.
* **REPLACE(***str, pod\_str1, pod\_str2***)** Возвращает строку str, в которой все подстроки pod\_str1 заменены подстроками pod\_str2.
* **REVERSE(***str***)** Возвращает строку str, записанную в обратном порядке.
* **LOAD\_FILE(***file\_name***)** Эта функция читает файл file\_name и возвращает его содержимое в виде строки.
* **CURDATE()**, **CURTIME()** и **NOW()** - Первая функция возвращает текущую дату, вторая - текущее время, а третья - текущую дату и время.
* **ADDDATE(***date*, **INTERVAL** *value***)** - Функция возвращает дату *date*, к которой прибавлено значение *value*. Значение *value* может быть отрицательным, тогда итоговая дата уменьшится. В качестве значения *value* могут выступать не только дни, но и недели (**WEEK**), месяцы (**MONTH**), кварталы (**QUARTER**) и годы (**YEAR**).
* **SUBDATE(***date*, **INTERVAL** *value***)** - функция идентична предыдущей, но производит операцию вычитания, а не сложения.
* **PERIOD\_ADD(***period, n***)** - функция добавляет n месяцев к значению даты period. Нюанс: значение даты должно быть представлено в формате YYYYMM.
* **TIMESTAMPADD(***interval, n, date***)** - функция добавляет к дате *date* временной интервал *n*, значения которого задаются параметром *interval*. Возможные значения параметра interval: **FRAC\_SECOND** - микросекунды **SECOND** - секунды **MINUTE** - минуты **HOUR** - часы **DAY** - дни **WEEK** - недели **MONTH** - месяцы **QUARTER** - кварталы **YEAR** – годы.
* **TIMEDIFF(***date1, date2***)** - вычисляет разницу в часах, минутах и секундах между двумя датами.
* **DATEDIFF(***date1, date2***)** - вычисляет разницу в днях между двумя датами.
* **PERIOD\_DIFF(***period1, period2***) -** функция вычисляет разницу в месяцах между двумя датами, представленными в формате YYYYMM.
* **TIMESTAMPDIFF(***interval, date1, date2***) -** функция вычисляет разницу между датами *date2* и *date1* в единицах, указанных в параметре *interval*.
* **SUBTIME(***date, time***)** функция вычитает из времени date время time.
* **DATE(***datetime***) -**  возвращает дату, отсекая время.
* **TIME(***datetime***) -** возвращает время, отсекая дату.
* **TIMESTAMP(***date***)** - функция принимает дату *date* и возвращает полный вариант со временем.
* **DAY(***date***)** и **DAYOFMONTH(***date***) -** функции-синонимы, возвращают из даты порядковый номер дня месяца.
* **DAYNAME(**date**)**, **DAYOFWEEK(***date***)** и **WEEKDAY(***date***) -** функции возвращают день недели, первая - его название, вторая - номер дня недели (отсчет от 1 - воскресенье до 7 - суббота), третья - номер дня недели (отсчет от 0 - понедельник, до 6 – воскресенье.
* **WEEK(***date***)**, **WEEKOFYEAR(***datetime***) -** обе функции возвращают номер недели в году, первая для типа *date*, а вторая - для типа *datetime*, у первой неделя начинается с воскресенья, у второй - с понедельника.
* **MONTH(***date***) и MONTHNAME(***date***)** - обе функции возвращают значения месяца. Первая - его числовое значение (от 1 до 12), вторая - название месяца.
* **QUARTER(***date***)** - функция возвращает значение квартала года (от 1 до 4).
* **YEAR(***date***) -** функция возвращает значение года (от 1000 до 9999).
* **DAYOFYEAR(***date*) возвращает порядковый номер дня в году (от 1 до 366).
* **HOUR(***datetime***)** возвращает значение часа для времени (от 0 до 23).
* **MINUTE(***datetime***)** возвращает значение минут для времени (от 0 до 59).
* **SECOND(***datetime***)** возвращает значение секунд для времени (от 0 до 59).
* **EXTRACT(***type* **FROM** *date***)** возвращает часть date определяемую параметром *type*:
* **SELECT EXTRACT(***YEAR* **FROM** *'2011-04-17 23:15:18'***) AS** *year***,**
* **EXTRACT(***MONTH* **FROM** *'2011-04-17 23:15:18'***) AS** *mon***,**
* **EXTRACT(***DAY* **FROM** *'2011-04-17 23:15:18'***) AS** *day***,**
* **EXTRACT(***HOUR* **FROM** *'2011-04-17 23:15:18'***) AS** *hour***,**
* **EXTRACT(***MINUTE* **FROM** *'2011-04-17 23:15:18'***) AS** *min***,**
* **EXTRACT(***SECOND* **FROM** *'2011-04-17 23:15:18'***) AS** *sec***;**
* **TO\_DAYS(***date***)** и **FROM\_DAYS(***n***)** взаимообратные функции. Первая преобразует дату в количество дней, прошедших с нулевого года. Вторая, наоборот, принимает число дней, прошедших с нулевого года и преобразует их в дату.
* **UNIX\_TIMESTAMP(***date***)** и **FROM\_UNIXTIME(***n***)** взаимообратные функции. Первая преобразует дату в количество секунд, прошедших с 1 января 1970 года. Вторая, наоборот, принимает число секунд, с 1 января 1970 года и преобразует их в дату.
* **TIME\_TO\_SEC(***time***)** и **SEC\_TO\_TIME(***n***)** взаимообратные функции. Первая преобразует время в количество секунд, прошедших от начала суток. Вторая, наоборот, принимает число секунд с начала суток и преобразует их во время.
* **MAKEDATE(***year, n***)** функция принимает год и номер дня в году и преобразует их в дату.
* **CREATE PROCEDURE** *имя\_процедуры* **(***параметры***) begin** *операторы* **end** – создание процедуры.
* **DELIMITER // -** точка с запятой означает конец запроса и отправляет его на выполнение, что в данном случае неприемлемо. Поэтому, прежде, чем написать процедуру необходимо переопределить разделитель с ; на "//", чтобы запрос не отправлялся раньше времени. Таким образом, мы указали СУБД, что выполнять команды теперь следует после //. Следует помнить, что переопределение разделителя осуществляется только на один сеанс работы, т.е. при следующем сеансе работы с MySql разделитель снова станет точкой с запятой и при необходимости его придется снова переопределять.
* **call** *имя\_процедуры*(*параметры*)// - вызов процедуры.
* **DROP PROCEDURE** *название\_процедуры* – удаление процедуры.
* **IF  EXISTS -** берет подзапрос, как аргумент, и оценивает его как верный, если подзапрос возвращает какие-либо записи и неверный, если тот не делает этого.
* **SHOW PROCEDURE STATUS -** позволяет просмотреть список имеющихся хранимых процедур. Правда просматривать этот список не очень удобно, т.к. по каждой процедуре выдается информация об имени БД, к которой процедура принадлежит, ее типе, учетной записи, от имени которой была создана процедура, о дате создания и изменения процедуры и т.д. И все-таки, если вам необходимо посмотреть, какие процедуры у вас есть, то стоит воспользоваться этим оператором.
* **SHOW CREATE PROCEDURE** *имя\_процедуры*- позволяет получить информацию о конкретной процедуре, в частности просмотреть ее код. Вид для просмотра также не очень удобный, но разобраться можно.
* **SELECT** *\** **FROM** *mysql.proc//*
* db - имя БД, в которую сохранена процедура.
* name - имя процедуры.
* *param\_list*- список параметров процедуры.
* *body* - тело процедуры.
* *comment* - комментарий к хранимой процедуре.
* **SELECT** *name* **FROM** *mysql.proc* **WHERE** **db**=”*имя\_базы\_данных*”// - список процедур в БД.
* **SELECT** *body* **FROM** *mysql.proc* **WHERE** *name*=*'имя\_процедуры'*// - просмотр кода процедуры.
* **COMMENT** *'здесь комментарий'*. - сразу после списка параметров, но еще до начала тела хранимой процедуры.
* **CREATE PROCEDURE** *имя\_процедуры (параметры)* **begin** **IF***(условие)* **THEN** *запрос* *1*; **ELSE** *запрос 2*; **END I**F; **end** //
* **ELSEIF***(условие)* **THEN** *запрос 2*;