Mysql

启动：net start mysql（服务名称）

停止：net stop mysql

提示错误：拒绝访问

用管理员身份启动cmd

# Linux安装MySQL

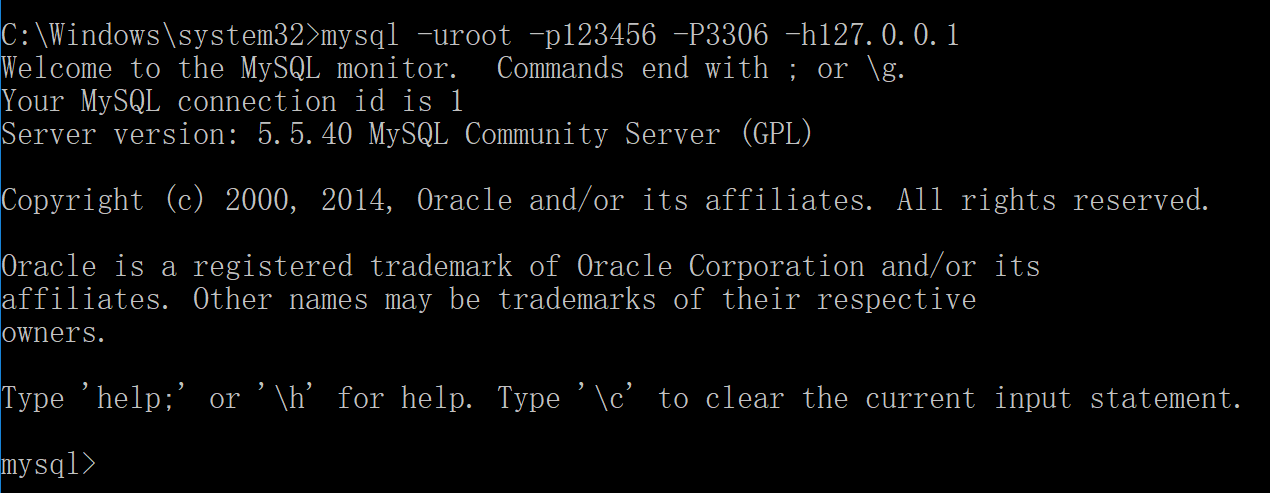
<https://www.jianshu.com/p/276d59cbc529>

# 登录、退出

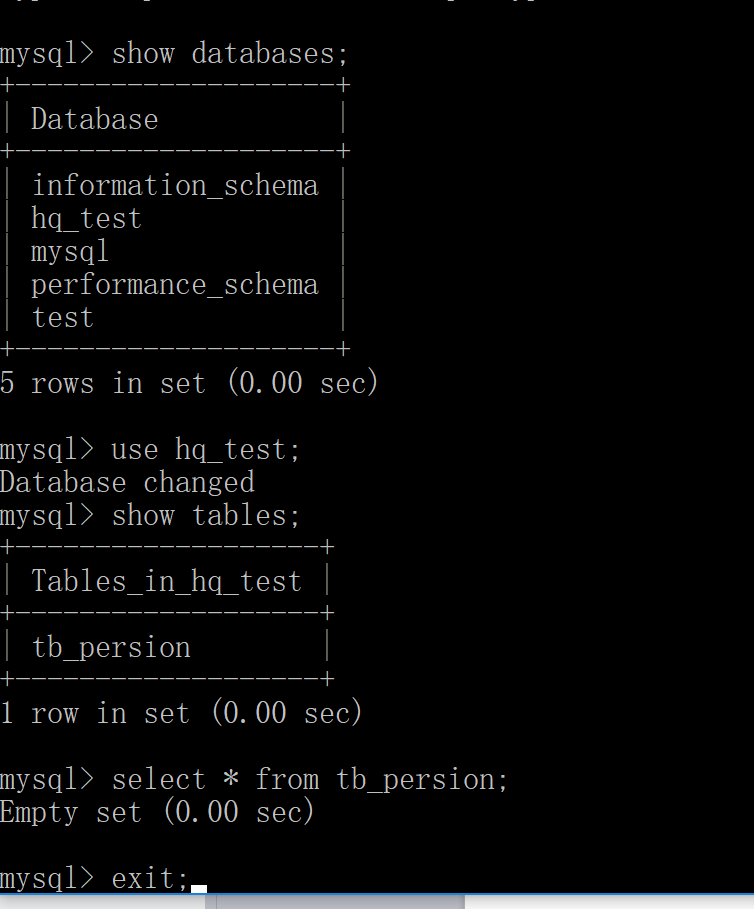


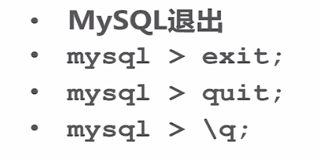
登录u:用户名 p:密码

P(大写P)端口号 -h:ip地址



退出





# 修改前面的提示符





# 常用命令



# 规范



# 创建数据库



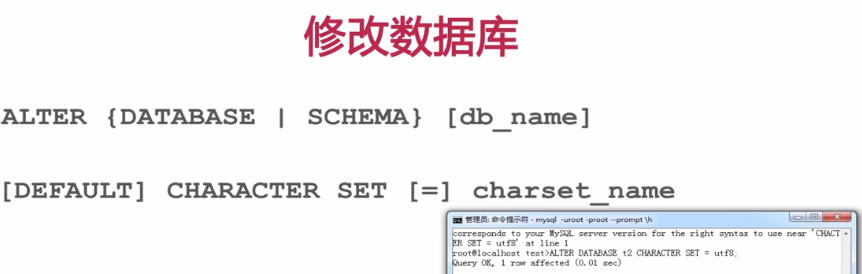
# 查看数据库



# 修改数据库编码方式

查看数据库编码方式：

show create database hq\_test;



# 删除数据库



# 数据类型

# 整型



# 浮点型



# 时间类型

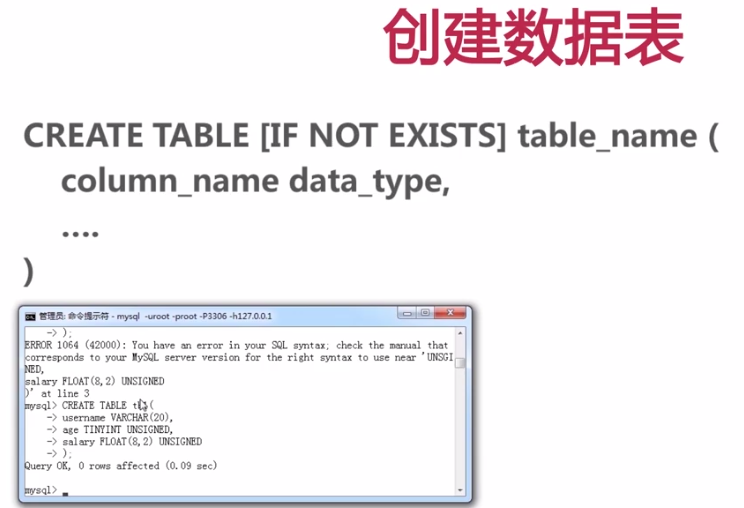


一般用时间戳来存储

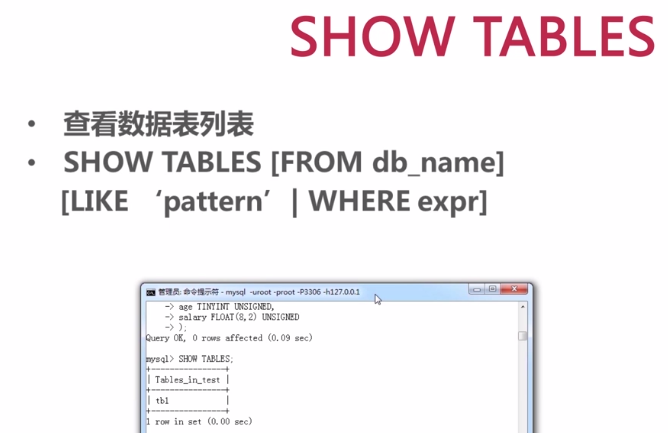
# 字符型



# 创建表



# 查看数据库表

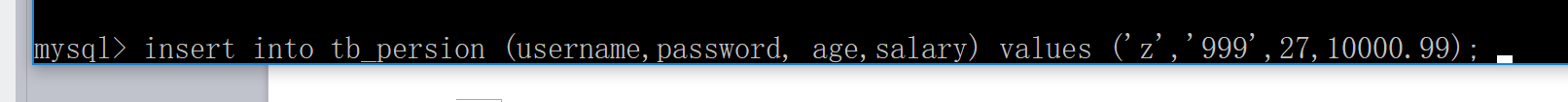


# 查看表结构



# 插入数据





省略字段名的话，那么所有字段都需要赋值。

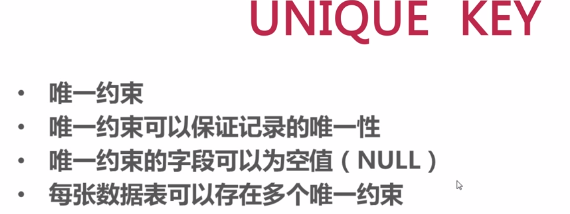
# 查询语句



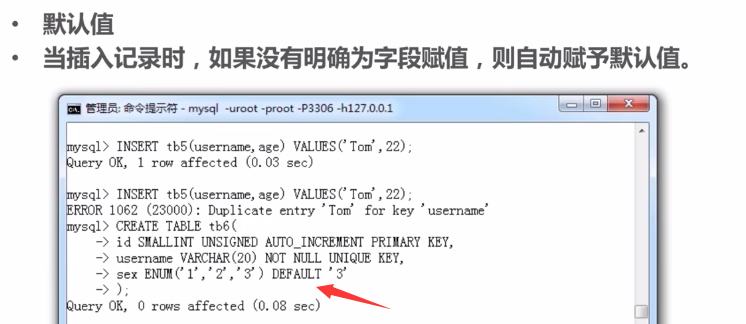
# 主键primary key



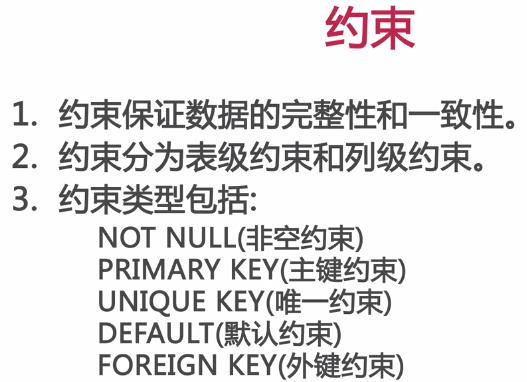
# 唯一约束unique key



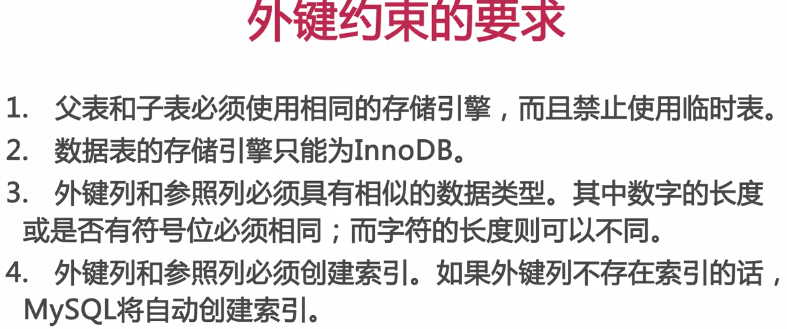
# 默认值

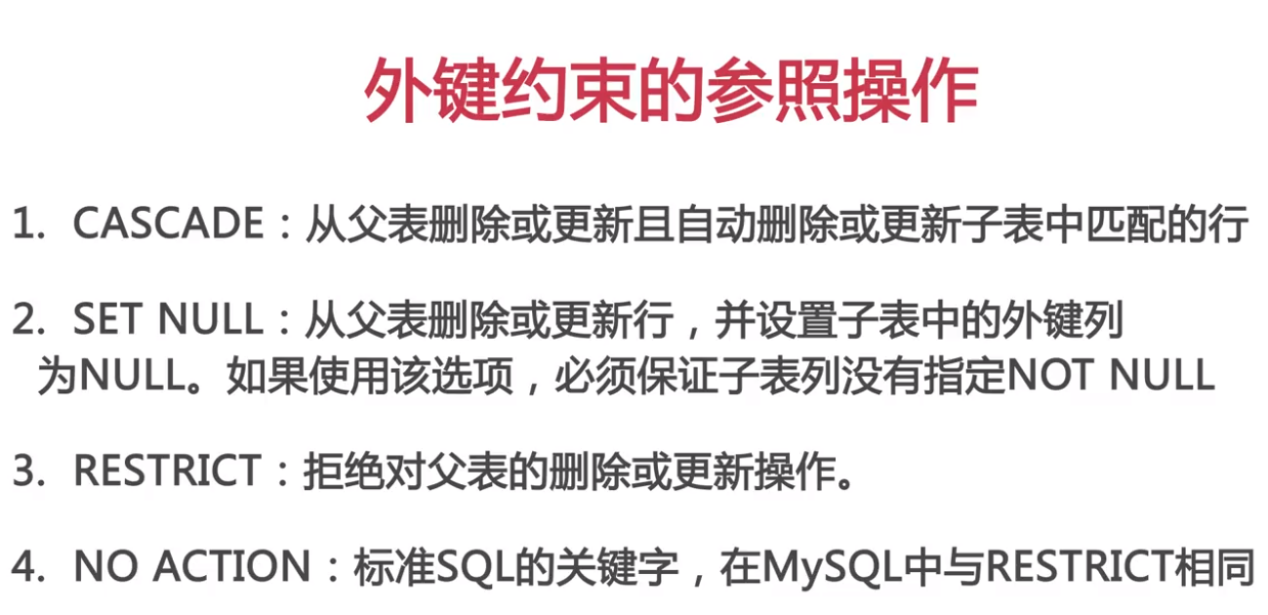


# 约束



# 外键约束(存储引擎为InnoDB)





# 修改存储引擎



# 修改数据库表结构

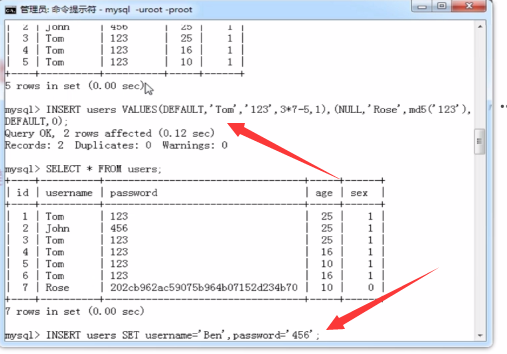
增加



删除一个或多个



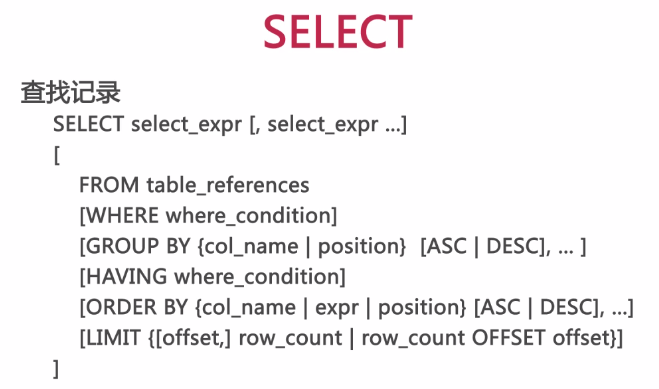
# 插入(两种方式)



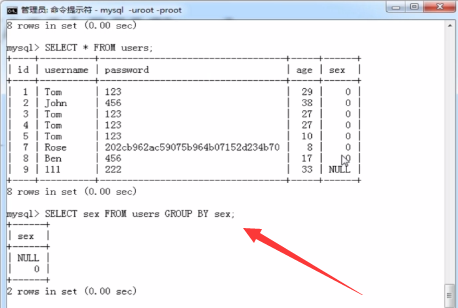
# 更新数据



# 查询数据



# Group by（会将字段数据去重）



对于选择最大值最小值等操作的字段必须是字符型，不能是varchar等类型。

# Having (接在group by后面使用)聚合函数

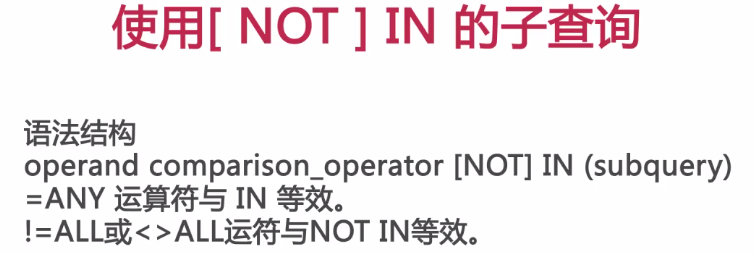
# 子查询 ANY SOME ALL关键字



用法： select \* from tb\_persion where id > ANY (select age from tb\_user);

解释：大于子查询多个返回值中的最小值(ANY)

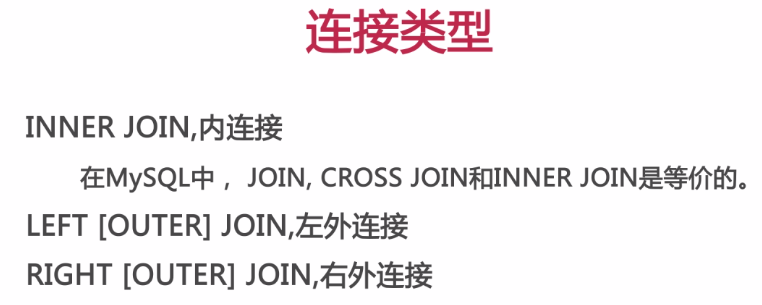
# IN的用法

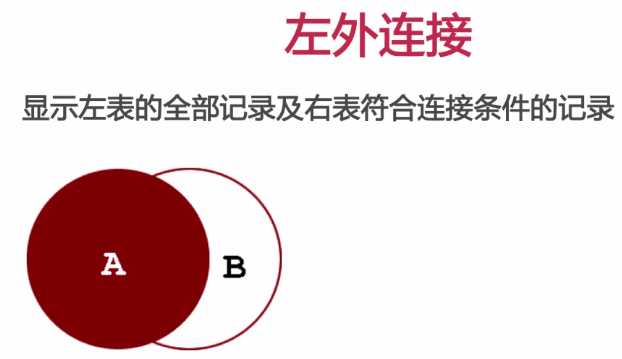
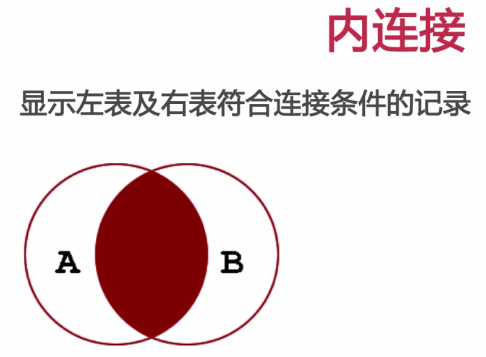


SELECT \* FROM tb\_persion WHERE id in (SELECT id FROM tb\_persion);

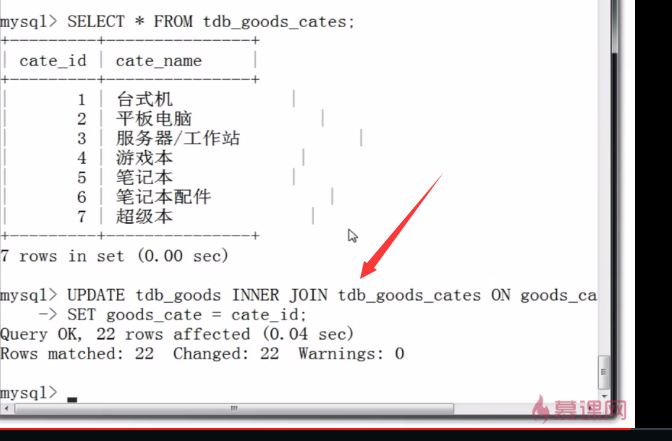
SELECT \* FROM tb\_persion WHERE id in (1,3);

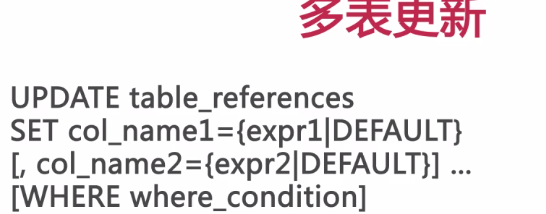
# 连接类型(左、右、内连接)





# 多表更新





多表更新语句

UPDATE tb\_persion\_bk2 a INNER JOIN tb\_info b ON a.bk\_id=b.id

SET a.username=b.address,a.password=b.hobby;

假设5表有六条数据，b表有2条数据。

内连接：放回的数据为最小数据行数。

SELECT a.age,b.hobby FROM tb\_persion a INNER JOIN tb\_info b ON a.id=b.id;



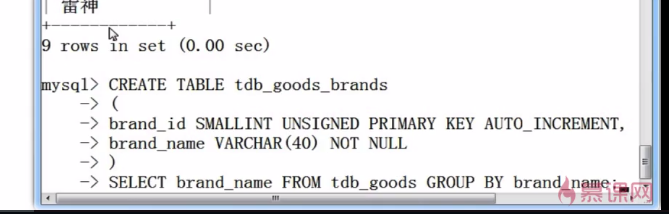
左连接：(那么左表为父表，它有多少条数据就返回多少条数据，如果关联的表比它少，与null显示)

SELECT a.age,b.hobby FROM tb\_persion a LEFT JOIN tb\_info b ON a.id=b.id;



右连接则与左连接正好相反

# 关联创建表



创建表时，将另一张表的数据填充进去

CREATE TABLE tb\_persion\_bk (

bk\_id INT UNSIGNED PRIMARY KEY AUTO\_INCREMENT,

username varchar(255),

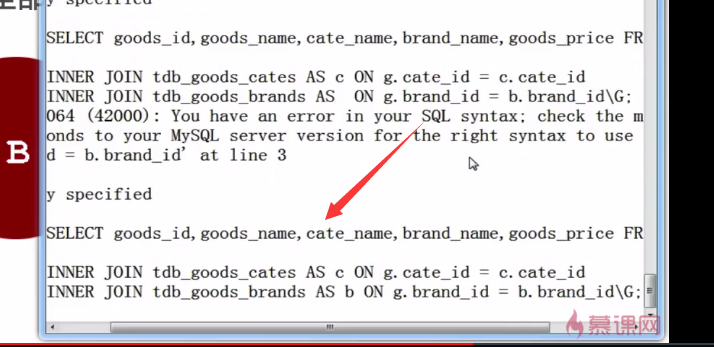
password varchar(255)

)

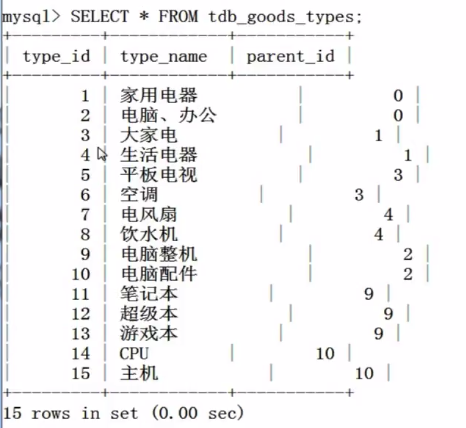
select username,password FROM tb\_persion;

注意事项：查询出来的字段类型要与创建时的类型一致。

# 三张表联合查询



# 无限级分类设计



根据parent\_id关联type\_id找到子类。

比如【大家电】属于【家用电器】

查询方式采用【自身连接】

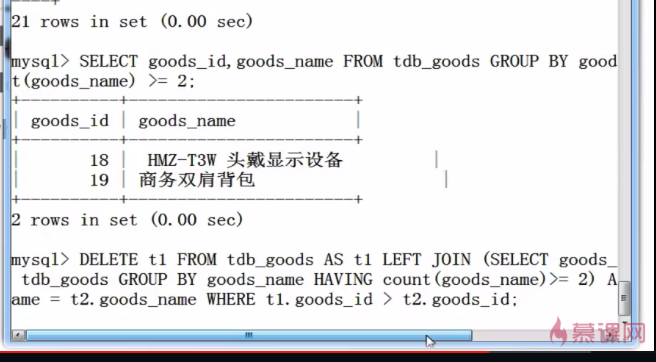
如何查询：思路，假设tdb\_goods\_types的右边还有一张tdb\_goods\_types表，要找出左边父表的type\_name属于哪个父类(一级分类)

Select a.type\_id , a.type\_name,b.type\_name AS parent from tdb\_goods\_types a LEFT JOIN

tdb\_goods\_types b ON a.parent\_id=b.type\_id;

# 自身连接

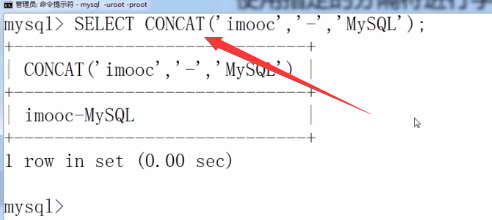
# 多表删除

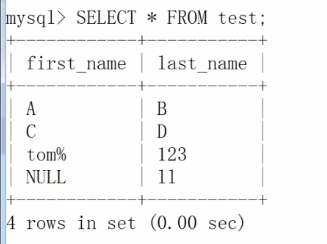


# 字符函数



Concat()函数使用：



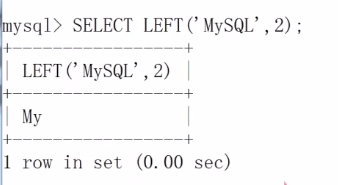


Select CONCAT(first\_name,last\_name) AS full\_name FROM test;

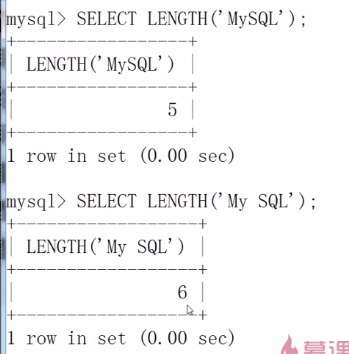
Format()数值格式：

Format(age,2);保留两位小数，四舍五入。

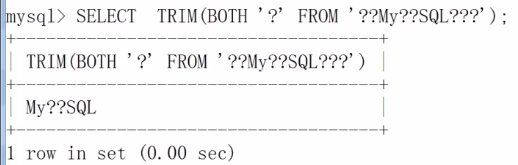
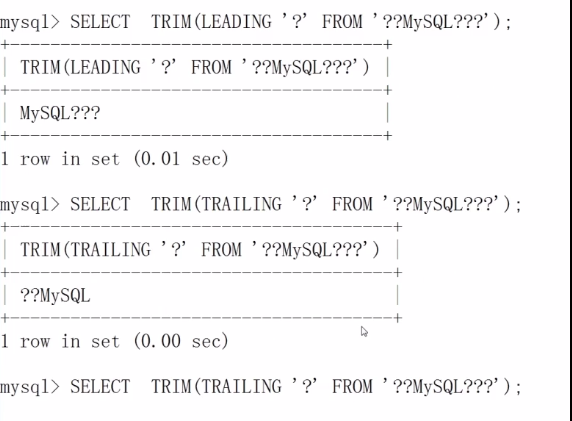
Left用法：第二个参数为位数



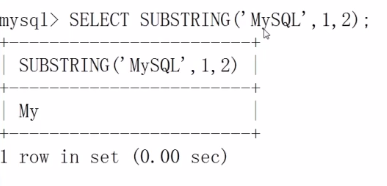


空格也算长度

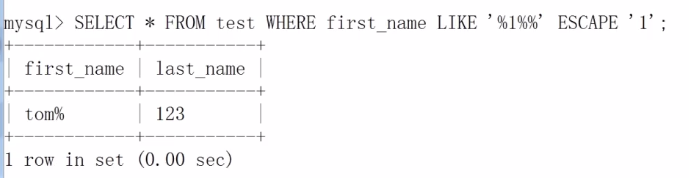
前导与后导







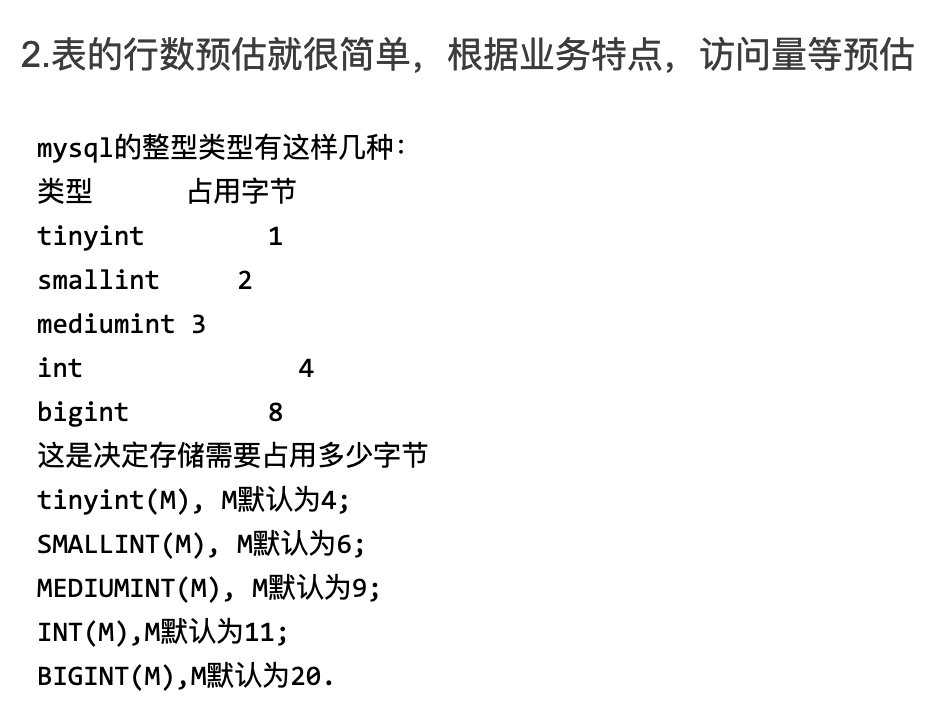
含有%但需要通配符



# MySQL各个数据类型默认值

参考博客：

<https://blog.csdn.net/weixin_41926301/article/details/94509640>



# linux安装MySQL

参考博客

Linux下安装mysql-5.7.24：

<https://www.jianshu.com/p/276d59cbc529>

<https://blog.csdn.net/qq_37598011/article/details/93489404>

启动的方式参考这个：

<https://blog.csdn.net/qq_37598011/article/details/93489404>

授权方式写的不一样：

<https://www.cnblogs.com/wendy-0901/p/12673705.html>

## 安装前检查是否已经安装过

一定要卸载干净，删除干净

|  |
| --- |
| [root@localhost local]# rpm -qa | grep mysql  [root@localhost local]# |

这样就表示没有，如果已经存在，就需要删除再重新安装。

查看mysql文件夹是否存在：

whereis mysql

存在就删除

find / -name mysql

参考博客： <https://www.cnblogs.com/wpnr/p/14072634.html>

//检查系统中有无安装过mysql

rpm -qa|grep mysql

//查询所有mysql 对应的文件夹，全部删除whereis mysql

find / -name mysql

# 查看系统自带的Mariadb

[root@CDH-141 ~]# rpm -qa|grep mariadb

mariadb-libs-5.5.44-2.el7.centos.x86\_64

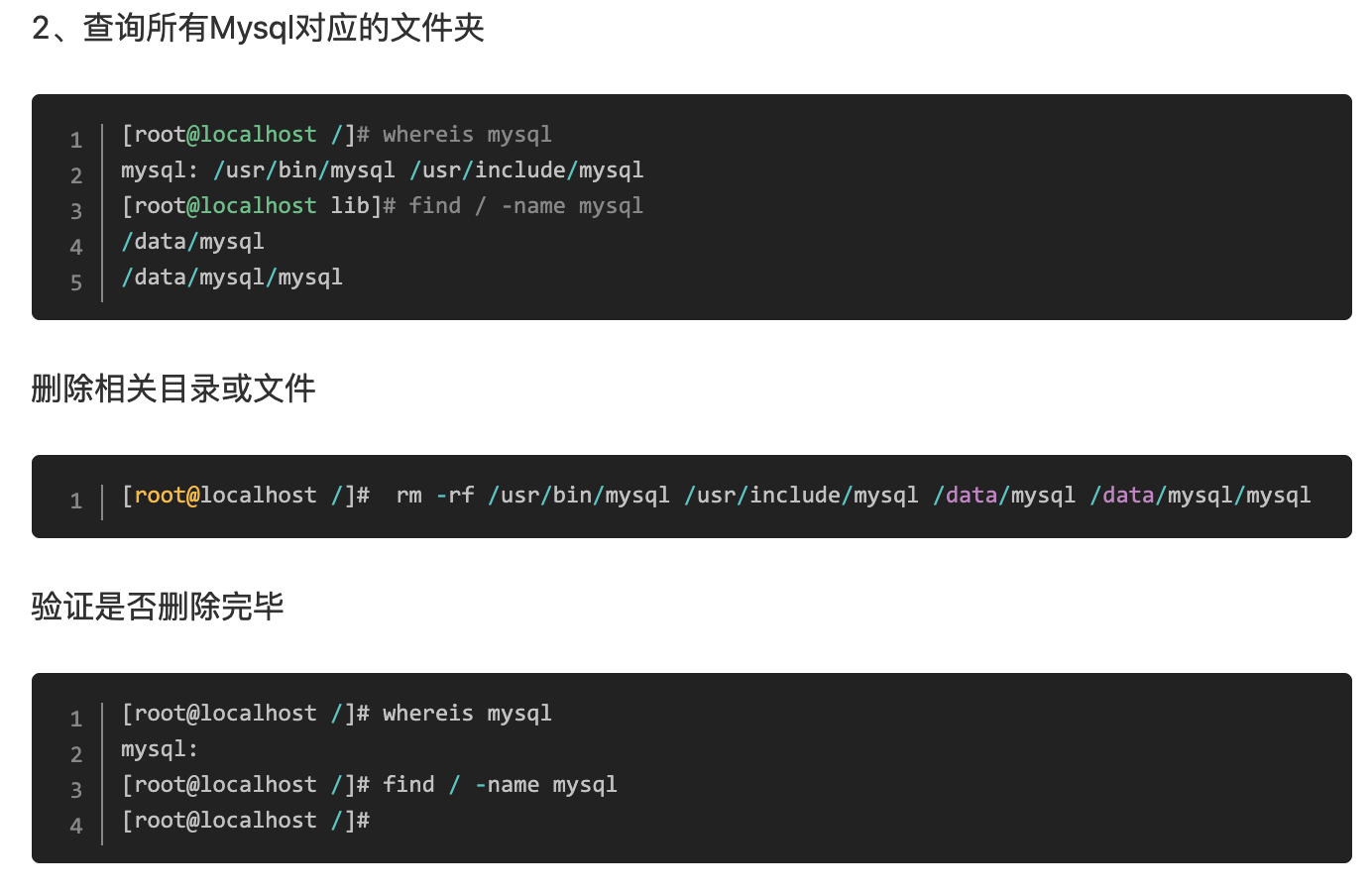
# 卸载系统自带的Mariadb

[root@CDH-141 ~]# rpm -e --nodeps mariadb-libs-5.5.44-2.el7.centos.x86\_64

# 删除etc目录下的my.cnf ，一定要删掉，等下再重新建，之前我将就用这个文件，后面改配置各种不生效

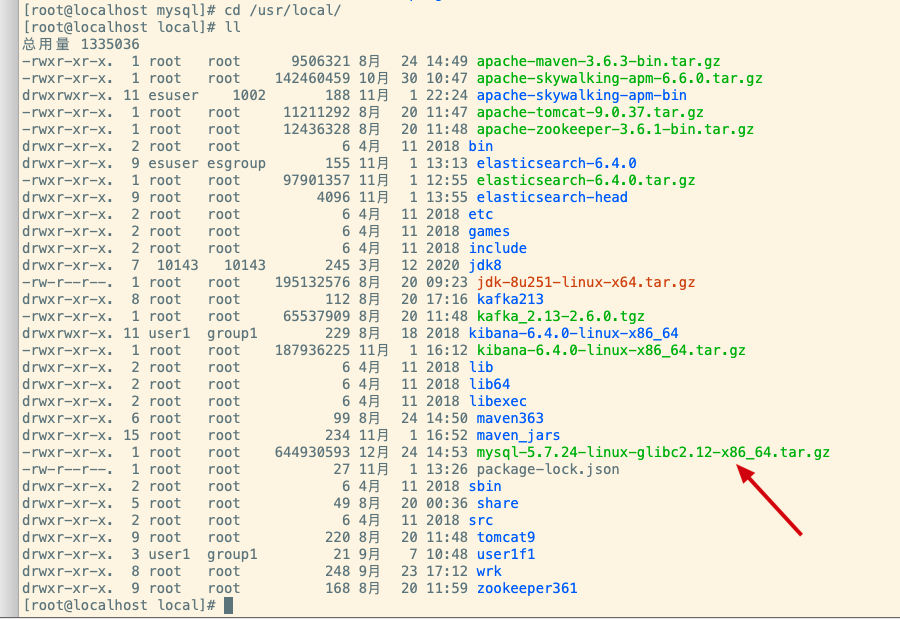
[root@CDH-141 ~]# rm /etc/my.cnf

rm -rf /var/lib/mysql



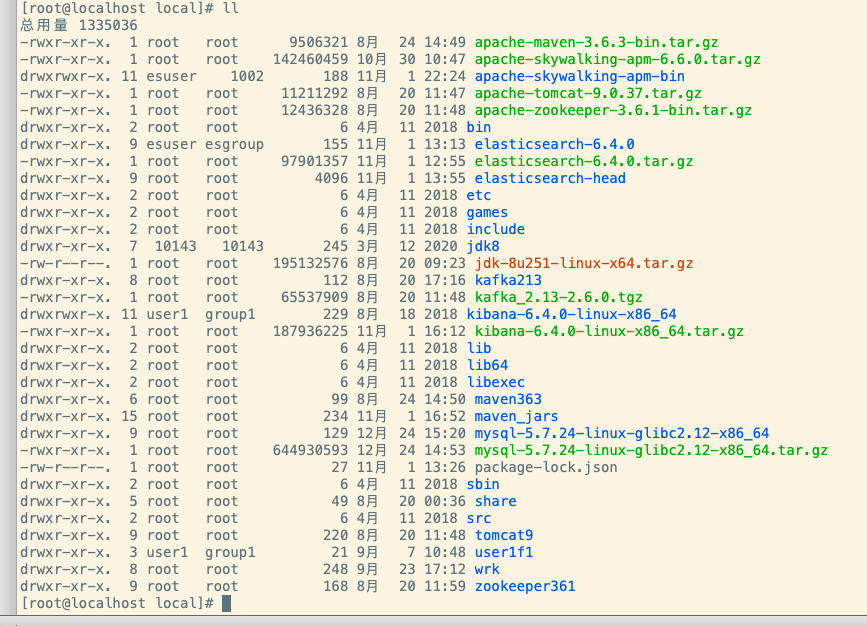
## 上传并解压

目录为： /usr/local/下



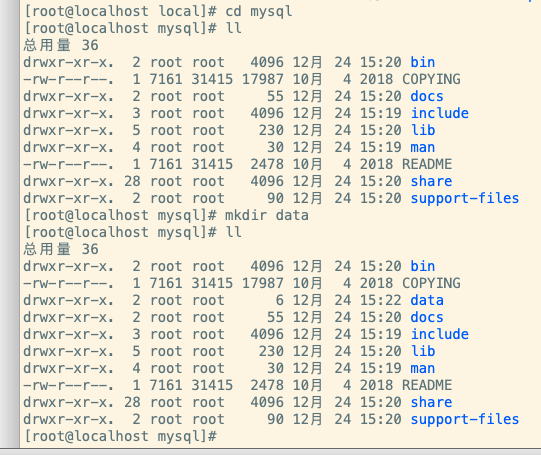
解压：

tar -zxvf



重新命名为mysql

## 在mysql目录下创建data目录



## mysql用户和用户组的创建

检查mysql用户组和用户是否存在，如果没有，则创建





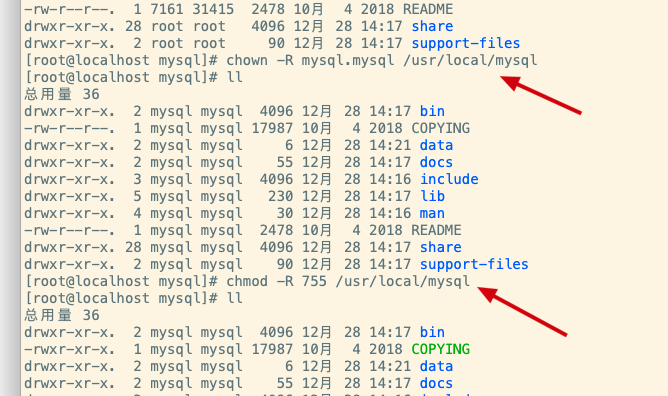




#useradd -r参数表示mysql用户是系统用户，不可用于登录系统

可以为后续操作使用

## 为文件夹赋权



chown -R mysql.mysql /usr/local/mysql

chmod -R 755 /usr/local/mysql

## 编译安装并初始化mysql

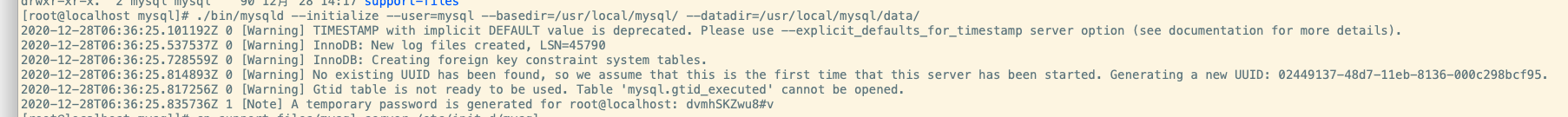
务必记住初始化输出日志末尾的密码（数据库管理员临时密码）

进入bin目录

cd /usr/local/mysql/bin

初始化mysql:

|  |
| --- |
| [root@localhost /]# cd /usr/local/mysql/bin  [root@localhost bin]# ./mysqld --initialize --user=mysql --datadir=/usr/local/mysql/data/ --basedir=/usr/local/mysql/ |



最后几位是密码。

## 编辑配置文件my.cnf

vim /etc/my.cnf

|  |
| --- |
| [mysqld]  #设置mysql的安装目录  basedir =/usr/local/mysql  #设置mysql数据库的数据存放目录  datadir = /usr/local/mysql/data  #设置端口  port = 3306  socket = /tmp/mysql.sock  #设置字符集  character-set-server=utf8  max\_connections=600  innodb\_file\_per\_table=1  lower\_case\_table\_names=1  symbolic-links=0  #日志存放目录  [mysqld\_safe]  log-error = /usr/local/mysql/data/mysqld.log  pid-file = /usr/local/mysql/data/mysqld.pid  #允许时间类型的数据为零(去掉NO\_ZERO\_IN\_DATE,NO\_ZERO\_DATE)  sql\_mode=ONLY\_FULL\_GROUP\_BY,STRICT\_TRANS\_TABLES,ERROR\_FOR\_DIVISION\_BY\_ZERO,NO\_AUTO\_CREATE\_USER,NO\_ENGINE\_SUBSTITUTION  #ONLY\_FULL\_GROUP\_BY,STRICT\_TRANS\_TABLES,NO\_ZERO\_IN\_DATE,NO\_ZERO\_DATE,ERROR\_FOR\_DIVISION\_BY\_ZERO,NO\_AUTO\_CREATE\_USER,NO\_ENGINE\_SUBSTITUTION |

lower\_case\_table\_names：是否区分大小写，1表示存储时表名为小写，操作时不区分大小写；0表示区分大小写；不能动态设置，修改后，必须重启才能生效：

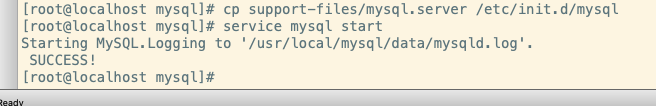
character\_set\_server：设置数据库默认字符集，如果不设置默认为latin1

innodb\_file\_per\_table：是否将每个表的数据单独存储，1表示单独存储；0表示关闭独立表空间，可以通过查看数据目录，查看文件结构的区别；

## 测试启动mysql服务器

先将mysql.server放置到/etc/init.d/mysql中 【把启动脚本放到开机初始化目录】

|  |
| --- |
| cp /usr/local/mysql/support-files/mysql.server /etc/init.d/mysql |

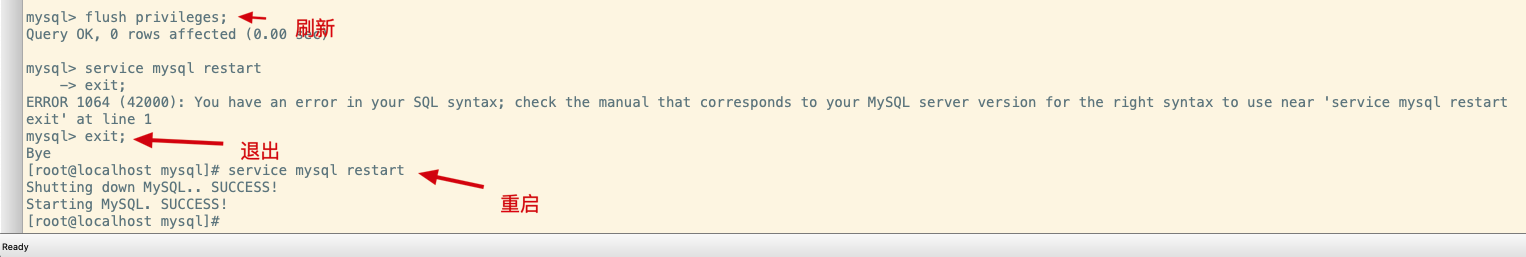


启动： service mysql start

修改密码：

|  |
| --- |
| 进入mysql并更改密码  cd /usr/local/mysql-5.7.28  ./bin/mysql -u root -p  mysql> set password=password('123456');  mysql> grant all privileges on \*.\* to root@'%' identified by '123456';  mysql> flush privileges;  添加远程访问权限  mysql> use mysql;  mysql> update user set host='%' where user = 'root';  mysql> flush privileges;  如果更改时报错：ERROR 1062 (23000): Duplicate entry '%-root' for key 'PRIMARY'，就先查询一下是否已更改，最后执行刷新。    重启mysql生效  service mysql restart |





## 两种启动方式:

1,没有加入开机自启，需要进入到bin目录中去启动

[root@localhost /]# /usr/local/mysql/support-files/mysql.server start

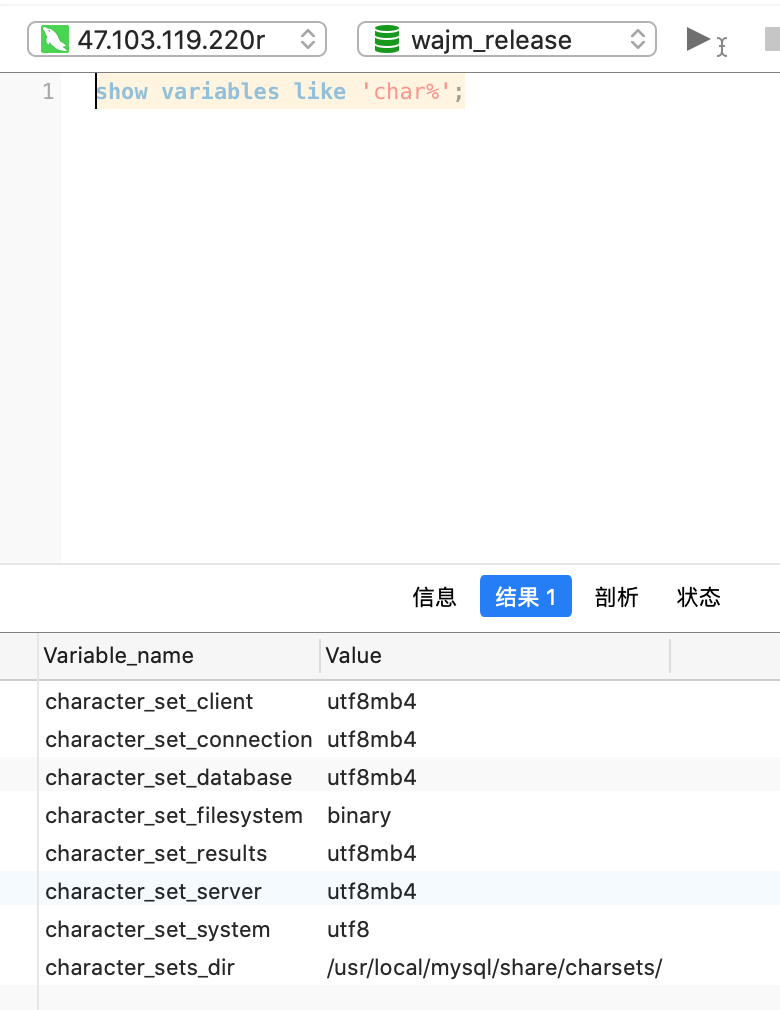
2，已经加入了开机自启

service mysql start

service mysql stop

## 查看MySQL字符集

show variables like 'char%';



# 使用终端进行用户操作

用root账号登录服务器，以39为例

## 登录：

mysql -u root -p

密码: mysql@dev

## 创建数据库：

创建一个数据库名为 spdjlyy的数据库。

|  |
| --- |
| CREATE DATABASE `spdjlyy` DEFAULT CHARACTER SET utf8mb4 COLLATE utf8mb4\_general\_ci; |

## 为数据库分配其他用户

创建一个用户【省略】

查看我们已经有什么用户，把这个用户绑定到我们的数据库上。

use mysql;

show tables;

select \* from user;

比如你的user表中有个bkgxapp的账号。

## 分配账号

grant all privileges on spdjlyy.\* to 'bkgxapp'@'%' identified by '2020#all^APP';

改动用户名和密码即可

flush privileges;

刷新权限

grant create,alter,drop,select,insert,update,delete on spdjlyy.\* to 'bkgxapp'@'%';

为用分配数据库操作的增删改查权限

flush privileges;

刷新权限

## 允许外网访问

#mysql -u root -p

Enter password：

mysql>use mysql;

mysql>update user set host='%' where user='root' and host='localhost';

mysql>flush privileges;

或者update user set host='%' where user='root' ;

flush privileges;//刷新权限

# window安装mysql

## 下载地址

<https://downloads.mysql.com/archives/community/>

<https://www.jb51.net/article/103685.htm>

# linux安装5.7.32

<https://www.cnblogs.com/wendy-0901/p/12673705.html>

# MySQL备份

主流的三种备份方式：<https://blog.csdn.net/xuanxuan_good/article/details/53671258>

xtrabakup <https://www.cnblogs.com/linuxk/p/9372990.html>

# MySQL配置

|  |
| --- |
| [client]  port = 3306  socket = /tmp/mysql.sock  [mysqld]  ###############################基础设置#####################################  #Mysql服务的唯一编号 每个mysql服务Id需唯一  server-id = 1  #服务端口号 默认3306  port = 3306  #mysql安装根目录  basedir = /opt/mysql  #mysql数据文件所在位置  datadir = /opt/mysql/data  #临时目录 比如load data infile会用到  tmpdir = /tmp  #设置socke文件所在目录  socket = /tmp/mysql.sock  #主要用于MyISAM存储引擎,如果多台服务器连接一个数据库则建议注释下面内容  skip-external-locking  #只能用IP地址检查客户端的登录，不用主机名  skip\_name\_resolve = 1  #事务隔离级别，默认为可重复读，mysql默认可重复读级别（此级别下可能参数很多间隙锁，影响性能）  transaction\_isolation = READ-COMMITTED  #数据库默认字符集,主流字符集支持一些特殊表情符号（特殊表情符占用4个字节）  character-set-server = utf8mb4  #数据库字符集对应一些排序等规则，注意要和character-set-server对应  collation-server = utf8mb4\_general\_ci  #设置client连接mysql时的字符集,防止乱码  init\_connect=‘SET NAMES utf8mb4‘  #是否对sql语句大小写敏感，1表示不敏感  lower\_case\_table\_names = 1  #最大连接数  max\_connections = 400  #最大错误连接数  max\_connect\_errors = 1000  #TIMESTAMP如果没有显示声明NOT NULL，允许NULL值  explicit\_defaults\_for\_timestamp = true  #SQL数据包发送的大小，如果有BLOB对象建议修改成1G  max\_allowed\_packet = 128M  #MySQL连接闲置超过一定时间后(单位：秒)将会被强行关闭  #MySQL默认的wait\_timeout 值为8个小时, interactive\_timeout参数需要同时配置才能生效  interactive\_timeout = 1800  wait\_timeout = 1800  #内部内存临时表的最大值 ，设置成128M。  #比如大数据量的group by ,order by时可能用到临时表，  #超过了这个值将写入磁盘，系统IO压力增大  tmp\_table\_size = 134217728  max\_heap\_table\_size = 134217728  #禁用mysql的缓存查询结果集功能  #后期根据业务情况测试决定是否开启  #大部分情况下关闭下面两项  query\_cache\_size = 0  query\_cache\_type = 0  #####################用户进程分配到的内存设置BEGIN#############################  ##每个session将会分配参数设置的内存大小  #用于表的顺序扫描，读出的数据暂存于read\_buffer\_size中，当buff满时或读完，将数据返回上层调用者  #一般在128kb ~ 256kb,用于MyISAM  #read\_buffer\_size = 131072  #用于表的随机读取，当按照一个非索引字段排序读取时会用到，  #一般在128kb ~ 256kb,用于MyISAM  #read\_rnd\_buffer\_size = 262144  #order by或group by时用到  #建议先调整为2M，后期观察调整  sort\_buffer\_size = 2097152  #一般数据库中没什么大的事务，设成1~2M，默认32kb  binlog\_cache\_size = 524288  ########################用户进程分配到的内存设置END############################  #在MySQL暂时停止响应新请求之前的短时间内多少个请求可以被存在堆栈中  #官方建议back\_log = 50 + (max\_connections / 5),封顶数为900  back\_log = 130  ############################日志设置##########################################  #数据库错误日志文件  log\_error = error.log  #慢查询sql日志设置  slow\_query\_log = 1  slow\_query\_log\_file = slow.log  #检查未使用到索引的sql  log\_queries\_not\_using\_indexes = 1  #针对log\_queries\_not\_using\_indexes开启后，记录慢sql的频次、每分钟记录的条数  log\_throttle\_queries\_not\_using\_indexes = 5  #作为从库时生效,从库复制中如何有慢sql也将被记录  log\_slow\_slave\_statements = 1  #慢查询执行的秒数，必须达到此值可被记录  long\_query\_time = 8  #检索的行数必须达到此值才可被记为慢查询  min\_examined\_row\_limit = 100  #mysql binlog日志文件保存的过期时间，过期后自动删除  expire\_logs\_days = 5  ############################主从复制设置#####################################  #开启mysql binlog功能  log-bin=mysql-bin  #binlog记录内容的方式，记录被操作的每一行  binlog\_format = ROW  #对于binlog\_format = ROW模式时，减少记录日志的内容，只记录受影响的列  binlog\_row\_image = minimal  #master status and connection information输出到表mysql.slave\_master\_info中  master\_info\_repository = TABLE  #the slave‘s position in the relay logs输出到表mysql.slave\_relay\_log\_info中  relay\_log\_info\_repository = TABLE  #作为从库时生效,想进行级联复制，则需要此参数  log\_slave\_updates  #作为从库时生效,中继日志relay-log可以自我修复  relay\_log\_recovery = 1  #作为从库时生效,主从复制时忽略的错误  slave\_skip\_errors = ddl\_exist\_errors  #####################redo log和binlog的关系设置BEGIN#########################  #(步骤1) prepare dml相关的SQL操作，然后将redo log buff中的缓存持久化到磁盘  #(步骤2)如果前面prepare成功，那么再继续将事务日志持久化到binlog  #(步骤3)如果前面成功，那么在redo log里面写上一个commit记录  #当innodb\_flush\_log\_at\_trx\_commit和sync\_binlog都为1时是最安全的，  #在mysqld服务崩溃或者服务器主机crash的情况下，binary log只有可能丢失最多一个语句或者一个事务。  #但是都设置为1时会导致频繁的io操作，因此该模式也是最慢的一种方式。  #当innodb\_flush\_log\_at\_trx\_commit设置为0，mysqld进程的崩溃会导致上一秒钟所有事务数据的丢失。  #当innodb\_flush\_log\_at\_trx\_commit设置为2，只有在操作系统崩溃或者系统掉电的情况下，上一秒钟所有事务数据才可能丢失。  #commit事务时,控制redo log buff持久化磁盘的模式 默认为1  innodb\_flush\_log\_at\_trx\_commit = 2  #commit事务时,控制写入mysql binlog日志的模式 默认为0  #innodb\_flush\_log\_at\_trx\_commit和sync\_binlog都为1时，mysql最为安全但性能上压力也是最大  sync\_binlog = 1  ####################redo log和binlog的关系设置END############################  ############################Innodb设置#####################################  #数据块的单位8k，默认是16k，16kCPU压力稍小，8k对select的吞吐量大  #innodb\_page\_size的参数值也影响最大索引长度，8k比16k的最大索引长度小  #innodb\_page\_size = 8192  #一般设置物理存储的60% ~ 70%  innodb\_buffer\_pool\_size = 1G  #5.7.6之后默认16M  #innodb\_log\_buffer\_size = 16777216  #该参数针对unix、linux，window上直接注释该参数.默认值为NULL  #O\_DIRECT减少操作系统级别VFS的缓存和Innodb本身的buffer缓存之间的冲突  innodb\_flush\_method = O\_DIRECT  #此格式支持压缩, 5.7.7之后为默认值  innodb\_file\_format = Barracuda  #CPU多核处理能力设置，假设CPU是2颗4核的，设置如下  #读多，写少可以设成2:6的比例  innodb\_write\_io\_threads = 4  innodb\_read\_io\_threads = 4  #提高刷新脏页数量和合并插入数量，改善磁盘I/O处理能力  #默认值200（单位：页）  #可根据磁盘近期的IOPS确定该值  innodb\_io\_capacity = 500  #为了获取被锁定的资源最大等待时间，默认50秒，超过该时间会报如下错误:  # ERROR 1205 (HY000): Lock wait timeout exceeded; try restarting transaction  innodb\_lock\_wait\_timeout = 30  #调整buffer pool中最近使用的页读取并dump的百分比,通过设置该参数可以减少转储的page数  innodb\_buffer\_pool\_dump\_pct = 40  #设置redoLog文件所在目录, redoLog记录事务具体操作内容  innodb\_log\_group\_home\_dir = /opt/mysql/redolog/  #设置undoLog文件所在目录, undoLog用于事务回滚操作  innodb\_undo\_directory = /opt/mysql/undolog/  #在innodb\_log\_group\_home\_dir中的redoLog文件数, redoLog文件内容是循环覆盖写入。  innodb\_log\_files\_in\_group = 3  #MySql5.7官方建议尽量设置的大些，可以接近innodb\_buffer\_pool\_size的大小  #之前设置该值较大时可能导致mysql宕机恢复时间过长，现在恢复已经加快很多了  #该值减少脏数据刷新到磁盘的频次  #最大值innodb\_log\_file\_size \* innodb\_log\_files\_in\_group <= 512GB,单文件<=256GB  innodb\_log\_file\_size = 1024M  #设置undoLog文件所占空间可以回收  #5.7之前的MySql的undoLog文件一直增大无法回收  innodb\_undo\_log\_truncate = 1  innodb\_undo\_tablespaces = 3  innodb\_undo\_logs = 128  #5.7.7默认开启该参数 控制单列索引长度最大达到3072  #innodb\_large\_prefix = 1  #5.7.8默认为4个, Inodb后台清理工作的线程数  #innodb\_purge\_threads = 4  #通过设置配置参数innodb\_thread\_concurrency来限制并发线程的数量，  #一旦执行线程的数量达到这个限制，额外的线程在被放置到对队列中之前，会睡眠数微秒，  #可以通过设定参数innodb\_thread\_sleep\_delay来配置睡眠时间  #该值默认为0,在官方doc上，对于innodb\_thread\_concurrency的使用，也给出了一些建议:  #(1)如果一个工作负载中，并发用户线程的数量小于64，建议设置innodb\_thread\_concurrency=0；  #(2)如果工作负载一直较为严重甚至偶尔达到顶峰，建议先设置innodb\_thread\_concurrency=128,  ###并通过不断的降低这个参数，96, 80, 64等等，直到发现能够提供最佳性能的线程数  #innodb\_thread\_concurrency = 0  #强所有发生的死锁错误信息记录到error.log中，之前通过命令行只能查看最近一次死锁信息  innodb\_print\_all\_deadlocks = 1  ############################其他设置########################################  [mysqldump]  quick  max\_allowed\_packet = 128M  [mysql]  no-auto-rehash  [myisamchk]  key\_buffer\_size = 20M  sort\_buffer\_size = 256k  read\_buffer = 2M  write\_buffer = 2M  [mysqlhotcopy]  interactive-timeout  [mysqld\_safe]  #增加每个进程的可打开文件数量  open-files-limit = 28192 |

# 常用配置

|  |
| --- |
| [client]  default-character-set=utf8mb4  port = 3306  [mysql]  default-character-set=utf8mb4  [mysqld]  port = 3306  basedir =/usr/local/mysql  datadir = /usr/local/mysql/data  default-storage-engine = InnoDB  innodb\_file\_per\_table = 1  default-character-set=utf8mb4  collation-server = utf8mb4\_general\_ci  init\_connect='SET NAMES utf8mb4'  lower\_case\_table\_names = 1  wait\_timeout = 28800  log-error = /usr/local/mysql/data/mysqld\_error.log  pid-file = /usr/local/mysql/data/mysqld.pid  slow\_query\_log = 1  slow\_query\_log\_file = /usr/local/mysql/data/slow.log  long\_query\_time = 5  sql\_mode=ONLY\_FULL\_GROUP\_BY,STRICT\_TRANS\_TABLES,ERROR\_FOR\_DIVISION\_BY\_ZERO,NO\_AUTO\_CREATE\_USER,NO\_ENGINE\_SUBSTITUTION |

查看MySQL状态：systemctl status mysqld

启动：systemctl start mysqld