# mysql

## 常用操作:

[root@localhost ~]# grep 'temporary password' /var/log/mysqld.log #第一次登陆之前可查看初始密码

[root@localhost ~]# mysqladmin -uroot –p[oldpassword] password newpassword #修改密码1

mysql> UPDATE user SET password=PASSWORD("new password") WHERE user='root'; #修改密码2要刷新配置

mysql> FLUSH PRIVILEGES;

mysql> SET PASSWORD FOR 'root'@'localhost' = PASSWORD('newpass'); #修改密码3

mysql> alter user root@"localhost" identified by "123qqq...AAA" ; #修改密码4

mysql> show variables like "%password%" ;

mysql>set global validate\_password\_policy=0; //只验证长度,临时修改

[root@sql51 ~]# vim /etc/my.cnf //永久修改

[mysqld]

validate\_password\_length=6

validate\_password\_policy=0

mysql> alter user user() identified by "123456"; //修改登陆密码1

mysql> alter user root@"localhost" identified by "123qqq...AAA" ; //修改登陆密码2

[root@sql51 ~]# mysql -uroot -p123456 mysql #指定mysql库名登录

mysql> system ls /var/lib/mysql #登录MySQL后运行系统命令

mysql> show databases;

mysql> use sys; //切换到sys库

mysql> create database newdb; //新建名为newdb的库

mysql> drop database newdb; //删除名为newdb的库,在当前库也能被删除

**查询授权,结构,配置等信息**

mysql> select database(); //查看当前所在库

mysql> show tables;

**mysql> desc columns\_priv\G** //查看表结构，以列表形式展现，末尾不用分号

**mysql> desc columns\_priv;**  //查看表结构，以表格形式展现末尾需要有分号

**mysql> desc mysql.columns\_priv; //查看mysql里面的columns表**

mysql> desc t3;

+-------+---------------------+------+-----+---------+-------+

| Field | Type | Null | Key | Default | Extra |

+-------+---------------------+------+-----+---------+-------+

| age | tinyint(3) unsigned | YES | | NULL | |

+-------+---------------------+------+-----+---------+-------+

Field:字段名 Type:数据类型 Null 是否允许为空值,

Key:键值(索引) Default:字段的默认值 Extra 额外设置

数据中”” “null” 都不是空值,null才是空(没双引号)

mysql> create table pwlist(

-> name char(16) not null,

-> password char(48)default '',

-> primary key(name)

-> );

包括name、password两列，其中name列作为主键。两个字段值均不允许为空，其中密码列赋予默认空值，相关操作如下所述。

mysql> show create table mydb.student; //show create table查看表的实际创建指令：

mysql> CREATE TABLE mydb.student(

-> 学号 char(9) NOT NULL,

-> 姓名 varchar(4) NOT NULL,

-> 性别 enum('男','女') NOT NULL,

-> 手机号 char(11) DEFAULT '',

-> 通信地址 varchar(64),

-> PRIMARY KEY(学号)

-> ) DEFAULT CHARSET=utf8; //手工指定字符集，采用utf8

mysql> SHOW VARIABLES LIKE 'character%'; //查看字符集,确认更改结果

mysql>show variables like ‘% %’; //搜索数据库中关键字

mysql> show variables like 'have\_dynamic\_loading' ; //查看mysql主从同步复制模块

mysql> show variables like "%password%" ; #查看密码策略

mysql>show grants ; #查看权限默认是看root

mysql>show grants for tom; #可查看tom的权限(查看所有权限)

mysql>show grants for mydba@"%"; #查看mydba用户权限详情,%表示所有库

mysql> select host,user from mysql.user; #查看用户登录授权信息

mysql> show index from t1; #查看索引

mysql> show index from t1\G; #将显示内容竖向显示,一定要大写G

mysql>show slave status\G; #查看主从信息

mysql>show master status; #查看binlog日志

mysql>show engines\G; #查看数据库此版本支持的所有存储引擎以及属性

mysql> show status like '%table\_lock%'; #查看当前表锁状态信息

mysql> show variables like '%autocommit%'; #查看自动提交开关,数据变更立即生效

mysql> show variables like '%secure%'; #查询默认检索文件目录：

mysql> desc columns\_priv\G //查看表结构，以列表形式展现，末尾不用分号

mysql> desc columns\_priv; //查看表结构，以表格形式展现末尾需要有分号

mysql> desc mysql.columns\_priv; //查看mysql里面的columns表

mysql> select \* from t1;

mysql> insert into t1 values ("bob",19,"boy") ;

mysql> update ygt set ygt\_id=8 where ygt\_id=3; //将ygt表中的ygt\_id=3数据中的ygt\_id数据改为8

mysql> delete from db1.t1 where name="bob";

mysql> delete from db1.t1; //delete删除所有表记录（表还在）

mysql> drop table t3; //删除表

假定stuinfo表用来记录每个家庭成员的姓名（name）主键、性别（gender）、出生日期（birth）、职业（job）、与户主关系（relation）。

mysql> CREATE TABLE stuinfo (

-> name varchar(16) NOT NULL,

-> gender enum('male','femal') DEFAULT 'male',

-> birth date NOT NULL,

-> job varchar(16) DEFAULT '',

-> relation varchar(24) NOT NULL,

-> PRIMARY KEY(name)

-> );

## 配置选项

[mysqld]

bind-address=0.0.0.0 #监听地址，有些版本差异，添上不会出问题

server-id= {{id.stdout}} #用ansible批量传送，传参方式获取

binlog-format=mixed #binglog日志格式

log\_bin=/var/log/mysql/mysql-bin #binlog日志存放路径和命名

relay-log=/var/log/mysql/relay-log #relay-log中继日志

relay-log-index=/var/log/mysql/relay-log.info #relay-log中继日志

relay-log-info-file=/var/log/mysql/relay-log.info

relay\_log\_purge=0 #不删除日志，MHA用得上

#以下是主从半同步复制的配置（semi sync master/slave）

plugin-load=rpl\_semi\_sync\_master=semisync\_master.so #加载主半同步复制模块

plugin-load=rpl\_semi\_sync\_slave=semisync\_slave.so #加载从半同步复制模块

rpl\_semi\_sync\_slave\_enabled=1 #启用从半同步复制模式

rpl\_semi\_sync\_master\_enabled=1 #启用主半同步复制模式

rpl\_semi\_sync\_master\_timeout=3000

## 数据类型

数值型:

整数型:tinyint(微小整数) smallint(小整数) mediumint(中整数) int(大整数) bigint(极大整数)

mysql> create table t3(age tinyint);

mysql> create table t3(age tinyint unsigned); #无符号范围

浮点型:float(单精度浮点数) double(双精度浮点数)

mysql> create table t6(pay float(7,2)); #总宽度7,小数位数2

字符型: 定长型char() 变长型varchar 大文本类型text/blob

mysql> create table t7 (name char(20),homedir char(15),mail varchar(30));

枚举型: 多选一enum() 多选多set()

mysql> create table t10(name char(10),

-> sex enum("boy","girl"),

-> likes set("hgn","game","sleep","eat")

-> );

**日 期:**

**日期时间：datetime**　占用４个字节

范围：1000-01-01 00:00:00.000000～9999-12-31 23:59:59.9999

**日期时间：timestamp** 占用４个字节

范围：1970-01-01 00:00:00.000000~2038-01-19 03:14:07.9999

Datetime和timestamp存储格式：yyyymmddHHMMSS，201901012018

**日　　期：date**　占用４个字节　范围：0001-01-01~9999-12-31　格式:yyyymmdd

**年　　份：year**  占用１个字节　范围：1901－2155 格式:yyyy

**时　　间：time**　占用３个字节 格式：HH:MM:SS

mysql> create table t8 ( name char(15),

-> age tinyint unsigned,

-> birthday date,

-> upclass time,

-> party datetime,

-> cs year );

## 日期函数:

mysql> select now(); #now()查看当前的日期和时间

mysql> SELECT sysdate(); #sysdate()查看系统日期和时间

mysql> SELECT curdate(); #curdate()获得当前的日期，不含时间

mysql> SELECT curtime(); #curtime()获得当前的时间，不含日期

mysql> SELECT year(now()),month(now()),day(now()); #分别获取当前日期时间中的年份、月份、日

+-------------+--------------+------------+

| year(now()) | month(now()) | day(now()) |

+-------------+--------------+------------+

| 2017 | 4 | 2 |

+-------------+--------------+------------+

mysql> SELECT month(sysdate()),day(sysdate()); #获取系统日期时间中的月份、日

+------------------+----------------+

| month(sysdate()) | day(sysdate()) |

+------------------+----------------+

| 4 | 2 |

+------------------+----------------+

mysql> SELECT time(sysdate());获取系统日期时间中的时刻

+-----------------+

| time(sysdate()) |

+-----------------+

| 04:06:08 |

+-----------------+

mysql> insert into db1.t8 values(

-> "zdd2",28,curdate(),curtime(),now(),year(now())　); #可根据日期函数插入实时数据

Query OK, 1 row affected (0.03 sec)

mysql> select \* from t8;

+------+------+------------+----------+---------------------+------+

| name | age | birthday | upclass | party | cs |

+------+------+------------+----------+---------------------+------+

| zdd | 17 | 2019-11-23 | 08:30:00 | 2019-02-14 20:30:00 | 1991 |

| zdd2 | 28 | 2019-02-13 | 17:33:29 | 2019-02-13 17:33:29 | 2019 |

+------+------+------------+----------+---------------------+------+

2 rows in set (0.00 sec)

## 建表约束条件：（表结构）

mysql> desc t3;

+-------+---------------------+------+-----+---------+-------+

| Field | Type | Null | Key | Default | Extra |

+-------+---------------------+------+-----+---------+-------+

| age | tinyint(3) unsigned | YES | | NULL | |

+-------+---------------------+------+-----+---------+-------+

Null 是否允许为空值(默认), not null不允许为空 Default:字段的默认值 Extra 额外设置

数据中”” “null” 都不是空值,null才是空(没双引号)

mysql> create table t1( #在当前库中创建t1表

-> name char(10) not null, #name 不为空

-> age tinyint unsigned not null default 18, #age不为空，默认值18

-> likes set("eat","game","film") default "eat" ); #likes枚举型set(多选多),默认值eat

## 修改表结构(alter)

**添加表字段(add)基本用法:**

alter table 表名 add 字段名 类型 约束条件; #默认在最后添加

alter table 表名 add 字段名 类型 约束条件 after 字段名a; #字段a后面添加

alter table 表名 add 字段名 类型 约束条件 first; #表最前面添加

mysql> alter table t1 add class char(7) default "nsd1811" first;

mysql> alter table t1 add email varchar(30),add qq varchar(11); #同时添加两个字段

**修改字段类型（modify）基本用法：**

Alter table 表名 modify 字段名 类型（宽度） 约束条件;

Alter table 表名 modify 字段名 类型(宽度) 约束条件 after 字段a; #将字段放到字段a后

Alter table 表名modify 字段名 类型（宽度） 约束条件 first;

1. 修改哪一部分就写新的，不修改就原样复制（否则变成默认值），
2. 如果该字段已有数据，修改的约束条件不能与原数据冲突

mysql> alter table t1 modify sex enum('man','woman','no') not null default "no";

mysql> alter table t1 modify age tinyint(3) unsigned not null default 18 after name;

**修改字段名（change）基本用法：**

Alter table 表名 change 原字段名 新字段名 类型(宽度) 约束条件;

mysql> alter table t1 change email mail varchar(30); #将email 改成mail

**删除字段名（drop）基本用法：**

Alter table 表名 drop 字段名1；

Alter table 表名 drop 字段名1，drop 字段名2；

mysql> alter table t1 drop mail , drop qq;

**修改表名（rename）基本用法：**

Alter table 表名 rename 新表名;

mysql> alter table t1 rename stuinfo;

**清空表数据,结构不变(delete)**

mysql> delete from stuinfo;

## Mysql 键值(索引,主键,外键)

键值类型：

* Index普通索引:key标志是MUL,
* Unique唯一索引: 这种索引和前面的“普通索引”基本相同，但有一个区别：索引列的所有值都只能出现一次，即必须唯一
* Fulltext:全文索引
* Foreign key:外键
* Primary key:主键,key标志是PRI,

**Index:普通索引(MUL)**

Primary key使用说明:

1. 一个表中可以有多个index 字段
2. 字段值允许有重复,且可以赋null值
3. 经常把做查询条件的字段设置为index字段
4. Index字段的key标志是MUL

**索引创建**

creater index 索引名on 表名(字段名);

mysql> create index age on t2(age); #方法一：在已有表中创建

mysql> alter table tablename add index [索引的名字] (列的列表); #方法二：修改表的方式

mysql>create table tablename ( 字段1，字段2, INDEX [索引的名字] (字段名) ); #方法三：建表时创建

mysql>drop index 索引名(键值类型) on 表名; #删除指定表中索引

**Primary key :主键(PRI)**

Primary key使用说明:

1. 对应的字段值不允许有重复,且不允许赋null值
2. 如果有多个字段都作为primary key 称为复合主键,必须一起创建
3. 主键字段的key标志是PRI
4. 通常与Auto\_Increment(自增属性)连用
5. 经常把表中是唯一标识记录的字段设置为主键字段:如[记录编号字段]

**建表时候创建主键:**第1种:在末尾添加

mysql> create table t3(

-> stu\_num char(9),

-> age int ,

-> primary key (stu\_num) );

**建表时候创建主键:**第2种在字段后面 输入关键字

mysql> create table t4(

-> name char(9) primary key,

-> age int );

**在已有表里添加字段时添加主键1**

选择一个无重复/无空值的字段,下面操作是先清空stuinfo表数据,再添加stu\_num字段做为主键

mysql> alter table stuinfo add stu\_num char(9) primary key after class;

#以上类似添加表字段方法:alter table 表名 add 字段名 类型 约束条件 after 字段名a; #字段a后面添加

**在已有表指定字段为主键中添加主键2**

将gzt表中的gzt\_id字段设置为主键

mysql> alter table gzt add primary key(gzt\_id);

**删除主键**

alter table 表名 modify id int not null #如果有自增属性(auto\_increment),必须先去掉

alter table 表名 drop primarey key;

主键删除后,字段可以有重复数据了,但是空值不行,需要修改

**创建复合主键(必须同时添加):**

复合主键作用:插入多条数据,只要复合主键的值没有同时重复,即可插入成功,

mysql> create table t5(

-> name char(10),

-> class char(7),

-> status enum("yes","no"),

-> primary key (name,class) ); #name和class都是主键

Query OK, 0 rows affected (0.19 sec)

mysql> insert into t5 values("bob","1811","no"); #插入name数据和class数据

Query OK, 1 row affected (0.04 sec)

mysql> insert into t5 values("bob","1811","yes"); #主键值同时相同,插入不成功

ERROR 1062 (23000): Duplicate entry 'bob-1811' for key 'PRIMARY'

mysql> insert into t5 values("bob","1812","yes"); #name主键相同,class主键不同

Query OK, 1 row affected (0.05 sec)

mysql> insert into t5 values("dc","1812","no"); #name主键不同,class相同

Query OK, 1 row affected (0.03 sec)

**与auto\_increment(自增长)连用**

Auto\_increment:作用是让字段的值自增长

* 字段类型必须是数值类型,且必须是主键
* 如果定义自增长的字段的值,就不自增长添加,按照你定义的值添加,添加的不能重复
* 自增长的值都是按照上一条数据的值自增长1,不管数据有没有被删除

mysql> create table t6(

-> id int primary key auto\_increment, #id为主键,且有自增长属性

-> name char(10),

-> sex enum("man","woman") );

**删除有auto\_increment(自增长)的主键**

alter table 表名 modify id int not null #如果有自增属性,必须先去掉

mysql> alter table t6 drop primary key; #无法删除主键

ERROR 1075 (42000): Incorrect table definition; there can be only one auto column and it must be defined as a key

mysql> alter table t6 modify id int not null; #去掉自增长

Query OK, 5 rows affected (0.52 sec)

Records: 5 Duplicates: 0 Warnings: 0

mysql> alter table t6 drop primary key; #删除主键成功

Query OK, 5 rows affected (0.49 sec)

Records: 5 Duplicates: 0 Warnings: 0

**Foreign key 外键**

**什么是外键:**

外键作用:让当前表字段的值在另一个表中字段值的范围内选择

A表中的name字段设置了外键,参考的是B表中的user字段

A表中的name字段数据内容必须在B表中的user字段内容选择

B表中的user字段必须是主键

**使用外键的条件:**

1. 表的存储引擎必须是innodb,只有innodb支持外键
2. 字段类型要一致
3. 被参照字段必须要是索引类型的一种(primary key)

外键规则

**创建外键(建表时创建)**

Create table 表A名(

字段名1 类型 ,

字段名2 类型 ,

Foreign key (表A的字段名1) references 表B名(字段名)

On update cascade #表A与表B名(字段名)数据同步更新

On delete cascade #表A与表B名(字段名)数据同步删除

)engine=innodb;

创建工资表gzt 创建外键,定义gzt\_id为外键,参考ygt表的(ygt\_id)字段,同步更新与删除

mysql> create table gzt(

-> gzt\_id int,

-> pay float(7,2),

-> foreign key(gzt\_id) references ygt(ygt\_id) on update cascade on delete cascade

-> )engine=innodb;

**查看表创建详情**

mysql> show create table gzt;

........此处省略

| gzt | CREATE TABLE `gzt` (

`gzt\_id` int(11) DEFAULT NULL,

`pay` float(7,2) DEFAULT NULL,

KEY `gzt\_id` (`gzt\_id`),

CONSTRAINT `gzt\_ibfk\_1` FOREIGN KEY (`gzt\_id`) REFERENCES `ygt` (`ygt\_id`) ON DELETE CASCADE ON UPDATE CASCADE

) ENGINE=InnoDB DEFAULT CHARSET=latin1 |

+-------+------------------------------------------------------------------------------------------------------------------------------.............此处省略

**解释:**CONSTRAINT `gzt\_ibfk\_1` 为外键名称

FOREIGN KEY (`gzt\_id`) 外键创建在(`gzt\_id`)字段上

REFERENCES `ygt` (`ygt\_id`) 参考为ygt表的ygt\_id字段

ON DELETE CASCADE 跟参考同步更新 ON UPDATE CASCADE 跟参考同步删除

ENGINE=InnoDB 存储引擎为InnoDB

**删除外键**

Alter table 表名 drop foreign key 约束名(外键名)

先查看表创建详情,再执行删除

mysql> alter table gzt drop foreign key gzt\_ibfk\_1; #查询出来的约束名(如上)再删除

**已有表中添加外键**

要保证外键字段的数据值与参考表中参考字段的值相匹配

mysql> alter table gzt add foreign key(gzt\_id) references ygt(ygt\_id) on update cascade on delete cascade;

**注意事项**

* 当表中有外键时:被参考的表和被参考的字段都不能删除
* 被参考的表的所有记录被删除,外键表的所有记录也将被删除

## MYSQL结构

**Management....(管理工具):**提供各种mysql 命令

**Connection pool(连接池):**检查有无空闲线程、有无硬件支持线程、登录用户名和密码是否正确.三者满足,建立链接

**Sql interface (sql 接口):**接受执行命令

**Parser解析器(分析器):**分析命令格式是否正确.给错误的命令报错

**Optimizer(优化器):**当执行命令没有错误.调用优化器,进行命令优化,以最高效率处理命令

**Caches & buffers(查询缓存):**提供存储空间,存储着曾经查找过的数据,从数据库服务器物理内存划分出来的存储空间给MY.提高查询速度:先检索查询缓存,然后在查询硬盘上的数据

**File system(文件系统):**存储数据,服务器磁盘硬件

**Pluggable storage englnes(存储引擎):**当调用数据库文件系统中的数据,就会调用表使用的存储引擎处理数据.

## 存储引擎

**常用存储引擎的特点(只用innodb支持外键)**

**Mysiam:**支持表级锁以及独享表空间.不支持事务/不支持事务回滚,节省资源

对应有三个表文件

表名.frm:存的是表结构 desc 表名

表名.MYI:存的是索引 index(name)

表名.MYD:存的是表数据 select \* from 表名

**Innodb :**支持行级锁定,支持事务/事务回滚/外键

对应二个表文件

表名.frm:存表结构 desc 表名

表名.ibd:存既存索引,又存数据

**设置数据库服务默认使用的存储引擎**

修改/etc/my.cnf 主配置文件如下:

[mysqld]

default-storage-engine=存储引擎名 #新加

mysql> show variables like '%storage\_engine%'; #查看数据库默认使用的存储引擎

mysql>show engines\G; #查看本版本数据库软件支持的所有存储引擎信息,

mysql> show create table t1; #查看已有表的存储引擎”ENGINE=MyISAM ”

mysql> create table t2(id int(4),name varchar(8))engine=innodb; #建表时指定存储引擎

mysql> alter table t2 engine=myisam; #修改已有表的存储引擎

**锁介绍(锁粒度):**

**为什么要加锁:**解决并发访问冲突问题,如多个客户端访问A表,有查询,插入,修改updete等操作,同时修改同一内容,造成写入冲突,第一个人访问之后,就将锁定内容.

**表级锁:**只要对表做访问,整张表被锁上

**行级锁:**只给访问的每一行记录分别加锁,

**页级锁:**对整个页面进行加锁,页面(MYSQL管理数据的基本存储单位)

**锁类型:** 读锁(共享锁):支持并发读

写锁(排他锁/互斥锁):独占锁,上锁期间,其他线程不能读表或写表

mysql> show status like '%table\_lock%'; #查看当前表锁状态信息

**事务介绍**

**事务:**对数据的一次访问,就是一次事务,只有innodb存储支持

**事务回滚介绍:**在事务执行过程中,任意一步执行失败,将回复之前所有操作,还原操作前

**如何回滚:**有事务日志文件.启动mysql服务,自动创建.记录对innodb存储引擎的表执行过 的操作,事务日志文件所在:/var/lib/mysql/ 目录下以下文件:

ib\_logfile0 ib\_logfile1 idbdata1

mysql> show variables like '%autocommit%'; #查看自动提交开关状态

mysql> set autocommit=off; #关闭自动提交

#事务的持久性开启了自动提交:命令输入插入/修改等命令,一回车,就将数据修改永久生效

mysql> set autocommit=off; #终端1关闭自动提交

mysql> delete from t6; #终端1删除t6表

mysql> select \* from t6; #终端1查看t6表,已无数据,终端1未提交

[root@sql51 ~]# mysql -uroot -p123456 #终端2登录mysql

mysql> use db3 #终端2进入db3库

mysql> select \* from t6; #终端2查看t6表还能看到数据

mysql> rollback; #终端1提交前进行回滚

mysql> select \* from t6; #终端1又能看到之前的数据

mysql> insert into t6 values(666); #终端1修改

mysql> select \* from t6; #终端1查看能看到新增数据

mysql> select \* from t6; #终端2上查看还是看不到,终端1未提交

mysql> commit ; #终端1提交修改

mysql> select \* from t6; #终端2查看就能看到新增的数据了

**实际环境中如何决定一个表使用哪种存储引擎**

**查询表记录**:查询(select)操作多的表适合选用**myisam**存储引擎,节省系统资源

myisam**表级锁:**只要对表做访问,整张表被锁上

读锁(共享锁):支持并发读

**插入表记录**:写入(insert update delete)操作多的表适合选用**innodb**存储引擎,并发访问量大

innodb**行级锁**:只给访问的每一行记录分别加锁,

写锁(排他锁/互斥锁):独占锁,上锁期间,其他线程不能读表或写表

## 数据导入导出:

**数据导入**

在MySQL 5.7.6版本之后，导入文件只能在secure\_file\_priv指定的文件夹下。

执行show variables like '%secure%'命令显示默认检索文件目录：

mysql> show variables like '%secure%';

[root@sql51 ~]# vim /etc/my.cnf

[mysqld]

secure\_file\_priv="/mydatadir" #可修改默认检索目录,必须给新的目录设置数主为mysql用户

[root@sql51 ~]# mkdir /mydatadir

[root@sql51 ~]# chown mysql /mydatadir #赋予权限

#导入之前必须先建立对应的表结构.(如下)

mysql> create table user(

-> name char(35),

-> password char(1),

-> uid int,

-> gid int,

-> comment char(150),

-> homedir char(100),

-> shell varchar(100),

-> index(name) );

mysql> load data infile "/mydatadir/passwd" into table user fields terminated by ":" lines terminated by "\n"; #导入passwd文件数据,以”:”号为分隔符,指定换行符为”\n”,

mysql> alter table user add id int primary key auto\_increment first; #插入id

**数据导出**

**导出作用:把表记录导出到文件**

命令格式:sql查询 into outfile “目录名/文件名”

Fields terminated by “分隔符” #分隔符,可单独指定

lines terminated by “\n” #换行符,可单独指定

**注意事项:导出的内容由sql查询语句决定**

导出的文件路径必须是 secure\_file\_priv(默认搜索路径) 指定的目录里面

导出的是表中的记录,不包括字段名

导出的/目录/文件名 必须唯一,不能有重复

禁用SElinux

mysql> select \* from user into outfile "/mydatadir/a1.txt"; #数据全部导出,

mysql> select \* from user where id <= 5 into outfile "/mydatadir/a2.txt"; #导出部分数据

mysql> select name,uid,shell from user where id <= 5 into outfile "/mydatadir/a3.txt";

mysql> select name,uid,shell from user where id <= 5 into outfile "/mydatadir/a4.txt" fields terminated by "#" lines terminated by "!!!"; #导出数据,并指定分隔符,默认空格分隔和”\n”换行

## 复制表

**复制表作用:备份表与快速建表**

**1.1:将源表xxx复制为新表yyy**

命令格式:crate tabe yyy select \* from xxx

例如:将db3库的user表,复制到db4库userbak表

mysql> create table db4.userbak select \* from db3.user;

**1.2:将指定查询结果复制为新表zzz**

mysql> create table user3 select name,uid,shell from db3.user order by uid desc limit 5;

mysql> select \* from user3;

**1.3:复制源表xxx的结构到新表vvv**

例如:将db3库的user表结构,复制到db4库的user1表,主键/索引等无法复制过来

mysql> create table user1 select \* from db3.user where 1=2;

mysql> select \* from user1;

Empty set (0.00 sec)

**1.4:复制部分字段到新表**

mysql> create table user4 select name,uid,shell from db3.user where 1=2;

**1.5:将源表vvv 的名称改为www**

Alter table vvv rename to www;

## 用户授权与权限撤销

**授权命令格式**

grant 权限列表 on 数据库名 to 用户名@”客户端地址” identified by “密码”

grant 权限列表 on 数据库名.表名 to 用户名@”客户端地址” identified by “密码”

grant 权限列表 on 数据库名 to 用户名@”客户端地址” identified by “密码”With grant option;

#给这个用户授权,这个用户也有授权权限

**权限列表**

all : #赋予所有权限 Select,update,insert #赋予权限操作(查询/修改/插入)

Select,update (字段1,...,字段N) #赋予部分字段的查询与修改

**客户端地址**

%:匹配所有主机 192.168.1.% :匹配指定的一个网段

192.168.1.1:匹配指定IP地址的单个主机 %.tarena.com:匹配一个DNS区域

svr1.tarena.con:匹配指定域名的单个主机(需要DNS服务器解析)

**授权示例**

新建用户mydba,对所有库所有表(语句表示\*.\*)有完全权限

允许从任何地址访问,密码设为”123456” 允许该用户为其他用户授权

mysql> grant all on \*.\* to mydba@"%" identified by "123456" with grant option;

Query OK, 0 rows affected, 1 warning (0.00 sec)

**查看授权**

mysql> select \* from mysql.user where user="mydba" \G; #查看指定数据库系统用户访问的权限

mysql> select host,user from mysql.user; #查看数据库系统用户访问的权限

mysql> show grants for mydba@"%"; #查看mydba用户权限详情

mysql> select user(); #查看登录的本机IP

mysql> select @@hostname; #查看已连接的数据库服务器的主机名

mysql> show grants; #查看当前登录本机的用户权限

mysql> grant select,insert on db3.user to admin@"localhost" identified by "123456";

#创建admin用户,授予只对db3.user ,有select与insert权限,只限于在数据库服务器本机上登录.登录密码是”123456”

**撤销权限命令格式**

revoke 权限列表 on 数据库名 from 用户名@”客户端地址”;

mysql> revoke grant option on \*.\* from mydba@"%"; #撤销授权权限

mysql> revoke delete,update on \*.\* from mydba@"%"; #撤销删除与修改权限

mysql> revoke select on db7.\* from yaya; #删除yaya用户,在db7库里面对所有库的查询权限

mysql> revoke all on \*.\* from mydba@"%"; #撤销所有权限

**删除授权用户**

Drop user 用户名@”客户端地址”;

mysql> drop user mydba@"%"; #删除mydba用户

mysql> select host,user from mysql.user; #查看系统库的用户表

+-----------+-----------+

| host | user |

+-----------+-----------+

| localhost | admin |

| localhost | mysql.sys |

| localhost | root |

+-----------+-----------+

3 rows in set (0.00 sec)

mysql> drop user admin@"localhost"; #注意:此处删除admin时.后面要填”localhost”

**授权库(mysql库)** ：记录授权信息的表

User表: 存储授权用户的访问权限

db表: 存储授权用户对数据库的访问权限

tables\_priv表: 存储授权用户对表的访问权限

Columns\_priv表: 存储授权用户对表字段的访问权限

## 密码恢复以及设置

修改数据库管理员root本机登录密码,只有操作系统管理员root能修改

只有操作系统管理员root才有权限修改配置文件,和修改user表记录

**恢复步骤：**

1.停止MySQL服务程序

2.跳过授权表启动MySQL服务程序,将skip-grant-tables 写入/etc/my.cnf

3.重设root密码(更新user表记录)

4.以正常方式重启MySQL服务程序

[root@sql51 ~]# systemctl stop mysqld #停用数据库

[root@sql51 ~]# vim /etc/my.cnf #配置文件

[mysqld]

#secure\_file\_priv="/mydatadir" #注释掉

#default-storage-engine=myisam #注释掉

#validate\_password\_length=6 #注释掉

#validate\_password\_policy=0 #注释掉

skip-grant-tables #新增---此行将跳过授权表启动MySQL数据库

[root@sql51 ~]# systemctl start mysqld

[root@sql51 ~]# mysql

mysql> #不需要密码直接登录数据库成功

mysql> use mysql

mysql> show tables;

mysql> desc user;

mysql> select host,user,authentication\_string from user;

+-----------+-----------+-------------------------------------------+

| host | user | authentication\_string |

+-----------+-----------+-------------------------------------------+

| localhost | root | \*6BB4837EB74329105EE4568DDA7DC67ED2CA2AD9 |

| localhost | mysql.sys | \*THISISNOTAVALIDPASSWORDTHATCANBEUSEDHERE |

+-----------+-----------+-------------------------------------------+

2 rows in set (0.00 sec)

mysql> update mysql.user set authentication\_string=password("654321") #修改密码

-> where

-> host="localhost" and user="root";

mysql> flush privileges; #刷新mysql 新设置用户或更改密码后需用flush privileges刷新MySQL的系统权限相关表，否则会出现拒绝访问，还有一种方法，就是重新启动mysql服务器，来使新设置生效。

## MySQL管理工具phpMyAdmi

## 数据库备份与恢复

**Mysqldump工具完全备份：**

1、mysqldump -u用户 -p密码 数据库名 >备份文件名.sql #备份单个数据库

2、mysqldump -u用户 -p密码 all >备份文件名.sql #备份所有的数据库

[root@dbsvr1 ~]# mysqldump -u root -p --all-databases > /root/alldb.sql

3、mysqldump -u用户 -p密码 数据库名 表名 >备份文件名.sql #备份单张表，数据库名表名不带点、

[root@sql51 mysqlbak]# mysqldump -uroot -p654321 db3 user >/mysqlbak/db3\_user.sql

4、mysqldump -u用户 -p密码 -B 数据库名1 数据库名2 表名 >备份文件名 #备份多个数据库，加-B

[root@sql51 mysqlbak]# mysqldump -uroot -p654321 -B db2 db4 >/mysqlbak/twodb.sql

**Mysqldump工具完全恢复：**恢复数据表需要加库名

恢复数据库之前,要先创建新库,备份数据库时用-B 备份的多个数据库的,恢复时可以省略数据库名

1、mysql -u用户 -p密码 < 备份文件名 #用户必须有登录备份权限

[root@sql53 ~]# mysql -uroot -p654321 </root/twodb.sql #备份时用-B备份的，恢复时省略了数据库名

2、mysql -u用户 -p密码 库名 <备份文件名 #用户必须有登录备份权限,库名可以为新库名

[root@sql53 ~]# mysql -uroot -p654321 db3 < /root/db3\_user.sql #将db3\_user.sql恢复到db3库中

**binlog日志**

binlog日志三大格式:

1. **row:**
2. **statement:**
3. **mixed:**

[root@sql51 ~]# mkdir /logdir

[root@sql51 ~]# chown mysql /logdir

[root@sql51 ~]# vim /etc/my.cnf

[mysqld]

server\_id=51

log\_bin=/logdir/zdd #日志文件将以zdd开头

binlog-format=mixed #binglog日志格式

log\_bin=/var/log/mysql/mysql-bin

relay-log=/var/log/mysql/relay-log #relay-log中继日志

relay-log-index=/var/log/mysql/relay-log.info #relay-log中继日志

relay-log-info-file=/var/log/mysql/relay-log.info

relay\_log\_purge=0 #不删除日志，MHA用得上

log\_slave\_updates #开启级联复制功能,既为主又为从的服务器

binlog\_do\_db=name #设置master对哪些库记日志(允许同步的库)

binlog\_ignore\_db=name #设置master对哪些库不记日志(不允许同步的库)

replicate\_do\_db=db8,db9 #主从只同步指定的数据库

replicate\_ignore\_db=db7,db6 #主从不同步指定的数据库

[root@sql51 ~]# systemctl restart mysqld

root@sql51 ~]# ls /logdir/ #日志文件产生在这里了

zdd.000001 zdd.index

mysql> show master status; #查看binlog日志

mysql> flush logs; #刷新日志,将会重新生成日志从0001开始

[root@sql51 ~]# systemctl restart mysqld #重启数据库服务,将会生成新的日志,接着原有编号

[root@sql51 ~]# mysql -uroot -p654321 -e "flush logs" #未登录,直接执行mysql命令方式刷新日志

[root@sql51 ~]# mysqldump -uroot -p654321 --flush-logs db1 > /mysqlbak/db1logtest.sql #导入时

mysql> purge master logs to "zdd.000005"; #删除zdd.000005之前的日志,不包括0005的mysql> reset master; #重置主服务.也会重新生成日志.从0001开始

**增量备份:**

[root@sql51 ~]# mysqlbinlog /logdir/plj.000002 #查看日志文件内容

[root@sql51 ~]# mysqlbinlog /logdir/plj.000001 | grep insert #查看日志文件内容并筛选

[root@sql52 mysqld]# mysqlbinlog /root/plj.000001 | mysql -uroot -p654321 #恢复日志内所有语句

[root@sql52 ~]# mysqlbinlog --start-position=560 --stop-position=924 /root/plj.000001 |mysql -uroot -p654321 #恢复指定部分范围内的数据

\*\*选择范围必须包含日志文件的commit \*\*不指定开头,默认从最前面开始 \*\*不指定结束,默认执行到最后

**差异备份:**

**XtraBackup 工具备份**

[root@sql51 ~]# yum -y install libev-4.15-1.el6.rf.x86\_64.rpm

[root@sql51 ~]# yum install -y percona-xtrabackup-24-2.4.7-1.el7.x86\_64.rpm

3.4.1数据完全备份与恢复

Innobackupex恢复数据要求数据库清空mysql目录,因此那怕你只需要备份多个库中其中一个库,也要将数据库三个系统库(mysql sys performance\_schema)一起备份

3.4.1.1完全备份:

案例:将所有库完全备份到/backup(目录不需事先创建,备份程序自动创建)

格式:Innobackupex --user root --password 123456 /allbak -no-timestamp

Innobackupex 选项

--host 主机名

--user 用户名

--port 端口号(默认3306可不用写)

--databases 数据库名(不写将备份所有数据库)

--no-timestamp 不用时间戳,不用日期命名备份文件存储的子目录名

Innobackupex 备份之后有两种文件1是数据文件,2是备份信息文件

1数据文件,用来恢复数据

2备份配置文件:有备份信息,用来做增量备份或者差异备份等的参考

## 主从

主库配置:

[mysqld]

bind-address=0.0.0.0 #监听地址，有些版本差异，添上不会出问题

server-id= {{id.stdout}} #用ansible批量传送，传参方式获取

binlog-format=mixed #binglog日志格式

log\_bin=/var/log/mysql/mysql-bin

relay-log=/var/log/mysql/relay-log #relay-log中继日志

relay-log-index=/var/log/mysql/relay-log.info #relay-log中继日志

relay-log-info-file=/var/log/mysql/relay-log.info

relay\_log\_purge=0 #不删除日志，MHA用得上

#以下是主从半同步复制的配置（semi sync master/slave）

plugin-load=rpl\_semi\_sync\_master=semisync\_master.so #加载主半同步复制模块

plugin-load=rpl\_semi\_sync\_slave=semisync\_slave.so #加载从半同步复制模块

rpl\_semi\_sync\_slave\_enabled=1 #启用从半同步复制模式

rpl\_semi\_sync\_master\_enabled=1 #启用主半同步复制模式

rpl\_semi\_sync\_master\_timeout=3000

log\_slave\_updates #开启级联复制功能,既为主又为从的服务器

binlog\_do\_db=name #设置master对哪些库记日志(允许同步的库)

binlog\_ignore\_db=name #设置master对哪些库不记日志(不允许同步的库)

replicate\_do\_db=db8,db9 #主从只同步指定的数据库

replicate\_ignore\_db=db7,db6 #主从不同步指定的数据库

mysql> show master status; #查看日志信息

mysql> grant replication slave on \*.\* to repluser@"%" identified by "123456";

从库配置:

[root@sql52 ~]# vim /etc/my.cnf

[mysqld]

server\_id=52

[root@sql52 ~]# systemctl restart mysqld

mysql> change master to

master\_host="192.168.4.51", #主服务器IP地址

master\_user="repluser", #主服务器授权的用户

master\_password="123456", #授权用户的密码

master\_log\_file="master51.000001", #日志文件

master\_log\_pos=441; #偏移位置

mysql> start slave;

mysql> show slave status\G;

## MySQL数据读写分离(maxscale)

1配主从并且创建相关权限的用户

mysql> grant replication slave,replication client on \*.\* to maxmon@"%" #maxscale用来监控服务

mysql> grant select on \*.\* to scaluser@"%" identified by "123456"; #maxscale用来检查查询用户

mysql> grant select,insert on \*.\* to jim@"%" identified by "123456"; #插入数据用户

2安装maxscale-2.1.2-1.rhel.7.x86\_64.rpm(单独安装包)

3配置:vim /etc/maxscale.cnf

定义数据库server主机,定义监视的数据库,定义只读,定义读写分离,手动添加管理端口

4启动服务:[root@sql57 ~]# maxscale -f /etc/maxscale.cnf

5停止服务:[root@sql57 ~]# killall -9 maxscale

6测试数据准备:在主库上插入若干数据,从库自动同步, 在从库上单独插入若干数据,主库不同步,用来测试读

7客户端远程:[root@client-50 ~]# mysql -h(maxscale服务主机地址) -P4006 -ujim -p123456

8测试:

1,在主库上查询数据,看不到从库的数据,在客户端远程登录后能查询到从库数据,说明读的是从库的数据

2,在客户端远程登录后插入数据,主库有数据,且从库也有数据

以上两点满足,则读写分离配置成功.

## MHA集群