一、二进制安装MySQL

二、MySQL完整 / 增量备份与恢复

MySQL完全备份与恢复

1.数据备份的重要性

造成数据丢失的原因如下:

2、数据库备份的类型

1.从物理与逻辑的角度

物理备份

逻辑备份

2.从数据库的备份策略角度

1.完全备份

2.差异备份

3.增量备份

3.常见的备份方法

3.MySQL完全备份操作

1、直接打包数据库文件夹

2、使用专用备份工具mysqldump

(1) 对单个库进行完全备份

(2)对多个库进行完全备份

(3)对所有库进行完全备份

(4)对表进行完全备份

(5)对表结构的备份---面试

4.使用mysqldump备份后，恢复数据库

1、source命令

2、mysql 命令

5.MySQL备份思路

6.MySQL完全备份案例

需求描述:

ps:这里涉及一个MySQL存储中文的问题

这里出现了字符集错误的问题：

在创建表的时候指定字符集：

模拟恢复数据：

定期备份数据的小脚本：

7.MySQL数据库备份脚本

实验环境:

实验要求:

对mysql-server的auth库和client库实现异地备份

撰写数据恢复的脚本：--有点复杂需要吸收

恢复测试：

8.MySQL增量备份与恢复

1.MySQL增量备份概念

2.增量备份的特点:

3.MySQL二进制日志对备份的意义:

4. 开启MySQL的二进制日志功能

方法一：

MySQL的配置文件的[mysgld]项中加入log-bin=文件存放路径/文件前缀，

方法二:

这个方法是临时的，一次性的，建议用上面的写入文件里，是永久的

2.MySQL增量恢复

1.应用场景

2.增量恢复的方法

1、一般的恢复

2、基于时间点的恢复:

3、基于位置的恢复:

9.制定企业备份策略的思路---作为了解

10.MySQL企业备份案例

需求描述:

先进行一次完全的备份

查看日志，分析日志

模拟误删除:

恢复数据:

模拟删除:

基于时间点恢复：找BEGIN　COMMITE之间是一个完整的语句

模拟删除:

基于位置的恢复数据:

模拟删除:

模拟删除:

实现完整/增量备份的脚本:

11.mysql日志种类

1.错误日志(error log)

一、二进制安装MySQL

```
[root@localhost ~]# rpm  -q libaio
[root@localhost ~]# useradd -s /sbin/nologin mysql
[root@localhost ~]# tar xf mysql-5.7.24-linux-glibc2.12-x86_64.tar.gz -C
/usr/src/
```

由于解压的名字比较长，可以改名

```
[root@localhost ~]# cd /usr/local/
[root@localhost local]# mv mysql-5.7.24-linux-glibc2.12-x86_64/ mysql
[root@localhost local]# chown -R mysql:mysql /usr/local/mysql/
```

修改配置文件：

```
[root@localhost ~]# vim /etc/my.cnf
datadir=/usr/local/mysql/data
socket=/tmp/mysql.sock
log-error=/usr/local/mysql/data/mysql.log
pid-file=/usr/local/mysql/data/mysql.pid
[root@localhost ~]# cd /usr/local/mysql/bin/
```

初始化：

```
[root@localhost bin]# ./mysqld --datadir=/usr/local/mysql/data --
basedir=/usr/local/mysql/ --user=mysql --initialize
[root@localhost bin]# cp ../support-files/mysql.server /etc/init.d/mysqld
[root@localhost bin]# chmod +x /etc/init.d/mysqld
[root@localhost bin]# chkconfig --add mysqld
```

起服务：

```
[root@localhost bin]# systemctl start mysqld
[root@localhost bin]# netstat -lnpt
tcp6       0      0 :::3306                 :::*                    LISTEN
6558/mysqld
```

```
[root@localhost ~]# ln -s /usr/local/mysql/bin/* /usr/local/bin/
[root@localhost ~]# mysqladmin -uroot -p'%ofMZf((r2+H' password '123456'
[root@localhost ~]# mysql -uroot -p123456
```

二、MySQL完整/增量备份与恢复

MySQL完全备份与恢复

随着自动化办公与电子商务的不断扩展，企业对于信息系统的依赖性越来越重要，
而数据库在信息系统中担任着非常重要的角色。
尤其一些对数据库可靠性要求非常高的行业，例如银行，证券，电信等，
如果发生意外宣机或数据丢失，其损失是非常重要的。
为此数据库管理员必须针对具体的业务要求定制详细的数据库备份与灾难恢复的策略，
并通过模拟故障对每种可能的情况进行严格的测试。而保障数据的可靠性

1.数据备份的重要性

备份的主要目的是灾难恢复
备份还可以测试应用，回滚数据修改，查询历史数据，审计等
我们将从生产运维的角度了解备份恢复的分类与方法
在企业中数据的价值至关重要
数据保障了企业的业务的运行
因此数据的安全性及可靠性是运维的重中之重
何数据的丢失都有可能会对企业产生严重的后果

造成数据丢失的原因如下:

- 程序错误---软件bug，造成数据修改，导致数据库里面的数据出问题
- 人为错误---管理人员或其他人员执行错误的语句
- 数据泄露---12306数据泄露
- 运算失败---类似程序错误
- 磁盘故障---硬盘出问题，reid出问题，造成数据丢失
- 灾难(如火灾、地震)

尽可能做异地栽培---两地三中心
rsync，NFS
公司规模，数据重要性

2、数据库备份的类型

1.从物理与逻辑的角度

备份可以分为物理备份和逻辑备份

物理备份

对数据库操作系统的物理文件(如数据文件、日志文件等)的备份。

```
mysql> create database abc;
[root@localhost ~]# ls /usr/local/mysql/data/
abc
```

对整个目录的备份即物理备份

物理备份

1. 脱机备份(冷备份)
2. 联机备份(热备份)

这种类型的备份适用于出现问题时需要快速恢复的大型重要数据库

1.冷备份:是在关闭数据库的时候进行的

---用的不多，除非有多台机器（多主多从）

2. 热备份:数据库处于运行状态，这种备份方法依赖于数据库的日志文件（二进制文件）

3. 温备份:数据库锁定表格(不可写入但可读)的状态下进行的

-----12306晚间不提供卖票只提供查询票的信息

逻辑备份

对数据库逻辑组件(如表等数据库对象)的备份

表示信息

逻辑数据库结构(create database、create table等语句)

内容(insert 语句或分割文本文件)

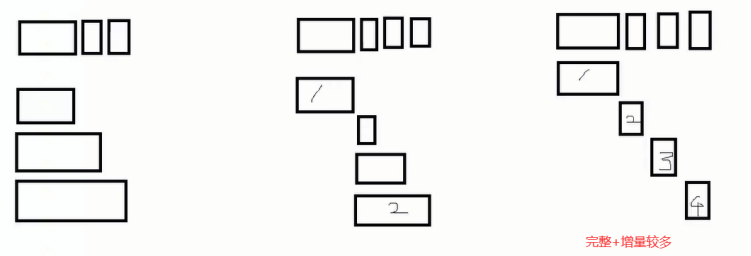
这种类型的备份适用于

可以编辑数据值或表结构较小的数据量

或者在不同机器体系结构上重新创建数据

2.从数据库的备份策略角度

备份可分为完全备份、差异备份和增量备份



1.完全备份

每次对数据进行完整的备份，

即对整个数据库的备份、数据库结构和文件结构的备份，

保存的是备份完成时刻的数据库状态，**是差异备份与增量备份的基础**

优点:备份与恢复操作简单方便

缺点:数据存在大量的重复;

占用大量的空间;

备份与恢复时间长

2.差异备份

备份那些自从上次完全备份之后被修改过的所有文件

备份的时间起点是从上次完整备份起，备份数据量会越来越大

恢复数据时，只需恢复上次的完全备份与最近的一次差异备份

3.增量备份

只有那些在上次完全备份或者增量备份后被修改的文件才会被备份

以上次完整备份或上次的增量备份的时间为时间点，仅备份这之间的数据变化

- 备份的数据量小
- 占用空间小
- 备份速度快

但恢复时，需要从上一次的完整备份起到最后一次增量备份依次恢复，

如中间某次的备份数据损坏，将导致数据的丢失

ps:

企业的每天的数据量特别大，且每天的增长量也大；使用完整+增量备份比较好

这种大公司的架构，一般来说其中有一个断开的可能性不大，他们备份的方式主要是biinlog

而对于数据量特别小的公司，使用完整备份，一共才几个G

3.常见的备份方法

MySQL数据库的备份可以采用很多种方式，如直接打包数据库文件(物理冷备份)

专用备份工具(mysgldump)， 二进制日志增量备份，第三方工具备份等

1)物理备份

物理冷备份时需要在数据库处于关闭状态下，能够较好的保证数据库的完整性。

物理冷备份：用于非核心业务，这类业务都允许中断

物理冷备份的特点：速度快，恢复时也是最为简单的，

通过直接打包数据库文件夹(/usr/local/mysql/data) 来实现备

2)专用备份工具mysqldump或mysglhotcopy

mysqldump和mysglhotcopy都可以做备份。

mysqldump是客户端常用逻辑备份程序，

能够产生一组被执行以再现原始数据库对象定义和表数据的SQL语句

(恢复就是将这些语句再执行一遍)

它可以转储一个到多个MySQL数据库

其进行备份或传输到远程SQL服务器

mysqldump更为通用，

因为它可以备份各种表

mysqlhotcopy仅适用于某些存储引擎

mysqlhotcopy是由Tim Bunce最初编写和贡献的Perl脚本

mysglhotcopy 仅用于备份MyISAM和ARCHIVE数据表

只能运行在Unix或Linux操作系统上

有些数据库也跑在windows上的

3)通过启用二进制(binary log，binlog) 日志进行增量备份

MySQL支持增量备份，进行增量备份时必须启用二进制日志。

二进制日志文件为用户提供复制。

对执行备份点后进行的数据库更改所需的信息进行备份。如果进行增量备份(包含上次完全备份或增量备份以来发生的数据修改)，需要刷新二进制日志

4)通过第三方工具备份

Percona XtraBackup是一个免费的MySQL热备份软件，

支持在线备份Innodb和XtraDB，也可以支持MySQL表备份，

不过MyISAM表的备份要在表锁的情况进行

Percona XtraBackup主要的工具: xtrabackup、 innobackupex、 xbstreame

---

xtrabackup:是一个编译 了的二进制文件，只能备份Innodb/Xtradb数据文件

innobackupex:

是一一个封装了xtrabackup的Perl脚本，

除了可以备份Innodb/Xtradb之外，还可以备份MyISAM

xbstream:是一个新组件，能够允许将文件格式转换成xbstream格式或从xbstream格式转到文件格式

---

xtrabackup工具可以单独使用，但推荐使用innobackupx来进行备份，

因为其本身已经包含了xtrabackup的所有功能

xtrabackup是基于Innodb的灾难恢复功能进行设计的，

备份工具复制Innodb的数据文件，

但是由于不锁表，这样复制出来的数据将不一致，

Innodb 维护了一个重要日志，包含Innodb数据的所有改动情况。

在xtrabackup备份Innodb的数据同时，

xtrabackup 还有另外一个线程用来监控重做日志，

一旦日志发生变化，就把发生变化的日志数据复制走

这样就可利用重做日志做灾难恢复了

思考类问题:面试

1.公司数据库的总数据量多大?

5个产品库，每个库，两三个G，有的七八个G，

所有库总共 25~30G

具体看公司的业务，日志里只存储文字

MySQL数据库管理软件，可以对外提供多个库，一个库对应一个产品，

数据量不大，5个库左右，平均每个库5个G左右

2.每天增长量多大?

50M左右

3.备份的策略?

完整备份

服务器的硬件比个人的更好，硬盘的转速更快，服务器还做了reid ,甚至有些公司的数据库是跑在固态上的，它的运营速度更快，备份速度更快，个人电脑拷贝数据要5个G，而用公司的可能半分钟都不到

4.备份数据量?

备份一个库5个G，备份完压缩一下，也就两三个G

5.备份的时长?

一分钟以内

3.MySQL完全备份操作

MySQL数据库的完全备份可以采用多种方式，

物理冷备份一般用tar命令直接打包数据库文件夹(数据目录)，而在**备份前需要先停库**

1、直接打包数据库文件夹

源码包的位置/usr/local/mysgl/data/

rpm 包的位置/var/ib/mariadb/

示例:

```
[root@localhost ~]# mysql -uroot -p123456
mysql> show databases;
mysql> drop database abc;
mysql> show databases;
mysql> create database auth;
mysql> use auth;
mysql> create table user(name char(10) not null,id int(48));
mysql> insert into user values('sofia','123');
mysql> select * from user;
[root@localhost ~]# ls /usr/local/mysql/data/auth
```

db.opt user.frm user.ibd

停库：

```
[root@localhost ~]# /etc/init.d/mysqld stop
Shutting down MySQL.. SUCCESS!
[root@localhost ~]# tar jcf backup/mysql_all-$(date +%F).tar.gz
/usr/local/mysql/data/
[root@localhost ~]# ls backup/
mysql_all-2020-04-24.tar.gz          # 对所有库做了完整备份
[root@localhost ~]# /etc/init.d/mysqld start
Starting MySQL. SUCCESS!
```

模拟数据删除：

```
[root@localhost ~]# mysql -uroot -p123456
mysql> drop database auth;
```

在删了auth库之后想还原该怎么办呢？

停库恢复数据：

```
[root@localhost ~]# /etc/init.d/mysqld stop
Shutting down MySQL.. SUCCESS!
[root@localhost ~]# mkdir restore
[root@localhost ~]# tar xf backup/mysql_all-2020-04-24.tar.gz -C restore/
[root@localhost ~]# ls restore/usr/local/mysql/data/
[root@localhost ~]# mv restore/usr/local/mysql/data/auth /usr/local/mysql/data/
[root@localhost ~]# /etc/init.d/mysqld start
Starting MySQL. SUCCESS!
```

在这出了一点小问题.....

```
[root@localhost ~]# /etc/init.d/mysqld stop
Shutting down MySQL.. SUCCESS!
[root@localhost ~]# rm -rf restore/*
[root@localhost ~]# rm -rf /usr/local/mysql/data/*
[root@localhost ~]# mv restore/usr/local/mysql/data/* /usr/local/mysql/data/
[root@localhost ~]# /etc/init.d/mysqld start
Starting MySQL. SUCCESS!
[root@localhost ~]# mysql -uroot -p123456 -e 'select * from auth.user;'
```

注意：

当我们恢复数据的时候，需要恢复所有的restore/usr/local/mysql/data/*，

而不能恢复某一个目录

PS：

这种方式用的不多，因为需要反复停库，比较麻烦，很多公司很少用

2、使用专用备份工具mysqldump

MySQL自带的备份工具mysqldump，可以很方便的对MySQL进行备份

通过该命令工具可以将数据库、数据表或全部的库导出为SQL脚本

（全都是SQL文件，都是SQL语句）

便于该命令在不同版本的MySQL服务器上使用

例如，当需要升级MySQL服务器时，

可以先使用mysgldump命令将原有库信息到导出，

然后直接在升级后的MySQL服务器中导入即可

(1) 对单个库进行完全备份

格式：mysqldump -u 用户名 -p[密码] [选项] --databases [数据库名] > /备份路径/备份文件名

示例：

```
[root@localhost ~]# mysqldump -uroot -p123456 --databases auth >
backup/auth-$(date +%Y%m%d).sql
[root@localhost ~]# ls backup/
auth-20200424.sql
[root@localhost ~]# cat backup/auth-20200424.sql
```

过滤有效信息：

```
[root@localhost ~]# cat backup/auth-20200424.sql | grep -Ev "^$|^/|^-"
```

都是SQL语句，复制粘贴就能执行

(2)对多个库进行完全备份

格式：

mysgldump -u 用户名 -p [密码] [选项] --databases 库名1[库名2]... > /备份路径/备份文件名

示例：

```
[root@localhost ~]# mysqldump -uroot -p123456 --databases mysql auth >
backup/mysql+auth-$(date +%Y%m%d).sql
[root@localhost ~]# ls backup/
mysql+auth-20200424.sql
```

(3)对所有库进行完全备份

格式：mysgldump -u 用户名 -p [密码] [选项] --opt --all-databases > /备份路径/备份文件名

当数据量比较大的时候，加上opt起到加速的作用

示例：

```
[root@localhost ~]# mysqldump -uroot -p123456 --opt --all-databases >
backup/mysql_all.$(date +%Y%m%d).sql
[root@localhost ~]# ls backup/
mysql_all.20200424.sql
```

(4)对表进行完全备份

格式：

```
mysqldump -u 用户名 -p [密码] [选项] 数据库名 表名 > /备份路径/备份文件名
```

示例：

```
[root@localhost ~]# mysqldump -uroot -p123456 auth user >
backup/auth_user-$(date +%Y%m%d).sql
[root@localhost ~]# ls backup/
auth_user-20200424.sql
```

(5)对表结构的备份---面试

创建表的语句，但不包括数据

他想设计一张表，但是他想参考这张表的结构，

所以将这个备份出来，那这个改巴改巴用~

格式：

```
mysqldump -u 用户名 -p [密码] -d数据库名 表名 > /备份路径/备份文件名
```

示例：

```
[root@localhost ~]# mysqldump -uroot -p123456 -d auth user >
backup/desc_auth_user-$(date +%Y%m%d).sql
[root@localhost ~]# ls backup/
desc_auth_user-20200424.sql
```

```
[root@localhost ~]# grep -Ev "^$|^/|^-" backup/auth_user-20200424.sql
DROP TABLE IF EXISTS `user`;
CREATE TABLE `user` (
  `name` char(10) NOT NULL,
  `id` int(48) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
LOCK TABLES `user` WRITE;
INSERT INTO `user` VALUES ('sofia',123);
UNLOCK TABLES;
[root@localhost ~]# grep -Ev "^$|^/|^-" backup/desc_auth_user-20200424.sql
DROP TABLE IF EXISTS `user`;
CREATE TABLE `user` (
  `name` char(10) NOT NULL,
  `id` int(48) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

数据

表结构

4.使用mysqldump备份后，恢复数据库

想办法让SQL文件执行一下

1、source命令

登录到MySQL数据库，执行

source备份sql_脚本路径

示例:

```
[root@localhost ~]# ll backup/auth-20200424.sql                    # auth库备份
-rw-r--r-- 1 root root 1962 4月  24 18:51 backup/auth-20200424.sql
[root@localhost ~]# mysql -uroot -p123456
```

模拟误删除库：

```
mysql> drop database auth;
```

还原数据：

```
mysql> source backup/auth-20200424.sql;
```

查看表的数据：

```
mysql> select * from auth.user;
```

模拟误删除库：

```
mysql> drop database auth;
[root@localhost ~]# mysql -uroot -p123456 < backup/auth-20200424.sql
[root@localhost ~]# mysql -uroot -p123456
mysql> select * from auth.user;
```

2、mysql 命令

格式:

```
mysql -u 用户名 -p [密码] < 库备份脚本的路径
mysql -u 用户名 -p [密码] 库名 < 表备份脚本的路径
```

注意：如果要恢复表的备份，需要指定在哪个库；恢复库不需要

示例:

```
mysql> use auth;
mysql> drop table user;
mysql> show databases;
| auth               |
```

恢复表的数据：

```
[root@localhost ~]# mysql -uroot -p123456 auth < backup/auth_user-20200424.sql
```

5.MySQL备份思路

1、定期实施备份，制定备份计划或策略，并严格遵守

2、除了进行完全备份，开启MySQL服务器的binlog日志功能是很重要的

(完全备份加上日志，可以对MySQL进行最大化还原)

3、使用统一和易理解的备份名称，推荐使用库名或者表名加上时间戳的命名规则，

如mysql_use-20181214.sql，不要使用backup1或者abc_之类没有意义的名字

6.MySQL完全备份案例

需求描述:

用户信息数据库为client,用户资费数据表为user_info

表结构如下所示

请为该公司制定合理的备份策略，

依据所制定的策略备份数据，

模拟数据丢失进行数据恢复

身份证	姓名	性别	用户ID号	资费
000000001	孙空武	男	011	100
000000002	蓝凌	女	012	98
000000003	姜纹	女	013	12
000000004	关园	男	014	38
000000005	罗中昆	男	015	39

ps:这里涉及一个MySQL存储中文的问题

```
[root@localhost ~]# mysql -uroot -p123456
mysql> create database client;
mysql> use client;
mysql> show variables like 'character_set_%';          # 查看系统的字符集；得让所有的字符集是utf8才可以
[root@localhost ~]# vim /etc/my.cnf
character_set_server=utf8          # 在 [mysqld]下插入
[root@localhost ~]# /etc/init.d/mysqld restart
[root@localhost ~]# mysql -uroot -p123456
mysql> show variables like 'character_set_%';
mysql> use client
mysql> create table user_info(身份证 int(20),姓名 char(20),性别 char(2),用户ID号 int(110),资费 in
t(10)) ;
```

这里出现了字符集错误的问题:

```
mysql> show create table user_info;
DEFAULT CHARSET=latin1
mysql> drop table user_info;
mysql> drop database client;
```

在创建表的时候指定字符集:

```
mysql> create table user_info(身份证 int(20),姓名 char(20),性别 char(2),用户ID号 int(110),资费 int(10)) DEFAULT CHARSET=utf8;
insert into user_info values('000000001','孙悟空','男','011','100');
insert into user_info values('000000003','姜文','女','013','12');
insert into user_info values('000000004','关元','男','014','38');
insert into user_info values('000000005','罗忠坤','男','015','39');
```

现在数据有啦，那么就来备份吧~

```
[root@localhost ~]# mysqldump -uroot -p123456 client user_info > backup/client.user_info-$(date +%Y%m%d).sql
[root@localhost ~]# ls -l backup/client.user_info-20200425.sql
-rw-r--r-- 1 root root 2074 4月  25 00:07 backup/client.user_info-20200425.sql
```

模拟恢复数据：

```
[root@localhost ~]# mysql -uroot -p123456 -e 'drop table client.user_info;'
[root@localhost ~]# mysql -uroot -p123456 -e 'use client; show tables;'
```

恢复表：

```
[root@localhost ~]# mysql -uroot -p123456 client < backup/client.user_info-20200425.sql
```

查看表：

```
[root@localhost ~]# mysql -uroot -p123456 -e 'select * from client.user_info';
```

定期备份数据的小脚本：

如果要做定时备份，最好找到mysqldump的绝对路径

```
[root@localhost ~]# which mysqldump
/usr/local/bin/mysqldump
```

ps:它的绝对路径应该在mysql/bin下，但是咱们做了软链接，

所以这里显示的快捷方式（软链接）

```
[root@localhost ~]# vim /opt/bak_client.sh
#!/bin/bash
# 备份client.user_info表 脚本
/usr/local/bin/mysqldump mysql -uroot -p123456 client user_info >backup/client.user_info-$(date +%Y%m%d).sql)
[root@localhost ~]# chmod +x /opt/bak_client.sh
[root@localhost ~]# crontab -e
0 0 * * *  /opt/bak_client.sh
```

7.MySQL数据库备份脚本

实验环境:

mysql-server: 192.168.200.107

mysql-client: 192.168.200.108

192.168.200.108

```
[root@mysql-client ~]# yum -y install mariadb mariadb-devel
```

实验要求:

对mysql-server的auth库和client库实现异地备份

每天凌晨2:00进行备份

撰写一个数据恢复脚本

MySQL服务端授权给予 select和 lock tables权限，用来备份

192.168.200.107

```
[root@mysql-server ~]# mysql -uroot -p123456
mysql> grant select,lock tables on auth.* to 'admin'@'192.168.200.108'
identified by '123456';
mysql> grant select,lock tables on client.* to 'admin'@'192.168.200.108'
identified by '123456';
mysql> flush privileges;
```

192.168.200.108

```
[root@mysql-client ~]# which mysql
[root@mysql-client ~]# which mysqldump
[root@mysql-client ~]# mysql -uadmin -p123456 -h 192.168.200.107
[root@mysql-client ~]# vim /opt/bakmysql.sh
    #!/bin/bash
    # MySQL数据库备份脚本
    # 设置登录变量
    MY_USER="admin"
    MY_PASS="123456"
    MY_HOST="192.168.200.107"
    MY_CONN="-u$MY_USER -p$MY_PASS -h$MY_HOST"
    # 设置备份的数据库
    MY_DB1="auth"
    MY_DB2="client"
    # 定义备份路径、工具、时间、文件名
    BF_DIR="backup"
    BF_CMD="/usr/bin/mysqldump"
```

```
    BF_TIME=$(date +%Y%m%d-%H%M)
    NAME_1="$MY_DB1-$BF_TIME"
    NAME_2="$MY_DB2-$BF_TIME"
    # 备份为.sql脚本，然后打包压缩（打包后删除源文件）
    [ -d $BF_DIR ] || mkdir -p $BF_DIR
    cd $BF_DIR
    $BF_CMD $MY_CONN --databases $MY_DB1 > $NAME_1.sql
    $BF_CMD $MY_CONN --databases $MY_DB2 > $NAME_2.sql
    /bin/tar zcf $NAME_1.tar.gz $NAME_1.sql --remove &>/dev/null
    /bin/tar zcf $NAME_2.tar.gz $NAME_2.sql --remove &>/dev/null
[root@mysql-client ~]# chmod +x /opt/bakmysql.sh
[root@mysql-client ~]# /opt/bakmysql.sh
[root@mysql-client ~]# ls
backup
[root@mysql-client ~]# ls backup/
auth-20200425-0117.tar.gz  client-20200425-0117.tar.gz
[root@mysql-client ~]# tar tf backup/auth-20200425-0117.tar.gz
auth-20200425-0117.sql
[root@mysql-client ~]# tar tf backup/client-20200425-0117.tar.gz
client-20200425-0117.sql
[root@mysql-client ~]# crontab -e
24 1   * * * /opt/bakmysql.sh
[root@mysql-client ~]# systemctl restart crond
[root@mysql-client ~]# cd backup/
[root@mysql-client backup]# rm -rf *
[root@mysql-client backup]# ls
auth-20200425-0124.tar.gz  client-20200425-0124.tar.gz
```

其实很多公司备份的套路和这个套路是一样的，

异地备份，无非就是给另外一台主机做一下授权

嘿嘿~还可以这么玩~

```
[root@mysql-client backup]# date -s "2020-04-26 1:23:50"
 #设定本机时间为后一天，在执行日志之前10秒
2020年 04月 26日 星期日 01:23:50 CST
[root@mysql-client backup]# systemctl restart crond
[root@mysql-client backup]# ls
```

```
auth-20200425-0124.tar.gz  client-20200425-0124.tar.gz
auth-20200426-0124.tar.gz  client-20200426-0124.tar.gz
[root@mysql-client backup]# date -s "2020-04-27 1:23:50"
2020年 04月 27日 星期一 01:23:50 CST
[root@mysql-client backup]# systemctl restart crond
[root@mysql-client backup]# ls
auth-20200425-0124.tar.gz  auth-20200427-0124.tar.gz    client-20200426-
0124.tar.gz
auth-20200426-0124.tar.gz  client-20200425-0124.tar.gz  client-20200427-
0124.tar.gz
```

现在这里呢，有三天的日志记录：

撰写数据恢复的脚本：--有点复杂需要吸收

这个脚本还不能直接引用

```
[root@mysql-client backup]# vim /opt/restore_mysql.sh
#!/bin/bash
#恢复MySQL数据库数据脚本
#设置变量
MY_USER="admin"
MY_PASS="123456"
MY_HOST="192.168.200.107"
BF_DIR="backup"
mkdir .aaa              # 隐藏文件
ls $BF_DIR |column -t > .aaa/db_list
awk -F'-' '{print $2}' .aaa/db_list > .aaa/dt.txt
read -p "请指定要恢复的数据库的日期(YYYYMMDD):" dt
if [ $dt -ge 20200401 ] && [ $dt -le 20200501 ];then
        grep "$dt" .aaa/dt.txt &>/dev/null
        if [ $? -ne 0 ];then
                echo "很抱歉，您恢复数据库的备份日期不在备份日期范围内"
        else
                echo "搜索到可恢复数据库如下:"
                awk -F'-' /$dt/'{print NR,$1}' .aaa/db_list
                read -p "请选择您要恢复数据库的编号:" nb
                nm=$(awk -F'-' /$dt/'{print NR,$1}' .aaa/db_list |awk
/$nb/'{print $2}')
```

```
                echo "现在开始恢复数据库:$nm到$dt"
                cd $BF_DIR
                onm=$(ls |grep "$nm-$dt")
                mkdir .bbb
                tar xf $onm -C .bbb
                mysql -u$MY_USER -p$MY_PASS -h$MY_HOST $nm < .bbb/*
                echo "$nm已经恢复到$dt"
                rm -rf .bbb
                cd - &>/dev/null
                rm -rf .aaa
        fi
else
echo "很抱歉，您恢复的数据库的备份不在备份日期范围内"
fi
```

其中的一些说明：

```
[root@mysql-client ~]# ls backup/ |column -t
auth-20200425-0124.tar.gz
auth-20200426-0124.tar.gz
auth-20200427-0124.tar.gz
client-20200425-0124.tar.gz
client-20200426-0124.tar.gz
client-20200427-0124.tar.gz
[root@mysql-client ~]# ls backup/ |column -t | awk -F'-' '/0425/{print $1}'
auth
client
[root@mysql-client ~]# ls backup/ |column -t | awk -F'-' '/0425/{print NR,$1}'
#定义序号
1 auth
4 client
```

把你想恢复的4月25号的两个库找出来，并且前面带序号

现在是没法恢复的，因为客户端没有权限恢复

192.168.200.107

```
mysql> grant all on auth.* to 'admin'@'192.168.200.108';
mysql> grant all on client.* to 'admin'@'192.168.200.108';
```

```
mysql> flush privileges;
```

恢复测试:

```
[root@mysql-client ~]# chmod +x /opt/restore_mysql.sh
[root@mysql-client ~]# /opt/restore_mysql.sh
请指定要恢复的数据库的日期(YYYYMMDD):20200426
搜索到可恢复数据库如下:
2 auth
5 client
请选择您要恢复数据库的编号:2
现在开始恢复数据库:auth到20200426
tar: auth-20200426-0124.sql：时间戳 2020-04-26 01:24:02 是未来的 25830.112796026 秒之后
auth已经恢复到20200426
```

8.MySQL增量备份与恢复

1.MySQL增量备份概念

使用mysgldump进行完全备份，备份的数据中有重复数据,备份时间与恢复时间过长

而增量备份就是备份自上一次备份之后增加或改变的文件或内容

2.增量备份的特点:

没有重复数据

备份量不大

时间短

恢复麻烦:需要上次完全备份及完全备份之后所有的增量备份才能恢复

而且要对所有增量备份进行逐个反推恢复

MySQL没有提供直接的增量备份办法

可以通过MySQL提供的二进制日志(binary logs(binlog))间接实现增量备份

3.MySQL二进制日志对备份的意义:

二进制日志保存了所有更新或者可能更新数据库的操作(SQL语句)

二进制日志在启动MySQL服务器后开始记录，

并在文件达到max_binlog_size所设置的大小

或者接收到flush-logs 命令后重新创建新的日志文件

```
[root@localhost ~]# vim /etc/my.cnf
52 max_binlog_size = 1024000       //二进制日志最大1M
```

只需定时执行flush-logs方法重新创建新的日志，生成二进制文件序列，
并及时把这些日志保存到安全的地方就完成了一个时间段的增量备份

要进行MySQL的增量备份
首先要开启二进制日志功能

4. 开启MySQL的二进制日志功能

方法一:

MySQL的配置文件的[mysgld]项中加入log-bin=文件存放路径/文件前缀，

如log-bin=mysql-bin，
然后重启mysqld服务
默认此配置存在

```
[root@localhost ~]# awk /log-bin/'{print NR,$0}' /etc/my.cnf
[root@localhost ~]# systemctl restart mysqld
[mysqld]
server-id=1
log-bin=mysql-bin          # 新版本要写，老版本不需要写
```

192.168.200.107：开启二进制日志

```
[root@mysql-server ~]# vim /etc/my.cnf
[mysqld]
server-id=1
log-bin=mysql-bin
[root@localhost ~]# systemctl restart mysqld
[root@localhost ~]# ls /usr/local/mysql/data/
auth       ib_buffer_pool  ib_logfile1    mysql              mysql-bin.index
sys
auto.cnf   ibdata1         ibtmp1         mysql-bin.000001   mysql.log
client     ib_logfile0     localhost.pid  mysql-bin.000002   performance_schema
```

正在写000002日志

方法二:

这个方法是临时的，一次性的，建议用上面的写入文件里，是永久的

使用mysqld-log-bin=文件存放路径/文件前缀
重新启动mysgld服务
每周选择服务器负载较轻的时间段，或者用户访问较少的时间段进行备份。

2.MySQL增量恢复

1.应用场景

1. 人为的SQL语句破坏了数据库的数据（误删除）
2. 在进行下一次全备之前发生系统故障导致数据库数据丢失
3. 在主从同步架构中，主库数据发生了故障，保证从库数据一致性

2.增量恢复的方法

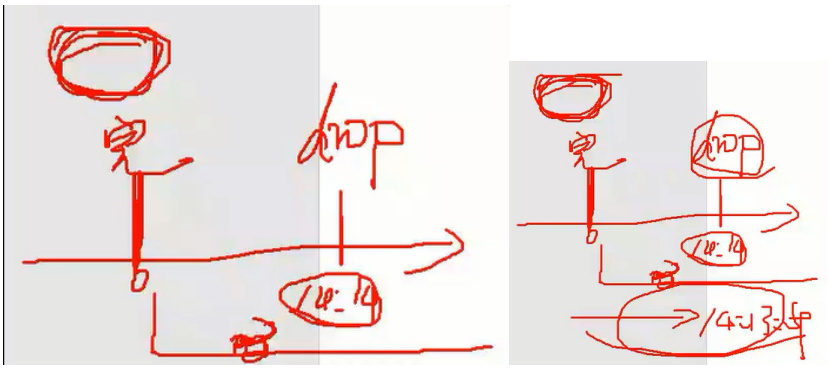
1、一般的恢复

备份的二进制日志内容全部恢复

格式: mysqlbinlog [-no-defaults] 增量备份文件 | mysql -u 用户名 -p 密码

有些情境下这种方式是不适用的：

比如：



我没有做增量备份，但是我开启了二进制日志，在凌晨时我做了一次完整备份，在今天14:14分时，我做了一个drop的操作；然后咱们在做完整备份的时候会产生一个备份，所以当我们恢复的时候，先把完整备份的数据恢复，现在我们就来到了0:00这个时间点的数据状态；因为我们平时的时候，二进制日志时开着的，比如它现在正在用的是3号日志（二进制日志里面会记录我们所有的SQL语句），所以现在呢，我们在14:14做了drop的误操作，我们在它的前一秒(14:13:59)停止对它的恢复，所以我们不能用上面的那条命令（一般恢复），它是把3号日志里面的内容全都恢复；而我要的可能只是一部分，所以我们可以基于时间点来恢复，即下面的这种方法；

2、基于时间点的恢复:

便于跳过某个发生错误的时间点实现数据恢复

格式:

从日志开头截止到某个时间点的恢复:

mysglbinlog [-no-defaults] -stop-datetime=’年-月-日 小时:分钟:秒’ 二进制日志 | mysql -u用户名 -p密码

从某个时间点到日志结尾的恢复:

mysglbinlog [--no-defaults] -start-datetime='年-月-日 小时:分钟:秒' 二进制日志 | mysql -u用户名 -p密码

从某个时间点到某个时间点的恢复:

mysqlbinlog [-no-defaults] --start-datetime='年-月-日 小时:分钟:秒' --stop-datetime='年-月-日 小时:分钟:秒' 二进制日志 | mysql -u用户名 -p密码

但是基于时间点的恢复也有问题，比如:

我在一秒钟内执行了，5个操作，而第5个操作是drop，所以我们在恢复的时候，前四个操作还得要啊，

我们每执行一个操作都会有一个**位置---position**

3、基于位置的恢复:

可能在同一时间点既有错误的操作也有正确的操作，基于位置进行恢复更加精准

格式:

mysqlbinlog --stop-position='操作id' 二进制日志lmysql-u用户名 -p 密码

mysqlbinlog -start-position='操作id' 二进制日志 |mysql -u用户名-p 密码

ps:两个一块用也是OK的~ （从一个位置到另一个位置，并非开头或结尾）

9.制定企业备份策略的思路---作为了解

1. 确定当前mysql是处于哪种表类型下工作的，它们支持事务处理还是非事务的，
   因为我们需要根据不同的特点来做一些设置。
2. 要选择备份的形式，是完全备份还是增量备份，它们各有优缺点
3. 为了保证恢复的完整性，我们得开启binary log功能，
   同时binlog给恢复工作也带来了很大的灵活性，
   可以基于时间点或是位置进行恢复。
   考虑到数据库性能，
   我们可以将binlog.文件保存到其他安全的硬盘中
4. 正如最初所提到的，备份操作和应用服务同时运行，
   这样就十分消耗系统资源了，会导致数据库服务性能下降，
   这就要求我们选择一个合适的时间(比如在应用负担很小的时候或者在从库上)再来进行备份操作
5. 不是备份完就万事大吉，
   我们还得确认备份是否可用
   所以之后的恢复测试是完全有必要的

备份在工作中就是写一个计划任务，让它去跑就OK了，但是我们要对它备份出来的文件定时的去做抽检；比如，统计一下大小（du -sh）,如果大小差不多，应该也没啥事，如果大小

不合理，那么这个备份可能就有异常；或者，我们可以把备份出来的数据往公司的测试环境或者开发环境啊，做还原测试；来去测试我们的备份是否可用；

注意：

- 根据数据更新频繁，则应该较为频繁的备份
- 数据重要，则在有适当更新时进行备份
- 在数据库压力小的时段进行备份，如一周一次完全备份，然后每天进行增量备份
- 中小公司，全备一般可一天一次
- 大公司可每周进行一次全备，每天进行一次增量备份
- 尽量为企业实现主从复制架构

10.MySQL企业备份案例

需求描述:

- 用户信息数据库为client,用户资费数据表为user_ info
- 请为该公司每周进行完全备份
- 每天为该公司进行增量备份
- 新增加的用户信息如表所示

```
mysql> select * from user_info;
+------------+--------+--------+-------------+--------+
| 身份证     | 姓名   | 性别   | 用户ID号    | 资费   |
+------------+--------+--------+-------------+--------+
| 000000006  | 张三   | 男     | 016         |     10 |
| 000000007  | 李四   | 女     | 017         |     91 |
| 000000007  | 王五   | 女     | 018         |     23 |
| 000000009  | 赵柳   | 男     | 019         |     37 |
| 0000000010 | 孙琦   | 男     | 020         |     36 |
+------------+--------+--------+-------------+--------+
```

```
[root@localhost ~]# mysql -uroot -p123456
mysql> use client
mysql> create table user_info(身份证 char(20) not null,姓名 char(20) not null,性别 char(4),用户ID号 char(10) not null,资费 int(10)) DEFAULT CHARSET=utf8;
mysql> insert into user_info values('000000006','张三','男','016','10');
mysql> insert into user_info values('000000007','李四','女','017','91');
mysql> insert into user_info values('000000007','王五','女','018','23');
mysql> select * from user_info;
```

```
mysql> select * from user_info;
+-----------+--------+--------+-------------+--------+
| 身份证    | 姓名   | 性别   | 用户ID号    | 资费   |
+-----------+--------+--------+-------------+--------+
| 000000006 | 张三   | 男     | 016         |     10 |
| 000000007 | 李四   | 女     | 017         |     91 |
| 000000007 | 王五   | 女     | 018         |     23 |
+-----------+--------+--------+-------------+--------+
```

先进行一次完全的备份

```
[root@localhost ~]# mkdir /mysql_bak
[root@localhost ~]# mysqldump -uroot -p123456 client user_info
>/mysql_bak/client_userinfo-$(date+%F).sql       # 备份表
[root@localhost ~]# mysqldump -uroot -p123456 --databases client
>/mysql_bak/client-$(date +%F).sql     # 备份库
[root@localhost ~]# ls /mysql_bak/
[root@localhost ~]# ls /usr/local/mysql/data/
mysql-bin.000002
# 使用2号日志，这个实验是在前边实验基础上做的，所以有000001号日志，日志应该从000002号开始
[root@localhost ~]# mysqladmin -uroot -p123456 flush-logs          # 切日志
[root@localhost ~]# ls /usr/local/mysql/data/
mysql-bin.000003       # 产生新的日志，接下来的操作，数据变更都会记录到3号日志中
[root@localhost ~]# mysql -uroot -p123456
mysql>  use client
mysql> insert into user_info values('000000009','赵柳','男','019','37');
mysql> insert into user_info values('0000000010','孙琦','男','020','36');
```

```
mysql> select * from user_info;
+------------+--------+--------+------------+--------+
| 身份证     | 姓名   | 性别   | 用户ID号   | 资费   |
+------------+--------+--------+------------+--------+
| 000000006  | 张三   | 男     | 016        |     10 |
| 000000007  | 李四   | 女     | 017        |     91 |
| 000000007  | 王五   | 女     | 018        |     23 |
| 000000009  | 赵柳   | 男     | 019        |     37 |
| 0000000010 | 孙琦   | 男     | 020        |     36 |
+------------+--------+--------+------------+--------+
```

```
[root@localhost ~]# mysqladmin -uroot -p123456 flush-logs             #切日志
[root@localhost ~]# ls /usr/local/mysql/data/
mysql-bin.000004
[root@localhost ~]# cp /usr/local/mysql/data/mysql-bin.000003 /mysql_bak/
# 备份
[root@localhost ~]# ls /mysql_bak/
client-2020-04-25.sql  client_userinfo-2020-04-25.sql  mysql-bin.000003
```

前两个是表和库的完整备份；最后是增量备份

查看日志，分析日志

```
[root@localhost ~]# mysqlbinlog  /mysql_bak/mysql-bin.000003       # 加 -v 查看完整信息
COMMIT/*!*/;           # 无法查看具体操作
### INSERT INTO `client`.`user_info`
```

```
### SET
###   @1='0000000010'
###   @2='孙琦'
###   @3='男'
###   @4='020'
###   @5=36
# at 726
```

[root@localhost ~]# mysqlbinlog -v /mysql_bak/mysql-bin.000003

模拟误删除：

```
[root@localhost ~]# mysql -uroot -p123456 -e 'drop table client.user_info;'
# 删表
[root@localhost ~]# mysql -uroot -p123456 -e 'select * from client.user_info;'
Table 'client.user_info' doesn't exist
```

恢复数据：

```
[root@localhost ~]# mysql -uroot -p123456 client < /mysql_bak/client_userinfo-2020-04-25.sql
#先还原表的完整备份；
[root@localhost ~]# mysql -uroot -p123456 -e 'select * from client.user_info;'
[root@localhost ~]# mysqlbinlog --no-defaults /mysql_bak/mysql-bin.000003 | mysql -uroot -p123456        #恢复增量备份里面的普通恢复，3号日志里面的全部恢复，把增量里面的语句都执行了一遍
```

模拟删除：

```
[root@localhost ~]# mysql -uroot -p123456 -e 'drop table client.user_info;'
[root@localhost ~]# mysql -uroot -p123456 client < /mysql_bak/client_userinfo-2020-04-25.sql
# 恢复完整
```

对于恢复增量时的选择：恢复赵柳而不恢复孙琦

基于时间点恢复：找BEGIN COMMITE之间是一个完整的语句

```
[root@localhost ~]# mysqlbinlog --no-defaults --stop-datetime="2020-04-25 21:05:01" /mysql_bak/mysql-bin.000003 | mysql -uroot -p123456
[root@localhost ~]# mysql -uroot -p123456 -e 'select * from client.user_info;'
+-----------+--------+--------+------------+--------+
| 身份证    | 姓名   | 性别   | 用户ID号   | 资费   |
+-----------+--------+--------+------------+--------+
| 000000006 | 张三   | 男     | 016        |     10 |
| 000000007 | 李四   | 女     | 017        |     91 |
| 000000007 | 王五   | 女     | 018        |     23 |
| 000000009 | 赵柳   | 男     | 019        |     37 |
+-----------+--------+--------+------------+--------+
```

模拟删除：

```
[root@localhost ~]# mysql -uroot -p123456 -e 'drop table client.user_info;'
[root@localhost ~]# mysql -uroot -p123456 -e 'drop table client.user_info;'
[root@localhost ~]# mysql -uroot -p123456 client < /mysql_bak/client_userinfo-2020-04-25.sql
# 恢复完整
[root@localhost ~]# mysqlbinlog --no-defaults --start-datetime="2020-04-25 21:05:01" /mysql_bak/mysql-bin.000003 | mysql -uroot -p123456
[root@localhost ~]# mysql -uroot -p123456 -e 'select * from client.user_info;'
+------------+--------+--------+------------+--------+
| 身份证     | 姓名   | 性别   | 用户ID号   | 资费   |
+------------+--------+--------+------------+--------+
| 000000006  | 张三   | 男     | 016        |     10 |
| 000000007  | 李四   | 女     | 017        |     91 |
| 000000007  | 王五   | 女     | 018        |     23 |
| 0000000010 | 孙琦   | 男     | 020        |     36 |
+------------+--------+--------+------------+--------+
```

基于位置的恢复数据：

模拟删除：

```
[root@localhost ~]# mysql -uroot -p123456 -e 'drop table client.user_info;'
[root@localhost ~]# mysql -uroot -p123456 -e 'drop table client.user_info;'
[root@localhost ~]# mysql -uroot -p123456 client < /mysql_bak/client_userinfo-2020-04-25.sql
# 恢复完整
[root@localhost ~]# mysqlbinlog --no-defaults --stop-position="455" /mysql_bak/mysql-bin.000003 | mysql -uroot -p123456
[root@localhost ~]# mysql -uroot -p123456 -e 'select * from client.user_info;'
+-----------+--------+--------+------------+--------+
| 身份证    | 姓名   | 性别   | 用户ID号   | 资费   |
+-----------+--------+--------+------------+--------+
| 000000006 | 张三   | 男     | 016        |     10 |
| 000000007 | 李四   | 女     | 017        |     91 |
| 000000007 | 王五   | 女     | 018        |     23 |
| 000000009 | 赵柳   | 男     | 019        |     37 |
+-----------+--------+--------+------------+--------+
```

模拟删除：

```
[root@localhost ~]# mysql -uroot -p123456 -e 'drop table client.user_info;'
[root@localhost ~]# mysql -uroot -p123456 -e 'drop table client.user_info;'
[root@localhost ~]# mysql -uroot -p123456 client < /mysql_bak/client_userinfo-2020-04-25.sql
# 恢复完整
[root@localhost ~]# mysqlbinlog --no-defaults --start-position="455" /mysql_bak/mysql-bin.000003
```

```
| mysql -uroot -p123456
[root@localhost ~]# mysql -uroot -p123456 -e 'select * from client.user_info;'
+------------+--------+--------+------------+--------+
| 身份证     | 姓名   | 性别   | 用户ID号   | 资费   |
+------------+--------+--------+------------+--------+
| 000000006  | 张三   | 男     | 016        |     10 |
| 000000007  | 李四   | 女     | 017        |     91 |
| 000000007  | 王五   | 女     | 018        |     23 |
| 0000000010 | 孙琦   | 男     | 020        |     36 |
+------------+--------+--------+------------+--------+
```

实现完整/增量备份的脚本：

```
[root@localhost ~]# vim /opt/mysql_bak_wanbei.sh
    #!/bin/bash
    # MySQL数据库完全备份脚本
    # 设置登录变量
    MY_USER="root"
    MY_PASS="123456"
    MY_HOST="localhost"
    MY_CONN="-u$MY_USER -p$MY_PASS -h$MY_HOST"
    # 设置备份的数据库(表)
    MY_DB="client"
    # 定义备份路径、工具、时间、文件名
    BF_DIR="/mysql_bak/wanbei"
    BF_CMD="/usr/bin/mysqldump"
    BF_TIME=$(date +%Y%m%d-%H%M)
    NAME="$MY_DB-$BF_TIME"
    # 备份为.sql脚本，然后打包压缩（打包后删除源文件）
    [ -d $BF_DIR ] || mkdir -p $BF_DIR
    cd $BF_DIR
    $BF_CMD $MY_CONN --databases $MY_DB > $NAME.sql
    /bin/tar zcf $NAME.tar.gz $NAME.sql --remove &>/dev/null
[root@localhost ~]# chmod +x /opt/mysql_bak_wanbei.sh
[root@localhost ~]# vim /opt/mysql_bak_zengbei.sh
    #!/bin/bash
    # MySQL数据库增量备份脚本
    # 设置登录变量
    MY_USER="root"
    MY_PASS="123456"
    MY_HOST="localhost"
```

```
MY_CONN="-u$MY_USER -p$MY_PASS -h$MY_HOST"
# 定义备份路径、工具、二进制日志前缀、二进制日志存放路径
BF_TIME=$(date +%Y%m%d)
BF_DIR="/mysql_bak/zengbei/$BF_TIME"
CMD="/usr/bin/mysqladmin"
QZ="mysql-bin"
LOG_DIR="/usr/local/mysql/data"
# 拷贝二进制日志
[ -d $BF_DIR ] || mkdir -p $BF_DIR
$CMD $MY_CONN flush-logs
/bin/cp -p $(ls $LOG_DIR/$QZ.* | awk -v RS="" '{print $(NF-2)}') $BF_DIR
[root@localhost ~]# chmod +x /opt/mysql_bak_zengbei.sh
```

注意：备份的是最新的上一个

对/bin/cp -p $(ls $LOG_DIR/$QZ.* | awk -v RS="" '{print $(NF-2)}') $BF_DIR 的说明

```
[root@localhost ~]# ls /usr/local/mysql/data/mysql-bin.*
/usr/local/mysql/data/mysql-bin.000001  /usr/local/mysql/data/mysql-bin.000004
/usr/local/mysql/data/mysql-bin.000002  /usr/local/mysql/data/mysql-bin.index
/usr/local/mysql/data/mysql-bin.000003
[root@localhost ~]# ls /usr/local/mysql/data/mysql-bin.* |column -t
/usr/local/mysql/data/mysql-bin.000001
/usr/local/mysql/data/mysql-bin.000002
/usr/local/mysql/data/mysql-bin.000003
/usr/local/mysql/data/mysql-bin.000004
/usr/local/mysql/data/mysql-bin.index
```

-v RS="" 定义变量的作用;当做行来对待

```
[root@localhost ~]# ls /usr/local/mysql/data/mysql-bin.* | awk -v RS="" '{print $0}'
/usr/local/mysql/data/mysql-bin.000001
/usr/local/mysql/data/mysql-bin.000002
/usr/local/mysql/data/mysql-bin.000003
/usr/local/mysql/data/mysql-bin.000004
```

```
/usr/local/mysql/data/mysql-bin.000005
/usr/local/mysql/data/mysql-bin.000006
/usr/local/mysql/data/mysql-bin.000007
/usr/local/mysql/data/mysql-bin.000008
/usr/local/mysql/data/mysql-bin.000009
/usr/local/mysql/data/mysql-bin.000010
/usr/local/mysql/data/mysql-bin.index
[root@localhost ~]# ls /usr/local/mysql/data/mysql-bin.* | awk -v RS=""
'{print $(NF-1)}'
/usr/local/mysql/data/mysql-bin.000010
[root@localhost ~]# ls /usr/local/mysql/data/mysql-bin.* | awk -v RS=""
'{print $(NF-2)}'
/usr/local/mysql/data/mysql-bin.000009
```

定义计划任务：

```
[root@localhost ~]# crontab -e
0 0 * * 1  /opt/mysql_bak_wanbei.sh           # 每周一凌晨进行完备
0 0 * * 2-7 /opt/mysql_bak_zengbei.sh        # 每天0:00 进行增量备份
```

11.mysql日志种类

1.错误日志(error log)

```
[root@localhost ~]# cat /usr/local/mysql/data/mysql.log
```

平时服务出现问题时，要查看的

- 服务器启动和关闭进程过程中的信息;
- 服务器运行过程中的错误信息;
- 事件调度器运行一个事件时产生的信息;

```
[root@localhost ~]# vim /etc/my.cnf
[mysqld_safe]
log-error=/usr/local/mysql/data/mysql.log
[root@localhost ~]# /etc/init.d/mysqld restart
[root@localhost ~]# cat /usr/local/mysql/data/mysql.log
```

2.二进制日志(binary log)

1、即 binlog，二进制日志存储修改数据库中表数据的所有动作

包含了所有更新了数据或者已经潜在更新了数据的所有语句

2、不包含没有修改任何数据的语句。

如果你想要记录所有语句（例如，为了识别有问题的查询）

你应使用一般查询日志

3、该日志还存储了语句执行期间耗时的相关信息

4、二进制日志有两种功能：

- 数据恢复（基于时间点 | 位置恢复的）
- 主从复制（将主服务器发生的任何改变复制到从服务器上保证数据同步）

```
[root@localhost ~]# vim /etc/my.cnf
[mysqld]
server-id=1
log-bin=mysql-bin
[root@localhost ~]# /etc/init.d/mysqld restart
[root@localhost ~]# ls -l /usr/local/mysql/data/mysql-bin.*
-rw-r----- 1 mysql mysql 1595 4月  23 14:24 /usr/local/mysql/data/mysql-bin.000001
-rw-r----- 1 mysql mysql  804 4月  23 14:27 /usr/local/mysql/data/mysql-bin.000002
-rw-r----- 1 mysql mysql 9504 4月  23 15:17 /usr/local/mysql/data/mysql-bin.000003
-rw-r----- 1 mysql mysql  114 4月  23 15:17 /usr/local/mysql/data/mysql-bin.index
```

3.查询日志 (general log)

一般很少开启这个功能

1、该日志文件记录服务器上所做的所有查询。

2、由于该日志记录了所有的查询操作，包括所有的select，

日志体积比较大，开启后会对存储性能有较大的影响，所以一般不开启

3、该日志一般用于跟踪某些特殊的SQL性能问题才会短暂开启

4、指定mysql查询日志存放路径：

- 使用mysqld --log[=file_name]或-l [file_name]选项启动它
- 如果没有给定file_name的值， 默认名是host_name.log

- 如果file_name为绝对路径则在该目录下创建日志文件，否则在数据目录下创建该日志文件

5、mysq按照它接收的顺序

而不是按照语句执行的顺序记录语句到查询日志

这就有可能与执行的顺序不同

6、服务器重新启动和日志刷新不会产生新的一般查询日志文件

开启日志：

```
[root@localhost ~]# vim /etc/my.cnf
  4 general_log=1
  5 general_log_file=/usr/local/mysql/data/query.log
```

重启服务：

```
[root@localhost ~]# /etc/init.d/mysqld restart
Shutting down MySQL.. SUCCESS!
Starting MySQL. SUCCESS!
```

查看日志：

```
[root@localhost ~]# ls /usr/local/mysql/data/
query.log
```

在另一终端进行查询操作：

```
[root@localhost ~]# mysql -uroot -p123456
mysql> show databases;
mysql> select * from client.user_info;
```

跟踪日志：

```
[root@localhost ~]# tail -f /usr/local/mysql/data/query.log
/usr/local/mysql/bin/mysqld, Version: 5.7.24-log (MySQL Community Server (GPL)).
started with:
Tcp port: 0  Unix socket: /tmp/mysql.sock
Time                 Id Command    Argument
2020-04-23T07:41:00.787428Z         2 Connect   root@localhost on  using Socket
2020-04-23T07:41:00.789059Z         2 Query     select @@version_comment limit 1
2020-04-23T07:41:08.964465Z         2 Query     show databases
2020-04-23T07:41:22.035673Z         2 Query     select * from client.user_info
```

4.慢查询日志 (slow log)

1、慢查询日志记录的就是执行时间较长的查询操作，即Slow query

2、用来收集那些花费太长时间（超过指定时间）执行的SQL语句

该指定时间由long_query_time服务器变量设定

3、指定mysql慢查询日志存放路径

- 用mysqld --log-slow-queries[=file_name]选项启动
- 如果没有给出file_name值，默认为主机名，后缀为-slow.log
- 如果给出了文件名，但不是绝对路径名，文件则写入数据目录

4、慢查询日志可以用来找到执行时间长的查询，可以用于SQL优化

使用mysqldumpslow命令获得日志中显示的查询摘要来处理慢查询日志

5、获得初始表锁定的时间不算作执行时间

6、语句执行完并且所有锁释放后记入慢查询日志，记录顺序可以与执行顺序不相同

开启日志：

```
[root@localhost ~]# vim /etc/my.cnf
  7 slow-query-log=1
  8 slow_query_log_file="/usr/local/mysql/data/slow_query.log"
  9 long_query_time=10
```

重启服务：

```
[root@localhost ~]# /etc/init.d/mysqld restart
Shutting down MySQL.... SUCCESS!
Starting MySQL.. SUCCESS !
[root@localhost ~]# ls -l /usr/local/mysql/data/slow_query.log
-rw-r----- 1 mysql mysql 181 4月  23 15:46 /usr/local/mysql/data/slow_query.log
[root@localhost ~]# tail -f /usr/local/mysql/data/slow_query.log
/usr/local/mysql/bin/mysqld, Version: 5.7.24-log (MySQL Community Server (GPL)).
started with:
Tcp port: 0  Unix socket: /tmp/mysql.sock
Time                 Id Command    Argument
```

5.中继日志(relay log)

1、存储IO线程从主服务器二进制日志复制的操作

2、主要用于MySQL Replication集群中的从服务器上

开启日志：

```
[root@localhost ~]# vim /etc/my.cnf
relay-log=relay-log-bin                                          #开启中继日志
功能,和日志的前缀名
relay-log-index=slave-relay-bin.index                #开启中继日志索引
[root@localhost ~]# /etc/init.d/mysqld restart
```

TAB键补全功能：

临时支持：

补全：表名 库名

```
[root@localhost ~]# vim /etc/my.cnf
 21 [mysql]
 22 auto-rehash
[root@localhost ~]# /etc/init.d/mysqld restart
Shutting down MySQL.. SUCCESS!
Starting MySQL. SUCCESS!
[root@localhost ~]# mysql -uroot -p123456
mysql> use client
mysql> select * from u                          # 双击TAB键
unlock tables         user_info             user_info.性别      user_info.资费
use                   user_info.姓名      user_info.用户ID号 user_info.身份证
```