任务背景

"手里有粮，心里不慌，数据有备，喝茶看报"。对于重要的数据做好备份，是我们每个运维人员和DBA的重要职责。备份只是一种手段，我们最终目的是当数据出现问题时能够及时的通过备份进行恢复。所以，现在需要你对我们现有的MySQL数据库进行备份策略的制定以及实施。

任务要求

1. 选择合适的工具和方法对mysql数据库进行备份
2. 编写脚本实现自动化备份

课程目标

- 了解MySQL常见的备份方式和类型
- 能够使用mysqldump工具进行数据库的备份。如全库备份，库级别备份，表级别备份
- 能够使用mysqldump工具+binlog日志实现增量备份
- 理解xtrabackup工具实现增量备份的原理和方法
- 能够使用xtrabackup工具对数据库进行全备和增备

理论储备

一、MySQL备份概述

1. 关于备份你要知道的

思考：备份和冗余有什么区别?

备份： 能够防止由于**机械故障**以及**人为误操作**带来的数据丢失，例如将数据库文件保存在了其它地方。

冗余： 数据有多份冗余，但不等备份，**只能防止机械故障**带来的数据丢失，例如主备模式、数据库集群。

2.备份什么

数据库：一堆物理文件的集合；日志文件+数据文件+配置文件

DB BINLOG my.cnf

3.备份过程须考虑的因素

- 必须制定详细的备份计划（备份频率、时间点、周期）
- 备份数据应该放在非数据库本地，并建议有多份副本
- 必须做好**数据恢复的演练**（每隔一段时间，对备份的数据在测试环境中进行模拟恢复，保证当出现数据灾难的时候能够及时恢复数据。）
- 根据数据应用的场合、特点选择正确的备份工具。
- 数据的一致性
- 服务的可用性

4.备份类型

4.1 逻辑备份

- 备份的是**建表、建库、插入等操作所执行SQL语句**（DDL DML DCL）。

- 适用于中小型数据库，效率相对较低。 一般在数据库**正常提供服务**的前提下进行，如：mysqldump、mydumper、into outfile（表的导出导入）等。

4.2 物理备份

- 直接复制数据库文件
- 适用于大型数据库环境，不受存储引擎的限制，但不能恢复到不同的MySQL版本。
- 一般是在数据库彻底关闭或者不能完成正常提供服务的前提下进行的备份）；如：tar、cp、xtrabackup（数据库可以正常提供服务）、lvm snapshot、rsync等

4.3 在线热备（冗余）

- MySQL的replication架构，如M-S|M-S-S|M-M-S等
- 实时在线备份

5.备份工具

- **社区版安装包中的备份工具：**

1. **mysqldump** （逻辑备份，只能全量备份）

   1）企业版和社区版都包含

   2）本质上使用SQL语句描述数据库及数据并导出

   3）在MYISAM引擎上锁表，Innodb引擎上锁行

   4）数据量很大时不推荐使用

2. **mysqlhotcopy**（物理备份工具）

   1）企业版和社区版都包含

   2）perl写的一个脚本，本质上是使用锁表语句后再拷贝数据

   3）只支持MYISAM数据引擎

- **企业版安装包中的备份工具：**

  mysqlbackup

  1）在线备份

  2）增量备份

  3）部分备份

  4）在某个特定时间的一致性状态的备份

- **第三方备份工具：**

  XtraBackup和innobackupex（物理备份）

  1）Xtrabackup是一个对**InnoDB**做数据备份的工具，支持在线热备份（备份时不影响数据读写）， 是商业备份工具**InnoDB Hotbackup**的一个很好的替代品。

  2）Xtrabackup有两个主要的工具：xtrabackup、innobackupex

  a、xtrabackup只能备份InnoDB和XtraDB两种数据表，不能备份myisam类型的表。

  b、innobackupex是将Xtrabackup进行封装的perl脚本，所以能同时备份处理innodb和myisam的存储引擎，但在处理myisam时需要加一个读锁。

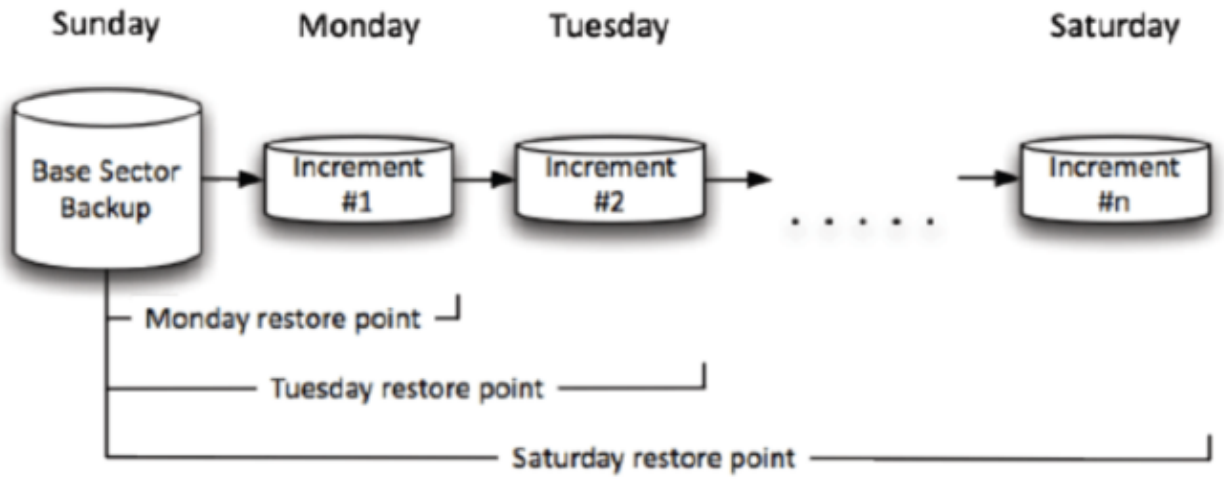
- **mydumper**（逻辑备份）

  多线程备份工具

https://launchpad.net/mydumper/mydumper-0.9.1.tar.gz 2015-11-06（最后更新时间）

6.备份方法

- 完全备份（全备）
- 增量备份（增量备份基于全量备份）



7.需要重点掌握的

- 逻辑备份

mysqldump工具

逻辑数据的导入导出（into outfile）

- 物理备份

xtrabackup和innobackupex

lvm-snapshot（mysql数据库在逻辑卷）

二、MySQL的逻辑备份

1. 了解binary log

1.1 二进制日志格式

- STATEMENT（5.7.7以前默认）：基于语句的日志记录
- ROW（5.7.7以后默认）：基于行的日志记录
- MIXED：混合日志记录

In MySQL 5.6, the default binary logging format is STATEMENT.

```
• Replication capabilities in MySQL originally were based on propagation of SQL statements from
master to slave. This is called statement-based logging. You can cause this format to be used by
starting the server with
--binlog-format=STATEMENT.

• In row-based logging, the master writes events to the binary log that indicate how individual
table rows are affected. You can cause the server to use row-based logging by starting it with
--binlog-format=ROW.

• A third option is also available: mixed logging. With mixed logging, statement-based logging
is used by default, but the logging mode switches automatically to row-based in certain cases as
described below. You can cause MySQL to use mixed logging explicitly by starting mysqld with the
option
 --binlog-format=MIXED.
```

1.2 二进制日志的用途

- 查看数据库的变更历史
- 数据库的增量备份
- 数据库的灾难恢复
- MySQL复制

1.3 开启二进制日志

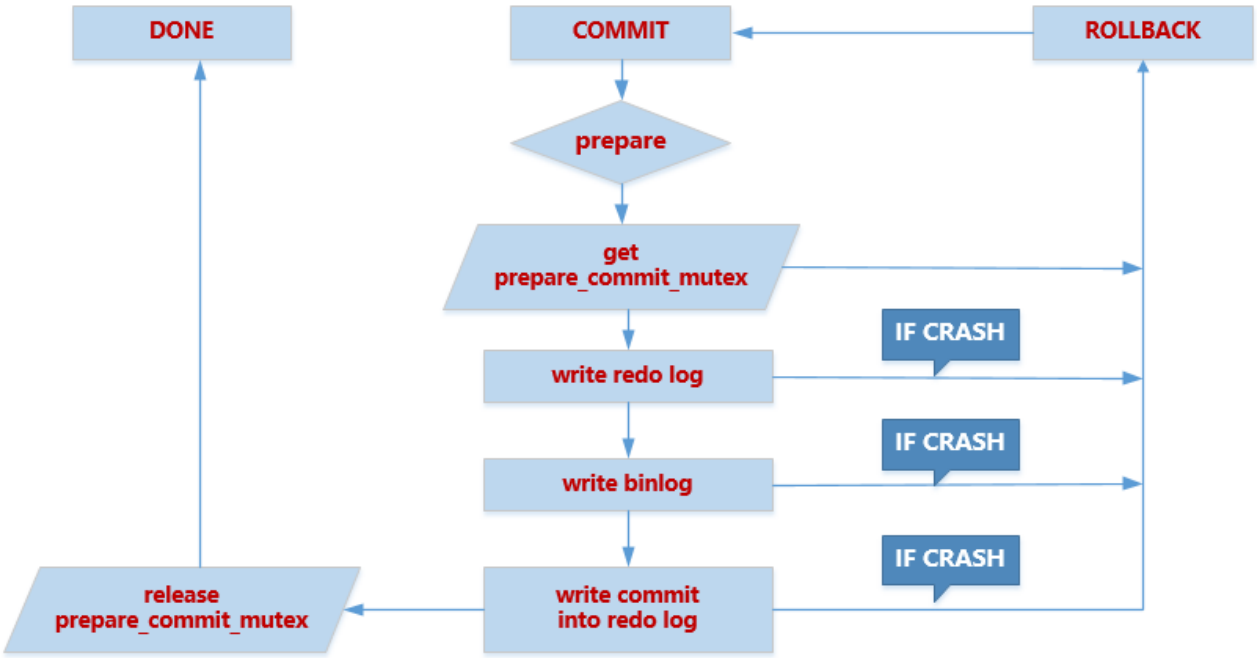
```
mysql> set log_bin=ON;
ERROR 1238 (HY000): Variable 'log_bin' is a read only variable

修改配置文件my.cnf
log_bin = /usr/local/mysql/data/mybinlog
```

1.4 redo log日志作用

- 用于**数据库实例**崩溃恢复，保证数据的一致性；
- **前滚**已经提交的事务到数据文件；**回滚**未提交的事务

1.5 日志记录过程



2. mysqldump备份

本质：导出的是sql语句文件

优点：无论是什么存储引擎，都可以用mysqldump备成sql语句

缺点：速度较慢,导入时可能会出现格式不兼容的突发状况.无法直接做增量备份.

提供三种级别的备份，表级，库级和全库级

```
Usage:
mysqldump [OPTIONS] database [tables]                表级别备份
mysqldump [OPTIONS] --databases [OPTIONS] DB1 [DB2 DB3...]     库级别备份
mysqldump [OPTIONS] --all-databases [OPTIONS]      全库级别备份

说明:
如果备份对象下的数据库绝大多数都是myisam类型表，为了保证数据的一致性，备份时需要锁定表。
如果备份的数据库里的表与其他库没有关系的话，那么只需要锁定该库下的表就可以了，如：--lock-tables
如果备份的数据库里的表与其他库有关系的话，那么需要锁定整个mysql数据库的所有库下的所有表；如:
--lock-all-tables
如果是针对innodb的表进行备份由于innodb是事务型的引擎，会话与会话之间是隔离的，所以备份的时候不影响数据库
的正常使用，无需锁表。

常用参数:
--flush-logs, -F         开始备份前刷新日志
--flush-privileges       备份包含mysql数据库时刷新授权表
--lock-all-tables, -x    MyISAM一致性，服务可用性（针对所有库所有表）
--lock-tables, -l        备份前锁表（针对要备份的库）
--master-data=1|2        该选项将会记录binlog的日志位置与文件名并追加到文件中
详细说明:
1. 该选项等于2表示将二进制日志的位置和文件名写入到备份文件，并在dump文件中注释掉这一行;
2. 该选项等于1表示将二进制日志的位置和文件名写入到备份文件，不在dump文件中注释掉这一行;
3. 恢复时会执行，默认是1
```

```
4. This option requires the RELOAD privilege and the binary log must be enabled.
5. 这个选项会自动打开--lock-all-tables，关闭--lock-tables;除非同时设置了
--single-transaction（这种情况下，全局读锁只会在开始dump的时候加上一小段时间）
6. 在任何情况下，所有日志中的操作都会发生在导出的准确时刻。

--single-transaction    适用InnoDB引擎，保证一致性，服务可用性
详细说明：
1. 将导出操作封装在一个事务内来使得导出的数据是一个一致性快照,进而保证数据的一致性。目前只支持InnoDB存储引擎，其他引擎不能保证导出是一致的。
2. 当导出开启了--single-transaction选项时，要确保导出文件有效（正确的表数据和二进制日志位置），就要保证没有其他连接会执行如下语句：ALTER TABLE, DROP TABLE, RENAME TABLE, TRUNCATE TABLE，这会导致一致性快照失效。这个选项开启后会自动关闭lock-tables。
```

2.1 示例1

使用mysqldump工具进行表、库、全库级别备份恢复

```
表级备份：
# mysqldump -p123 db01 emp > /tmp/mysqlbak/emp.sql 备份单个表
# mysqldump -p123 db01 emp dept > /tmp/mysqlbak/emp.sql  备份多个表

表级恢复：
# mysql -p db01 </tmp/mysqlbak/emp.sql
或者在mysql数据库内使用source命令来执行外部的sql文件
mysql> source /tmp/mysqlbackup/emp.sql

库级备份：
# mysqldump --databases db01 -p > /tmp/mysqlbak/db01.sql    备份单个库
# mysqldump --databases db01 db02 -p > /tmp/mysqlbak/db01.sql  备份多个库

库级恢复：
# mysql -p </tmp/mysqlbak/db01.sql
mysql> source /tmp/mysqlbak/db01.sql

-o, --one-database  指定恢复某个库

全库级备份：
考虑到数据库里有innodb，也有其他类型的表，那么就只能锁表备份
# mysqldump -p --lock-tables  --all-databases > /tmp/mysqlbak/alldb.sql

全库恢复：     注意mysqldump的恢复速度较慢，所以数据太大需要的时间较长
rm /mysqldata56/*  -rf     这样删除数据是需要初始化的

恢复步骤：
1、初始化
2、把数据库先启起来
3、恢复（备份的sql文件来恢复业务数据）
```

2.2 示例2

完全备份(mysqldump)+增量备份(binlog)

- 适用于中小型数据库；通过结合二进制日志文件，把数据库恢复到最新的状态
- 二进制日志文件默认会记录下所有对数据库数据变化的操作
- 二进制日志文件中会记录某个操作的详细sql语句，还有执行的时候环境、时间、以及该记录在二进制日志文件的起始和结束点pos值

```
回顾:
1. error log  记录mysql服务端在运行时产生的错误信息，以及mysql启动和关闭时的日志信息（排错）
2. slow log   慢查询时间阀值，以秒为单位，如果超过这个阀值就是慢查询   （调优）
3. bin log  记录对数据库增、删、改的SQL操作，可以使用这个日志做增量备份   （备份）
4. Relay log  中继日志（主从复制日志）   从机器上从主机器复制过来日志，根据日志来同步数据（复制）
```

```
mysqlbinlog  工具，查看二进制日志的
--start-datetime=name    开始的时间
--stop-datetime=name     结束的时间
--start-position=#    开始的位置(POS)
--stop-position=#     结束的位置

mysql < xxx.sql
命令执行结果（sql）  |mysql

用法:
mysqlbinlog --start-position=120 --stop-position=455   mybinlog.000002 |mysql -p456
```

demo1：先做全量备份，然后更新数据并误操作，binlog进行数据恢复

```
思路:
1. 先做全量备份（mysqldump）    01:30
2. 对数据库做更改操作
3. 备份二进制日志文件           8:00
4. 模拟故障进行测试验证


步骤:
1. 全库级别备备
# mysqldump --single-transaction --flush-logs --master-data=2 --all-databases >
/tmp/backup/all.sql -p
Enter password:

2. 数据库更改操作
mysql> select * from db01.t1;
+------+-------+----------+-----------+
| id   | name  | salary   | dept      |
+------+-------+----------+-----------+
|    1 | harry |  1300.00 | 市场部    |
|    2 | amy   |  2200.00 | 人事部    |
```

```
|    3 | tom   |   600.00 | 财务部    |
|    4 | jack  |  3300.00 | 市场部    |
|    8 | aaa   | 12345.23 |           |
+------+-------+----------+-----------+
5 rows in set (0.00 sec)
```

更改：

```
mysql> delete from db01.t1 where id=8;
Query OK, 1 row affected (0.01 sec)

mysql> insert into db01.t1 values(5,'aaa',11111.23,'安保'),(6,'bbb',22222.22,'运维');
Query OK, 2 rows affected (0.00 sec)
Records: 2  Duplicates: 0  Warnings: 0

mysql> select * from db01.t1;
+------+-------+----------+-----------+
| id   | name  | salary   | dept      |
+------+-------+----------+-----------+
|    1 | harry |  1300.00 | 市场部    |
|    2 | amy   |  2200.00 | 人事部    |
|    3 | tom   |   600.00 | 财务部    |
|    4 | jack  |  3300.00 | 市场部    |
|    5 | aaa   | 11111.23 | 安保      |
|    6 | bbb   | 22222.22 | 运维      |
+------+-------+----------+-----------+
6 rows in set (0.00 sec)
```

3. 备份二进制日志文件

方法1：直接拷贝物理文件

方法2：备份改变过的数据（SQL）

```
# mysqlbinlog --start-position=120 mybinlog.000002 > /opt/mybinlog.sql
```

4. 模拟故障测试验证

1）在数据库层面删除所有的库

```
mysql > drop database mysql;
mysql > drop database myblog;
mysql > drop database db01;
```

2）通过刚刚全量备份进行恢复

```
# mysql < /tmp/backup/all.sql -p
```

结果：

```
mysql> select * from t1;
+------+-------+----------+-----------+
| id   | name  | salary   | dept      |
+------+-------+----------+-----------+
|    1 | harry |  1300.00 | 市场部    |
|    2 | amy   |  2200.00 | 人事部    |
|    3 | tom   |   600.00 | 财务部    |
|    4 | jack  |  3300.00 | 市场部    |
|    8 | aaa   | 12345.23 |           |
+------+-------+----------+-----------+
```

说明：以上数据不是最新数据

结合二进制日志进行最新数据的恢复

```
# mysqlbinlog --start-position=120 /opt/mybinlog.000002 |mysql -p456
```

1.备份

```
# mysqldump -p  --flush-logs --master-data=2 --all-databases > /tmp/all.sql
--flush-logs      //备份时先将内存中日志写回磁盘，然后截断日志,并产生新的日志文件
```

查看完整备份文件中的字段

```
# vim /tmp/all.sql
-- CHANGE MASTER TO MASTER_LOG_FILE='mysqld-bin.000003', MASTER_LOG_POS=120;
```

2、更新相关数据

3.还原

1）全库恢复

2）增量恢复

```
# at 199
#160822  6:16:14 server id 1  end_log_pos 307 CRC32 0x9f22d696  Query   thread_id=6
exec_time=0      error_code=0
use `db01`/*!*/;
SET TIMESTAMP=1471817774/*!*/;
insert into t2 set id=6,name='tom'
......
# at 307
#160822  16:16:14 server id 1  end_log_pos 338 CRC32 0xb69e555d  Xid = 16541
COMMIT/*!*/;
# at 338 误操作点
#160822  16:16:33 server id 1  end_log_pos 430 CRC32 0xa371739a  Query   thread_id=6
exec_time=0      error_code=0
SET TIMESTAMP=1471817793/*!*/;
drop database db01

# mysqlbinlog --start-position=120 --stop-position=338 mysqld-bin.000003|mysql -S
/mysql56/mysql56.sock -p
Enter password:
```

课堂练习1：

使用mysqldump备份整个数据库，做一些更新后在新的机器上进行恢复

1、备份整个数据库
2、备份之后，在原有的数据库上进行一些数据更新（可以直接在论坛上发帖模拟）
3、在同一台机器上重新再跑一个全新的数据库（模拟新的机器）
4、在这个全新的数据库上进行恢复
由于恢复的是整库，包括了数据库的授权信息，要让授权信息生效，必须刷新
mysql> flush privileges;
强烈建议重启数据库，这样肯定是应用了恢复后的数据进行启动
5、结合二进制让数据库恢复到最新的状态

3. 逻辑导入导出

```
select  xxx  into outfile '/path/file' from table_name;
```

无论是什么存储引擎，本身是一种数据导出的方法，同时可以用来辅助备份，它可以对一个表的其中一列或者某几列做备份
逻辑数据导入（恢复）：
方法一：

```
load data local infile '/path/file'  into  table table_name;
```

说明：该方法要求在编译数据库时要加上--enable-local-infile参数才可以使用

```
+---------------+-------+
| Variable_name | Value |
+---------------+-------+
| local_infile  | ON    |
+---------------+-------+
```

方法二：

```
# mysqlimport   dbname /path/file
```

它就是一个load data local infile的一个功能的打包实现

demo1：把db01库的emp表导出成文本，然后误删掉数据后进行恢复

```
mysql> select * into outfile '/tmp/backup/emp.bak' from emp;
ERROR 1290 (HY000): The MySQL server is running with the --secure-file-priv option so it cannot execute this statement
```

原因：没有给目录授权
解决：修改配置文件加入以下内容，重启数据库

```
secure_file_priv=/tmp/backup/
```

```
mysql> select * into outfile '/tmp/backup/emp.bak' from emp;
ERROR 1 (HY000): Can't create/write to file '/tmp/backup/emp.bak' (Errcode: 13 - Permission denied)
```

原因：mysql用户没有权限在/tmp/backup目录里创建文件
解决办法：

```
[root@mysql01 ~]# ll -d /tmp/backup/
drwxr-xr-x 2 root root 22 Aug  3 16:35 /tmp/backup/
[root@mysql01 ~]# setfacl -m u:mysql:rwx /tmp/backup/
```

demo2：创建一个表，把你系统里的/etc/passwd导入到数据库
1、创建表password

```
CREATE TABLE `password` (
  `uname` varchar(50) DEFAULT NULL,
  `pass` char(2) DEFAULT NULL,
```

```
  `uid` int(11) DEFAULT NULL,
  `gid` int(11) DEFAULT NULL,
  `comment` varchar(60) DEFAULT NULL,
  `home` varchar(50) DEFAULT NULL,
  `shell` varchar(50) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4

2、创建password.txt文件(文件内容必须是\t分割的多列)并导入到数据库
方法一：使用sed或者awk处理成新文本
方法二：直接用mysqlimport指定分隔符
# mysqlimport db01 --fields-terminated-by=':' --lines-terminated-by='\n' /tmp/mysqlbak/password
-p

\n 代表linux系统回车键的行结束符号
windows默认为\r\n
```

课堂练习2:

需求：把用户登录系统的信息存放到数据库里 要求如下显示： 用户名 登录终端 来源IP

```
# last|grep -vE 'reboot|^$|wtmp|tty1'|awk -F'[ ]+' '{print $1"\t"$2"\t"$3}'
```

三、MySQL的物理备份

1. lvm-snapshot备份

优点:

- 几乎是热备(创建快照前把表上锁，创建完后立即释放)
- 支持所有存储引擎
- 备份速度快
- 无需使用昂贵的商业软件(它是操作系统级别的)

缺点:

- 可能需要跨部门协调(使用操作系统级别的命令，DBA一般没权限)
- 数据如果分布在多个卷上比较麻烦(针对存储级别而言)

备份流程:

1. flush table with read locak;
2. create snapshot;
3. show master status; show slave status;
4. unlock tables;
5. copy data from cow to backup
6. remove snapshot

1.1 环境准备:

```
centos 6.5 系统:
```

```
正常安装MySQL：
1. 安装系统
2. 准备LVM, 例如 /dev/vg_back/lv-mysql, mount /usr/local/mysql
3. 源码安装MySQL到 /usr/local/mysql

可选操作：  将现在的数据迁移到LVM
1. 准备lvm及文件系统
# lvcreate -L 2G -n lv-mysql vg_back
# mkfs.ext4 /dev/vg_back/lv-mysql

2. 将数据迁移到LVM
# service mysqld stop
# mount /dev/vg_back/lv-mysql /u01/              //临时挂载点
# rsync -va /usr/local/mysql/ /u01/                       //将MySQL原数据镜像到临时挂载点

# umount /u01/
# mount /dev/vg_back/lv-mysql /usr/local/mysql       //加入fstab开机挂载
# df -Th
/dev/mapper/vg_back-lv-mysql ext4  2.0G  274M  1.7G  15% /usr/local/mysql
# service mysqld start


lvextend xxx
resize2fs xxx     centos 6
xfs_growfs



xfs  ext4
```

1.2 手动基于LVM快照实现备份：

```
1. 加锁
mysql> flush table with read lock;

2.创建快照
# lvcreate -L 500M -s -n lv-mysql-snap /dev/vg_back/lv-mysql
# mysql -uroot -p123 -e 'show master status' > /backup/`date +%F`_position.txt
或者
mysql> show master status;
+------------------+----------+--------------+------------------+-------------------+
| File             | Position | Binlog_Do_DB | Binlog_Ignore_DB | Executed_Gtid_Set |
+------------------+----------+--------------+------------------+-------------------+
| mysqld-bin.00003 |     135 |              |                  |                   |
+------------------+----------+--------------+------------------+-------------------+

3. 释放锁
mysql> unlock tables;

4. 从快照中备份
```

```
# mount -o ro /dev/vg_back/lv-mysql-snap /u01/
# mkdir /backup/`date +%F`
# rsync -a /u01/ /backup/2015-07-02/

5. 移除快照
# umount /u01/
# lvremove -f /dev/vg_back/lv-mysql-snap


脚本  + Cron完成:
#!/bin/bash
#LVM backmysql...
back_dir=/backup/`date +%F`

[ -d $back_dir ] || mkdir -p $back_dir

mysql -uroot -p123 -e 'flush table with read lock'
lvcreate -L 500M -s -n lv-mysql-snap /dev/vg_back/lv-mysql
mysql -uroot -p123 -e 'show master status' |grep mysql > $back_dir/position.txt
mysql -uroot -p123 -e 'flush logs'
mysql -uroot -p123 -e 'unlock tables'

mount -o ro /dev/vg_back/lv-mysql-snap /u01

rsync -a /u01/ $back_dir

if [ $? -eq 0 ];then
         umount /u01/
         lvremove -f /dev/vg_back/lv-mysql-snap
fi
```

1.3 mylvmbackup工具基于LVM自动备份

功能：利用LVM快照实现物理备份，即LVM快照备份的自动版

```
安装perl模块
1. 在线安装
http://www.lenzg.net/mylvmbackup
它依赖于perl 模块，可用以下命令安装
perl  -MCPAN  -e  'install Config::IniFiles'

2. 离线安装
# rpm -ivh mylvmbackup-0.16-0.noarch.rpm
warning: mylvmbackup-0.16-0.noarch.rpm: Header V4 DSA/SHA1 Signature, key ID b27291f2: NOKEY
error: Failed dependencies:
   perl(Config::IniFiles) is needed by mylvmbackup-0.16-0.noarch
   perl(Date::Format) is needed by mylvmbackup-0.16-0.noarch
   perl(File::Copy::Recursive) is needed by mylvmbackup-0.16-0.noarch

解决:
# yum -y localinstall atrpms-77-1.noarch.rpm perl-File-Copy-Recursive-0.38-1.el6.rfx.noarch.rpm
perl-IO-stringy-2.110-1.2.el6.rfx.noarch.rpm perl-Config-IniFiles-2.56-1.el6.rf.noarch.rpm
```

安装mylvmbackup软件包

```
# yum -y install mylvmbackup-0.15-0.noarch.rpm  解决依赖关系perl-TimeDate

备份方法一：
# mylvmbackup  --user=root  --password=123  --host=localhost  --mycnf=/etc/my.cnf  --
vgname=vg_back  --lvname=lv-mysql  --backuptype=tar   --lvsize=100M  --backupdir=/backup

# tar xf backup-20140903_000236_mysql.tar.gz
# ls
backup                                backup-cnf-20150702_000236_mysql
backup-20150702_000236_mysql.tar.gz   backup-pos


备份方法二：
# vim /etc/mylvmbackup.conf
[mysql]                      #连接数据库配置
    user=root
    password=123456
    host=localhost
    port=3306
    socket=/tmp/mysql.sock
    mycnf=/etc/my.cnf
    [lvm]                            #LVM逻辑卷的配置
    vgname=vg_server       #卷组名称
    lvname=lv_mysql         #逻辑卷名称
    backuplv=mysql_snap  #快照卷名称
    lvsize=500M
    [fs]                             #文件系统配置
    xfs=0
    mountdir=/var/tmp/mylvmbackup/mnt/          #挂载目录
    backupdir=/backup       #备份目录，也可以备份到行程主机
    [misc]                           #定义备份选项
    backuptype=tar            #定义备份的类型
    backupretention=0
    prefix=backup              #定义备份文件名前缀
    suffix=_mysql               #定义备份文件名后缀
    tararg=cvf                   #定义tar参数，默认为cvf
    tarfilesuffix=.tar.gz       #定义备份文件后缀名格式
    datefmt=%Y%m%d_%H%M%S  #定义备份文件名时间戳格式
    keep_snapshot=0        #是否保留snaphot
    keep_mount=0           #是否卸载snaphot
    quiet=0                      #定义记录日志类型
    注释：其他配置保持输入即可

    然后直接执行mylvmbackup即可
```

2. xtrabackup备份

xtrabackup|innobackupex的优点：

1. 备份过程快速、可靠(因为是物理备份)；
2. 支持增量备份，更为灵活
3. 备份过程**不会打断正在执行的事务**；

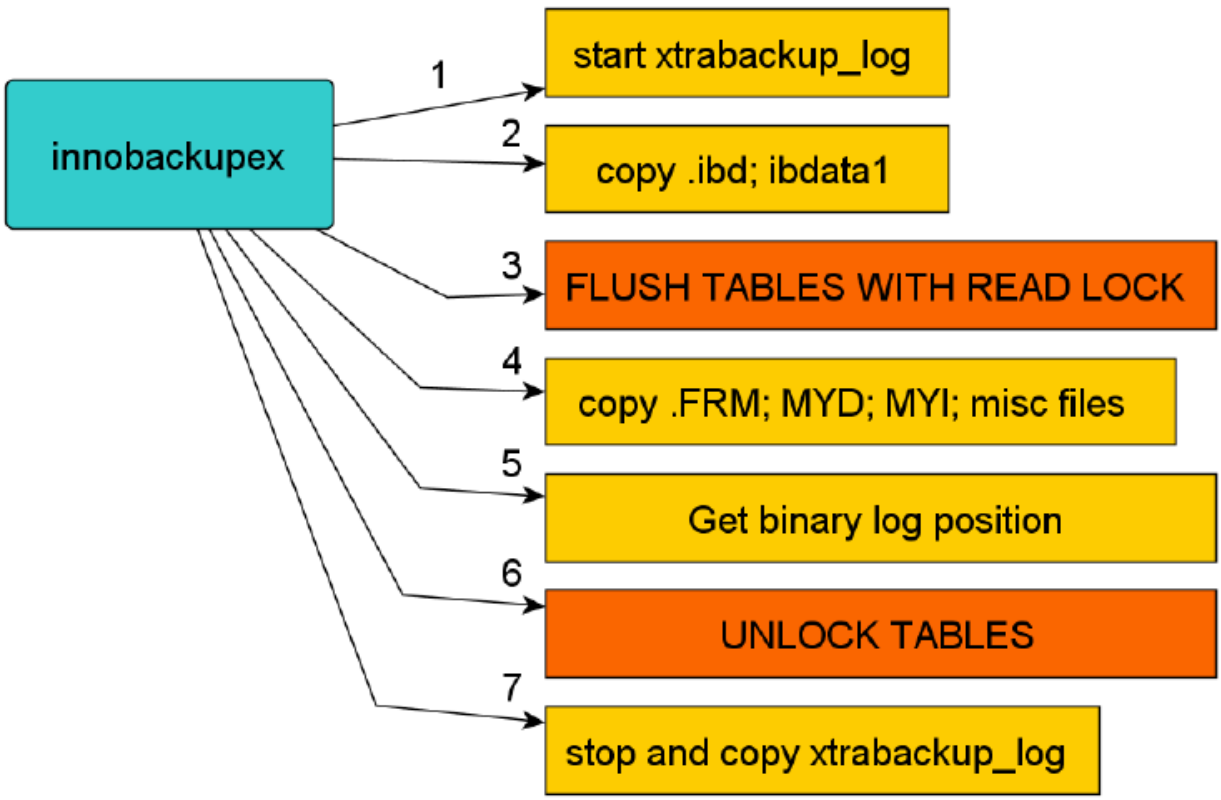
4. 能够基于压缩等功能节约磁盘空间和流量；
5. 自动实现备份检验；
6. 还原速度快；

缺点：

1. 只能对innodb表增量备份，myisam表增量备份时是全备
2. innobackupex备份MyISAM表之前要对全库进行加READ LOCK，阻塞写操作，若备份是在从库上进行的话会影响主从同步，造成延迟。对InnoDB表备份不会阻塞读写。

innobackupex全备原理：

```
1. innobackupex首先会启动一个xtrabackup_log后台检测的进程，实时检测mysql的redo log的变化，一旦发现redo有新的日志写入，立刻将日志写入到日志文件xtrabackup_log中。
2. 物理拷贝innodb的数据文件和系统表空间文件idbdata1到对应的以默认时间戳为备份目录的地方
3. 复制结束后，执行flush table with read lock操作进行全库锁表准备备份非InnoDB文件
4. 复制.frm .myd .myi等非InnoDB引擎文件
5. 查看binary log 的位置
6. 解锁unlock tables
7. 停止xtrabackup_log进程
```



图片来自网络

全备恢复原理

画图讲解

具体文字描述如下：

- 在InnoDB内部会维护一个redo日志文件，我们也可以叫做事务日志文件。事务日志会存储每一个InnoDB表数据的记录修改。当InnoDB启动时，InnoDB会检查数据文件和事务日志，并执行两个步骤：它应用（前滚）已经提交的事务日志到数据文件，并将修改过但没有提交的数据进行回滚操作。
- xtrabackup在启动时会记住log sequence number（LSN），并且复制所有的数据文件。复制过程需要一些时间，所以这期间如果数据文件有改动，那么将会使数据库处于一个不同的时间点。这时，xtrabackup会运行一个后台进程，用于监视事务日志，并从事务日志复制最新的修改。xtrabackup必须持续的做这个操作，是因为事务日志是会轮转重复的写入，并且事务日志可以被重用。所以xtrabackup自启动开始，就不停的将事务日志中每个数据文件的修改都记录下来。
- 上面就是xtrabackup的备份过程。接下来是准备（prepare）过程。在这个过程中，xtrabackup使用之前复制的事务日志，对各个数据文件执行灾难恢复（就像MySQL刚启动时要做的一样）。当这个过程结束后，数据库就可以做恢复还原了。
- 以上的过程在xtrabackup的编译二进制程序中实现。程序innobackupex可以允许我们备份MyISAM表和frm文件从而增加了便捷和功能。Innobackupex会启动xtrabackup，直到xtrabackup复制数据文件后，然后执行FLUSH TABLES WITH READ LOCK来阻止新的写入进来并把MyISAM表数据刷到硬盘上，之后复制MyISAM数据文件，最后释放锁。
- 备份MyISAM和InnoDB表最终会处于一致，在准备（prepare）过程结束后，InnoDB表数据已经前滚到整个备份结束的点，而不是回滚到xtrabackup刚开始时的点。这个时间点与执行FLUSH TABLES WITH READ LOCK的时间点相同，所以MyISAM表数据与InnoDB表数据是同步的。类似Oracle的recover和restore，InnoDB的prepare过程可以称为recover（恢复），MyISAM的数据复制过程可以称为restore（还原）。
- Mysql5.7.3以后开启二进制日志需要加上server-id选项，不然报错

实例1：使用innobackupex+binlog实现增量备份并恢复演练

思路：
1.inobackupex全备
2.inobackupex预备
3.修改数据，备份binlog日志 （增量备份）
4.删除数据（模拟故障）
5.全备恢复
6.恢复binlog 增量备份

安装innobackupex工具：

```
# yum -y install libev-4.15-3.el7.x86_64.rpm
# yum -y install percona-xtrabackup-24-2.4.7-1.el7.x86_64.rpm

[root@mysql01 RPM]# rpm -ql percona-xtrabackup-24
/usr/bin/innobackupex
/usr/bin/xtrabackup
/usr/share/doc/percona-xtrabackup-24-2.4.7
/usr/share/doc/percona-xtrabackup-24-2.4.7/COPYING
/usr/share/man/man1/innobackupex.1.gz
/usr/share/man/man1/xtrabackup.1.gz
```

步骤：
建立myisam存储引擎和innodb存储引擎的表测试：
开启bin-log日志：

```
mysql> create database db01;
```

```
mysql> use db01
mysql> create table t1(id int,name varchar(10)) engine=myisam;
mysql> insert into t1 values(1,'mona');
mysql> create table t2(id int,name varchar(10)) engine=innodb;
mysql> insert into t2 values(2,'tom');
```

创建一个备份用户admin并授予相应的权限：

注意：如果不使用root账号备份
需要的权限：
连接到服务是为了执行备份，需要在/backup上有read，write和execute权限。

在数据库中需要以下权限：
RELOAD和LOCK TABLES权限:为了执行FLUSH TABLES WITH READ LOCK
REPLICATION CLIENT权限：为了获取binary log位置
PROCESS权限：显示有关在服务器中执行的线程的信息（即有关会话执行的语句的信息），允许使用SHOW ENGINE

```
mysql> grant reload,process,lock tables,replication client on *.* to 'admin'@'localhost'
identified by '123';
mysql> flush privileges;
```

完全备份：
1、创建一个备份目录【可选项】
```
mkdir /full_xtrabak
```
2、执行完全备份
```
[root@db01 ~]# innobackupex --user=admin --password=123 /full_xtrabckup
```

如果报以下错误：
```
Error: failed to execute query SHOW ENGINE INNODB STATUS: Access denied; you need (at least one
of) the PROCESS privilege(s) for this operation
```
原因：缺少process权限

备份成功后显示：
```
MySQL binlog position: filename 'master.000001', position '120'
160909 13:15:07 [00] Writing backup-my.cnf
160909 13:15:07 [00]        ...done
160909 13:15:07 [00] Writing xtrabackup_info
160909 13:15:07 [00]        ...done
xtrabackup: Transaction log of lsn (1644102) to (1644102) was copied.
160909 13:15:07 completed OK!
```

查看备份结果：
```
[root@db01 2018-10-12_23-23-57]# pwd
/full_xtrabckup/2018-10-12_23-23-57
[root@db01 2018-10-12_23-23-57]# ls
backup-my.cnf  ibdata1  mysql              test                    xtrabackup_checkpoints
 xtrabackup_logfile
db01           myblog   performance_schema  xtrabackup_binlog_info  xtrabackup_info
```

了解如下文件：
xtrabackup_checkpoints:

备份类型（如完全或增量）、备份状态（如是否已经为prepared状态）和LSN(日志序列号)范围信息；每个InnoDB页(通常为16k大小)都会包含一个日志序列号，即LSN。
LSN是整个数据库系统的系统版本号，每个页面相关的LSN能够表明此页面最近是如何发生改变的。
xtrabackup_binlog_info:
mysql服务器当前正在使用的二进制日志文件及至备份这一刻为止二进制日志事件的位置。
xtrabackup_info:
xtrabackup工具在备份时记录的使用工具及数据库信息
backup-my.cnf：备份命令用到的配置选项信息
xtrabackup_logfile：xtrabackup记录innodb事物日志的信息

准备完全备份（预备）：
一般情况下，在备份完成后，数据尚且不能用于恢复操作，因为备份的数据中可能会包含尚未提交的事务或已经提交但尚未同步至数据文件中的事务。
因此，此时数据文件仍处于不一致状态。“准备”的主要作用正是通过回滚未提交的事务及同步已经提交的事务至数据文件也使得数据文件处于一致性状态。

--apply-log选项来应用日志，实现以上功能，如下：

```
[root@db01 ~]# innobackupex --user=admin --password=123 --apply-log /full_xtrabckup/2018-10-12_23-23-57/
```

准备完全备份成功后提示信息：

```
InnoDB: Starting shutdown...
InnoDB: Shutdown completed; log sequence number 2613288
170713 23:45:16 completed OK!
```

说明：
在实现“准备”的过程中，innobackupex通常还可以使用--use-memory选项来指定其可以使用的内存的大小，默认通常为100M。如果有足够的内存可用，可以多划分一些内存给prepare的过程，以提高其完成速度。

对完全备份后的数据库更改，进行二进制日志增量备份：
1、查看完全备份时日志的位置

```
[root@db01 2018-10-12_23-23-57]# cat xtrabackup_binlog_info
mybinlog.000005   12229
```

2、更改数据库信息

```
mysql> select * from t2;
+------+------+
| id   | name |
+------+------+
|    2 | tom  |
+------+------+
1 row in set (0.00 sec)

mysql> insert into t2 values(1,'harry');
mysql> insert into t2 values(3,'jack');
mysql> select * from t2;
+------+-------+
| id   | name  |
+------+-------+
|    2 | tom   |
|    1 | harry |
|    3 | jack  |
+------+-------+
3 rows in set (0.00 sec)
```

3、增量备份二进制日志

```
# mysqlbinlog --start-position=12229 mybinlog.000005 > /backup/`date +%F`.sql
```

模拟数据库损坏：
删除所有的数据库文件

```
rm -rf /usr/local/mysql/data/*
```

注意：
innobackupex工具是物理拷贝文件，所以在恢复时不需要连接数据库，这样就不需要初始化数据库并启动服务。

通过备份进行还原：
1、还原完全备份

```
# innobackupex --defaults-file=/etc/my.cnf --copy-back /full_xtrabckup/2018-10-12_23-23-57/

180804 12:06:29 [01]        ...done
180804 12:06:29 [01] Copying ./xtrabackup_binlog_pos_innodb to
/usr/local/mysql/data/xtrabackup_binlog_pos_innodb
180804 12:06:29 [01]        ...done
180804 12:06:29 [01] Copying ./ibtmp1 to /usr/local/mysql/data/ibtmp1
180804 12:06:29 [01]        ...done
180804 12:06:29 completed OK!


# chown mysql. -R /usr/local/mysql/
# service mysql start
```

注：datadir必须是为空的，innobackupex –copy-back不会覆盖已存在的文件，除非指定--force-non-empty-directories参数；
还要注意，还原时需要先关闭服务，如果服务是启动的，那么就不能还原到datadir

```
mysql> select * from t2;
+------+------+
| id   | name |
+------+------+
|    2 | tom  |
+------+------+
1 row in set (0.00 sec)
```

2、还原增量备份(结合二进制日志文件进行数据的恢复)

```
mysql> source /xtrabak/2017-07-13_23-05-10/2017-07-14.sql
mysql> select * from t2;
+------+-------+
| id   | name  |
+------+-------+
|    2 | tom   |
|    1 | harry |
|    3 | jack  |
+------+-------+
```

实例2：通过innobackupex实现增量备份
说明：
需要注意的是，增量备份仅能应用于InnoDB或XtraDB表，对于MyISAM表而言，执行增量备份时其实进行的是完全备份。
语法：
innobackupex --incremental /data/backups --incremental-basedir=BASEDIR
注意：
1、BASEDIR指的是完全备份所在的目录，此命令执行结束后，innobackupex命令会在/data/backups目录中创建一个新的以时间命名的目录以存放所有的增量备份数据。
2、在执行过增量备份之后再一次进行增量备份时，其--incremental-basedir应该指向上一次的增量备份所在的目录。

全量备份：
最开始数据：

```
mysql> select * from t1;
+------+------+
| id   | name |
+------+------+
|    1 | mona |
+------+------+
1 row in set (0.00 sec)

mysql> select * from t2;
+------+-------+
| id   | name  |
+------+-------+
|    2 | tom   |
|    1 | harry |
|    3 | jack  |
+------+-------+
3 rows in set (0.10 sec)

[root@db01 ~]# innobackupex --user=admin --password=123 /full_xtrabackup

[root@db01 2018-10-14_17-31-17]# cat xtrabackup_checkpoints
backup_type = full-backuped
from_lsn = 0
to_lsn = 2700237
last_lsn = 2700237
compact = 0
recover_binlog_info = 0
```

修改：

```
mysql> update t2 set name='lisi' where id=2;
mysql> insert into t2 values(4,'sarsh');
mysql> insert into t1 values(2,'fire');

mysql> select * from t1;
+------+------+
| id   | name |
+------+------+
```

```
|    1 | mona |
|    2 | fire |
+------+------+
2 rows in set (0.00 sec)

mysql> select * from t2;
+------+-------+
| id   | name  |
+------+-------+
|    2 | lisi  |
|    1 | harry |
|    3 | jack  |
|    4 | sarsh |
+------+-------+
4 rows in set (0.00 sec)
```

增量备份1:

```
[root@db01 ~]# innobackupex --user=admin --password=123 --incremental /incremental1_backup --
incremental-basedir=/full_xtrabackup/2018-10-14_17-31-17/

180804 14:49:44 [00] Writing xtrabackup_info
180804 14:49:44 [00]        ...done
xtrabackup: Transaction log of lsn (2183726) to (2183726) was copied.
180804 14:49:44 completed OK!


[root@db01 2018-10-14_17-36-35]# cat xtrabackup_checkpoints
backup_type = incremental        增量备份
from_lsn = 2700237
to_lsn = 2701750
last_lsn = 2701750
compact = 0
recover_binlog_info = 0

[root@db01 2018-10-14_17-36-35]# cat xtrabackup_binlog_info
mybinlog.000003   820
```

再次修改数据:

```
mysql> delete from t2 where id>3;
mysql> insert into t1 values(3,'jack');
mysql> create table t3 select * from t1;

mysql> select * from t1;
+------+------+
| id   | name |
+------+------+
|    1 | mona |
|    2 | fire |
|    3 | jack |
+------+------+
3 rows in set (0.00 sec)
```

```
mysql> select * from t2;
+------+-------+
| id   | name  |
+------+-------+
|    2 | lisi  |
|    1 | harry |
|    3 | jack  |
+------+-------+
3 rows in set (0.00 sec)
mysql> show tables;
+----------------+
| Tables_in_db01 |
+----------------+
| t1             |
| t2             |
| t3             |
+----------------+
3 rows in set (0.00 sec)
```

增量备份2:

```
[root@db01 incremental1_backup]# innobackupex --user=admin --password=123 --incremental
/incremental1_backup --incremental-basedir=/incremental1_backup/2018-10-14_17-36-35
```

查看当前二进制日志文件信息:

```
[root@db01 2018-10-14_17-41-36]# cat xtrabackup_binlog_info
mybinlog.000003   1399
```

说明:

/incremental1_backup/ 保存增量备份集目录
/incremental1_backup/2018-10-14_17-36-35 第一次增备的目录
/incremental1_backup/2018-10-14_17-41-36 第二次增备的目录

再次更改数据库:

```
mysql> create table t4 (id int(10));
Query OK, 0 rows affected (0.12 sec)

mysql> insert into t4 values(1);
Query OK, 1 row affected (0.02 sec)

mysql> select * from t4;
+------+
| id   |
+------+
|    1 |
+------+
1 row in set (0.00 sec)
```

预备增量备份:

预备增量备份需要2个步骤:

1.需要先预备全备，但是只重做已提交事务，不回滚未提交事务，然后应用到全备，也是只重做已提交事务，不回滚未提交事务

2.最后回滚未提交事务
注意：如果已经回滚了未提交事务，那么就无法再应用增量备份。

If you replay the committed transactions and rollback the uncommitted ones onthe base backup,
you will not be able to add the incremental ones. If you do this on an incremental one, you won't be able to add data from that moment and the remaining increments.
Having this in mind, the procedure is very straight-forward using the --redo-only option, starting with the base
backup:
innobackupex --apply-log --redo-only BASE-DIR

应用全备日志到备份集（全备）：
[root@db01 ~]# innobackupex --user=admin --password=123 --apply-log --redo-only /full_xtrabackup/2018-10-14_17-31-17/

应用增量备份1：
[root@db01 ~]# innobackupex --user=admin --password=123 --apply-log --redo-only /full_xtrabackup/2018-10-14_17-31-17/ --incremental-dir=/incremental1_backup/2018-10-14_17-36-35

应用增量备份2：
[root@db01 ~]# innobackupex --user=admin --password=123 --apply-log /full_xtrabackup/2018-10-14_17-31-17/ --incremental-dir=/incremental1_backup/2018-10-14_17-41-36

NOTE:
--redo-only should be used when merging all incrementals except the last one. That's why the previous line doesn't contain the --redo-only option. Even if the --redo-only was used on the last step, backup would still be consistent but in that case server would perform the rollback phase.

注：
1、--redo-only除了最后一个不用加之外，其他的增量应用都要加，最后一个应用的时候可以直接进入回滚未提交事务阶段。
如果加了也没事，服务启动的时候会进入recovery过程，来回滚
2、应用增量备份的时候只能按照备份的顺序来应用。如果应用顺序错误，那么备份就不可用。如果无法确定顺序，可以使用xtrabackup-checkpoints来确定顺序。

回滚未提交事务【可选项】：
[root@server ~]# innobackupex --apply-log /xtrabak/2017-07-15_07-46-49/ （完全备份的备份集）
InnoDB: Shutdown completed; log sequence number 2627131
170715 08:26:03 completed OK!

删除所有数据后进行恢复：

备份好二进制日志文件：
[root@db01 2018-10-14_17-41-36]# cat xtrabackup_binlog_info
mybinlog.000003 1399
[root@db01 data]# cp mybinlog.000003 /opt/

停止数据库：
```
service mysql stop
```
删除所有数据文件：
```
rm -rf /usr/local/mysql/data/*
```

在准备步骤完成后，还原时只需要还原完全备份即可
```
[root@db01 data]# innobackupex --defaults-file=/etc/my.cnf --copy-back /full_xtrabackup/2018-10-14_17-31-17/

[root@db01 data]# chown -R mysql.  /usr/local/mysql/data/

mysql> select * from t1;
+------+------+
| id   | name |
+------+------+
|    1 | mona |
|    2 | fire |
|    3 | jack |
+------+------+
3 rows in set (0.00 sec)

mysql> select * from t2;
+------+-------+
| id   | name  |
+------+-------+
|    2 | mona  |
|    1 | harry |
|    3 | jack  |
+------+-------+
3 rows in set (0.00 sec)

mysql> select * from t3;
+------+------+
| id   | name |
+------+------+
|    1 | mona |
|    2 | fire |
|    3 | jack |
+------+------+
3 rows in set (0.00 sec)

mysql>
```

最后一步：结合二进制日志文件将数据库恢复到最新状态

```
[root@db01 data]# mysqlbinlog /opt/mybinlog.000003 |less
[root@db01 data]# mysqlbinlog --start-position=1399 /opt/mybinlog.000003 |mysql -p
Enter password:
```
说明：没有指定结束pos值，表示恢复到最后。

总结

```
1.如果数据库在第2次增量备份后发生故障，那么数据恢复时只能够使用xtrabackup全量备份+xtrabackup增量备份恢复到第2次增量的点
2.如果要将数据库恢复到最新状态，需要结合binlog日志恢复
3.以上全量和增量的备份集是不能用的，需要将xtrabackup_log应用到全量的备份集中才有效
4. 应用所有日志到全量备份集中时，需注意：
   1). 除了最后一次增量备份应用日志可以不加--redo-only外,其他都要加;
       只应用已经提交的事务，不回滚未提交的事务!!!
   2). 应用日志到全量备份集中时一定要严格按照时间顺序执行，否则无效!
```

课后实战

根据任务要求完成MySQL数据库的备份。

1. 备份策略制定
2. 备份工具选择
3. 脚本编写
4. 计划任务执行