第1章 备份恢复

1.备份恢复的职责

```
1.备份、恢复策略的设计。
   备份周期、备份工具、备份方式、恢复方式全部流程化
2.日常备份检查
   日志、备份内容
3.定期的恢复演练
4.数据故障时，利用现有的资源，快速恢复
5.数据迁移、升级。
```

第2章 备份工具介绍

1.逻辑备份

```
mysqldump / source   *****
mysqlbinlog / source
mydumper / myloader
select into outfile / load data infile
binlog2sql
myflashback
```

2.物理备份

```
Percona Xtrabackup (PXB,XBK) *****
```

3.选型

```
100G 以内:  逻辑
100G 以上:  物理
```

第3章 mysqldump工具使用

1.介绍

```
mdp数据逻辑备份工具。(Create database\   create  table \ insert)
MySQL 自带的客户端命令。可以实现远程和本地备份。
```

2.参数

2.1 连接参数

```
-u
-p
-S
-h
-P
```

2.2 备份参数

```
# -A  全备
mkdir /data/backup
mysqldump -uroot -p123 -A >/data/backup/full.sql

# -B  单库或多库
mysqldump -uroot -p123 -B world gtdb test >/data/backup/db.sql

```

```
# 备份单表或多表
mysqldump -uroot -p123 world t1 country >/data/backup/tab.sql

# --master-data=2
1.自动记录备份时的binlog信息（注释）
2.自动锁定所有表，自动解锁（global read lock）。最好配合--single-transaction参数，减少锁表时间。
mysqldump -uroot -p123 -A --master-data=2 >/data/backup/full.sql

# --single-transaction
对于InnoDB表，开启独立事务，通过快照备份表数据，不锁表备份，可以理解为热备。
mysqldump -uroot -p123  -A  --master-data=2  --single-transaction >/data/backup/full.sql

# --max_allowed_packet=64M 最大允许的数据包大小
mysqldump -uroot -p123  -A  --master-data=2  --single-transaction  --max_allowed_packet=64M  >/data/backup/full.sql

# -R  -E  --triggers 备份特殊对象使用
mysqldump -uroot -p123  -A  --master-data=2  --single-transaction  --max_allowed_packet=64M  -R -E --triggers >/data/backup/full.sql

# 按日期备份定义文件名
mysqldump -uroot -p  -A  --master-data=2  --single-transaction  --max_allowed_packet=64M  -R -E  --triggers >/data/backup/full_`date +%F`.sql
```

3.故障恢复演练(mysqldump+binlog)

3.1 模拟环境

```
create database mdp charset utf8mb4;
use mdp
create table t1(id int);
insert into t1 values(1),(2),(3);
commit;
```

3.2 模拟周一23:00 全备

备份命令：

```
mysqldump -uroot -p  -A  --master-data=2  --single-transaction  --max_allowed_packet=64M  -R -E  --triggers >/data/backup/full_`date +%F`.sql
```

查看GTID相关信息，GTID截取起点

```
SET @@GLOBAL.GTID_PURGED='9b52b744-eb82-11ea-986c-000c294983f8:1-6';
```

查看pos号，备份开始时binlog位置点信息

```
-- CHANGE MASTER TO MASTER_LOG_FILE='mysql-bin.000006', MASTER_LOG_POS=1507;
```

3.3 模拟周二白天数据变化

```
use mdp;
create table t2 (id int);
insert into t2 values(1),(2),(3);
commit;
```

3.4 模拟周二下午2点，误删除了mdb核心库

```
mysql> drop database mdp;
```

3.5 恢复数据

a.恢复全备到周一晚23:00

```
# 检查全备:
vim /data/backup/full_2020-09-14.sql

# 查看 GTID相关信息 : GTID截取起点。
SET @@GLOBAL.GTID_PURGED='9b52b744-eb82-11ea-986c-000c294983f8:1-6';

# 查看pos号，备份开始时binlog位置点信息。
-- CHANGE MASTER TO MASTER_LOG_FILE='mysql-bin.000006',
MASTER_LOG_POS=1507;
```

b.截取日志

```
# 起点:
mysql-bin.000006 9b52b744-eb82-11ea-986c-000c294983f8:7 或者 mysql-bin.000006 pos=1507

# 终点:  找到drop事件
[root@db-51 ~]#  mysql -uroot -p123456 -e "show binlog events in 'mysql-bin.000006'"|grep -B 1 "drop database mdp"
mysql: [Warning] Using a password on the command line interface can be insecure.
mysql-bin.000006        1929    Gtid    6       1994    SET @@SESSION.GTID_NEXT= '9b52b744-eb82-11ea-986c-000c294983f8:9'
mysql-bin.000006        1994    Query   6       2083    drop database mdp

# 截取日志
[root@db-51 ~]# mysqlbinlog --skip-gtids --include-gtids='9b52b744-eb82-11ea-986c-000c294983f8:7-8' /data/mysql_3306/logs/mysql-bin.000006 >/data/backup/bin.sql
```

c.恢复

```
set sql_log_bin=0;
source /data/backup/full_2020-09-14.sql
source /data/backup/bin.sql
set sql_log_bin=1;
```

d.检查数据

```
use mdp
show tables;
select * from t1;
select * from t2;
```

4.mysqldump多种备份策略和恢复策略介绍

4.1 场景

```
100G 全库数据 全库备份 30分钟-40分钟，恢复整库需要5倍时间2.5-3小时之间
一张表 1G 被误删除了。
```

4.2 备份策略

a. mdp full+ binlog 增量备份思路

```
1.提取full全备中的故障表数据 ， 恢复数据
# sed -e'/./{H;$!d;}' -e 'x;/CREATE TABLE `t1`/!d;q'
full.sql>createtable.sql
# grep -i 'INSERT INTO `t1`'  full.sql >data.sql
2.binlog中截取全备到误删除t1之间对于这张表的修改
```

b.单库单表备份+binlog 增量思路

```
1.恢复单表的备份
2.binlog中截取备份到误删除t1之间对于这张表的修改
```

4.3 模拟故障

a.模拟原始数据

```
create database oldboy charset utf8mb4;
use oldboy;
create table oldguo (id int);
insert into oldguo values(1),(2),(3);
commit;
```

b.周一晚上全库备份

```
mysqldump -uroot -p  -A  --master-data=2  --single-transaction  --max_allowed_packet=64M  -R -E  --triggers >/data/backup/full.sql
```

c.模拟周二白天的数据变化

```
use oldboy ;
insert into oldguo values(11),(22),(33);
commit;
create table  oldli(id int);
insert into oldli values(1),(2),(3);
commit;
insert into oldguo values(111),(222),(333);
commit;
```

d.模拟周二下午2点，误删除数据库

```
drop table  oldguo;
```

4.4 模拟恢复

a.处理全备

```
[root@db-51 ~]# sed -n '/CREATE TABLE `oldguo` /,/\;/p' /data/backup/full.sql >/data/backup/create.sql
[root@db-51 ~]# grep -i 'INSERT INTO `oldguo`'  /data/backup/full.sql >/data/backup/insert.sql
```

b.binlog的截取

起点：9b52b744-eb82-11ea-986c-000c294983f8:13

```
SET @@GLOBAL.GTID_PURGED='9b52b744-eb82-11ea-986c-000c294983f8:1-12';
```

终点：9b52b744-eb82-11ea-986c-000c294983f8:16

```
[root@db-51 ~]# mysql -uroot -p123456 -e "show binlog events in 'mysql-bin.000007'" |grep -B 1 'DROP TABLE\ `oldguo`'
mysql: [Warning] Using a password on the command line interface can be insecure.
mysql-bin.000007        1799    Gtid    6       1864    SET @@SESSION.GTID_NEXT= '9b52b744-eb82-11ea-986c-000c294983f8:17'
mysql-bin.000007        1864    Query   6       1987    use `oldboy`; DROP TABLE `oldguo` /* generated by server */
```

gtid范围：

```
[root@db-51 ~]# mysqlbinlog --include-gtids='9b52b744-eb82-11ea-986c-000c294983f8:13-16' /data/mysql_3306/logs/mysql-bin.000007 |grep -B 16 'oldguo'|grep "GTID_NEXT"
SET @@SESSION.GTID_NEXT= '9b52b744-eb82-11ea-986c-000c294983f8:13'/*!*/;
SET @@SESSION.GTID_NEXT= '9b52b744-eb82-11ea-986c-000c294983f8:16'/*!*/;
```

截取方法1：

```
mysqlbinlog --skip-gtids --include-gtids='9b52b744-eb82-11ea-986c-000c294983f8:13-16' --exclude-gtids='9b52b744-eb82-11ea-986c-000c294983f8:14-15' /data/mysql_3306/logs/mysql-bin.000007 >/data/backup/bin.sql
```

截取方法2:

```
mysqlbinlog --skip-gtids --include-gtids='9b52b744-eb82-11ea-986c-000c294983f8:13','9b52b744-eb82-11ea-986c-000c294983f8:16' /data/mysql_3306/logs/mysql-bin.000007 >/data/backup/bin1.sql
```

c.恢复数据

```
use oldboy;
set sql_log_bin=0;
source /data/backup/create.sql
source /data/backup/insert.sql
commit;
source /data/backup/bin.sql
set sql_log_bin=1;
```

5.mysqldump实现单库单表备份

设置安全导出文件：

```
[root@db-51 ~]# vim /etc/my.cnf
[mysqld]
secure_file_priv=/tmp

[root@db-51 ~]# systemctl restart mysqld
```

构造备份语句脚本：

```
[root@db-51 ~]# mkdir -p /data/backup/single_bak
[root@db-51 ~]# mysql -uroot -p123
mysql> select concat("mysqldump -uroot -p123 -A --master-data=2 --
single-transaction --max_allowed_packet=64M -R -E --triggers
",table_schema," ",table_name,"
>/data/backup/single_bak/",table_schema,"_",table_name,".sql")
from information_schema.tables
where table_schema not in
('sys','information_schema','performance_schema')
into outfile '/tmp/single_bak.sh';
[root@db-51 ~]# sh /tmp/single_bak.sh &>/tmp/bak.log
```

第4章 Xtrabackup工具使用

1.介绍

```
percona公司研发
xtrabackup  --》C  C++
innobackupex  --》perl语言
8.0之前，2.4.x
8.0之后，8.0
物理备份工具，类似于cp文件。支持：全备和增量备份
```

2.安装

2.1 安装依赖包

```
wget -O /etc/yum.repos.d/epel.repo http://mirrors.aliyun.com/repo/epel-7.repo
yum -y install perl perl-devel libaio libaio-devel perl-Time-HiRes perl-DBD-MySQL libev
```

2.2 下载软件并安装

```
wget https://www.percona.com/downloads/XtraBackup/Percona-XtraBackup-2.4.12/binary/redhat/7/x86_64/percona-xtrabackup-24-2.4.12-1.el7.x86_64.rpm
yum -y install percona-xtrabackup-24-2.4.4-1.el7.x86_64.rpm
```

3.全备

3.1 介绍

```
拷贝/data/mysql_3306/data/下的数据文件。
InnoDB  : 热备。拷贝ibdataN,UNDO00N ,ibtmpN ,ibd 。通过截取变化redo。
非InnoDB: FTWRL, 全局锁。拷贝非INNODB的文件frm\myi\myd\...
只能本地备份。
```

3.2 实现全备

修改配置文件并重启

```
[root@db-51 ~]# vim /etc/my.cnf
[client]
socket=/tmp/mysql.sock
[root@db-51 ~]# systemctl restart mysqld
```

全备命令:

```
innobackupex --user=root --password=123456 /data/backup/test
```

查看备份完成的目录:

```
[root@db-51 ~]# ll /data/backup/test/2020-09-14_22-06-11/
总用量 12348
-rw-r----- 1 root root      487 9月  14 22:06 backup-my.cnf
drwxr-x--- 2 root root       48 9月  14 22:06 gtdb
-rw-r----- 1 root root    10056 9月  14 22:06 ib_buffer_pool
-rw-r----- 1 root root 12582912 9月  14 22:06 ibdata1
drwxr-x--- 2 root root       52 9月  14 22:06 ku
drwxr-x--- 2 root root       52 9月  14 22:06 linux5
drwxr-x--- 2 root root       76 9月  14 22:06 mdp
drwxr-x--- 2 root root     4096 9月  14 22:06 mysql
drwxr-x--- 2 root root       90 9月  14 22:06 oldboy
drwxr-x--- 2 root root      134 9月  14 22:06 oldya
drwxr-x--- 2 root root     8192 9月  14 22:06 performance_schema
drwxr-x--- 2 root root      160 9月  14 22:06 school
```

```
drwxr-x--- 2 root root     8192 9月  14 22:06 sys
drwxr-x--- 2 root root       54 9月  14 22:06 test
drwxr-x--- 2 root root      144 9月  14 22:06 world
-rw-r----- 1 root root       63 9月  14 22:06 xtrabackup_binlog_info
-rw-r----- 1 root root      117 9月  14 22:06 xtrabackup_checkpoints
-rw-r----- 1 root root      546 9月  14 22:06 xtrabackup_info
-rw-r----- 1 root root     2560 9月  14 22:06 xtrabackup_logfile
```

目录文件介绍:

```
1.xtrabackup_binlog_info
记录binlog位置点， 截取binlog起点位置

2.xtrabackup_checkpoints
from_lsn = 0         # 一般增量备份会关注，一般上次备份的to_lsn的位置
to_lsn = 180881595   # CKPT-LSN 最近的内存数据落地到磁盘上的LSN号
last_lsn = 180881604 # xtrabackup_logfile LSN

3.xtrabackup_info
备份总览信息

4.xtrabackup_logfile
备份期间产生的redo变化
```

自定义备份目录

```
innobackupex --user=root --password=123 --no-timestamp
/data/backup/xbk/full_`date +%F`
```

3.3 全备恢复应用

模拟删除

```
pkill mysqld
rm -rf /data/mysql_3306/*
```

使用全备恢复数据

a.prepare 准备备份阶段

```
innobackupex --apply-log /data/backup/xbk/full
```

b.copy-back 恢复

方法1:

```
cp -a /data/backup/test/2020-09-14_22-06-11/* /data/mysql_3306/
mkdir /data/mysql_3306/logs/
touch /data/mysql_3306/logs/mysql.err
chown -R mysql.mysql /data/*
```

方法2:

```
innobackupex --copy-back /data/backup/test/2020-09-14_22-06-11/
innobackupex --move-back /data/backup/test/2020-09-14_22-06-11/
```

4.增量备份功能

4.1 介绍

```
自带的功能。
每次增量一般是将最近一次备份作为参照物。
自动读取参照物cat xtrabackup_checkpoints中to_lsn值，与当前CKPT的LSN对比，备份变化过page。
备份期间新的数据变化，通过redo自动备份。
恢复数据时，需要把所有需要的增量合并到FULL中。无法通过增量单独恢复数据，依赖与全备。
```

4.2 增量备份演练（FULL(周日)+inc1(周一)+inc2(周二)+inc3(周三)）

1.备份前数据准备：

```
create database xbk charset utf8mb4;
use xbk
create table full (id int);
insert into full values(1),(2),(3);
commit;
```

2.模拟周日 23:00 全备

```
innobackupex --user=root --password=123 --no-timestamp
/data/backup/full_`date +%F`
```

3.模拟周一白天数据变化

```
use xbk
create table inc1 (id int);
insert into  inc1 values(1),(2),(3);
commit;
```

4.模拟周一23:00增量备份

```
innobackupex --user=root --password=123  --no-timestamp   --incremental
--incremental-basedir=/data/backup/full_2020-09-15
/data/backup/inc1_`date +%F`
```

5.模拟周二白天数据变化

```
use xbk
create table inc2 (id int);
insert into  inc2 values(1),(2),(3);
commit;
```

6.模拟周二23:00增量备份

```
innobackupex --user=root --password=123  --no-timestamp   --incremental
--incremental-basedir=/data/backup/inc1_2020-09-15
/data/backup/inc2_`date +%F`
```

7.模拟周三白天数据变化

```
use xbk
create table inc3(id int);
insert into  inc3 values(1),(2),(3);
commit;
```

8.模拟周三23:00增量备份

```
innobackupex --user=root --password=123  --no-timestamp   --incremental
--incremental-basedir=/data/backup/inc2_2020-09-15
/data/backup/inc3_`date +%F`
```

9.模拟周四白天的数据变化。

```
use xbk
create table inc4(id int);
insert into  inc4 values(1),(2),(3);
commit;
```

10.周四下午出现数据损坏。如何恢复到误删除之前。

```
pkill mysqld
rm -rf /data/mysql_3306/*
```

11.恢复思路

```
1.我们有什么?
备份:
full+inc1+inc2+inc3
binlog:
full以来全量的binlog

2.处理备份
需要将inc1\inc2\inc3按顺序依次合并到全备，并进行prepare.
从官方角度：基础全备和合并所有增量（排除最后一个）都需要此参数
原理角度： 使所有备份合并时，LSN必须是连续的
```

12.处理base_full

```
innobackupex --apply-log --redo-only  /data/backup/full_2020-09-15/
```

13.inc1合并到full中，并且prepare

```
cd /data/backup/
innobackupex --apply-log --redo-only  --incremental-dir=inc1_2020-09-15
full_2020-09-15
```

检验合并结果

```
cat full_2020-09-15/xtrabackup_checkpoints |grep "to_lsn"
cat inc1_2020-09-15/xtrabackup_checkpoints |grep "to_lsn"
```

14.inc2合并到full中，并且prepare

```
cd /data/backup/
innobackupex --apply-log --redo-only  --incremental-dir=inc2_2020-09-15 full_2020-09-15
```

检验合并结果

```
cat full_2020-09-15/xtrabackup_checkpoints |grep "to_lsn"
cat inc2_2020-09-15/xtrabackup_checkpoints |grep "to_lsn"
```

15.inc3合并到full中，并且prepare

```
cd /data/backup/
innobackupex --apply-log --redo-only  --incremental-dir=inc3_2020-09-15 full_2020-09-15
```

检验合并结果

```
cat full_2020-09-15/xtrabackup_checkpoints |grep "to_lsn"
cat inc3_2020-09-15/xtrabackup_checkpoints |grep "to_lsn"
```

16.将合并后全备再次prepare

```
innobackupex --apply-log  /data/backup/full_2020-09-15
```

17.恢复并启动

```
innobackupex --copy-back /data/backup/full_2020-09-15
mkdir /data/mysql_3306/logs/
touch /data/mysql_3306/logs/mysql.err
chown -R mysql.mysql /data/*
systemctl restart mysqld
```

18.截取周三增量备份后到故障前所有的binlog日志并进行恢复

前提条件：binlog没有被删掉，如果binlog和数据目录放在一起，刚才的操作就把binlog也一起删掉了。

起点：2029

```
[root@db-51 ~]# cat /data/backup/inc3_2020-09-15/xtrabackup_binlog_info
mysql-bin.000001        2029    9b52b744-eb82-11ea-986c-000c294983f8:1-17,
cb0fd847-f6e3-11ea-af80-000c294983f8:1-9
```

终点：binlog 结尾 2423

```
mysql -uroot -p123456 -e "show binlog events in 'mysql-bin.000001';"
```

19.截取命令

```
mysqlbinlog --skip-gtids --start-position=2029  /binlog/mysql-bin.000001 >/data/backup/bin.sql
```

20.恢复binlog

```
mysql -uroot -p123
mysql> set sql_log_bin=0;
mysql> source /data/backup/bin.sql
mysql> set sql_log_bin=1;
```

21.xbk恢复完成后，清空所有日志

```
mysql> reset master;
```

21.立即再做个全备

```
1  innobackupex --user=root --password=123456 --no-timestamp
   /data/backup/full_bak_`date +%F`
```