MySQL 备份恢复操作记录

目 录

1. 制定说明

仅实验。

1.1.术语说明

本规范所用到的术语均为数据库领域相关的内容，如下：

冷备：在数据库实例关闭、数据库读写服务不可用的情况下备份。

温备：对数据库施加读锁的情况下备份。

热备：在数据库读写服务都可用的情况下备份。

物理备份：拷贝归档数据文件以此作为备份。

逻辑备份：通过数据库生成数据关联信息作为备份。

全量备份：对数据的某时刻一致性拷贝作为备份。

增量备份：基于某个备份的状态，变化的数据提取出作为备份。

2. 环境说明

2.1. 版本说明

操作系统版本	CentOS Linux release 7.5.1804
数据库版本	5.7.31-log MySQL Community Server(GPL)

2.2. 目录说明

/home/mysqldata	MySQL 服务器数据目录
/home/mysqlbinlog	MySQL 服务器二进制文件目录
/home/backups	备份存储目录

2.3. 用户说明

Dbuser	Linux 服务器操作用户
Mysql	MySQL 服务器进程启动用户
Rep	用于备份数据的 MySQL 用户
dmp	用于备份数据的 MySQL 用户

注：本次测试使用 dbuser 用户，具有 sudo 权限。

3. 物理文件备份

物理文件备份旨在快速高效地备份数据，不能跨操作系统恢复。

3.1. 热备工具 Percona XtraBackup

XtraBackup 是 Percona 公司针对 MySQL、MariaDB、Percona Server for MySQL 开发的数据备份恢复工具，遵循 GPL 协议，备份速度快且并不阻塞 InnoDB 引擎的事务。

3.1.1 工具安装

推荐使用 yum 安装，该方式支持以下 Linux 操作系统版本：

CentOS 5 and *RHEL* 5
CentOS 6 and *RHEL* 6(Current Stable)
CentOS 7 and *RHEL* 7

1. 安装 Percona 的 yum 软件源：

```
[dbuser@VM_0_7_centos ~]# wget https://repo.percona.com/yum/percona-release-latest.noarch.rpm
//从 percona 源网站下载最新的 yum 源的安装包
[dbuser@VM_0_7_centos ~]# sudo yum install percona-release-latest.noarch.rpm
//用 yum 安装或者如下用 rpm 安装
[dbuser@VM_0_7_centos ~]# sudo rpm -ivh percona-release-latest.noarch.rpm
[dbuser@VM_0_7_centos ~]# yum list | grep percona
percona-release.noarch                1.0.25              installed
…
percona-xtrabackup.x86_64             2.3.10-1.el7        percona-release-x86_64
percona-xtrabackup-24.x86_64          2.4.20-1.el7        percona-release-x86_64
…
qpress.x86_64                         11-1.el7            percona-release-x86_64
//上述列出众多 percona 公司开发的相关软件包可以安装,包括 2.3 和 2.4 版本的 xtrabackup
```

2. 安装 xtrabackup2.4 和 qpress：

```
[dbuser@VM_0_7_centos ~]#sudo yum install percona-xtrabackup-24.x86_64
[dbuser@VM_0_7_centos ~]# sudo yum deplist percona-xtrabackup-24.x86_64
//yum 列出安装 xtrabackup 所需的基础依赖，当使用 rpm 安装时可供参考。
```

安装依赖如下表：

软件包	版本
bash.x86_64	4.2.46-34.el7
libaio.x86_64	0.3.109-13.el7
glibc.x86_64	2.17-307.el7.1
openssl-libs.x86_64	1:1.0.2k-19.el7
libcurl.x86_64	7.29.0-57.el7_8.1
libev.x86_64	4.15-7.el7

libgcc.x86_64	4.8.5-39.el7
libgcrypt.x86_64	1.5.3-14.el7
libgpg-error.x86_64	1.12-3.el7
libstdc++.x86_64	4.8.5-39.el7
zlib.x86_64	1.2.7-18.el7
perl-DBD-MySQL.x86_64	4.023-6.el7
perl-Digest-MD5.x86_64	2.52-3.el7
rsync.x86_64	3.1.2-10.el7
glibc.i686	2.17-307.el7.1

3.内网 rpm 安装方式：获取 xtrabackup 的 rpm 安装包，然后逐个解决所需的安装依赖：

```
[dbuser@VM_0_7_centos~]#wget
https://www.percona.com/downloads/XtraBackup/Percona-XtraBackup-2.4.4/binary/redhat/7/x86_64/percona-xtrabackup-24-2.4.4-1.el7.x86_64.rpm
[dbuser@VM_0_7_centos ~]# sudo rpm -ivh percona-xtrabackup-24-2.4.4-1.el7.x86_64.rpm
```

4.推荐使用二进制包方式：获取二进制包即可使用。

3.1.2 MySQL 用户最小权限

在正式环境中，MySQL 备份恢复应创建仅用于备份的专属数据库用户。xtrabackup 备份数据所需的最小数据库权限如下所示：

RELOAD、*LOCK TABLES*、*PROCESS*、*REPLICATION CLIENT*

在数据库中执行创建用户授权命令：

```
mysql> GREATE USER rep@'localhost' IDENTIFIED BY '!qazxsw@';
mysql> GRANT RELOAD, LOCK TABLES, PROCESS, REPLICATION CLIENT ON *.* TO rep@'localhost';
mysql> FLUSH PRIVILEGES;
```

3.1.3 本地全量备份

用 xtrabackup 创建本地全量备份，命令格式如下：

xtrabackup --backup --target-dir=[备份存储目录]

```
[dbuser@VM_0_7_centos ~]# xtrabackup --defaults-file=/etc/my.cnf --user=rep --password='!qazxsw@'
--port=3306 --backup --target-dir=/home/backups/backup_2020102
201022 16:23:53 Executing FLUSH NO_WRITE_TO_BINLOG ENGINE LOGS...
xtrabackup: The latest check point (for incremental): '24005125566'
xtrabackup: Stopping log copying thread.
.201022 16:23:53 >> log scanned up to (24005125575)
201022 16:23:53 Executing UNLOCK TABLES
201022 16:23:53 All tables unlocked
201022 16:23:53 [00] Copying ib_buffer_pool to /home/backups/backup_20201022/ib_buffer_pool
201022 16:23:53 [00]            ...done
201022 16:23:53 Backup created in directory '/home/backups/backup_20201022'
MySQL binlog position: filename 'mysql-bin.000011', position '15065623', GTID of the last change
'54a6c06d-f891-11ea-ad5b-525400136b97:1-5,
88aa525f-fc74-11ea-87a0-525400136b97:1-95,
8c7c177b-fbe4-11ea-ba33-525400136b97:1-3'
201022 16:23:53 [00] Writing backup-my.cnf
201022 16:23:53 [00]            ...done
201022 16:23:53 [00] Writing xtrabackup_info
201022 16:23:53 [00]            ...done
xtrabackup: Transaction log of lsn (24005125566) to (24005125575) was copied.
201022 16:23:53 completed OK!
```

//最后一行显示 completed OK！，说明备份完成

注:从 2.3 版本开始将 innobackupex 合并入 xtrabackup，故而非 InnoDB 引擎表也会备份。

检查备份是否成功：

```
[dbuser@VM_0_7_centos~]#cd /home/backups/backup_20201022
[dbuser@VM_0_7_centos~]#cat xtrabackup_checkpoints xtrabackup_info
backup_type = full-backuped
from_lsn = 0
to_lsn = 24005125566
last_lsn = 24005125575
compact = 0
recover_binlog_info = 0
uuid = ebbe9398-143f-11eb-8961-525400136b97
name =
tool_name = xtrabackup
tool_command = --defaults-file=/etc/my.cnf --user=rep --password=... --backup
--target-dir=/home/backups/backup_20201022
tool_version = 2.4.4
ibbackup_version = 2.4.4
server_version = 5.7.31-log
start_time = 2020-10-22 16:23:31
end_time = 2020-10-22 16:23:53
lock_time = 0
binlog_pos = filename 'mysql-bin.000011', position '15065623', GTID of the last change
```

```
'54a6c06d-f891-11ea-ad5b-525400136b97:1-5,
88aa525f-fc74-11ea-87a0-525400136b97:1-95,
8c7c177b-fbe4-11ea-ba33-525400136b97:1-3'
innodb_from_lsn = 0
innodb_to_lsn = 24005125566
partial = N
incremental = N
format = file
compact = N
compressed = N
encrypted = N
```

//可以查看 backup_type、to_lsn 两个指标。backup_type=full-backuped 表明全备完成；to_lsn 表示当前数据文件 checkpoint 点在 24005125566；

执行命令参数说明如下：

--defaults-file	指定 MySQL 服务器配置文件，默认/etc/my.cnf
--user	指定备份用户的名称
--password	指定备份用户的密码
--port	指定 MySQL 服务器端口，默认 3306

注：命令中若加入 port 参数，则以此为准，否则以配置文件的端口为准，若无配置端口，则默认 3306。

3.1.4 本地增量备份

本地增量备份原理是从备份数据的 to_lsn 值开始，收集 lsn 值大于 to_lsn 值的数据页作为增量备份。

首先需要准备一个全量备份：

xtrabackup --backup --target-dir=[备份存储目录]

```
[dbuser@VM_0_7_centos ~]# xtrabackup --defaults-file=/etc/my.cnf --user=rep --password='!qazxsw@'
--port=3306 --backup --target-dir=/home/backups/backup_20201023_base
[dbuser@VM_0_7_centos ~]# cd /home/backups/backup_20201023_base
[dbuser@VM_0_7_centos backup_20201023_base]# cat xtrabackup_checkpoints
backup_type = full-backuped
from_lsn = 0
to_lsn = 24005128478
last_lsn = 24005128487
compact = 0
recover_binlog_info = 0
```

//从 from_lsn 和 to_lsn 观察，此次全备数据的 lsn 范围在 0-24005128478

其次开启一个线程不断插入 100w 数据：

```
[dbuser@VM_0_7_centos ~]# mysql –u –p
mysql> desc t_user;
+---------------+-------------+------+-----+---------+----------------+
```

```
| Field          | Type         | Null | Key | Default | Extra          |
+--------------+-------------+------+-----+---------+----------------+
| id             | int(11)      | NO   | PRI | NULL    | auto_increment |
| c_user_id      | varchar(36) | NO   |     |         |                |
| c_name         | varchar(22) | NO   |     |         |                |
| c_province_id | int(11)      | NO   |     | NULL    |                |
| c_city_id      | int(11)      | NO   | MUL | NULL    |                |
| create_time    | datetime     | NO   |     | NULL    |                |
+--------------+-------------+------+-----+---------+----------------+
6 rows in set (0.00 sec)

mysql>insert into t_user select id, uuid(), CONCAT('user', id), FLOOR(Rand() * 1000), FLOOR(Rand()
* 100), NOW() FROM tmp_table where id < 1000000;
Query OK, 999999 rows affected (36.43 sec)
```

//插入数据的同时开启第一次增量备份

第一次增量备份：

xtrabackup --backup --target-dir=[增量备份存储目录] --incremental-basedir=[所基于的备份数据目录]

```
[dbuser@VM_0_7_centos ~]# xtrabackup --defaults-file=/etc/my.cnf --user=rep --password='!qazxsw@'
--port=3306 --backup --target-dir=/home/backups/backup_20201023_inc1
--incremental-basedir=/home/backups/backup_20201023_base
[dbuser@VM_0_7_centos ~]# cd /home/backups/backup_20201023_inc1
[dbuser@VM_0_7_centos backup_20201023_inc1]# cat xtrabackup_checkpoints
backup_type = incremental
from_lsn = 24005128478
to_lsn = 26519008580
last_lsn = 26563610923
compact = 0
recover_binlog_info = 0
```

// backup_type 显示为 incremental，表明是增量备份，从 from_lsn 和 to_lsn 观察，此次增备数据的 lsn 范围在 24005128478-26519008580。（当进行远程增量备份时，将 incremental-basedir 参数改为 incremental-lsn，指定起始 lsn 即某个备份 to_lsn 值，可以执行增量备份。）

//增量备份过程伴随着数据插入事务的进行一直进行备份，直到该事务终结，同时还有部分数据在拷贝数据文件时存于重做日志中，表现为 to_lsn = 26519008580，last_lsn = 26563610923 之间的差值。

再开启一个线程不断插入 100w 数据：

```
[dbuser@VM_0_7_centos ~]# mysql –u –p
mysql>insert into t_user select id, uuid(), CONCAT('user', id), FLOOR(Rand() * 1000), FLOOR(Rand()
* 100), NOW() FROM tmp_table where id between 1000000 and 2000000;
Query OK,1000001 rows affected (25.68 sec)
```

//插入数据后再开启第二次增量备份

第二次增量备份：

xtrabackup --backup --target-dir=[增量备份存储目录] --incremental-basedir=[所基于的备份数据目录]

```
[dbuser@VM_0_7_centos ~]# xtrabackup --defaults-file=/etc/my.cnf --user=rep --password='!qazxsw@'
--port=3306 --backup --target-dir=/home/backups/backup_20201023_inc2
--incremental-basedir=/home/backups/backup_20201023_inc1
[dbuser@VM_0_7_centos ~]# cd /home/backups/backup_20201023_inc2
[dbuser@VM_0_7_centos backup_20201023_inc2]# cat xtrabackup_checkpoints
backup_type = incremental
from_lsn = 26519008580
to_lsn = 26713358553
last_lsn = 26713358562
compact = 0
recover_binlog_info = 0
// backup_type 显示为 incremental，表明是增量备份，从 from_lsn 和 to_lsn 观察，此次增备数据的 lsn 范围在 26519008580-26713358553。
//to_lsn 和 last_lsn 实际相差 8 个字节，而 lsn 本身占用 8 个字节，说明存在新开的一个 redo 数据页，但里面不包含数据。
```

3.1.5 流式备份

Percona XtraBackup 支持流式备份，将备份以指定 tar 或 xbstream 格式发送到 STDOUT，允许其他程序进行过滤，生成 tar 或者 xbstream 格式文件。优点是能够远程备份到其他主机，在本机空间不充足时非常适用，同时能实现数据压缩，传输过程加密。

本地全量流备份，命令如下：

xtrabackup --backup --stream=xbstream > base.xbstream

```
[dbuser@VM_0_7_centos ~]# mkdir /home/backups/streamback_20201023_base/
[dbuser@VM_0_7_centos ~]# xtrabackup --defaults-file=/etc/my.cnf –user=rep –password=--backup
–-stream=xbstream > /home/backups/streamback_20201023_base/base.xbstream
//备份出来的文件为 xbstream 格式，需要进行解压
[dbuser@VM_0_7_centos ~]# cd /home/backups/streamback_20201023_base
[dbuser@VM_0_7_centos streamback_20201023_base]# xbstream –x < base.xbsream
```

本地增量流备份，命令如下：

```
[dbuser@VM_0_7_centos ~]# mkdir /home/backups/streamback_20201023_inc1/
[dbuser@VM_0_7_centos ~]# xtrabackup --defaults-file=/etc/my.cnf --user=rep --password='!qazxsw@'
--port=3306 --backup –-stream=xbstream
–-incremental-basedir=/home/backups/streamback_20201023_base > \
/home/backups/streamback_20201023_inc1/inc.xbstream
//备份出来的文件为 xbstream 格式，需要进行解压
[dbuser@VM_0_7_centos ~]# cd /home/backups/streamback_20201023_inc1
[dbuser@VM_0_7_centos streamback_20201023_inc1]# xbstream –x < inc.xbsream
```

增量流备份到远程，命令如下：

```
[dbuser@VM_0_7_centos ~]# xtrabackup --defaults-file=/etc/my.cnf --user=rep --password='!qazxsw@'
--port=3306 --backup –-stream=xbstream
–-incremental-basedir=/home/backups/streamback_20201023_base | ssh dbuser@*** "xbstream –x –C
/dir"
//流备份通过 ssh 传输到远程服务器并解压到指定的存在目录，实现备份到远程需要两个条件：一是用户设置免密，二是指定目录必须存在。
```

3.1.6 性能参数优化

下面指出 xtrabackup 工具针对于提升性能的部分参数：

--parallel=NUMER-OF-THREADS	在备份和恢复表文件过程中使用多线程，线程数最多不超过 ibd 文件数量。
--rsync	加速本地文件传输，但不能与--stream 同时使用。
--compress	利用 CPU 使用 quicklz 算法压缩数据，一是节省空间，二是以便在远程流传输中更高效。
--compress-threads	压缩线程数。
--throttle=NUMBER-OF-10MB/S	尽管备份过程不阻塞 InnoDB 引擎的事务，但是会增加机器的 io 负载，该参数可以限制 io 使用，默认不限制。

压缩备份时，数据文件的格式均为.qp，需要使用 qpress 工具解压缩，安装步骤如下：

```
[dbuser@VM_0_7_centos ~]# yum list | grep percona
percona-release.noarch                1.0.25              installed
…
percona-xtrabackup.x86_64             2.3.10-1.el7        percona-release-x86_64
percona-xtrabackup-24.x86_64          2.4.20-1.el7        percona-release-x86_64
…
qpress.x86_64                         11-1.el7            percona-release-x86_64
//qpress 包存在于 percona 源中
[dbuser@VM_0_7_centos ~]# sudo yum install qpress
//假设 compressbackup 目录为压缩备份的数据目录
[dbuser@VM_0_7_centos ~]# cd ./compressbackup
[dbuser@VM_0_7_centos compressbackup]# for bf in `find . -iname "*\.qp"`; do qpress -d $bf $(dirname
$bf) && rm $bf; done
//解压并询问是否删除.qp 文件
```

3.2.冷备 MySQL 数据文件

此法为 MySQL 正常关闭，将存于内存中的数据刷入磁盘，拷贝数据文件。

3.2.1 关闭 MySQL 服务器

临时设置 set global innodb_fast_shutdown=0，以 slow shutdown 方式关停数据库，将一些变更缓冲刷入磁盘，同时要确保 errorlog 无错误出现。

```
[dbuser@VM_0_7_centos ~]# mysql -u -p
mysql> show global variables like 'innodb_fast_shutdown';
+----------------------+-------+
| Variable_name        | Value |
+----------------------+-------+
| innodb_fast_shutdown | 1     |
+----------------------+-------+
1 row in set (0.00 sec)

mysql> set global innodb_fast_shutdown=0;
Query OK, 0 rows affected (0.00 sec)

mysql> shutdown;
Query OK, 0 rows affected (0.00 sec)

mysql> exit
Bye
[dbuser@VM_0_7_centos ~]# sudo grep 'ERROR' /var/log/error.log
//错误日志在配置文件中设置为/var/log/error.log，检查是否有错误发生。
```

3.2.2 备份 InnoDB 数据和日志文件

数据文件包括 ibdata 共享表空间、.ibd 后缀的各个表空间.frm 表结构文件，将其拷贝到备份服务器留作备份，将存储重做日志的 ib_logfile0、ib_logfile1 等多个文件拷贝到备份服务器留作备份。

3.2.3 备份 MySQL 配置文件

配置文件作为 MySQL 启动的参数，同样拷贝到备份服务器。

4. 逻辑文件备份

4.1.mysqldump

mysqldump 是 MySQL 自带的一款单线程逻辑备份工具，可以通过 TCP 或者 SOCKET 连接备份。对于 MyISAM 引擎而言，在备份时对表加读锁，其他线程无法同时修改数据，属于温备工具。对于 InnoDB 引擎而言，参数--single-transaction 设置能在备份时设置事务隔离级别为可重复读，利用多版本并发机制不阻塞其他事务修改数据。mysqldump 除了能够备份出 SQL 语句，还能输出 csv、txt、XML 等格式。

4.1.1 工具安装

MySQL 工具包自带。

4.1.2 MySQL 用户最小权限

用户备份时需要最小权限为 SELECT、PROCESS，备份触发器、视图、事件等需要额外对应的权限。

```
mysql> GRANT SELECT,PROCESS ON *.* TO dmp@'localhost' identified by '!qazxsw@';
```

4.1.3 备份

全部数据库备份命令如下：

mysqldump -u -p -h -P --all-databases > alldb.sql

```
[dbuser@VM_0_7_centos ~]# mysqldump -udmp -p'!qazxsw@' -h127.0.0.1 -P3306
--set-gtid-purged=OFF --single-transaction --all-databases > alldb20201023.sql
```

//--set-gtid-purged

此为 MySQL5.6 引入 GTID 机制后的参数，OFF 关闭参数意味着，备份出的逻辑文件不包含 set gtid-purged = 'GTID 事务集'语句；默认 on 开启，意味着备份文件执行先关闭 binlog，purge 事务集，再导入数据，最后开启 binlog。当不涉及主从关系恢复时，选择 OFF。当恢复到从库时，可以选择 ON，更为方便地恢复主从关系。

//-- single-transaction

该参数在备份时设置可重复读事物隔离级别，利用多版本机制，不阻塞 Innodb 引擎存储的表，与之相反，--lock-tables 锁住所有表。

指定数据库备份命令如下：

mysqldump -u -p -h -P --databases db1 db2 > db1db2.sql

```
[dbuser@VM_0_7_centos ~]# mysqldump -udmp -p'!qazxsw@' -h127.0.0.1 -P3306
--set-gtid-purged=OFF --single-transaction --databases dtadb > dtadb.sql
```

指定表备份命令如下：

mysqldump -u -p -h -P --databases db1 --tables tb1> tb1023.sql

```
[dbuser@VM_0_7_centos ~]# mysqldump -udmp -p'!qazxsw@' -h127.0.0.1 -P3306
--set-gtid-purged=OFF --single-transaction --databases dtadb --tables t_user > t_user20201023.sql
```

表结构备份：

mysqldump -u -p -h -P --no-data --databases db1 > dbstrc1023.sql

```
[dbuser@VM_0_7_centos ~]# mysqldump -udmp -p'!qazxsw@' -h127.0.0.1 -P3306
--set-gtid-purged=OFF --single-transaction --no-data --databases dtadb> tbstruct20201023.sql
[dbuser@VM_0_7_centos ~]# cat tbstruct20201023.sql
[dbuser@VM_0_7_centos backups]# cat dtastruc.sql
-- MySQL dump 10.13  Distrib 5.7.31, for Linux (x86_64)
--
-- Host: 127.0.0.1       Database: dtadb
-- ------------------------------------------------------
-- Server version        5.7.31-log
```

```
/*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
/*!40101 SET @OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS */;
/*!40101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION */;
/*!40101 SET NAMES utf8 */;
/*!40103 SET @OLD_TIME_ZONE=@@TIME_ZONE */;
/*!40103 SET TIME_ZONE='+00:00' */;
/*!40014 SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0 */;
/*!40014 SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0
*/;
/*!40101 SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='NO_AUTO_VALUE_ON_ZERO' */;
/*!40111 SET @OLD_SQL_NOTES=@@SQL_NOTES, SQL_NOTES=0 */;

--
-- Current Database: `dtadb`
--

CREATE DATABASE /*!32312 IF NOT EXISTS*/ `dtadb` /*!40100 DEFAULT CHARACTER SET latin1
*/;

USE `dtadb`;

--
-- Table structure for table `t_user`
--
DROP TABLE IF EXISTS `t_user`;
/*!40101 SET @saved_cs_client      = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `t_user` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `c_user_id` varchar(36) NOT NULL DEFAULT '',
  `c_name` varchar(22) NOT NULL DEFAULT '',
  `c_province_id` int(11) NOT NULL,
  `c_city_id` int(11) NOT NULL,
  `create_time` datetime NOT NULL,
  PRIMARY KEY (`id`),
  KEY `idx1` (`c_city_id`,`c_province_id`)
) ENGINE=InnoDB AUTO_INCREMENT=2000001 DEFAULT CHARSET=utf8mb4;
/*!40101 SET character_set_client = @saved_cs_client */;
--
-- Table structure for table `tmp_table`
--

DROP TABLE IF EXISTS `tmp_table`;
```

```
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `tmp_table` (
  `id` int(11) NOT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
/*!40101 SET character_set_client = @saved_cs_client */;
/*!40103 SET TIME_ZONE=@OLD_TIME_ZONE */;

/*!40101 SET SQL_MODE=@OLD_SQL_MODE */;
/*!40014 SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS */;
/*!40014 SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS */;
/*!40101 SET CHARACTER_SET_CLIENT=@OLD_CHARACTER_SET_CLIENT */;
/*!40101 SET CHARACTER_SET_RESULTS=@OLD_CHARACTER_SET_RESULTS */;
/*!40101 SET COLLATION_CONNECTION=@OLD_COLLATION_CONNECTION */;
/*!40111 SET SQL_NOTES=@OLD_SQL_NOTES */;

-- Dump completed on 2020-10-23 20:05:08
```

//上述仅备份了 dtadb 库的所有表结构，不备份数据。

4.1.4 工具参数优化

下面是在特殊情况下可能需要用到的参数，如某些代理层管理的数据库需要指定列名。

--complete-insert,-c	备份出的插入语句带有完整的列名。
--max_allowed_packet	服务器发送和接收的最大包长度，适当提高以提高效率。
--master-data	指定主从集群中的主库进行备份，可选 1 或者 2,1 为最后执行 change master 连接主库操作，2 为不执行，备份用户需要 REPLICATION CLIENT 权限。

4.2.mydumper

mydumper 是针对 MySQL 的一款多线程逻辑备份开源工具，遵循 GPL 协议，除了能多线程备份恢复，同时还能压缩文件，在库或者表的临时数据迁移中比 mysqldump 效率更高，但是在备份开始会加全局只读锁，FLUSH TABLES WITH READ LOCK。

4.2.1 工具安装

该工具为 C 语言开发的开源软件，遵循 GPL-3.0 Licence，下面是从 github 上下载 rpm 包，并安装。

```
[dbuser@VM_0_7_centos ~]# sudo yum install
https://github.com/maxbube/mydumper/releases/download/v0.9.5/mydumper-0.9.5-2.el7.x86_64.rpm
```

4.2.2 MySQL 用户最小权限

所需备份最小权限为 SELECT、RELOAD，备份触发器、视图等需要额外对应的权限。

```
mysql> GRANT SELECT,RELOAD ON *.* TO dmp@'localhost' identified by '!qazxsw@';
```

4.2.3 备份

全库 16 线程备份(除去系统库)：

mydumper --regex=’^(?!(sys|mysql|test))’ -o /allbackup

```
[dbuser@VM_0_7_centos ~]# mydumper -u dmp -p '!qazxsw@' -h 127.0.0.1 -P 3306 --threads=16
--regex='^(?!(sys|mysql|test))' -o /home/backups/alluserdbs
[dbuser@VM_0_7_centos backups]# cd /home/backups/alluserdbs
[dbuser@VM_0_7_centos alluserdbs]# ls -lh
总用量 302M
-rw-r--r-- 1 dbuser dbuser    66 10 月  23 20:48 dtadb-schema-create.sql
-rw-r--r-- 1 dbuser dbuser   216 10 月  23 20:48 dtadb.tmp_table-schema.sql
-rw-r--r-- 1 dbuser dbuser 127M 10 月  23 20:49 dtadb.tmp_table.sql
-rw-r--r-- 1 dbuser dbuser   490 10 月  23 20:48 dtadb.t_user-schema.sql
-rw-r--r-- 1 dbuser dbuser 175M 10 月  23 20:49 dtadb.t_user.sql
-rw-r--r-- 1 dbuser dbuser    75 10 月  23 20:49 metadata
//--threads 指定线程数
//--regex 正则匹配，指定除去 sys、mysql、test 系统库的其他库
//-o 输出文件目录
```

注：每一个缩写参数与值之间必须严格空格，如-u[空格]dmp

备份正则匹配 adb 库中 t 开头的表：

mydumper -B adb --regex=’t.*’ -o /alluser

```
[dbuser@VM_0_7_centos ~]# mydumper -u dmp -p '!qazxsw@' -h 127.0.0.1 -P 3306 --threads=16 –B
dtadb --regex='t.*' -o /home/backups/alluser
//-B 指定需要备份的库
```

4.2.4 工具参数优化

mydumper 参数：

-B,--database	备份出的插入语句带有完整的列名。
-T,--tables-list	服务器发送和接收的最大包长度，适当提高以提高效率。
-c,--compress	压缩备份出来的文件
-x,--regex	正则匹配目标库名表名
-m,--no-schemas	仅备份结构
-d,--no-data	仅备份数据
--complete-insert	INSERT 语句完整填上字段名称
-t,--threads	使用线程数，默认 4

4.3. 二进制文件

MySQL 二进制文件记录着数据的逻辑信息，可用于复制、备份和数据回退。通常，可以以物理拷贝的方式备份二进制文件，同时 MySQL 自带工具包提供 mysqlbinlog 工具，自 MySQL5.6 之后，通过 Replication API 能实现实时远程备份二进制文件 binlog。事实上，利用此接口开发实时二进制读取过滤产生消息，在 ETL、缓存维护、收集数据更新指标、采样到搜索引擎、数据分区迁移、binlog 回滚、大数据应用等场景都有着较大的作用。

4.3.1 MySQL 用户最小权限

从 MySQL 服务器连接获取 MySQL 二进制文件的用户需要 Replication slave 权限。

```
mysql> GRANT Replication slave ON *.* TO dmp@'%' identified by '!qazxsw@';
```

4.3.2 备份

从远程服务器连接备份 MySQL 服务器的指定二进制日志：

mysqlbinlog --read-from-remote-server --raw mysql-bin.000001

```
[dbuser@VM_0_7_centos ~]# mysqlbinlog –udmp –p'!qazxsw@' --read-from-remote-server --raw
mysql-bin.000015
[dbuser@VM_0_7_centos ~]# ls -lh
 总用量 12K
 -rw-r----- 1 dbuser dbuser     9.0K 10 月  23 20:50 mysql-bin.000015
// --read-from-remote-server 表示从远程连接
//--raw 表示直接拷贝二进制文件不进行解析
```

从远程服务器连接备份 MySQL 服务器的所有二进制日志：

mysqlbinlog --read-from-remote-server --raw --stop-never mysql-bin.000001

```
[dbuser@VM_0_7_centos ~]# mysqlbinlog –udmp –p'!qazxsw@' --read-from-remote-server --raw
--stop-never mysql-bin.000015
//----dumping----
[dbuser@VM_0_7_centos ~]# ls -lh
 总用量 12K
 -rw-r----- 1 dbuser dbuser     9.0K 10 月  23 21:50 mysql-bin.000015
 -rw-r----- 1 dbuser dbuser     1.1K 10 月  23 21:50 mysql-bin.000016
// --stop-never 表示从指定二进制文件 mysql-bin.000015 开始，不间断备份二进制文件。
```

4.3.3 工具参数优化

备份过程参数：

--raw	备份原生二进制文件，不能与解析过滤参数共用。
-R,--read-from-remote-server	从远程服务器读取二进制日志
--stop-never	不间断备份二进制日志

解析过程参数：

-d,--database	二进制日志解析时，仅过滤指定数据库
--base64-output=decode-rows	decode-rows 将基于 row 模式的二进制日志解析成逻辑 SQL
--result-file=name,-r name	指定解析二进制日志后的目标存储文件
--start-datetime=datetime	二进制日志解析时，基于事件开始时间过滤
--stop-datetime=datetime	二进制日志解析时，基于事件结束时间过滤
--stop-never	不间断解析二进制日志
-v,-vv	二进制转换为逻辑 SQL 的详细程度，vv 更详细

5. 恢复

5.1 物理文件:本地全量恢复

利用备份数据文件去恢复 MySQL 服务器无需数据库用户的密码，但需要两个步骤：

第一个步骤为利用 crash recovery 重做日志应用，称作 PREPARE，命令如下：

xtrabackup --prepare --target-dir=[备份存储目录]

```
[dbuser@VM_0_7_centos ~]# xtrabackup --prepare --target-dir=/home/backups/ backup_20201022
xtrabackup version 2.4.4 based on MySQL server 5.7.13 Linux (x86_64) (revision id: df58cf2)
xtrabackup: cd to /home/backups/backup_20201022
xtrabackup: This target seems to be not prepared yet.
InnoDB: Number of pools: 1
xtrabackup: xtrabackup_logfile detected: size=8388608, start_lsn=(24005125566)
...
xtrabackup: Starting InnoDB instance for recovery.
xtrabackup: Using 104857600 bytes for buffer pool (set by --use-memory parameter)
InnoDB: Number of pools: 1
InnoDB: Using CPU crc32 instructions
InnoDB: Initializing buffer pool, total size = 100M, instances = 1, chunk size = 100M
InnoDB: page_cleaner coordinator priority: -20
...
InnoDB: Starting crash recovery.
...
201022 18:37:46 completed OK!
```

//屏幕输出日志中，从 start_lsn=24005125566 开始进行 redo 日志回滚，由于此次全备过程中并没有数据写入，故最终 lsn 仍为 24005125566。

注:此步骤无需连接 MySQL 服务器，xtrabackup 自带 InnoDB 回滚工具。

检查重做日志是否应用成功：

```
[dbuser@VM_0_7_centos backup_20201022]# cat xtrabackup_checkpoints xtrabackup_info
backup_type = full-prepared
from_lsn = 0
to_lsn = 24005125566
last_lsn = 24005125575
compact = 0
recover_binlog_info = 0
```

```
uuid = ebbe9398-143f-11eb-8961-525400136b97
name =
tool_name = xtrabackup
tool_command=--defaults-file=/etc/my.cnf --user=rep --password=... --backup
--target-dir=/home/backups/backup_20201022
tool_version = 2.4.4
ibbackup_version = 2.4.4
server_version = 5.7.31-log
start_time = 2020-10-22 16:23:31
end_time = 2020-10-22 16:23:53
lock_time = 0
binlog_pos = filename 'mysql-bin.000011', position '15065623', GTID of the last change
'54a6c06d-f891-11ea-ad5b-525400136b97:1-5,
88aa525f-fc74-11ea-87a0-525400136b97:1-95,
8c7c177b-fbe4-11ea-ba33-525400136b97:1-3'
innodb_from_lsn = 0
innodb_to_lsn = 24005125566
partial = N
incremental = N
format = file
compact = N
compressed = N
encrypted = N
```

//backup_type 显示为 full-prepared（重做日志应用完成），to_lsn 和 innodb_to_lsn 仍然为定位在 24005125566，说明重做日志中不存在数据更新，binlog_pos 指定当前数据 checkpoint 点。

第二个步骤为恢复数据，目的是替换 MySQL 服务器的数据目录，启动数据库前需要确保备份数据的读写权限归属在 MySQL 服务启动用户下，步骤如下：

1. 制造清空表误操作。

```
[dbuser@VM_0_7_centos ~]# mysql -u -p
mysql> select count(*) from dtadb.tmp_table;
+----------+
| count(*) |
+----------+
| 12000000 |
+----------+
1 row in set (2.94 sec)

mysql> truncate table dtadb.tmp_table;
Query OK, 0 rows affected (0.11 sec)

mysql> exit
Bye
```

2. 关闭目标数据库，移走数据目录。

```
[dbuser@VM_0_7_centos ~]# sudo systemctl stop mysqld
[dbuser@VM_0_7_centos ~]# sudo mv /home/mysqldata/ /home/mysqldata_wait_to_delete
```

3.执行下面命令，根据指定配置文件，将备份目录拷贝到 MySQL 数据库原先数据目录处。

xtrabackup --defaults-file=/etc/my.cnf --copy-back --target-dir=[备份存储目录]

```
[dbuser@VM_0_7_centos ~]# xtrabackup --defaults-file=/etc/my.cnf --copy-back
--target-dir=/home/backups/backup_20201022
xtrabackup version 2.4.4 based on MySQL server 5.7.13 Linux (x86_64) (revision id: df58cf2)
201022 19:11:11 [01] Copying ib_logfile0 to /home/mysqldata/ib_logfile0
201022 19:11:11 [01]            ...done
201022 19:11:11 [01] Copying ib_logfile1 to /home/mysqldata/ib_logfile1
201022 19:11:12 [01]            ...done
201022 19:11:12 [01] Copying ibdata1 to /home/mysqldata/ibdata1
201022 19:11:17 [01]            ...done
201022 19:11:17 [01] Copying ./dtadb/db.opt to /home/mysqldata/dtadb/db.opt
201022 19:11:17 [01]            ...done
201022 19:11:17 [01] Copying ./dtadb/tmp_table.frm to /home/mysqldata/dtadb/tmp_table.frm
201022 19:11:17 [01]            ...done
201022 19:11:17 [01] Copying ./dtadb/t_user.frm to /home/mysqldata/dtadb/t_user.frm
201022 19:11:17 [01]            ...done
201022 19:11:17 [01] Copying ./dtadb/t_user.ibd to /home/mysqldata/dtadb/t_user.ibd
201022 19:11:28 [01]            ...done
...
...
...
201022 19:11:32 [01] Copying ./mysql/proxies_priv.frm to
/home/mysqldata/mysql/proxies_priv.frm
201022 19:11:32 [01]            ...done
201022 19:11:33 [01] Copying ./xtrabackup_binlog_pos_innodb to
/home/mysqldata/xtrabackup_binlog_pos_innodb
201022 19:11:33 [01]            ...done
201022 19:11:33 completed OK!
```

//最后一行显示 completed OK!说明拷贝完成，同时已经改名为原先数据目录名称/home/mysqldata。

//实际上此步骤等价于,手动 copy 把 full-prepare 的备份文件拷到 mysql 数据目录下，并修改权限即可。

4.修改数据目录属主、属组为 mysql，并启动 MySQL。

```
[dbuser@VM_0_7_centos ~]# ls -lh /home/mysqldata/
```

```
总用量 705M
-rw-r----- 1 dbuser dbuser   425 10 月  22 16:23 backup-my.cnf
drwxr-x--- 2 dbuser dbuser 4.0K 10 月  22 16:23 dtadb
-rw-r----- 1 dbuser dbuser   16K 10 月  22 16:23 ib_buffer_pool
-rw-r----- 1 dbuser dbuser 588M 10 月  22 18:39 ibdata1
-rw-r----- 1 dbuser dbuser   48M 10 月  22 18:39 ib_logfile0
-rw-r----- 1 dbuser dbuser   48M 10 月  22 18:37 ib_logfile1
-rw-r----- 1 dbuser dbuser   12M 10 月  22 18:39 ibtmp1
drwxr-x--- 2 dbuser dbuser 4.0K 10 月  22 16:23 mysql
drwxr-x--- 2 dbuser dbuser 4.0K 10 月  22 16:23 performance_schema
drwxr-x--- 2 dbuser dbuser 4.0K 10 月  22 16:23 sys
drwxr-x--- 2 dbuser dbuser 4.0K 10 月  22 16:23 test
-rw-r----- 1 dbuser dbuser   152 10 月  22 16:23 xtrabackup_binlog_info
-rw-r--r-- 1 dbuser dbuser    26 10 月  22 18:39 xtrabackup_binlog_pos_innodb
-rw-r----- 1 dbuser dbuser   121 10 月  22 18:39 xtrabackup_checkpoints
-rw-r----- 1 dbuser dbuser   695 10 月  22 16:23 xtrabackup_info
-rw-r----- 1 dbuser dbuser 8.0M 10 月  22 18:37 xtrabackup_logfile
[dbuser@VM_0_7_centos ~]# sudo chown -R mysql:mysql /home/mysqldata
[dbuser@VM_0_7_centos ~]# sudo systemctl start mysqld
```

5. 检查数据是否恢复完成

```
[dbuser@VM_0_7_centos ~]# mysql –u -p
mysql> select count(*) from dtadb.tmp_table;
+----------+
| count(*) |
+----------+
| 12000000 |
+----------+
1 row in set (2.82 sec)

//统计 dtadb.tmp_table 显示 1200w 数据，同清表前一致。
```

性能参数优化:

--parallel=NUMER-OF-THREADS	在备份和恢复表文件过程中使用多线程，线程数最多不超过 ibd 文件数量。
--use-memory	在 PREPARE 阶段重做日志应用时起作用，默认 100MB，1-2G 更好。
--rsync	加速本地文件传输，但不能与--stream 同时使用。
--compress	利用 CPU 使用 quicklz 算法压缩数据，一是节省空间，二是以便在远程流传输中更高效。
--compress-threads	压缩线程数。
--throttle=NUMBER-OF-10MB/S	尽管备份过程不阻塞 InnoDB 引擎的事务，但是会增加机器的 io 负载，该参数可以限制 io 使用，默认不限制。

5.2 物理文件:本地增量恢复

假设数据需要恢复第二次增量备份之后的状态：

数据备份目录如下：

/home/backups/backup_20201023_base	数据全备目录
/home/backups/backup_20201023_inc1	第一次增量备份目录
/home/backups/backup_20201023_inc2	第二次增量备份目录

利用备份数据文件去本地恢复 MySQL 服务器无需数据库用户的密码，但需要两个步骤：

第一个步骤为利用 crash recovery 重做日志应用，称作 PREPARE，但是 PREPARE 过程中需要增加--apply-log-only 参数，避免跨备份之间的增量数据页的提交被重做日志应用机制回滚，造成数据不一致。

重做日志应用机制：未提交的事务进行回滚，已经提交的事务前滚。--apply-log-only 参数意义在于对未提交的事务不进行操作，已经提交的事务刷盘。在所有增量备份 PREPARE 之后，结合所有重做日志，回滚未提交的事务。

1. 全量备份 PREPARE

xtrabackup --prepare --apply-log-only --target-dir=[全备存储目录]

```
[dbuser@VM_0_7_centos ~]# xtrabackup --prepare --apply-log-only
--target-dir=/home/backups/backup_20201023_base
[dbuser@VM_0_7_centos ~]# cd /home/backups/backup_20201023_base
[dbuser@VM_0_7_centos backup_20201023_base]# cat xtrabackup_checkpoints xtrabackup_info
backup_type = log-applied
from_lsn = 0
to_lsn = 24005128478
last_lsn = 24005128487
compact = 0
recover_binlog_info = 0
uuid = e71cd973-14dc-11eb-b01b-525400136b97
name =
tool_name = xtrabackup
tool_command = --user=rep --password=... --backup
--target-dir=/home/backups/backup_20201023_base
tool_version = 2.4.4
ibbackup_version = 2.4.4
server_version = 5.7.31-log
start_time = 2020-10-23 11:07:15
end_time = 2020-10-23 11:07:36
lock_time = 0
binlog_pos = filename 'mysql-bin.000013', position '600', GTID of the last change
'54a6c06d-f891-11ea-ad5b-525400136b97:1-5,
88aa525f-fc74-11ea-87a0-525400136b97:1-97,
8c7c177b-fbe4-11ea-ba33-525400136b97:1-3,
```

```
be98892c-1457-11eb-8d96-525400136b97:1'
innodb_from_lsn = 0
innodb_to_lsn = 24005128478
partial = N
incremental = N
format = file
compact = N
compressed = N
encrypted = N
```

// backup_type 显示为 log-applied，日志应用过程中表明已提交的前滚，未提交的则跳过。

2. 第一次增量备份 PREPARE

xtrabackup --prepare --apply-log-only --target-dir=[全备存储目录] –-incremental-dir=[第一次增量备份存储目录]

```
[dbuser@VM_0_7_centos ~]# xtrabackup --prepare --apply-log-only
--target-dir=/home/backups/backup_20201023_base
--incremental-dir=/home/backups/backup_20201023_inc1
[dbuser@VM_0_7_centos ~]# cd /home/backups/backup_20201023_base
[dbuser@VM_0_7_centos backup_20201023_base]# cat xtrabackup_checkpoints xtrabackup_info
backup_type = log-applied
from_lsn = 0
to_lsn = 26519008580
last_lsn = 26563610923
compact = 0
recover_binlog_info = 0
uuid = cd0f7ec6-14e8-11eb-b01b-525400136b97
name =
tool_name = xtrabackup
tool_command = --user=rep --password=... --backup
--target-dir=/home/backups/backup_20201023_inc1
--incremental-basedir=/home/backups/backup_20201023_base
tool_version = 2.4.4
ibbackup_version = 2.4.4
server_version = 5.7.31-log
start_time = 2020-10-23 12:32:12
end_time = 2020-10-23 12:32:47
lock_time = 0
binlog_pos = filename 'mysql-bin.000013', position '66173880', GTID of the last change
'54a6c06d-f891-11ea-ad5b-525400136b97:1-5,
88aa525f-fc74-11ea-87a0-525400136b97:1-97,
8c7c177b-fbe4-11ea-ba33-525400136b97:1-3,
be98892c-1457-11eb-8d96-525400136b97:1-3'
innodb_from_lsn = 24005128478
innodb_to_lsn = 26519008580
```

```
partial = N
incremental = Y
format = file
compact = N
compressed = N
encrypted = N
```

// backup_type 显示为 log-applied，to_lsn 已经从 24005128478 升至 26519008580，和第一次增量备份的 to_lsn 对应上，说明第一次增量备份已经整合进全量备份文件中。

3. 第二次增量备份 PREPARE

xtrabackup --prepare --apply-log-only --target-dir=[全备存储目录] --incremental-dir=[第二次增量备份存储目录]

```
[dbuser@VM_0_7_centos ~]# xtrabackup --prepare --apply-log-only
--target-dir=/home/backups/backup_20201023_base
--incremental-dir=/home/backups/backup_20201023_inc2
[dbuser@VM_0_7_centos ~]# cd /home/backups/backup_20201023_base
[dbuser@VM_0_7_centos backup_20201023_base]# cat xtrabackup_checkpoints xtrabackup_info
backup_type = log-applied
from_lsn = 0
to_lsn = 26713358553
last_lsn = 26713358562
compact = 0
recover_binlog_info = 0
uuid = 5417c680-14f0-11eb-b01b-525400136b97
name =
tool_name = xtrabackup
tool_command = --user=rep --password=... --backup
--target-dir=/home/backups/backup_20201023_inc2
--incremental-basedir=/home/backups/backup_20201023_inc1
tool_version = 2.4.4
ibbackup_version = 2.4.4
server_version = 5.7.31-log
start_time = 2020-10-23 13:26:19
end_time = 2020-10-23 13:26:40
lock_time = 0
binlog_pos = filename 'mysql-bin.000013', position '133461079', GTID of the last change
'54a6c06d-f891-11ea-ad5b-525400136b97:1-5,
88aa525f-fc74-11ea-87a0-525400136b97:1-97,
8c7c177b-fbe4-11ea-ba33-525400136b97:1-3,
be98892c-1457-11eb-8d96-525400136b97:1-4'
innodb_from_lsn = 26519008580
innodb_to_lsn = 26713358553
partial = N
incremental = Y
```

```
format = file
compact = N
compressed = N
encrypted = N
```

// backup_type 显示为 log-applied，to_lsn 已经从 26519008580 升至 26713358553。

3. 全量备份回滚 PREPARE

xtrabackup --prepare --target-dir=[全备存储目录]

```
[dbuser@VM_0_7_centos ~]# xtrabackup --prepare
--target-dir=/home/backups/backup_20201023_base
[dbuser@VM_0_7_centos ~]# cd /home/backups/backup_20201023_base
[dbuser@VM_0_7_centos backup_20201023_base]# cat xtrabackup_checkpoints xtrabackup_info
backup_type = full-prepared
from_lsn = 0
to_lsn = 26713358553
last_lsn = 26713358562
compact = 0
recover_binlog_info = 0
uuid = 5417c680-14f0-11eb-b01b-525400136b97
name =
tool_name = xtrabackup
tool_command = --user=rep --password=... --backup
--target-dir=/home/backups/backup_20201023_inc2
--incremental-basedir=/home/backups/backup_20201023_inc1
tool_version = 2.4.4
ibbackup_version = 2.4.4
server_version = 5.7.31-log
start_time = 2020-10-23 13:26:19
end_time = 2020-10-23 13:26:40
lock_time = 0
binlog_pos = filename 'mysql-bin.000013', position '133461079', GTID of the last change
'54a6c06d-f891-11ea-ad5b-525400136b97:1-5,
88aa525f-fc74-11ea-87a0-525400136b97:1-97,
8c7c177b-fbe4-11ea-ba33-525400136b97:1-3,
be98892c-1457-11eb-8d96-525400136b97:1-4'
innodb_from_lsn = 26519008580
innodb_to_lsn = 26713358553
partial = N
incremental = Y
format = file
compact = N
compressed = N
encrypted = N
```

// backup_type 显示为 full-prepared，说明备份文件已经达到可以用来恢复的状态。

第二个步骤为恢复数据，目的是替换 MySQL 服务器的数据目录，启动数据库前需要确保备份数据的读写权限归属在 MySQL 服务启动用户下，步骤如 5.2 本地全量恢复所示。

5.3 逻辑文件:SQL 数据恢复

在恢复时，SQL 逻辑文件中可能存在 DDL、DML 等语句，恢复用户需要拥有对应的执行权限。

管道符：

```
[dbuser@VM_0_7_centos ~]# mysql -u -p -h -P < alldb20200922.sql
```

source：

```
[dbuser@VM_0_7_centos ~]# mysql -u -p -h –P
mysql > source alldb20200922.sql
```

注：数据库执行逻辑文件可能会分多个事务，一旦执行，数据即被修改，不能直接回滚，谨慎操作。

5.4 逻辑文件:myloader 数据恢复

在恢复时，SQL 逻辑文件中可能存在 DDL、DML 等语句，恢复用户需要拥有对应的执行权限。

全库恢复：

```
[dbuser@VM_0_7_centos ~]# myloader -u -p -h -P --threads=16 -d /allbackup
```

恢复 adb 库中 user 表：

```
[dbuser@VM_0_7_centos ~]# myloader -u -p -h -P --threads=16 –d /alluser
```

注：数据库执行逻辑文件可能会分多个事务，一旦执行，数据即被修改，不能直接回滚，谨慎操作

myloader 参数：

-q,--queries-per-transaction	每个事物集合的语句数，默认 1000
-o,--overwrite-tables	如果存在表，则覆盖
-B,database	指定某个数据库导入
-e,--enable-binlog	导入时开启 binlog，默认情况导入不写 binlog
-t,--threads	使用线程数，默认 4

5.5 二进制日志:point-in-time 恢复

恢复的过程分为解析二进制文件生成逻辑语句和导入数据库两个步骤。

解析指定 binlog 命令如下：

```
[dbuser@VM_0_7_centos ~]# mysqlbinlog -vv --base64-output=decode-rows mysql-bin.000001 > binlog1.sql
[dbuser@VM_0_7_centos ~]# more binlog1.sql
SET @@SESSION.GTID_NEXT= '88aa525f-fc74-11ea-87a0-525400136b97:91'/*!*/;
# at 339
#201020 19:43:02 server id 1  end_log_pos 412 CRC32 0xe908b92c  Query  thread_id=315
```

```
   exec_time=0  error_code=0
SET TIMESTAMP=1603194182/*!*/;
SET @@session.pseudo_thread_id=315/*!*/;
SET    @@session.foreign_key_checks=1,    @@session.sql_auto_is_null=0,    @@session.unique_checks=1,
@@session.autocommit=1/*!*/;
SET @@session.sql_mode=1436549152/*!*/;
SET @@session.auto_increment_increment=1, @@session.auto_increment_offset=1/*!*/;
/*!\C utf8 *//*!*/;
SET
@@session.character_set_client=33,@@session.collation_connection=33,@@session.collation_server=8/
*!*/;
SET @@session.lc_time_names=0/*!*/;
SET @@session.collation_database=DEFAULT/*!*/;
BEGIN
/*!*/;
# at 412
#201020 19:43:02 server id 1    end_log_pos 465 CRC32 0x2dfacfba      Table_map:      `dtadb`.`tmp_table`
mapped to number 1799
# at 465
#201020 19:43:02 server id 1    end_log_pos 8680 CRC32 0xbecb765b          Delete_rows: table id 1799
…
# at 510580
#201020 19:43:02 server id 1    end_log_pos 502665 CRC32 0x83cd5d95      Delete_rows:  table  id  1799
flags: STMT_END_F
### DELETE FROM `dtadb`.`tmp_table`
### WHERE
###     @1=14900001 /* INT meta=0 nullable=0 is_null=0 */
//每个事项是从# at pos 开始，上一行是上个事项末尾，指定这个事项的 GTID 值，且均带有一个时间值。
//末尾行是实际的执行部分，此处为 DELETE 删除记录。
```

导入逻辑语句命令如下：

```
[dbuser@VM_0_7_centos ~]# mysql –u –p
mysql > source binlog1.sql
```

利用管道符合并命令：

```
[dbuser@VM_0_7_centos  ~]#  mysqlbinlog  -vv  --base64-output=decode-rows  --stop-datetime='dtime'
mysql-bin.000001 | mysql -u –p –h –P
//stop-datetime 参数指定恢复到某个时间点为止
```

6. 备份恢复策略

事先制定切合实际的备份策略具有重要的意义，根据数据量、磁盘容量、磁盘 io、网络、cpu 等均可以定制备份方式，同时为了确保恢复的及时性，需要周期验证备份的可用性。

6.1. 策略一：XtraBackup 全量增量备份

数据库使用 InnoDB 引擎，单实例数据量大于 50G，磁盘剩余空间一半以上，目的是用于容灾备份，当数据库宕机，主从切换时，实现宕机从库快速数据恢复。缺点是本地备份时，磁盘空间需要大于数据空间的一倍以上。

全量备份：

xtrabackup 本地全量备份速度快，cpu 占用低，同时可以设置写速率阈值。命令如下：

xtrabackup --backup --rsync --parallel=16 --target-dir=[备份存储目录]

```
[dbuser@VM_0_7_centos ~]# xtrabackup --defaults-file=/etc/my.cnf --user=rep --password='!qazxsw@'
--port=3306 --backup --rsync --parallel=4 --target-dir=/home/backups/backup_20201024_base
[dbuser@VM_0_7_centos ~]# cd /home/backups/backup_20201024_base
[dbuser@VM_0_7_centos ~]# scp –r –p /home/backups/backup_20201024_base db@****:/backupdir
//利用 scp 传输文件，对本地 IO 和 CPU 影响小，-p 为指定 db 用户密码，若设置了免密则无需指定。
[dbuser@VM_0_7_centos backup_20201024_base]# cat xtrabackup_checkpoints
backup_type = full-backuped
from_lsn = 0
to_lsn = 26713434952
last_lsn = 26713434961
compact = 0
recover_binlog_info = 0
//此处 to_lsn 将用于增量备份
```

注:若网络传输过慢，可以考虑增加参数 qpress 实行压缩备份，但相应地 CPU 负载增加。

增量备份：

常规增量备份是基于全备目录进行，但当本地全备数据传输到备份主机后，增量备份可以通过 incremental-lsn 参数设置全备目录中 xtrabackup_checkpoints 文件中 to_lsn 值，进行增量备份。这里需要注意一个问题，若是使用错误的 to_lsn 值会导致增量备份不可用，故需要注意。

方式一:本地增量 scp 传输

```
[dbuser@VM_0_7_centos ~]# xtrabackup --defaults-file=/etc/my.cnf --user=rep --password='!qazxsw@'
--port=3306 --backup --target-dir=/home/backups/backup_20201024_inc1
--incremental-lsn=26713434952
[dbuser@VM_0_7_centos ~]# scp –r –p /home/backups/backup_20201024_inc1 db@****:/backupdir
```

方式二：流备份到远程

```
[dbuser@VM_0_7_centos ~]# xtrabackup --defaults-file=/etc/my.cnf --user=rep --password='!qazxsw@'
--port=3306 --backup --stream=xbstream --incremental-lsn=26713434952 | ssh db@**** "xbstream –x
–C /backupdir/backup_20201024_inc1"
//同理—stream=tar，解压缩命令为"tar –xf –C /backupdir/backup_20201024_inc1"
```

备份恢复：

将指定时间点的全备文件、增备文件传输到目的数据库主机，需要按步骤进行日志的 PREPARE，再进行恢复操作，设置主从复制。

备份周期：

备份周期根据实际数据变化速率制定，如数据平均每小时新增或修改较大时，即工作负

载较大时，设置每天一次的全量备份，加上每间隔 2 小时一次的增量备份能够将数据可能丢失的量限制在 2 个小时之内。同理，若数据变化评率不大，可以设置较大间隔的全量备份和增量备份。

全量备份	每日凌晨 1 点
增量备份	间隔 2 小时一次

存储周期：

归档数据应设置一个合适的过期时间，根据实际需求，一般保留 7 到 10 天前的数据副本以应对查找历史数据的需求。

备份验证：

备份数据的可靠性通常是一个难以度量的指标，除了备份工具所保证的数据完整性，还应设计一定的周期备份数验证方案，如周期性地从备份所在的存储服务器拉取数据到验证数据库，执行恢复操作并通过 mysqlcheck 方式确保备份数据的可用性。

6.2.策略二：mysqldump 备份（mydumper）

数据库使用 InnoDB 引擎，单实例数据量小于 50G，特点的备份和恢复较为简单，支持远程网络备份恢复。用于数据量较小的数据库备份和临时库表备份。

备份：

mysqldump -u -p -h -P --all-databases > alldb.sql

```
mysql> GRANT SELECT,PROCESS,REPLICATION CLIENT ON *.* TO dmp@'%' identified by '!qazxsw@';
//先赋权远程用户
[dbuser@VM_0_7_centos ~]# mysqldump –udmp –p '!qazxsw@' –h(ip) –P3306 –single-transaction –c
--flush-logs --master-data=2 --all-databases > backalldb20201024.sql
//flush-logs 参数是为了备份结束，新起二进制文件，以便二进制文件记录增量数据。
[dbuser@VM_0_7_centos ~]# more backalldb20201024.sql
-- MySQL dump 10.13  Distrib 5.7.31, for Linux (x86_64)
--
-- ------------------------------------------------------
-- Server version       5.7.31-log

/*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
/*!40101 SET @OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS */;
/*!40101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION */;
/*!40101 SET NAMES utf8 */;
/*!40103 SET @OLD_TIME_ZONE=@@TIME_ZONE */;
/*!40103 SET TIME_ZONE='+00:00' */;
/*!40014 SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0 */;
/*!40014 SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0
*/;
/*!40101 SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='NO_AUTO_VALUE_ON_ZERO' */;
/*!40111 SET @OLD_SQL_NOTES=@@SQL_NOTES, SQL_NOTES=0 */;


--
-- Position to start replication or point-in-time recovery from
```

```
--

-- CHANGE MASTER TO MASTER_LOG_FILE='mysql-bin.000023', MASTER_LOG_POS=314;
```

恢复：

mysql -u -p -h -P < backalldb20201024.sql

备份周期：

逻辑备份和恢复效率较低，当用作容灾备份时，必须考虑每次备份的资源消耗和时长，同时恢复的资源消耗和时长。由于每次备份均相当于全量备份，故而一般备份 MySQL 二进制日志当做增量备份，另一个优点在于二进制日志恢复能达到时间点级别。

6.3. 策略三：POINT-IN-TIME

MySQL 二进制日志记录数据的修改，形式为 SQL，意味着文件可以修改，实现数据回退，以此特点实现数据 point-in-time 基于时机的回溯。

二进制日志备份：

利用 mysqlbinlog 工具远程持续地备份二进制日志。一般地，数据库会设置二进制日志过期时间 expire_logs_days，所以持续不断的备份二进制日志是必须的。

```
[dbuser@备份机]# mysqlbinlog --user=dmp --password= '!qazxsw@' --host=**** --port=3306
--read-from-remote-server --raw --stop-never mysql-bin.000001
//从已经存在 mysql-bin.000001 开始不间断备份二进制文件
```

数据恢复：

当数据 DML 操作失误时，能利用开源工具反向解析 row 模式的二进制日志，生成可执行的数据修复 DML。

当数据 DDL 操作失误时，此时二进制日志无法记录数据变更情况，解决思路是跳过误操作：通过历史数据备份结合二进制日志解析，修改二进制解析出的 SQL 文件，实现数据回溯。

1.全备恢复：通过策略一或者二生成一个历史数据副本库，记录副本库最后一个事务在二进制日志中的位置和 GTID 信息。XtraBackup 备份结束后会在 xtrabackup_checkpoints 文件中记录，mysqldump 的参数 master-data 会生成 change master 信息记录在备份文件中。

2.增量恢复：已知副本库数据所在位置，通过解析事务位置点之后的所有二进制日志，删除解析文件中误操作语句，再回放给副本库，此时已经完成了数据回溯。此步骤的效率取决于增量备份的数据量，适当增大增量备份频率能提高此步骤的速度。

3.增量恢复到指定时间点:若存在获取某个时间点的数据需求时，stop-datetime 是二进制日志中事务的时间 id，借此能实现副本库补全数据到 stop-datetime 时间点。

mysqlbinlog mysql-bin.000023 --stop-datetime=’’ | mysql -u -p -h –P