MySQL备份浅析

简易流程图：Xtrabackup0907.png

binlog浅析：MySQL MYSQLBINLOG.md

物理备份：

- MySQL Enterprise Backup
- Percona XtraBackup

逻辑备份：

- MySQLDump
- MyDumper
- MySQL Shell

MEB没接触过 据了解备份锁很轻量，暂时略过；

MyDumper 没接触，暂时略过；{myloader（ mydumper 中的恢复工具） 是多线程导入，且一个 INSERT 语句中包含多条记录，多个 INSERT 操作批量提交；有助于提升导入速度。}

Percona XtraBackup

生产XtraBackup居多、大致流程是 会复制Innodb数据文件 同时有两个步骤：1检查数据文件 2检查事务日志；XtraBackup通过记住日志序列号（LSN）来工作 然后拷贝数据文件，拷贝过程中XtraBackup后台会运行一个进程 监控事务日志、并从复制文件中更改，XtraBackup需要不断地这样做，因为事务日志是以循环方式写入的，并且可以重用；XtraBackup需要自开始执行以来对数据文件的每次更改的事务日志记录。

XtraBackup原理以及源码解析

Xtrabackup 2.4 的备份流程

大致如下：

1. start backup
2. copy ibdata1 / copy .ibd file
3. Excuted ftwrl
4. backup non-InnoDB tables and files
5. Writing xtrabackup_binlog_info
6. Executed FLUSH NO_WRITE_TO_BINLOG ENGINE LOGS
7. Executed UNLOCK TABLES
8. Copying ib_buffer_pool
9. completed OK!

Xtrabackup 2.4 在执行 ftwrl 并备份完所有非 InnoDB 表格的文件后通过 show master status 获取了 binlog position 和 GTID 的信息，将其记录到 xtrabackup_binlog_info 文件中.

那么 show master status 获取的是哪些信息?

命令提供实例的 binlog 文件的状态信息，显示正在写入的 binlog 文件，以及当前的 binlog position，并且 MySQL 5.7 在 MySQL 库下引入了 gtid_executed 表，该表会记录当前执行过的 GTID。

表格虽然 是 InnoDB 表，但是其中的数据并非是实时更新的，且该表格记录信息的方式存在以下两个情况：

- 如果禁用了 log_bin，实例不会在该表格记录任何信息；若从库的 log_slave_updates 为 OFF，那么从 库会在应用 relay-log 中的每个事务时执行一次 insert mysql.gtid_executed 的操作。
- 如果启用了 log_bin，则该表格记录的是在 binlog 发生切换（rotate）的时候直到上一个 binlog 文件 执行过的全部 GTID，而此时 show master status 获取的 Gtid 信息不再由 mysql.gtid_executed 表提 供，而是由全局系统变量 gtid_exected 提供；如果服务器意外停止，则当前 binlog 文件中的 Gtid 集合不会保存在 mysql.gtid_executed 表中，在实例恢复期间，这些 Gtid 从 binlog 文件中读取并添 加到表中。

XtraBackup 8.0 的备份过程

大致如下：

1. start backup
2. copy .ibd file
3. backup non-InnoDB tables and files
4. Executed FLUSH NO_WRITE_TO_BINLOG BINARY LOGS
5. Selecting LSN and binary log position from p_s.log_status
6. copy last binlog file
7. Writing /mysql/backup/backup/binlog.index
8. Writing xtrabackup_binlog_info
9. Executing FLUSH NO_WRITE_TO_BINLOG ENGINE LOGS
10. copy ib_buffer_pool
11. completed OK!

Xtrabackup 8.0 对 MySQL 8.0 的备份与 Xtrabackup 2.4 略有不同，根据 percona 官方 文档的信息，当 MySQL 8.0 中仅存在 InnoDB 引擎的表格时，不再执行 ftwrl（当存在非 InnoDB 的表格 或者使用 --slave-info 选项时会执行），而是根据上述步骤的第 5 步，Xtrabackup 8.0 会通过 SELECT server_uuid, local, replication, storage_engines FROM performance_schema.log_status 来获取 LSN 、binlog position and Gtid。

Xtrabackup 8.0 会通过：

```
SELECT server_uuid, local, replication, storage_engines FROM performance_schema.log_status
```

来获取 LSN 、binlog position and Gtid;

- performance_schema.log_status 是 MySQL 8.0 提供给在线备份工具获取复制日志文件信息的表格。查询 log_status 表时，服务器将阻止日志的记录和相关的更改来获取足够的时间以填充该表，然后释放资源。 Log_status 表通知在线备份工具应记录主库的 binlog 的哪个位点和 gtid_executed 的值，还有每个复制通 道的 relay log。它还为各个存储引擎提供了相关信息，例如 InnoDB 存储引擎使用的最后一个日志序列号 （LSN）和最后一个检查点的 LSN。
- 经过测试发现，当无数据写入时， performance_schema.log_status 提供的 binlog position 与 GTID 是 一致的，但是当有大量数据持续写入时，该表格提供的 binlog position 与 GTID 信息将不再一致
- 既然 performance_schema.log_status 提供的信息不一致，那么为什么备份恢复后，GTID 没有缺失？{是因为 Xtrabackup 8.0 在备份过程中多了两步操作，FLUSH NO_WRITE_TO_BINLOG BINARY LOGS 和 copy binlog，Xtrabackup 8.0 在备份完非 InnoDB 表格的文件时会先切换 binlog，然后将切换后的 binlog 也进行备份，这样使用该备份恢复的新实例在启动后不仅会读取 gtid_executed 表，也会读取 binlog 文件 来更新 GTID，就可以保持与备份时 xtrabackup_binlog_info 文件记录的 binlog position 保持一致（需要 注意的是 MySQL 8.0 的 gtid_executed 表格不再是当 binlog 切换时更新，而是会不断的实时更新，但需 要注意在有大量数据写入时也不能做到和全局变量 gtid_exeuted 保持严格一致）}
- MySQL 8.0 中存在非 InnoDB 的表格，比如 MyISAM 表时，Xtrabackup 8.0 会在执行完 FLUSH NO_WRITE_TO_BINLOG BINARY LOGS 后执行 ftwrl，此时查询 performance_schema.log_status 得到的 binlog position 与 GTID 是一致的，且备份恢复后 show master status 显示的信息也与 xtrabackup_binlog_info 文件记录的信息一致。

```
/*小结*/
1. Xtrabackup 2.4 备份后生成的 xtrabackup_binlog_info 文件记录的 GTID 信息是准确的，但是备份恢复后 show master status 显示的 GTID 是不准确的。
2. Xtrabackup 8.0 在备份只有 InnoDB 表的实例时，xtrabackup_binlog_info 文件记录的 GTID 信息不一定是准确的，但是备份恢复后 show master status 显示的 GTID 是准确的。
3. Xtrabackup 8.0 在备份有非 InnoDB 表格的实例时，xtrabackup_binlog_info 文件记录的 GTID 信息是准确的，备份恢复后 show master status 显示的 GTID 也是准确的。
```

注意：此处的“准确”主要指 xtrabackup_binlog_info 文件中记录的 GTID 与备份中实际的 binlog position & 数据是否一致

XtraBackup 相关源码

XtraBackup的main函数定义在 storage/innobase/xtrabackup/src/xtrabackup.cc 文件中

- xtrabackup_backup_func 该函数位于xtrabackup.cc文件中（该函数的处理流程如下：）

1. 创建redo log拷贝线程，从最近的checkpoint lsn开始拷贝redo log。
2. 创建数据文件拷贝线程，拷贝ibdata1，undo tablespaces及所有的ibd文件。

3. 这里可通过设置--parallel进行多线程备份，提高物理文件的拷贝效率。不设置则默认为1。
4. ibd文件拷贝完成后，调用backup_start函数。
5. 停止redo log拷贝线程。
6. 调用backup_finish函数。

- backup_start 该函数位于backup_copy.cc文件中（该函数的处理流程如下：）

1. 调用lock_tables_maybe函数加全局读锁。lock_tables_maybe函数的处理逻辑会在下篇文章介绍。
2. 调用backup_files函数备份非ibd文件。具体来说，会备份以下面这些关键字作为后缀的文件。

```
const char *ext_list[] = {"frm", "isl", "MYD", "MYI", "MAD", "MAI",
                "MRG", "TRG", "TRN", "ARM", "ARZ", "CSM", "CSV", "opt", "par",
                NULL};
```

3. 如果命令行中指定了 --slave-info ，则会执行 SHOW SLAVE STATUS 获取复制的相关信息。
4. 如果命令行中指定了 --binlog-info ，则会执行 SHOW MASTER STATU 获取 Binlog 的位置点信息。binlog-info无需显式指定，因为它的默认值为AUTO，如果开启了Binlog，则为ON。

- backup_finish 该函数位于backup_copy.cc文件中（该函数的处理流程如下：）

1. 释放全局读锁。
2. 拷贝ib_buffer_pool和ib_lru_dump文件。
3. 将备份的相关信息记录在xtrabackup_info文件中。
   如果设置了--history ，还会将备份信息记录在 PERCONA_SCHEMA库下的xtrabackup_history表中。

XtraBackup备份锁简述（要比FWRL更轻量):

在Percona Server for MySQL 5.6+中可用，XtraBackup使用此功能自动复制非InnoDB数据，以避免阻塞修改InnoDB表的DML查询。当服务器支持备份锁时，xtrabackup将首先复制InnoDB数据，运行 `LOCK TABLES FOR BACKUP` 并复制MyISAM表和 `.frm` 文件。完成此操作后，将开始备份文件。它将备份 `.frm` 、 `.MRG` 、 `.MYD` 、 `.MYI` 、 `.TRG` 、 `.TRN` 、 `.ARM` 、 `.ARZ` 、 `.CSM` 、 `.CSV` 、 `.par` 和 `.opt` 文件;

将使用 `LOCK BINLOG FOR BACKUP` 来阻止所有 可能更改二进制日志位置或 `Exec_Master_Log_Pos` 或 `Exec_Gtid_Set` （即源二进制记录 与复制上的当前SQL线程状态对应的坐标 复制品），如 `SHOW MASTER/SLAVE STATUS` 所报告。然后，xtrabackup将 完成复制重做日志文件并获取二进制日志坐标。完成此操作后，xtrabackup将解锁二进制日志和表;

最后，二进制日志位置将被打印到 `STDERR` ， xtrabackup退出，如果一切正常，则返回0。

```
//XtraBackup官方网址
https://docs.percona.com/percona-server/5.7/management/backup_locks.html?_gl=1*1v5pq*_gcl_au*N
```

XtraBackup使用方法：

安装..略过（ Xtraback 5.7x && 8.0x 官方下载地址：https://www.percona.com/downloads/XtraBackup/LATEST/ ）

```
//相关依赖：解压qp时依赖于qpress(安装方法)5.7x需要qp 8.0不需要
1. 添加Percona存储库：
# dnf install https://repo.percona.com/yum/percona-release-latest.noarch.rpm
2. 安装qpress rpm包：
# dnf install qpress
```

目录结构（简述解释）：

```
[root@postgre bin]# tree
.
├── innobackupex -> xtrabackup
├── xbcloud
├── xbcloud_osenv
├── xbcrypt --加解密
├── xbstream --流式文件格式
└── xtrabackup
```

innobackupex 和 xtrabackup，前者是一个 perl 脚本，后者是 C/C++ 编译的二进制；

xtrabackup 是用来备份 InnoDB 表的，不能备份非 InnoDB 表，和 mysqld server 没有交互；innobackupex 脚本用来备份非 InnoDB 表，同时会调用 xtrabackup 命令来备份 InnoDB 表，还会和 mysqld server 发送命令进行交互，如加读锁（FTWRL）、获取位点（SHOW SLAVE STATUS）等。简单来说，innobackupex 在 xtrabackup 之上做了一层封装。

8.0x

备份： .xbstream 8.0x（# --compress： 参数表示使用压缩# --compress-threads：表示使用几个线程压缩 --use-memory=256M）

```
/home/soft/percona-xtrabackup-8.0.26-18-Linux-x86_64.glibc2.17/bin/xtrabackup --defaults-file=/
```

image-20230606154552222.png

解压恢复 --解压备份到指定目录： (--parallel=4线程数可选)

```
/home/soft/percona-xtrabackup-8.0.26-18-Linux-x86_64.glibc2.17/bin/xbstream -x --decompress< /r
```

image-20230606154628265.png

数据恢复：

把文件copy到数据目录 赋予mysql权限 重启尝试,数据恢复前后保证mysql stoped状态

启动mysql 校验数据;

5.7x

备份： .xbstream 5.7x

```
/opt/percona-xtrabackup-2.4.21-Linux-x86_64.glibc2.12/bin/innobackupex --user=root --password=r
```

image-20230606155323515.png

解压数据恢复 --解压备份到指定目录:

```
1. /opt/percona-xtrabackup-2.4.21-Linux-x86_64.glibc2.12/bin/xbstream -x < /root/back/Fund_comp
2. /opt/percona-xtrabackup-2.4.21-Linux-x86_64.glibc2.12/bin/innobackupex --decompress /root/ba
3. prepare流程：1、2流程走完后，数据目录会有qp文件与原生文件共存(mysql8x是没有此步骤的，一步到位。)
4. --copy-back，也可以使用cp  mv将文件还原到数据目录。最终尝试重启mysql 校验数据
```

MySQLDump

```
//将名为 "sign" 的数据库以及其中的表、触发器、存储过程、函数和事件进行备份。备份结果将保存到 /root/s
/usr/local/mysql/bin/mysqldump -h 127.0.0.1 -P 3306 -uroot -p --single-transaction --skip-opt -
```

- `-h`： 指定连接的主机名或IP地址。
- `-u`： 指定连接数据库的用户名。
- `-p`： 提示输入密码。
- `--single-transaction`： 在备份期间使用单个事务来确保数据的一致性。
- `--skip-opt`： 跳过优化选项，以减少备份文件的大小。
- `--databases`： 指定要备份的数据库，这里是备份名为 "sign" 的数据库。
- `--triggers`： 备份触发器。
- `--routines`： 备份存储过程和函数。
- `--events`： 备份事件。
- `--master-data=2`： 在备份文件中包含主服务器的二进制日志文件和位置信息。
- `--delete-master-logs`： 备份完成后删除主服务器的二进制日志文件。
- `--add-drop-database`： 在每个数据库备份之前添加 DROP DATABASE 语句。
- `--create-options`： 在 CREATE TABLE 语句中包含额外的选项。
- `--complete-insert`： 生成完整的 INSERT 语句。
- `--extended-insert`： 生成扩展的 INSERT 语句，可以减少备份文件的大小。
- `--disable-keys`： 在备份期间禁用索引。
- `--set-charset`： 设置备份文件的字符集。
- `--tz-utc`： 在备份文件中使用 UTC 时间。

- `--quick`： 使用更快的方式进行备份。
- `/opt/mysqldump_log.log`： 指定错误日志文件的路径和名称。
- `> /opt/mysqldump_20230607.sql`： 将备份输出保存到指定的文件路径和名称。
  -- 单表条件dump
  /usr/local/mysql/bin/mysqldump -h 192.168.100.166 -P 3306 -uroot -p --single-transaction --skip-opt --databases payment --tables pay_sign_platform --where="create_date < '2023-08-08'" --triggers --rouines --events --master-data=2 --delete-master-logs --add-drop-database --create-options --complete-insert --extended-insert --disable-keys --set-charset --tz-utc --quick --log-error=/opt/mysqldump_pay_sign_platform.log >/opt/mysqldump_pay_sign_platform.sql

MySQL Shell

官方在 MySQL Shell 8.0.21 中推出的 Dump & Load 工具，与 myloader 不一样 MySQL Shell Load 是通过 LOAD DATA LOCAL INFILE 命令来导入数据；LOAD DATA 操作，按照官方文档的说法，比 INSERT 操作快近20倍；

- checkForServerUpgrade：检测目标实例能否升级到指定版本。
- dumpInstance：备份实例。
- dumpSchemas：备份指定库。
- dumpTables：备份指定表。
- loadDump：恢复通过上面三个工具生成的备份。
- exportTable：将指定的表导出到文本文件中。只支持单表，效果同 `SELECT INTO OUTFILE` 一样。
- importTable：将指定文本的数据导入到表中。
  在线上，如果我们有个大文件需要导入，建议使用这个工具。它会将单个文件进行拆分，然后多线程并行执行 LOAD DATA LOCAL INFILE 操作。不仅提升了导入速度，还规避了大事务的问题。
- importJson：将 JSON 格式的数据导入到 MySQL 中，譬如将 MongoDB 中通过 mongoexport 导出的数据导入到 MySQL 中。

注意事项：

1. 通过 dumpInstance，dumpSchemas，dumpTables 生成的备份只能通过 loadDump 来恢复。
2. 通过 exportTable 生成的备份只能通过 importTable 来恢复。

安装

https://dev.mysql.com/downloads/shell/

解压即可、依赖关系软连即可。

使用

util.dumpInstance(outputUrl[, options])

```
[root@dingjia-mysql bin]# ./mysqlsh -h172.18.100.59 -P3308 -uroot -p
Please provide the password for 'root@172.18.100.59:3308': ************
Save password for 'root@172.18.100.59:3308'? [Y]es/[N]o/Ne[v]er (default No): n
MySQL Shell 8.0.33

Copyright (c) 2016, 2023, Oracle and/or its affiliates.
Oracle is a registered trademark of Oracle Corporation and/or its affiliates.
Other names may be trademarks of their respective owners.

Type '\help' or '\?' for help; '\quit' to exit.
Creating a session to 'root@172.18.100.59:3308'
Fetching schema names for auto-completion... Press ^C to stop.
Your MySQL connection id is 38
Server version: 8.0.28 MySQL Community Server - GPL
No default schema selected; type \use <schema> to set one.
 MySQL  172.18.100.59:3308 ssl  JS > util.dumpInstance('/data/back/full',{compression: "none"})
Acquiring global read lock
Global read lock acquired
Initializing - done
1 out of 5 schemas will be dumped and within them 66 tables, 0 views.
7 out of 10 users will be dumped.
Gathering information - done
All transactions have been started
Locking instance for backup
Global read lock has been released
Writing global DDL files
Writing users DDL
WARNING: User 'bkpuser'@'%' has a grant statement on an object which is not included in the dum
WARNING: User 'bkpuser'@'%' has a grant statement on an object which is not included in the dum
Running data dump using 4 threads.
NOTE: Progress information uses estimated values and may not be accurate.
NOTE: Table statistics not available for `pps`.`wp_card_ticket`, chunking operation may be not
Writing schema metadata - done
Writing DDL - done
Writing table metadata - done
Starting data dump
102% (970.88K rows / ~943.80K rows), 524.68K rows/s, 95.57 MB/s
Dump duration: 00:00:02s
Total duration: 00:00:02s
Schemas dumped: 1
Tables dumped: 66
Data size: 184.91 MB
Rows written: 970879
Bytes written: 184.91 MB
Average throughput: 86.77 MB/s
 MySQL  172.18.100.59:3308 ssl  JS >
```

命令中的 /data/backup/full 是备份目录，compression: "none" 指的是不压缩。

```
[root@dingjia-mysql opt]# ll /data/back/full|head -n 10
总用量 181588
-rw-r----- 1 root root     6199 6月   6 16:10 @.done.json
-rw-r----- 1 root root     1038 6月   6 16:10 @.json
-rw-r----- 1 root root      244 6月   6 16:10 @.post.sql
-rw-r----- 1 root root        0 6月   6 16:10 pps@attrep_apply_exceptions@@0.tsv
-rw-r----- 1 root root        8 6月   6 16:10 pps@attrep_apply_exceptions@@0.tsv.idx
-rw-r----- 1 root root      707 6月   6 16:10 pps@attrep_apply_exceptions.json
-rw-r----- 1 root root      840 6月   6 16:10 pps@attrep_apply_exceptions.sql
-rw-r----- 1 root root      368 6月   6 16:10 pps@attrep_truncation_safeguard@@0.tsv
-rw-r----- 1 root root        8 6月   6 16:10 pps@attrep_truncation_safeguard@@0.tsv.idx
```

- @.done.json：会记录备份的结束时间，备份集的大小。备份结束时生成。
- @.json：会记录备份的一些元数据信息，包括备份时的一致性位置点信息：binlogFile，binlogPosition 和 gtidExecuted，这些信息可用来建立复制。
- @.sql，@.post.sql：这两个文件只有一些注释信息。不过在通过 util.loadDump 导入数据时，我们可以通过这两个文件自定义一些 SQL。其中，@.sql 是数据导入前执行，@.post.sql 是数据导入后执行。
- sbtest.json：记录 sbtest 中已经备份的表、视图、定时器、函数和存储过程。
- *.tsv：数据文件。数据文件内容：（TSV 格式，每一行储存一条记录，字段与字段之间用制表符（\t）分隔）

```
[root@dingjia-mysql opt]# head -2 /data/back/full/pps@attrep_truncation_safeguard@@0.tsv
[249_231_weixin_to_P30_LifeMir] 07ba4d88-f136-8c4e-a9b1-66bf696b8d39    A       2023-03-27 11:0
[249_231_weixin_to_P30_LifeMir] 07ba4d88-f136-8c4e-a9b1-66bf696b8d39    B       2023-03-27 11:0
```

- sbtest@sbtest1.json：记录了表相关的一些元数据信息，如列名，字段之间的分隔符（fieldsTerminatedBy）等。
- sbtest@sbtest1.sql：sbtest.sbtest1 的建表语句。
- sbtest.sql：建库语句。如果这个库中存在存储过程、函数、定时器，也是写到这个文件中。
- @.users.sql：创建账号及授权语句。默认不会备份 mysql.session，mysql.session，mysql.sys 这三个内部账号。

util.dumpSchemas(schemas, outputUrl[, options])

```
 MySQL  172.18.100.59:3308 ssl  JS > util.dumpSchemas(['pps'],'/data/back/schema')
Acquiring global read lock
Global read lock acquired
Initializing - done
1 schemas will be dumped and within them 66 tables, 0 views.
Gathering information - done
All transactions have been started
Locking instance for backup
Global read lock has been released
Writing global DDL files
Running data dump using 4 threads.
NOTE: Progress information uses estimated values and may not be accurate.
NOTE: Table statistics not available for `pps`.`wp_card_ticket`, chunking operation may be not
Writing schema metadata - done
Writing DDL - done
Writing table metadata - done
Starting data dump
102% (970.88K rows / ~943.80K rows), 439.63K rows/s, 95.54 MB/s uncompressed, 21.84 MB/s compre
Dump duration: 00:00:02s
Total duration: 00:00:02s
Schemas dumped: 1
Tables dumped: 66
Uncompressed data size: 184.91 MB
Compressed data size: 40.41 MB
Compression ratio: 4.6
Rows written: 970879
Bytes written: 40.41 MB
Average uncompressed throughput: 79.82 MB/s
Average compressed throughput: 17.44 MB/s
 MySQL  172.18.100.59:3308 ssl  JS >
```

支持的配置大部分与 util.dumpInstance 相同。

从 MySQL Shell 8.0.28 开始，可直接使用 util.dumpInstance 中的 includeSchemas 选项进行指定库的备份。

util.dumpTables(schema, tables, outputUrl[, options])

备份指定表的数据。

用法同 util.dumpInstance 类似。其中，第二个参数必须为数组，如:

```
 MySQL  172.18.100.59:3308 ssl  JS > util.dumpTables('pps',['pps_c_bak_tree'],'/data/back/table
Acquiring global read lock
Global read lock acquired
Initializing - done
1 tables and 0 views will be dumped.
Gathering information - done
All transactions have been started
Locking instance for backup
Global read lock has been released
Writing global DDL files
Running data dump using 4 threads.
NOTE: Progress information uses estimated values and may not be accurate.
Writing schema metadata - done
Writing DDL - done
Writing table metadata - done
Starting data dump
96% (282.74K rows / ~291.77K rows), 157.84K rows/s, 43.24 MB/s uncompressed, 14.59 MB/s compres
Dump duration: 00:00:01s
Total duration: 00:00:01s
Schemas dumped: 1
Tables dumped: 1
Uncompressed data size: 63.91 MB
Compressed data size: 19.19 MB
Compression ratio: 3.3
Rows written: 282739
Bytes written: 19.19 MB
Average uncompressed throughput: 40.18 MB/s
Average compressed throughput: 12.07 MB/s
 MySQL  172.18.100.59:3308 ssl  JS >
```

支持的配置大部分与 util.dumpInstance 相同。

从 MySQL Shell 8.0.28 开始，可直接使用 util.dumpInstance 中的 includeTables 选项进行指定表的备份。

```
//校验数据、且删掉某张表
Database changed
mysql> select count(*) from pps_c_bak_tree;
+----------+
| count(*) |
+----------+
|   282739 |
+----------+
1 row in set (0.01 sec)

mysql> drop table pps_c_bak_tree;
Query OK, 0 rows affected (0.19 sec)

//使用util.loadDump进行还原
 MySQL  172.18.100.59:3308 ssl  JS > util.loadDump("/data/back/table",{loadUsers: false})
Loading DDL and Data from '/data/back/table' using 4 threads.
Opening dump...
Target is MySQL 8.0.28. Dump was produced from MySQL 8.0.28
Scanning metadata - done
Checking for pre-existing objects...
Executing common preamble SQL
Executing DDL - done
Executing view DDL - done
Starting data load
1 thds loading | 100% (63.91 MB / 63.91 MB), 11.61 MB/s, 1 / 1 tables done
Executing common postamble SQL
Recreating indexes - done
2 chunks (282.74K rows, 63.91 MB) for 1 tables in 1 schemas were loaded in 6 sec (avg throughpu
0 warnings were reported during the load.
 MySQL  172.18.100.59:3308 ssl  JS >

mysql> select count(*) from pps_c_bak_tree;
+----------+
| count(*) |
+----------+
|   282739 |
+----------+
1 row in set (0.02 sec)
```

/data/backup/full 是备份目录，loadUsers: true 是导入账号，**默认不会导入。**

util.dumpInstance 的关键特性如下：

1. 多线程备份。并发线程数由 threads 决定，默认是 4。
2. 支持单表 chunk 级别的并行备份，前提是表上存在主键或唯一索引。
3. 默认是压缩备份。
4. 支持备份限速。可通过 maxRate 限制单个线程的数据读取速率

util.loadDump 的关键特性如下：

1. 多线程恢复。并发线程数由 threads 决定，默认是 4。
2. 支持断点续传功能。
   在导入的过程中，会在备份目录生成一个进度文件，用于记录导入过程中的进度信息。
   文件名由 progressFile 指定，默认是 load-progress.<server_uuid>.progress。
   导入时，如果备份目录中存在 progressFile，默认会从上次完成的地方继续执行。如果要从头开始执行，需将 resetProgress 设置为 true。
3. 支持延迟创建二级索引。
4. 支持边备份，边导入。
5. 通过 LOAD DATA LOCAL INFILE 命令来导入数据。
6. 如果单个文件过大，util.loadDump 在导入时会自动进行切割，以避免产生大事务。

待续..