MySQL学习笔记（Day039：backup_2）

MySQL学习

一. 备份（二）

1.1. mysqldump

1.1.1. mysqldump介绍

> mysqldump官方文档

```
mysqldump [OPTIONS] --single-transaction database [tables]  # 备份某个数据库下的表
mysqldump [OPTIONS] --single-transaction --databases [OPTIONS] DB1 [DB2 DB3...]  # 备份指定数据库
mysqldump [OPTIONS] --single-transaction --all-databases [OPTIONS] # 备份所有数据库
For more options, use mysqldump --help
```

1.1.2. mysqldump重要参数

- `--all-databases` ：备份所有的数据库
- `--databases DB1 [DB2 DB3]` ：备份指定的数据库
- **`--single-transaction`** ： 在一个事物中导出，确保产生一致性的备份 ，当前只对innodb支持

1.1.3. mysqldump演示

```
shell> mysqldump --databases burn_test -S /tmp/mysql.sock_58  > burn_test.sql  # 导出的是SQL文件
# 暂时忽略Warning
Warning: A partial dump from a server that has GTIDs will by default include the GTIDs of all transactions, even those that changed suppressed parts of the database. If you don't want to restore GTIDs, pass --set-gtid-purged=OFF. To make a complete dump, pass --all-databases --triggers --routines --events.

shell> mysqldump --single-transaction --databases burn_test -S /tmp/mysql.sock_58  > burn_test_2.sql
# 暂时忽略Warning
Warning: A partial dump from a server that has GTIDs will by default include the GTIDs of all transactions, even those that changed suppressed parts of the database. If you don't want to restore GTIDs, pass --set-gtid-purged=OFF. To make a complete dump, pass --all-databases --triggers --routines --events.

mysqldump: Couldn't execute 'SAVEPOINT sp': The MySQL server is running with the --transaction-write-set-extraction!=OFF option so it cannot execute this statement (1290)
# 产生Error。需要设置下面的参数


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


shell> mysqldump --single-transaction --databases burn_test -S /tmp/mysql.sock_58  > burn_test_2.sql
Warning: A partial dump from a server that has GTIDs will by default include the GTIDs of all transactions, even those that changed suppressed parts of the database. If you don't want to restore GTIDs, pass --set-gtid-purged=OFF. To make a complete dump, pass --all-databases --triggers --routines --events.

##
## 从下面的diff看，结果没有差别，
## 如果备份的时候由其他事物在进行插入或者修改操作，其实是可以看得出差别的
##
shell> diff burn_test.sql burn_test_2.sql
69c69
< -- Dump completed on 2016-03-05 16:19:21
---
> -- Dump completed on 2016-03-05 16:35:59
```

> **single-transaction**
>
> 1. 当开始备份的时候，备份的是 备份点（备份开始的时刻） 时的数据（即使在备份过程中，表中的数据发生了改变）
> 2. 实现方式：在开启事物前，先设置为 RR 隔离级别（*事物隔离级别是会话级别，由mysqldump自己设置*），由于RR级别 解决 了 不可重复读 和 幻读 问题，所以在备份的时刻开启一个事物后，读取的数据是能保证一致性的

```
--
-- 终端会话1
--
-- 测试环境，仅mysqldump在做操作，所以可以看到mysqldump在备份过程中的动作
mysql> set global general_log=1;
Query OK, 0 rows affected (0.00 sec)

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


##
## 终端会话2
##
shell> mysqldump --single-transaction --databases burn_test -S /tmp/mysql.sock_58  > burn_test_3.sql


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

mysql> use mysql;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed

mysql> desc general_log;
+--------------+---------------------+------+-----+----------------------+-------------------------------+
| Field        | Type                | Null | Key | Default              | Extra                         |
+--------------+---------------------+------+-----+----------------------+-------------------------------+
| event_time   | timestamp(6)        | NO   |     | CURRENT_TIMESTAMP(6) | on update CURRENT_TIMESTAMP(6) |
| user_host    | mediumtext          | NO   |     | NULL                 |                               |
| thread_id    | bigint(21) unsigned | NO   |     | NULL                 |                               |
| server_id    | int(10) unsigned    | NO   |     | NULL                 |                               |
| command_type | varchar(64)         | NO   |     | NULL                 |                               |
| argument     | mediumblob          | NO   |     | NULL                 |                               |
+--------------+---------------------+------+-----+----------------------+-------------------------------+
6 rows in set (0.00 sec)

mysql> select event_time, thread_id,left(argument, 64) from general_log;
+----------------------------+-----------+------------------------------------------------------------------+
| event_time                 | thread_id | left(argument, 64)                                               |
+----------------------------+-----------+------------------------------------------------------------------+
| 2016-03-06 15:22:49.502979 |         6 | root@localhost on  using Socket                                  |
| 2016-03-06 15:22:49.511253 |         6 | /*!40100 SET @@SQL_MODE='' */                                    |
| 2016-03-06 15:22:49.511604 |         6 | /*!40103 SET TIME_ZONE='+00:00' */                               |
| 2016-03-06 07:22:49.511951 |         6 | SET SESSION TRANSACTION ISOLATION LEVEL REPEATABLE READ          |  -- mysqldump设置了一个会话级别的 RR 隔离级别
| 2016-03-06 07:22:49.512102 |         6 | START TRANSACTION /*!40100 WITH CONSISTENT SNAPSHOT */           |  -- 开启一个会话
| 2016-03-06 07:22:49.512319 |         6 | SHOW VARIABLES LIKE 'gtid\_mode'                                 |
| 2016-03-06 07:22:49.513373 |         6 | SELECT @@GLOBAL.GTID_EXECUTED                                    |
| 2016-03-06 07:22:49.513529 |         6 | UNLOCK TABLES                                                    |
| 2016-03-06 07:22:49.513770 |         6 | SELECT LOGFILE_GROUP_NAME, FILE_NAME, TOTAL_EXTENTS, INITIAL_SIZ |
| 2016-03-06 07:22:49.516539 |         6 | SELECT DISTINCT TABLESPACE_NAME, FILE_NAME, LOGFILE_GROUP_NAME,  |
| 2016-03-06 07:22:49.517618 |         6 | SHOW VARIABLES LIKE 'ndbinfo\_version'                           |
| 2016-03-06 07:22:49.518415 |         6 | burn_test                                                        |
| 2016-03-06 07:22:49.518534 |         6 | SHOW CREATE DATABASE IF NOT EXISTS `burn_test`                   |
| 2016-03-06 07:22:49.518665 |         6 | SAVEPOINT sp                                                     |  -- 使用savepoint sp，便于回滚，用于快速释放metadata数据共享锁
| 2016-03-06 07:22:49.518780 |         6 | show tables                                                      |
| 2016-03-06 07:22:49.519040 |         6 | show table status like 'test\_purge'                             |
| 2016-03-06 07:22:49.519379 |         6 | SET SQL_QUOTE_SHOW_CREATE=1                                      |
| 2016-03-06 07:22:49.519509 |         6 | SET SESSION character_set_results = 'binary'                     |
| 2016-03-06 07:22:49.519631 |         6 | show create table `test_purge`                                   |
| 2016-03-06 07:22:49.519784 |         6 | SET SESSION character_set_results = 'utf8'                       |
| 2016-03-06 07:22:49.519946 |         6 | show fields from `test_purge`                                    |
| 2016-03-06 07:22:49.520447 |         6 | show fields from `test_purge`                                    |
| 2016-03-06 07:22:49.520940 |         6 | SELECT /*!40001 SQL_NO_CACHE */ * FROM `test_purge`              |  -- 通过select 逻辑备份出test_purge中表的数据
| 2016-03-06 07:22:49.521308 |         6 | SET SESSION character_set_results = 'binary'                     |
| 2016-03-06 07:22:49.521469 |         6 | use `burn_test`                                                  |
| 2016-03-06 07:22:49.521634 |         6 | select @@collation_database                                      |
| 2016-03-06 07:22:49.521821 |         6 | SHOW TRIGGERS LIKE 'test\_purge'                                 |
| 2016-03-06 07:22:49.522342 |         6 | SET SESSION character_set_results = 'utf8'                       |
| 2016-03-06 07:22:49.522461 |         6 | ROLLBACK TO SAVEPOINT sp                                         |  -- 回滚到savepoint sp，释放metadata的锁
| 2016-03-06 07:22:49.522562 |         6 | RELEASE SAVEPOINT sp                                             |
-- ----------------------------------省略其他输出-------------------------------
+----------------------------+-----------+------------------------------------------------------------------+
67 rows in set (0.00 sec)
```

> 每取一张表的数据，就 `rollback to savepoint sp`（ *一个savepoint就够了* ）

- **`--master-data`** ： 备份的时候dump出 `CHANGE MASTER` 信息（*file 和 pos*），可供 主从复制 的时候使用， 默认值为1 。
  - 当值设置为 2 的时候，也会dump出信息，但是会被 注释 掉

```
shell>  mysqldump --single-transaction --master-data=1 --databases burn_test -S /tmp/mysql.sock_58  > burn_test_4.sql

shell>  mysqldump --single-transaction --master-data=2 --databases burn_test -S /tmp/mysql.sock_58  > burn_test_5.sql

shell> diff burn_test_4.sql  burn_test_5.sql
30c30
< CHANGE MASTER TO MASTER_LOG_FILE='bin.000021', MASTER_LOG_POS=194;   # burn_test_4.sql中是一个语句
---
> -- CHANGE MASTER TO MASTER_LOG_FILE='bin.000021', MASTER_LOG_POS=194; # burn_test_5.sql中是一个注释
75c75
< -- Dump completed on 2016-03-06 16:38:24
---
> -- Dump completed on 2016-03-06 16:38:32
```

> CHANGE MASTER 信息表示，这个mysqldump出来的文件，是在这个 MASTER_LOG_FILE 文件的 MASTER_LOG_POS 位置备份出来的，是一个 起始位置 信息

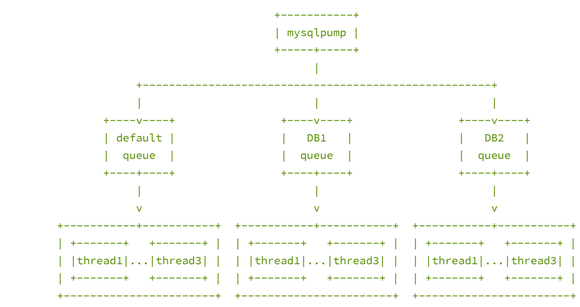
```
--
-- 增加master-data后的general-log
--
mysql> select event_time, thread_id,left(argument, 64) from general_log;
+----------------------------+-----------+------------------------------------------------------------------+
| event_time                 | thread_id | left(argument, 64)                                               |
+----------------------------+-----------+------------------------------------------------------------------+
| 2016-03-06 16:49:58.005425 |        12 | root@localhost on  using Socket                                  |
| 2016-03-06 16:49:58.013061 |        12 | /*!40100 SET @@SQL_MODE='' */                                    |
| 2016-03-06 16:49:58.013316 |        12 | /*!40103 SET TIME_ZONE='+00:00' */                               |
| 2016-03-06 08:49:58.013545 |        12 | FLUSH /*!40101 LOCAL */ TABLES                                   |
| 2016-03-06 08:49:58.019493 |        12 | FLUSH TABLES WITH READ LOCK                                      |  -- 增加一个读锁（读不受影响，写阻塞），在MySQL 上层实现的，和innodb的锁不同
| 2016-03-06 08:49:58.031595 |        12 | SET SESSION TRANSACTION ISOLATION LEVEL REPEATABLE READ          |  -- 设置RR隔离级别
| 2016-03-06 08:49:58.037625 |        12 | START TRANSACTION /*!40100 WITH CONSISTENT SNAPSHOT */           |
| 2016-03-06 08:49:58.037742 |        12 | SHOW VARIABLES LIKE 'gtid\_mode'                                 |
| 2016-03-06 08:49:58.038477 |        12 | SELECT @@GLOBAL.GTID_EXECUTED                                    |
| 2016-03-06 08:49:58.038601 |        12 | SHOW MASTER STATUS                                               |  -- 通过show master status就能看到 file 和 pos
| 2016-03-06 08:49:58.038700 |        12 | UNLOCK TABLES                                                    |  -- 解锁表
| 2016-03-06 08:49:58.039353 |        12 | SELECT LOGFILE_GROUP_NAME, FILE_NAME, TOTAL_EXTENTS, INITIAL_SIZ |
| 2016-03-06 08:49:58.041267 |        12 | SELECT DISTINCT TABLESPACE_NAME, FILE_NAME, LOGFILE_GROUP_NAME,  |
| 2016-03-06 08:49:58.042243 |        12 | SHOW VARIABLES LIKE 'ndbinfo\_version'                           |
| 2016-03-06 08:49:58.042854 |        12 | burn_test                                                        |
| 2016-03-06 08:49:58.042943 |        12 | SHOW CREATE DATABASE IF NOT EXISTS `burn_test`                   |
| 2016-03-06 08:49:58.043042 |        12 | SAVEPOINT sp                                                     |
| 2016-03-06 08:49:58.043129 |        12 | show tables                                                      |
| 2016-03-06 08:49:58.043325 |        12 | show table status like 'test\_purge'                             |
| 2016-03-06 08:49:58.043570 |        12 | SET SQL_QUOTE_SHOW_CREATE=1                                      |
| 2016-03-06 08:49:58.043665 |        12 | SET SESSION character_set_results = 'binary'                     |
| 2016-03-06 08:49:58.043754 |        12 | show create table `test_purge`                                   |
| 2016-03-06 08:49:58.043889 |        12 | SET SESSION character_set_results = 'utf8'                       |
| 2016-03-06 08:49:58.043992 |        12 | show fields from `test_purge`                                    |
| 2016-03-06 08:49:58.044405 |        12 | show fields from `test_purge`                                    |
| 2016-03-06 08:49:58.044955 |        12 | SELECT /*!40001 SQL_NO_CACHE */ * FROM `test_purge`              |
| 2016-03-06 08:49:58.045334 |        12 | SET SESSION character_set_results = 'binary'                     |
| 2016-03-06 08:49:58.045445 |        12 | use `burn_test`                                                  |
| 2016-03-06 08:49:58.045549 |        12 | select @@collation_database                                      |
| 2016-03-06 08:49:58.045667 |        12 | SHOW TRIGGERS LIKE 'test\_purge'                                 |
| 2016-03-06 08:49:58.046089 |        12 | SET SESSION character_set_results = 'utf8'                       |
| 2016-03-06 08:49:58.046210 |        12 | ROLLBACK TO SAVEPOINT sp                                         |
| 2016-03-06 08:49:58.046300 |        12 | RELEASE SAVEPOINT sp                                             |
| 2016-03-06 08:49:58.067149 |        12 |                                                                  |
| 2016-03-06 16:50:02.391098 |         7 | set global general_log=0                                         |
+----------------------------+-----------+------------------------------------------------------------------+
35 rows in set (0.00 sec)


-- 在之前做一次FLUSH TABLES WITH READ LOCK，保证没有事物去写binlog
-- 然后迅速UNLOCK TABLES
-- 这样start transaction后，记录的file和pos，以及备份出来的数据就是一致的
mysql> show master status;  -- 就能看到file 和 pos了
+------------+----------+--------------+------------------+-------------------------------------------+
| File       | Position | Binlog_Do_DB | Binlog_Ignore_DB | Executed_Gtid_Set                         |
+------------+----------+--------------+------------------+-------------------------------------------+
| bin.000021 |      347 |              |                  | cc7de234-dfa3-11e5-96e2-5254a03976fb:1-161 |
+------------+----------+--------------+------------------+-------------------------------------------+
1 row in set (0.00 sec)
```

1.2. mysqlpump

1.2.1. mysqlpump介绍

注意：确保在 MySQL5.7.11 版本上使用 mysqlpump 。，在之前的版本上，并行导出 和 single-transaction 是互斥的。



1. mysqlpump 是多线程的，但是只能到 表级别 ，对于 一张表 来说，还是 单线程 的
2. mysqlpump 有默认的队列（default），队列下面可以有N个线程去备份数据库/数据库下的表
3. mysqlpump 可以开多个队列（对应不同的库/表），然后每个队列设置不同的线程数，进行并发备份

1.2.2. mysqlpump重要参数

mysqlpump参数常用参数同mysqldump类似，以下参数和并发相关

1. --default-parallelism=# 线程数，默认开 2 个线程进行并发备份
2. --parallel-schemas=name 哪些数据库进行并发备份

1.2.3. mysqlpump演示

```
shell> mysqlpump --single-transaction --databases employees -S /tmp/mysql.sock_58  > employees_pump_1.sql
# 和mysqldump类似，只是目前还不支持master-data
Dump progress: 0/2 tables, 9/330409 rows   ## 下面这些都是进度条
Dump progress: 2/6 tables, 462033/3830787 rows
Dump progress: 4/6 tables, 1176910/3830787 rows
Dump progress: 5/6 tables, 1989968/3830787 rows
Dump progress: 5/6 tables, 2874968/3830787 rows
Dump progress: 5/6 tables, 3669718/3830787 rows
Dump completed in 5629 milliseconds


-- 备份的同时，可以看到有两个线程在select数据（--default-parallelism default is 2）
mysql> show processlist;
+----+------+-----------+------+---------+------+------------------+-----------------------------------------------------------------------------------------------------------+
| Id | User | Host      | db   | Command | Time | State            | Info                                                                                                      |
+----+------+-----------+------+---------+------+------------------+-----------------------------------------------------------------------------------------------------------+
|  2 | root | localhost | NULL | Query   |    0 | starting         | show processlist                                                                                          |
|  8 | root | localhost | NULL | Query   |    0 | Sending to client | SELECT SQL_NO_CACHE `emp_no`,`title`,`from_date`,`to_date`  FROM `employees`.`titles`                    |
|  9 | root | localhost | NULL | Query   |    0 | Sending to client | SELECT SQL_NO_CACHE `emp_no`,`salary`,`from_date`,`to_date`  FROM `employees`.`salaries`                 |
| 10 | root | localhost | NULL | Sleep   |    0 |                  | NULL                                                                                                      |
+----+------+-----------+------+---------+------+------------------+-----------------------------------------------------------------------------------------------------------+
4 rows in set (0.00 sec)


[root@MyServer new_data]>mysqlpump -S /tmp/mysql.sock_58  --single-transaction --parallel-schemas=2:employees --parallel-schemas=4:dbt3 -B employees dbt3 > backup.sql
Dump progress: 0/4 tables, 25/6800319 rows
Dump progress: 3/15 tables, 300808/11535192 rows
## -----------------省略部分输出--------------------
Dump completed in 52317 milliseconds

# --parallel-schemas=2:employees  表示备份employees库使用2个线程
# --parallel-schemas=4:dbt3  表示备份dbt3库使用4个线程
# -B employees dbt3 表示指定备份 employees 和 dbt3 这两个库


mysql> show processlist;
+----+------+-----------+------+---------+------+------------------+-----------------------------------------------------------------------------------------------------------+
| Id | User | Host      | db   | Command | Time | State            | Info                                                                                                      |
+----+------+-----------+------+---------+------+------------------+-----------------------------------------------------------------------------------------------------------+
|  2 | root | localhost | NULL | Query   |    0 | starting         | show processlist                                                                                          |
|  3 | root | localhost | NULL | Sleep   |    5 |                  | NULL                                                                                                      |
|  4 | root | localhost | NULL | Query   |    1 | Sending to client | SELECT SQL_NO_CACHE `emp_no`,`salary`,`from_date`,`to_date`  FROM `employees`.`salaries`                 |
|  5 | root | localhost | NULL | Query   |    0 | Sending to client | SELECT SQL_NO_CACHE `emp_no`,`title`,`from_date`,`to_date`  FROM `employees`.`titles`                    |
|  6 | root | localhost | NULL | Query   |    5 | Sending to client | SELECT SQL_NO_CACHE `o_orderkey`,`o_custkey`,`o_orderstatus`,`o_totalprice`,`o_orderDATE`,`o_orderpr      |
|  7 | root | localhost | NULL | Query   |    5 | Sending to client | SELECT SQL_NO_CACHE `p_partkey`,`p_name`,`p_mfgr`,`p_brand`,`p_type`,`p_size`,`p_container`,`p_retai      |
|  8 | root | localhost | NULL | Query   |    5 | Sending to client | SELECT SQL_NO_CACHE `l_orderkey`,`l_partkey`,`l_suppkey`,`l_linenumber`,`l_quantity`,`l_extendedpric      |
|  9 | root | localhost | NULL | Query   |    0 | Sending to client | SELECT SQL_NO_CACHE `ps_partkey`,`ps_suppkey`,`ps_availqty`,`ps_supplycost`,`ps_comment`  FROM `dbt3      |
| 10 | root | localhost | NULL | Sleep   |    5 |                  | NULL                                                                                                      |
| 11 | root | localhost | NULL | Sleep   |    1 |                  | NULL                                                                                                      |
+----+------+-----------+------+---------+------+------------------+-----------------------------------------------------------------------------------------------------------+
10 rows in set (0.00 sec)
-- id为4和5的两个线程在select备份employees库
-- id为6、7、8、9的 线程在备份dbt3库
```

> mysqlpump 会先插入数据，在建立索引 ；而mysqldump在建立表的时候就把索引加上了，所以mysqlpump在导入数据的时候也比mysqldump要快

- mysqlpump原理

```
mysql> truncate general_log;
Query OK, 0 rows affected (0.03 sec)

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

mysql> set global general_log=1;
Query OK, 0 rows affected (0.00 sec)


[root@MyServer new_data]> mysqlpump -S /tmp/mysql.sock_58  --single-transaction burn_test > burn_test_1.sql
Dump progress: 0/1 tables, 8/9 rows
Dump completed in 440 milliseconds
```

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

mysql> select event_time, thread_id,left(argument, 64) from general_log;                                         +----------------------------+-----------+------------------------------------------------------------------+
| event_time                 | thread_id | left(argument, 64)                                               |
+----------------------------+-----------+------------------------------------------------------------------+
| 2016-03-07 21:15:56.065456 |        12 | root@localhost on  using Socket                                  |
| 2016-03-07 21:15:56.072240 |        12 | FLUSH TABLES WITH READ LOCK                                      | -- 线程 12 做 一个读锁，保证数据不会被更改
| 2016-03-07 21:15:56.089271 |        12 | SHOW WARNINGS                                                    |
| 2016-03-07 21:15:56.094442 |        12 | SET SESSION TRANSACTION ISOLATION LEVEL REPEATABLE READ          |  -- 线程12 设置会话隔离级别为RR级别
| 2016-03-07 21:15:56.094505 |        12 | SHOW WARNINGS                                                    |
| 2016-03-07 21:15:56.094556 |        12 | START TRANSACTION WITH CONSISTENT SNAPSHOT                       |  -- 线程12 开启一个事物
| 2016-03-07 21:15:56.094616 |        12 | SHOW WARNINGS                                                    |
| 2016-03-07 21:15:56.094947 |        13 | root@localhost on  using Socket                                  |
| 2016-03-07 21:15:56.095208 |        13 | SET SESSION TRANSACTION ISOLATION LEVEL REPEATABLE READ          | -- 线程13 设置会话隔离级别为RR级别
| 2016-03-07 21:15:56.095396 |        13 | SHOW WARNINGS                                                    |
| 2016-03-07 21:15:56.095452 |        13 | START TRANSACTION WITH CONSISTENT SNAPSHOT                       | -- 线程13 开启一个事物
| 2016-03-07 21:15:56.095501 |        13 | SHOW WARNINGS                                                    |
| 2016-03-07 21:15:56.095827 |        14 | root@localhost on  using Socket                                  |
| 2016-03-07 21:15:56.096066 |        14 | SET SESSION TRANSACTION ISOLATION LEVEL REPEATABLE READ          | -- 线程14 设置会话隔离级别为RR级别
| 2016-03-07 21:15:56.096441 |        14 | SHOW WARNINGS                                                    |
| 2016-03-07 21:15:56.096501 |        14 | START TRANSACTION WITH CONSISTENT SNAPSHOT                       | -- 线程14 开启一个事物
| 2016-03-07 21:15:56.096550 |        14 | SHOW WARNINGS                                                    |
| 2016-03-07 21:15:56.096602 |        12 | UNLOCK TABLES                                                    |
-- ------------------省略部分输出---------------------
| 2016-03-07 13:15:56.156289 |        12 | SELECT `COLUMN_NAME`, `EXTRA` FROM `INFORMATION_SCHEMA`.`COLUMNS` | -- 使用了元数据表，mysql5.5等版本可能不支持mysqlpump（有可能，需要测试）
-- ------------------省略部分输出---------------------
```

> 1. 线程12 进行 `FLUSH TABLES WITH READ LOCK` 。对表加一个读锁
> 2. 线程12、13、14 分别开启一个事物（RR隔离级别）去备份数据，由于之前锁表了，所以这三个线程备份出的数据是具有一致性的
> 3. 线程12 解锁 `UNLOCK TABLE`
>
> 目前，**mysqlpump** 不支持 **master-data**

1.2.4. compress-output

mysqlpump 直接 支持压缩 功能，支持 `LZ4` 和 `ZLIB`（*ZLIB压缩比相对较高，但是速度较慢*）

```
[root@MyServer new_data]> mysqlpump -S /tmp/mysql.sock_58  --single-transaction  --compress-output=lz4  burn_test > burn_test_2.sql
Dump progress: 0/1 tables, 8/9 rows
Dump completed in 391 milliseconds
```

1.2.5. mysqlpump恢复

1. **未压缩的库**

```
shell> mysql < backup.sql # mysql -u root -p -S /tmp/mysql.sock1
```

2. **压缩的库**

```
# zlib_decompress
# lz4_decompress
shell> lz4_decompress backup.sql.lz4 backup.sql  # 先解压
shell> mysql < backup.sql # mysql -u root -p -S /tmp/mysql.sock1
```

> `mysqlpump` 目前 导入 的时候是 单线程 的

以下内容根据 **M006-上海-SHK-Peter** 同学测试后，重新整理。

1.3. Xtrabackup

> xtrabackup 2.4.1 download

1. xtrabackup 只能备份innodb存储引擎表（用的较少）
2. innobackupex 可以备份内其他存储引擎（含innodb）
   - innobackupex 在 xtrabackup 的基础上做了包装，以兼容各种存储引擎
   - 平时在 备份/恢复 操作的过程中， `使用innobackupex`
3. 备份时，默认读取MySQL配置文件（读取datadir）

1.3.1. xtrabackup备份的原理

xtrabackup备份的是 备份结束点 **的数据**（*而mysqldump是备份开始时的数据*）

1. 备份表空间文件(*frm*、*ibd*、*ibdata1*、*undo*...)
2. `持续备份REDO LOG`（*log scanned up*）
3. FLUSH TABLES WITH READ LOCK – 把表锁住，保证在结束点查看master status时，没有数据插入
4. `FLUSH NO_WRITE_TO_BINLOG ENGINE LOGS` – 强制把 **redo log** 刷新到磁盘 **（主要是commit log）**
   - 确保 `commit log` 已经刷盘（*commit中的第三个步骤*）
     - 因为xtrabackup `不会备份binlog`
     - 如果没有 `commit log` ，就只有 `prepare log`（*commit中的第一步*），那对于这部分的事物就要 回滚（*意味着备份不完整，丢失数据*）
   - 之前老版本的xtrabackup没法备份5.6+，就是因为少做这个步骤
5. UNLOCK TABLES

1.3.2. 创建独立备份用户

如果希望xtrabackup使用单独的 备份用户 ，可以使用下列SQL创建备份用户，也可以直接使用root用户

```
mysql> CREATE USER 'bkpuser'@'localhost' IDENTIFIED BY 's3cret';
mysql> GRANT RELOAD, LOCK TABLES, REPLICATION CLIENT ON *.* TO 'bkpuser'@'localhost';
mysql> FLUSH PRIVILEGES;
```

1.3.3. xtrabackup完全备份

```
# 指定一个特定的用户进行备份，或者使用my.cnf中[client]的user和pass
shell> innobackupex --user=username --password=pass /path/to/backup_dir/   # 默认全部备份
```

由于使用了多实例环境，默认的 `my.cnf` 无法直接被innobackupex使用，需要按照官方文档指定 `--defaults-file`

1. 指定配置文件

```
[root@MyServer new_data]> cat my_5711.cnf
##
## 注意，这里最后使用的是/etc/my.cnf的文件，把其中datadir,port,socket等参数适当修改，以复合实例的需求
## innodb_page_size 十分重要
##
[mysqld]
datadir = /data/mysql_data/5.7.11
basedir = /usr/local/mysql_5_7_11
port = 3358
socket = /tmp/mysql.sock_58
innodb_page_size=8192  # 由于my.cnf模板中配置的是8K的页大小
                       # 如果这里不指定，innobackupex会用16K的页大小去做校验
                       # 一备份就提示 InnoDB: Checksum mismatch in datafile: ./ibdata1,
##
## ----------省略my.cnf中的其他配置项--------
##
```

2. 安装 `perl-DBD-MySQL`

```
shell> yum install perl-DBD-MySQL
# 如果不安装，会报如下错误
# Failed to connect to MySQL server as DBD::mysql module is not installed at - line 1327.
```

3. 开始备份

```
[root@MyServer new_data]> innobackupex --defaults-file=./my_5711.cnf  --user=root --password=123  ./backup_test/

#---------------------省略部分输出--------------------

160315 13:07:19 Finished backing up non-InnoDB tables and files
160315 13:07:19 [00] Writing xtrabackup_binlog_info
160315 13:07:19 [00]        ...done
160315 13:07:19 Executing FLUSH NO_WRITE_TO_BINLOG ENGINE LOGS...
##
## 执行了 FLUSH NO_WRITE_TO_BINLOG ENGINE LOGS，保证commit log全部落盘
##
xtrabackup: The latest check point (for incremental): '6668380834'
xtrabackup: Stopping log copying thread.
.160315 13:07:19 >> log scanned up to (6668380843)

160315 13:07:19 Executing UNLOCK TABLES
160315 13:07:19 All tables unlocked
160315 13:07:19 [00] Copying ib_buffer_pool to /new_data/backup_test/2016-03-15_13-06-43/ib_buffer_pool
160315 13:07:19 [00]        ...done
160315 13:07:19 Backup created in directory '/new_data/backup_test/2016-03-15_13-06-43'
MySQL binlog position: filename 'bin.000042', position '194', GTID of the last change 'cc7de234-dfa3-11e5-96e2-5254a03976fb:1-1986'
160315 13:07:19 [00] Writing backup-my.cnf
160315 13:07:19 [00]        ...done
160315 13:07:19 [00] Writing xtrabackup_info
160315 13:07:19 [00]        ...done
xtrabackup: Transaction log of lsn (6668380834) to (6668380843) was copied.
160315 13:07:20 completed OK!
```

4. 查看备份出来的文件

```
[root@MyServer new_data]> ll
total 4
drwxr-x---. 8 root root 4096 Mar 15 13:07 2016-03-15_13-06-43

[root@MyServer backup_test]# ll 2016-03-15_13-06-43/
total 266300
-rw-r-----. 1 root root      412 Mar 15 13:07 backup-my.cnf  # 配置文件
drwxr-x---. 2 root root     4096 Mar 15 13:07 burn_test
drwxr-x---. 2 root root     4096 Mar 15 13:07 dbt3
drwxr-x---. 2 root root     4096 Mar 15 13:07 employees
-rw-r-----. 1 root root     6174 Mar 15 13:07 ib_buffer_pool
-rw-r-----. 1 root root 12582912 Mar 15 13:06 ibdata1
drwxr-x---. 2 root root     4096 Mar 15 13:07 mysql
drwxr-x---. 2 root root     4096 Mar 15 13:07 performance_schema
drwxr-x---. 2 root root    12288 Mar 15 13:07 sys
-rw-r-----. 1 root root 83886080 Mar 15 13:06 undo001
-rw-r-----. 1 root root 88080384 Mar 15 13:06 undo002
-rw-r-----. 1 root root 88080384 Mar 15 13:06 undo003
-rw-r-----. 1 root root       59 Mar 15 13:07 xtrabackup_binlog_info
-rw-r-----. 1 root root      119 Mar 15 13:07 xtrabackup_checkpoints
-rw-r-----. 1 root root      568 Mar 15 13:07 xtrabackup_info
-rw-r-----. 1 root root     2560 Mar 15 13:07 xtrabackup_logfile
```

> xtrabackup除了备份指定的库，还要备份共享表空间，undo表空间等等。
> 并且生成对应的4个文件
>
> 1. **xtrabackup_binlog_info** – 包含了binlog的文件名和position
> 2. **xtrabackup_checkpoints** – 包含了备份过程中的checkpoint、LSN信息
> 3. **xtrabackup_info** – 包含了备份过程中的整体信息
> 4. **xtrabackup_logfile** – 持续备份的日志文件（redo）

- **xtrabackup_binlog_info**

```
[root@MyServer 2016-03-15_13-06-43]> cat xtrabackup_binlog_info
bin.000042      194     cc7de234-dfa3-11e5-96e2-5254a03976fb:1-1986
```

- **xtrabackup_checkpoints**

```
[root@MyServer 2016-03-15_13-06-43]> cat xtrabackup_checkpoints
backup_type = full-backuped
from_lsn = 0
to_lsn = 6668380834
last_lsn = 6668380843
compact = 0
recover_binlog_info = 0
```

- **xtrabackup_info**

```
[root@MyServer 2016-03-15_13-06-43]> cat xtrabackup_info
uuid = cb63320c-ea6b-11e5-b149-5254a03976fb
name =
tool_name = innobackupex
tool_command = --defaults-file=./my_5711.cnf --user=root --password=... ./backup_test/
tool_version = 2.4.1
ibbackup_version = 2.4.1
server_version = 5.7.11-log
start_time = 2016-03-15 13:06:43
end_time = 2016-03-15 13:07:19
lock_time = 0
binlog_pos = filename 'bin.000042', position '194', GTID of the last change 'cc7de234-dfa3-11e5-96e2-5254a03976fb:1-1986'
innodb_from_lsn = 0
innodb_to_lsn = 6668380834
partial = N
incremental = N
format = file
compact = N
compressed = N
encrypted = N
```

> 建议将全备份的文本目录做一次备份，防止后面的测试过程中出错

1.3.4. xtrabackup完全恢复

在单台机器上（*多实例*）进行恢复测试（ *不同的机器需要增加备份拷贝* ），新建一个 `datadir` 用于恢复之前的备份数据，并修改 `my_5711_2.cnf`

```
## my_5711_2.cnf
[mysqld]
datadir = /data/mysql_data/5.7.11_2  # 为了恢复，新建的目录
basedir = /usr/local/mysql_5_7_11
port = 3359
socket = /tmp/mysql.sock_59
innodb_page_size=8192
## ----------其他配置省略输出----------
```

1. **apply-log**

```
[root@MyServer new_data]> innobackupex  --defaults-file=./my_5711_2.cnf --apply-log  ./backup_test/2016-03-15_13-06-43/

## ----------------省略部分输出---------------

InnoDB: Shutdown completed; log sequence number 6668381224
160315 14:29:16 completed OK!
```

2. **copy-back**

```
[root@MyServer new_data]> innobackupex  --defaults-file=./my_5711_2.cnf --copy-back ./backup_test/2016-03-15_13-06-43/

## ----------------省略部分输出---------------

160315 14:31:08 completed OK!

## 成功以后，数据会拷贝到 my_5711_2.cnf 中指定的 datadir 目录中去
```

3. **启动测试**

在 `my.cnf` 中增加 `[mysql57112]` ，用于多实例环境

```
[mysqld57112]
server-id = 5711
datadir = /data/mysql_data/5.7.11_2/
basedir = /usr/local/mysql_5_7_11
port = 3359
socket = /tmp/mysql.sock_59
plugin_dir=/usr/local/mysql_5_7_11/lib/plugin


## 初始化一次证书
[root@MyServer mysql_5_7_11]> bin/mysql_ssl_rsa_setup  --datadir=/data/mysql_data/5.7.11_2/ --user=mysql --uid=mysql

# 修改权限
[root@MyServer mysql_5_7_11]> chown mysql.mysql /data/mysql_data/5.7.11_2 -R

# 启动新的实例
[root@MyServer mysql_5_7_11]> mysqld_multi  start 57112

# 进入mysql
[root@MyServer 5.7.11_2]> mysql -u root -p -S /tmp/mysql.sock_59
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 2
Server version: 5.7.11-log MySQL Community Server (GPL)

Copyright (c) 2000, 2016, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
```

> `--apply-log` 与 `--copy-back` 步骤可以调换，可以先拷贝到 `datadir` 后，再做 `redo` 日志重放（apply-log）

1.3.5. xtrabackup增量备份

增量备份可以是链式的，每次增量备份都要基于上一次的备份的LSN信息

```
+--------------------------+                +--------------------------+                +--------------------------+
|       Full Backlup       |        +-------+  Incremental Backup 1   |        +-------+  Incremental Backup 2   |
+--------------------------+        |       +--------------------------+        |       +--------------------------+
|  xtrabackup_checkpoint  <--------+       |  xtrabackup_checkpoint  <--------+       |  xtrabackup_checkpoint  |
+--------------------------+                +--------------------------+                +--------------------------+
| log/ibd/frm/ibdata1...  |  read lsn info  | log/ibd/frm/ibdata1...  |  read lsn info  | log/ibd/frm/ibdata1...  |
+--------------------------+                +--------------------------+                +--------------------------+
```

也可以每一次增量备份都是基于上一次全量备份

```
                                            +--------------------------+
                                read lsn info   |  Incremental Backup 1   |
                                            +--------------------------+
                                       +--------+  xtrabackup_checkpoint  |
+--------------------------+           |       +--------------------------+
|       Full Backlup       |           |       | log/ibd/frm/ibdata1...  |
+--------------------------<-----------+       +--------------------------+
|  xtrabackup_checkpoint  |
+--------------------------<-----------+       +--------------------------+
| log/ibd/frm/ibdata1...  |           |       |  Incremental Backup 2   |
+--------------------------+           |       +--------------------------+
                                       +--------+  xtrabackup_checkpoint  |
                                            +--------------------------+
                                read lsn info   | log/ibd/frm/ibdata1...  |  (include Incremental Backup 1 Data)
                                            +--------------------------+
```

```
-- 在原来的 5711实例 的 burn_test 数据库上增加 test_inc 表
mysql> create table test_inc(a int primary key);
Query OK, 0 rows affected (0.16 sec)
```

```
[root@MyServer backup_test]> innobackupex  --defaults-file=./my_5711.cnf  --user=root --password=123 --incremental /new_data/backup_test/inc --incremental-basedir=/new_data/backup_test/2016-03-15_13-06-43/
```

1. `--incremental` – 指定进行增量备份
2. `--incremental-basedir` – 指定之前 完整备份/上一次增量备份 的文件夹

```
[root@MyServer 2016-03-15_16-43-31]> ll
total 340
-rw-r-----. 1 root root    412 Mar 15 16:43 backup-my.cnf
drwxr-x---. 2 root root   4096 Mar 15 16:43 burn_test
drwxr-x---. 2 root root   4096 Mar 15 16:43 dbt3
drwxr-x---. 2 root root   4096 Mar 15 16:43 employees
-rw-r-----. 1 root root   6174 Mar 15 16:43 ib_buffer_pool
-rw-r-----. 1 root root 172032 Mar 15 16:43 ibdata1.delta
-rw-r-----. 1 root root     43 Mar 15 16:43 ibdata1.meta
drwxr-x---. 2 root root   4096 Mar 15 16:43 mysql
drwxr-x---. 2 root root   4096 Mar 15 16:43 performance_schema
drwxr-x---. 2 root root  12288 Mar 15 16:43 sys
-rw-r-----. 1 root root  40960 Mar 15 16:43 undo001.delta
-rw-r-----. 1 root root     43 Mar 15 16:43 undo001.meta
-rw-r-----. 1 root root  16384 Mar 15 16:43 undo002.delta
-rw-r-----. 1 root root     43 Mar 15 16:43 undo002.meta
-rw-r-----. 1 root root  40960 Mar 15 16:43 undo003.delta
-rw-r-----. 1 root root     43 Mar 15 16:43 undo003.meta
-rw-r-----. 1 root root     59 Mar 15 16:43 xtrabackup_binlog_info
-rw-r-----. 1 root root    126 Mar 15 16:43 xtrabackup_checkpoints
-rw-r-----. 1 root root    667 Mar 15 16:43 xtrabackup_info
-rw-r-----. 1 root root   2560 Mar 15 16:43 xtrabackup_logfile
[root@MyServer 2016-03-15_16-43-31]> du -sh
3.8M    .   # 增量备份的数据很小，因为值增加了一个table
```

```
-- 再建立一个表，产生增量数据
mysql> create table test_inc_2(a int primary key);
Query OK, 0 rows affected (0.13 sec)
```

```
[root@MyServer 2016-03-15_16-43-31]> innobackupex  --defaults-file=./my_5711.cnf  --user=root --password=123 --incremental /new_data/backup_test/inc --incremental-basedir=/new_data/backup_test/inc/2016-03-15_16-43-31/
# 通过上一次的增量备份，建立新的增量备份
```

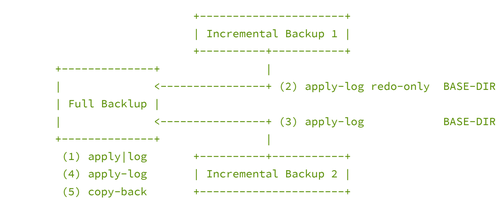
至此，我们做好了 一次全备，两次增量备份 （每次增量备份都是创建了一张表）

```
# 全备份的LSN信息
[root@MyServer 2016-03-15_13-06-43]> cat xtrabackup_checkpoints
backup_type = full-prepared  # 类型是全备份
from_lsn = 0
to_lsn = 6668380834    # 备份到这个位置
last_lsn = 6668380843
compact = 0
recover_binlog_info = 0

# 第一次增量备份的LSN信息
[root@MyServer backup_test]> cat inc/2016-03-15_16-43-31/xtrabackup_checkpoints
backup_type = incremental  # 类型是增量备份
from_lsn = 6668380834   # 接着全备份的 to_lsn 开始
to_lsn = 6668395094     # 备份到这个位置
last_lsn = 6668395103
compact = 0
recover_binlog_info = 0

# 第二次增量备份的LSN信息
[root@MyServer backup_test]> cat inc/2016-03-15_19-04-51/xtrabackup_checkpoints
backup_type = incremental  # 类型是增量备份
from_lsn = 6668395094   # 接着第一次增量备份的 to_lsn 开始
to_lsn = 6668402282
last_lsn = 6668402291
compact = 0
recover_binlog_info = 0
```

1.3.6. xtrabackup增量恢复



如果你有多个增量备份，增量恢复的步骤相对较多

1. `innobackupex --apply-log --redo-only BASE-DIR`
   - `BASE-DIR` 指完整的全部备份目录
2. `innobackupex --apply-log --redo-only BASE-DIR --incremental-dir=INCREMENTAL-DIR-1`
   - `INCREMENTAL-DIR-1` 指第一次增量备份的目录
3. `innobackupex --apply-log BASE-DIR --incremental-dir=INCREMENTAL-DIR-2`
   - `INCREMENTAL-DIR-2` 第二次增量备份的目录
   - 如果此时是 最后一个增量备份 ，就 不要使用**`--redo-only`**选项
4. `innobackupex --apply-log BASE-DIR` （可选）
   - 原文：*Once you merge the base with all the increments, you can prepare it to roll back the uncommitted transactions*
     - 官方建议做一次，来回滚掉没有提交的事物
     - 即使你不做这个一步，数据库在启动的时候也会回滚掉未提交的事物（*启动会变慢*）
5. `innobackupex --copy-back BASE-DIR`
   - 再通过 全量的copy-back 进行还原

增量备份还原就是，把 增量目录 下的数据， 整合 到 全备份目录 下，然后在进行 全量数据的还原

> 在应用 最后一次 apply-log之前，都需要增加 `--redo-only` 参数
>
> **这里的 `BASE-DIR` 和 `--incremental-dir` 请保持 绝对路径 ，不然可能会提示 找不到`xtrabackup_logfile`**
> (*主要是 `--incremental-dir` 这个参数需要绝对路径，因为innobackupex会先 cd 到 BASE-DIR目录中去*)

```
[root@MyServer backup_test]>  tree | grep -E "2016|inc" | grep -v test_inc | grep -v frm
├── 2016-03-15_13-06-43  #  这个是一个全备份的文件目录
└── inc
    ├── 2016-03-15_16-43-31   # 第一次增量备份的文件目录
    └── 2016-03-15_19-04-51   # 第二次增量备份的文件目录
#
# 步骤一：先完整应用整个备份
#
[root@MyServer new_data]> innobackupex  --defaults-file=./my_5711_2.cnf  --user=root --password=123  --apply-log --redo-only ./backup_test/2016-03-15_13-06-43/
## ----------------省略部分输出--------------
160315 23:16:09 completed OK!
#
# 步骤二：应用第一个增量备份
# 切记，使用绝对路径
#
[root@MyServer new_data]> innobackupex  --defaults-file=./my_5711_2.cnf  --user=root --password=123  --apply-log --redo-only /new_data/backup_test/2016-03-15_13-06-43 --incremental-dir=/new_data/backup_test/inc/2016-03-15_16-43-31
## ----------------省略部分输出--------------
160315 23:26:10 completed OK!
#
# 步骤三：应用第二个增量备份
# 切记，使用绝对路径
# 不要使用 --redo-only 选项
#
[root@MyServer new_data]> innobackupex  --defaults-file=./my_5711_2.cnf  --user=root --password=123  --apply-log /new_data/backup_test/2016-03-15_13-06-43 --incremental-dir=/new_data/backup_test/inc/2016-03-15_19-04-51
## ----------------省略部分输出--------------
160315 23:29:42 completed OK!
#
# 步骤四：可选，再次apply-log
#
[root@MyServer new_data]> innobackupex  --defaults-file=./my_5711_2.cnf  --user=root --password=123  --apply-log  ./backup_test/2016-03-15_13-06-43/
160315 23:32:14 completed OK!
```

至此，之前 全量备份 的目录中 数据 是 两次增量备份整合的数据

```
[root@MyServer new_data]>  rm -rf /data/mysql_data/5.7.11_2/*  # 清空需要还原的测试实例中的数据
#
# 步骤五：  还原数据库
#
[root@MyServer new_data]> innobackupex  --defaults-file=./my_5711_2.cnf  --user=root --password=123  --copy-back  ./backup_test/2016-03-15_13-06-43/

# 同样做一次证书初始化 bin/mysql_ssl_rsa_setup  --datadir=/data/mysql_data/5.7.11_2/ --user=mysql --uid=mysql
# 修改datadir权限 chown mysql.mysql /data/mysql_data/5.7.11_2/ -R
# 启动 [mysql57112] 实例

[root@MyServer mysql_5_7_11]> mysql -uroot -p -S /tmp/mysql.sock_59
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 4
Server version: 5.7.11-log MySQL Community Server (GPL)

Copyright (c) 2000, 2016, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> use burn_test;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> show tables;
+---------------------+
| Tables_in_burn_test |
+---------------------+
| test_inc            |  ## 增量备份1 中创建的表
| test_inc_2          |  ## 增量备份2 中创建的表
| test_purge          |
+---------------------+
3 rows in set (0.00 sec)
```

1.3.7. xtrabackup完全备份 – 压缩

> 官方文档

之前的备份如果要压缩需要手动去进行tar压缩，而 `innobackupex` 可以通过 `stream` 的方式（流的方式），然后通过 管道 进行压缩，或者直接使用 `--compress` 进行自动压缩

```
## 备份压缩示例
shell> innobackupex --stream=tar ./  | gzip - > backup.tar.gz
shell> innobackupex --stream=tar ./  | bzip2 - > backup.tar.bz2
```

> 以上两种方式无法实现并行备份和压缩
> **`xbstream` 的方式可以实现并行备份和压缩**

```
## 备份压缩示例
shell> innobackupex --compress --compress-threads=8 --stream=xbstream --parallel=4 ./ > backup.xbstream
## --parallel            表示有多少个线程拷贝表空间文件
## --compress-threads    表示有多少个线程进行压缩


## 使用innobackupex做一次全备
[root@MyServer new_data]> innobackupex --defaults-file=./my_5711.cnf  --user=root --password=123 --compress --compress-threads=4 --stream=xbstream --parallel=4  ./backup_test  > ./backup_test/backup.xbstream
# 输出信息同之前类似，省略

[root@MyServer backup_test]> ll -h | grep stream
-rw-r--r--. 1 root root 1.4G Mar  8 17:31 backup.xbstream
```

1.3.8. xtrabackup完全恢复 – 压缩

```
## 解压缩
[root@MyServer new_data]> mkdir output
[root@MyServer backup_test]> xbstream -x < backup.xbstream -C output/
[root@MyServer backup_test]> ll output/
total 134452
-rw-r-----. 1 root root      403 Mar  8 17:39 backup-my.cnf.qp
drwxr-x---. 2 root root     4096 Mar  8 17:39 burn_test
drwxr-x---. 2 root root     4096 Mar  8 17:39 dbt3
drwxr-x---. 2 root root     4096 Mar  8 17:39 employees
-rw-r-----. 1 root root    69142 Mar  8 17:39 ib_buffer_pool.qp
-rw-r-----. 1 root root   725279 Mar  8 17:39 ibdata1.qp
drwxr-x---. 2 root root     4096 Mar  8 17:39 mysql
drwxr-x---. 2 root root     4096 Mar  8 17:39 performance_schema
drwxr-x---. 2 root root    12288 Mar  8 17:39 sys
-rw-r-----. 1 root root 45984404 Mar  8 17:39 undo001.qp
-rw-r-----. 1 root root 45421820 Mar  8 17:39 undo002.qp
-rw-r-----. 1 root root 45403650 Mar  8 17:39 undo003.qp
-rw-r-----. 1 root root      160 Mar  8 17:39 xtrabackup_binlog_info.qp
-rw-r-----. 1 root root      119 Mar  8 17:39 xtrabackup_checkpoints
-rw-r-----. 1 root root      611 Mar  8 17:39 xtrabackup_info.qp
-rw-r-----. 1 root root      591 Mar  8 17:39 xtrabackup_logfile.qp
## 以上文件通过qpress进行了压缩，需要使用--decompress进行解压缩
```

在解压前，需要下载qpress，然后将解压出的二进制文件qpress放到 `/usr/local/bin` ，或者其他系统的 `PATH` 中去即可，在解压的时候，innobackupex会调用该程序进行解压缩

```
[root@MyServer backup_test]> innobackupex  --decompress output/
160308 17:41:11 innobackupex: Starting the decrypt and decompress operation

IMPORTANT: Please check that the decrypt and decompress run completes successfully.
           At the end of a successful decrypt and decompress run innobackupex
           prints "completed OK!".

innobackupex version 2.4.1 based on MySQL server 5.7.10 Linux (x86_64) (revision id: a2dc9d4)
160308 17:41:11 [01] decompressing ./xtrabackup_logfile.qp
160308 17:41:11 [01] decompressing ./burn_test/db.opt.qp
160308 17:41:11 [01] decompressing ./burn_test/test_purge.ibd.qp
160308 17:41:11 [01] decompressing ./burn_test/test_purge.frm.qp
160308 17:41:11 [01] decompressing ./ibdata1.qp
160308 17:41:11 [01] decompressing ./employees/current_dept_emp.frm.qp
160308 17:41:11 [01] decompressing ./employees/dept_emp.ibd.qp
## -----------省略部分输出------------
160308 17:41:33 [01] decompressing ./undo001.qp
160308 17:41:33 completed OK!

## 至此解压成功
```

1.3.9. xtrabackup增量备份 – 压缩

先重新做一次 未压缩的全备 ，然后 基于该全备 ，做 压缩的增量备份 （ *全备过程省略，也不做新数据的插入，只是演示命令* ）

```
[root@MyServer backup_test]> innobackupex --defaults-file=./my_5711.cnf  --user=root --password=123 --compress --compress-threads=4 --stream=xbstream --parallel=4  --incremental --incremental-basedir=/new_data/backup_test/2016-03-16_09-49-24/ ./backup_test  > ./backup_test/inc/backup_inc_1.xbstream
## -----------省略部分输出------------
xtrabackup: Transaction log of lsn (6668450360) to (6668450369) was copied.
160316 09:56:43 completed OK!
```

> 但是这样做有一个问题， 没法在这个增量备份的基础上再做一次增量备份 ，因为LSN信息被打包在xbstream中了

基于上述原因， 需要把 LSN信息 与 xbstream备份 进行 分离
使用 `--extra-lsndir` 参数，将checkpoint/LSN信息独立成一个文件

```
[root@MyServer backup_test]> innobackupex --defaults-file=./my_5711.cnf  --user=root --password=123 --compress --compress-threads=4 --stream=xbstream --parallel=4  --incremental --incremental-basedir=/new_data/backup_test/2016-03-16_09-49-24/ --extra-lsndir=/new_data/backup_test/inc/inc_1/LSN_INFO ./backup_test  > ./backup_test/inc/inc_1/backup_inc_1.x
bstream
## -----------省略部分输出------------
xtrabackup: Transaction log of lsn (6668450360) to (6668450369) was copied.
160316 10:10:08 completed OK!
```

```
[root@MyServer inc_1]> ll
total 376
-rw-r--r--. 1 root root 378646 Mar 16 10:10 backup_inc_1.xbstream
drwxr-xr-x. 2 root root   4096 Mar 16 10:11 LSN_INFO
[root@MyServer inc_1]> cat LSN_INFO/xtrabackup_checkpoints # 这个就是独立出来的lsn信息
backup_type = incremental
from_lsn = 6668450360
to_lsn = 6668450360    # 此次备份到的LSN
last_lsn = 6668450369
compact = 0
recover_binlog_info = 0
```

基于上次压缩的增量备份，进行第二次增量备份

```
-- 插入新的表
mysql> create table test_inc_3(a int primary key);
Query OK, 0 rows affected (0.13 sec)
```

`--incremental-basedir` 需要指向上一次增量备份中 `xtrabackup_checkpoints` 所在的路径，即 `/new_data/backup_test/inc/inc_1/LSN_INFO`

再通过 `--extra-lsndir` 指向新的目录，将此次的 `xtrabackup_checkpoints` 单独保存，为下一次增量更新提供 LSN 信息

```
[root@MyServer inc_1]> innobackupex --defaults-file=./my_5711.cnf  --user=root --password=123 --compress --compress-threads=4 --stream=xbstream --parallel=4  --incremental --incremental-basedir=/new_data/backup_test/inc/inc_1/LSN_INFO/ --extra-lsndir=/new_data/backup_test/inc/inc_2/LSN_INFO ./backup_test  > ./backup_test/inc/inc_2/backup_inc_2.xbstream
## -----------省略部分输出------------
xtrabackup: Transaction log of lsn (6668454747) to (6668454756) was copied.
160316 10:21:13 completed OK!


[root@MyServer inc_2]> ll
total 400
-rw-r--r--. 1 root root 402155 Mar 16 10:21 backup_inc_2.xbstream
drwxr-x---. 2 root root   4096 Mar 16 10:21 LSN_INFO
[root@MyServer inc_2]> cat LSN_INFO/xtrabackup_checkpoints
backup_type = incremental
from_lsn = 6668450360  # 紧接着上一次的LSN，开始备份
to_lsn = 6668454747
last_lsn = 6668454756
compact = 0
recover_binlog_info = 0
```

> 我们第一次全备的 base-dir 是 没有压缩 过的，如果增量备份要 基于压缩的全备 ，需要将全备的LSN信息也 分离 出来

```
[root@MyServer new_data]> innobackupex --defaults-file=./my_5711.cnf  --user=root --password=123 --compress --compress-threads=4 --stream=xbstream --parallel=4  --extra-lsndir=/new_data/backup_test/full/LSN_INFO ./backup_test  > ./backup_test/full/backup_full.xbstream
```

```
[root@MyServer full]> ll
total 1376760
-rw-r--r--. 1 root root 1409792960 Mar 16 10:34 backup_full.xbstream
drwxr-x---. 2 root root       4096 Mar 16 10:34 LSN_INFO
[root@MyServer full]> cat LSN_INFO/xtrabackup_checkpoints
backup_type = full-backuped
from_lsn = 0
to_lsn = 6668454747
last_lsn = 6668454756
compact = 0
recover_binlog_info = 0
```

> 第一个 压缩的 增量备份 的 `--incremental-basedir` 参数，指向 `/new_data/backup_test/full/LSN_INFO` 即可
> 后续的增量备份指向前一个增量备份的 `xtrabackup_checkpoints` 的路径

1.3.10. xtrabackup增量恢复 – 压缩

压缩的增量恢复其实和非压缩的增量恢复一样，只是多一步解压缩的操作

1. **压缩增量备份 解压**

```
# 压缩备份1
[root@MyServer inc_1]> mkdir extract
[root@MyServer inc_1]> xbstream -x < backup_inc_1.xbstream  -C extract/
[root@MyServer inc_1]> innobackupex  --decompress extract/

# 压缩备份2
[root@MyServer inc_2]> mkdir extract
[root@MyServer inc_2]> xbstream -x < backup_inc_2.xbstream  -C extract/
[root@MyServer inc_2]> innobackupex  --decompress extract/
```

> 如果第一次全备数据也是压缩的，还需要把全备的xbstream数据进行解压缩

2. **增量恢复**
   解压完成后，和之前的增量恢复就一样了（5个步骤）

```
# 步骤1
[root@MyServer new_data]> innobackupex  --defaults-file=./my_5711_2.cnf  --user=root --password=123  --apply-log --redo-only ./backup_test/2016-03-16_09-49-24/

# 步骤2
[root@MyServer new_data]> innobackupex  --defaults-file=./my_5711_2.cnf  --user=root --password=123  --apply-log --redo-only /new_data/backup_test/2016-03-16_09-49-24/ --incremental-dir=/new_data/backup_test/inc/inc_1/extract/

# 步骤3  -- 注意 不要 redo-only
[root@MyServer new_data]> innobackupex  --defaults-file=./my_5711_2.cnf  --user=root --password=123  --apply-log  /new_data/backup_test/2016-03-16_09-49-24/ --incremental-dir=/new_data/backup_test/inc/inc_2/extract/

# 步骤4
[root@MyServer new_data]> innobackupex  --defaults-file=./my_5711_2.cnf  --user=root --password=123  --apply-log ./backup_test/2016-03-16_09-49-24/

# 步骤5
[root@MyServer new_data]> rm -rf /data/mysql_data/5.7.11_2/*
[root@MyServer new_data]> innobackupex  --defaults-file=./my_5711_2.cnf  --user=root --password=123  --copy-back ./backup_test/2016-03-16_09-49-24/
```

3. **验证压缩的增量恢复**

```
[root@MyServer mysql_5_7_11]> bin/mysql_ssl_rsa_setup  --datadir=/data/mysql_data/5.7.11_2/ --user=mysql --uid=mysql

[root@MyServer mysql_5_7_11]> chown mysql.mysql /data/mysql_data/5.7.11_2/ -R

[root@MyServer mysql_5_7_11]> mysqld_multi  start 57112

[root@MyServer mysql_5_7_11]# mysql -u root -p -S /tmp/mysql.sock_59
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 3
Server version: 5.7.11-log MySQL Community Server (GPL)

Copyright (c) 2000, 2016, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> use burn_test;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> show tables;
+---------------------+
| Tables_in_burn_test |
+---------------------+
| test_inc            |
| test_inc_2          |
| test_inc_3          |  ## 新增加的 test_inc_3
| test_purge          |
+---------------------+
4 rows in set (0.00 sec)
```

1.3.11. xtrabackup部分备份以及恢复

xtrabackup备份部分库和表 官方文档

> 注意事项：
>
> 1. `--include` 参数目前没有实验成功
> 2. `--tables-file` 后面的路径要是 绝对路径 ，且备份成功
> 3. `--databases` 备份成功

```
--
-- [mysql5711]实例中新建一个 burn_test_2 的库
--
mysql> create database burn_test_2;
Query OK, 1 row affected (0.02 sec)

mysql> use burn_test_2
Database changed
mysql> create table test_backup1(a int primary key);
Query OK, 0 rows affected (0.15 sec)

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

- **单独备份新的数据库**

```
[root@MyServer new_data]> innobackupex  --defaults-file=./my_5711_2.cnf  --user=root --password=123  --databases="burn_test_2" /new_data/backup_test/single_database/


[root@MyServer 2016-03-16_12-31-13]> pwd
/new_data/backup_test/single_database/2016-03-16_12-31-13
[root@MyServer 2016-03-16_12-31-13]> ll
total 266268
-rw-r-----. 1 root root      412 Mar 16 12:31 backup-my.cnf
drwxr-x---. 2 root root     4096 Mar 16 12:31 burn_test_2  # 单独备份出来的数据库
-rw-r-----. 1 root root     2345 Mar 16 12:31 ib_buffer_pool
-rw-r-----. 1 root root 12582912 Mar 16 12:31 ibdata1
-rw-r-----. 1 root root 83886080 Mar 16 12:31 undo001
-rw-r-----. 1 root root 88080384 Mar 16 12:31 undo002
-rw-r-----. 1 root root 88080384 Mar 16 12:31 undo003
-rw-r-----. 1 root root      101 Mar 16 12:31 xtrabackup_binlog_info
-rw-r-----. 1 root root      119 Mar 16 12:31 xtrabackup_checkpoints
-rw-r-----. 1 root root      660 Mar 16 12:31 xtrabackup_info
-rw-r-----. 1 root root     2560 Mar 16 12:31 xtrabackup_logfile
```

- **还原单独备份数据库**

```
[root@MyServer new_data]> innobackupex  --defaults-file=./my_5711_2.cnf  --user=root --password=123 --apply-log --export  /new_data/backup_test/single_database/2016-03-16_12-31-13
#
# InnoDB: Failed to find tablespaceXXXXXX
# 备份过程中会出现上述类似的错误，这个是正常的，因为我们是备份部分数据，所以有些表空间是找不到的

## -----------省略部分输出------------

InnoDB: Shutdown completed; log sequence number 6668467752
160316 12:38:21 completed OK!


[root@MyServer burn_test_2]> pwd
/new_data/backup_test/single_database/2016-03-16_12-31-13/burn_test_2
[root@MyServer burn_test_2]> ll
total 76
-rw-r-----. 1 root root    67 Mar 16 12:31 db.opt
-rw-r--r--. 1 root root   352 Mar 16 12:38 test_backup1.cfg # --export参数生成的文件
-rw-r-----. 1 root root  8192 Mar 16 12:38 test_backup1.exp # --export参数生成的文件
-rw-r-----. 1 root root  8554 Mar 16 12:31 test_backup1.frm
-rw-r-----. 1 root root 49152 Mar 16 12:31 test_backup1.ibd


#
# 安装一个新的数据库，用于还原
#
[root@MyServer mysql_5_7_11]> bin/mysqld --initialize --user=mysql --datadir=/data/mysql_data/5.7.11_2

[root@MyServer mysql_5_7_11]> bin/mysql_ssl_rsa_setup  --datadir=/data/mysql_data/5.7.11_2/ --user=mysql --uid=mysql

#
# copy 数据到新的库中
#
[root@MyServer 2016-03-16_12-31-13]> pwd
/new_data/backup_test/single_database/2016-03-16_12-31-13
[root@MyServer 2016-03-16_12-31-13]> cp ./* /data/mysql_data/5.7.11_2/ -r

#
# 修改权限
#
[root@MyServer 2016-03-16_12-31-13]> chown mysql.mysql /data/mysql_data/5.7.11_2/ -R

#
# 启动实例
#
[root@MyServer 2016-03-16_12-31-13]> mysqld_multi  start 57112

[root@MyServer 2016-03-16_12-31-13]> mysql -u root -p -S /tmp/mysql.sock_59  # 注意，由于是新安装的数据库，需要更新密码
Enter password:    # 新密码在 error.log 里面
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 3
Server version: 5.7.11-log

Copyright (c) 2000, 2016, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.
# 记得要修改密码
mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| burn_test_2        |  # 单独备份出来的数据库
| mysql              |
| performance_schema |
| sys                |
+--------------------+
5 rows in set (0.00 sec)

mysql> show tables;
+----------------------+
| Tables_in_burn_test_2 |
+----------------------+
| test_backup1         |  # 单独备份出来的数据库中的表
+----------------------+
1 row in set (0.00 sec)
```