MySQL学习笔记（Day035：redo_binlog_xa）

MySQL学习

一. binlog（二）

1.1. binlog_rows_query_log_events

打开该参数，可以在 row格式 下，看到对应的sql信息

```
mysql> show variables like "binlog_rows_query_log_events";
+-------------------------------+-------+
| Variable_name                 | Value |
+-------------------------------+-------+
| binlog_rows_query_log_events | OFF   | -- 默认是关闭的（from 5.6.2）
+-------------------------------+-------+
1 row in set (0.00 sec)

mysql> flush binary logs;
Query OK, 0 rows affected (0.12 sec)

mysql> create table test_bin_2 (a int);
Query OK, 0 rows affected (0.11 sec)

mysql> insert into test_bin_2 values(1),(2);
Query OK, 2 rows affected (0.00 sec)
Records: 2  Duplicates: 0  Warnings: 0

mysql> commit;
Query OK, 0 rows affected (0.04 sec)

mysql> show master status;
+------------+----------+--------------+------------------+-------------------------------------------+
| File       | Position | Binlog_Do_DB | Binlog_Ignore_DB | Executed_Gtid_Set                         |
+------------+----------+--------------+------------------+-------------------------------------------+
| bin.000058 |      650 |              |                  | c1f87a6a-98f2-11e5-b873-5254a03976fb:1-1838 |
+------------+----------+--------------+------------------+-------------------------------------------+
1 row in set (0.00 sec)

mysql> show binlog events in "bin.000058";
+------------+-----+----------------+-----------+-------------+-----------------------------------------------------------------------------+
| Log_name   | Pos | Event_type     | Server_id | End_log_pos | Info                                                                        |
+------------+-----+----------------+-----------+-------------+-----------------------------------------------------------------------------+
| bin.000058 |   4 | Format_desc    |      5709 |         123 | Server ver: 5.7.9-log, Binlog ver: 4                                        |
| bin.000058 | 123 | Previous_gtids |      5709 |         194 | c1f87a6a-98f2-11e5-b873-5254a03976fb:1-1836                                 |
| bin.000058 | 194 | Gtid           |      5709 |         259 | SET @@SESSION.GTID_NEXT= 'c1f87a6a-98f2-11e5-b873-5254a03976fb:1837'        |
| bin.000058 | 259 | Query          |      5709 |         374 | use `burn_test`; create table test_bin_2 (a int)                            |
| bin.000058 | 374 | Gtid           |      5709 |         439 | SET @@SESSION.GTID_NEXT= 'c1f87a6a-98f2-11e5-b873-5254a03976fb:1838'        |
| bin.000058 | 439 | Query          |      5709 |         516 | BEGIN                                                                       |
| bin.000058 | 516 | Table_map      |      5709 |         574 | table_id: 160 (burn_test.test_bin_2)                                        |
| bin.000058 | 574 | Write_rows     |      5709 |         619 | table_id: 160 flags: STMT_END_F                                             |
| bin.000058 | 619 | Xid            |      5709 |         650 | COMMIT /* xid=159 */                                                        |
+------------+-----+----------------+-----------+-------------+-----------------------------------------------------------------------------+
9 rows in set (0.00 sec)
-- 只能看到页的变化


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

mysql> show variables like "binlog_rows_query_log_events";
+-------------------------------+-------+
| Variable_name                 | Value |
+-------------------------------+-------+
| binlog_rows_query_log_events | ON    |
+-------------------------------+-------+
1 row in set (0.00 sec)

mysql> insert into test_bin_2 values(3),(4);
Query OK, 2 rows affected (0.00 sec)
Records: 2  Duplicates: 0  Warnings: 0

mysql> commit;
Query OK, 0 rows affected (0.03 sec)

mysql> show binlog events in "bin.000058";
+------------+-----+----------------+-----------+-------------+-----------------------------------------------------------------------------+
| Log_name   | Pos | Event_type     | Server_id | End_log_pos | Info                                                                        |
+------------+-----+----------------+-----------+-------------+-----------------------------------------------------------------------------+
| bin.000058 |   4 | Format_desc    |      5709 |         123 | Server ver: 5.7.9-log, Binlog ver: 4                                        |
| bin.000058 | 123 | Previous_gtids |      5709 |         194 | c1f87a6a-98f2-11e5-b873-5254a03976fb:1-1836                                 |
| bin.000058 | 194 | Gtid           |      5709 |         259 | SET @@SESSION.GTID_NEXT= 'c1f87a6a-98f2-11e5-b873-5254a03976fb:1837'        |
| bin.000058 | 259 | Query          |      5709 |         374 | use `burn_test`; create table test_bin_2 (a int)                            |
| bin.000058 | 374 | Gtid           |      5709 |         439 | SET @@SESSION.GTID_NEXT= 'c1f87a6a-98f2-11e5-b873-5254a03976fb:1838'        |
| bin.000058 | 439 | Query          |      5709 |         516 | BEGIN                                                                       |
| bin.000058 | 516 | Table_map      |      5709 |         574 | table_id: 160 (burn_test.test_bin_2)                                        |
| bin.000058 | 574 | Write_rows     |      5709 |         619 | table_id: 160 flags: STMT_END_F                                             |
| bin.000058 | 619 | Xid            |      5709 |         650 | COMMIT /* xid=159 */                                                        |
| bin.000058 | 650 | Gtid           |      5709 |         715 | SET @@SESSION.GTID_NEXT= 'c1f87a6a-98f2-11e5-b873-5254a03976fb:1839'        |
| bin.000058 | 715 | Query          |      5709 |         792 | BEGIN                                                                       |
| bin.000058 | 792 | Rows_query     |      5709 |         852 | # insert into test_bin_2 values(3),(4)                                      |  -- 可以看到对应的SQL语句，下面还是页的改动信息
| bin.000058 | 852 | Table_map      |      5709 |         910 | table_id: 160 (burn_test.test_bin_2)                                        |
| bin.000058 | 910 | Write_rows     |      5709 |         955 | table_id: 160 flags: STMT_END_F                                             |
| bin.000058 | 955 | Xid            |      5709 |         986 | COMMIT /* xid=165 */                                                        |
+------------+-----+----------------+-----------+-------------+-----------------------------------------------------------------------------+
15 rows in set (0.00 sec)

-- 多了Rows_query的类型，可以看到对应着SQL信息
```

1.2. ROW

1. 当写入的数据量 较小 时，ROW和Statement所占用的空间差不多；
2. 当写入的数据量 较大 的时候（比如导入数据，或者批量操作时*[update tb set a=a+1;]*），ROW要记录每行的变化，所以比较占用空间。
3. 且写入数据量很大时，ROW格式下，commit会比较耗时间，因为他还要写binlog（ binlog在提交时才写入 ）
    - *假设更新一张几百万的表，产生的binlog可能会有几百兆，当commit时，写入的数据量就是几百兆，所以会有"阻塞"等待的效果。但其实是在写binlog到磁盘而已。*

1.3. binlog_cache

binlog默认写入到 `binlog_cache` 中

```
mysql> show variables like "binlog_cache_size";
+-----------------+-------+
| Variable_name     | Value |
+-----------------+-------+
| binlog_cache_size | 32768 | -- 默认为32K（内存中），线程级别的变量，别设置的太大
+-----------------+-------+
1 row in set (0.00 sec)
```

当有一个大的事物时（*几百兆*），内存中显然放不下那么多binlog，所以会记录到磁盘上

```
mysql> show global status like "binlog%";
+-----------------------------+-------+
| Variable_name               | Value |
+-----------------------------+-------+
| Binlog_cache_disk_use       | 0     |  -- 记录了使用临时文件写二进制日志的次数（做监控需要关注这个）
| Binlog_cache_use            | 6     |  -- 记录了使用缓冲写二进制日志的次数
| Binlog_stmt_cache_disk_use | 0     |
| Binlog_stmt_cache_use       | 2     |
+-----------------------------+-------+
4 rows in set (0.00 sec)
```

> 写日志本来就挺慢的，现在cache写不下，再写入磁盘，然后再写binlog，就是两次写磁盘，就更慢了。
> 如果参数 `Binlog_cache_disk_use` 次数很多，就要看一下 `binlog_cache_size` 设置是否 太小 ，或者 事物本身 是否 太大
> MySQL使用在OLTP的场景下，应该是很快速的小事物。如果有大的事物，应该把 大的事物拆成小事物 去执行。

1.4. binlog与redo的一致性

使用 内部分布式事物 来保证一致性

在 `commit` 时（*无论用户自己输入，或者系统自动添加*），会有如下几个步骤：

1. `InnoDB` 层 写 `prepare log`
    - 写的还是 `redo file`（或者就是redo log，只是内容不一样，这里不是记录页的变化了）
    - 写入的是 `xid`（*事物id，show binlog events in "bin.000056"*）
        - 准确的说，`xid` 是写在 `undo` 页上的（后面会提到）
2. `MySQL` 层 写 `binlog`
3. `InnoDB` 层 写 `commit log`（这里同样也是redo log file）

注意：这里的写入是指写入到磁盘（落盘成功）

> 1. 假设，如果 没有 第一步的 `prepare log`，而是直接写第二步的 `MySQL binlog`，以及接着写第三步的 `InnoDB commit log`：
>   此时假设出现 `binlog写入成功`，而 `commit log(redo)写入失败` 的情况（*比如宕机*），那随后机器重启时 恢复 时，就会对该事物 回滚；
>   万一此时的 `binlog` 已经传递到了 `slave` 机器上，且 `slave上commit` 了。那此时 主从就不一致 了（*Master上回滚了*）
> 2. 现在有 `prepare log` 了以后，`prepare log写入成功`，假设还是 `binlog写入成功`，而 `commit log(redo)写入失败` 的情况下；
>   此时事物恢复的时候，`检查到prepare log写入成功，binlog写入成功`，那就直接 `commit` 了（*可以理解成补了那次失败的commit*），而 不管**`commit log`**是否写入成功 了。
> 3. 如果 `prepare log写入成功`，`binlog写入失败` 了，那恢复时，也会回滚
> 4. 如果 没开`binlog`，就没有第一和第二步，只写第三步的`commit log`，恢复的时候没有commit log，就会回滚。

- **一个事物在prepare log中写入成功，在binlog中写入成功，那就 必须要提交 （commit）**
- **用户/系统 `commit` ===> redo file（ `prepare` log ）===> `binlog` ===> redo file（ `commit` log ）**
- **`xid` 即 写入`prepare log` 中，也会 写入到`binlog` 中，当恢复时，会 对比 一下某个 `xid` 在两个文件中是否都存在，如果都存在，该xid对应的事物才会提交**

> 1. 在MySQL5.6以后，`写入binlog（步骤二）` 和 `写入commit log（步骤三）`，都是通过 组提交 的方式刷入（fsync）到磁盘的。
> 2. 在MySQL 5.7以后，写入 `prepare log（步骤一）` 也是通过 组提交 的方式刷入（fsync）到磁盘的（在写binlog之前执行一次fsync，就批量刷入prepare log）
>
> **注意：**组提交中失败了，并 不会回滚 该组中的 所有事物，而是哪个失败了，就回滚哪个。

二. 分布式事物

XA 官方文档-1
XA 官方文档-2

```
--
-- 终端会话1
--
mysql> create table test_xa_1(a int primary key);
Query OK, 0 rows affected (0.11 sec)

mysql> xa start 'A';   -- 开始一个分布式事物A，不是传统的begin。而是 xa start
Query OK, 0 rows affected (0.00 sec)

mysql> insert into test_xa_1 values(10);
Query OK, 1 row affected (0.00 sec)

mysql> xa end 'A';  -- 结束一个分布式事物A，此时并未提交
Query OK, 0 rows affected (0.00 sec)

mysql> xa prepare 'A';  -- 两阶段事物 - prepare
Query OK, 0 rows affected (0.02 sec)

mysql> xa recover;  -- 查看分布式事物
+----------+--------------+--------------+------+
| formatID | gtrid_length | bqual_length | data |
+----------+--------------+--------------+------+
|        1 |            1 |            0 | A    |
+----------+--------------+--------------+------+
1 row in set (0.00 sec)

--
-- 终端会话2
--
mysql> select * from test_xa_1;  -- 虽然在会话1中已经 end 了，但是其实在会话2中是看不到的，符合ACID
Empty set (0.00 sec)

--
-- 终端会话1
--
mysql> xa commit 'A';  -- xa commit 才是提交分布式事物
Query OK, 0 rows affected (0.02 sec)

--
-- 终端会话2
--
mysql> select * from test_xa_1;
+----+
| a  |
+----+
| 10 |  -- 会话1提交后，会话2中能看到插入的数据
+----+
1 row in set (0.00 sec)


XA START xid   # 开启一个分布式事物

XA END xid     # 结束一个分布式事物

XA PREPARE xid  # 将分布式事物变成prepare状态

XA COMMIT xid   # 提交一个分布式事物

XA ROLLBACK xid  # 回滚一个分布式事物

XA RECOVER       # 查看分布式事物
```

> 上述在单实例中操作分布式事物其实是没有意义的，仅仅作为一个语法的演示。
>
> 1. 分布式事物是 串行 执行的（不能快照读），在分布式事物中，使用的是两阶段事物，如果prepare成功了，就一定要提交。
> 2. 如果发生commit失败，事物就变成了 悬挂 事物，需要人工介入，查看prepare是否成功，而后决定commit 或者 rollback