注意事项

参考资料文章提到在 MariaDB 中，子查询有 group by 分组操作时能用到 Semi-join Materialization 优化策略（其他的 Duplicate Weedout、FirstMatch、LooseScan 不能用）。而在 MySQL 中，子查询有 group by 分组操作时所有的 Semi-join 策略都无法使用，即无法使用 Semi-join 优化，举例：

```
select dept_name from departments where dept_no in \
(select min(dept_no) from dept_emp where emp_no<10020 group by dept_no);
+----+-------------+-------------+-------+----------------+----------+---------+-
| id | select_type | table       | type  | possible_keys  | key      | key_len |
   rows | Extra                                          |
+----+-------------+-------------+-------+----------------+----------+---------+-
|  1 | PRIMARY     | departments | index | NULL           | dept_name | 42     |    9
   | Using where; Using index                          |
|  2 | SUBQUERY    | dept_emp    | range | PRIMARY,dept_no | PRIMARY  | 4      |
   21 | Using where; Using index; Using temporary; Using filesort |
+----+-------------+-------------+-------+----------------+----------+---------+-
```

可以看到这里使用的是 Non-semijoin materialization 优化策略，也就是 MySQL 子查询优化 文中的 Materialization 优化策略。所以 optimizer_switch 参数中的 materialization=on 标志也可以单独用于 Non-semijoin materialization 优化策略。

3.12 基于 GTID 的多源复制

作者：马文斌

一、背景

有 4 个地区工厂的数据，需要同步到 idc。之前有个方案是用阿里的 otter 管理平台去同步到 idc 机房。运行一段时间过后，发现 otter 平台会不断的往 idc 发包，建立几百个空连接，这样导致 idc 的网络造成拥堵。

后来经过讨论，还是采用 MySQL 源生自带的主从复制方案，那源生自带的主从复制方案又有两种：一种是 GTID；一种是 pos 位点信息。那为什么要采用 GTID 复制呢？肯定有他的优势。

这里简单列举了几点：

1. 如果是单源复制的情况下，可以很方便的搭建主从；
    - 为什么方便？比如只需要开启 master_auto_Position=1 即可；
2. 基于 GTID 的复制可以忽略已经执行过的事务，减少了数据发生不一致的风险；

3. 避免因为设置位点信息不准确而造成主从不一致的情况。

下面开始做多源复制的操作。

二、服务器情况说明

ip	database	角色	备注
192.168.100.1	db01	ims_guangzhou	广州工厂 master_01
192.168.100.2	db02	ims_chongqing	重庆工厂 master_02
192.168.100.3	db03	ims_tianjin	天津工厂 master_03
192.168.100.4	db04	ims_kunshan	昆山工厂 master_04
192.168.100.5	db01,db03,db04,db02	idc_slave	4 个工厂数据的汇总 从库

MySQL 版本：8.0.18

三、导出数据

导出 4 个工厂的数据到 192.168.100.5，在该服务器上运行：

```
cd /data/backup
mysqldump -uroot -p -h192.168.100.1 --master-data=2 --single-transaction
    db01 >db01.sql
mysqldump -uroot -p -h192.168.100.2 --master-data=2 --single-transaction
    db02 >db02.sql
mysqldump -uroot -p -h192.168.100.3 --master-data=2 --single-transaction
    db03 >db03.sql
mysqldump -uroot -p -h192.168.100.4 --master-data=2 --single-transaction
    db04 >db04.sql
```

四、设置 GTID 值

4.1 导入数据

这里只先导入 3 个实例的数据，然后开启主从，最后再添加一个，我们看看这样是否方便添加。

4.2 导入三个实例

```
use db01;
source /data/backup/db01.sql;
use db03;
source /data/backup/db03.sql;
use db04;
source /data/backup/db04.sql;
stop slave ;
```

```
reset slave all;
reset master;
# 这个命令会把本地的 binlog 给清理掉，慎用
show master status \G
            File: mysql-bin.000001
        Position: 155
     Binlog_Do_DB:
 Binlog_Ignore_DB:
Executed_Gtid_Set:       # GTID 为空
```

4.3 设置 GTID

```
SET @@GLOBAL.GTID_PURGED=/*!80000 '+'*/ '39e42c4d-876f-11ea-8229-286ed488e793:1-31,
8a426026-b5e7-11ea-8816-0050568399c4:1-62183,f3d8b026-ba76-11ea-985d-000c29b82d1f:1-
    36244,39e42c4d-876f-11ea-8229-286ed488e793:1-31,
5d7ef438-f249-11ea-a518-0894ef181fcf:1-5980,
8a426026-b5e7-11ea-8816-0050568399c4:1-56548';
```

4.4 设置主从关系

```
change master to
    master_host='192.168.100.1',master_user='repl',master_password='123456',master_au
    to_Position=1 for channel 'ims_guangzhou';

change master to
    master_host='192.168.100.3',master_user='repl',master_password='123456',master_au
    to_Position=1 for channel 'ims_tianjin';

change master to
    master_host='192.168.100.4',master_user='repl',master_password='123456',master_au
    to_Position=1 for channel 'ims_kunshan';

start slave;
show slave status\G

# 在 100.1 广州工厂建一个测试表试下，能不能同步
use db01;
create table tb1(id int primary key);
```

在 100.5 的 db01 的库上是可以查的，说明可以同步。

五、尝试添加从库

为什么要尝试添加一个从库，这里主要是测试下，看看添加一个从库，麻不麻烦？如果不麻烦的话，还是可以用 GTID 的模式的。

5.1 找到最后一个库的 GTID

```
SET @@GLOBAL.GTID_PURGED=/*!80000 '+'*/ 'a97612c1-b947-11ea-bcf9-005056812835:1-
    19065';
```

5.2 终止 SLAVE

```
stop slave;
```

5.3 加入其他的 GTID

先把主库的 binlog 先备份下，

```
/data/mysqlLog/logs
mkdir binlogbak
cp mysql-bin.000001 binlogbak/

# 设置多个 GTID
SET @@GLOBAL.GTID_PURGED=/*!80000 '+'*/ 'a97612c1-b947-11ea-bcf9-005056812835:1-
    19065,
39e42c4d-876f-11ea-8229-286ed488e793:1-31,
5d7ef438-f249-11ea-a518-0894ef181fcf:1-6107,
8a426026-b5e7-11ea-8816-0050568399c4:1-62290,
f3d8b026-ba76-11ea-985d-000c29b82d1f:1-36379'
```

5.4 添加从库

```
change master to
    master_host='192.168.100.2',
    master_user='repl',
    master_password='123456',
    master_auto_Position=1 for channel 'ims_chongqing';
```

5.5 查看是否同步

```
# 在 100.1 上删除 tb1
use db01;
drop table tb1;

# 在 100.5 上查看是否同步：
use db01;
show tables;
+--------------------------------+
| Tables_in_db01 |
+--------------------------------+
| equitment_info                |
| event_code_info               |
```

```
| ims_monitor                  |
| oee_data                     |
| oee_process_re               |
+------------------------------+
```

表不见了，说明已经同步。

六、总结

GTID 对于单源复制还是很方便，但是对于多源复制，这里就需要特别注意：

- 要先停止所有的从库 stop slave;
- 然后清理本机所有的 GTID，reset master;
- 再进行 SET @@GLOBAL.GTID_PURGED='xxxxx' gtid 设置

这里就会引入一个问题，如果是级联复制的情况下，reset master 的时候，会把本机的所有 binlog 清理掉。如果下一级的从库存在延迟，没有及时的把 binlog 传过去，就会造成主从中断，这里我们该怎么避免呢？看这里：

- 做 reset master 的时候，先看看下游的从库是否存在很大的延迟。如果存在，把当前的 binlog 和后面未同步的 binlog 全部备份下；
- 待添加好从库的 channel 后，再把未同步的 binlog 文件手动拷贝到 binlog 目录；
- 更新下 mysql-bin.index 文件；
- 注意，binlog 不能同名，需要手动更新下文件。