# mysql binlog Flashback测试报告

使用sysbench工具生成10000条数据：

sysbench /usr/share/sysbench/oltp\_read\_write.lua --db-driver=mysql --mysql-host=localhost --mysql-port=3306 --mysql-user=root --mysql-socket=/data/mysqldata/mysql.sock --mysql-db=test --table\_size=10000 --tables=1 prepare

查看当前二进制日志，并刷新：

root@localhost : test 10:03:48>show binary logs;

+------------------+-----------+

| Log\_name | File\_size |

+------------------+-----------+

| mysql-bin.000001 | 45105791 |

| mysql-bin.000002 | 14078592 |

| mysql-bin.000003 | 47748613 |

+------------------+-----------+

3 rows in set (0.00 sec)

root@localhost : test 10:06:33>flush logs;

Query OK, 0 rows affected (0.16 sec)

root@localhost : test 10:07:24>show binary logs;

+------------------+-----------+

| Log\_name | File\_size |

+------------------+-----------+

| mysql-bin.000001 | 45105791 |

| mysql-bin.000002 | 14078592 |

| mysql-bin.000003 | 47748660 |

| mysql-bin.000004 | 191 |

+------------------+-----------+

4 rows in set (0.00 sec)

root@localhost : test 10:07:27>

sbtest1的表结构如下：

root@localhost : test 10:13:18>show create table sbtest1\G

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 1. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Table: sbtest1

Create Table: CREATE TABLE `sbtest1` (

 `id` int(11) NOT NULL AUTO\_INCREMENT,

 `k` int(11) NOT NULL DEFAULT '0',

 `c` char(120) NOT NULL DEFAULT '',

 `pad` char(60) NOT NULL DEFAULT '',

PRIMARY KEY (`id`),

KEY `k\_1` (`k`)

) ENGINE=InnoDB AUTO\_INCREMENT=50001 DEFAULT CHARSET=utf8

1 row in set (0.00 sec)

root@localhost : test 10:13:31>

第一种情况：对sbtest1表中的c字段进行了无where条件的更新，且后期表中再无任何操作。

更新语句如下：

root@localhost : test 10:14:28>update sbtest1 set c='xihui';

Query OK, 50000 rows affected (2.02 sec)

Rows matched: 50000 Changed: 50000 Warnings: 0

root@localhost : test 10:15:41>

更新后的数据为：

root@localhost : test 10:16:28>select \* from sbtest1 limit 10;



使用binlogflashback进行binlog解析，找到全表更新开始与结束时的position点：

[root@hadoop103 ~]# binlogflashback -v /data/mysqldata/mysql-bin.000004 > /home/xihui.zhang/mysql01.sql

找到事务开始点为311，结束点为13360380，使用-B参数对update进行逆向操作：

[root@hadoop103 ~]# binlogflashback -v -B --start-position=311 --stop-position=13360380 /data/mysqldata/mysql-bin.000004 > /home/xihui.zhang/flashback01.sql

source反转后的sql文件：

root@localhost : test 10:32:51>source /home/xihui.zhang/flashback01.sql;

Query OK, 0 rows affected (0.00 sec)

Query OK, 0 rows affected (0.00 sec)

Query OK, 0 rows affected (0.00 sec)

Query OK, 0 rows affected (2.55 sec)

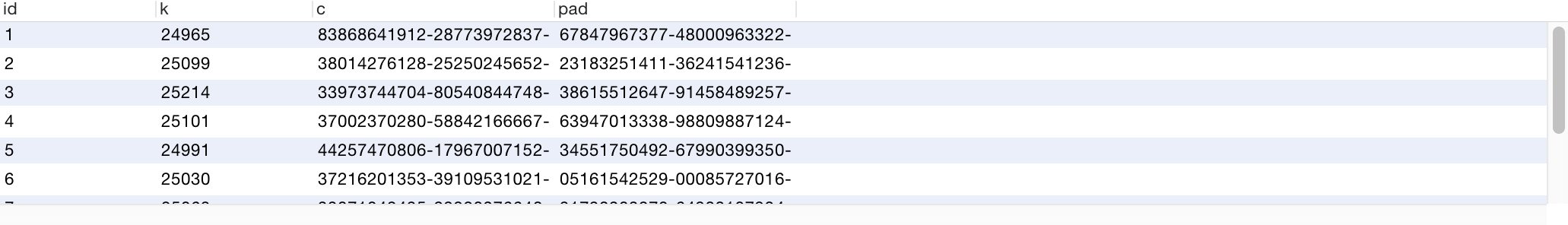
Query OK, 0 rows affected (0.00 sec)

Query OK, 0 rows affected (0.00 sec)

root@localhost : test 10:33:13>

回退后的数据为：

root@localhost : test 10:33:51>select \* from sbtest1 limit 10;



从结果可以看出，数据回退到了全表更新之前的状态了。

第二种情况，全表更新后，对表中的数据进行了delete操作。

语句如下：

root@localhost : test 10:53:31>update sbtest1 set c='xihui.zhang';

Query OK, 10000 rows affected (0.06 sec)

Rows matched: 10000 Changed: 10000 Warnings: 0

root@localhost : test 10:53:50>delete from sbtest1 where id between 500 and 1000;

Query OK, 501 rows affected (0.00 sec)

使用flashback进行binlog解析，找到全表更新开始与结束时的position点：

[root@DB160-25 mysqldata]# flashback -v /data/mysqldata/mysql-bin.000014 > /home/qihang.li/0014.sql

找到事务开始点为123，结束点为2773960，使用-B参数对 update和delete进行逆向操作：

[root@DB160-25 mysqldata]# flashback -v -B --start-position=123 --stop-position=2773960 --skip-gtids /data/mysqldata/mysql-bin.000014 > /home/qihang.li/0014\_backup\_allupdate\_delte\_test.sql

source反转后的sql文件：

root@localhost : test 10:54:12>source /home/qihang.li/0014\_backup\_allupdate\_delte\_test.sql

ERROR 2006 (HY000): MySQL server has gone away

No connection. Trying to reconnect...

Connection id: 42

Current database: test

Query OK, 0 rows affected (0.00 sec)

Query OK, 0 rows affected (0.00 sec)

Query OK, 0 rows affected (0.00 sec)

Query OK, 0 rows affected (0.00 sec)

Query OK, 0 rows affected (0.01 sec)

Query OK, 0 rows affected (0.12 sec)

Query OK, 0 rows affected (0.00 sec)

Query OK, 0 rows affected (0.00 sec)

root@localhost : test 11:00:01>

执行source时，存在ERROR，查看数据，发现回退成功

root@localhost : test 11:00:23>select count(\*) from sbtest1 where c='xihui.zhang';

+----------+

| count(\*) |

+----------+

| 0 |

+----------+

1 row in set (0.00 sec)

root@localhost : test 11:00:28>select count(\*) from sbtest1 where id between 500 and 1000;

+----------+

| count(\*) |

+----------+

| 501 |

+----------+

1 row in set (0.00 sec)

root@localhost : test 11:00:40>

从结果可以看出，数据已经闪回到全表更新前的数据

第三种情况，全表更新后，有insert操作。

语句如下：

root@localhost : test 10:18:31>insert into sbtest1(k,c,pad) values(123,'123','123');

Query OK, 1 row affected (0.00 sec)

root@localhost : test 10:18:43>insert into sbtest1(k,c,pad) values(1234,'1234','1234');

Query OK, 1 row affected (0.00 sec)

root@localhost : test 10:18:45>update sbtest1 set c='qihang.li';

Query OK, 10002 rows affected (0.06 sec)

Rows matched: 10002 Changed: 10002 Warnings: 0

root@localhost : test 10:18:48>select count(\*) from sbtest1 where c='qihang.li' ;

+----------+

| count(\*) |

+----------+

| 10002 |

+----------+

1 row in set (0.00 sec)

root@localhost : test 10:20:08>select \* from sbtest1 where c in ('123','1234');

Empty set (0.00 sec)

由于舒心了binary log，所以直接使用日志文件即可，不需要找position点。

[root@DB160-25 qihang.li]# flashback -vv -B --skip-gtids /data/mysqldata/mysql-bin.000010 > /home/qihang.li/0010\_backup\_update\_insert.sql

source反转后的sql文件：

root@localhost : test 10:22:28>source /home/qihang.li/0011\_backup\_update\_insert\_delete.sql

Query OK, 0 rows affected (0.00 sec)

Query OK, 0 rows affected (0.00 sec)

Query OK, 0 rows affected (0.00 sec)

Query OK, 0 rows affected (0.00 sec)

Query OK, 0 rows affected (0.05 sec)

Query OK, 0 rows affected (0.00 sec)

Query OK, 0 rows affected (0.00 sec)

Query OK, 0 rows affected (0.12 sec)

Query OK, 0 rows affected (0.00 sec)

Query OK, 0 rows affected (0.00 sec)

root@localhost : test 10:23:04>

查看回退后的数据：

root@localhost : test 10:27:10>select \* from sbtest1 where c in ('123','1234');

ERROR 2006 (HY000): MySQL server has gone away

No connection. Trying to reconnect...

Connection id: 39

Current database: test

Empty set (0.00 sec)

root@localhost : test 10:28:55>select count(\*) from sbtest1 where id between 100 and 5000;

+----------+

| count(\*) |

+----------+

| 4901 |

+----------+

1 row in set (0.00 sec)

root@localhost : test 10:29:37>select count(\*) from sbtest1 where c='qihang.li'

-> ;

+----------+

| count(\*) |

+----------+

| 0 |

+----------+

1 row in set (0.00 sec)

root@localhost : test 10:30:02>

从结果可以看出，数据回退到了全表更新之前的状态。

总结：

在误将全表更新后，务必将 所有误操作的日志文件都使用-B参数进行闪回，并将闪回文件 倒序（mysql-bin.000011 → mysql-bin.000010）进行恢复（source），这样才能保证数据的正确性。