一、binlog解析简单说明

MySQL binlog以event的形式，记录了MySQL server从启用binlog以来所有的变更信息，能够帮助重现这之间的所有变化。MySQL引入binlog主要有两个目的：一是为了主从复制；二是某些备份还原操作后需要重新应用binlog。

有三种可选的binlog格式，各有优缺点：

\* statement：基于SQL语句的模式，binlog数据量小，但是某些语句和函数在复制过程可能导致数据不一致甚至出错；

\* row：基于行的模式，记录的是行的完整变化。很安全，但是binlog会比其他两种模式大很多；

\* mixed：混合模式，根据语句来选用是statement还是row模式；

利用binlog闪回，\*\*需要将binlog格式设置为row\*\*。

从MySQL binlog解析出你要的SQL。根据不同选项，你可以得到原始SQL、回滚SQL、去除主键的INSERT SQL等。

二、安装

依赖文件及其最低版本：

PyMySQL==0.7.11

wheel==0.29.0

mysql-replication==0.13

下载后用pip安装，pip install ......gz.

pip软件先安装，./configure.....

MySQL server必须设置以下参数:

[mysqld]

server\_id = 1

log\_bin = mysql-bin

max\_binlog\_size = 1G

binlog\_format = row

binlog\_row\_image = full

user需要的最小权限集合：

select, super/replication client, replication slave

建议授权

GRANT SELECT, REPLICATION SLAVE, REPLICATION CLIENT ON \*.\* TO

权限说明

select：需要读取server端information\_schema.COLUMNS表，获取表结构的元信息，拼接成可视化的sql语句

super/replication client：两个权限都可以，需要执行'SHOW MASTER STATUS', 获取server端的binlog列表

replication slave：通过BINLOG\_DUMP协议获取binlog内容的权限

三、基本选项

mysql连接配置

-h host; -P port; -u user; -p password

解析模式

--stop-never 持续解析binlog。可选。，默认False，同步至执行命令时最新的binlog位置。

-K, --no-primary-key 对INSERT语句去除主键。可选。默认False

-B, --flashback 生成回滚SQL，可解析大文件，不受内存限制。可选。默认False。与stop-never或no-primary-key不能同时添加。

--back-interval -B模式下，每打印一千行回滚SQL，加一句SLEEP多少秒，如不想加SLEEP，请设为0。可选。默认1.0。

解析范围控制

--start-file 起始解析文件，只需文件名，无需全路径 。必须。

--start-position/--start-pos 起始解析位置。可选。默认为start-file的起始位置。

--stop-file/--end-file 终止解析文件。可选。默认为start-file同一个文件。若解析模式为stop-never，此选项失效。

--stop-position/--end-pos 终止解析位置。可选。默认为stop-file的最末位置；若解析模式为stop-never，此选项失效。

--start-datetime 起始解析时间，格式'%Y-%m-%d %H:%M:%S'。可选。默认不过滤。

--stop-datetime 终止解析时间，格式'%Y-%m-%d %H:%M:%S'。可选。默认不过滤。

对象过滤

-d, --databases 只解析目标db的sql，多个库用空格隔开，如-d db1 db2。可选。默认为空。

-t, --tables 只解析目标table的sql，多张表用空格隔开，如-t user1 user2。可选。默认为空。

--only-dml 只解析dml，忽略ddl。可选。默认TRUE。

--sql-type 只解析指定类型，支持INSERT, UPDATE, DELETE。多个类型用空格隔开，如--sql-type INSERT DELETE。可选。默认为增删改都解析。用了此参数但没填任何类型，则三者都不解析。

四、一个例子

小明在11:44时误删了test库user表部分数据，需要紧急回滚。

test库user表原有数据，没有的话可以创建一个。。。

mysql> select \* from user;

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

| id | name | addtime |

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

| 1 | xiaozhao | 2013-11-11 00:04:33 |

| 2 | xiaoqian | 2014-11-11 00:04:48 |

| 3 | xiaosun | 2016-11-11 20:25:00 |

| 4 | xiaoli | 2013-11-11 00:00:00 |

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

4 rows in set (0.00 sec)

11:44时，user表数据被误删除。与此同时，正常业务数据是在继续写入的

delete from user where addtime>'2014-01-01';

Query OK, 2 rows affected (0.02 sec)

mysql> select count(\*) from user;

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

| count(\*) |

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

| 2 |

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

1 row in set (0.00 sec)

mysql> select \* from user;

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

| id | name | addtime |

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

| 1 | xiaozhao | 2013-11-11 00:04:33 |

| 4 | xiaoli | 2013-11-11 00:00:00 |

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

2 rows in set (0.00 sec)

```

\*\*恢复数据步骤\*\*：

1. 登录mysql，查看目前的binlog文件

```bash

mysql> show master logs;

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

| Log\_name | File\_size |

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

| mysql-bin.000001 | 475 |

| mysql-bin.000002 | 449038846 |

| mysql-bin.000003 | 568962489 |

| mysql-bin.000004 | 177 |

| mysql-bin.000005 | 7859 |

| mysql-bin.000006 | 737 |

| mysql-bin.000007 | 7469 |

| mysql-bin.000008 | 664 |

| mysql-bin.000009 | 446 |

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

9 rows in set (0.00 sec)

```

2. 最新的binlog文件是mysql-bin.000009。我们的目标是筛选出需要回滚的SQL，由于误操作人只知道大致的误操作时间，我们首先根据时间做一次过滤。只需要解析test库user表。(注：如果有多个sql误操作，则生成的binlog可能分布在多个文件，需解析多个文件)

```bash

[root@localhost binlog2sql]# ./binlog2sql.py -h127.0.0.1 -P3306 -uroot -p'Aa123456' -dtest -tuser --start-file='mysql-bin.000009' > ./raw.sql

raw.sql输出：

DELETE FROM `test`.`user` WHERE `addtime`='2014-11-11 00:04:48' AND `id`=2 AND `name`='xiaoqian' LIMIT 1; #start 4 end 415 time 2018-01-03 09:40:57

DELETE FROM `test`.`user` WHERE `addtime`='2016-11-11 20:25:00' AND `id`=3 AND `name`='xiaosun' LIMIT 1; #start 4 end 415 time 2018-01-03 09:40:57

...

```

参数检查./binlog2sql.py -h127.0.0.1 -P3306 -uroot -p'Aa123456' --start-file='mysql-bin.000008' > ./raw.sql

[root@localhost binlog2sql]# ./binlog2sql.py -uroot -p'Aa123456' --start-file='mysql-bin.000008' > ./raw.sql

3. 根据位置信息，我们确定了误操作sql来自同一个事务，准确位置在4-415之间(binlog2sql对于同一个事务会输出同样的start position)。再根据位置过滤，使用 \_-B\_ 选项生成回滚sql，检查回滚sql是否正确。(注：真实场景下，生成的回滚SQL经常会需要进一步筛选。结合grep、编辑器等)

```bash

[root@localhost binlog2sql]# ./binlog2sql.py -h127.0.0.1 -P3306 -uroot -p'Aa123456' -dtest -tuser --start-file='mysql-bin.000009' --start-position=4 --stop-position=415 -B > ./rollback.sql

rollback.sql 输出：

INSERT INTO `test`.`user`(`addtime`, `id`, `name`) VALUES ('2016-11-11 20:25:00', 3, 'xiaosun'); #start 4 end 415 time 2018-01-03 09:40:57

INSERT INTO `test`.`user`(`addtime`, `id`, `name`) VALUES ('2014-11-11 00:04:48', 2, 'xiaoqian'); #start 4 end 415 time 2018-01-03 09:40:57

[root@localhost binlog2sql]# wc -l /tmp/rollback.sql

2 /tmp/rollback.sql

```

4. 与业务方确认回滚sql没问题，执行回滚语句。登录mysql，确认回滚成功。

```bash

[root@localhost binlog2sql]# mysql -P3306 -uroot -p'Aa123456' < ./rollback.sql

mysql: [Warning] Using a password on the command line interface can be insecure.

以上就完成了回滚，可以登录数据库看看数据在不在里面。。。

mysql> select \* from user;

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

| id | name | addtime |

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

| 1 | xiaozhao | 2013-11-11 00:04:33 |

| 2 | xiaoqian | 2014-11-11 00:04:48 |

| 3 | xiaosun | 2016-11-11 20:25:00 |

| 4 | xiaoli | 2013-11-11 00:00:00 |

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

4 rows in set (0.00 sec)

mysql> select count(\*) from user;

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

| count(\*) |

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

| 4 |

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

1 row in set (0.00 sec)

不停mysql服务还原到本机数据库完成。