Mysql之binlog日志说明及利用binlog日志恢复数据操作记录

众所周知，binlog日志对于mysql数据库来说是十分重要的。在数据丢失的紧急情况下，我们往往会想到用binlog日志功能进行数据恢复（定时全备份+binlog日志恢复增量数据部分），化险为夷！

一、简单了解binlog

MySQL的二进制日志binlog可以说是MySQL最重要的日志，它记录了所有的DDL和DML语句（除了数据查询语句select），以事件形式记录，还包含语句所执行的消耗的时间，MySQL的二进制日志是事务安全型的。

==========

DDL

- Data Definition Language 数据库定义语言

主要的命令有CREATE、ALTER、DROP等，DDL主要是用在定义或改变表（TABLE）的结构，数据类型，表之间的链接和约束等初始化工作上，他们大多在建立表时使用。

DML

- Data Manipulation Language 数据操纵语言

主要的命令是SELECT、UPDATE、INSERT、DELETE，就象它的名字一样，这4条命令是用来对数据库里的数据进行操作的语言

==========

mysqlbinlog常见的选项有以下几个：

--start-datetime：从二进制日志中读取指定等于时间戳或者晚于本地服务器的时间

--stop-datetime：从二进制日志中读取指定小于时间戳或者等于本地服务器的时间 取值和上述一样

--start-position：从二进制日志中读取指定position 事件位置作为开始。

--stop-position：从二进制日志中读取指定position 事件位置作为事件截至

==========

一般来说开启binlog日志大概会有1%的性能损耗。

binlog日志有两个最重要的使用场景

1）MySQL主从复制：MySQL Replication在Master端开启binlog，Master把它的二进制日志传递给slaves来达到
master-slave数据一致的目的。

2）自然就是数据恢复了，通过使用mysqlbinlog工具来使恢复数据。

binlog日志包括两类文件

1）二进制日志索引文件（文件名后缀为.index）用于记录所有的二进制文件

2）二进制日志文件（文件名后缀为.00000*）记录数据库所有的DDL和DML(除了数据查询语句select)语句事件。

二、开启binlog日志功能

1）编辑打开mysql配置文件/etc/mys.cnf

```
[root@vm-002 ~]# vim /etc/my.cnf
```

在[mysqld] 区块添加

log-bin=mysql-bin 确认是打开状态(mysql-bin 是日志的基本名或前缀名)

注意：每次服务器（数据库）重启，服务器会调用flush logs;，新创建一个binlog日志！

2）重启mysqld服务使配置生效

```
[root@vm-002 ~]# /etc/init.d/mysqld stop
[root@vm-002 ~]# /etc/init.d/mysqld restart
Stopping mysqld: [ OK ]
Starting mysqld: [ OK ]
```

3）查看binlog日志是否开启

```
mysql> show variables like 'log_%';
+---------------------------------+---------------------+
| Variable_name | Value |
```


昵称：散尽浮华
园龄：3年4个月
粉丝：1985
关注：23
+加关注

< 2019年7月 >

日	一	二	三	四	五	六
30	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30	31	1	2	3
4	5	6	7	8	9	10

搜索

找找看

谷歌搜索

常用链接

我的随笔
我的评论
我的参与
最新评论
我的标签

随笔分类

Ansible(3)
Apache(6)
Ceph(5)
ClusterShell(1)
DNS(5)
Docker(27)
DRBD(2)
Elasticsearch(6)
Etcd
Expect(2)
Fabric(1)
FastDFS(1)
FTP(4)
GlusterFS (5)
Haproxy(6)
IP SAN(1)
Iptables(6)
Jenkins(8)
Jira and Confluence(7)
Jumpserver(4)
Kafka(2)
LB+HA(23)
LDAP(2)
LVM(3)
LVS(5)
Maven/Nexus(1)
Memcached(4)
MongoDB(11)

```
+---------------------------------+---------------------+
| log_bin | ON |
| log_bin_trust_function_creators | OFF |
| log_bin_trust_routine_creators | OFF |
| log_error | /var/log/mysqld.log |
| log_output | FILE |
| log_queries_not_using_indexes | OFF |
| log_slave_updates | OFF |
| log_slow_queries | OFF |
| log_warnings | 1 |
+---------------------------------+---------------------+
9 rows in set (0.00 sec)
```

三、常用的binlog日志操作命令

1) 查看所有binlog日志列表

```
mysql> show master logs;
+------------------+-----------+
| Log_name | File_size |
+------------------+-----------+
| mysql-bin.000001 | 149 |
| mysql-bin.000002 | 4102 |
+------------------+-----------+
2 rows in set (0.00 sec)
```

2) 查看master状态，即最后(最新)一个binlog日志的编号名称，及其最后一个操作事件pos结束点(Position)值

```
mysql> show master status;
+------------------+----------+--------------+------------------+
| File | Position | Binlog_Do_DB | Binlog_Ignore_DB |
+------------------+----------+--------------+------------------+
| mysql-bin.000002 | 4102 | | |
+------------------+----------+--------------+------------------+
1 row in set (0.00 sec)
```

3) flush刷新log日志，自此刻开始产生一个新编号的binlog日志文件

```
mysql> flush logs;
Query OK, 0 rows affected (0.13 sec)

mysql> show master logs;
+------------------+-----------+
| Log_name | File_size |
+------------------+-----------+
| mysql-bin.000001 | 149 |
| mysql-bin.000002 | 4145 |
| mysql-bin.000003 | 106 |
+------------------+-----------+
3 rows in set (0.00 sec)
```

注意：每当mysqld服务重启时，会自动执行此命令，刷新binlog日志；在mysqldump备份数据时加 -F 选项也会刷新binlog日志；

4) 重置(清空)所有binlog日志

```
mysql> reset master;
Query OK, 0 rows affected (0.12 sec)

mysql> show master logs;
+------------------+-----------+
| Log_name | File_size |
+------------------+-----------+
| mysql-bin.000001 | 106 |
+------------------+-----------+
1 row in set (0.00 sec)
```

四、查看binlog日志内容，常用有两种方式

1) 使用mysqlbinlog自带查看命令法：

注意：

-->binlog是二进制文件，普通文件查看器cat、more、vim等都无法打开，必须使用自带的mysqlbinlog命令查看

-->binlog日志与数据库文件在同目录中

-->在MySQL5.5以下版本使用mysqlbinlog命令时如果报错，就加上 “--no-defaults”选项

查看mysql的数据存放目录，从下面结果可知是/var/lib//mysql

```
[root@vm-002 ~]# ps -ef|grep mysql
root 9791 1 0 21:18 pts/0 00:00:00 /bin/sh /usr/bin/mysqld_safe --datadir=/var/lib/mysql --socket=/var/lib/mysql/mysql.sock --pid-file=/var/run/mysqld/mysqld.pid --basedir=/usr --
```


随笔档案

Linux加油站

```
user=mysql
mysql 9896 9791 0 21:18 pts/0 00:00:00 /usr/libexec/mysqld --basedir=/usr --
datadir=/var/lib/mysql --user=mysql --log-error=/var/log/mysqld.log --pid-
file=/var/run/mysqld/mysqld.pid --socket=/var/lib/mysql/mysql.sock
root 9916 9699 0 21:18 pts/0 00:00:00 mysql -px xxxx
root 9919 9715 0 21:23 pts/1 00:00:00 grep --color mysql

[root@vm-002 ~]# cd /var/lib/mysql/
[root@vm-002 mysql]# ls
ibdata1 ib_logfile0 ib_logfile1 mysql mysql-bin.000001 mysql-bin.000002 mysql-bin.index
mysql.sock ops test
```

使用mysqlbinlog命令查看binlog日志内容，下面截取其中的一个片段分析：

```
[root@vm-002 mysql]# mysqlbinlog mysql-bin.000002
.....
# at 624
#160925 21:29:53 server id 1 end_log_pos 796 Query thread_id=3 exec_time=0 error_code=0
SET TIMESTAMP=1474810193/*!*/;
insert into member(`name`,`sex`,`age`,`classid`) values('wangshibo','m',27,'cls1'),
('guohuihui','w',27,'cls2')        #执行的sql语句
/*!*/;
# at 796
#160925 21:29:53 server id 1 end_log_pos 823 Xid = 17                #执行的时间
.....
```

解释：
server id 1 ： 数据库主机的服务号；
end_log_pos 796： sql结束时的pos节点
thread_id=11： 线程号

2）上面这种办法读取出binlog日志的全文内容比较多，不容易分辨查看到pos点信息
下面介绍一种更为方便的查询命令：
命令格式：

```
mysql> show binlog events [IN 'log_name'] [FROM pos] [LIMIT [offset,] row_count];
```

参数解释：
IN 'log_name' ：指定要查询的binlog文件名(不指定就是第一个binlog文件)
FROM pos ：指定从哪个pos起始点开始查起(不指定就是从整个文件首个pos点开始算)
LIMIT [offset,] ：偏移量(不指定就是0)
row_count ：查询总条数(不指定就是所有行)

```
mysql> show master logs;
+------------------+-----------+
| Log_name | File_size |
+------------------+-----------+
| mysql-bin.000001 | 125 |
| mysql-bin.000002 | 823 |
+------------------+-----------+
2 rows in set (0.00 sec)

mysql> show binlog events in 'mysql-bin.000002'\G;
*************************** 1. row ***************************
Log_name: mysql-bin.000002
Pos: 4
Event_type: Format_desc
Server_id: 1
End_log_pos: 106
Info: Server ver: 5.1.73-log, Binlog ver: 4
*************************** 2. row ***************************
Log_name: mysql-bin.000002
Pos: 106
Event_type: Query
Server_id: 1
End_log_pos: 188
Info: use `ops`; drop table customers
*************************** 3. row ***************************
Log_name: mysql-bin.000002
Pos: 188
Event_type: Query
Server_id: 1
End_log_pos: 529
Info: use `ops`; CREATE TABLE IF NOT EXISTS `member` (
`id` int(10) unsigned NOT NULL AUTO_INCREMENT,
`name` varchar(16) NOT NULL,
```


最新评论

```
`sex` enum('m','w') NOT NULL DEFAULT 'm',
`age` tinyint(3) unsigned NOT NULL,
`classid` char(6) DEFAULT NULL,
PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8
*************************** 4. row ***************************
Log_name: mysql-bin.000002
Pos: 529
Event_type: Query
Server_id: 1
End_log_pos: 596
Info: BEGIN
*************************** 5. row ***************************
Log_name: mysql-bin.000002
Pos: 596
Event_type: Intvar
Server_id: 1
End_log_pos: 624
Info: INSERT_ID=1
*************************** 6. row ***************************
Log_name: mysql-bin.000002
Pos: 624
Event_type: Query
Server_id: 1
End_log_pos: 796
Info: use `ops`; insert into member(`name`,`sex`,`age`,`classid`)
values('wangshibo','m',27,'cls1'),('guohuihui','w',27,'cls2')
*************************** 7. row ***************************
Log_name: mysql-bin.000002
Pos: 796
Event_type: Xid
Server_id: 1
End_log_pos: 823
Info: COMMIT /* xid=17 */
7 rows in set (0.00 sec)

ERROR:
No query specified

mysql>
```

上面这条语句可以将指定的binlog日志文件，分成有效事件行的方式返回，并可使用limit指定pos点的起始偏移，查询条数！
如下操作示例：
a) 查询第一个(最早)的binlog日志：

```
mysql> show binlog events\G;
```

b) 指定查询 mysql-bin.000002这个文件：

```
mysql> show binlog events in 'mysql-bin.000002'\G;
```

c) 指定查询 mysql-bin.000002这个文件，从pos点:624开始查起：

```
mysql> show binlog events in 'mysql-bin.000002' from 624\G;
```

d) 指定查询 mysql-bin.000002这个文件，从pos点:624开始查起，查询10条（即10条语句）

```
mysql> show binlog events in 'mysql-bin.000002' from 624 limit 10\G;
```

e) 指定查询 mysql-bin.000002这个文件，从pos点:624开始查起，偏移2行（即中间跳过2个），查询10条

```
mysql> show binlog events in 'mysql-bin.000002' from 624 limit 2,10\G;
```

五、利用binlog日志恢复mysql数据

以下对ops库的member表进行操作

```
mysql> use ops ;
mysql> CREATE TABLE IF NOT EXISTS `member` (
-> `id` int(10) unsigned NOT NULL AUTO_INCREMENT,
-> `name` varchar(16) NOT NULL,
-> `sex` enum('m','w') NOT NULL DEFAULT 'm',
-> `age` tinyint(3) unsigned NOT NULL,
-> `classid` char(6) DEFAULT NULL,
-> PRIMARY KEY (`id`)
-> ) ENGINE=InnoDB DEFAULT CHARSET=utf8;
Query OK, 0 rows affected (0.10 sec)

mysql> show tables;
+---------------+
```

--南风深蓝

5. Re:Kubernetes容器集群管理环境 - 完整部署（下篇）
@散尽浮华好的，谢谢！！！E0715 07:28:07.211366 33412 available_controller.go:316] v1beta1.metrics.k8s.io failed ......
--herocjx

6. Re:Centos下安装破解Jira7的操作记录
邮箱发送QQ 发送不成功！
--dingzhj

7. Re:Kubernetes容器集群管理环境 - 完整部署（下篇）
@herocjx证书问题。浏览器地址访问需要提前创建和导入证书，可参考"中篇"的"9.3 - 浏览器访问kube-apiserver等安全端口，创建和导入证书的做法".....
--散尽浮华

8. Re:Kubernetes容器集群管理环境 - 完整部署（下篇）
通过地址访问： kind "Status"apiVersion "v1"metadata {}status "Failure"message "Unauthorized"reason "Unautho.....
--herocjx

9. Re:Kubernetes容器集群管理环境 - 完整部署（下篇）
metrics-server E0715 07:28:07.211366 33412 available_controller.go:316] v1beta1.metrics.k8s.io f.....
--herocjx

10. Re:Redis主从复制原理总结
总结的比较细致
--袁码

阅读排行榜

1. Git忽略提交规则 - .gitignore配置运维总结(378922)
2. ELK实时日志分析平台环境部署--完整记录(158997)
3. 完整部署CentOS7.2+OpenStack+kvm 云平台环境（1）--基础环境搭建(125491)
4. mysql数据库误删除后的数据恢复操作说明(114331)
5. Linux终端复用神器-Tmux使用梳理(110933)
6. Mysql之binlog日志说明及利用binlog日志恢复数据操作记录(66883)
7. mysql表名忽略大小写问题记录(65828)
8. MySQL两种存储引擎: MyISAM和InnoDB 简单总结(63565)
9. Gitlab利用Webhook实现Push代码后的jenkins自动构建(62260)
10. python报错问题解决：'ascii' codec can't encode character(61669)

评论排行榜

1. 完整部署CentOS7.2+OpenStack+kvm 云平台环境（1）--基础环境搭建(119)
2. ELK实时日志分析平台环境部署--完整记录(24)
3. [原创]CI持续集成系统环境--Gitlab+Gerrit+Jenkins完整对接(20)
4. kvm虚拟化管理平台WebVirtMgr部署-完整记录(1)(18)
5. Centos下安装破解Jira7的操作记录(18)

```
| Tables_in_ops |
+---------------+
| member |
+---------------+
1 row in set (0.00 sec)

mysql> desc member;
+---------+---------------------+------+-----+---------+----------------+
| Field | Type | Null | Key | Default | Extra |
+---------+---------------------+------+-----+---------+----------------+
| id | int(10) unsigned | NO | PRI | NULL | auto_increment |
| name | varchar(16) | NO | | NULL | |
| sex | enum('m','w') | NO | | m | |
| age | tinyint(3) unsigned | NO | | NULL | |
| classid | char(6) | YES | | NULL | |
+---------+---------------------+------+-----+---------+----------------+
5 rows in set (0.00 sec)
```

事先插入两条数据

```
mysql> insert into member(`name`,`sex`,`age`,`classid`) values('wangshibo','m',27,'cls1'),
('guohuihui','w',27,'cls2');
Query OK, 2 rows affected (0.08 sec)
Records: 2 Duplicates: 0 Warnings: 0
mysql> select * from member;
+----+-----------+-----+-----+---------+
| id | name | sex | age | classid |
+----+-----------+-----+-----+---------+
| 1 | wangshibo | m | 27 | cls1 |
| 2 | guohuihui | w | 27 | cls2 |
+----+-----------+-----+-----+---------+
2 rows in set (0.00 sec)
```

下面开始进行场景模拟：

1）

ops库会在每天凌晨4点进行一次完全备份的定时计划任务，如下：

```
[root@vm-002 ~]# crontab -l
0 4 * * * /usr/bin/mysqldump -uroot -p -B -F -R -x --master-data=2 ops|gzip
>/opt/backup/ops_$(date +%F).sql.gz
```

这里手动执行下，将ops数据库备份到/opt/backup/ops_$(date +%F).sql.gz文件中：

```
[root@vm-002 ~]# mysqldump -uroot -p -B -F -R -x --master-data=2 ops|gzip
>/opt/backup/ops_$(date +%F).sql.gz
Enter password:
[root@vm-002 ~]# ls /opt/backup/
ops_2016-09-25.sql.gz
```

参数说明：

-B：指定数据库

-F：刷新日志

-R：备份存储过程等

-x：锁表

--master-data：在备份语句里添加CHANGE MASTER语句以及binlog文件及位置点信息

待到数据库备份完成，就不用担心数据丢失了，因为有完全备份数据在！！

由于上面在全备份的时候使用了-F选项，那么当数据备份操作刚开始的时候系统就会自动刷新log，这样就会自动产生

一个新的binlog日志，这个新的binlog日志就会用来记录备份之后的数据库“增删改”操作

查看一下：

```
mysql> show master status;
+------------------+----------+--------------+------------------+
| File | Position | Binlog_Do_DB | Binlog_Ignore_DB |
+------------------+----------+--------------+------------------+
| mysql-bin.000003 | 106 | | |
+------------------+----------+--------------+------------------+
1 row in set (0.00 sec)
```

也就是说， mysql-bin.000003 是用来记录4:00之后对数据库的所有“增删改”操作。

2）

早上9点上班了，由于业务的需求会对数据库进行各种“增删改”操作。

比如：在ops库下member表内插入、修改了数据等等：

先是早上进行插入数据：
mysql> insert into ops.member(`name`,`sex`,`age`,`classid`) values('yiyi','w',20,'cls1'),
('xiaoer','m',22,'cls3'),('zhangsan','w',21,'cls5'),('lisi','m',20,'cls4'),('wangwu','w',26,'cls6');
Query OK, 5 rows affected (0.08 sec)
Records: 5 Duplicates: 0 Warnings: 0

mysql> select * from member;
+----+-----------+-----+-----+---------+
| id | name | sex | age | classid |
+----+-----------+-----+-----+---------+
1	wangshibo	m	27	cls1
2	guohuihui	w	27	cls2
3	yiyi	w	20	cls1
4	xiaoer	m	22	cls3
5	zhangsan	w	21	cls5
6	lisi	m	20	cls4
7	wangwu	w	26	cls6
+----+-----------+-----+-----+---------+
7 rows in set (0.00 sec)

3）
中午又执行了修改数据操作：
mysql> update ops.member set name='李四' where id=4;
Query OK, 1 row affected (0.07 sec)
Rows matched: 1 Changed: 1 Warnings: 0

mysql> update ops.member set name='小二' where id=2;
Query OK, 1 row affected (0.06 sec)
Rows matched: 1 Changed: 1 Warnings: 0

mysql> select * from member;
+----+-----------+-----+-----+---------+
| id | name | sex | age | classid |
+----+-----------+-----+-----+---------+
1	wangshibo	m	27	cls1
2	小二	w	27	cls2
3	yiyi	w	20	cls1
4	李四	m	22	cls3
5	zhangsan	w	21	cls5
6	lisi	m	20	cls4
7	wangwu	w	26	cls6
+----+-----------+-----+-----+---------+
7 rows in set (0.00 sec)

4）
在下午18:00的时候，悲剧莫名其妙的出现了！
手贱执行了drop语句，直接删除了ops库！吓尿！
mysql> drop database ops;
Query OK, 1 row affected (0.02 sec)

5）
这种时候，一定不要慌张！！！
先仔细查看最后一个binlog日志，并记录下关键的pos点，到底是哪个pos点的操作导致了数据库的破坏(通常在最后几步)；

先备份一下最后一个binlog日志文件：
[root@vm-002 ~]# cd /var/lib/mysql/
[root@vm-002 mysql]# cp -v mysql-bin.000003 /opt/backup/
`mysql-bin.000003' -> `/opt/backup/mysql-bin.000003'
[root@vm-002 mysql]# ls /opt/backup/
mysql-bin.000003 ops_2016-09-25.sql.gz

接着执行一次刷新日志索引操作，重新开始新的binlog日志记录文件。按理说mysql-bin.000003
这个文件不会再有后续写入了，因为便于我们分析原因及查找ops节点，以后所有数据库操作都会写入到下一个日志文件。
mysql> flush logs;
Query OK, 0 rows affected (0.13 sec)

mysql> show master status;
+------------------+----------+--------------+------------------+
| File | Position | Binlog_Do_DB | Binlog_Ignore_DB |
+------------------+----------+--------------+------------------+
| mysql-bin.000004 | 106 | | |
+------------------+----------+--------------+------------------+
1 row in set (0.00 sec)

6 ）
读取binlog日志，分析问题。
读取binlog日志的方法上面已经说到。
方法一：使用mysqlbinlog读取binlog日志：
[root@vm-002 ~]# cd /var/lib/mysql/
[root@vm-002 mysql]# mysqlbinlog mysql-bin.000003

方法二：登录服务器，并查看(推荐此种方法)
mysql> show binlog events in 'mysql-bin.000003';

```
+------------------+-----+-------------+-----------+-------------+---------------------------------------
----------------------------------------------------------------------------+
| Log_name | Pos | Event_type | Server_id | End_log_pos | Info |
+------------------+-----+-------------+-----------+-------------+---------------------------------------
----------------------------------------------------------------------------+
| mysql-bin.000003 | 4 | Format_desc | 1 | 106 | Server ver: 5.1.73-log, Binlog ver: 4 |
| mysql-bin.000003 | 106 | Query | 1 | 173 | BEGIN |
| mysql-bin.000003 | 173 | Intvar | 1 | 201 | INSERT_ID=3 |
| mysql-bin.000003 | 201 | Query | 1 | 444 | use `ops`; insert into
ops.member(`name`,`sex`,`age`,`gsan','w',21,'cls5'),('lisi','m',20,'cls4'),('wangwu','w',26,'cls6') |
| mysql-bin.000003 | 444 | Xid | 1 | 471 | COMMIT /* xid=66 */ |
| mysql-bin.000003 | 471 | Query | 1 | 538 | BEGIN |
| mysql-bin.000003 | 538 | Query | 1 | 646 | use `ops`; update ops.member set name='李四'
where id= |
| mysql-bin.000003 | 646 | Xid | 1 | 673 | COMMIT /* xid=68 */ |
| mysql-bin.000003 | 673 | Query | 1 | 740 | BEGIN |
| mysql-bin.000003 | 740 | Query | 1 | 848 | use `ops`; update ops.member set name='小二'
where id= |
| mysql-bin.000003 | 848 | Xid | 1 | 875 | COMMIT /* xid=69 */ |
| mysql-bin.000003 | 875 | Query | 1 | 954 | drop database ops |
| mysql-bin.000003 | 954 | Rotate | 1 | 997 | mysql-bin.000004;pos=4 |
+------------------+-----+-------------+-----------+-------------+---------------------------------------
----------------------------------------------------------------------------+
13 rows in set (0.00 sec)
```

或者：

```
mysql> show binlog events in 'mysql-bin.000003'\G;
.........
.........
*************************** 12. row ***************************
Log_name: mysql-bin.000003
Pos: 875
Event_type: Query
Server_id: 1
End_log_pos: 954
Info: drop database ops
*************************** 13. row ***************************
Log_name: mysql-bin.000003
Pos: 954
Event_type: Rotate
Server_id: 1
End_log_pos: 997
Info: mysql-bin.000004;pos=4
13 rows in set (0.00 sec)
```

通过分析，造成数据库破坏的pos点区间是介于 875--954 之间（这是按照日志区间的pos节点算的），只要恢复到875前就可。

7 ）
先把凌晨4点全备份的数据恢复：

```
[root@vm-002 ~]# cd /opt/backup/
[root@vm-002 backup]# ls
mysql-bin.000003 ops_2016-09-25.sql.gz
[root@vm-002 backup]# gzip -d ops_2016-09-25.sql.gz
[root@vm-002 backup]# mysql -uroot -p -v < ops_2016-09-25.sql
Enter password:
--------------
/*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */
--------------

--------------
/*!40101 SET @OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS */
--------------
```

.....
.....

/*!40111 SET SQL_NOTES=@OLD_SQL_NOTES */

这样就恢复了截至当日凌晨(4:00)前的备份数据都恢复了。

```
mysql> show databases;                    #发现ops库已经恢复回来了
mysql> use ops;
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_ops |
+---------------+
| member |
+---------------+
1 row in set (0.00 sec)

mysql> select * from member;
+----+-----------+-----+-----+---------+
| id | name | sex | age | classid |
+----+-----------+-----+-----+---------+
| 1 | wangshibo | m | 27 | cls1 |
| 2 | guohuihui | w | 27 | cls2 |
+----+-----------+-----+-----+---------+
2 rows in set (0.00 sec)

mysql>
```

但是这仅仅只是恢复了当天凌晨4点之前的数据，在4:00--18:00之间的数据还没有恢复回来！！
怎么办呢？
莫慌！这可以根据前面提到的mysql-bin.000003的新binlog日志进行恢复。

8）
从binlog日志恢复数据
恢复命令的语法格式：
mysqlbinlog mysql-bin.0000xx | mysql -u用户名 -p密码 数据库名

常用参数选项解释：
--start-position=875 起始pos点
--stop-position=954 结束pos点
--start-datetime="2016-9-25 22:01:08" 起始时间点
--stop-datetime="2019-9-25 22:09:46" 结束时间点
--database=zyyshop 指定只恢复zyyshop数据库(一台主机上往往有多个数据库，只限本地log日志)

不常用选项：
-u --user=name 连接到远程主机的用户名
-p --password[=name] 连接到远程主机的密码
-h --host=name 从远程主机上获取binlog日志
--read-from-remote-server 从某个MySQL服务器上读取binlog日志

小结：实际是将读出的binlog日志内容，通过管道符传递给mysql命令。这些命令、文件尽量写成绝对路径；

a）完全恢复(需要手动vim编辑mysql-bin.000003，将那条drop语句剔除掉)
```
[root@vm-002 backup]# cp /var/lib/mysql/mysql-bin.000003 /opt/backup
[root@vm-002 backup]# mysqlbinlog /opt/backup/mysql-bin.000003 > /opt/backup/000003.sql
[root@vm-002 backup]# vim /opt/backup/000003.sql #删除里面的drop语句
[root@vm-002 backup]# mysql -uroot -p -v < /opt/backup/000003.sql
```

温馨提示：
在恢复全备数据之前必须将该binlog文件移出，否则恢复过程中，会继续写入语句到binlog，最终导致增量恢复数据部分变得比较混乱！
可参考：https://www.cnblogs.com/kevingrace/p/5904800.html

b）指定pos结束点恢复(部分恢复)：
--stop-position=471 pos结束节点（按照事务区间算，是471）
注意：
此pos结束节点介于“member表原始数据”与更新“name='李四'”之前的数据，这样就可以恢复到更改“name='李四'”之前的数据了。
操作如下：

[root@vm-002 ~]# /usr/bin/mysqlbinlog --stop-position=471 --database=ops
/var/lib/mysql/mysql-bin.000003 | /usr/bin/mysql -uroot -p123456 -v ops

mysql> select * from member;
+----+-----------+-----+-----+---------+
| id | name | sex | age | classid |
+----+-----------+-----+-----+---------+
1	wangshibo	m	27	cls1
2	guohuihui	w	27	cls2
3	yiyi	w	20	cls1
4	xiaoer	m	22	cls3
5	zhangsan	w	21	cls5
6	lisi	m	20	cls4
7	wangwu	w	26	cls6
+----+-----------+-----+-----+---------+
7 rows in set (0.00 sec)

恢复截止到更改"name='李四'"之间的数据（按照事务区间算，是673）
[root@vm-002 ~]# /usr/bin/mysqlbinlog --stop-position=673 --database=ops
/var/lib/mysql/mysql-bin.000003 | /usr/bin/mysql -uroot -p123456 -v ops

mysql> select * from member;
+----+-----------+-----+-----+---------+
| id | name | sex | age | classid |
+----+-----------+-----+-----+---------+
1	wangshibo	m	27	cls1
2	guohuihui	w	27	cls2
3	yiyi	w	20	cls1
4	李四	m	22	cls3
5	zhangsan	w	21	cls5
6	lisi	m	20	cls4
7	wangwu	w	26	cls6
+----+-----------+-----+-----+---------+
7 rows in set (0.00 sec)

c) 指定pso点区间恢复(部分恢复)：
更新 name='李四' 这条数据，日志区间是Pos[538] --> End_log_pos[646]，按事务区间是：Pos[471] -->
End_log_pos[673]

更新 name='小二' 这条数据，日志区间是Pos[740] --> End_log_pos[848]，按事务区间是：Pos[673] -->
End_log_pos[875]

c1)
单独恢复 name='李四' 这步操作，可这样：
按照binlog日志区间单独恢复：
[root@vm-002 ~]# /usr/bin/mysqlbinlog --start-position=538 --stop-position=646 --database=ops
/var/lib/mysql/mysql-bin.000003 | /usr/bin/mysql -uroot -p123456 -v ops

按照事务区间单独恢复
[root@vm-002 ~]# /usr/bin/mysqlbinlog --start-position=471 --stop-position=673 --database=ops
/var/lib/mysql/mysql-bin.000003 | /usr/bin/mysql -uroot -p123456 -v ops

c2)
单独恢复 name='小二' 这步操作，可这样：
按照binlog日志区间单独恢复：
[root@vm-002 ~]# /usr/bin/mysqlbinlog --start-position=740 --stop-position=848 --database=ops
/var/lib/mysql/mysql-bin.000003 | /usr/bin/mysql -uroot -p123456 -v ops

按照事务区间单独恢复
[root@vm-002 ~]# /usr/bin/mysqlbinlog --start-position=673 --stop-position=875 --database=ops
/var/lib/mysql/mysql-bin.000003 | /usr/bin/mysql -uroot -p123456 -v ops

c3)
将 name='李四'、name='小二' 多步操作一起恢复，需要按事务区间，可这样：
[root@vm-002 ~]# /usr/bin/mysqlbinlog --start-position=471 --stop-position=875 --database=ops
/var/lib/mysql/mysql-bin.000003 | /usr/bin/mysql -uroot -p123456 -v ops

查看数据库：
mysql> select * from member;
+----+-----------+-----+-----+---------+
| id | name | sex | age | classid |
+----+-----------+-----+-----+---------+
1	wangshibo	m	27	cls1
2	小二	w	27	cls2
3	yiyi	w	20	cls1

4	李四	m	22	cls3
5	zhangsan	w	21	cls5
6	lisi	m	20	cls4
7	wangwu	w	26	cls6
+----+-----------+-----+-----+---------+
7 rows in set (0.00 sec)

这样，就恢复了删除前的数据状态了！！

==========

另外：也可以指定时间节点区间恢复(部分恢复)，就是说除了用pos节点的办法进行恢复，也可以通过指定时间节点区间进行恢复，
按时间恢复需要用mysqlbinlog命令读取binlog日志内容，找时间节点。

如上，误删除ops库后：
先进行全备份恢复
[root@vm-002 backup]# mysql -uroot -p -v < ops_2016-09-25.sql

查看ops数据库
mysql> select * from member;
+----+-----------+-----+-----+---------+
| id | name | sex | age | classid |
+----+-----------+-----+-----+---------+
| 1 | wangshibo | m | 27 | cls1 |
| 2 | guohuihui | w | 27 | cls2 |
+----+-----------+-----+-----+---------+
2 rows in set (0.00 sec)

mysql>

查看mysq-bin00003日志，找出时间节点
[root@vm-002 ~]# cd /var/lib/mysql
[root@vm-002 mysql]# mysqlbinlog mysql-bin.000003
.....
.....
BEGIN
/*!*/;
at 173
#160925 21:57:19 server id 1 end_log_pos 201 Intvar
SET INSERT_ID=3/*!*/;
at 201
#160925 21:57:19 server id 1 end_log_pos 444 Query thread_id=3 exec_time=0 error_code=0
use `ops`/*!*/;
SET TIMESTAMP=1474811839/*!*/;
insert into ops.member(`name`,`sex`,`age`,`classid`) values('yiyi','w',20,'cls1'),
('xiaoer','m',22,'cls3'),('zhangsan','w',21,'cls5'),('lisi','m',20,'cls4'),('wangwu','w',26,'cls6')
#执行的sql语句
/*!*/;
at 444
#160925 21:57:19 server id 1 end_log_pos 471 Xid = 66 **#开始执行的时间**
COMMIT/*!*/;
at 471
#160925 21:58:41 server id 1 end_log_pos 538 Query thread_id=3 exec_time=0 error_code=0
#结束时间
SET TIMESTAMP=1474811921/*!*/;
BEGIN
/*!*/;
at 538
#160925 21:58:41 server id 1 end_log_pos 646 Query thread_id=3 exec_time=0 error_code=0
SET TIMESTAMP=1474811921/*!*/;
update ops.member set name='李四' where id=4 **#执行的sql语句**
/*!*/;
at 646
#160925 21:58:41 server id 1 end_log_pos 673 Xid = 68 **#开始执行的时间**
COMMIT/*!*/;
at 673
#160925 21:58:56 server id 1 end_log_pos 740 Query thread_id=3 exec_time=0 error_code=0 **#结束时间**
SET TIMESTAMP=1474811936/*!*/;
BEGIN
/*!*/;
at 740
#160925 21:58:56 server id 1 end_log_pos 848 Query thread_id=3 exec_time=0 error_code=0

```
SET TIMESTAMP=1474811936/*!*/;
update ops.member set name='小二' where id=2      #执行的sql语句
/*!*/;
# at 848
#160925 21:58:56 server id 1 end_log_pos 875 Xid = 69   #开始执行的时间
COMMIT/*!*/;
# at 875
#160925 22:01:08 server id 1 end_log_pos 954 Query thread_id=3 exec_time=0 error_code=0
 #结束时间
SET TIMESTAMP=1474812068/*!*/;
drop database ops
/*!*/;
# at 954
#160925 22:09:46 server id 1 end_log_pos 997 Rotate to mysql-bin.000004 pos: 4
DELIMITER ;
# End of log file
ROLLBACK /* added by mysqlbinlog */;
/*!50003 SET COMPLETION_TYPE=@OLD_COMPLETION_TYPE*/;
```

恢复到更改"name='李四'"之前的数据

```
[root@vm-002 ~]# /usr/bin/mysqlbinlog --start-datetime="2016-09-25 21:57:19" --stop-
datetime="2016-09-25 21:58:41" --database=ops /var/lib/mysql/mysql-bin.000003 |
/usr/bin/mysql -uroot -p123456 -v ops

mysql> select * from member;
+----+-----------+-----+-----+---------+
| id | name | sex | age | classid |
+----+-----------+-----+-----+---------+
| 1 | wangshibo | m | 27 | cls1 |
| 2 | guohuihui | w | 27 | cls2 |
| 3 | yiyi | w | 20 | cls1 |
| 4 | xiaoer | m | 22 | cls3 |
| 5 | zhangsan | w | 21 | cls5 |
| 6 | lisi | m | 20 | cls4 |
| 7 | wangwu | w | 26 | cls6 |
+----+-----------+-----+-----+---------+
7 rows in set (0.00 sec)

[root@vm-002 ~]# /usr/bin/mysqlbinlog --start-datetime="2016-09-25 21:58:41" --stop-
datetime="2016-09-25 21:58:56" --database=ops /var/lib/mysql/mysql-bin.000003 |
/usr/bin/mysql -uroot -p123456 -v ops
mysql> select * from member;
+----+-----------+-----+-----+---------+
| id | name | sex | age | classid |
+----+-----------+-----+-----+---------+
| 1 | wangshibo | m | 27 | cls1 |
| 2 | guohuihui | w | 27 | cls2 |
| 3 | yiyi | w | 20 | cls1 |
| 4 | 李四 | m | 22 | cls3 |
| 5 | zhangsan | w | 21 | cls5 |
| 6 | lisi | m | 20 | cls4 |
| 7 | wangwu | w | 26 | cls6 |
+----+-----------+-----+-----+---------+
7 rows in set (0.00 sec)

[root@vm-002 ~]# /usr/bin/mysqlbinlog --start-datetime="2016-09-25 21:58:56" --stop-
datetime="2016-09-25 22:01:08" --database=ops /var/lib/mysql/mysql-bin.000003 |
/usr/bin/mysql -uroot -p123456 -v ops
mysql> select * from member;
+----+-----------+-----+-----+---------+
| id | name | sex | age | classid |
+----+-----------+-----+-----+---------+
| 1 | wangshibo | m | 27 | cls1 |
| 2 | 小二 | w | 27 | cls2 |
| 3 | yiyi | w | 20 | cls1 |
| 4 | 李四 | m | 22 | cls3 |
| 5 | zhangsan | w | 21 | cls5 |
| 6 | lisi | m | 20 | cls4 |
| 7 | wangwu | w | 26 | cls6 |
+----+-----------+-----+-----+---------+
7 rows in set (0.00 sec)
```

这样，就恢复了删除前的状态了！

总结：所谓恢复，就是让mysql将保存在binlog日志中指定段落区间的sql语句逐个重新执行一次而已。

**********当你发现自己的才华撑不起野心时，就请安静下来学习吧**********

分类: Mysql

散尽浮华
关注 - 23
粉丝 - 1985
+加关注

11 0

« 上一篇：mysql数据库误删除后的数据恢复操作说明
» 下一篇：网络知识梳理--OSI七层网络与TCP/IP五层网络架构及二层/三层网络

posted @ 2016-09-25 22:26 散尽浮华 阅读(66884) 评论(9) 编辑 收藏

评论

#1楼 2017-08-13 22:22 | 简单--生活 回复 引用

但测试以后发现，如果想恢复Delete了的数据，必须要找到这条数居insert的语句才成样，例如某张表中不太确认是什么时候插入的数据，但在今天被误删除了，好像没有办法在被删除的指定时间区间来恢复数据，望指教。

支持(0) 反对(0)

#2楼 2018-05-14 11:36 | 一缕清风1 回复 引用

学习了

支持(0) 反对(0)

#3楼 2018-08-27 11:37 | 领导来视察 回复 引用

@ 简单--生活
需要用全备份+增量备份来恢复到delete数据之前的状态。

支持(0) 反对(0)

#4楼 2018-09-20 18:29 | 小豹子加油 回复 引用

大佬，二进制文件怎么手工从里面把drop database这条记录删除呢？

支持(0) 反对(0)

#5楼[楼主] 2019-01-08 13:24 | 散尽浮华 回复 引用

@ 瑜珈山脚
mysqlbinlog 将binlong二进制文件转化成可导入的sql文件，然后就可以删除drop记录了

支持(0) 反对(0)

#6楼 2019-01-18 17:27 | 626zhangjun 回复 引用

我靠，写这么详细，基本都看懂了，真是厉害呀，专门登录上来赞一个

支持(0) 反对(0)

#7楼 2019-03-04 22:43 | 莫林1 回复 引用

温馨提示：
在恢复全备数据之前必须将该binlog文件移出，否则恢复过程中，会继续写入语句到binlog，最终导致增量恢复数据部分变得比较混乱！

你好，我将mysql-bing.000003.log移到/tmp目录，但是等我恢复完数据之后，再将该文件移动回来的时候，发现mysql-bing.000003.log里面的文件还是存放了恢复的日志，请问如何将binlog文件移出，才不会记录，谢谢！

支持(0) 反对(0)

#8楼 2019-05-28 18:07 | SuyUha 回复 引用

博主你好,我通过您的文章学习了很多知识,也在工作中帮我解决了非常重要的问题,我想把我的工作和所学的知识记录下来,未来发布在自己的私人网站上,请问我可以使用您的文章中的描述 吗,我会在文章开头 放上您文章的连接 并提示.期望获得您的同意。

支持(0) 反对(0)

#9楼 2019-05-30 15:15 | show_your_power 回复 引用

博主，你的博客都是干货，记录清楚详细，每次闲暇之余都会点开你的博客学习，希望博主博客常开，我持续关注

支持(1) 反对(0)

刷新评论 刷新页面 返回顶部

发表评论

昵称： hzhuan

评论内容：

提交评论 退出 订阅评论

[Ctrl+Enter快捷键提交]

相关博文：

- Mysql之binlog日志说明及利用binlog日志恢复数据操作记录
- Mysql之binlog日志说明及利用binlog日志恢复数据操作记录
- 转：Mysql之binlog日志说明及利用binlog日志恢复数据操作记录
- Mysql之Binlog日志完全备份|增量+binlog日志基于时间点恢复数据
- MySQL的binlog日志

最新新闻：

- 科创板开市暴涨，详解25家企业的"造富"能力
- 印度"月船二号"探测器升空 欲冲击软着陆月球第四国
- NASA 的月球轨道站是在做无用功

· 这比蚂蚁还小的机器人，不用电就能跑
· 世界500强最赚钱50家公司：苹果第二 三星第四
» 更多新闻...