

第1章 错误日志

1.作用

- 1 记录数据库启动以来，状态、警告、报错。诊断数据库报错问题。

2.配置

- 1 默认： 开启状态。存放在数据目录下(/data/3306/data),名字：主机名.err

3.查看

```
1  mysql> select @@log_error;
2  +-----+
3  | @@log_error |
4  +-----+
5  | ./db-51.err |
6  +-----+
7  1 row in set (0.00 sec)
8
9  mysql> select @@datadir;
10 +-----+
11 | @@datadir      |
12 +-----+
13 | /data/mysql_3306/ |
14 +-----+
15 1 row in set (0.00 sec)
```

4.自定义配置

修改配置：

```
1 [root@db-51 ~]# vim /etc/my.cnf
2 [mysqld]
3 #新增加参数
4 log_error=/data/mysql_3306/logs/mysql.err
```

创建日志目录并更改授权：

```
1 [root@db-51 ~]# mkdir /data/mysql_3306/logs/ -p
2 [root@db-51 ~]# chown -R mysql:mysql /data/mysql_3306/logs/
```

重启mysql：

```
1 systemctl status mysqld.service
```

重启后报错：

```
1 9月 13 17:18:57 db-51 mysqld[1439]: Starting MySQL.2020-09-
13T09:18:57.124858Z mysqld_safe error: log-error set to
'/data/mysql_3306/logs/mysql.err', however file don't exists. Create
writable for user 'mysq
```

解决方法：

```
1 mkdir /data/mysql_3306/logs/ -p
2 touch /data/mysql_3306/logs/mysql.err
3 chown -R mysql:mysql /data/mysql_3306/logs/
4 systemctl restart mysqld.service
```

第2章 慢日志

1.作用

1. 记录MySQL工作过程中较慢的语句
2. 默认没有开启,按需求打开。

2.配置

在线配置:

```
1 mysql> select @@slow_query_log;          # 开关
2 mysql> set global slow_query_log=1;      # 在线改
3 mysql> select @@slow_query_log_file;      # 文件位置。离线改。
4 mysql> select @@long_query_time;         # 慢查询时间设定。
5 mysql> set global long_query_time=0.1;    # 在线设置，最低微秒级别。
6 mysql> select @@log_queries_not_using_indexes #如果没走索引会被记录
7 mysql> set global log_queries_not_using_indexes=1; #在线设置
```

永久生效:

```
1 vim /etc/my.cnf
2 slow_query_log=1          #是否启用慢查询日志，1为启用，0为禁用
3 slow_query_log_file=/data/mysql_3306/logs/slow.log #慢日志路径
4 long_query_time=0.1      #SQL语句运行时间阈值，执行时间大于参数值的语句才会被记录下来
5 log_queries_not_using_indexes=1 #将没有使用索引的语句记录到慢查询日志
```

3.模拟慢语句

```
1 select * from t100w as a join t100w as b limit N;
2 select k1,count(*) from t100w where id<N group by k1 having count(*)>N;
3 select k1,count(*) from t100w where num<N group by k1,k2;
4 select * from t100w where id<N order by num desc;
5 select k1,count(*) from t100w where id<N group by k1,k2;
```

4.慢日志分析

```
1 [root@db01 logs]# mysqldumpslow -s c -t 5 slow.log
2 Reading mysql slow query log from slow.log
3 Count: 7  Time=0.01s (0s)  Lock=0.00s (0s)  Rows=177.1 (1240),
  root[root]@localhost
4   select * from t100w as a join t100w as b limit N
5 Count: 6  Time=0.57s (3s)  Lock=0.00s (0s)  Rows=33.7 (202),
  root[root]@localhost
6   select k1,count(*) from t100w where id<N group by k1 having
  count(*)>N
7 Count: 5  Time=0.59s (2s)  Lock=0.00s (0s)  Rows=893.0 (4465),
  root[root]@localhost
8   select k1,count(*) from t100w where num<N group by k1,k2
9 Count: 5  Time=0.61s (3s)  Lock=0.00s (0s)  Rows=37.0 (185),
  root[root]@localhost
10  select * from t100w where id<N order by num desc
11 Count: 4  Time=0.61s (2s)  Lock=0.00s (0s)  Rows=496.0 (1984),
  root[root]@localhost
12  select k1,count(*) from t100w where id<N group by k1,k2
```

5.拓展

```
1 工具: pt-query-digest
```

第3章 二进制日志binlog

1.作用

1. 数据恢复
2. 主从复制

2.记录的内容

- 1 记录修改类操作(逻辑日志, 类似于SQL记录)
- 2 DML: insert update delete
- 3 DDL: create drop alter truncate
- 4 DCL: grant revoke

3.配置方法

3.1 基础参数查看

```
1 mysql> select @@log_bin;
2 mysql> select @@log_bin_basename;
3 mysql> select @@server_id;
```

3.2 设置基础参数

```
1 vim /etc/my.cnf
2 [mysqld]
3 #新增加参数
4 server_id=51                                #主机ID, 在主从复制会使用
5 log_bin=/data/mysql_3306/logs/mysql-bin    #开关+文件路径+文件名前缀, 最终格式:
mysql-bin.000001
```

3.3 重启并查看

```
1 [root@db-51 ~]# systemctl restart mysqld
2 [root@db-51 ~]# ll /data/mysql_3306/logs/
3 总用量 20
4 -rw-r----- 1 mysql mysql    154 9月  13 17:29 mysql-bin.000001
5 -rw-r----- 1 mysql mysql     39 9月  13 17:29 mysql-bin.index
6 -rw-r--r-- 1 mysql mysql 11256 9月  13 17:29 mysql.err
```

4.binlog内容的记录格式

4.1 事件(event)的记录方式

每个事件:

1. 事件描述: 时间戳、server_id、加密方式、开始的位置 (start_pos)、结束位置点 (end_pos)
2. 事件内容: 修改类的操作: SQL 语句 或者 数据行变化。

重点关注:

- 1 开始的位置 (start_pos)
- 2 结束位置点 (end_pos)
- 3 事件内容

4.2 二进制日志事件内容格式

查看日志格式:

```

1  mysql> select @@binlog_format;
2  +-----+
3  | @@binlog_format |
4  +-----+
5  | ROW              |
6  +-----+

```

作用:

- 1 对于DDL、DCL语句，直接将SQL本身记录到binlog中
- 2 对于DML：insert、update、delete 受到binlog_format参数控制。
- 3 SBR：Statement：语句模式。之前版本，默认模式
- 4 RBR：ROW：行记录模式。5.7以后，默认模式
- 5 MBR：mixed：混合模式。

区别:

- 1 Statement、ROW区别：
- 2 update t1 set name='zhangsan' where id<100;
- 3
- 4 Statement：记录SQL本身。
- 5 ROW：100个数据行的变化。
- 6
- 7 Statement日志量少
- 8 ROW日志量大
- 9
- 10 Statement记录不够准确
- 11 ROW记录够准确。
- 12
- 13 例如函数操作：
- 14 now()
- 15 rand()

5.binlong查询

5.1 日志文件情况查询

查看所有的日志文件信息

```
1  mysql> show binary logs;
2  +-----+-----+
3  | Log_name          | File_size |
4  +-----+-----+
5  | mysql-bin.000001  |      154 |
6  +-----+-----+
```

刷新新日志

```
1  mysql> flush logs;
2  mysql> flush logs;
3  mysql> flush logs;
4  mysql> flush logs;
5  mysql> show binary logs;
6  +-----+-----+
7  | Log_name          | File_size |
8  +-----+-----+
9  | mysql-bin.000001  | 2074265 |
10 | mysql-bin.000002  |    1181 |
11 | mysql-bin.000003  |     201 |
12 | mysql-bin.000004  |     201 |
13 | mysql-bin.000005  |     154 |
14 +-----+-----+
```

当前数据库使用的二进制日志


```

1  mysql> show master status;
2  +-----+-----+-----+-----+-----+
   |
3  | File           | Position | Binlog_Do_DB | Binlog_Ignore_DB | Executed_Gtid_Set |
   |
4  +-----+-----+-----+-----+-----+
   |
5  | mysql-bin.000005 |      154 |              |                  |                    |
   |
6  +-----+-----+-----+-----+-----+
   |

```

5.2 内容查询

数据模拟

```

1  mysql> create database ku charset utf8mb4;
2  mysql> use ku
3  mysql> create table biao (id int);
4  mysql> insert into biao values(1);
5  mysql> commit;

```

查看日志事件

```

1  mysql> show binlog events in 'mysql-bin.000005';
2  +-----+-----+-----+-----+-----+-----+
   |
3  | Log_name          | Pos | Event_type | Server_id | End_log_pos | Info |
   |
4  +-----+-----+-----+-----+-----+-----+
   |
5  | mysql-bin.000005 |    4 | Format_desc |          6 |          123 |      |
   |
   | Server ver: 5.7.28-log, Binlog ver: 4 |

```

```

6 | mysql-bin.000005 | 123 | Previous_gtid | 6 | 154 |
  |
7 | mysql-bin.000005 | 154 | Anonymous_Gtid | 6 | 219 |
  SET @@SESSION.GTID_NEXT= 'ANONYMOUS' |
8 | mysql-bin.000005 | 219 | Query | 6 | 323 |
  create database ku charset utf8mb4 |
9 | mysql-bin.000005 | 323 | Anonymous_Gtid | 6 | 388 |
  SET @@SESSION.GTID_NEXT= 'ANONYMOUS' |
10 | mysql-bin.000005 | 388 | Query | 6 | 484 |
  use `ku`; create table biao (id int) |
11 | mysql-bin.000005 | 484 | Anonymous_Gtid | 6 | 549 |
  SET @@SESSION.GTID_NEXT= 'ANONYMOUS' |
12 | mysql-bin.000005 | 549 | Query | 6 | 619 |
  BEGIN |
13 | mysql-bin.000005 | 619 | Table_map | 6 | 664 |
  table_id: 111 (ku.biao) |
14 | mysql-bin.000005 | 664 | Write_rows | 6 | 704 |
  table_id: 111 flags: STMT_END_F |
15 | mysql-bin.000005 | 704 | Xid | 6 | 735 |
  COMMIT /* xid=88 */ |
16 +-----+-----+-----+-----+-----+-----+
  -----+

```

5.3 查看日志内容

查看日志内容

```
1 [root@db01 logs]# mysqlbinlog /data/mysql_3306/logs/mysql-bin.000005
```

Create database日志内容

```

1 # at 219
2 ...略
3 create database ku charset utf8mb4

```

create table 日志内容

```
1 # at 388
2 ...略
3 create table biao (id int)
```

insert 操作的日志内容

```
1 # at 664
2 #200914 8:15:15 server id 6 end_log_pos 704 CRC32 0x0a91b6f8
  Write_rows: table id 111 flags: STMT_END_F
3
4 BINLOG '
5 E7ZeXxMGAAAALQAAAJgCAAAAAG8AAAAAAAEAAmt1AARidWFvAAEDAAGlD2wp
6 E7ZeXx4GAAAkAAAAAMACAAAAG8AAAAAAAEAAgAB//4BAAAA+LaRCg==
7 '/*!*/;
8 # at 704
9 #200914 8:15:15 server id 6 end_log_pos 735 CRC32 0x1e620e90 Xid =
  88
10 COMMIT/*!*/;
```

查看解密后的insert

```
1 [root@db-51 ~]# mysqlbinlog --base64-output=decode-rows -vv
  /data/mysql_3306/logs/mysql-bin.000005
2 .....略
3 # at 664
4 #200914 8:15:15 server id 6 end_log_pos 704 CRC32 0x0a91b6f8
  Write_rows: table id 111 flags: STMT_END_F
5 ### INSERT INTO `ku`.`biao`
6 ### SET
7 ### @1=1 /* INT meta=0 nullable=1 is_null=0 */
```

6.binlog日志截取及恢复演练

6.1 前提说明

- 1 创建或导入数据库之前就配置并开启了binlog

6.2 故障说明

- 1 模拟误删库，要求恢复到删库之前

6.3 模拟故障

```
1  1.创建库
2  create database linux5;
3
4  2.创建表
5  use linux5;
6  CREATE TABLE `user` (
7    `id` int(11) NOT NULL AUTO_INCREMENT COMMENT 'id',
8    `name` varchar(10) NOT NULL COMMENT 'name',
9    `age` tinyint(4) NOT NULL COMMENT 'age',
10   PRIMARY KEY (`id`)
11 ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
12
13 3.写入数据
14 insert into user(name,age)
15 values
16 ('z3',22),
17 ('l4',22),
18 ('w5',22);
19
20 4.查看数据
21 mysql> select * from user;
22 +-----+-----+-----+
23 | id | name | age |
```

```

24  +-----+-----+-----+
25  |  1  | z3    |  18  |
26  |  2  | l4    |  20  |
27  |  3  | w5    |  21  |
28  +-----+-----+-----+
29
30  5. 模拟删除
31  drop database linux5;

```

6.4 恢复思路

第一步：截取从建库以来到删库之前的所有binlog

查看当前处于什么哪个binlog：

```

1  mysql> show master status;
2  +-----+-----+-----+-----+-----+
   |-----+
3  | File           | Position | Binlog_Do_DB | Binlog_Ignore_DB |
   | Executed_Gtid_Set |
4  +-----+-----+-----+-----+-----+
   |-----+
5  | mysql-bin.000002 |      1134 |              |                  |
   |                  |
6  +-----+-----+-----+-----+-----+
   |-----+

```

第二步: 找到起点,建库的位置点 (position)

```

1  mysql> show binlog events in 'mysql-bin.000002';
2  ....略
3  | mysql-bin.000002 | 219 | Query          | 6 | 319 |
   | create database linux5

```

第三步: 找到终点

```
1  ....略
2  | mysql-bin.000002 | 1036 | Query          |          6 |          1134 |
   drop database linux5
```

导出数据：

```
1  mysqlbinlog --start-position=219 --stop-position=1036
   /data/mysql_3306/logs/mysql-bin.000002 >/tmp/bin.sql
```

将截取的日志进行回放

```
1  mysql> set sql_log_bin=0;
2  mysql> source /tmp/bin.sql;
3  mysql> set sql_log_bin=1;
```

7.生产中日志在多个文件中,如何截取?

7.1 场景模拟

```
1  flush logs;
2
3  #mysql-bin.000005
4  show master status ;
5  create database tongdian charset=utf8mb4;
6  use tongdian
7  create table t1 (id int);
8  flush logs;
9
10 #mysql-bin.000006
11 show master status ;
12 insert into t1 values(1),(2),(3);
13 commit;
14 flush logs;
15
```

```
16 #mysql-bin.000007
17 show master status ;
18 create table t2(id int);
19 insert into t2 values(1),(2),(3);
20 commit;
21 flush logs;
22
23 #mysql-bin.000008
24 show master status ;
25 insert into t2 values(11),(22),(33);
26 commit;
27 drop database tongdian;
```

7.2 恢复方法

方法1:分段截取

```
1 --start-position      --stop-position
```

方法2:时间戳截取

a.找起点: 建库的时间戳

起点posting号

```
1 show binlog events in 'mysql-bin.000005';
```

通过position过滤时间戳

```
1 mysqlbinlog --start-position=951 --stop-position=1073 mysql-bin.000005
  |grep -A 1 '^#\ at\ 951'
```

b.找终点

```
1 mysql -e "show binlog events in 'mysql-bin.000008'"
```

c.截取日志

```
1 mysqlbinlog --start-datetime="2020-05-09 17:11:23" --stop-  
  datetime="2020-05-09 17:14:01" mysql-bin.000005 mysql-bin.000006  
  mysql-bin.000007 mysql-bin.000008 >/tmp/data.sql
```

7.3 binlog其他注意

binlog属于全局日志，日志中有其他库的操作，怎么排除掉？

```
1 mysqlbinlog -d oldboy mysql-bin.000008 > /tmp/bin.sql
```

binlog中100w个事件，怎么快速找到drop database的位置点？

```
1 [root@db01 ~]# mysql -e "show binlog events in 'mysql-bin.000014'" |less  
2 [root@db01 ~]# mysql -e "show binlog events in 'mysql-bin.000014'" |grep
```

比如删除的库，建库是在2年前操作的。这种情况怎么办？

- 1 每天全备，binlog完好的。
- 2 可以使用 全备+binlog方式实现恢复数据故障之前。

8.基于GTID的binlog应用

8.1 GTID全局事务ID

- 1 对每个事务，进行单独编号。连续不断进行增长。

8.2 表示方式

```
1 server_uuid:N
```

8.3 GTID配置

查看参数:

```
1 mysql> show variables like '%GTID%';
2 +-----+-----+
3 | Variable_name          | Value      |
4 +-----+-----+
5 | binlog_gtid_simple_recovery | ON         |
6 | enforce_gtid_consistency  | OFF        |
7 | gtid_executed_compression_period | 1000       |
8 | gtid_mode                | OFF        |
9 | gtid_next                 | AUTOMATIC  |
10 | gtid_owned                |            |
11 | gtid_purged                |            |
12 | session_track_gtid        | OFF        |
13 +-----+-----+
```

设置参数:

```
1 vim /etc/my.cnf
2 [mysqld]
3 gtid_mode=ON          #开关
4 enforce_gtid_consistency=ON #强制GTID一致性
5 log_slave_updates=ON   #强制从库更新binlog
```

重启服务:

```
1 systemctl restart mysqld
```

建议:

- 1 5.7版本以后, 都开启GTID。最好是搭建环境就开启。

8.4 GTID应用

模拟环境:

a.创建库并查看gtid

```
1 mysql> create database gtdb charset utf8mb4;
2 mysql> show master status ;
3 +-----+-----+-----+-----+-----+
   | File | Position | Binlog_Do_DB | Binlog_Ignore_DB | Executed_Gtid_Set |
4 +-----+-----+-----+-----+-----+
6 | mysql-bin.000006 | 329 | | | 9b52b744-eb82-11ea-986c-000c294983f8:1 |
7 +-----+-----+-----+-----+-----+
```

b.创建表并查看gtid

```

1  mysql> use gtdb;
2  mysql> create table t1(id int);
3  mysql> show master status ;
4  +-----+-----+-----+-----+-----+
    | File           | Position | Binlog_Do_DB | Binlog_Ignore_DB | Executed_Gtid_Set |
5  +-----+-----+-----+-----+-----+
6  | mysql-bin.000006 |      491 |              |                  | 9b52b744-eb82-11ea-986c-000c294983f8:1-2 |
7  +-----+-----+-----+-----+-----+

```

c.插入数据并查看

```

1  begin;
2  insert into t1 values(1);
3  insert into t1 values(2);
4  insert into t1 values(3);
5  commit;
6  show master status ;
7  +-----+-----+-----+-----+-----+
    | File           | Position | Binlog_Do_DB | Binlog_Ignore_DB | Executed_Gtid_Set |
8  +-----+-----+-----+-----+-----+
9  | mysql-bin.000006 |      914 |              |                  | 9b52b744-eb82-11ea-986c-000c294983f8:1-3 |
10 +-----+-----+-----+-----+-----+
11

```

d.查看事件

```

1 mysql> show binlog events in 'mysql-bin.000006';
2 +-----+-----+-----+-----+-----+-----+
   | Log_name          | Pos | Event_type          | Server_id | End_log_pos |
   | Info              |
3 +-----+-----+-----+-----+-----+-----+
   | mysql-bin.000006 | 4   | Format_desc         | 6         | 123         |
   | Server ver: 5.7.28-log, Binlog ver: 4 |
4 +-----+-----+-----+-----+-----+-----+
   | mysql-bin.000006 | 123 | Previous_gtids      | 6         | 154         |
   |
5 | mysql-bin.000006 | 154 | Gtid                | 6         | 219         |
   | SET @@SESSION.GTID_NEXT= '9b52b744-eb82-11ea-986c-000c294983f8:1' |
6 | mysql-bin.000006 | 219 | Query              | 6         | 329         |
   | create database gtdb charset utf8mb4 |
7 | mysql-bin.000006 | 329 | Gtid                | 6         | 394         |
   | SET @@SESSION.GTID_NEXT= '9b52b744-eb82-11ea-986c-000c294983f8:2' |
8 | mysql-bin.000006 | 394 | Query              | 6         | 491         |
   | use `gtdb`; create table t1(id int) |
9 | mysql-bin.000006 | 491 | Gtid                | 6         | 556         |
   | SET @@SESSION.GTID_NEXT= '9b52b744-eb82-11ea-986c-000c294983f8:3' |
10 | mysql-bin.000006 | 556 | Query              | 6         | 628         |
   | BEGIN |
11 | mysql-bin.000006 | 628 | Table_map          | 6         | 673         |
   | table_id: 108 (gtdb.t1) |
12 | mysql-bin.000006 | 673 | Write_rows         | 6         | 713         |
   | table_id: 108 flags: STMT_END_F |
13 | mysql-bin.000006 | 713 | Table_map          | 6         | 758         |
   | table_id: 108 (gtdb.t1) |
14 | mysql-bin.000006 | 758 | Write_rows         | 6         | 798         |
   | table_id: 108 flags: STMT_END_F |
15 | mysql-bin.000006 | 798 | Table_map          | 6         | 843         |
   | table_id: 108 (gtdb.t1) |
16 | mysql-bin.000006 | 843 | Write_rows         | 6         | 883         |
   | table_id: 108 flags: STMT_END_F |

```

```

19 | mysql-bin.000006 | 883 | Xid | 6 | 914 |
    COMMIT /* xid=12 */ |
20 +-----+-----+-----+-----+-----+-----+
    -----+

```

8.5 通过GTID方式截取日志

错误的截取命令:

```

1  mysqlbinlog --include-gtids='9b52b744-eb82-11ea-986c-000c294983f8:1-3'
   /data/mysql_3306/logs/mysql-bin.000006 >/tmp/gt.sql

```

为什么恢复报错?

- 1 gtid有“幂等性”检查。GTID的生成, 通过Set gtid_next命令实现的。
- 2 例如:
- 3 SET @@SESSION.GTID_NEXT= '202628e9-9265-11ea-b4a0-000c29248f69:1'
- 4 执行Set命令时, 自动检查当前系统是否包含这个GTID信息, 如果有就跳过。

正确的方式:

```

1  mysqlbinlog --skip-gtids --include-gtids='2ddd7a11-4747-11eb-b274-
   000c29116b18:1-3' /data/mysql_3306/binlog/mysql-bin.000009 >/tmp/gt.sql

```

8.6 恢复操作

```

1  set sql_log_bin=0;
2  source /tmp/gt.sql;
3  set sql_log_bin=1;

```

9.日志滚动

命令触发：

```
1  mysql> flush logs;
2  shell# mysqladmin flush-logs
3  shell# mysql -e "flush logs"
4  shell# mysqldump -F
```

自动触发：

```
1  mysql> select @@max_binlog_size;
2  +-----+
3  | @@max_binlog_size |
4  +-----+
5  |           1073741824 |
6  +-----+
7
8  重启数据库，会触发刷新
```

10.日志删除

10.1 默认方式

```
1  不自动清理。直到空间写满。
```

10.2 配置自动清理

```
1  mysql> select @@expire_logs_days;
2  最少设置多少天合适?
3  参考全备时间周期。
4  例如： 全备周期是7天。可以保留8天。一般生产中保留两轮备份周期的日志，15天。
5
6  设置命令：
7  set GLOBAL expire_logs_days=7;
```

10.3 手工清理

```
1  Examples:
2  PURGE BINARY LOGS TO 'mysql-bin.010';
3  PURGE BINARY LOGS BEFORE '2019-04-02 22:46:26';
```