# 概述

日志是记录了MySQL数据库的各种类型活动的数据。DBA可以利用这些日志文件定位故障，优化性能等。

# 分类

在MySQL中，有4种不同的日志，分别是：错误日志（errorlog）、二进制日志（binlog）、查询日志（log）和慢查询日志（slow query log）。



## 错误日志

### 概述

错误日志记录了当mysqld进程启动和停止时，以及服务器在运行过程中发生任何严重错误时的相关信息。

当数据库出现任何故障导致无法正常使用时，可以首先查看此日志。

### 设置

默认是开启的，可以通过修改my.cnf文件自定义，如：

log\_error=path默认存在$datadir/hostname.err

**查看：**

SHOW VARIABLES LIKE ‘log\_error’\G;

**路径设置：**

可以使用—log-error=[file\_name]选项来指定mysqld（MySQL服务器）保存错误日志文件的位置。如果没有给定file\_name值，mysqld使用错误日志名host\_name.err并默认在参数DATADIR（数据目录）指定的目录中写入日志文件。

在实际应用中，如果数据库启动报错或者crash，或者出现告警信息等，可以通过查看错误日志获取必要的信息。

## 二进制日志

### 概述

二进制文件记录了对MySQL数据库的所有更新操作（其中还包括执行更新操作的时间等额外信息），不包括查询和SHOW这类操作（binlog作用就是备份恢复使用的，所以只需要记录修改操作即可）。

二进制文件默认关闭，需要手动指定参数启动。根据MySQL官方手册的测试数据，开启二进制日志会使性能下降1%，但是考虑到可以使用复制（replication）和point-in-time的恢复，这些性能的损失绝对是可以接受的。

**查看：**

SHOW BINLOG EVENT IN ‘mysqld.00001’\G;

**配置：**

通过参数log-bin[=name]可以开启二进制日志，如果不指定name，默认二进制日志文件名为主机名，后缀名为二进制日志的序列号，所在路径为数据库所在目录（datadir）。

查看datadir：show variables like ‘datadir’;

### 参数

max\_binlog\_size：指定了单个二进制日志文件最大值，如果超过该值，则产生新的二进制日志文件后缀名+1，并记录到.index文件。

binlog\_cache\_size：控制缓冲大小，默认大小32K，基于会话的，因此每开启一个事务就分配一个binlog\_cache\_size大小的缓存，所以不能设置过大。当一个事务的记录大于binlog\_cache\_size时，MySQL会把缓冲中的日志写入一个临时文件中，因此该值又不能设置太小。

通过SHOW GLOBAL STATUS命令查看binlog\_cache\_use、binlog\_cache\_disk\_use的状态，可以判断当前binlog\_cache\_size的设置是否合理。binlog\_cache\_use记录了使用缓冲写二进制日志的次数，binlog\_cache\_disk\_use记录了使用临时文件写二进制日志的次数。

sync\_binlog：表示每写缓冲多少次就要同步到磁盘。如果设置为1，表示采用同步写磁盘的方式来写二进制日志，这时候写操作不使用操作系统的缓冲来写二进制日志。sync\_binlog的默认值为0，如果使用InnoDB存储引擎进行复制，并且想得到最大的可用性，建议将该值设置为ON（对数据库IO系统带来一定影响）。

但是，即使将sync\_binlog设置为1，还是会有一种情况导致问题发生。当使用InnoDB存储引擎时，在一个事务发出COMMIT动作之前，由于sync\_binlog为1，因此会将二进制日志立即写入磁盘。如果这时写入了二进制日志，但是提交还没有发生，并且此时发生了宕机，那么在MySQL数据库下次启动时，由于COMMIT操作并没有发生，这个事务被回滚。但是二进制日志已经记录了该事物信息，不能被回滚。这个问题可以通过参数innodb\_support\_xa设置为1来解决，虽然innodb\_support\_xa与XA事务有关，但是它同时也确保了二进制日志和InnoDB存储引擎文件的同步。

binlog-do-db：表示需要写入哪些库的日志，默认为空，表示需要同步所有库的日志到二进制日志。

binlog-ignore-db：表示需要忽略写入哪些库的日志，默认为空，表示需要同步所有库的日志到二进制日志。

log-slave-update：如果当前数据库是复制中的slave节点，则它不会将从master取得并执行的二进制日志写入自己的二进制文件中。如果需要写入，要设置log-slave-update。如果需要搭建master->slave->slave这种架构的复制，则必须设置该参数。

binlog\_format：记录二进制日志的格式。在MySQL5.1之前，没有这个参数，所有二进制文件的格式都是基于SQL语句（statement）级别的，因此基于这个格式的二进制日志文件的复制（Replication）和Oracle的逻辑Standby有点类似。

该值可以设置为STATEMENT、ROW和MIXED。

1. STATEMENT格式下，记录的是逻辑SQL语句。
2. ROW格式下，记录表的行更改情况。
3. MIXED格式下，MySQL默认采用STATEMENT格式进行二进制文件记录，但是在一些情况下会使用ROW格式，这些情况包括：
4. 表的存储引擎为NDB，这时对表的DML操作都会以ROW格式记录；
5. 使用了UUID()、USER()、CURRENT\_USER()、FOUND\_ROWS()、ROW\_COUNT()等不确定函数；
6. 使用了INSERT DELAY语句；
7. 使用了用户定义函数（UDF）；
8. 使用了临时表（temporary table）。

此外，binlog\_format参数还有对于存储引擎的限制：



在通常情况下，我们将参数binlog\_format设置为ROW，这可以为数据库的恢复和复制带来更好的可靠性。但是不能忽略的一点是，这会带来二进制文件大小的增加，这些语句的ROW格式可能需要更大的容量。而由于复制是采用传输二进制日志方式实现的，因此复制的网络开销也会增加。

要查看二进制日志文件内容，必须通过MySQL提供的工具mysqlbinlog。对于STATEMENT格式的二进制日志文件，在使用mysqlbinlog后，看到的就是执行的逻辑SQL语句。但是，如果使用ROW格式记录，会发现mysqlbinlog的结果变得“不可读”，其实只要加上参数-v或-vv就能清楚地看到执行的具体信息了（-vv会比-v显示更新的类型）。

### 作用

二进制日志的主要作用如下：

1. 恢复（recovery）：某些数据的恢复需要二进制日志，例如，如果需要恢复数据库全量备份的文件，可以通过二进制日志进行point-in-time恢复。
2. 复制（replication）：通过复制和执行二进制日志使得远程的MySQL数据库（一般称为slave或standby）与一台MySQL数据库（一般称为master或primary）进行实时同步。
3. 审计（audit）：用户通过二进制日志中的信息来进行审计，判断是否有对数据库进行注入的攻击。

## 查询日志

### 概述

当客户端连接或断开时，服务器会将信息写入该日志，并记录从客户端收到的每一条SQL语句。当你怀疑客户端的错误并想知道客户端发送给mysqqld的确切消息时，一般查询日志可能非常有用。

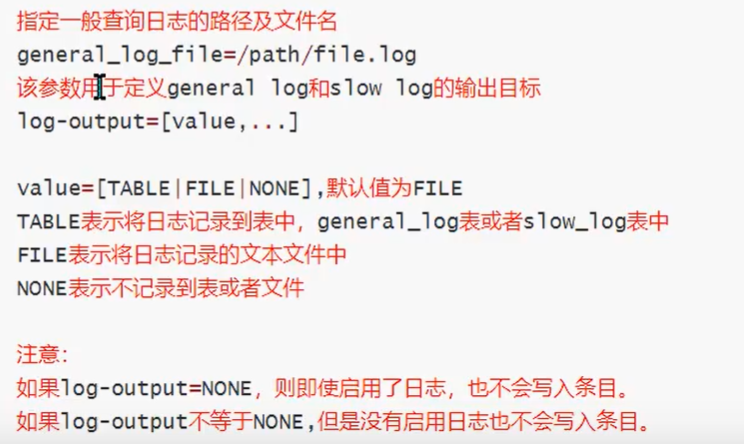
查询日志记录了所有对MySQL数据库的请求信息，不论这些请求是否得到正确的响应。

默认文件名为：主机名.log。

默认情况下，一般查询日志是被禁用的。如果需要开启，可以使用一下参数：

general\_log={0|1} //0表示禁用，1表示开启

默认情况下，系统会在数据目录下创建host\_name.log命令的一般查询日志。如果要自己指定，可以使用以下参数：



### 区别

**二进制日志与查询日志区别：**

二进制日志不包含只查询数据的语句，查询日志记录了客户端的所有语句。

## 慢查询日志

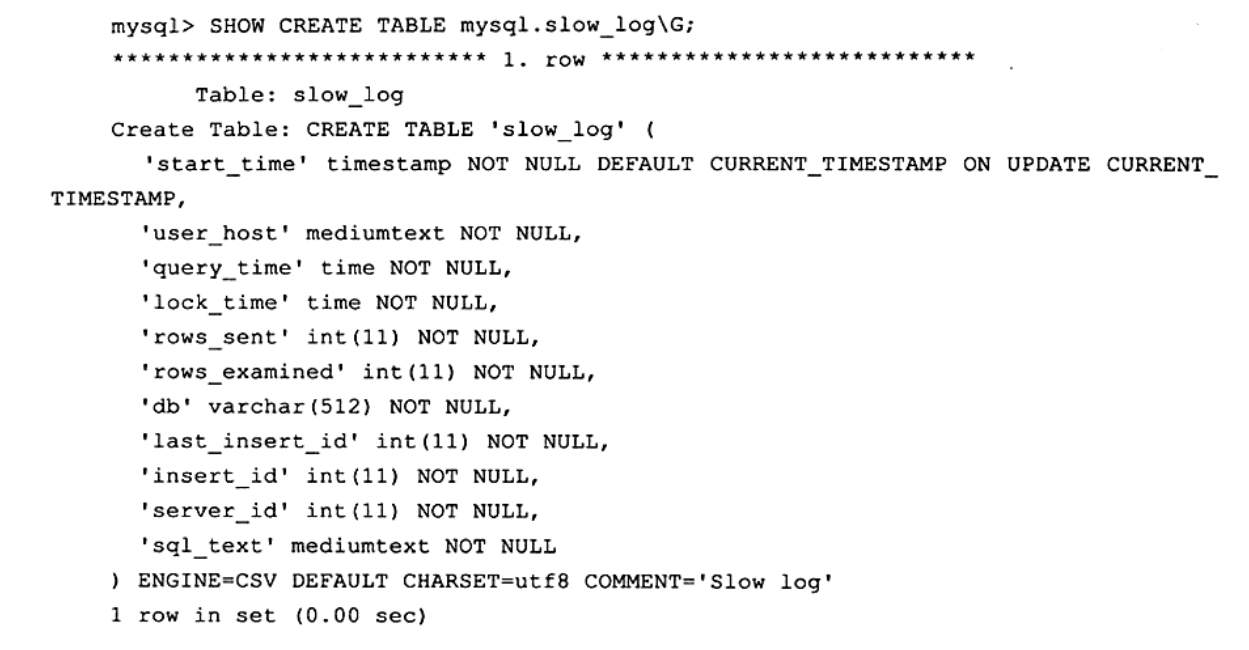
### 概述

慢查询日志记录了所有执行时间超过参数long\_query\_time（单位：秒，运行时间等于long\_query\_time的情况不会被记录）设置值并且扫描记录数不小于min\_examinied\_row\_limit的所有SQL语句的日志（注意，获得表锁定的时间不算做指定时间）。

慢查询日志默认关闭，需要手动开启。



MySQL5.1开始，支持将慢查询日志记录到表中，这样用户查询就更加方便直观了。慢查询表在MySQL架构下，名称为slow\_log，表结构定义如下：



### 参数

set slow\_query\_log：是否开启慢查询日志，通过set slow\_query\_log=on;开启。

long\_query\_time：阈值。long\_query\_time默认为10秒，最小为0，从MySQL5.1开始，精度可以到微秒。

min\_examinied\_row\_limit：允许扫描的最小行数。

另一个和慢查询日志相关的参数是long\_queries\_not\_using\_indexes，如果运行SQL未命中索引，则MySQL数据库同样会将这条SQL语句记录到慢查询日志中（这样方便后续优化）。

MySQL5.6新增一个参数log\_throttle\_queries\_not\_using\_indexes，用来表示每分钟允许记录到slow log且未命中索引的SQL语句次数。默认为0，表示没有限制。在生产环境中，如果没有命中索引，则此类SQL语句会频繁地被记录到slow log，从而导致慢查询日志大小不断增加，可通过修改此参数配置。

用户可以通过参数long\_query\_io将超过指定逻辑IO次数的SQL语句记录到slow\_log中，该值默认为100。

为了兼容原MySQL数据库的运行方式，还添加了参数slow\_query\_type，用来表示启用slow log的方式。，可选的值为：

0：表示不把SQL语句记录到slow log

1：表示根据运行时间将SQL语句记录到slow log

2：表示根据逻辑IO次数将SQL语句记录到slow log

3：表示根据运行时间及逻辑IO次数将SQL语句记录到slow log

### 分析

如果在慢查询日志中搜索，则会非常不方便，分析起来很麻烦，MySQL提供了mysqldumpslow工具帮助分析慢查询日志。

指令：mysqldumpslow \*\*\*-slow.log

如果希望得到执行时间最长的5条SQL语句，操作如下：

mysqldumpslow -s -al -n 5 \*\*\*-slow.log

### 分布式数据库实践

分布式数据库的慢查询包括两种：计算节点proxy慢查询、数据节点DB慢查询和DB锁等待日志。

#### Proxy慢查询

Proxy慢查询开关、阈值、日志文件路径都可以通过配置文件设置。Proxy慢查询日志关键内容分析：

TotalExecTime：从应用接收SQL到返回应用响应的总时间

SQL：执行的SQL语句

MsgToExecTime：从消息线程发送到执行线程的时间

ParserSQLTime：解析SQL语法的时间

PlanTreeCreateTime：创建执行计划的时间

GetGTIDTime：获取GTID的时间

GreeGTIDTime：释放GTID时间

PlanTreeExecTime：执行计划树的执行时间

SubSQL[N]：第N条子查询

ExecTime：子查询的执行时间

FinishTime：proxy处理子查询的时间

g1 num:1,duration:104577us：执行子查询的某个分片的执行时间

DB connection\_id:2534578,duration:5467us：DB上执行时间

#### DB慢查询

#### DB锁等待