

Mysql主从复制

1. mysql的uuid不能一致
2. 主MySQL的server_id必须比从的小（默认没有，需要设置）
3. 开启主mysql的日志（默认未开启）
4. 设置主mysql的日志名称

原作者链接传送门

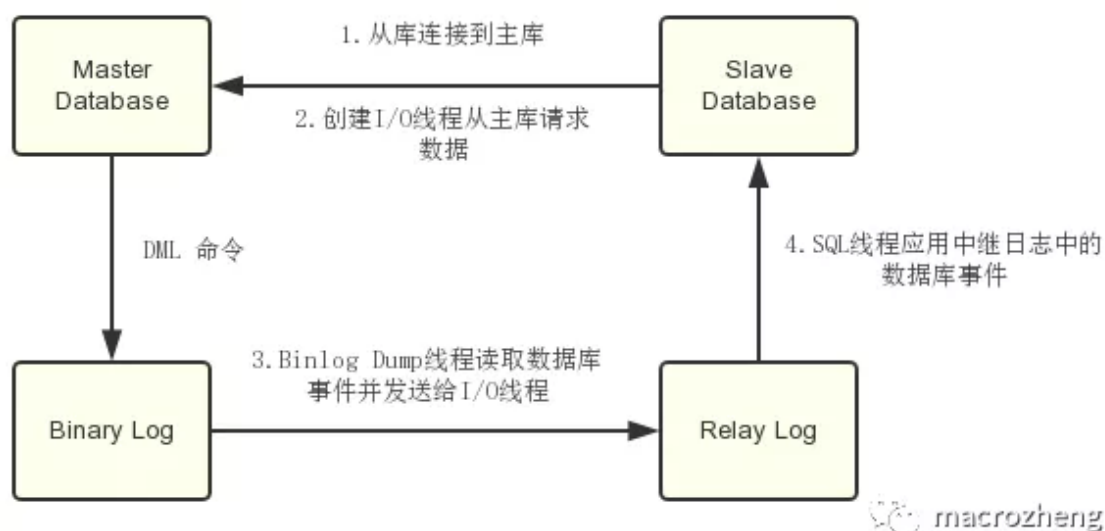
<https://mp.weixin.qq.com/s/eEWMSTAUF1H-gFBx26jujw>

什么是主从复制？

主从复制是指将主数据库的DDL和DML操作通过二进制日志传到从数据库上，然后在从数据库上对这些日志进行重新执行，从而使从数据库和主数据库的数据保持一致。

主从复制的原理

- MySQL主库在事务提交时会把数据变更作为事件记录在二进制日志Binlog中；
- 主库推送二进制日志文件Binlog中的事件到从库的中继日志Relay Log中，之后从库根据中继日志重做数据变更操作，通过逻辑复制来达到主库和从库的数据一致性；
- MySQL通过三个线程来完成主从库间的数据复制，其中Binlog Dump线程跑在主库上，I/O线程和SQL线程跑着从库上；
- 当在从库上启动复制时，首先创建I/O线程连接主库，主库随后创建Binlog Dump线程读取数据库事件并发送给I/O线程，I/O线程获取到事件数据后更新到从库的中继日志Relay Log中去，之后从库上的SQL线程读取中继日志Relay Log中更新的数据库事件并应用，如下图所示。



主实例搭建

- 运行mysql主实例：

```
docker run -p 3307:3306 --name mysql-master \
-v /mydata/mysql-master/log:/var/log/mysql \
-v /mydata/mysql-master/data:/var/lib/mysql \
-v /mydata/mysql-master/conf:/etc/mysql \
-e MYSQL_ROOT_PASSWORD=root \
-d mysql:5.7
```

- 在mysql的配置文件夹 /mydata/mysql-master/conf 中创建一个配置文件 my.cnf：

```
touch my.cnf
```

- 修改配置文件my.cnf，配置信息如下：

```
[mysqld]
## 设置server_id，同一局域网中需要唯一
server_id=101
## 指定不需要同步的数据库名称
binlog-ignore-db=mysql
## 开启二进制日志功能
log-bin=mysql-bin
## 设置二进制日志使用内存大小（事务）
binlog_cache_size=1M
## 设置使用的二进制日志格式（mixed,statement,row）
binlog_format=mixed
## 二进制日志过期清理时间。默认值为0，表示不自动清理。
expire_logs_days=7
## 跳过主从复制中遇到的所有错误或指定类型的错误，避免slave端复制中断。
## 如：1062错误是指一些主键重复，1032错误是因为主从数据库数据不一致
slave_skip_errors=1062
```

- 修改完配置后重启实例：

```
docker restart mysql-master
```

- 进入mysql-master容器中：

```
docker exec -it mysql-master /bin/bash
```

- 在容器中使用mysql的登录命令连接到客户端：

```
mysql -uroot -proot
```

- 创建数据同步用户：

```
CREATE USER 'slave'@'%' IDENTIFIED BY '123456';
GRANT REPLICATION SLAVE, REPLICATION CLIENT ON *.* TO 'slave'@'%';
```

从实例搭建

- 运行mysql从实例：

```
docker run -p 3308:3306 --name mysql-slave \
-v /mydata/mysql-slave/log:/var/log/mysql \
-v /mydata/mysql-slave/data:/var/lib/mysql \
-v /mydata/mysql-slave/conf:/etc/mysql \
-e MYSQL_ROOT_PASSWORD=root \
-d mysql:5.7
```

- 在mysql的配置文件夹 /mydata/mysql-slave/conf 中创建一个配置文件 my.cnf：

```
touch my.cnf
```

- 修改配置文件my.cnf：

```
[mysqld]
## 设置server_id，同一局域网中需要唯一
server_id=102
## 指定不需要同步的数据库名称
binlog-ignore-db=mysql
## 开启二进制日志功能，以备Slave作为其它数据库实例的Master时使用
log-bin=mall-mysql-slave1-bin
## 设置二进制日志使用内存大小（事务）
binlog_cache_size=1M
## 设置使用的二进制日志格式（mixed,statement,row）
binlog_format=mixed
## 二进制日志过期清理时间。默认值为0，表示不自动清理。
expire_logs_days=7
## 跳过主从复制中遇到的所有错误或指定类型的错误，避免slave端复制中断。
## 如：1062错误是指一些主键重复，1032错误是因为主从数据库数据不一致
slave_skip_errors=1062
## relay_log配置中继日志
relay_log=mall-mysql-relay-bin
## log_slave_updates表示slave将复制事件写进自己的二进制日志
log_slave_updates=1
## slave设置为只读（具有super权限的用户除外）
read_only=1
```

- 修改完配置后重启实例：

```
docker restart mysql-slave
```

将主从数据库进行连接

- 连接到主数据库的mysql客户端，查看主数据库状态：

```
show master status;
```

- 主数据库状态显示如下：

```
mysql> show master status;
+-----+-----+-----+-----+-----+
| File                               | Position | Binlog_Do_DB | Binlog_Ignore_DB | Executed_Gtid_Set |
+-----+-----+-----+-----+-----+
| mall-mysql-bin.000001             | 617      | mysql        |                   |                   |
+-----+-----+-----+-----+-----+
1 row in set (0.00 sec)
```

- 进入 `mysql-slave` 容器中：

```
docker exec -it mysql-slave /bin/bash
```

- 在容器中使用mysql的登录命令连接到客户端：

```
mysql -uroot -proot
```

- 在从数据库中配置主从复制：

```
change master to master_host='192.168.6.132', master_user='slave',  
master_password='123456', master_port=3307, master_log_file='mall-mysql-  
bin.000001', master_log_pos=617, master_connect_retry=30;
```

- 主从复制命令参数说明：

- `master_host`：主数据库的IP地址；
- `master_port`：主数据库的运行端口；
- `master_user`：在主数据库创建的用于同步数据的用户账号；
- `master_password`：在主数据库创建的用于同步数据的用户密码；
- `master_log_file`：指定从数据库要复制数据的日志文件，通过查看主数据的状态，获取File参数；
- `master_log_pos`：指定从数据库从哪个位置开始复制数据，通过查看主数据的状态，获取Position参数；
- `master_connect_retry`：连接失败重试的时间间隔，单位为秒。

- 查看主从同步状态：

```
show slave status \G;
```

- 从数据库状态显示如下：

```
mysql> show slave status \G;
***** 1. row *****
Slave_IO_State:
  Master_Host: 192.168.6.132
  Master_User: slave
  Master_Port: 3307
  Connect_Retry: 30
  Master_Log_File: mall-mysql-bin.000001
  Read_Master_Log_Pos: 617
  Relay_Log_File: mall-mysql-relay-bin.000001
  Relay_Log_Pos: 4
  Relay_Master_Log_File: mall-mysql-bin.000001
  Slave_IO_Running: No
  Slave_SQL_Running: No
  Replicate_Do_DB:
  Replicate_Ignore_DB:
  Replicate_Do_Table:
  Replicate_Ignore_Table:
  Replicate_Wild_Do_Table:
  Replicate_Wild_Ignore_Table:
  Last_Errno: 0
  Last_Error:
  Skip_Counter: 0
  Exec_Master_Log_Pos: 617
  Relay_Log_Space: 154
  Until_Condition: None
  Until_Log_File:
  Until_Log_Pos: 0
  Master_SSL_Allowed: No
  Master_SSL_CA_File:
  Master_SSL_CA_Path:
  Master_SSL_Cert:
  Master_SSL_Cipher:
  Master_SSL_Key:
  Seconds_Behind_Master: NULL
Master_SSL_Verify_Server_Cert: No
  Last_IO_Errno: 0
  Last_IO_Error:
  Last_SQL_Errno: 0
  Last_SQL_Error:
```


表示还没开始同步

- 开启主从同步：

```
start slave;
```

- 查看从数据库状态发现已经同步：

```
mysql> show slave status \G;
***** 1. row *****
      Slave_IO_State: Waiting for master to send event
        Master_Host: 192.168.6.132
        Master_User: slave
        Master_Port: 3307
        Connect_Retry: 30
        Master_Log_File: mall-mysql-bin.000001
        Read_Master_Log_Pos: 617
        Relay_Log_File: mall-mysql-relay-bin.000002
        Relay_Log_Pos: 325
        Relay Master Log File: mall-mysql-bin.000001
        Slave_IO_Running: Yes
        Slave_SQL_Running: Yes
        Replicate_Do_DB:
        Replicate_Ignore_DB:
        Replicate_Do_Table:
        Replicate_Ignore_Table:
        Replicate_Wild_Do_Table:
        Replicate_Wild_Ignore_Table:
```

 macrozheng


主从复制测试

主从复制的测试方法有很多，可以在主实例中创建一个数据库，看看从实例中是否有该数据库，如果有，表示主从复制已经搭建成功。

- 在主实例中创建一个数据库 mall;

```
mysql> create database mall;
Query OK, 1 row affected (0.01 sec)

mysql> show databases;
+-----+
| Database |
+-----+
| information_schema |
| mall      |
| mysql     |
| performance_schema |
| sys       |
+-----+
5 rows in set (0.00 sec)
```

 macrozheng

- 在从实例中查看数据库，发现也有一个 mall 数据库，可以判断主从复制已经搭建成功。

```
mysql> show databases;  
+-----+  
| Database |  
+-----+  
| information_schema |  
| mall |  
| mysql |  
| performance_schema |  
| sys |  
+-----+  
5 rows in set (0.01 sec)
```