# 数据备份

## 数据备份方式：

### 完全备份：

备份整个数据集,即备份整个数据库;

### 部分备份：

备份部分数据集，如只备份一个表;

# Mysql数据备份：

## 相关资料：

Mysql备份的解决方案：

<https://blog.csdn.net/zizhilianqiu/article/details/89297596>

Mysql备案方式：

<https://www.jianshu.com/p/25cca24a54cf>

备份操作：

<https://blog.csdn.net/wu631464569/article/details/78242038>

全备份：

<https://www.jianshu.com/p/e80041ea9d93>

## Mysql的备份方式：

### 热备份

当数据库进行备份时, 数据库的读写操作均不是受影响；

### 温备份

当数据库进行备份时, 数据库的读操作可以执行, 但是不能执行写操作

### 冷备份

当数据库进行备份时, 数据库不能进行读写操作, 即数据库要下线

## Mysql备份需要考虑的问题

### Mysql存储引擎是否支持

|  |  |  |  |
| --- | --- | --- | --- |
| 存储引擎 | 热备份 | 温备份 | 冷备份 |
| MyISAM | 不支持 | 支持 | 支持 |
| InnoDB | 支持 | 支持 | 支持 |

### Mysql数据库中数据的备份方式

#### 物理备份

通过tar,cp等命令直接打包复制数据库的数据库文件，达到备份的效果;

#### 逻辑备份

通过特定的工具从数据库中导出数据并另存备份，逻辑备份会丢失数据精度；

### 我们需要备份什么数据

了解自己需要备份什么数据，选择合适的备份策略；

### 备份工具

#### mysqldump

**mysqldump**:逻辑备份工具, 适用于所有的存储引擎, 支持温备、完全备份、部分备份、对于InnoDB存储引擎支持热备

#### cp, tar 等归档复制工具

**cp, tar 等归档复制工具**: 物理备份工具, 适用于所有的存储引擎, 冷备、完全备份、部分备份

#### lvm2 snapshot

**lvm2 snapshot**: 几乎热备, 借助文件系统管理工具进行备份

#### mysqlhotcopy

mysqlhotcopy:名不副实的的一个工具, 几乎冷备, 仅支持**MyISAM存储引擎**

#### xtrabackup

xtrabackup:一款非常强大的InnoDB/XtraDB热备工具, 支持完全备份、增量备份, 由percona提供

### 选择合适的备份策略

#### 直接cp,tar复制数据库文件

1.数据量较小, 直接复制数据库文件;

#### mysqldump+复制BIN LOGS

如果数据量还行, 可以先使用mysqldump对数据库进行完全备份, 然后定期备份BINARY LOG达到增量备份的效果

#### lvm2快照+复制BIN LOGS

如果数据量一般, 而又不过分影响业务运行, 可以使使用lvm2的快照对数据文件进行备份, 而后定期备份BINARY LOG达到增量备份的效果

#### Xtrabackup

如果数据量很大, 而又不过分影响业务运行, 可以使用xtrabackup进行完全备份后, 定期使用xtrabackup进行增量备份或差异备份

## 使用cp进行备份

### 查看数据库的信息

#显示数据库

SHOW DATABASES;

#查询当前看库中的表

SHOW TABLES;

### ****向数据库施加读锁****

**当mysql为一个表开启读锁的时候,其他进程包括进程本身没有权利去修改这张表的内容,也没有权利更新,但是可以读取表里面的内容;**

#向所有表施加读锁

FLUSH TABLES WITH READ LOCK;

### ****备份数据文件****

#创建文件夹存放备份数据库文件

mkdir /backup

#拷贝源数据文件

cp -a /var/lib/mysql/\* /backup

#查看目录下的文件

ls /backup

### 4.执行解锁

**unlock tables**

### 5.数据恢复

#删除文件

rm -rf /var/lib/mysql/\*

#将备份的数据文件拷贝回去

1. cp -a /backup/\* /var/lib/mysql/
2. 重启MySQL#重新连接数据并查看mysql

## 使用mysqldump+复制BINARY LOG备份

通过mysqldump进行一次完全备份, 再修改表中的数据, 然后再通过binary log进行恢复 二进制日志需要在mysql配置文件中添加 log\_bin=on 开启；

mysqldump是一个客户端的逻辑备份工具, 可以生成一个重现创建原始数据库和表的SQL语句, 可以支持所有的存储引擎, 对于InnoDB支持热备

官方文档：<https://dev.mysql.com/doc/refman/5.7/en/mysqldump.html>

### #基本语法格式

基本语法:

**mysqldump [options] db\_name [tbl\_name ...]**  恢复需要手动

m**ysqldump [options] --databases db\_name ...**  恢复不需要手动创建数据库

**mysqldump [options] --all-databases**  恢复不需要手动创建数据库

其他选项:

-E, --events: 备份事件调度器

-R, --routines: 备份存储过程和存储函数

--triggers: 备份表的触发器;

--skip-triggers

--master-date[=value]

1: 记录为CHANGE MASTER TO 语句、语句不被注释

2: 记录为注释的CHANGE MASTER TO语句 基于二进制还原只能全库还原 --flush-logs: 日志滚动 锁定表完成后执行日志滚动

### ****查看数据库的信息****

### ****使用****mysqldump****备份数据库****

#查看当前二进制文件的状态, 并记录下position的数字

**mysql -uroot -p -e 'SHOW MASTER STATUS'**

#备份数据库到backup.sql文件中

**mysqldump --all-databases --lock-all-tables > backup.sql**

**mysqldump -uroot -pmysql@958958 itour>**

#创建一个数据库

CREATE DATABASE TEST1;

#记下现在的position

SHOW MASTER STATUS;

cp /var/lib/mysql/mysql-bin.000003 /root #备份二进制文件

#停止MySQL

## 使用MYSQL主从复制实现数据热备份

### 主从复制的优势

1.使用主从复制，让主库负责写，从库负责读，可提高程序的稳定性;

2.可以做数据热备份

3.架构拓展。

### 什么是MYSQL的主从复制

MySQL 主从复制是指数据可以从一个MySQL数据库服务器主节点复制到一个或多个从节点。

MySQL 默认采用异步复制方式，这样从节点不用一直访问主服务器来更新自己的数据，数据的更新可以在远程连接上进行，从节点可以复制主数据库中的所有数据库或者特定的数据库，或者特定的表。

### MYSQL的主从复制原理：

Slave会从master读取binlog来镜像数据同步：

1.当master上的数据发生改变时，master则将改变记录到二进制日志（binary log）,这些记录过程叫做二进制日志事件（binary log events）。

2.slave将master的binary log event拷贝到它的中断日志中（relay log）；

3.slave重做中断日志中的事件，将改变应用到自己的数据库中，MYSQL复制是异步的且事件化的。

### MYSQL主从复制的基本原则：

每个slave只有一个maser

每个slave只能有一个唯一的服务器ID

每个master可以有多个slave

### MYSQ主从复制中最大的问题：

延迟

### MYSQL的主从复制配置（Docker方式）

1. 准备两台机器，安装MYSQL（版本建议一致）
2. 配置MYSQL，开启二进制日志
3. 构建主从复制

相关资料：

主从复制配置：

<https://www.cnblogs.com/phpstudy2015-6/p/6485819.html>

<https://blog.csdn.net/mrbuffoon/article/details/103082295>

<https://www.cnblogs.com/songwenjie/p/9371422.html>

<https://blog.csdn.net/daicooper/article/details/79905660>

主从复制-主库已有数据的解决方案

<https://www.cnblogs.com/songwenjie/p/9376719.html>

#### 1.配置主MySQL：

**1.1修改配置文件**

#必须,启用mysql二进制日志

log-bin = 自己的路径/mysql-bin

#必须,主服务器ID

server-id = 1

#可选,启用错误日志

log-err = 自己的路径/mysqlerr

#可选，根目录

baseurl = “自己本地路径”

#可选，临时目录

tmpdir= “自己本地路径”

#可选，数据目录

datadir=” 自己本地路径/data”

#可选，读写都可以

read-only=0

#可选，设置不要复制的数据库

binlog-ignore-db=mysql

#可选，设置需要复制的数据库

Binlog-do-db=需要复制的主数据库名字

1.2重启MySQL关闭防火墙或开放端口

Docker restart 容器名称或容器ID

1.3查看二进制日志是否开启

show global variables like ‘%log%’;

1.4查看主节点二进制日志列表

show master logs;

1.5查看主节点的server id

show global variables like '%server%';

1.6在主机上创建用于主从复制的账号并授权个REPLICATION SLAVE；

CREATE USER 'ben'@'%' IDENTIFIED BY 'ben@958958';

GRANT REPLICATION SLAVE, REPLICATION CLIENT ON \*.\* TO 'ben'@'%';

从机器数据库IP：%

1.7刷新

flush privileges;

1.8查看**主MYSQL**的Status

show master status;

#### 配置从MYSQL

2.1修改配置文件

#可选,开启mysql二进制日志（在主从复制的场景可以不配置,建议配置）

log-bin = mysql-bin

#必须,全局唯一的server\_id

server-id = 2

2.2重启mysql关闭防火墙或开发端口

Docker restart 容器名称或容器ID

2.3查看日志是否开启

show global variables like '%log%';

2.4查看从节点的server id

show global variables like '%server%';

2.5查看**主MYSQL**的Status

show master status;

2.6在从节点配置访问主节点的参数信息

change master to　master\_host='1.116.226.147', master\_user='ben', master\_password='ben@958958',　 master\_port=3308, master\_log\_file='binlog.000025', master\_log\_pos=134954, master\_connect\_retry=30,　 master\_heartbeat\_period=10;

2.7查看从节点的状态信息

show slave status\G;

2.8启动复制进程

start slave;

2.7再次查看从节点的状态信息

show slave status\G;

Slave\_IO\_Running: Yes //此状态必须YES

Slave\_SQL\_Running: Yes //此状态必须YES

2.8测试主从配置

### MYSQL主从形式

#### 1.一主一从

#### 2.主主复制

#### 3.一主多从

#### 4.多主一从

#### 5.级联复制

相关资料

<https://zhuanlan.zhihu.com/p/96212530>

<https://blog.csdn.net/zai_xia/article/details/90379016>

<https://www.cnblogs.com/xihuineng/p/10438969.html>

## 主从配置中遇到问题

### 1.Mysql配置错误

1.没有正确的挂载Mysql配置文件；

### 2.Authentication plugin 'caching\_sha2\_password'

在做MYSQL主从配置时,从库上执行  show slave [status](https://so.csdn.net/so/search?q=status&spm=1001.2101.3001.7020)\G 命令提示:

Last\_IO\_Errno: 2061

Last\_IO\_Error: error connecting to master 'ben@1.116.226.147:3311' - retry-time: 30 retries: 1 message: Authentication plugin 'caching\_sha2\_password' reported error: Authentication requires secure connection.

Why:

**用户配置有问题**

**解决方案：**

1. 去主库mysql上查询当前用户信息

SELECT plugin FROM `user` where user ='ben';

发现主库ben的plugin是caching\_sha2\_password 导致连接不上，修改为mysql\_native\_password即可解决：

2.Use mysql；

ALTER USER 'ben'@'%' IDENTIFIED WITH mysql\_native\_password BY

'ben@958958';

1. 修改后，重新配置从库binlog位置后重启

**stop slave;**

change master to

master\_host='1.116.226.147',

master\_user='ben',

master\_password='ben@958958',

master\_port=3311,

master\_log\_file='mysql-bin.000002',

master\_log\_pos=443,

master\_connect\_retry=30,

master\_heartbeat\_period=10;

**start slave;**

**解决方案：**[**https://blog.csdn.net/shangguoli/article/details/118143737**](https://blog.csdn.net/shangguoli/article/details/118143737)

**1.1解决1的时候修改密码修改错误**

**Last\_IO\_Error: error connecting to master 'ben@1.116.226.147:3311' - retry-time: 30 retries: 2 message: Access denied for user 'ben'@'172.17.0.1' (using password: YES)**

### 3.Error\_code: 1032; handler error HA\_ERR\_KEY\_NOT\_FOUND;

**MYSQL主从配置时，**从库上执行  show slave [status](https://so.csdn.net/so/search?q=status&spm=1001.2101.3001.7020)\G 命令提示：

Error\_code: 1032; handler error HA\_ERR\_KEY\_NOT\_FOUND;

Why:

主从数据库数据不一致,导致同步操作在从库上无法执行;

解决方案：

1. 保证主从数据一致；
2. Stop slave;
3. 重新配置访问主库的参数信息
4. Start slave;

相关资料:

查看具体日志

select \* from performance\_schema.replication\_applier\_status\_by\_worker\G;

1. https://blog.csdn.net/eagle89/article/details/106412402

## 主主复制

在主从复制的基础上做；

### 配置主服务器

在主服务器中增加

#步进值auto\_imcrement。一般有n台主MySQL就填n

auto\_increment\_increment=2

#起始值。一般填第n台主MySQL。此时为第一台主MySQL

auto\_increment\_offset=1

### 配置从服务器

#步进值auto\_imcrement。一般有n台主MySQL就填n

auto\_increment\_increment=2

#起始值。一般填第n台主MySQL。

auto\_increment\_offset=2

创建账号

CREATE USER 'ben'@'%' IDENTIFIED BY 'ben@958958';

GRANT REPLICATION SLAVE, REPLICATION CLIENT ON \*.\* TO 'ben'@'%';

# Oracle数据备份

## 相关资料

https://www.cnblogs.com/klb561/p/10828935.html