一、Mysql

> 纸上得来终觉浅，绝知此事要躬行。

一、前言

MySQL是一个**关系型数据库管理系统**，由瑞典MySQL AB 公司开发，目前属于 Oracle 旗下产品。MySQL 是最流行的关系型数据库管理系统之一，在 WEB 应用方面，MySQL是最好的 RDBMS (Relational Database Management System，关系数据库管理系统) 应用软件之一。关系数据库将数据保存在不同的表中，而不是将所有数据放在一个大仓库内，这样就增加了速度并提高了灵活性。

MySQL所使用的 SQL 语言是用于访问数据库的最常用标准化语言。MySQL 软件采用了双授权政策，分为社区版和商业版，由于其体积小、速度快、总体拥有成本低，尤其是开放源码这一特点，一般中小型网站的开发都选择 MySQL 作为网站数据库。

RDBMS即关系数据库管理系统(Relational Database Management System)

关系型数据库:

> SQLServer、Oracle、mysql等。

什么是关系型数据库?

关系型数据库是依据关系模型来创建的数据库。

所谓关系模型就是"一对一、一对多、多对多"等关系模型，关系模型就是指二维表格模型,因而一个关系型数据

库就是由二维表及其之间的联系组成的一个数据组织。

关系型数据可以很好地存储一些关系模型的数据，比如一个老师对应多个学生的数据（"一对多"），一本书对应

多个作者（"一对多"），一本书对应一个出版日期（"一对一"）

数据的存储形式：

id	grade_name
1	python
2	linux
3	AI
4	Java

非关系型数据库redis,mongodb,memcached

```
1.以键值对的方式存储数据---（Key-Value）的形式

2.缓存数据库
```

什么是非关系型数据库?

非关系型数据库主要是基于“非关系模型”的数据库（由于关系型太大，所以一般用“非关系型”来表示其他类型的

数据库）。

关系型数据库与非关系型数据库的区别:

```
1.关系型数据库：
优点：
1、易于维护：都是使用表结构，格式一致；
2、使用方便：SQL语言通用，可用于复杂查询；
3、复杂操作：支持SQL，可用于一个表以及多个表之间非常复杂的查询。
缺点：
1、读写性能比较差，尤其是海量数据的高效率读写；
2、固定的表结构，灵活度稍欠；
3、高并发读写需求，传统关系型数据库来说，硬盘I/O是一个很大的瓶颈。
==========================================================================
2.非关系型数据库严格上不是一种数据库，应该是一种数据结构化存储方法的集合，可以是文档或者键值对等。
优点：
1、格式灵活：存储数据的格式可以是key,value形式、文档形式、图片形式等等，文档形式、图片形式等等，使用灵活，应用场景广泛，而关系型数据库则只支持基础类型。
2、速度快：nosql可以使用硬盘或者随机存储器作为载体，而关系型数据库只能使用硬盘；
3、高扩展性；
4、成本低：nosql数据库部署简单，基本都是开源软件。

缺点：
1、不提供sql支持，学习和使用成本较高；
2、无事务处理；
3、数据结构相对复杂，复杂查询方面稍欠。
```

MySQL的官方网址： http://www.mysql.com/ ，MySQL的社区版本下载地址为： http://dev.mysql.com/downloads/mysql/ ，在写本文时，当前的MySQL最新版本是：8.0 。

什么是sql?

SQL代表结构化查询语言(Structured Query Language)。SQL是用于访问数据库的标准化语言。

SQL包含三个部分：

数据定义语言包含定义数据库及其对象的语句，例如表，视图，触发器，存储过程等。

数据操作语言包含允许您更新和查询数据的语句。

数据控制语言允许授予用户权限访问数据库中特定数据的权限。

二、mysql安装

关闭防火墙和selinux

1、编译安装mysql5.7

1、清理安装环境：

```
# yum erase mariadb mariadb-server mariadb-libs mariadb-devel -y
# userdel -r mysql
# rm -rf /etc/my*
# rm -rf /var/lib/mysql
```

2、创建mysql用户

```
[root@mysql-server ~]# useradd -r mysql -M -s /bin/false
```

3、从官网下载tar包

wget https://dev.mysql.com/get/Downloads/MySQL-5.7/mysql-boost-5.7.27.tar.gz

4、安装编译工具

```
# yum -y install ncurses ncurses-devel openssl-devel bison gcc gcc-c++ make
cmake:
# yum -y install cmake
```

5、创建mysql目录

```
[root@mysql-server ~]# mkdir -p /usr/local/{data,mysql,log}
```

6、解压

```
[root@mysql-server ~]# tar xzvf mysql-boost-5.7.27.tar.gz -C /usr/local/
注:如果安装的MySQL5.7及以上的版本，在编译安装之前需要安装boost,因为高版本mysql需要boots库的
安装才可以正常运行。否则会报CMake Error at cmake/boost.cmake:81错误
安装包里面自带boost包
```

7、编译安装

```
cd 解压的mysql目录
[root@mysql-server ~]# cd /usr/local/mysql-5.7.27/
[root@mysql-server mysql-5.7.27]#cmake . \
-DWITH_BOOST=boost/boost_1_59_0/ \
-DCMAKE_INSTALL_PREFIX=/usr/local/mysql \
-DSYSCONFDIR=/etc \
-DMYSQL_DATADIR=/usr/local/mysql/data \
-DINSTALL_MANDIR=/usr/share/man \
-DMYSQL_TCP_PORT=3306 \
-DMYSQL_UNIX_ADDR=/tmp/mysql.sock \
-DDEFAULT_CHARSET=utf8 \
-DEXTRA_CHARSETS=all \
-DDEFAULT_COLLATION=utf8_general_ci \
-DWITH_READLINE=1 \
-DWITH_SSL=system \
-DWITH_EMBEDDED_SERVER=1 \
-DENABLED_LOCAL_INFILE=1 \
-DWITH_INNOBASE_STORAGE_ENGINE=1

提示：boost也可以使用如下指令自动下载，如果不下载bost压缩包，把下面的这一条添加到配置中第二行
-DDOWNLOAD_BOOST=1/
参数详解:
```

```
-DCMAKE_INSTALL_PREFIX=/usr/local/mysql \    安装目录
-DSYSCONFDIR=/etc \    配置文件存放 （默认可以不安装配置文件）
-DMYSQL_DATADIR=/usr/local/mysql/data \    数据目录    错误日志文件也会在这个目录
-DINSTALL_MANDIR=/usr/share/man \      帮助文档
-DMYSQL_TCP_PORT=3306 \      默认端口
-DMYSQL_UNIX_ADDR=/tmp/mysql.sock \  sock文件位置，用来做网络通信的，客户端连接服务器的时候用
-DDEFAULT_CHARSET=utf8 \     默认字符集。字符集的支持，可以调
-DEXTRA_CHARSETS=all \   扩展的字符集支持所有的
-DDEFAULT_COLLATION=utf8_general_ci \  支持的
-DWITH_READLINE=1 \     上下翻历史命令
-DWITH_SSL=system \     使用私钥和证书登陆（公钥）  可以加密。 适用与长连接。坏处：速度慢
-DWITH_EMBEDDED_SERVER=1 \    嵌入式数据库
-DENABLED_LOCAL_INFILE=1 \     从本地倒入数据，不是备份和恢复。
-DWITH_INNOBASE_STORAGE_ENGINE=1   默认的存储引擎，支持外键
```

```
CMake Warning:
  Manually-specified variables were not used by the project:

    WITH_READLINE


-- Build files have been written to: /usr/local/mysql-5.7.27
```

```
[root@mysql-server mysql-5.7.27]# make && make install
如果安装出错，想重新安装：
    不用重新解压，只需要删除安装目录中的缓存文件CMakeCache.txt
```

```
[  0%] Building C object zlib/CMakeFiles/zlib.dir/crc32.o
[  0%] Building C object zlib/CMakeFiles/zlib.dir/deflate.o
[  0%] Building C object zlib/CMakeFiles/zlib.dir/gzclose.o
[  0%] Building C object zlib/CMakeFiles/zlib.dir/gzlib.o
[  0%] Building C object zlib/CMakeFiles/zlib.dir/gzread.o
[  0%] Building C object zlib/CMakeFiles/zlib.dir/gzwrite.o
[  0%] Building C object zlib/CMakeFiles/zlib.dir/inflate.o
[  0%] Building C object zlib/CMakeFiles/zlib.dir/infback.o
[  0%] Building C object zlib/CMakeFiles/zlib.dir/inftrees.o
[  0%] Building C object zlib/CMakeFiles/zlib.dir/inffast.o
[  0%] Building C object zlib/CMakeFiles/zlib.dir/trees.o
[  0%] Building C object zlib/CMakeFiles/zlib.dir/uncompr.o
```

需要很长时间！大约半小时

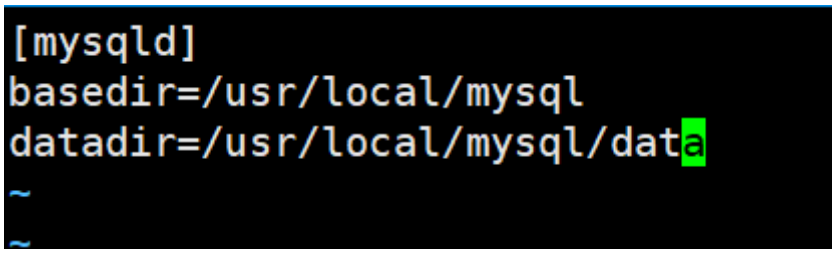
8、初始化

```
[root@mysql-server mysql-5.7.27]# cd /usr/local/mysql
[root@mysql-server mysql]# chown -R mysql.mysql .
[root@mysql-server mysql]# ./bin/mysqld --initialize --user=mysql --basedir=/usr/local/mysql --datadir=/usr/local/mysql/data      ---初始化完成之后，一定要记住提示最后的密码用于登陆或者修改密码
```

```
[root@mysql-server mysql]# ./bin/mysqld --initialize --user=mysql --basedir=/usr/local/mysql --data
dir=/usr/local/mysql/data
2019-08-18T21:03:57.661984Z 0 [Warning] TIMESTAMP with implicit DEFAULT value is deprecated. Please
 use --explicit_defaults_for_timestamp server option (see documentation for more details).
2019-08-18T21:03:58.080506Z 0 [Warning] InnoDB: New log files created, LSN=45790
2019-08-18T21:03:58.145511Z 0 [Warning] InnoDB: Creating foreign key constraint system tables.
2019-08-18T21:03:58.210021Z 0 [Warning] No existing UUID has been found, so we assume that this is
the first time that this server has been started. Generating a new UUID: b2323e6c-c1fb-11e9-bff5-00
0c29bbec75.
2019-08-18T21:03:58.214204Z 0 [Warning] Gtid table is not ready to be used. Table 'mysql.gtid_execu
ted' cannot be opened.
2019-08-18T21:03:58.538868Z 0 [Warning] CA certificate ca.pem is self signed.
2019-08-18T21:03:58.631539Z 1 [Note] A temporary password is generated for root@localhost: GP9TKGgY
9i/8
```

初始化,只需要初始化一次

```
[root@mysql-server ~]# vim /etc/my.cnf     ---添加如下内容
[mysqld]
basedir=/usr/local/mysql      #指定安装目录
datadir=/usr/local/mysql/data   #指定数据存放目录
```



9、启动mysql

```
[root@mysql-server ~]# cd /usr/local/mysql
[root@mysql-server mysql]# ./bin/mysqld_safe --user=mysql &
```

```
[root@mysql-server mysql]# netstat -lntp | grep 3306
tcp6       0      0 :::3306                 :::*                    LISTEN      31005/mysqld
```

10、登录mysql

```
[root@mysql-server mysql]# /usr/local/mysql/bin/mysql -uroot -p'GP9TKGgY9i/8'
mysql: [Warning] Using a password on the command line interface can be insecure.
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 2
Server version: 5.7.27

Copyright (c) 2000, 2019, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> exit
```

11、修改密码

```
[root@mysql-server mysql]# /usr/local/mysql/bin/mysqladmin -u root -
p'GP9TKGgY9i/8'  password 'QianFeng@123'
mysqladmin: [Warning] Using a password on the command line interface can be
insecure.
Warning: Since password will be sent to server in plain text, use ssl connection
to ensure password safety.
```

12、添加环境变量

```
[root@mysql-server mysql]# vim /etc/profile    ---添加如下
PATH=$PATH:$HOME/bin:/usr/local/mysql/bin
[root@mysql-server mysql]# source /etc/profile
之后就可以在任何地方使用mysql命令登陆Mysql服务器:
[root@mysql-server mysql]# mysql -uroot -p'QianFeng@123'
mysql: [Warning] Using a password on the command line interface can be insecure.
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 5
Server version: 5.7.27 Source distribution

Copyright (c) 2000, 2019, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

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

mysql>exit
```

13、配置mysqld服务的管理工具:

```
[root@mysql-server mysql]# cd /usr/local/mysql/support-files/
[root@mysql-server support-files]# cp mysql.server /etc/init.d/mysqld
[root@mysql-server support-files]# chkconfig --add mysqld
[root@mysql-server support-files]# chkconfig mysqld on
先将原来的进程杀掉
[root@mysql-server ~]# /etc/init.d/mysqld start
Starting MySQL. SUCCESS!
[root@mysql-server ~]# netstat -lntp
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
PID/Program name
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN
 1087/sshd
```

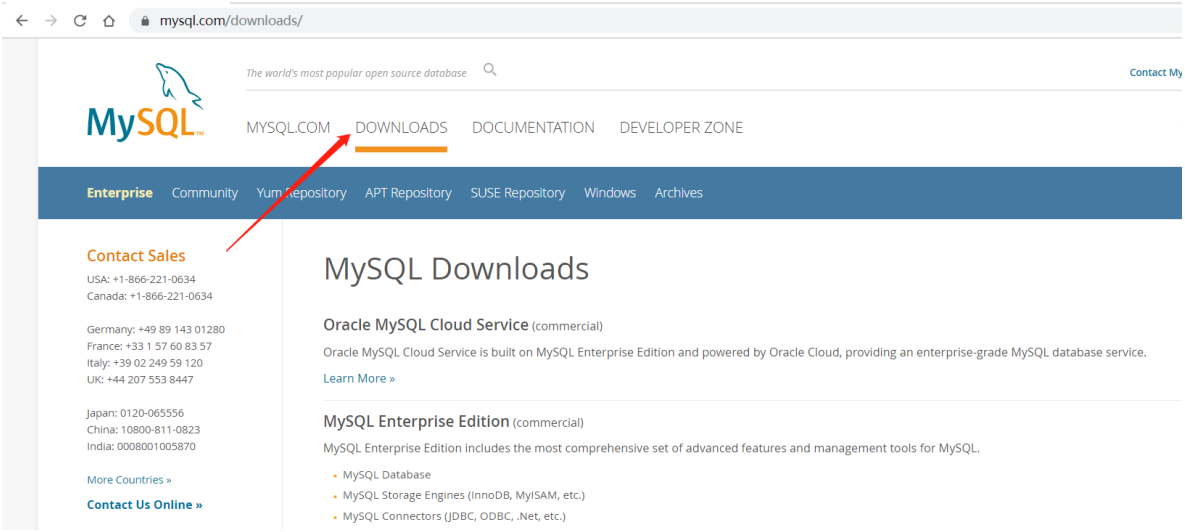
```
tcp6       0      0 :::22                   :::*                    LISTEN
 1087/sshd
tcp6       0      0 :::3306                 :::*                    LISTEN
 31249/mysqld
[root@mysql-server ~]# /etc/init.d/mysqld stop
```

数据库编译安装完成.

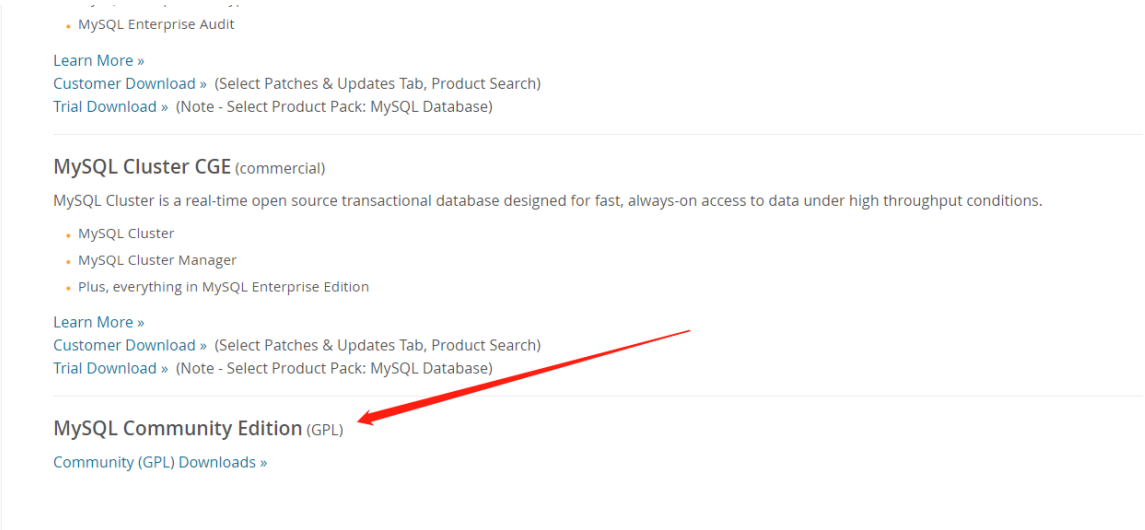
2、yum安装方式

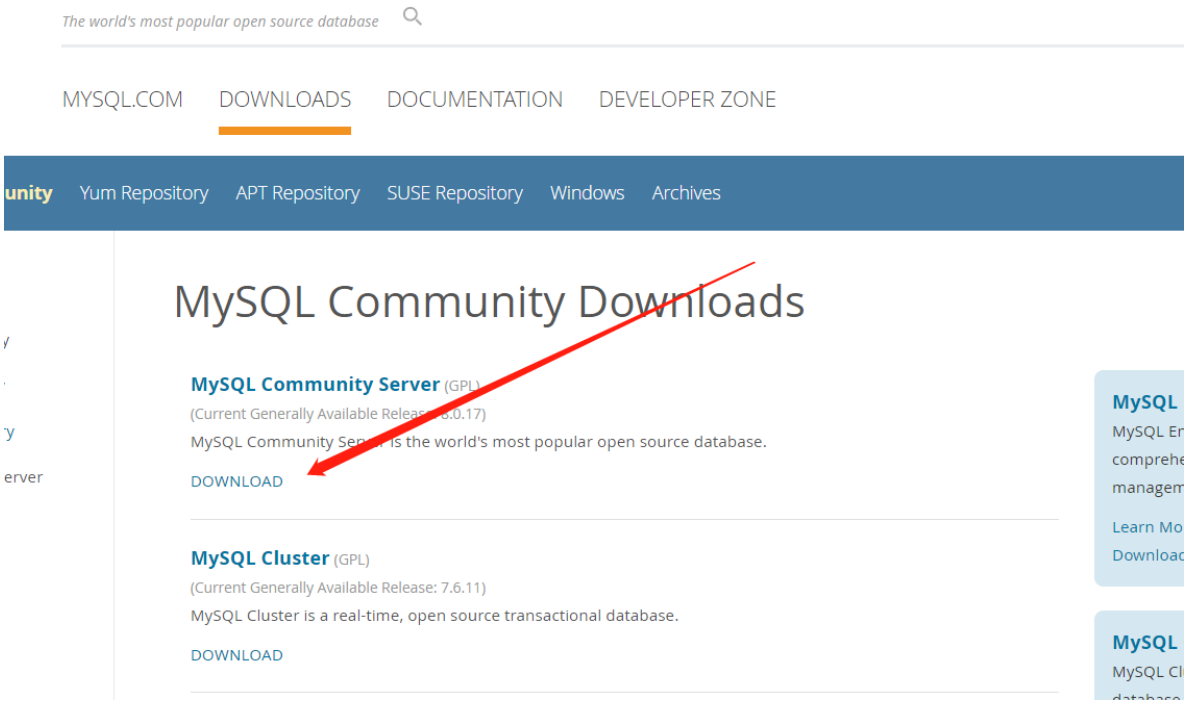
关闭防火墙和selinux

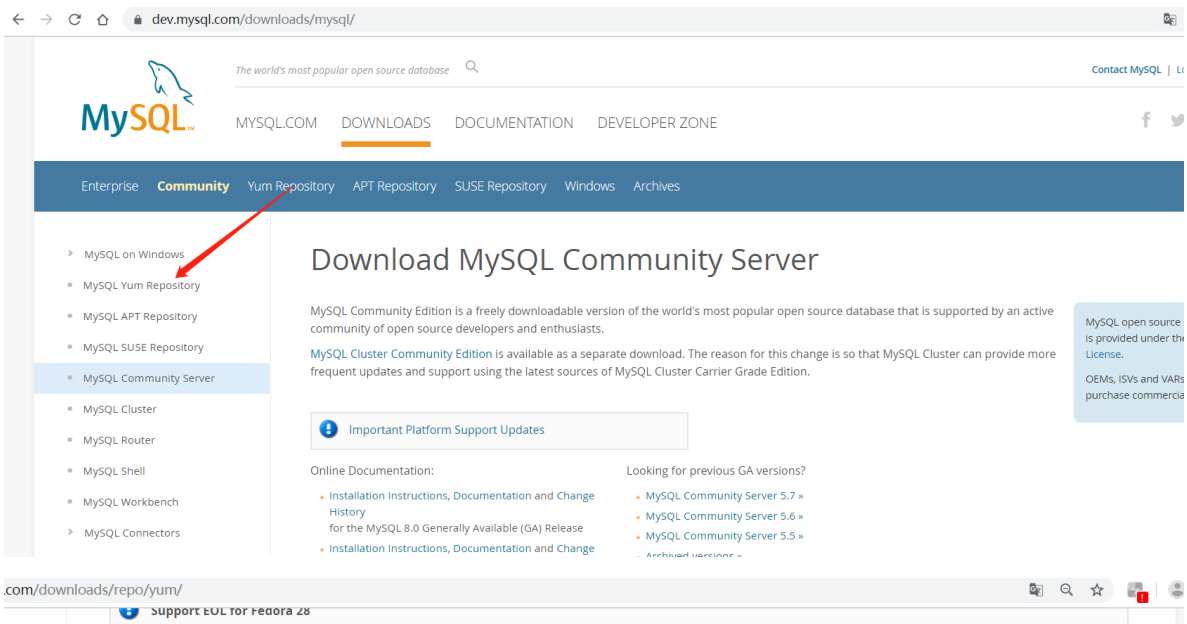
mysql的官方网站：www.mysql.com

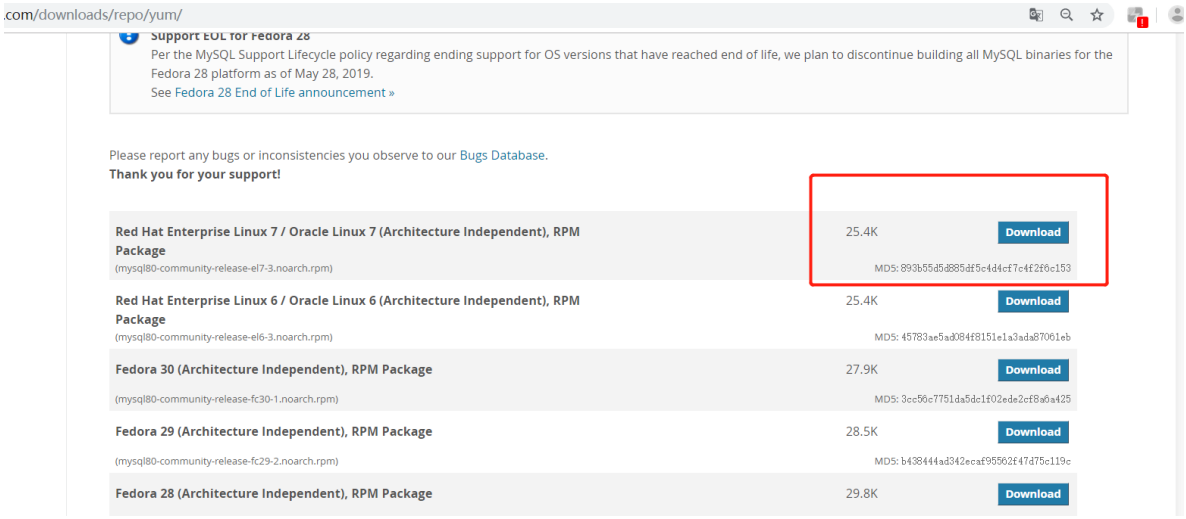


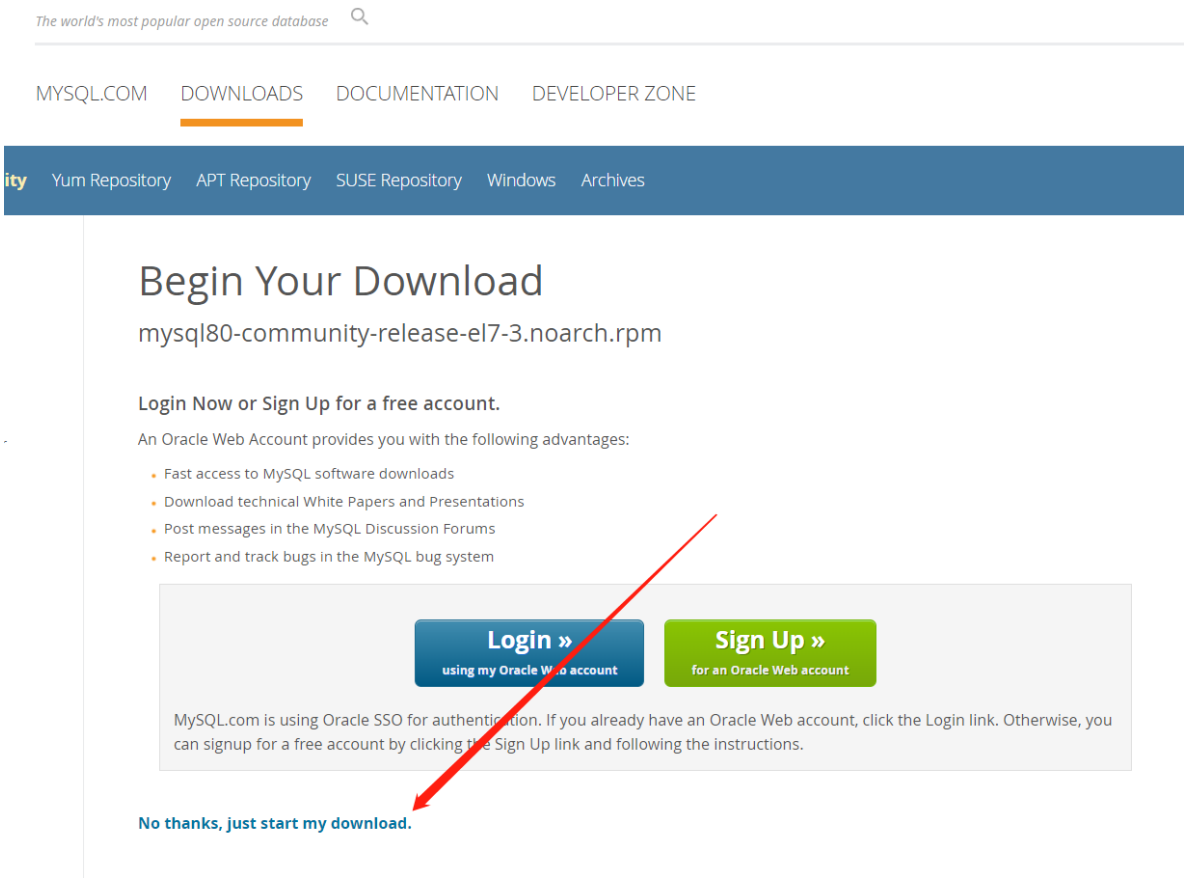
拉到底




The world's most popular open source database
MYSQL.COM
DOWNLOADS
DOCUMENTATION
DEVELOPER ZONE
unity
Yum Repository
APT Repository
SUSE Repository
Windows
Archives
MySQL Community Downloads
MySQL Community Server (GPL)
(Current Generally Available Release: 8.0.17)
MySQL Community Server is the world's most popular open source database.
DOWNLOAD
MySQL Cluster (GPL)
(Current Generally Available Release: 7.6.11)
MySQL Cluster is a real-time, open source transactional database.
DOWNLOAD


dev.mysql.com/downloads/mysql/
The world's most popular open source database
Contact MySQL
MySQL
MYSQL.COM
DOWNLOADS
DOCUMENTATION
DEVELOPER ZONE
Enterprise
Community
Yum Repository
APT Repository
SUSE Repository
Windows
Archives
MySQL on Windows
MySQL Yum Repository
MySQL APT Repository
MySQL SUSE Repository
MySQL Community Server
MySQL Cluster
MySQL Router
MySQL Shell
MySQL Workbench
MySQL Connectors
Download MySQL Community Server
MySQL Community Edition is a freely downloadable version of the world's most popular open source database that is supported by an active community of open source developers and enthusiasts.
MySQL Cluster Community Edition is available as a separate download. The reason for this change is so that MySQL Cluster can provide more frequent updates and support using the latest sources of MySQL Cluster Carrier Grade Edition.
Important Platform Support Updates
Online Documentation:
Installation Instructions, Documentation and Change History for the MySQL 8.0 Generally Available (GA) Release
Looking for previous GA versions?
MySQL Community Server 5.7 »
MySQL Community Server 5.6 »
MySQL Community Server 5.5 »


.com/downloads/repo/yum/
Support EOL for Fedora 28
Per the MySQL Support Lifecycle policy regarding ending support for OS versions that have reached end of life, we plan to discontinue building all MySQL binaries for the Fedora 28 platform as of May 28, 2019.
See Fedora 28 End of Life announcement »
Please report any bugs or inconsistencies you observe to our Bugs Database.
Thank you for your support!
Red Hat Enterprise Linux 7 / Oracle Linux 7 (Architecture Independent), RPM Package
(mysql80-community-release-el7-3.noarch.rpm)
25.4K
Download
MD5: 893b55d5d885df5c4d4cf7c4f2f6c153
Red Hat Enterprise Linux 6 / Oracle Linux 6 (Architecture Independent), RPM Package
(mysql80-community-release-el6-3.noarch.rpm)
25.4K
Download
MD5: 45783ae5ad084f8151e1a3ada87061eb
Fedora 30 (Architecture Independent), RPM Package
(mysql80-community-release-fc30-1.noarch.rpm)
27.9K
Download
MD5: 3cc56c7751da5dc1f02ede2cf8a6a425
Fedora 29 (Architecture Independent), RPM Package
(mysql80-community-release-fc29-2.noarch.rpm)
28.5K
Download
MD5: b438444ad342ecaf95562f47d75c119c
Fedora 28 (Architecture Independent), RPM Package
29.8K
Download



下载

```
[root@mysql-server ~]# wget https://dev.mysql.com/get/mysql80-community-release-el7-3.noarch.rpm
```

或者下载到本地上传到服务器

2.安装mysql的yum仓库

```
[root@mysql-server ~]# rpm -ivh mysql80-community-release-el7-3.noarch.rpm
[root@mysql-server ~]# yum -y install yum-utils    #安装yum工具包
```

3、配置yum源

```
[root@mysql-server ~]# vim /etc/yum.repos.d/mysql-community.repo    #修改如下
```

```
# Enable to use MySQL 5.6
[mysql56-community]
name=MySQL 5.6 Community Server
baseurl=http://repo.mysql.com/yum/mysql-5.6-community/el/7/$basearch/
enabled=0
gpgcheck=1
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-mysql

# Enable to use MySQL 5.7
[mysql57-community]
name=MySQL 5.7 Community Server
baseurl=http://repo.mysql.com/yum/mysql-5.7-community/el/7/$basearch/
enabled=1
gpgcheck=1
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-mysql

[mysql80-community]
name=MySQL 8.0 Community Server
baseurl=http://repo.mysql.com/yum/mysql-8.0-community/el/7/$basearch/
enabled=0
gpgcheck=1
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-mysql

[mysql-connectors-community]
name=MySQL Connectors Community
baseurl=http://repo.mysql.com/yum/mysql-connectors-community/el/7/$basearch/
enabled=1
```

将原来的0改为1

将原来的1改为0

1表示开启，0表示关闭

或者

```
# yum-config-manager --enable mysql57-community    将禁用的yum源库启用
# yum-config-manager --disable mysql80-community    将启用的yum源库禁用
```

4、安装数据库

```
[root@mysql-server ~]# yum install -y mysql-community-server
启动服务
[root@mysql-server ~]# systemctl start mysqld
设置开机启动
[root@mysql-server ~]# systemctl enable mysqld
```

5、查找密码

```
密码保存在日志文件中
[root@mysql-server ~]# grep password /var/log/mysqld.log
2019-08-18T14:03:51.991454Z 1 [Note] A temporary password is generated for
root@localhost: woHtkMgau9,w
```

6、修改密码

```
[root@mysql-server ~]# mysql -uroot -p'woHtkMgau9,w'   #登录
两种方式:
第一种:
mysql> alter user 'root'@'localhost' identified by 'QianFeng@123';
[root@mysql-server ~]# mysql -uroot -p'woHtkMgau9,w'
mysql: [Warning] Using a password on the command line interface can be insecure.
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 2
Server version: 5.7.27
```

```
....
mysql> alter user 'root'@'localhost' identified by 'QianFeng@123';
Query OK, 0 rows affected (0.01 sec)
mysql> exit
Bye
[root@mysql-server ~]# mysql -uroot -p'QianFeng@123'
mysql: [Warning] Using a password on the command line interface can be insecure.
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 3
Server version: 5.7.27 MySQL Community Server (GPL)
...
mysql> exit
Bye
```

第二种：

```
# mysqladmin -u root -p'旧密码' password '新密码'
```

注：修改密码必须大小写数字和特殊符号都有。

扩展

通过配置文件设置密码强度

```
[root@mysql-server ~]# vim /etc/my.cnf   #在最后添加如下内容
validate_password=off
[root@mysql-server ~]# systemctl restart mysqld   #重启mysql生效
```

可以用第二种方式修改为简单的密码：

```
[root@mysql-server ~]# mysqladmin -uroot -p'QianFeng@123' password 'qf123'
mysqladmin: [Warning] Using a password on the command line interface can be
insecure.
Warning: Since password will be sent to server in plain text, use ssl connection
to ensure password safety.
[root@mysql-server ~]# mysql -uroot -pqf123
mysql: [Warning] Using a password on the command line interface can be insecure.
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 3
Server version: 5.7.27 MySQL Community Server (GPL)

Copyright (c) 2000, 2019, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> exit
Bye
```

编译安装：

```
# ls
COPYING       README       bin   include  mysql-test  support-files
COPYING-test  README-test  docs  lib      share
```

1、bin目录

用于放置一些可执行文件，如mysql、mysqld、mysqlbinlog等。

2、include目录

用于放置一些头文件，如：mysql.h、mysql_ername.h等。

3、lib目录

用于放置一系列库文件。
4、share目录
用于存放字符集、语言等信息。

yum安装：
/var/lib/mysql　存放数据文件
/usr/share/mysql　用于存放字符集、语言等信息。

二、数据库基本操作

一、数据库存储引擎（扩展）

数据库存储引擎是数据库底层软件组织，数据库管理系统（DBMS）使用数据引擎进行创建、查询、更新和删除数据。不同的存储引擎提供不同的存储机制、索引、锁定等功能，使用不同的存储引擎，还可以 获得特定的功能。现在许多不同的数据库管理系统都支持多种不同的数据引擎。MySQL的核心就是存储引擎。

MySQL存储引擎介绍
文件系统： 操作系统组织和存取数据的一种机制。文件系统是一种软件。
类型：ext2 3 4 ，xfs 数据
不管使用什么文件系统，数据内容不会变化 不同的是，存储空间、大小、速度。
MySQL引擎： 可以理解为，MySQL的“文件系统”，只不过功能更加强大。
MySQL引擎功能： 除了可以提供基本的存取功能，还有更多功能事务功能、锁定、备份和恢复、优化以及特殊功能。和磁盘打交道，mysql中 组织。

1、InnoDB存储引擎:默认引擎，最常用的。

InnoDB是事务型数据库的首选引擎，支持事务安全表（ACID），支持行锁定和外键;InnoDB是默认的MySQL引擎 InnoDB特点： 支持事务处理，支持外键，支持崩溃修复和并发控制。如果需要对事务的完整性要求比较高（比如银行），要求实现并发控制（比如售票），那选择InnoDB有很大的优势。如果需要频繁的更新、删除操作的数据库，也可以选择InnoDB，因为支持事务的提交（commit）和回滚（rollback）。

2、MyISAM存储引擎：（了解） MyISAM基于ISAM存储引擎，并对其进行扩展。它是在Web、数据仓储和其他应用环境下最常使用的存储引擎之一。MyISAM拥有较高的插入、查询速度，但不支持事务。MyISAM特点： 插入数据快，空间和内存使用比较低。如果表主要是用于插入新记录和读出记录，那么选择MyISAM能实现处理高效率。如果应用的完整性、并发性要求比较低，也可以使用。

3、MEMORY内存型引擎(了解)

MEMORY存储引擎将表中的数据存储到内存中，未查询和引用其他表数据提供快速访问

MEMORY特点： 所有的数据都在内存中，数据的处理速度快，但是安全性不高。如果需要很快的读写速度，对数据的安全性要求较低，可以选择MEMOEY。它对表的大小有要求，不能建立太大的表。所以，这类数据库只使用在相对较小的数据库表。

4、Archive(归档引擎)

如何选择引擎：如果要提供提交、回滚、并要求实现并发控制，InnoDB是一个好的选择；如果数据表主要用来插入和查询记录，则MyISAM引擎能提供较高的处理效率；如果只是临时存放数据，数据量不大，并且不需要较高的数据安全性，可以选择将数据保存在内存中的Memory引擎；MySQL中使用该引擎作为临时表，存放查询的中间结果；如果只有INSERT和SELECT操作，可以选择Archive，Archive支持高并发的插入操作，但是本身不是事务安全的。Archive非常适合存储归档数据，如记录日志信息可以使用Archive。

使用哪一种引擎需要灵活选择，一个数据库中多个表可以使用不同引擎以满足各种性能和实际需求，使用合适的存储引擎，将会提高整个数据库的性能。

存储引擎查看：

```
mysql> show engines;
```

```
mysql> mysql> show engines;
+--------------------+---------+----------------------------------------------------------------+--------------+------+------------+
| Engine             | Support | Comment                                                        | Transactions | XA   | Savepoints |
+--------------------+---------+----------------------------------------------------------------+--------------+------+------------+
| InnoDB             | DEFAULT | Supports transactions, row-level locking, and foreign keys     | YES          | YES  | YES        |
| MRG_MYISAM         | YES     | Collection of identical MyISAM tables                          | NO           | NO   | NO         |
| MEMORY             | YES     | Hash based, stored in memory, useful for temporary tables      | NO           | NO   | NO         |
| BLACKHOLE          | YES     | /dev/null storage engine (anything you write to it disappears) | NO           | NO   | NO         |
| MyISAM             | YES     | MyISAM storage engine                                          | NO           | NO   | NO         |
| CSV                | YES     | CSV storage engine                                             | NO           | NO   | NO         |
| ARCHIVE            | YES     | Archive storage engine                                         | NO           | NO   | NO         |
| PERFORMANCE_SCHEMA | YES     | Performance Schema                                             | NO           | NO   | NO         |
| FEDERATED          | NO      | Federated MySQL storage engine                                 | NULL         | NULL | NULL       |
+--------------------+---------+----------------------------------------------------------------+--------------+------+------------+
9 rows in set (0.00 sec)
```

修改搜索引擎

```
ALTER TABLE 表名 ENGINE=引擎;
```

看你的mysql当前默认的存储引擎:

```
mysql> show variables like '%storage_engine%';
```

如何查看Mysql服务器上的版本

```
mysql> select version();
```

创建时候指定引擎

```
mysql> create table t1(id int,manager char(10)) engine =innodb;
```

了解:

1.什么是外键：外键的主要作用是保持数据的一致性、完整性。

2.什么是索引:索引相当于书中的目录，可以提高数据检索的效率，降低数据库的IO。MySQL在300万条记录左右性能开始逐渐下降，虽然官方文档说500~800w记录，所以大数据量建立索引是非常有必要的

3.什么是事务：事务是由一步或几步数据库操作这系列操作要么全部执行，要么全部放弃执行。程序和事务是两个不同的概念。

事务具有**四个特性**：**原子性**（Atomicity）、**一致性**（Consistency）、**隔离性**（Isolation）和**持续性**（Durability）。这四个特性也简称**ACID**性。

（1）**原子性**：事务是应用中最小的执行单位，就如原子是自然界最小颗粒，具有不可再分的特征一样。事务是应用中不可再分的最小执行体。（**最小了，不可再分了**）

（2）**一致性**：事务执行的结果，必须使数据库从一个一致性状态，变到另一个一致性状态。当数据库中只包含事务成功提交的结果时，数据库处于一致性状态。一致性是通过原子性来保证的。（**说罢了就是白狗变成了黑狗，不能出现斑点狗！**）

（3）**隔离性**：各个事务的执行互不干扰，任意一个事务的内部操作对其他并发的事务，都是隔离的。也就是说：并发执行的事务之间不能看到对方的中间状态，并发执行的事务之间不能相互影响。（**说白了，就是你做你的，我做我的！**）

（4）**持续性**：持续性也称为持久性，指事务一旦提交，对数据所做的任何改变，都要记录到永久存储器中，通常是**保存进物理数据库。（说白了就是一条道跑到黑）**

4.什么是行锁定与锁表：可以将一张表锁定和可以单独锁一行的记录。为了防止你在操作的同时也有别人在操作。

二、sql语句

增删改查

```
SQL（Structured Query Language 即结构化查询语言）
SQL语言主要用于存取数据、查询数据、更新数据和管理关系数据库系统，SQL语言由IBM开发。
DDL语句        数据库定义语言：  数据库、表、视图、索引、存储过程，例如CREATE DROP ALTER**
DML语句    数据库操纵语言（对记录的操作）：  插入数据INSERT、删除数据DELETE、更新数据UPDATE**
DCL语句        数据库控制语言（和权限有关）：  例如控制用户的访问权限GRANT、REVOKE**
DQL语句    数据库查询语言：  查询数据SELECT**
```

程序连接数据库的文件:

```
A. ODBC --------- PHP<.php>
B. JDBC ----------- JAVA <.jsp>
```

==========

库----相当于一个目录，存放数据的

```
库里面存放的表，  相当于是文件。
每一行叫做记录，除第一行。
每一列叫一个字段。列上面的第一个叫字段名称。
创建一个库：---->查看库--->进入这个库----->创建表----->查看表：查看表名，表的字段（表结构），表里面的内容（表记录），查看表的状态----->修改表：添加字段，删除字段，修改字段----->修改记录（更新记录），添加记录，删除记录。各种查询，删除表，删除库。
```

```
1.创建库
mysql> create database 库名;
2.查看数据库
mysql> show databases;
3.进入数据库
mysql> use 库名
4.查看当前所在的库
mysql> select database();
```

创建表

```
语法:
create table 表名(
        字段名1  类型[(宽度) 约束条件],
        字段名2  类型[(宽度) 约束条件],
        字段名3  类型[(宽度) 约束条件]
)[存储引擎 字符集];
==在同一张表中，字段名是不能相同
==宽度和约束条件可选
==字段名和类型是必须的
==========================================================
1.创建表:
```

```
创建表 create table t1(id int,name varchar(20),age int);
                        字段 类型 字段 类型(长度), 字段 类型
mysql> create table t1(id int,name varchar(50),sex enum('m','f'),age int);
2.查看有哪些表
mysql> show tables;
3.查看表结构:
mysql> desc t1;
4.查看表里面的所有记录:
语法: select 内容 from 表名;
mysql> select * from t1;
*:代表所有内容
5.查看表里面的指定字段:
语法:select 字段, 字段 from 表名;
mysql> select name,sex from t1;
6.查看表的状态
mysql> show table status like '表名'\G     ---每条SQL语句会以分号结尾，想看的清楚一些以\G结尾，一条记录一条记录显示。（把表90度向左反转，第一列显示字段，第二列显示记录）使用的\G就不用添加分号了
7.修改表名称
方式一、语法:rename table 旧表名 to 新表名;
mysql> rename table t1 to t2;
Query OK, 0 rows affected (0.00 sec)
方式二、语法:alter table 旧表名 rename 新表名;
mysql> alter table t2 rename t3;
8.使用edit(\e)编辑------了解
mysql> \e  #可以写新的语句，调用的vim编辑器，在里面结尾的时候不加分号，保存退出之后在加“;”
    -> ;
9.删除表
mysql> drop table 表名;
10.删除库
mysql> drop database 库名;
```

三、数据类型

1、数据类型

在MySQL数据库管理系统中，可以通过存储引擎来决定表的类型。同时，MySQL数据库管理系统也 提供了数据类型决定表存储数据的类型。

```
1.整型
作用：用于存储用户的年龄、游戏的Level、经验值等。
分类：tinyint  smallint  mediumint  int  bigint
常用的是int
显示宽度：类型后面小括号内的数字是显示宽度，不能限制插入数值的大小
比如：bigint(2)     2是显示宽度
结论：整形的宽度仅为显示宽度，不是限制。因此建议整形无须指定宽度。
=====================================================
2.浮点数类型        FLOAT DOUBLE
作用：用于存储用户的身高、体重、薪水等
float(5.3)       #一共5位，小数占3位.做了限制
mysql> create table test4(float_test float(5,2));  #案例
宽度不算小数点
=========================================================================
==
定点数类型          DEC
```

```
定点数在MySQL内部以字符串形式存储，比浮点数更精确，适合用来表示货币等精度高的数据。
3.字符串类型
作用：用于存储用户的姓名、爱好、发布的文章等
字符类型 char varchar  --存字符串
char(10)           根据10，占10个.
    列的长度固定为创建表时声明的长度： 0 ~ 255
varchar(10)      根据实际字符串长度占空间，最多10个
    列中的值为可变长字符串，长度： 0 ~ 65535
案例:
mysql> create table t8(c char(5),v varchar(12));
Query OK, 0 rows affected (0.42 sec)

mysql> insert into t8 values('abcde','abcdef');
Query OK, 1 row affected (0.38 sec)

mysql> insert into t8 values('abc','abcdef');  #char可以少于规定长度。
Query OK, 1 row affected (0.05 sec)

mysql> insert into t8 values('abc777','abcdef7');  #char不能大于规定的长度。
ERROR 1406 (22001): Data too long for column 'c' at row 1
mysql>
=====================================================================
1.经常变化的字段用varchar
2.知道固定长度的用char
3.尽量用varchar
4.超过255字符的只能用varchar或者text
5.能用varchar的地方不用text
text：文本格式
---------------------------------------------------------------------
4.枚举类型 enum
mysql> create table t101(name enum('tom','jim'));
只能从tom,jim两个里面2选其1
（enumeration）
 有限制的时候用枚举
=====================================================================
5.日期类型
===时间和日期类型测试：year、date、time、datetime、timestamp
作用：用于存储用户的注册时间，文章的发布时间，文章的更新时间，员工的入职时间等
注意事项:
==插入年份时，尽量使用4位值
==插入两位年份时，<=69，以20开头，比如65，  结果2065
                 >=70，以19开头，比如82，结果1982
案例:
mysql> create table test_time(d date,t time,dt datetime);
Query OK, 0 rows affected (0.03 sec)
mysql> desc test_time;
+-------+----------+------+-----+---------+-------+
| Field | Type     | Null | Key | Default | Extra |
+-------+----------+------+-----+---------+-------+
| d     | date     | YES  |     | NULL    |       |
| t     | time     | YES  |     | NULL    |       |
| dt    | datetime | YES  |     | NULL    |       |
+-------+----------+------+-----+---------+-------+
3 rows in set (0.01 sec)
mysql> insert into test_time values(now(),now(),now());
Query OK, 1 row affected, 1 warning (0.02 sec)
mysql> select * from test_time;
+------------+----------+---------------------+
```

```
| d          | t        | dt                  |
+------------+----------+---------------------+
| 2019-08-23 | 00:26:29 | 2019-08-23 00:26:29 |
+------------+----------+---------------------+
1 row in set (0.00 sec)
```

测试年：

```
mysql> create table t3(born_year year);
Query OK, 0 rows affected (0.40 sec)

mysql> desc t3;
+-----------+---------+------+-----+---------+-------+
| Field     | Type    | Null | Key | Default | Extra |
+-----------+---------+------+-----+---------+-------+
| born_year | year(4) | YES  |     | NULL    |       |
+-----------+---------+------+-----+---------+-------+
1 row in set (0.00 sec)
mysql> insert into t3 values (12),(80);
Query OK, 2 rows affected (0.06 sec)
Records: 2  Duplicates: 0  Warnings: 0
mysql> select * from t3;
+-----------+
| born_year |
+-----------+
|      2012 |
|      1980 |
+-----------+
2 rows in set (0.00 sec)
mysql> insert into t3 values (2019),(80);
Query OK, 2 rows affected (0.00 sec)
Records: 2  Duplicates: 0  Warnings: 0
mysql> select * from t3;
+-----------+
| born_year |
+-----------+
|      2012 |
|      1980 |
|      2019 |
|      1980 |
+-----------+
4 rows in set (0.00 sec)
mysql>
```

2、约束

表完整性约束

作用：用于保证数据的完整性和一致性

约束条件	说明
PRIMARY KEY (PK)	标识该字段为该表的主键，可以唯一的标识记录，不可以为空 UNIQUE + NOT NULL
FOREIGN KEY (FK)	标识该字段为该表的外键，实现表与表（父表主键/子表1外键/子表2外键）之间的关联
NULL	标识是否允许为空，默认为NULL。
NOT NULL	标识该字段不能为空，可以修改。
UNIQUE KEY (UK)	标识该字段的值是唯一的，可以为空，一个表中可以有多个UNIQUE KEY
AUTO_INCREMENT	标识该字段的值自动增长（整数类型，而且为主键）
DEFAULT	为该字段设置默认值
UNSIGNED	无符号，正数

1.主键
每张表里只能有一个主键，不能为空，而且唯一。
定义两种方式：

```
mysql> create table t7(hostname char(20) primary key,ip char(150));
mysql> create table t9(hostname char(20),ip char(150),primary key(hostname));
```

删除主键

```
mysql> alter table t7  drop  primary key;
```

```
+----------+-----------+------+-----+---------+-------+
| Field    | Type      | Null | Key | Default | Extra |
+----------+-----------+------+-----+---------+-------+
| hostname | char(20)  | NO   | PRI | NULL    |       |
| ip       | char(150) | YES  |     | NULL    |       |
+----------+-----------+------+-----+---------+-------+
```

index(key)每张表可以有很多列做index，必须的起名
面试题：
导致SQL执行慢的原因：
1.硬件问题。如网络速度慢，内存不足，I/O吞吐量小，磁盘空间满了等。
2.没有索引或者索引失效.
3.数据过多（分库分表）
4.服务器调优及各个参数设置（调整my.cnf）
索引：当查询速度过慢可以通过建立优化查询速度，可以当作调优
创建索引：两种

```
mysql> create table t100(hostname char(20) primary key,ip char(150),index (ip));
mysql> create table t101(hostname char(20) primary key,ip char(150),index
dizhi(ip));
                                                                    #给ip做的索引，
名字叫dizhi
```

auto_increment--------自增 （每张表只能有一个字段为自曾） （成了key才可以自动增长）

```
mysql> CREATE TABLE department3 (
    -> dept_id INT PRIMARY KEY AUTO_INCREMENT,
    -> dept_name VARCHAR(30),
    -> comment VARCHAR(50)
    -> );
```

```
mysql> desc department3;
+-----------+-------------+------+-----+---------+----------------+
| Field     | Type        | Null | Key | Default | Extra          |
+-----------+-------------+------+-----+---------+----------------+
| dept_id   | int(11)     | NO   | PRI | NULL    | auto_increment |
| dept_name | varchar(30) | YES  |     | NULL    |                |
| comment   | varchar(50) | YES  |     | NULL    |                |
+-----------+-------------+------+-----+---------+----------------+
3 rows in set (0.00 sec)
```

```
设置唯一约束 UNIQUE
mysql> CREATE TABLE department2 (
    -> dept_id INT,
    -> dept_name VARCHAR(30) UNIQUE,
    -> comment VARCHAR(50)
    -> );
```

```
mysql> desc department2;
+-----------+-------------+------+-----+---------+-------+
| Field     | Type        | Null | Key | Default | Extra |
+-----------+-------------+------+-----+---------+-------+
| dept_id   | int(11)     | YES  |     | NULL    |       |
| dept_name | varchar(30) | YES  | UNI | NULL    |       |
| comment   | varchar(50) | YES  |     | NULL    |       |
+-----------+-------------+------+-----+---------+-------+
3 rows in set (0.00 sec)
```

```
mysql> insert into department2(dept_id,dept_name,comment) values(1,"tom","good");
Query OK, 1 row affected (0.01 sec)

mysql> insert into department2(dept_id,dept_name,comment) values(2,"tom","good");
ERROR 1062 (23000): Duplicate entry 'tom' for key 'dept_name'
mysql> insert into department2(dept_id,dept_name,comment) values(2,"jack","good");
Query OK, 1 row affected (0.00 sec)

mysql> insert into department2(dept_id,dept_name,comment) values(2,"jim","good");
Query OK, 1 row affected (0.00 sec)

mysql>
```

插入数据的时候id和comment字段相同可以插入数据，如果有相同的名字不唯一。所以插入数据失败。

```
1. 是否允许为空，默认NULL，可设置NOT NULL，字段不允许为空，必须赋值
2. 字段是否有默认值，缺省的默认值是NULL，如果插入记录时不给字段赋值，此字段使用默认值
sex enum('male','female') not null default 'male'  #只能选择maie和female，不允许为空，默认是male
age int unsigned NOT NULL default 20          #必须为正值（无符号） 不允许为空  默认是20
```

mysql> create table t5(id int(5),name varchar(10),age int(5) not null default 20);

```
mysql> desc t5;
+-------+-----------------+------+-----+---------+-------+
| Field | Type            | Null | Key | Default | Extra |
+-------+-----------------+------+-----+---------+-------+
| id    | int(5)          | YES  |     | NULL    |       |
| name  | varchar(10)     | YES  |     | NULL    |       |
| age   | int(5) unsigned | NO   |     | 20      |       |
+-------+-----------------+------+-----+---------+-------+
3 rows in set (0.00 sec)
```

```
mysql> insert into t5(id,name,age) values(1,"tom","18");
Query OK, 1 row affected (0.00 sec)

mysql> insert into t5(id,name,age) values(1,"jim","null");
ERROR 1366 (HY000): Incorrect integer value: 'null' for column 'age' at row 1
mysql>
```

指定字符集:

```
修改字符集 ：在创建表的最后面指定一下： default charset=utf8  #可以指定中文
```

注意

```
如果报错进入server端服务器登陆mysql执行:

mysql> use mysql

mysql> update user set host = '%' where user = 'root';

mysql> flush privileges;
```

作业

```
1.查一下oracle和mysql的区别写出几点即可。---面试
```

四、表操作

1、添加字段

```
1.添加新字段
alter table 表名 add 字段 修饰符;
mysql> alter table t3  add math int(10);-------添加的字段
mysql> alter table t3  add (chinese int(10),english int(10));------添加多个字段,中间用逗号隔开。
alter table 表名 add 添加的字段(和修饰) after name; --------把添加的字段放到name后面
alter table 表名 add 添加的字段(和修饰) first; ----------把添加的字段放在第一个
```

2.修改字段数据类型、修饰符

```
1.修改名称、数据类型、修饰符
alter table 表名 change 旧字段 新字段 修饰符; #change修改字段名称，类型，约束，顺序
mysql> alter table t3 change max maxs int(15) after id;  #修改字段名称与修饰并更换了位置
2.修改字段类型，约束，顺序
alter table 表名 modify 字段 属性 修饰符;  #modify 不能修改字段名称
mysql> alter table t3 modify maxs int(20) after math;    #修改修饰并更换位置
3.删除字段
mysql> alter table t3 drop maxs;  #drop 丢弃的字段。
```

3.插入数据(添加纪录)

```
字符串必须引号引起来
记录与表头相对应，表头与字段用逗号隔开。
1.添加一条记录
insert into 表名(字段1,字段2,字段3,字段4) values(1,"tom","m",90);
mysql> insert into t3(id,name,sex,age) values(1,"tom","m",18);
Query OK, 1 row affected (0.00 sec)
注:添加的记录与表头要对应,
2.添加多条记录
mysql> insert into t3(id,name,sex,age) values(2,"jack","m",19),
(3,"xiaoli","f",20);
Query OK, 2 rows affected (0.34 sec)
3.用set添加记录
mysql> insert into t3 set id=4,name="zhangsan",sex="m",age=21;
Query OK, 1 row affected (0.00 sec)
4.更新记录
update 表名 set  修改的字段  where  给谁修改;
mysql> update t3 set id=6 where name="xiaoli";
5.删除记录
1.删除单条记录
mysql> delete from t3 where id=6;   #删除那个记录，等于几会删除那个整条记录
Query OK, 1 row affected (0.35 sec)
2.删除所有记录
mysql> delete from t3;
```

4.单表查询

```
测试表:company.employee5
mysql> create database company;   #创建一个库;
创建一个测试表:
mysql> CREATE TABLE company.employee5(
     id int primary key AUTO_INCREMENT not null,
    name varchar(30) not null,
    sex enum('male','female') default 'male' not null,
     hire_date date not null,
     post varchar(50) not null,
     job_description varchar(100),
     salary double(15,2) not null,
     office int,
     dep_id int
     );
插入数据:
mysql> insert into
company.employee5(name,sex,hire_date,post,job_description,salary,office,dep_id)
values
```

```
    ('jack','male','20180202','instructor','teach',5000,501,100),
    ('tom','male','20180203','instructor','teach',5500,501,100),
    ('robin','male','20180202','instructor','teach',8000,501,100),
    ('alice','female','20180202','instructor','teach',7200,501,100),
    ('tianyun','male','20180202','hr','hrcc',600,502,101),
    ('harry','male','20180202','hr',NULL,6000,502,101),
    ('emma','female','20180206','sale','salecc',20000,503,102),
    ('christine','female','20180205','sale','salecc',2200,503,102),
    ('zhuzhu','male','20180205','sale',NULL,2200,503,102),
    ('gougou','male','20180205','sale','',2200,503,102);
mysql> use company
语法：
mysql> select   字段名称,字段名称2    from  表名    条件
简单查询
mysql> select * from employee5;
多字段查询：
mysql> select id,name,sex from employee5;
有条件查询：where
mysql> select id,name from employee5 where id<=3;
mysql> select id,name,salary from employee5 where salary>2000;
设置别名：as
mysql> select id as "id_num" from employee5 where id>5;
给 id 的值起个别名，显示值的表头会是设置的别名
统计记录数量：count()
mysql> select count(*) from employee5;
统计字段得到数量:
mysql> select count(id) from employee5;
避免重复DISTINCT:表里面的数据有相同的
mysql> select distinct post from employee5;
                        #字段     表名
```

```
表复制：key不会被复制： 主键、外键和索引
复制表
1.复制表结构＋记录 （key不会复制： 主键、外键和索引）
语法:create table 新表 select * from 旧表;
mysql> create table new_t1 select * from employee5;
2.复制单个字段:
mysql> create table new_t2(select id,name from employee5);
3.多条件查询:  and   ----和
语法: select   字段，字段2 from   表名   where   条件 and where 条件;
mysql> SELECT name,salary from employee5 where post='hr' AND salary>1000;
mysql> SELECT name,salary from employee5 where post='instructor' AND
salary>1000;
4.多条件查询:  or   ----或者
语法:        select   字段，字段2 from   表名   where   条件   or   条件;
mysql> select name from employee5 where salary>5000 and salary<10000 or
dep_id=102;
mysql> select name from employee5 where salary>2000 and salary<6000 or
dep_id=100;
5.关键字 BETWEEN AND  什么和什么之间。
mysql> SELECT name,salary FROM employee5 WHERE salary BETWEEN 5000 AND 15000;
mysql> SELECT name,salary FROM employee5 WHERE salary NOT BETWEEN 5000 AND
15000;
mysql> select name,dep_id,salary from employee5 where  not salary>5000 ;
注:not  给条件取反
6.关键字IS NULL   空的
mysql> SELECT name,job_description FROM employee5 WHERE job_description IS NULL;
```

```
mysql> SELECT name,job_description FROM employee5  WHERE job_description IS NOT NULL;  #-取反 不是null
mysql> SELECT name,job_description FROM employee5 WHERE job_description=''; #什么都没有==空
```

NULL说明：

1、等价于没有任何值、是未知数。

2、NULL与0、空字符串、空格都不同,NULL没有分配存储空间。

3、对空值做加、减、乘、除等运算操作，结果仍为空。

4、比较时使用关键字用“is null”和“is not null”。

5、排序时比其他数据都小（索引默认是降序排列，小→大），所以NULL值总是排在最前。

7.关键字IN集合查询

一般查询：

```
mysql> SELECT name,salary FROM employee5 WHERE salary=4000 OR salary=5000 OR salary=6000 OR salary=9000;
```

IN集合查询

```
mysql> SELECT name, salary FROM employee5 WHERE salary IN (4000,5000,6000,9000);
mysql> SELECT name, salary FROM employee5 WHERE salary NOT IN (4000,5000,6000,9000); #取反
```

8.排序查询　　order by　：命令指令，在mysql是排序的意思。

```
mysql> select name,salary from employee5 order by salary; #-默认从小到大排序。
mysql> select name,salary from employee5 order by salary desc; #降序，从大到小
```

9.limit 限制

```
mysql> select * from employee5 limit 5;  #只显示前5行
mysql> select name,salary from employee5 order by salary desc limit 0,1; #从第几行开始，打印一行
```

查找什么内容从那张表里面降序排序只打印第二行。

注意：

0-------默认第一行

1------第二行　依次类推...

```
mysql> SELECT * FROM employee5 ORDER BY salary DESC LIMIT 0,5;  #降序，打印5行
mysql> SELECT * FROM employee5 ORDER BY salary DESC LIMIT 4,5;  #从第5条开始，共显示5条
mysql> SELECT * FROM employee5 ORDER BY salary  LIMIT 4,3;  #默认从第5条开始显示3条。
```

10.分组查询 ：group　by

```
mysql> select count(name),post from employee5 group by post;
+-------------+------------+
| count(name) | post       |
+-------------+------------+
|           2 | hr         |
|           4 | instructor |
|           4 | sale       |
+-------------+------------+
```

count可以计算字段里面有多少条记录，如果分组会分组做计算

```
mysql> select count(name),group_concat(name) from employee5 where salary>5000;
```

查找 统计（条件：工资大于5000）的有几个人（count(name)），分别是谁（group_concat(name)）

```
+-------------+---------------------------+
| count(name) | group_concat(name)        |
+-------------+---------------------------+
|           5 | tom,robin,alice,harry,emma |
+-------------+---------------------------+
```

11.GROUP BY和GROUP_CONCAT()函数一起使用

GROUP_CONCAT()-------组连接

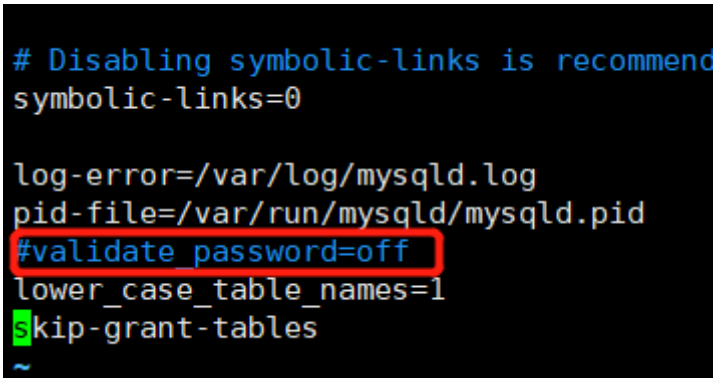
```
mysql> SELECT dep_id,GROUP_CONCAT(name) FROM employee5 GROUP BY dep_id; #以dep_id分的组，dep_id这个组里面都有谁
mysql> SELECT dep_id,GROUP_CONCAT(name) as emp_members FROM employee5 GROUP BY dep_id; #给组连接设置了一个别名
```

12.函数

```
max() 最大值
mysql> select max(salary) from employee5;
查询薪水最高的人的详细信息：
mysql> select name,sex,hire_date,post,salary,dep_id from employee5 where salary
= (SELECT MAX(salary) from employee5);
min()最小值
select min(salary) from employee5;
avg()平均值
select avg(salary) from employee5;
now()  现在的时间
select now();
sum()  计算和
select sum(salary) from employee5 where post='sale';
```

5.破解密码

先将这个修改成简单密码注释掉




```
root账户没了或者root密码丢失：
关闭Mysql使用下面方式进入Mysql直接修改表权限
  5.6/5.7版本：
    # mysqld --skip-grant-tables --user=mysql &

    # mysql -uroot
    mysql> UPDATE mysql.user SET authentication_string=password('QianFeng@123')
WHERE user='root' AND host='localhsot';
    mysql> FLUSH PRIVILEGES;
```

五、权限管理

1.用户管理

```
1. 登录和退出MySQL
远程登陆：
客户端语法：mysql  -u  用户名  -p  密码  -h  ip地址   -P端口号:如果没有改端口号就不用-P指
定端口
# mysql -h192.168.246.253 -P 3306 -uroot -pqf123
```

如果报错进入server端服务器登陆mysql执行:

mysql> use mysql

mysql> update user set host = '%' where user = 'root';

mysql> flush privileges;

```
# mysql -h192.168.246.253 -P 3306 -uroot -pqf123 -e 'show databases;'
-h   指定主机名                【默认为localhost】
-大P MySQL服务器端口          【默认3306】
-u   指定用户名                【默认root】
-p   指定登录密码              【默认为空密码】
-e   接SQL语句，可以写多条拿;隔开
# mysql -h192.168.246.253 -P 3306 -uroot -pqf123 -D mysql -e 'select host from
user;'
此处 -D mysql为指定登录的数据库
修改端口rpm安装：vim /etc/my.cnf
在到【mysql】标签下面添加port=指定端口。重启服务
```

2.创建用户

```
方法一：CREATE USER语句创建
mysql> create user tom@'localhost' identified by 'qf@123'; #创建用户为tom，并设置密
码。
mysql> FLUSH PRIVILEGES; #更新授权表
注:
identified by ：设置密码
在用户tom@'  ' 这里 选择:
%：允许远程登陆。也可以指定某个ip，允许某个ip登陆。也可以是一个网段。
%：包括所有的主机，不包括本机（127.0.0.1），但是不包括（localhost）
==客户端主机      %                    所有主机
              192.168.246.%          192.168.246.0网段的所有主机
              192.168.246.252        指定主机
              localhost               指定主机
```

```
方法二：  GRANT语句创建   ---授权。
mysql> GRANT ALL ON *.* TO 'user3'@'localhost' IDENTIFIED BY 'Qf@123';
            #权限 库名.表名  账户名                设置密码
mysql> FLUSH PRIVILEGES;
Query OK, 0 rows affected (0.00 sec)
```

```
修改远程登陆:
将原来的localhost修改为%或者ip地址
mysql> use mysql
mysql> update user set host = '192.168.246.%' where user = 'user3';
mysql> FLUSH PRIVILEGES;
```

3、刷新权限

```
修改表之后需要刷新权限
方式1:
    mysql > flush privileges;
```

```
使用命令授权：grant
   也可创建新账户(不过后面的版本会移除这个功能，建议使用create user)
语法格式:
grant 权限列表  on 库名.表名 to '用户名'@'客户端主机' IDENTIFIED BY 'Qf@123';
 ==权限列表      all          所有权限（不包括授权权限）
              select,update
              select, insert
```

```
 ==数据库.表名     *.*          所有库下的所有表                    Global level
                 web.*         web库下的所有表                     Database level
              web.stu_info     web库下的stu_info表                  Table level
给刚才创建的用户tom授权：
mysql> grant select,insert on *.* to 'tom'@'localhost';
mysql> FLUSH PRIVILEGES;
```

4、查看权限

```
查看权限
1.看自己的权限：
mysql> SHOW GRANTS\G
*************************** 1. row ***************************
Grants for root@%: GRANT ALL PRIVILEGES ON *.* TO 'root'@'%' WITH GRANT OPTION

2.看别人的权限：
mysql> SHOW GRANTS FOR tom@'localhost'\G
*************************** 1. row ***************************
Grants for tom@localhost: GRANT SELECT, INSERT ON *.* TO 'tom'@'localhost'
```

5、修改密码

```
===root修改自己密码
方法一：
# mysqladmin -uroot -p'123' password 'new_password'     //123为旧密码
案例：
# mysqladmin -uroot -p'qf123' password 'qf@123';
方法二：
mysql>SET PASSWORD='new_password';

==root修改其他用户密码
mysql> use mysql
mysql> SET PASSWORD FOR user3@'localhost'='new_password'
                                  用户              =   新密码
```

6、删除用户

```
方法一：DROP USER语句删除
DROP USER 'user3'@'localhost';

方法二：DELETE语句删除
DELETE FROM mysql.user  WHERE user='tom' AND host='localhost';

更新授权表：   FLUSH PRIVILEGES;
```

7、权限控制机制

```
四张表：user   db    tables_priv  columns_priv
1.用户认证
查看mysql.user表
2.权限认证
以select权限为例：
1.先看 user表里的select_priv权限
 Y:不会接着查看其他的表   拥有查看所有库所有表的权限
 N:接着看db表
```

2.db表: 存储了某个用户对一个数据库的权限。
 Y:不会接着查看其他的表 拥有查看所有库所有表的权限
 N:接着看tables_priv表
3.tables_priv表：可以对单个表进行权限设置
 table_priv:如果这个字段的值里包括select 拥有查看这张表所有字段的权限，不会再接着往下看了
 table_priv:如果这个字段的值里不包括select，接着查看下张表还需要有column_priv字段权限
4.columns_priv:针对数据列设置操作权限.
 column_priv:有select，则只对某一列有select权限
             没有则对所有库所有表没有任何权限
 注：其他权限设置一样。

六、日志管理

Log Type	Information Written to Log
Error log	Problems encountered starting, running, or stopping **mysqld**
General query log	Established client connections and statements received from clients
Binary log	Statements that change data (also used for replication)
Relay log	Data changes received from a replication master server
Slow query log	Queries that took more than `long_query_time` seconds to execute
DDL log (metadata log)	Metadata operations performed by DDL statements

1 错误日志 ：启动，停止，关闭失败报错。rpm安装日志位置 /var/log/mysqld.log #默认开启
2 通用查询日志：所有的查询都记下来。 #默认关闭，一般不开启
3 二进制日志(bin log)：实现备份，增量备份。只记录改变数据，除了select都记。
4 中继日志(Relay log)：读取主服务器的binlog，在slave机器本地回放。保持与主服务器数据一致。
5 slow log：慢查询日志，指导调优，定义某一个查询语句，执行时间过长，通过日志提供调优建议给开发人员。
6 DDL log： 定义语句的日志。

```
Error Log
[root@mysql-server ~]# vim /etc/my.cnf
log-error=/var/log/mysqld.log
编译安装的在/usr/local/mysql/

Binary Log:前提需要开启
[root@mysql-server ~]# vim /etc/my.cnf
log-bin=/var/log/mysql-bin/mylog  #如果不指定路径默认在/var/lib/mysql
server-id=1   #AB复制的时候使用，为了防止相互复制，会设置一个ID，来标识谁产生的日志

[root@mysql-server ~]# mkdir /var/log/mysql-bin
[root@mysql-server ~]# chown mysql.mysql /var/log/mysql-bin/
[root@mysql-server ~]# systemctl restart mysqld
```

```
查看binlog日志：开启之后等一会
[root@mysql-server mysql]# mysqlbinlog mylog.000001 -v
# at 4     #位置点
#190820 19:41:26  #时间点
注：
1. 重启mysqld 会截断
2. mysql> flush logs; 会截断
3. mysql> reset master; 删除所有binlog,不要轻易使用，相当于：rm -rf /
4. 删除部分
```

```
mysql> PURGE BINARY LOGS TO 'mylog.000004';   #删除mysqllog.000004之前的日志
5. 暂停 仅当前会话
SET SQL_LOG_BIN=0;  #关闭
SET SQL_LOG_BIN=1;  #开启
=====================================
解决binlog日志不记录insert语句
登录mysql后，设置binlog的记录格式：
mysql> set binlog_format=statement;
然后，最好在my.cnf中添加：
binlog_format=statement
修改完配置文件之后记得重启服务
=============================================
Slow Query Log ： 慢查询日志
slow_query_log=1  #开启
slow_query_log_file=/var/log/mysql-slow/slow.log
long_query_time=3    #设置慢查询超时间，单位是秒

# mkdir /var/log/mysql-slow/
# chown mysql.mysql /var/log/mysql-slow/
# systemctl restart mysqld

验证查看慢查询日志
mysql> select sleep(6);
# cat /var/log/mysql-slow/slow.log
```

七、数据备份与恢复

1.MySQL数据备份

所有备份数据都应放在非数据库本地，而且建议有多份副本。

测试环境中做日常恢复演练，恢复较备份更为重要。

备份： 能够防止由于机械故障以及人为误操作带来的数据丢失，例如将数据库文件保存在了其它地方。
冗余： 数据有多份冗余，但不等备份，只能防止机械故障还来的数据丢失，例如主备模式、数据库集群。

备份过程中必须考虑因素：

```
1. 数据的一致性
2. 服务的可用性
```

逻辑备份： 备份的是建表、建库、插入等操作所执行SQL语句（DDL DML DCL），适用于中小型数据库，效率相对较低。

mysqldump binlog mydumper **phpmyadmin**

物理备份： 直接复制数据库文件，适用于大型数据库环境，不受存储引擎的限制，但不能恢复到不同的MySQL版本。 tar,cp mysqlhotcopy 只能用于备份MyISAM。 **xtrabackup** inbackup lvm snapshot

2.物理备份的方式

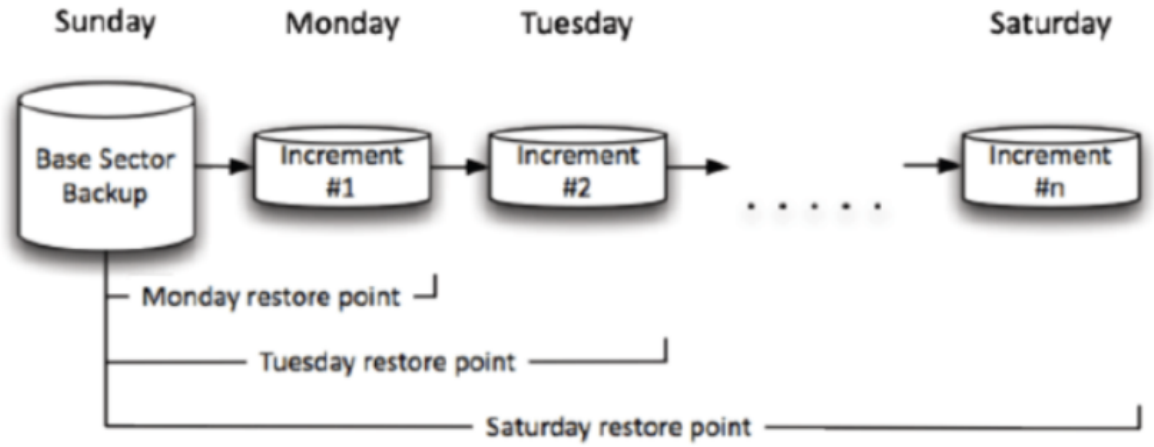
1.完全备份-----完整备份：

每次都将所有数据（不管自第一次备份以来有没有修改过），进行一次完整的复制.

特点：占用空间大，备份速度慢，但恢复时一次恢复到位，恢复速度快。

2.增量备份: 每次备份上一次备份到现在产生的新数据

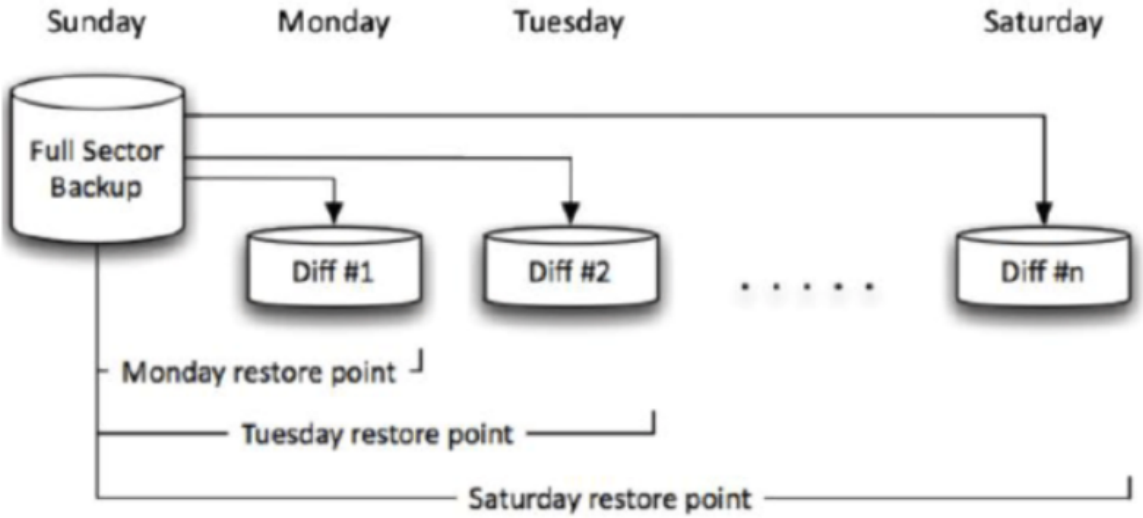
特点：因每次仅备份自上一次备份（注意是上一次，不是第一次）以来有变化的文件，所 以备份体积小，备份速度快，但是恢复的时候，需要按备份时间顺序，逐个备份版本进行恢 复，恢复时间长。



3.差异备份:只备份跟完整备份不一样的

在第一次完整备份之后，第二次开始每次都将所有文件与第一次完整备份的文件做比较，把自第一次完整备份以来所有修改过的文件进行备份，且以后每次备份都是和第一次完整备份进行比较（注意是第一次，不是上一次），备份自第一次完整备份以来所有的修改过的文件。

特点：占用空间比增量备份大，比完整备份小，恢复时仅需要恢复第一个完整版本和最后一次的差异版本，恢复速度介于完整备份和增量备份之间。

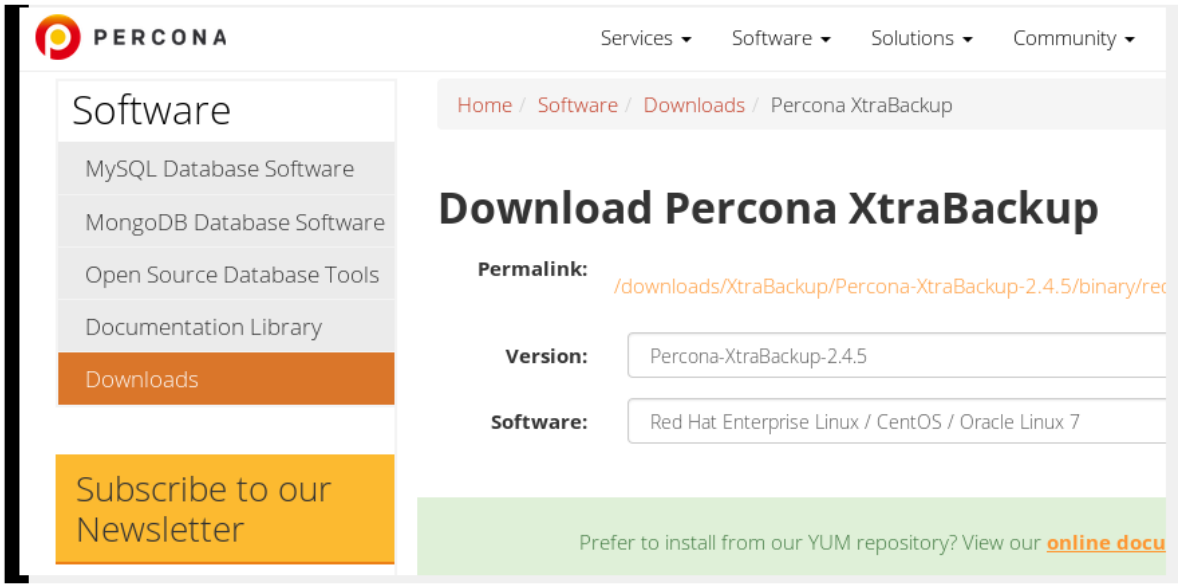


简单的讲，完整备份就是不管三七二十一，每次都把指定的备份目录完整的复制一遍，不管目录下的文件有没有变化；增量备份就是每次将之前（第一次、第二次、直到前一次）做过备份之后有变化的文件进行备份；差异备份就是每次都将第一次完整备份以来有变化的文件进行备份。

```
热备份
数据库启动同时给客户端提供服务的情况下
冷备份
数据库要关掉或者不能给客户端提供服务
```

3、percona-xtrabackup 物理备份

是开源免费的支持MySQL 数据库热备份的软件，它能对InnoDB和XtraDB存储引擎的数据库非阻塞地备份。它不暂停服务创建Innodb**热备份**； 为mysql做增量备份；在mysql服务器之间做在线表迁移；使创建replication更加容易；备份mysql而不增加服务器的负载。 percona是一家老牌的mysql技术咨询公司。它不仅提供mysql的技术支持、培训、咨询，还发布了mysql的分支版本--percona Server。并围绕percona Server还发布了一系列的mysql工具。



1、安装xtrabackup

```
安装xtrabackup
# wget http://www.percona.com/downloads/percona-release/redhat/0.1-4/percona-release-0.1-4.noarch.rpm
# rpm -ivh percona-release-0.1-4.noarch.rpm
[root@mysql-server yum.repos.d]# vim percona-release.repo
```

修改如下内容：将原来的1改为0

```
########################################
# Percona releases and sources, stable #
########################################
[percona-release-$basearch]
name = Percona-Release YUM repository - $basearch
baseurl = http://repo.percona.com/release/$releasever/RPMS/$basearch
enabled = 1
gpgcheck = 0
gpgkey = file:///etc/pki/rpm-gpg/RPM-GPG-KEY-Percona

[percona-release-noarch]
name = Percona-Release YUM repository - noarch
baseurl = http://repo.percona.com/release/$releasever/RPMS/noarch
enabled = 1
gpgcheck = 0
gpgkey = file:///etc/pki/rpm-gpg/RPM-GPG-KEY-Percona

[percona-release-source]
name = Percona-Release YUM repository - Source packages
baseurl = http://repo.percona.com/release/$releasever/SRPMS
enabled = 0
gpgcheck = 1
gpgkey = file:///etc/pki/rpm-gpg/RPM-GPG-KEY-Percona
```

```
[root@mysql-server yum.repos.d]# yum -y install percona-xtrabackup-24.x86_64
```

注意

```
如果安装不上报错：
Transaction check error:
  file /etc/my.cnf from install of Percona-Server-shared-56-5.6.46-
rel86.2.1.el7.x86_64 conflicts with file from package mysql-community-server-
5.7.28-1.el7.x86_64
Error Summary  #说是冲突
解决方式如下：
1.先安装yum install mysql-community-libs-compat -y #安装包
2.在安装yum -y install percona-xtrabackup-24.x86_64

参考：https://www.cnblogs.com/EikiXu/p/10217931.html

方式二：
1.先安装percona-xtrabackup
2.在安装mysql
或者先将mysql源back了，重新建立yum缓存。在安装percona-xtrabackup。
```

2.完全备份流程:

```
创建备份目录：
[root@mysql-server ~]# mkdir /xtrabackup/full -p
备份：
[root@mysql-server ~]# innobackupex --user=root --password='qf123'
/xtrabackup/full
```

```
190820 11:47:52 [00]        ...done
190820 11:47:52 [00] Writing /xtrabackup/full/2019-08-20_11-47-49/xtrabackup_info
190820 11:47:52 [00]        ...done
xtrabackup: Transaction log of lsn (3028497) to (3028506) was copied.
190820 11:47:52 completed OK!
```

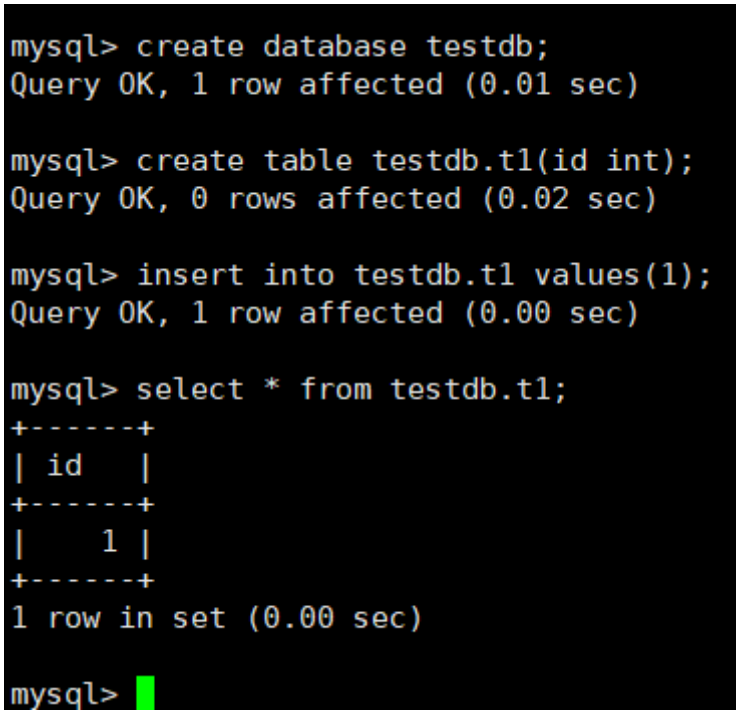
```
可以查看一下：
[root@mysql-server ~]# cd /xtrabackup/full/
[root@mysql-server full]# ls
2019-08-20_11-47-49
=======================================================
完全备份恢复流程
1. 停止数据库
2. 清理环境
3. 重演回滚——> 恢复数据
4. 修改权限
5. 启动数据库
1.关闭数据库：
[root@mysql-server ~]# systemctl stop mysqld
[root@mysql-server ~]# rm -rf /var/lib/mysql/*
[root@mysql-server ~]# rm -rf /var/log/mysqld.log
[root@mysql-server ~]# rm -rf /var/log/mysql-slow/slow.log
2.重演恢复：
[root@mysql-server ~]# innobackupex --apply-log /xtrabackup/full/2019-08-20_11-
47-49
3.确认数据库目录：
恢复之前需要确认配置文件内有数据库目录指定，不然xtrabackup不知道恢复到哪里
```

```
# cat /etc/my.cnf
[mysqld]
datadir=/var/lib/mysql
4.恢复数据：
[root@mysql-server ~]# innobackupex --copy-back /xtrabackup/full/2019-08-20_11-47-49
5.修改权限：
[root@mysql-server ~]# chown mysql.mysql  /var/lib/mysql  -R
启动数据库：
[root@mysql-server ~]# systemctl start mysqld
```

3.增量备份流程

原理：每次备份上一次备份到现在产生的新数据

```
1.在数据库上面创建一个测试的库
```



1.完整备份:周一

```
[root@mysql-server ~]# rm -rf /xtrabackup/*
[root@mysql-server ~]# innobackupex --user=root --password='qf123' /xtrabackup
[root@mysql-server ~]# cd /xtrabackup/
[root@mysql-server xtrabackup]# ls
2019-08-20_14-51-35
[root@mysql-server xtrabackup]# cd 2019-08-20_14-51-35/
[root@mysql-server 2019-08-20_14-51-35]# ls
backup-my.cnf  ib_buffer_pool  mysql               sys   testdb
xtrabackup_info
company        ibdata1         performance_schema  test  xtrabackup_checkpoints
xtrabackup_logfile
```

2、增量备份：周二　——　周三

在数据库中插入周二的数据：

```
mysql> insert into testdb.t1 values(2);    #模拟周二
[root@mysql-server ~]# innobackupex --user=root --password='qf123' --incremental /xtrabackup/ --incremental-basedir=/xtrabackup/2019-08-20_14-51-35/
--incremental-basedir:基于哪个增量
[root@mysql-server ~]# cd /xtrabackup/
[root@mysql-server xtrabackup]# ls
2019-08-20_14-51-35  2019-08-20_15-04-29    ---相当于周二的增量备份
```

在数据库中插入周三的数据：

```
mysql> insert into testdb.t1 values(3);   #模拟周三
[root@mysql-server ~]# innobackupex --user=root --password='qf123' --incremental /xtrabackup/ --incremental-basedir=/xtrabackup/2019-08-20_15-04-29/      #基于前一天的备份为目录
[root@mysql-server ~]# cd /xtrabackup/
[root@mysql-server xtrabackup]# ls
2019-08-20_14-51-35  2019-08-20_15-04-29  2019-08-20_15-10-56   ---相当于周三的增量备份
```

查看一下备份目录：

```
[root@mysql-server ~]# ls /xtrabackup/
2019-08-20_14-51-35  2019-08-20_15-04-29  2019-08-20_15-10-56
    全备周一              增量周二                增量周三
```

增量备份恢复流程

1. 停止数据库
2. 清理环境
3. 依次重演回滚redo log——> 恢复数据
4. 修改权限
5. 启动数据库

```
[root@mysql-server ~]# systemctl stop mysqld
[root@mysql-server ~]# rm -rf /var/lib/mysql/*
依次重演回滚redo log:
[root@mysql-server ~]# innobackupex --apply-log --redo-only /xtrabackup/2019-08-20_14-51-35
周二 ---  周三
[root@mysql-server ~]# innobackupex --apply-log --redo-only /xtrabackup/2019-08-20_14-51-35 --incremental-dir=/xtrabackup/2019-08-20_15-04-29
--incremental-dir：增量目录
[root@mysql-server ~]# innobackupex --apply-log --redo-only /xtrabackup/2019-08-20_14-51-35 --incremental-dir=/xtrabackup/2019-08-20_15-10-56/
恢复数据：
[root@mysql-server ~]# innobackupex --copy-back /xtrabackup/2019-08-20_14-51-35/
修改权限
[root@mysql-server ~]# chown -R mysql.mysql /var/lib/mysql
[root@mysql-server ~]# systemctl start mysqld
登陆上去看一下：
```

```
[root@mysql-server ~]# mysql -uroot -pqf123
mysql: [Warning] Using a password on the command line interface can be insecure.
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 2
Server version: 5.7.27 MySQL Community Server (GPL)

Copyright (c) 2000, 2019, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> select * from testdb.t1;
+------+
| id   |
+------+
|    1 |
|    2 |
|    3 |
+------+
3 rows in set (0.01 sec)

mysql>
```

4、差异备份流程

清理备份的环境:

```
[root@mysql-server ~]# rm -rf /xtrabackup/*
登陆数据库，准备环境
mysql> delete from testdb.t1;
mysql> insert into testdb.t1 values(1);    #插入数据1，模拟周一
mysql> select * from testdb.t1;
+------+
| id   |
+------+
|    1 |
+------+
mysql> \q
查看时间:
[root@mysql-server ~]# date
Tue Aug 20 15:39:59 CST 2019
1、完整备份：周一
[root@mysql-server ~]# innobackupex --user=root --password='qf123' /xtrabackup
2、差异备份：周二  --  周三
语法: # innobackupex --user=root --password=888 --incremental /xtrabackup --
incremental-basedir=/xtrabackup/完全备份目录（周一）
3.修改时间:
[root@mysql-server ~]# date 08211543
Wed Aug 21 15:43:00 CST 2019
4.在登陆mysql:
mysql> insert into testdb.t1 values(2);  #插入数据2，模拟周二
差异备份周二的
[root@mysql-server ~]# innobackupex --user=root --password='qf123' --incremental
/xtrabackup --incremental-basedir=/xtrabackup/2019-08-20_15-42-02/  #备份目录基于
周一的备份
5.再次登陆mysql
mysql> insert into testdb.t1 values(3);  #插入数据，模拟周三
6.在次修改时间
```

```
[root@mysql-server ~]# date 08221550
Thu Aug 22 15:50:00 CST 2019
7.在次差异备份
[root@mysql-server ~]# innobackupex --user=root --password='qf123' --incremental /xtrabackup --incremental-basedir=/xtrabackup/2019-08-20_15-42-02/  #还是基于周一的备份
8.延申到周四
mysql> insert into testdb.t1 values(4);
9.修改时间
[root@mysql-server ~]# date 08231553
Fri Aug 23 15:53:00 CST 2019
10.差异备份周四
[root@mysql-server ~]# innobackupex --user=root --password='qf123' --incremental /xtrabackup --incremental-basedir=/xtrabackup/2019-08-20_15-42-02/   #还是基于周一的备份
11.查看一下备份目录
[root@mysql-server ~]# ls /xtrabackup/
2019-08-20_15-42-02  2019-08-21_15-46-53  2019-08-22_15-51-15  2019-08-23_15-53-28
   周一                    周二                   周三                  周四
```

```
差异备份恢复流程
1. 停止数据库
2. 清理环境
3. 重演回滚redo log（周一，某次差异）——> 恢复数据
4. 修改权限
5. 启动数据库
停止数据库
[root@mysql-server ~]# systemctl stop mysqld
[root@mysql-server ~]# rm -rf /var/lib/mysql/*
1.恢复全量的redo log
语法: # innobackupex --apply-log --redo-only /xtrabackup/完全备份目录（周一）
[root@mysql-server ~]# innobackupex --apply-log --redo-only /xtrabackup/2019-08-20_15-42-02/
2.恢复差异的redo log
语法:# innobackupex --apply-log --redo-only /xtrabackup/完全备份目录（周一）--incremental-dir=/xtrabacku/某个差异备份
这里我们恢复周三的差异备份
[root@mysql-server ~]# innobackupex --apply-log --redo-only /xtrabackup/2019-08-20_15-42-02/ --incremental-dir=/xtrabackup/2019-08-22_15-51-15/   #我们恢复周三的差异备份
3.恢复数据
语法:# innobackupex --copy-back /xtrabackup/完全备份目录（周一）
[root@mysql-server ~]# innobackupex --copy-back /xtrabackup/2019-08-20_15-42-02/
修改权限:
[root@mysql-server ~]# chown -R mysql.mysql /var/lib/mysql
[root@mysql-server ~]# systemctl start mysqld
```

登陆mysql查看一下:

```
[root@mysql-server ~]# mysql -uroot -pqf123
mysql: [Warning] Using a password on the command line interface can be insecure.
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 2
Server version: 5.7.27 MySQL Community Server (GPL)

Copyright (c) 2000, 2019, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> select * from testdb.t1;
+------+
| id   |
+------+
|    1 |
|    2 |
|    3 |
+------+
```

只有123.因为我们恢复的是周三的差异备份。

4.mysqldump逻辑备份 ---- 推荐优先使用

mysqldump可以保证 数据一致，服务可用。

如何保证数据一致?在备份的时候进行锁表会自动锁表。锁住之后在备份。

```
本身为客户端工具:
远程备份语法: # mysqldump  -h 服务器  -u用户名  -p密码   数据库名  > 备份文件.sql
本地备份语法: # mysqldump  -u用户名  -p密码   数据库名  > 备份文件.sql
```

1.常用备份选项

-A, --all-databases
备份所有库

-B, --databases bbs test mysql 备份多个数据库

-F, --flush-logs
备份之前刷新binlog日志

--default-character-set 指定导出数据时采用何种字符集，如果数据表不是采用默认的 latin1 字符集的话，那么导出 时必须指定该选项，否则再次导入数据后将产生乱码问题。

--no-data，-d
不导出任何数据，只导出数据库表结构。

注意

```
使用 mysqldump 备份数据库时避免锁表
对一个正在运行的数据库进行备份请慎重！！ 如果一定要 在服务运行期间备份，可以选择添加 --single-transaction选项，

类似执行： mysqldump --single-transaction -u root -p123456 dbname > mysql.sql
```

2.备份表

```
语法: # mysqldump -u root -p1 db1  t1 > /db1.t1.bak
[root@mysql-server ~]# mkdir /home/back  #创建备份目录
[root@mysql-server ~]# mysqldump -uroot -p'qf123' company employee5 >
/home/back/company.employee5.bak
备份多个表:
语法: mysqldump -u root -p1 db1 t1   t2 > /db1.t1_t2.bak
[root@mysql-server ~]# mysqldump -uroot -p'qf123' company new_t1 new_t2  >
/home/back/company.new_t1_t2.bak
```

3、备份库

```
备份一个库:相当于将这个库里面的所有表全部备份。
语法: # mysqldump -u root -p1 db1 > /db1.bak
[root@mysql-server ~]# mysqldump -uroot -p'qf123' company >
/home/back/company.bak
备份多个库:
语法:mysqldump  -u root -p1 -B  db1 db2 db3 > /db123.bak
[root@mysql-server ~]# mysqldump -uroot -p'qf123' -B company testdb >
/home/back/company_testdb.bak
备份所有的库:
语法:# mysqldump  -u root -p1 -A > /alldb.bak
[root@mysql-server ~]# mysqldump -uroot -p'qf123' -A > /home/back/allbase.bak
```

到目录下面查看一下:

```
[root@mysql-server ~]# cd /home/back/
[root@mysql-server back]# ls
allbase.bak  company.bak  company.employee5.bak  company.new_t1_t2.bak  company_testdb.bak
[root@mysql-server back]#
```

4、恢复数据库和表

为保证数据一致性，应在恢复数据之前停止数据库对外的服务,停止binlog日志 因为binlog使用binlog日志恢复数据时也会产生binlog日志。

为实验效果先将刚才备份的数据库和表删除了。登陆数据库:

[root@mysql-server ~]# mysql -uroot -pqf123

mysql> show databases;

```
+--------------------+
| Database           |
+--------------------+
| information_schema |
| company            |
| mysql              |
| performance_schema |
| sys                |
| test               |
| testdb             |
+--------------------+
7 rows in set (0.00 sec)
```

mysql> drop database company; mysql> \q

5、恢复库

```
登陆mysql创建一个库
mysql> create database company;
恢复：
[root@mysql-server ~]# mysql -uroot -p'qf123' company < /home/back/company.bak
```

6、恢复表

```
登陆到刚才恢复的库中将其中的一个表删除掉
mysql> show databases;
mysql> use company
mysql> show tables;
+-------------------+
| Tables_in_company |
+-------------------+
| employee5         |
| new_t1            |
| new_t2            |
+-------------------+
mysql> drop table employee5;
开始恢复：
mysql> set sql_log_bin=0;   #停止binlog日志
Query OK, 0 rows affected (0.00 sec)
mysql> source /home/back/company.employee5.bak;  -------加路径和备份的文件
恢复方式二：
# mysql -u root -p1  db1  < db1.t1.bak
                     库名     备份的文件路径
```

7、备份及恢复表结构

```
1.备份表结构：
语法：mysqldump  -uroot -p123456 -d database table > dump.sql
[root@mysql-server ~]# mysqldump -uroot -p'qf123' -d company employee5 >
/home/back/emp.bak
恢复表结构：
登陆数据库创建一个库
mysql> create database t1;
语法：# mysql -u root -p1 -D db1  < db1.t1.bak
[root@mysql-server ~]# mysql -uroot -p'qf123' -D t1 < /home/back/emp.bak
```

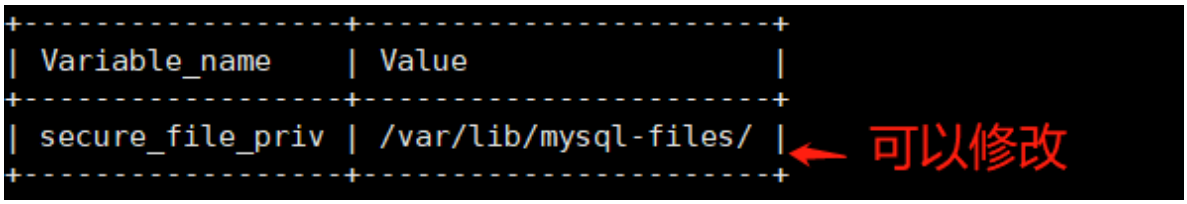
登陆数据查看：

```
mysql> desc employee5;
+-----------------+-----------------------+------+-----+---------+----------------+
| Field           | Type                  | Null | Key | Default | Extra          |
+-----------------+-----------------------+------+-----+---------+----------------+
| id              | int(11)               | NO   | PRI | NULL    | auto_increment |
| name            | varchar(30)           | NO   |     | NULL    |                |
| sex             | enum('male','female') | NO   |     | male    |                |
| hire_date       | date                  | NO   |     | NULL    |                |
| post            | varchar(50)           | NO   |     | NULL    |                |
| job_description | varchar(100)          | YES  |     | NULL    |                |
| salary          | double(15,2)          | NO   |     | NULL    |                |
| office          | int(11)               | YES  |     | NULL    |                |
| dep_id          | int(11)               | YES  |     | NULL    |                |
+-----------------+-----------------------+------+-----+---------+----------------+
9 rows in set (0.00 sec)
```

8、数据的导入导出。没有表结构。

表的导出和导入只备份表内记录，不会备份表结构，需要通过mysqldump备份表结构，恢复时先恢复表结构，再导入数据。

```
mysql> show variables like "secure_file_priv";  ----查询导入导出的目录。（保证数据安全做共享）
```




```
修改安全文件目录：
1.创建一个目录：mkdir  路径目录
[root@mysql-server ~]# mkdir /sql
2.修改权限
[root@mysql-server ~]# chown mysql.mysql /sql
3.编辑配置文件：
vim /etc/my.cnf
在[mysqld]里追加
secure_file_priv=/sql
4.重新启动mysql.
```

```
1.到出数据
登陆数据查看数据
mysql> show databases;    #找到test库
mysql> use test   #进入test库
mysql> show tables;  #找到它t3表
mysql> select * from t3 into outfile '/sql/test.t3.bak';
添加修饰的：
mysql> select * from t3 into outfile '/sql/test.t3.bak1' fields terminated by ',' lines terminated by '\n';
注：
fields terminated by ',' ：字段以逗号分割
lines terminated by '\n'：结尾换行
```

```
2.数据的导入
先将原来表里面的数据清除掉，只保留表结构
mysql> delete from t3;
mysql> load data infile '/sql/test.t3.bak' into table t3;
如果将数据导入别的表，需要创建这个表并创建相应的表结构。
```

5、通过binlog恢复

开启binlog日志：

```
# Remove leading # to set options mainly useful for reporting servers.
# The server defaults are faster for transactions and fast SELECTs.
# Adjust sizes as needed, experiment to find the optimal values.
# join_buffer_size = 128M
# sort_buffer_size = 2M
# read_rnd_buffer_size = 2M
datadir=/var/lib/mysql
socket=/var/lib/mysql/mysql.sock

# Disabling symbolic-links is recommended to prevent assorted security r
symbolic-links=0

log-error=/var/log/mysqld.log
pid-file=/var/run/mysqld/mysqld.pid
validate_password=off
lower_case_table_names=1
#skip-grant-tables
secure_file_priv=/sql
log-bin=/var/log/sql-bin/mylog
server-id=1
#slow_query_log=1
#slow_query_log_file=/var/log/mysql-slow/slow.log
#long_query_time=3
~
~
```

id号做主从时用到，这里必须写

创建目录并修改权限

```
[root@mysql-server ~]# mkdir /var/log/sql-bin
[root@mysql-server ~]# chown mysql.mysql /var/log/sql-bin
[root@mysql-server ~]# systemctl restart mysqld
```

```
[root@mysql-server ~]# cd /var/log/sql-bin/
[root@mysql-server sql-bin]# ls
mylog.000001  mylog.index
[root@mysql-server sql-bin]#
```

```
mysql> flush logs; #刷新binlog日志会截断产生新的日志文件

mysql> create table testdb.t3(id int);   #创建一个表
```

```
[root@mysql-server ~]# cd /var/log/sql-bin/
[root@mysql-server sql-bin]# ls
mylog.000001  mylog.000002  mylog.index
[root@mysql-server sql-bin]#
```

根据位置恢复

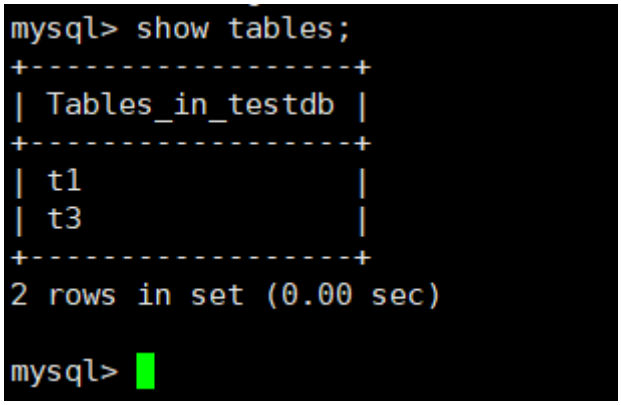
找到要恢复的sql语句的起始位置、结束位置

```
[root@mysql-server sql-bin]# mysqlbinlog mylog.000002
```

测试

```
[root@mysql-server ~]# mysql -uroot -p'qf123'
mysql> drop table testdb.t3;   #将这个表删除
Query OK, 0 rows affected (0.01 sec)
恢复：
[root@mysql-server ~]# cd /var/log/sql-bin/
[root@mysql-server sql-bin]# mysqlbinlog --start-position 219 --stop-position
321 mylog.000002 |mysql -uroot -p'qf123'
mysql: [Warning] Using a password on the command line interface can be insecure.
```

查看：



八、AB复制

一、什么是主从复制?

1、主从复制，是用来建立一个和主数据库完全一样的数据库环境，称为从数据库；主数据库一般是准实时的业务数据库。

2、主从复制的作用

```
1.做数据的热备，作为后备数据库，主数据库服务器故障后，可切换到从数据库继续工作，避免数据丢失。
2.架构的扩展。业务量越来越大，I/O访问频率过高，单机无法满足，此时做多库的存储，降低磁盘I/O访问
的频率,提高单个机器的I/O性能。
3.读写分离，使数据库能支撑更大的并发。
 1--在从服务器可以执行查询工作(即我们常说的读功能)，降低主服务器压力;（主库写，从库读，降压）
 2--在从服务器进行备份，避免备份期间影响主服务器服务;（确保数据安全）
```

二、主从复制原理

原理：
实现整个主从复制，需要由Master服务器上的IO进程，和Slave服务器上的Sql进程和IO进程共同完成.
要实现主从复制，首先必须打开Master端的binary log（bin-log）功能，因为整个MySQL 复制过程实际上就是Slave从Master端获取相应的二进制日志，然后再在自己slave端完全顺序的执行日志中所记录的各种操作。
==========
1. 在主库上把数据更改（DDL DML DCL）记录到二进制日志（Binary Log）中。
2. 备库I/O线程将主库上的日志复制到自己的中继日志（Relay Log）中。
3. 备库SQL线程读取中继日志中的事件，将其重放到备库数据库之上。
==========
master　　负责写 -----A
slave　　relay-log　-----B
I/o　负责通信读取binlog日志
SQL　负责写数据

mysql主从复制原理

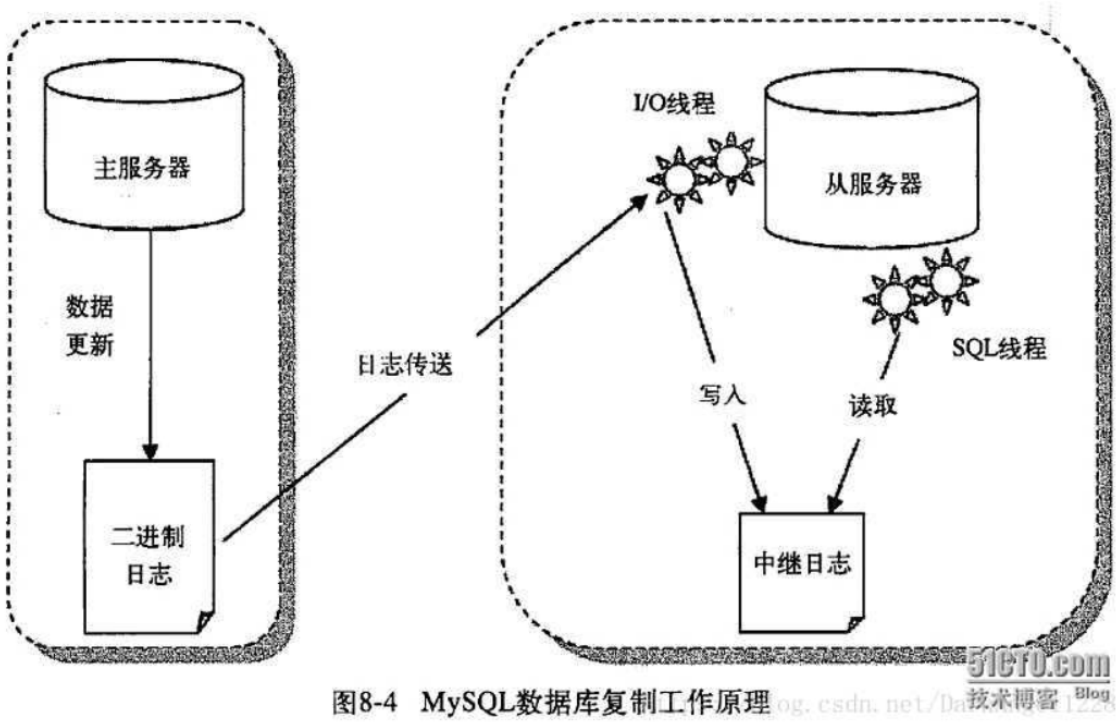


图8-4　MySQL数据库复制工作原理

步骤一：主库db的更新事件(update、insert、delete)被写到binlog
步骤二：从库发起连接，连接到主库
步骤三：此时主库创建一个binlog dump thread线程，把binlog的内容发送到从库
步骤四：从库启动之后，创建一个I/O线程，读取主库传过来的binlog内容并写入到relay log.
步骤五：还会创建一个SQL线程，从relay log里面读取内容，将更新内容写入到slave的db.

面试：
1.主从复制延迟大比较慢原因：
主服务器配置高，从服务器的配置低。
并发量大导致主服务器读的慢。从服务器写的慢
网络延迟比较高
从服务器的读写速度慢
2.从数据库的读的延迟问题了解吗？如何解决？
解决方法：
半同步复制–解决数据丢失的问题
并行复制--解决从库复制延迟的问题

三、M-S 架构GTID 基于事务ID复制

1、什么是GTID?

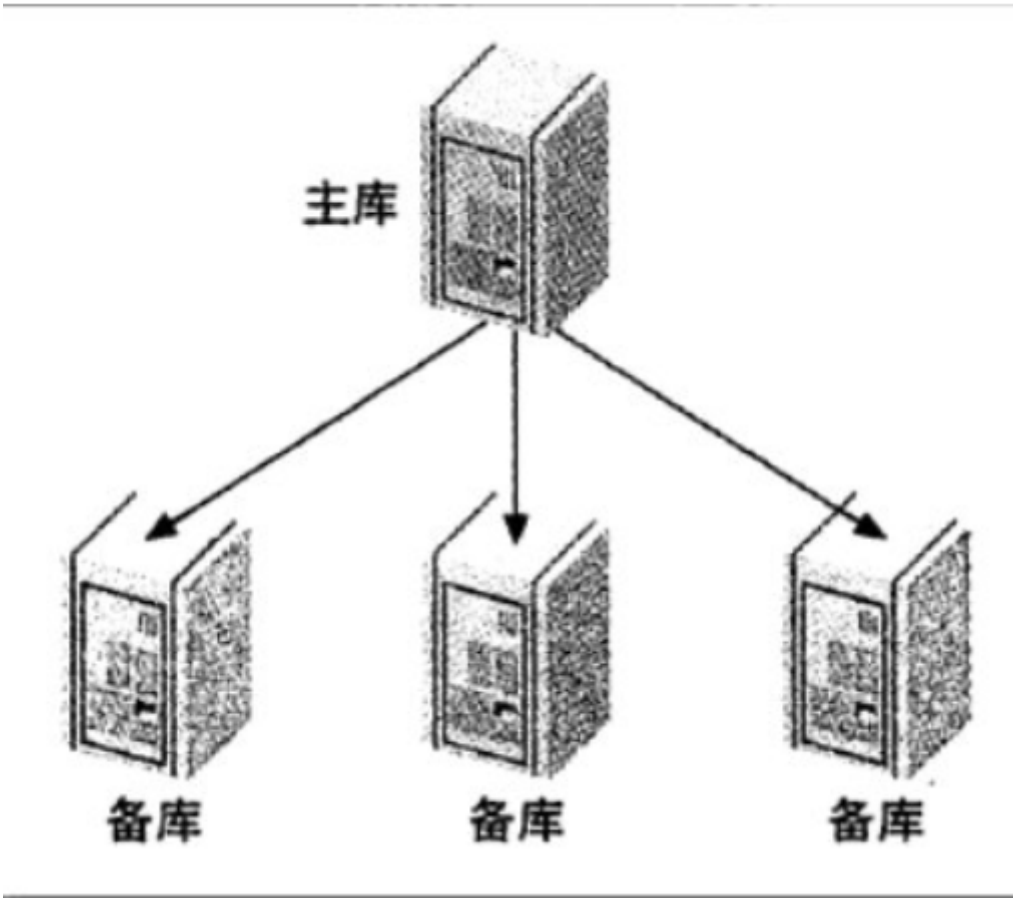
全局事务标识：global transaction identifiers 是用来代替传统复制的方法，GTID复制与普通复制模式的最大不同就是不需要指定二进制文件名和位置。

2、GTID工作原理

```
1、master更新数据时，会在事务前产生GTID，一同记录到binlog日志中。
2、slave端的i/o 线程将变更的binlog，写入到本地的relay log中。
3、sql线程从relay log中获取GTID，然后对比slave端的binlog是否有记录。
4、如果有记录，说明该GTID的事务已经执行，slave会忽略。
5、如果没有记录，slave就会从relay log中执行该GTID的事务，并记录到binlog。
```

3、部署主从复制

1、架构：



2、准备环境两台机器，关闭防火墙和selinux。---两台机器环境必须一致。时间也得一致

192.168.246.129 mysql-master

192.168.246.128 mysql-slave

两台机器安装mysql5.7

```
[root@mysql-master ~]# wget https://dev.mysql.com/get/mysql80-community-release-el7-3.noarch.rpm
安装略...
[root@mysql-master ~]# systemctl start mysqld
[root@mysql-master ~]# systemctl enable mysqld
[root@mysql-master ~]# netstat -lntp | grep 3306
tcp6       0      0 :::3306                 :::*                    LISTEN
 11669/mysqld
[root@mysql-slave ~]# netstat -lntp | grep 3306
tcp6       0      0 :::3306                 :::*                    LISTEN
 11804/mysqld
```

配置并修改密码

略....

master操作：

```
[root@mysql-master ~]# vim /etc/my.cnf   #在[mysqld]下添加如下内容
server-id=1   #定义server id master必写
log-bin = mylog #开启binlog日志，master比写
gtid_mode = ON    #开启gtid
enforce_gtid_consistency=1   #强制gtid
[root@mysql-master ~]# systemctl restart mysqld   #重启
```

主服务器创建账户：

```
mysql> grant replication  slave,reload,super on *.*  to 'slave'@'%' identified by 'Qf@12345! ';
#注:生产环境中密码采用高级别的密码，实际生产环境中将'%'换成slave的ip
mysql> flush privileges;
```

注意:如果不成功删除以前的binlog日志

replication slave：拥有此权限可以查看从服务器，从主服务器读取二进制日志。

super权限：允许用户使用修改全局变量的SET语句以及CHANGE MASTER语句

reload权限：必须拥有reload权限，才可以执行flush [tables | logs | privileges]

slave操作：

```
[root@mysql-slave ~]# vim /etc/my.cnf  #添加如下配置
server-id=2
gtid_mode = ON
enforce_gtid_consistency=1
master-info-repository=TABLE
relay-log-info-repository=TABLE
[root@mysql-slave ~]# systemctl restart mysqld
[root@mysql-slave ~]# mysql -uroot -p'qf123'   #登陆mysql
mysql> \e
change master to
master_host='master1',      #主ip 地址   最好用域名
master_user='授权用户',     #主服务上面创建的用户
master_password='授权密码',
master_auto_position=1;
-> ;
Query OK, 0 rows affected, 2 warnings (0.02 sec)
```

```
change master to
master_host='192.168.246.129',
master_user='slave',
master_password='123',
master_auto_position=1;
~
```

```
mysql> start slave;   #启动slave角色
Query OK, 0 rows affected (0.00 sec)

mysql> show slave status\G  #查看状态，验证sql和IO是不是yes。
```

```
mysql> show slave status\G
*************************** 1. row ***************************
               Slave_IO_State: Waiting for master to send event
                  Master_Host: 192.168.246.129
                  Master_User: slave
                  Master_Port: 3306
                Connect_Retry: 60
              Master_Log_File: mylog.000003
          Read_Master_Log_Pos: 154
               Relay_Log_File: mysql-slave-relay-bin.000002
                Relay_Log_Pos: 359
        Relay_Master_Log_File: mylog.000003
             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:
                   Last_Errno: 0
```

测试

```
[root@mysql-master ~]# mysql -uroot -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 3
Server version: 5.7.27-log MySQL Community Server (GPL)

Copyright (c) 2000, 2019, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

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

mysql> create table test.t1(id int);
Query OK, 0 rows affected (0.02 sec)

mysql> insert into test.t1 values (1);
Query OK, 1 row affected (0.03 sec)

mysql> select  * from test.t1;
+------+
| id   |
+------+
|    1 |
+------+
1 row in set (0.01 sec)

mysql>
```

在slave上面查看一下有没有同步过去：

```
[root@mysql-slave ~]# mysql -uroot -p'qf123'
mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| sys                |
| test               |
+--------------------+
mysql> use test
mysql> show tables;
+----------------+
| Tables_in_test |
+----------------+
| t1             |
+----------------+
mysql> select * from t1;
+------+
| id   |
+------+
|    1 |
+------+
```

主从同步完成。

注意: 在关闭和启动mysql服务的时候按顺序先启动master。

面试题

```
mysql主从，master宕机，如何进行切换？
主机故障或者宕机：

1)在salve执行：
mysql> stop slave;
mysql> reset master;
2)查看是否只读模式：show variables like 'read_only';
只读模式需要修改my.cnf文件，注释read-only=1并重启mysql服务。
或者不重启使用命令关闭只读，但下次重启后失效：set global read_only=off;
3)查看show slave status \G;
4)在程序中将原来主库IP地址改为现在的从库IP地址，测试应用连接是否正常
```

四、主从复制binlog日志方式

192.168.246.135 mysql-master

192.168.246.136 mysql-slave

准备两台机器，关闭防火墙和selinux。---两台机器环境必须一致。时间也得一致

```
两台机器配置hosts解析
192.168.246.135 mysql-master
192.168.246.136 mysql-slave
两台机器安装mysql
# wget https://dev.mysql.com/get/mysql80-community-release-el7-3.noarch.rpm
略...
[root@mysql-master ~]# systemctl start mysqld
[root@mysql-master ~]# systemctl enable mysqld
```

开始配置主服务

1、在主服务器上，必须启用二进制日志记录并配置唯一的服务器ID。需要重启服务器。

编辑主服务器的配置文件 `my.cnf`，添加如下内容

```
添加配置
[mysqld]
log-bin=/var/log/mysql/mysql-bin
server-id=1
```

创建日志目录并赋予权限

```
[root@mysql-master ~]# mkdir /var/log/mysql
[root@mysql-master ~]# chown mysql.mysql /var/log/mysql
```

重启服务

```
[root@mysql-master ~]# systemctl restart mysqld
```

查找密码

```
[root@mysql-master ~]# grep pass /var/log/mysqld.log
```

```
2019-08-23T02:43:23.291577Z 0 [Note] Shutting down plugin 'sha256_password'
2019-08-23T02:43:23.291579Z 0 [Note] Shutting down plugin 'mysql_native_password'
2019-08-23T02:43:23.944693Z 1 [Note] A temporary password is generated for root@localhost: Ns0_3jgPIM*5
2019-08-23T02:43:28.329460Z 0 [Note] Shutting down plugin 'validate_password'
2019-08-23T02:43:29.141438Z 0 [Note] Shutting down plugin 'sha256_password'
2019-08-23T02:43:29.141440Z 0 [Note] Shutting down plugin 'mysql_native_password'
2019-08-23T02:44:04.850676Z 0 [Note] Shutting down plugin 'validate_password'
2019-08-23T02:44:06.374201Z 0 [Note] Shutting down plugin 'sha256_password'
2019-08-23T02:44:06.374204Z 0 [Note] Shutting down plugin 'mysql_native_password'
2019-08-23T02:44:06.696018Z 0 [Note] Plugin 'validate_password' is disabled.
2019-08-23T02:45:21.578289Z 3 [Note] Access denied for user 'root'@'localhost' (using password: YES)
[root@mysql-master ~]#
```

修改密码

```
[root@mysql-master ~]# mysqladmin -uroot -p'Ns0_3jgPIM*5' password 'Qf@12345!'
创建主从同步的用户：
mysql> GRANT REPLICATION SLAVE ON *.*  TO  'repl'@'%'  identified by
'Qf@12345!';
mysql> flush privileges;
```

在主服务器上面操作

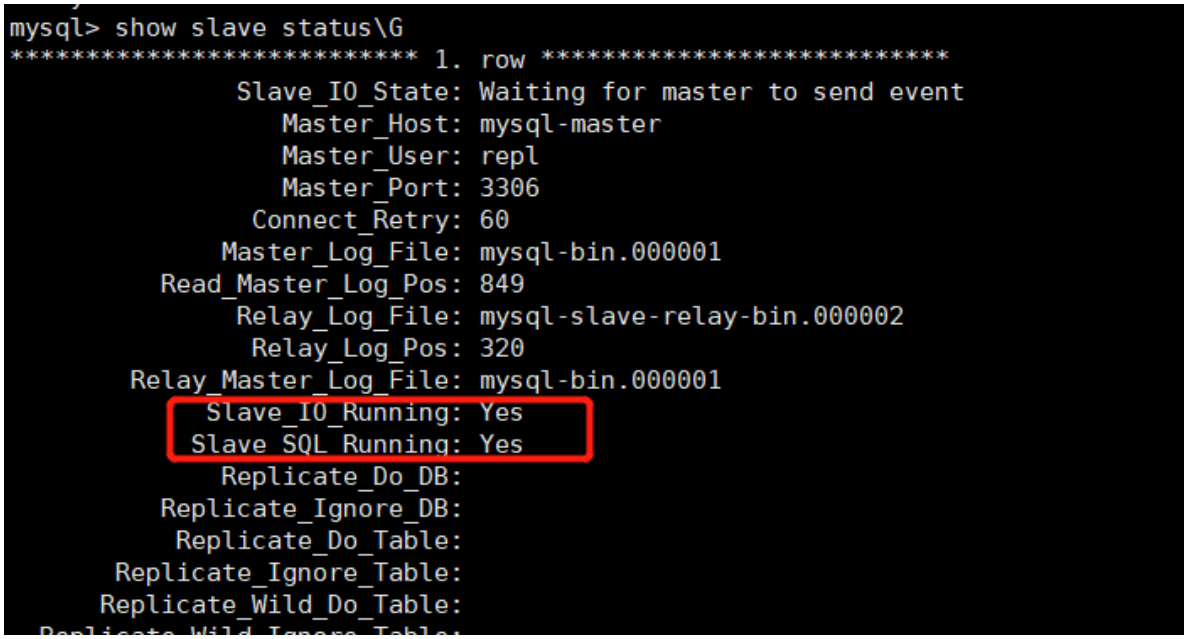
```
mysql> show master status \G
*************************** 1. row ***************************
             File: mysql-bin.000001
         Position: 849
     Binlog_Do_DB:
 Binlog_Ignore_DB:
Executed_Gtid_Set:
1 row in set (0.00 sec)

mysql>
```

在从服务上面操作：

`my.cnf`配置文件

```
[mysqld]
server-id=2
重启服务
[root@mysql-slave ~]# systemctl restart mysqld
设置密码
[root@mysql-slave ~]# grep pass /var/log/mysqld.log
[root@mysql-slave ~]# mysqladmin -uroot -p'ofeUcgA)4/Yg' password 'Qf@12345!'
登录mysql
[root@mysql-slave ~]# mysql -uroot -p'Qf@12345!'
mysql> \e
CHANGE MASTER TO
MASTER_HOST='mysql-master',
MASTER_USER='repl',
MASTER_PASSWORD='Qf@12345!',
MASTER_LOG_FILE='mysql-bin.000001',
MASTER_LOG_POS=849;
    -> ;
mysql> start slave;
mysql> show slave status\G
```




```
参数解释：
CHANGE MASTER TO
MASTER_HOST='master2.example.com',       //主服务器ip
MASTER_USER='replication',                          //主服务器用户
MASTER_PASSWORD='password',                 //用户密码
MASTER_PORT=3306,                                      //端口
MASTER_LOG_FILE='master2-bin.001',       //binlog日志文件名称
MASTER_LOG_POS=4,                 //日志位置
```

在master上面执行：

mysql> create database testdb; #创建一个库 Query OK, 1 row affected (0.10 sec)

mysql> \q

故障排错

```
#### UUID一致，导致主从复制I/O线程不是yes

> Fatal error: The slave I/O thread stops because master and slave have equal
MySQL server UUIDs; these UUIDs must be different for replication to work

致命错误：由于master和slave具有相同的mysql服务器uuid，导致I/O线程不进行；这些uuid必须不同才能使复制工作。

问题提示主从使用了相同的server UUID，一个个的检查：

检查主从server_id

主库：

mysql>  show variables like 'server_id';
+---------------+-------+
| Variable_name | Value |
+---------------+-------+
| server_id     | 1     |
+---------------+-------+
1 row in set (0.01 sec)
```

从库：

```
mysql>  show variables like 'server_id';
+---------------+-------+
| Variable_name | Value |
+---------------+-------+
| server_id     | 2     |
+---------------+-------+
1 row in set (0.01 sec)
```

server_id不一样，排除。

检查主从状态：

主库：

```
mysql> show master status;
+------------------+----------+--------------+------------------+-------------------
----+
| File             | Position | Binlog_Do_DB | Binlog_Ignore_DB |
Executed_Gtid_Set |
+------------------+----------+--------------+------------------+-------------------
----+
| mysql-bin.000001 |      154 |              |                  |
    |
+------------------+----------+--------------+------------------+-------------------
----+
1 row in set (0.00 sec)
```

从库：

```
mysql> show master status;
+------------------+----------+--------------+------------------+-------------------
----+
| File             | Position | Binlog_Do_DB | Binlog_Ignore_DB |
Executed_Gtid_Set |
+------------------+----------+--------------+------------------+-------------------
----+
| mysql-bin.000001 |      306 |              |                  |
    |
+------------------+----------+--------------+------------------+-------------------
----+
1 row in set (0.00 sec)
```

File一样，排除。

最后检查发现他们的auto.cnf中的server-uuid是一样的。。。

```
[root@localhost ~]# vim /var/lib/mysql/auto.cnf

[auto]

server-uuid=4f37a731-9b79-11e8-8013-000c29f0700f
```

修改uuid并重启服务

九、mysql优化

引擎：
查看引擎：
```
mysql> show engines;
mysql> SHOW VARIABLES LIKE '%storage_engine%';
mysql> show create table t1;
mysql> show table status like 't1';
```

临时指定引擎：
```
mysql> create table innodb1(id int)engine=innodb;
```
修改默认引擎：
```
/etc/my.cnf
[mysqld]
default-storage-engine=INNODB  ----引擎
```
修改已经存在的表的引擎：
```
mysql> alter table t2 engine=myisam;
```

优化：

调优思路：

1.数据库设计与规划--以后再修该很麻烦，估计数据量，使用什么存储引擎
2.数据的应用--怎样取数据，sql语句的优化
3.mysql服务优化--内存的使用，磁盘的使用
4.操作系统的优化--内核、tcp连接数量
5.升级硬件设备

只对innodb引擎

物理分区：

```
#mkdir /data
#chown mysql.mysql /data
mysql> create table t1(id int,name char(20)) data directory='/data/';----指定
```
存储表的内容目录

表结构存储的文件是.frm结尾的文件。不能指定。

如果创建不成功给data目录增加权限。chown mysql.mysql data

mysql服务优化

```
mysql> show status          看系统的资源      ---各种状态参数。
mysql> show variables     看变量，在my.cnf配置文件里定义的
mysql> show warnings     查看最近一个sql语句产生的错误警告，看其他的需要看.err日志
mysql> show processlist 显示系统中正在运行的所有进程。
mysql> show errors
```

字符集设置

临时：

```
mysql> create database db1 CHARACTER SET = utf8;
mysql> create table t1(id int(10)) CHARACTER SET = utf8;
```

5.7/ 5.5版本设置：

```
[mysqld]
character_set_server = utf8
```

==========

慢查询：

查看是否设置成功：

```
mysql> show variables like '%query%';
```

```
当连接数的数值过小会经常出现ERROR 1040: Too many connections错误。
这是是查询数据库当前设置的最大连接数
mysql> show variables like '%max_connections%';

这是查看超时时间
mysql> show global variables like '%timeout%';

强制限制mysql资源设置:
vim /etc/my.cnf
max_connections = 1024   并发连接数，根据实际情况设置连接数。
connect_timeout= 5    单位秒 ----超时时间，默认30秒
```

```
innodb引擎:
   innodb-buffer-pool-size    //缓存 InnoDB 数据和索引的内存缓冲区的大小
innodb-buffer-pool-size=#    ----值
        这个值设得越高,访问表中数据需要得磁盘 I/O 越少。在一个专用的数据库服务器上,你可以设置这个参数达机器物理内存大小的 80%。
vim /etc/my.cnf
innodb-buffer-pool-size=2G
wait_timeout=10 终止空闲时间超过10秒的链接，避免长连接
max_connect_errors=10 //10次连接失败就锁定，使用flush hosts 解锁，

mysql> flush hosts;      #解锁

#锁表
锁定数据表，避免在备份过程中，表被更新
mysql>LOCK TABLES tbl_name READ;
#为表增加一个写锁定:
mysql>LOCK TABLES tbl_name WRITE;

#查询是否锁表
mysql> show open tables where in_use >0;
#解锁
mysql> UNLOCK TABLES;
```

作业

一主双从（gtid方式）：

在准备一台虚拟机做为slave2，关闭防火墙和selinux

192.168.246.130 #mysql-slave2

```
1.由于刚才测试已经有了数据，需要先将master服务上面的数据备份出来。导入slave2的mysql中。这样才能保证集群中的机器环境一致。
在master操作:
[root@mysql-master ~]# mysqldump -uroot -p'qf123' --set-gtid-purged=OFF test > test.sql
[root@mysql-master ~]# ls
test.sql
[root@mysql-master ~]# scp test.sql 192.168.246.130:/root/   #拷贝到slave2
```

开启 GTID 后的导出导入数据的注意点

> Warning: A partial dump from a server that has GTIDs will by default include the GTIDs of all transactions, even those that changed suppressed parts of the database. If you don't want to restore GTIDs, pass --set-gtid-purged=OFF. To make a complete dump, pass --all-databases --triggers --routines --events

意思是： 当前数据库实例中开启了 GTID 功能，在开启有 GTID 功能的数据库实例中，导出其中任何一个库，如果没有显示地指定--set-gtid-purged参数，都会提示这一行信息. 意思是默认情况下，导出的库中含有 GTID 信息，如果不想导出包含有 GTID 信息的数据库，需要显示地添加--set-gtid-purged=OFF参数.

```
mysqldump -uroot  -p  --set-gtid-purged=OFF   --all-databases > alldb.db
```

导入数据是就可以相往常一样导入了。

slave2操作：
安装mysql5.7

```
[root@mysql-slave2 ~]# wget https://dev.mysql.com/get/mysql80-community-release-el7-3.noarch.rpm
```

安装略...

```
[root@mysql-slave2 ~]# systemctl start mysqld
[root@mysql-slave2 ~]# systemctl enable mysqld
```

登陆mysql创建一个test的库

```
[root@mysql-slave2 ~]# mysql -uroot -p'Qf@12345!'
mysql> create database test;
[root@mysql-slave2 ~]# mysql -uroot -p'Qf@12345!'  test < test.sql #将数据导入。
```

```
[root@mysql-slave2 ~]# mysql -uroot -p'Qf@12345!'
mysql: [Warning] Using a password on the command line interface can be insecure.
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 11
Server version: 5.7.27 MySQL Community Server (GPL)

Copyright (c) 2000, 2019, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> select * from test.t1;
+------+
| id   |
+------+
|    1 |
+------+
1 row in set (0.00 sec)
```

开始配置slave2

```
[root@mysql-slave2 ~]# vim /etc/my.cnf
server-id=3    #每台机器的id不一样
gtid_mode = ON
enforce_gtid_consistency=1
master-info-repository=TABLE
relay-log-info-repository=TABLE
```

```
[root@mysql-slave2 ~]# systemctl restart mysqld
[root@mysql-slave2 ~]# mysql -uroot -p'Qf@12345!'
mysql> \e
change master to
master_host='192.168.246.129',
master_user='slave',
master_password='123',
master_auto_position=1;
    -> ;
mysql> start slave;  #将slave启动起来
mysql> show slave status\G  #查看一下状态
```

```
mysql> show slave status\G
*************************** 1. row ***************************
               Slave_IO_State: Waiting for master to send event
                  Master_Host: 192.168.246.129
                  Master_User: slave
                  Master_Port: 3306
                Connect_Retry: 60
              Master_Log_File: mylog.000003
          Read_Master_Log_Pos: 1981
               Relay_Log_File: mysql-slave2-relay-bin.000002
                Relay_Log_Pos: 402
        Relay_Master_Log_File: mylog.000003
             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:
                   Last_Errno: 0
                   Last_Error:
```

测试：

在master上面在创建一个库：

```
[root@mysql-master ~]# mysql -uroot -p'qf123'
mysql> create database qfedu;
mysql> create table qfedu.t1(id int);
mysql> insert into qfedu.t1 values (1);
mysql>
```

两台slave

```
[root@mysql-slave ~]# mysql -uroot -p'qf123'
mysql: [Warning] Using a password on the command line interface can be insecure.
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 8
Server version: 5.7.27 MySQL Community Server (GPL)

Copyright (c) 2000, 2019, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> select * from qfedu.t1;
+------+
| id   |
+------+
|    1 |
+------+
1 row in set (0.00 sec)

mysql>
```

```
[root@mysql-slave2 ~]# mysql -uroot -p'Qf@12345!'
mysql: [Warning] Using a password on the command line interface can be insecure.
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 5
Server version: 5.7.27 MySQL Community Server (GPL)

Copyright (c) 2000, 2019, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> select * from qfedu.t1;
+------+
| id   |
+------+
|    1 |
+------+
1 row in set (0.00 sec)

mysql>
```

主从同步完成!

注意: 在关闭和启动mysql服务的时候按顺序先启动master。

十、读写分离

1.什么是读写分离

- **在数据库集群架构中，让主库负责处理写入操作，而从库只负责处理select查询，让两者分工明确达到提高数据库整体读写性能。**当然，主数据库另外一个功能就是负责将数据变更同步到从库中，也就是写操作。

2. 读写分离的好处

```
1. 分摊服务器压力，提高机器的系统处理效率
2. 在写入不变，大大分摊了读取，提高了系统性能。另外，当读取被分摊后，又间接提高了写入的性能。所以，总体性能提高了。
3. 增加冗余，提高服务可用性，当一台数据库服务器宕机后可以调整另外一台从库以最快速度恢复服务
```

Mycat 数据库中间件

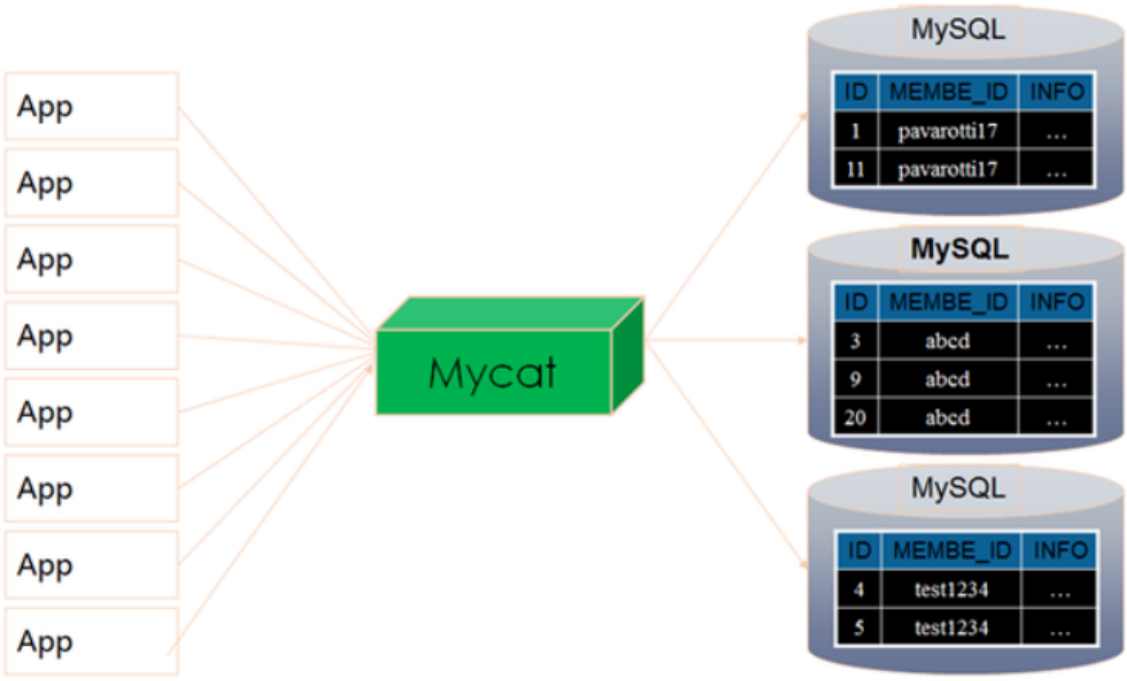
Mycat 是一个开源的数据库系统，但是由于真正的数据库需要存储引擎，而 Mycat 并没有存 储引擎，所以并不是完全意义的数据库系统。 那么 Mycat 是什么？**Mycat 是数据库中间件，就是介于数据库与应用之间，进行数据处理与交互的中间服 务是实现对主从数据库的读写分离、读的负载均衡。**

常见的数据库中间件:

MyCAT 是使用 JAVA 语言进行编写开发，使用前需要先安装 JAVA 运行环境(JRE),由于 MyCAT 中使用了 JDK7 中的一些特性，所以要求必须在 JDK7 以上的版本上运行。

```
准备一台新的主机放到master的前面做代理
192.168.246.133  mysql-mycat
并将三台机器互做本地解析
```

读写分离过程：



部署环境：

安装jdk

```
下载jdk账号：
账号：liwei@xiaostudy.com
密码：OracleTest1234
将jdk上传到服务器中，
[root@mycat ~]# tar xzf jdk-8u221-linux-x64.tar.gz -C /usr/local/
[root@mycat ~]# cd /usr/local/
[root@mycat local]# mv jdk1.8.0_221/ java
设置环境变量
[root@mycat local]# vim /etc/profile  #添加如下内容，
JAVA_HOME=/usr/local/java
PATH=$JAVA_HOME/bin:$PATH
export CLASSPATH=.:$JAVA_HOME/lib/dt.jar:$JAVA_HOME/lib/tools.jar
[root@mycat local]# source /etc/profile
```

部署mycat

```
下载
[root@mycat ~]# wget http://dl.mycat.io/1.6.5/Mycat-server-1.6.5-release-20180122220033-linux.tar.gz
解压
[root@mycat ~]# tar xf Mycat-server-1.6.5-release-20180122220033-linux.tar.gz -C /usr/local/
```

配置mycat

```
认识配置文件

MyCAT 目前主要通过配置文件的方式来定义逻辑库和相关配置:

/usr/local/mycat/conf/server.xml  #定义用户以及系统相关变量，如端口等。其中用户信息是前端应用程序连接 mycat 的用户信息。
/usr/local/mycat/conf/schema.xml  #定义逻辑库，表、分片节点等内容。
```

配置server.xml

以下为代码片段

下面的用户和密码是应用程序连接到 MyCat 使用的，可以自定义配置

而其中的schemas 配置项所对应的值是逻辑数据库的名字，也可以自定义，但是这个名字需要和后面 schema.xml 文件中配置的一致。

```
[root@mycat ~]# cd /usr/local/mycat/conf/
[root@mycat conf]# vim server.xml
...
  <!--下面的用户和密码是应用程序连接到 MyCat 使用的.schemas 配置项所对应的值是逻辑数据库的名字,这个名字需要和后面 schema.xml 文件中配置的一致。-->

        <user name="root" defaultAccount="true">
                <property name="password">Qf@12345!</property>
                <property name="schemas">testdb</property>

                <!-- 表级 DML 权限设置 -->
                <!--
                <privileges check="false">
                        <schema name="TESTDB" dml="0110" >
                                <table name="tb01" dml="0000"></table>
                                <table name="tb02" dml="1111"></table>
                        </schema>
                </privileges>
                 -->
        </user>
        <!--
<!--下面是另一个用户，并且设置的访问 TESTED 逻辑数据库的权限是 只读。可以注释掉
        <user name="user">
                <property name="password">user</property>
                <property name="schemas">TESTDB</property>
                <property name="readOnly">true</property>
        </user>
         -->
</mycat:server>
```

== 上面的配置中，假如配置了用户访问的逻辑库，那么必须在 `schema.xml` 文件中也配置这个逻辑库，否则报错，启动 mycat 失败 ==

配置schema.xml

以下是配置文件中的每个部分的配置块儿

逻辑库和分表设置

```
<schema name="testdb"            // 逻辑库名称,与server.xml的一致
        checkSQLschema="false"    // 不检查sql
        sqlMaxLimit="100"          // 最大连接数
        dataNode="dn1">         //  数据节点名称
<!--这里定义的是分表的信息-->
</schema>
```

数据节点

```
<dataNode name="dn1"              // 此数据节点的名称
          dataHost="localhost1"      // 主机组虚拟的
          database="testdb" />  // 真实的数据库名称
```

主机组

```
<dataHost name="localhost1"                          // 主机组
          maxCon="1000" minCon="10"                  // 连接
          balance="0"                                // 负载均衡
          writeType="0"                              // 写模式配置
          dbType="mysql" dbDriver="native"           // 数据库配置
          switchType="1"  slaveThreshold="100">
<!--这里可以配置关于这个主机组的成员信息，和针对这些主机的健康检查语句-->
</dataHost>
```

```
balance 属性
负载均衡类型,目前的取值有 3 种:
1. balance="0", 不开启读写分离机制,所有读操作都发送到当前可用的 writeHost 上。
2. balance="1", 全部的 readHost 与  writeHost 参与 select 语句的负载均衡,简单的说,当双主双从模式(M1->S1,M2->S2,并且 M1 与 M2 互为主备),正常情况下,M2,S1,S2 都参与 select 语句的负载均衡。
3. balance="2", 所有读操作都随机的在 writeHost、readhost 上分发。
4. balance="3", 所有读请求随机的分发到 wiriterHost 对应的 readhost 执行,writerHost 不负担读压力,注意 balance=3 只在 1.4 及其以后版本有,1.3 没有。

writeType 属性
负载均衡类型
1. writeType="0", 所有写操作发送到配置的第一个 writeHost,第一个挂了切换到还生存的第二个 writeHost,重新启动后已切换后的为准.
2. writeType="1",所有写操作都随机的发送到配置的 writeHost,#版本1.5 以后废弃不推荐。
```

健康检查

```
<heartbeat>select user()</heartbeat>    #对后端数据进行检测，执行一个sql语句，user()内部函数
```

读写配置

```
<writeHost host="hostM1" url="192.168.246.135:3306" user="mycat"
password="Qf@12345!">
                        <!-- can have multi read hosts -->
<readHost host="hostS2" url="192.168.246.136:3306" user="mycat"
password="Qf@12345!" />
</writeHost>
```

以下是组合为完整的配置文件，适用于一主一从的架构

```
[root@mycat ~]# cd /usr/local/mycat/conf/
[root@mycat conf]# cp schema.xml schema.xml.bak
[root@mycat conf]# vim schema.xml
<?xml version="1.0"?>
<!DOCTYPE mycat:schema SYSTEM "schema.dtd">
<mycat:schema xmlns:mycat="http://io.mycat/">

        <schema name="testdb" checkSQLschema="false" sqlMaxLimit="100"
dataNode="dn1">
        </schema>

        <dataNode name="dn1" dataHost="localhost1" database="testdb" />

        <dataHost name="localhost1" maxCon="1000" minCon="10" balance="0"
                          writeType="0" dbType="mysql" dbDriver="native"
switchType="1"  slaveThreshold="100">
                <heartbeat>select user()</heartbeat>
                <!-- can have multi write hosts -->
                <writeHost host="mysql-master" url="mysql-master:3306"
user="mycat" password="Qf@1234!">
                        <!-- can have multi read hosts -->
                 <readHost host="mysql-slave" url="mysql-slave:3306"
user="mycat" password="Qf@1234!" />
                </writeHost>
        </dataHost>
</mycat:schema>
```

在真实的 master 数据库上给用户授权

```
mysql> grant all on testdb.* to mycat@'%' identified by 'Qf@1234!';
mysql> flush privileges;
```

在mycat的机器上面测试mycat用户登录：

```
安装客户端：
# yum install -y mysql
```

```
[root@mycat ~]# mysql -uroot -hmysql-slave -u mycat -p'Qf@12345!'
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MySQL connection id is 12
Server version: 5.7.27 MySQL Community Server (GPL)

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MySQL [(none)]> use testdb;
Database changed
MySQL [testdb]>
```

启动Mycat

启动之前需要调整JVM

```
在wrapper.conf中添加
[root@mycat mycat]# cd conf/
[root@mycat conf]# vim wrapper.conf  #在设置JVM哪里添加如下内容
wrapper.startup.timeout=300 //超时时间300秒
wrapper.ping.timeout=120
启动:
[root@mycat conf]# /usr/local/mycat/bin/mycat start
Starting Mycat-server...
[root@mycat ~]# jps   #查看mycat是否启动
13377 WrapperSimpleApp
13431 Jps
[root@mycat ~]# netstat -lntp | grep java
```

```
[root@mycat ~]# netstat -lntp | grep java
tcp        0      0 127.0.0.1:32000         0.0.0.0:*               LISTEN      13377/java
tcp6       0      0 :::45163                :::*                    LISTEN      13377/java
tcp6       0      0 :::39979                :::*                    LISTEN      13377/java
tcp6       0      0 :::1984                 :::*                    LISTEN      13377/java
tcp6       0      0 :::8066                 :::*                    LISTEN      13377/java
tcp6       0      0 :::9066                 :::*                    LISTEN      13377/java
[root@mycat ~]#
```


测试mycat

```
将master当做mycat的客户端
[root@mysql-master ~]# mysql -uroot -h mysql-mycat -p'Qf@12345!' -P 8066
```

```
[root@mysql-master ~]# mysql -uroot -h mysql-mycat -p'Qf@12345!' -P 8066
mysql: [Warning] Using a password on the command line interface can be insecure.
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 6
Server version: 5.6.29-mycat-1.6.5-release-20180122220033 MyCat Server (OpenCloundDB)

Copyright (c) 2000, 2019, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> show databases;
+----------+
| DATABASE |
+----------+
| testdb   |
+----------+
1 row in set (0.00 sec)

mysql>
```

```
mysql> use testdb
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_testdb |
+------------------+
| t1               |
+------------------+
1 row in set (0.00 sec)

mysql> select * from t1;
Empty set (0.14 sec)
```

```
mysql> insert into t1(id) values (1);
Query OK, 1 row affected (0.37 sec)

mysql> select * from t1;
+------+
| id   |
+------+
|    1 |
+------+
1 row in set (0.00 sec)

mysql>
```

如果在show table报错：

mysql> show tables; ERROR 3009 (HY000): java.lang.IllegalArgumentException: Invalid DataSource:0

解决方式：
登录master服务将mycat的登录修改为%

```
mysql> update user set Host = '%' where User = 'mycat' and Host = 'localhost';
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> flush privileges;
```

或者在授权用户mycat权限为*.*