

第 1 章 MySQL 的架构介绍

第 1 章 MySQL 的架构介绍

1、MySQL 简介

什么是 MySQL?

1. 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公司。

2. MySQL是一种关联数据库管理系统，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。

3. Mysql是开源的，所以你不需支付额外的费用。

4. Mysl支持大型的数据库，可以处理拥有上千万条记录的大型数据库。

5. MySQL使用标准的SQL数据语言形式。

6. Mysql可以允许于多个系统上，并且支持多种语言，这些编程语言包括C、C++、Python、Java、Ped、PHP、Eifel、Ruby和TCL等。

7. Mysql对PHP有很好的支持，PHP是目前最流行的Web开发语言。

8. MySQL支持大型数据库，支持5000万条记录的数据仓库，32位系统表文件最大可支持4GB，64位系统支持最大的表文件为8TB。

9. Mysql是可以定制的，采用了GPL协议，你可以修改源码来开发自己的Mysql系统。

MySQL 高手是怎样练成的?

1. mysql内核

2. sql优化工程师

3. mysql服务器的优化

4. 查询语句优化

5. 主重复制

6. 软硬件升级

7. 容灾备份

8. sql编程

2、Linux 安装 MySQL

2.1、安装 MySQL

CentOS 6 通过 yum 安装 mysql 5.6

emmm... 老师课件给的安装方式失败了。。。我在网上找了些资料，终于安装成功啦~~~废了老半天力气。。。

1. <https://www.cnblogs.com/micfox/articles/10989905.html>

2. <https://blog.csdn.net/MissDreamY/article/details/104938194>

检查当前系统是否安装过 mysql

• 检查系统是否安装其他版本的 MySQL 数据库

1 | yum list installed | grep mysql

2 | mysql-libs.x86_64 5.1.73-5.el6_6 @anaconda-CentOS-201508042137.x86_64/6.7

• 有的话干掉老版本数据库

1 | yum -y remove mysql-libs.x86_64

安装及配置

• 拉取 rpm 包

```
1 | wget http://repo.mysql.com/mysql-community-release-el6-5.noarch.rpm
```

- 通过 rpm 安装 mysql

```
1 | rpm -ivh mysql-community-release-el6-5.noarch.rpm
```

- 列举可用的 mysql 版本

```
1 | yum repolist all | grep mysql
```

- 安装 mysql-community-server 版本 (好像默认自带 mysql-community-client)

```
1 | yum install mysql-community-server -y
```

mysql 服务的启动与停止

- mysql 服务的启动

```
1 | service mysqld start
```

- mysql 服务的停止

```
1 | service mysqld stop
```

- 查看 mysql 服务的状态

```
1 | service mysql status
```

备注: 正在启动 mysqld: [失败] 问题

- 打下面这一套组合拳

```
1 | rm -fr /var/lib/mysql/*
2 | rm /var/lock/subsys/mysqld
3 | killall mysqld
```

- 然后启动 mysql 的服务

```
1 | service mysqld start
```

2.2、配置 MySQL

查看安装时创建的 mysql 用户和 mysql 组

- 查看 mysql 用户

```
1 | cat /etc/passwd | grep mysql
```

- 查看 mysql 用户组

```
1 | cat /etc/group | grep mysql
```

设置远程 root

- 启动 mysql 服务

```
1 | service mysqld start
```

- 设置 root 用户密码：这里面会有比较多的选项，按照提示操作即可

```
1 | mysql_secure_installation
```

- 登录至 mysql root 账号

```
1 | mysql -uroot -p
```

设置 mysql 开机启动

- 设置 mysql 为开机自启动

```
1 | chkconfig mysqld on
```

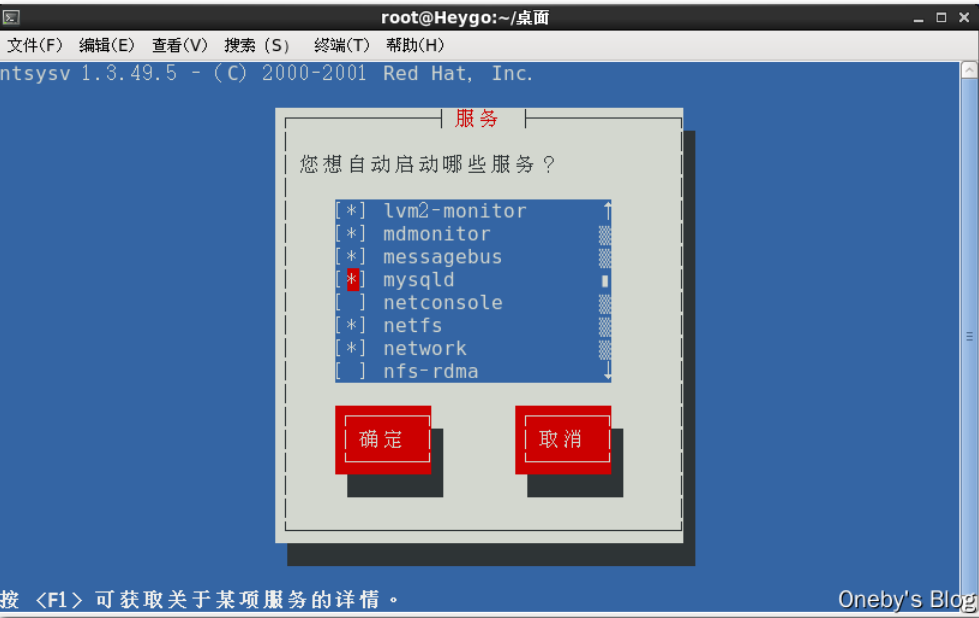
- 查看是否设置成功：2、3、4都是on代表开机自动启动

```
1 | chkconfig --list | grep mysqld
```

```
[root@Heygo 桌面]# chkconfig --list | grep mysqld
mysqld          0:关闭  1:关闭  2:启用  3:启用  4:启用  5:启用  6:关闭
[root@Heygo 桌面]# ~~
```

- 如果想要关闭 mysql 的开机自启动，输入如下命令，找到 mysqld ，单击空格取消 mysql 的开机自启动，Tab 键选中确定，敲击 Enter 键确定即可

```
1 | ntsysv
```



修改 mysql 配置文件位置

- 进入 /usr/share/mysql/ 目录，找到 my-default.cnf 配置文件

```
1 | cd /usr/share/mysql/
```

```
root@Heygo:/usr/share/mysql
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
[root@Heygo 桌面]# cd /usr/share/mysql/
[root@Heygo mysql]# ls -l
总用量 1624
drwxr-xr-x. 2 root root 4096 8月 3 19:05 bulgarian
drwxr-xr-x. 2 root root 4096 8月 3 19:05 charsets
drwxr-xr-x. 2 root root 4096 8月 3 19:05 czech
drwxr-xr-x. 2 root root 4096 8月 3 19:05 danish
-rw-r--r--. 1 root root 25575 6月 2 13:32 dictionary.txt
drwxr-xr-x. 2 root root 4096 8月 3 19:05 dutch
drwxr-xr-x. 2 root root 4096 8月 3 19:05 english
-rw-r--r--. 1 root root 506879 6月 2 13:32 errmsg-utf8.txt
drwxr-xr-x. 2 root root 4096 8月 3 19:05 estonian
-rw-r--r--. 1 root root 887806 6月 2 13:40 fill_help_tables.sql
drwxr-xr-x. 2 root root 4096 8月 3 19:05 french
drwxr-xr-x. 2 root root 4096 8月 3 19:05 german
drwxr-xr-x. 2 root root 4096 8月 3 19:05 greek
drwxr-xr-x. 2 root root 4096 8月 3 19:05 hungarian
-rw-r--r--. 1 root root 3963 6月 2 13:32 innodb_memcached_config.sql
drwxr-xr-x. 2 root root 4096 8月 3 19:05 italian
drwxr-xr-x. 2 root root 4096 8月 3 19:05 japanese
drwxr-xr-x. 2 root root 4096 8月 3 19:05 korean
-rw-r--r--. 1 root root 773 6月 2 13:32 magic
-rw-r--r--. 1 root root 1126 6月 2 14:08 my-default.cnf
-rw-r--r--. 1 root root 844 6月 2 14:08 mysql-log-rotate
-rw-r--r--. 1 root root 2121 6月 2 13:32 mysql_security_commands.sql
-rw-r--r--. 1 root root 3972 6月 2 13:32 mysql_system_tables_data.sql
-rw-r--r--. 1 root root 93667 6月 2 13:32 mysql_system_tables.sql
-rw-r--r--. 1 root root 10834 6月 2 13:32 mysql_test_data_timezone.sql
drwxr-xr-x. 2 root root 4096 8月 3 19:05 norwegian
```

- 将其拷贝至 /etc 目录下，重命名为 my.cnf，覆盖原有的配置文件

```
1 | cp /usr/share/mysql/my-default.cnf /etc/my.cnf
```

```
[root@Heygo mysql]# cp /usr/share/mysql/my-default.cnf /etc/my.cnf
cp: 是否覆盖"/etc/my.cnf" ? y
```

修改 mysql 字符集

查看 mysql 编码字符集

- 如果在建库建表的时候，没有明确指定字符集，则采用默认客户端和服务器的字符集都用了 latin1，其中是不包含中文字符的。
- 如何查看默认的编码字符集：

```
1 | show variables like '%char%';
2 | # 或者
3 | show variables like '%character%';
```

```
mysql> show variables like '%char%';
+-----+-----+
| Variable_name | Value |
+-----+-----+
| character_set_client | utf8 |
| character_set_connection | utf8 |
| character_set_database | latin1 |
| character_set_filesystem | binary |
| character_set_results | utf8 |
| character_set_server | latin1 |
| character_set_system | utf8 |
| character_sets_dir | /usr/share/mysql/charsets/ |
+-----+-----+
8 rows in set (0.00 sec)
```

修改配置文件

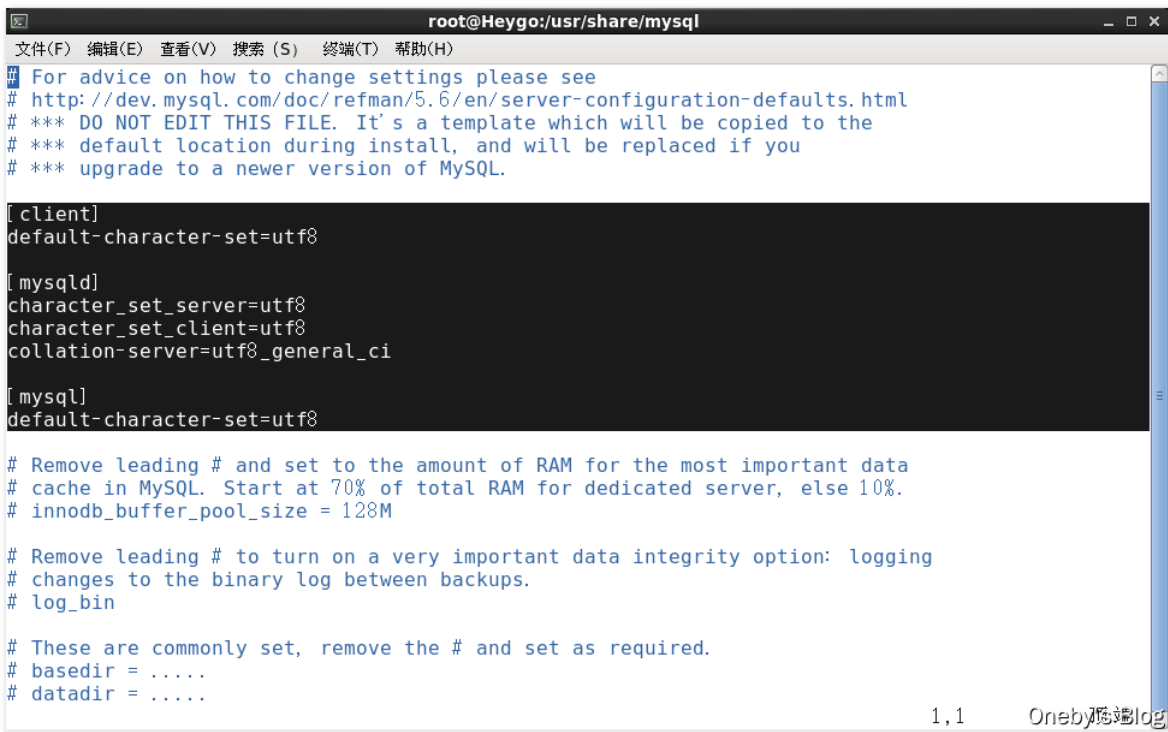
- 打开 /etc/my.cnf 配置文件，修改字符编码

```
1 | vim /etc/my.cnf
```

- 修改或添加如下内容

```
1 | [client]
2 | default-character-set=utf8
3 |
4 | [mysqld]
5 | character_set_server=utf8
6 | character_set_client=utf8
7 | collation-server=utf8_general_ci
8 |
9 | [mysql]
10 | default-character-set=utf8
```

- 好像比老师的配置文件少了很多内容???



```
root@Heygo:/usr/share/mysql
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
# For advice on how to change settings please see
# http://dev.mysql.com/doc/refman/5.6/en/server-configuration-defaults.html
# *** DO NOT EDIT THIS FILE. It's a template which will be copied to the
# *** default location during install, and will be replaced if you
# *** upgrade to a newer version of MySQL.

[client]
default-character-set=utf8

[mysqld]
character_set_server=utf8
character_set_client=utf8
collation-server=utf8_general_ci

[mysql]
default-character-set=utf8

# Remove leading # and set to the amount of RAM for the most important data
# cache in MySQL. Start at 70% of total RAM for dedicated server, else 10%.
# innodb_buffer_pool_size = 128M

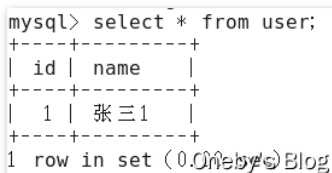
# Remove leading # to turn on a very important data integrity option: logging
# changes to the binary log between backups.
# log_bin

# These are commonly set, remove the # and set as required.
# basedir = .....
# datadir = .....
```

测试配置文件是否生效

- 建表测试：中文无乱码

```
1 | create database db01;
2 | create table user(id int not null, name varchar(20));
3 | insert into user values(1,'张三1');
4 | select * from user;
```



```
mysql> select * from user;
+----+-----+
| id | name  |
+----+-----+
| 1  | 张三1 |
+----+-----+
1 row in set (0.00 sec)
```

- 查看 mysql 全局变量

```
1 | show variables like '%char%';
```

```
mysql> show variables like '%char%';
+-----+-----+
| Variable_name | Value |
+-----+-----+
| character_set_client | utf8 |
| character_set_connection | utf8 |
| character_set_database | utf8 |
| character_set_filesystem | binary |
| character_set_results | utf8 |
| character_set_server | utf8 |
| character_set_system | utf8 |
| character_sets_dir | /usr/share/mysql/charsets/ |
+-----+-----+
8 rows in set (0.00 sec)
```

Oneby's Blog

mysql 的安装位置

- 执行命令

```
1 | ps -ef|grep mysql
```

```
[root@Heygo 桌面]# ps -ef|grep mysql
root      5302      1  0 19:27 ?                00:00:00 /bin/sh /usr/bin/mysqld_safe --datadir=/var/lib/mysql --socket=/var/lib/mysql/mysql.sock --pid-file=/var/run/mysqld/mysqld.pid --basedir=/usr --user=mysql
mysql     5506    5302  0 19:27 ?                00:00:01 /usr/sbin/mysqld --basedir=/usr --datadir=/var/lib/mysql --plugin-dir=/usr/lib64/mysql/plugin --user=mysql --log-error=/var/log/mysqld.log --pid-file=/var/run/mysqld/mysqld.pid --socket=/var/lib/mysql/mysql.sock
root      5670    5659  0 19:57 pts/0          00:00:00 grep mysql
[root@Heygo 桌面]#
```

Oneby's Blog

- 目录含义列举如下

路径	解释
/var/lib/mysql/	mysql 数据库文件的存放路径
/usr/share/mysql	配置文件目录
/usr/bin	相关命令目录
/etc/init.d/mysql	服务启停相关

3、MySQL 用户组管理

3.1、用户管理相关命令

创建用户

- 创建名称为 zhang3 的用户，密码设为 123123；

```
1 | create user zhang3 identified by '123123';
```

查看用和权限的相关信息

- 查看用和权限的相关信息的 SQL 指令

```
1 | select host, user, password, select_priv, insert_priv,drop_priv from mysql.user;
```

1. host :表示连接类型

1. % 表示所有远程通过 TCP 方式的连接
2. IP 地址 如 (192.168.1.2,127.0.0.1) 通过制定 ip 地址进行的 TCP 方式的连接
3. 机器名 通过制定 i 网络中的机器名进行的 TCP 方式的连接
4. ::1 IPv6 的本地 ip 地址 等同于 IPv4 的 127.0.0.1
5. localhost 本地方式通过命令行方式的连接，比如 `mysql -u xxx -p 123xxx` 方式的连接。

2. user:表示用户名

1. 同一用户通过不同方式链接的权限是不一样的。

3. password:密码

- 1. 所有密码串通过 password(明文字符串) 生成的密文字符串。加密算法为 MYSQLSHA1 , 不可逆。
- 2. mysql 5.7 的密码保存到 authentication_string 字段中不再使用 password 字段。

4. select_priv , insert_priv 等

- 1. 为该用户所拥有的权限。

```
1 | mysql> select host, user, password, select_priv, insert_priv,drop_priv from mysql.user;
2 | -----+-----+-----+-----+-----+-----+-----+
3 | | host      | user | password                                     | select_priv | insert_priv | drop_priv |
4 | +-----+-----+-----+-----+-----+-----+-----+
5 | | localhost | root | *81F5E21E35407D884A6CD4A731AEBFB6AF209E1B | Y           | Y           | Y           |
6 | | heygo     | root | *81F5E21E35407D884A6CD4A731AEBFB6AF209E1B | Y           | Y           | Y           |
7 | | 127.0.0.1 | root | *81F5E21E35407D884A6CD4A731AEBFB6AF209E1B | Y           | Y           | Y           |
8 | | ::1       | root | *81F5E21E35407D884A6CD4A731AEBFB6AF209E1B | Y           | Y           | Y           |
9 | +-----+-----+-----+-----+-----+-----+-----+
10| 4 rows in set (0.04 sec)
```

修改当前用户的密码

```
1 | set password =password('123456');
```

修改其他用户的密码

- 修改李四用户的密码

```
1 | update mysql.user set password=password('123456') where user='li4';
```

- 注意：所有通过 user 表的修改，必须用 flush privileges; 命令才能生效

修改用户名

- 将王五的用户名修改为李四

```
1 | update mysql.user set user='li4' where user='wang5';
```

- 注意：所有通过 user 表的修改，必须用 flush privileges; 命令才能生效

删除用户

- 删除李四用户

```
1 | drop user li4
```

- 注意：不要通过 delete from user u where user='li4' 进行删除，系统会有残留信息保留。

3.2、MySQL 的权限管理

授予权限

- 该权限如果发现没有该用户，则会直接新建一个用户。

```
1 | grant 权限 1,权限 2,...权限 n on 数据库名称.表名称 to 用户名@用户地址 identified by '连接口令'
```

- 示例：给 li4 用户用本地命令行方式下，授予 atguigudb 这个库下的所有表的插删改查的权限。

```
1 | grant select,insert,delete,drop on atguigudb.* to li4@localhost ;
```

- 授予通过网络方式登录的的joe 用户，对所有库所有表的全部权 限， 密码设为 123

```
1 | grant all privileges on *.* to joe@'%' identified by '123';
```

收回权限

- 查看当前用户权限：

```
1 | show grants;
```

- 收回权限命令

```
1 | revoke [权限 1,权限 2,...权限 n] on 库名.表名 from 用户名@用户地址;
```

- 收回全库全表的所有权限

```
1 | REVOKE ALL PRIVILEGES ON mysql.* FROM joe@localhost;
```

- 收回 mysql 库下的所有表的插删改查 权限

```
1 | REVOKE select,insert,update,delete ON mysql.* FROM joe@localhost;
```

- 注意：权限收回后， 必须用户重新登录后， 才能生效。

查看权限

1. 查看当前用户权限：`show grants;`
2. 查看所有用户权限：`select * from user ;`

4、MySQL 配置文件

二进制日志文件 log-bin

二进制日志文件 log-bin ： 用于主重复制

错误日志 log-error

默认是关闭的，记录严重的警告和错误信息，每次启动和关闭的详细信息等

查询日志 log

默认关闭，记录查询的sql语句，如果开启会减低mysql的整体性能，因为记录日志也是需要消耗系统资源的

数据文件

1. 数据库文件：默认路径：/var/lib/mysql
2. frm文件：存放表结构
3. myd文件：存放表数据
4. myi文件：存放表索引

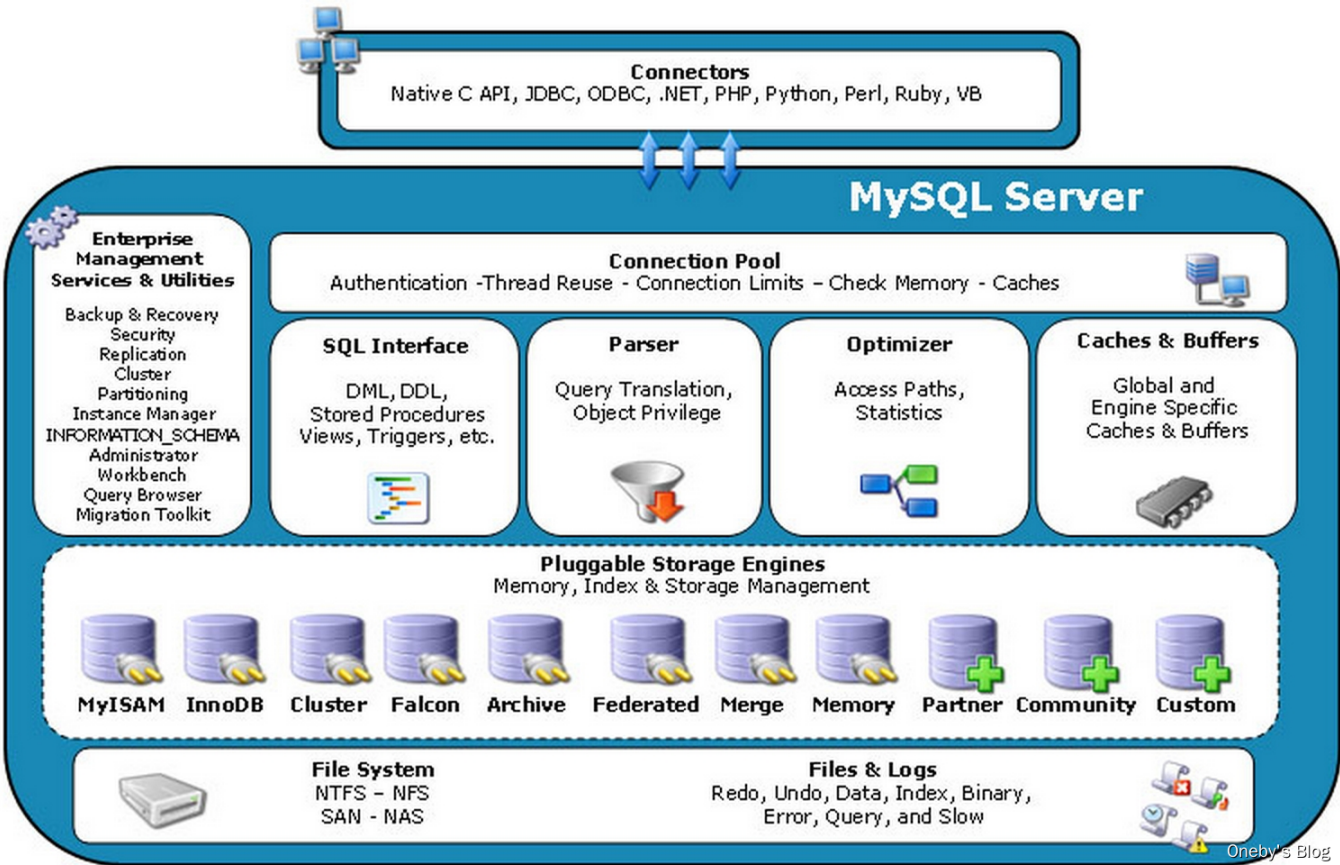
如何配置

1. Windows 系统下：my.ini文件
2. Linux 系统下：etc/my.cnf

5、MySQL 逻辑架构介绍

mysql 的分层思想

1. 和其它数据库相比，MySQL有点与众不同，它的架构可以在多种不同场景中应用并发挥良好作用。主要体现在存储引擎的架构上。
2. 插件式的存储引擎架构将查询处理和别的系统任务以及数据的存储提取相分离。这种架构可以根据业务的需求和实际需要选择合适的存储引擎。



mysql 四层架构

1. **连接层：**最上层是一些客户端和连接服务，包含本地sock通信和大多数基于客户端/服务端工具实现的类似于tcp/ip的通信。主要完成一些类似于连接处理、授权认证、及相关的安全方案。在该层上引入了线程池的概念，为通过认证安全接入的客户端提供线程。同样在该层上可以实现基于SSL的安全链接。服务器也会为安全接入的每个客户端验证它所具有的操作权限。
2. **服务层：**第二层架构主要完成大多数的核心服务功能，如SQL接口，并完成缓存的查询，SQL的分析和优化及部分内置函数的执行。所有跨存储引擎的功能也在这一层实现，如过程、函数等。在该层，服务器会解析查询并创建相应的内部解析树，并对其完成相应的优化如确定查询表的顺序，是否利用索引等，最后生成相应的执行操作。如果是select语句，服务器还会查询内部的缓存。如果缓存空间足够大，这样在解决大量读操作的环境中能够很好的提升系统的性能。

Management Services & Utilities	系统管理和控制工具
SQL Interface	SQL 接口。接受用户的 SQL 命令，并且返回用户需要查询的结果。比如 select from 就是调用 SQL Interface
Parser	解析器。SQL 命令传递到解析器的时候会被解析器验证和解析
Optimizer	查询优化器。SQL 语句在查询之前会使用查询优化器对查询进行优化，比如有 where 条件时，优化器来决定先投影还是先过滤。
Cache 和 Buffer	查询缓存。如果查询缓存有命中的查询结果，查询语句就可以直接去查询缓存中取数据。这个缓存机制是由一系列小缓存组成的。比如表缓存，记录缓存，key 缓存，权限缓存等

3. **引擎层：**存储引擎层，存储引擎真正的负责了MySQL中数据的存储和提取，服务器通过API与存储引擎进行通信。不同的存储引擎具有的功能不同，这样我们可以根据自己的实际需要进行选取。后面介绍MyISAM和InnoDB
4. **存储层：**数据存储层，主要是将数据存储在运行于裸设备的文件系统之上，并完成与存储引擎的交互。

MySQL 部件

1. **Connectors：**指的是不同语言中与SQL的交互

2. Management Serveices & Utilities：系统管理和控制工具

3. Connection Pool：连接池

1. 管理缓冲用户连接，线程处理等需要缓存的需求。负责监听对 MySQL Server 的各种请求，接收连接请求，转发所有连接请求到线程管理模块。
2. 每一个连接上 MySQL Server 的客户端请求都会被分配（或创建）一个连接线程为其单独服务。而连接线程的主要工作就是负责 MySQL Server 与客户端的通信。接受客户端的命令请求，传递 Server 端的结果信息等。线程管理模块则负责管理维护这些连接线程。包括线程的创建，线程的 cache 等。

4. SQL Interface：SQL接口。接受用户的SQL命令，并且返回用户需要查询的结果。比如select from就是调用SQL Interface

5. Parser：解析器

1. SQL命令传递到解析器的时候会被解析器验证和解析。解析器是由Lex和YACC实现的，是一个很长的脚本。
2. 在MySQL中我们习惯将所有 Client 端发送给 Server 端的命令都称为 Query，在 MySQL Server 里面，连接线程接收到客户端的一个 Query 后，会直接将该 Query 传递给专门负责将各种 Query 进行分类然后转发给各个对应的处理模块。
3. 解析器的主要功能：
 1. 将SQL语句进行语义和语法的分析，分解成数据结构，然后按照不同的操作类型进行分类，然后做出针对性的转发到后续步骤，以后SQL语句的传递和处理就是基于这个结构的。
 2. 如果在分解构成中遇到错误，那么就说明这个sql语句是不合理的

6. Optimizer：查询优化器

1. SQL语句在查询之前会使用查询优化器对查询进行优化。就是优化客户端发送过来的 sql 语句，根据客户端请求的 query 语句，和数据库中的一些统计信息，在一系列算法的基础上进行分析，得出一个最优的策略，告诉后面的程序如何取得这个 query 语句的结果
2. 他使用的是“选取-投影-联接”策略进行查询。
 1. 用一个例子就可以理解： select uid,name from user where gender = 1;
 2. 这个select 查询先根据where 语句进行选取，而不是先将表全部查询出来以后再进行gender过滤
 3. 这个select查询先根据uid和name进行属性投影，而不是将属性全部取出以后再进行过滤
 4. 将这两个查询条件联接起来生成最终查询结果

7. Cache和Buffer：查询缓存

1. 他的主要功能是将客户端提交 给MySQL 的 Select 类 query 请求的返回结果集 cache 到内存中，与该 query 的一个 hash 值 做一个对应。该 Query 所取数据的基表发生任何数据的变化之后， MySQL 会自动使该 query 的Cache 失效。在读写比例非常高的应用系统中， Query Cache 对性能的提高是非常显著的。当然它对内存的消耗也是非常大的。
2. 如果查询缓存有命中的查询结果，查询语句就可以直接去查询缓存中取数据。这个缓存机制是由一系列小缓存组成的。比如表缓存，记录缓存，key 缓存，权限缓存等

8. 存储引擎接口

1. 存储引擎接口模块可以说是 MySQL 数据库中最有特色的一点了。目前各种数据库产品中，基本上只有 MySQL 可以实现其底层数据存储引擎的插件式管理。这个模块实际上只是一个抽象类，但正是因为它成功地将各种数据处理高度抽象化，才成就了今天 MySQL 可插拔存储引擎的特色。
2. 从上图还可以看出，MySQL区别于其他数据库的最重要的特点就是其插件式的表存储引擎。MySQL插件式的存储引擎架构提供了一系列标准的管理和服务支持，这些标准与存储引擎本身无关，可能是每个数据库系统本身都必需的，如SQL分析器和优化器等，而存储引擎是底层物理结构的实现，每个存储引擎开发者都可以按照自己的意愿来进行开发。
3. 注意：存储引擎是基于表的，而不是数据库。

SQL 大致的查询流程

mysql 的查询流程大致是：

1. mysql 客户端通过协议与 mysql 服务器建连接，发送查询语句，先检查查询缓存，如果命中，直接返回结果，否则进行语句解析,也就是说，在解析查询之前，服务器会先访问查询缓存(query cache)——它存储 SELECT 语句以及相应的查询结果集。如果某个查询结果已经位于缓存中，服务器就不会再对查询进行解析、优化、以及执行。它仅仅将缓存中的结果返回给用户即可，这将大大提高系统的性能。
2. 语解析器和预处理：首先 mysql 通过关键字将 SQL 语句进行解析，并生成一颗对应的“解析树”。mysql 解析器将使用 mysql 语法规则验证和解析查询；预处理器则根据一些 mysql 规则进一步检查解析数是否合法。
3. 查询优化器当解析树被认为是合法的了，并且由优化器将其转化成执行计划。一条查询可以有多种执行方式，最后都返回相同的结果。优化器的作用就是找到这其中最好的执行计划。
4. 然后，mysql 默认使用的 BTREE 索引，并且一个大致方向是：无论怎么折腾 sql，至少在目前来说，mysql 最多只用到表中的一个索引。

6、MySQL 存储引擎

查看 mysql 存储引擎

- 查看 mysql 支持的存储引擎

1 | show engines;

```
mysql> show engines;
```

Engine	Support	Comment	Transactions	XA	Savepoints
PERFORMANCE_SCHEMA	YES	Performance Schema	NO	NO	NO
CSV	YES	CSV storage engine	NO	NO	NO
MRG_MYISAM	YES	Collection of identical MyISAM 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
FEDERATED	NO	Federated MySQL storage engine	NULL	NULL	NULL
ARCHIVE	YES	Archive storage engine	NO	NO	NO
MEMORY	YES	Hash based, stored in memory, useful for temporary tables	NO	NO	NO
InnoDB	DEFAULT	Supports transactions, row-level locking, and foreign keys	YES	YES	YES

9 rows in set (0.00 sec)

Oneby's Blog

- 查看 mysql 默认的存储引擎

```
1 | show variables like '%storage_engine%';
```

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

Variable_name	Value
default_storage_engine	InnoDB
default_tmp_storage_engine	InnoDB
storage_engine	InnoDB

3 rows in set (0.00 sec)

Oneby's Blog

MyISAM 引擎和 InnoDB 引擎的对比

对比项	MyISAM	InnoDB
主外键	不支持	支持
事务	不支持	支持
行表锁	表锁, 即使操作一条记录也会锁住整个表, 不适合高并发的操作	行锁,操作时只锁某一行, 不对其它行有影响, 适合高并发的操作
缓存	只缓存索引, 不缓存真实数据	不仅缓存索引还要缓存真实数据, 对内存要求较高, 而且内存大小对性能有决定性的影响
表空间	小	大
关注点	性能	事务
默认安装	Y	Y

Oneby's Blog

阿里巴巴用的是啥数据库?

- Percona为MySQL数据库服务器进行了改进, 在功能和性能上较MySQL有着很显著的提升。该版本提升了在高负载情况下的InnoDB的性能、为DBA提供一些非常有用的性能诊断工具; 另外有更多的参数和命令来控制服务器行为。
- 该公司新建了一款存储引擎叫xtradb完全可以替代innodb, 并且在性能和并发上做得更好, 阿里巴巴大部分mysql数据库其实使用的percona的原型加以修改。

产品	价格	目标	主要功能	是否可投入生产？
Percona Server	免费	提供 XtraDB 存储引擎的包装器和其他分析工具	XtraDB	是
MariaDB	免费	扩展 MySQL 以包含 XtraDB 和其他性能改进	XtraDB	是
Drizzle	免费	提供比 MySQL 更强大的可扩展性和性能改进	高可用性	是

Oneby's Blog

利用傲腾技术，助力企业突破内存存储瓶颈

提升数据存储和处理的整体性能，加速数据洞察，挖掘数据背后价值

广告 关闭