

第01章_Linux下MySQL的安装与使用

讲师：尚硅谷-宋红康（江湖人称：康师傅）

官网：<http://www.atguigu.com>

1.安装前说明

1.1Linux系统及工具的准备

- 安装并启动好两台虚拟机：CentOS 7
 - 掌握克隆虚拟机的操作
 - mac地址
 - 主机名
 - ip地址
 - UUID
- 安装有 xshell 和 xftp 等访问CentOS系统的工具
- CentOS6和CentOS7在MySQL的使用中的区别

1.防火墙：6是iptables，7是firewalld

2.启动服务的命令：6是service，7是systemctl

1.2查看是否安装过MySQL

- 如果你是用rpm安装,检查一下RPM PACKAGE:

```
1 | rpm -qa | grep -i mysql # -i 忽略大小写
```

- 检查mysql service:

```
systemctl status mysqld.service
```

- 如果存在mysql-libs的旧版本包，显示如下：

```
[root@atguigu01 opt]# rpm -qa | grep -i mysql
mysql-community-server-8.0.25-1.el7.x86_64
mysql-community-client-plugins-8.0.25-1.el7.x86_64
mysql-community-libs-8.0.25-1.el7.x86_64
mysql-community-client-8.0.25-1.el7.x86_64
mysql-community-common-8.0.25-1.el7.x86_64
[root@atguigu01 opt]#
```

- 如果不存在mysql-lib的版本，显示如下：

```
[root@atguigu02 opt]# rpm -qa | grep -i mysql
[root@atguigu02 opt]#
```

1.3MySQL的卸载

1. 关闭mysql服务

```
1 | systemctl stop mysqld.service
```

2. 查看当前mysql安装状况

```
1 | rpm -qa | grep -i mysql
2 | # 或
3 | yum list installed | grep mysql
```

3. 卸载上述命令查询出的已安装程序

```
1 | yum remove mysql-xxx mysql-xxx mysql-xxx mysql-xxxx
```

务必卸载干净，反复执行 `rpm -qa | grep -i mysql` 确认是否有卸载残留

4. 删除 mysql 相关文件

- 查找相关文件

```
1 | find / -name mysql
```

- 删除上述命令查找出的相关文件

```
1 | rm -rf xxx
```

5. 删除 my.cnf

```
1 | rm -rf /etc/my.cnf
```

2.MySQL的Linux版安装

2.1MySQL的4大版本

- **MySQL Community Server 社区版本**，开源免费，自由下载，但不提供官方技术支持，适用于大多数普通用户。
 - **MySQL Enterprise Edition 企业版本**，需付费，不能在线下载，可以试用30天。提供了更多的功能和更完备的技术支持，更适用于对数据库的功能和可靠性要求较高的企业客户。
 - **MySQL Cluster 集群版**，开源免费。用于架设集群服务器，可将几个MySQLServer封装成一个Server。需要在社区版或企业版的基础上使用。
 - **MySQL ClusterCGE 高级集群版**，需付费。
- 截止目前，官方最新版本为 **8.0.27**。此前，8.0.0在 2016.9.12日就发布了。
 - 本课程中主要使用 **8.0.25版本**。同时为了更好的说明MySQL8.0新特性，还会安装 MySQL5.7版本，作为对比。

此外，官方还提供了 **MySQL workbench** (GUITOOL) 一款专为MySQL设计的 **ER/数据库建模工具**。它是著名的数据库设计工具DBDesigner4的继任者。MySQLWorkbench又分为两个版本，分别是 **社区版** (MySQL Workbench OSS)、**商用版** (MySQL WorkbenchSE)。

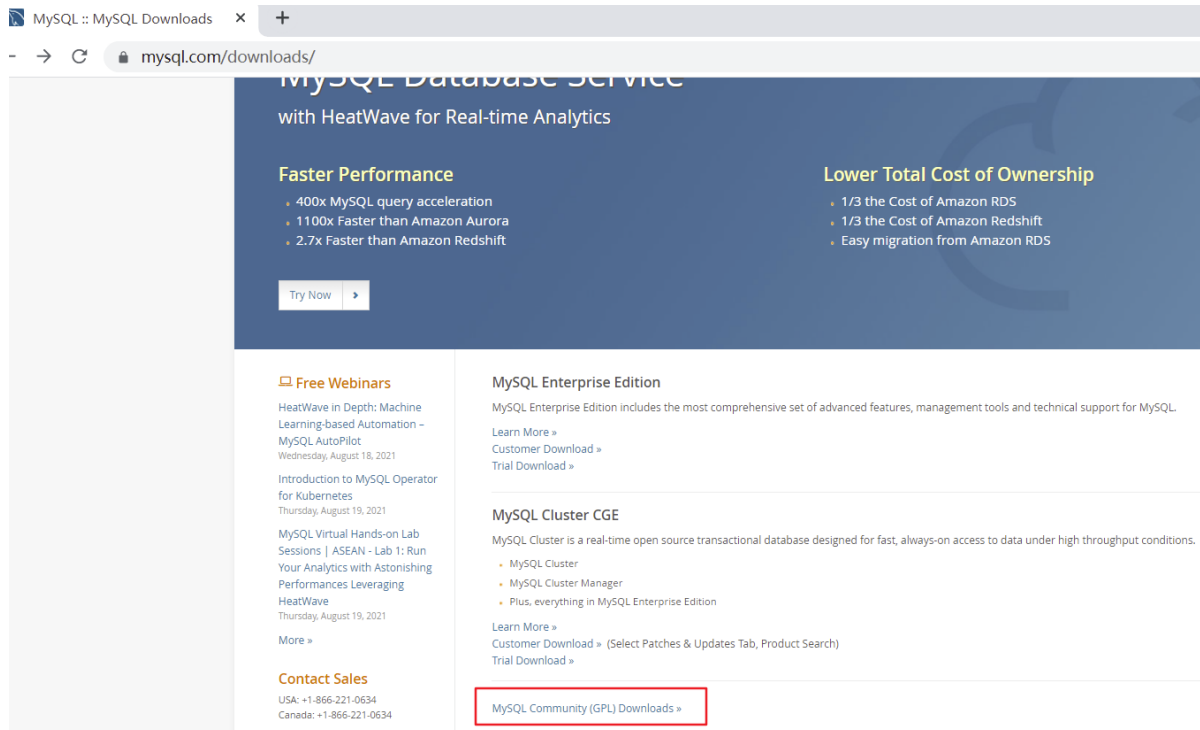
2.2 下载 MySQL 指定版本

1. 下载地址

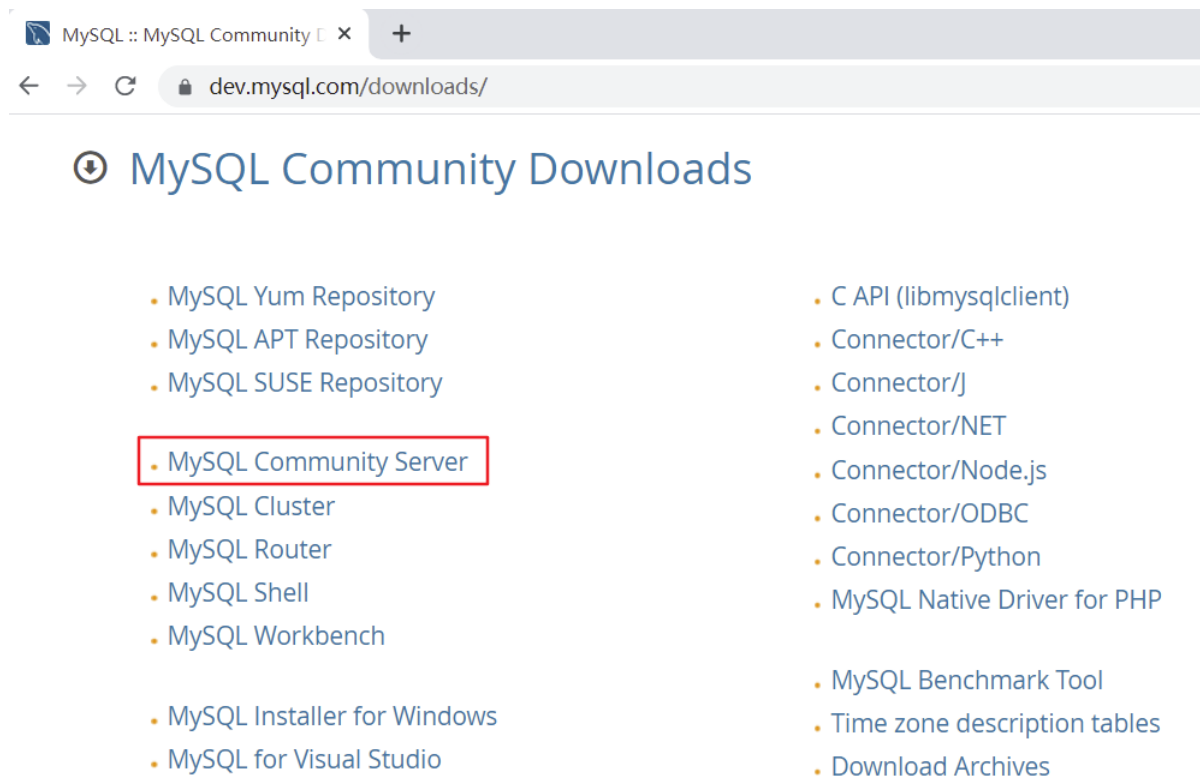
官网: <https://www.mysql.com>

2. 打开官网, 点击 DOWNLOADS

然后, 点击 MySQL Community (GPL) Downloads

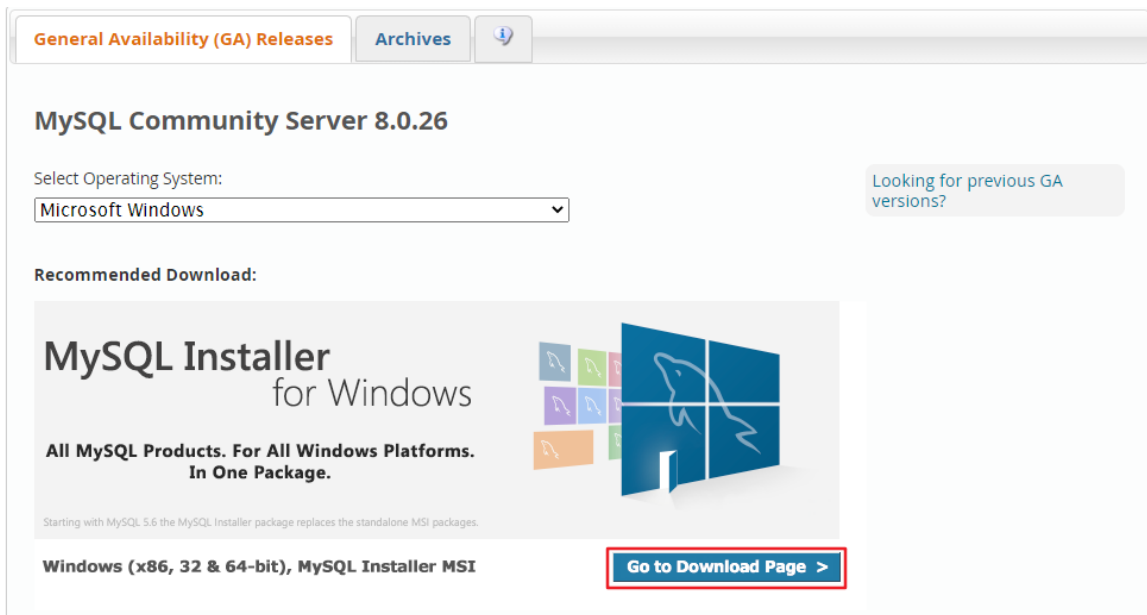


3. 点击 MySQL Community Server



4. 在 General Availability (GA) Releases 中选择适合的版本

- 如果安装 Windows 系统下 MySQL, 推荐下载 MSI 安装程序; 点击 Go to Download Page 进行下载即可



- Windows下的MySQL安装有两种安装程序
 - `mysql-installer-web-community-8.0.25.0.msi` 下载程序大小：2.4M；安装时需要联网安装组件。
 - `mysql-installer-community-8.0.25.0.msi` 下载程序大小：435.7M；安装时离线安装即可。**推荐。**

5. Linux系统下安装MySQL的几种方式

6. Linux系统下安装软件的常用三种方式：

方式1：rpm命令

使用rpm命令安装扩展名为".rpm"的软件包。

.rpm包的一般格式：



方式2：yum命令

需联网，从互联网获取的yum源，直接使用yum命令安装。

方式3：编译安装源码包

针对 `tar.gz` 这样的压缩格式，要用tar命令来解压；如果是其它压缩格式，就使用其它命令。

7. Linux系统下安装MySQL，官方给出多种安装方式

安装方式	特点
rpm	安装简单，灵活性差，无法灵活选择版本、升级
rpmrepository	安装包极小，版本安装简单灵活，升级方便，需要联网安装
通用二进制包	安装比较复杂，灵活性高，平台通用性好

安装方式	特点
源码包	安装最复杂，时间长，参数设置灵活，性能好

- 这里不能直接选择CentOS7系统的版本，所以选择与之对应的 Red Hat Enterprise Linux
- <https://downloads.mysql.com/archives/community/>直接点Download下载RPMBundle全量包。包括了所有下面的组件。不需要一个一个下载了。


Product Version:

Operating System:










OS Version:

Red Hat Enterprise Linux 8 / Oracle Linux 8 (x86, 64-bit), RPM Bundle (mysql-8.0.25-1.el8.x86_64.rpm-bundle.tar)	Apr 26, 2021	730.2M	Download
Red Hat Enterprise Linux 8 / Oracle Linux 8 (ARM, 64-bit), RPM Bundle (mysql-8.0.25-1.el8.aarch64.rpm-bundle.tar)	Apr 26, 2021	741.1M	Download
Red Hat Enterprise Linux 8 / Oracle Linux 8 (x86, 64-bit), RPM Package MySQL Server (mysql-community-server-8.0.25-1.el8.x86_64.rpm)	Apr 26, 2021	52.9M	Download
Red Hat Enterprise Linux 8 / Oracle Linux 8 (ARM, 64-bit), RPM Package MySQL Server (mysql-community-server-8.0.25-1.el8.aarch64.rpm)	Apr 26, 2021	60.2M	Download
Red Hat Enterprise Linux 8 / Oracle Linux 8 (x86, 64-bit), RPM Package Client Utilities (mysql-community-client-8.0.25-1.el8.x86_64.rpm)	Apr 26, 2021	13.4M	Download
Red Hat Enterprise Linux 8 / Oracle Linux 8 (ARM, 64-bit), RPM Package Client Utilities (mysql-community-client-8.0.25-1.el8.aarch64.rpm)	Apr 26, 2021	14.9M	Download

5. 下载的tar包，用压缩工具打开

 mysql-8.0.25-1.el7.x86_64.rpm-bundle.tar

- 解压后rpm安装包（红框为抽取出来的安装包）

MySQL安装包 > linux版 > MySQL8.0.25 > mysql-8.0.25-1.el7.x86_64.rpm-bundle	
名称	修改日期
 mysql-community-client-8.0.25-1.el7.x86_64.rpm	2021/4/26 15:36
 mysql-community-client-plugins-8.0.25-1.el7.x86_64.rpm	2021/4/26 15:36
 mysql-community-common-8.0.25-1.el7.x86_64.rpm	2021/4/26 15:36
 mysql-community-devel-8.0.25-1.el7.x86_64.rpm	2021/4/26 15:36
 mysql-community-embedded-compat-8.0.25-1.el7.x86_64.rpm	2021/4/26 15:37
 mysql-community-libs-8.0.25-1.el7.x86_64.rpm	2021/4/26 15:37
 mysql-community-libs-compat-8.0.25-1.el7.x86_64.rpm	2021/4/26 15:37
 mysql-community-server-8.0.25-1.el7.x86_64.rpm	2021/4/26 15:38
 mysql-community-test-8.0.25-1.el7.x86_64.rpm	2021/4/26 15:40

2.3CentOS7下检查 MySQL依赖

- 检查 /tmp临时目录权限（必不可少）

由于mysql安装过程中，会通过mysql用户在/tmp目录下新建tmp_db文件，所以请给/tmp较大的权限。执行：

```
1 | chmod -R 777 /tmp
```

```
dr-xr-xr-x. 13 root root    0 12月 31 16:57 sys
drwxrwxrwt. 35 root root 4096 12月 31 17:26 tmp
drwxr-xr-x. 13 root root   155 12月  6 15:39 usr
drwxr-xr-x. 21 root root 4096 12月  6 15:52 var
```

2. 安装前，检查依赖

```
1 | rpm -qa | grep libaio
```

- 如果存在libaio包如下：

```
[root@atguigu01 ~]# rpm -qa|grep libaio
libaio-0.3.109-13.el7.x86_64
```

```
1 | rpm -qa | grep net-tools
```

- 如果存在net-tools包如下：

```
[root@atguigu01 ~]# rpm -qa|grep net-tools
net-tools-2.0-0.22.20131004git.el7.x86_64
```

```
1 | rpm -qa | grep net-tools
```

- 如果不存在需要到centos安装盘里进行rpm安装。安装 linux如果带图形化界面，这些都是安装好的。

2.4CentOS7下MySQL安装过程

1. 将安装程序拷贝到 /opt目录下

在mysql的安装文件目录下执行：（必须按照顺序执行）

```
1 | tar -xvf mysql-8.4.0-1.el7.x86_64.rpm-bundle.tar
2 | rpm -ivh mysql-community-common-8.4.0-1.el7.x86_64.rpm
3 | rpm -ivh mysql-community-client-plugins-8.4.0-1.el7.x86_64.rpm
4 | yum remove mysql-libs
5 | rpm -ivh mysql-community-libs-8.4.0-1.el7.x86_64.rpm
6 | rpm -ivh mysql-community-client-8.4.0-1.el7.x86_64.rpm
7 | rpm -ivh mysql-community-icu-data-files-8.4.0-1.el7.x86_64.rpm
8 | rpm -ivh mysql-community-server-8.4.0-1.el7.x86_64.rpm
```

- 注意:如在检查工作时，没有检查mysql依赖环境在安装mysql-community-server会报错
- rpm是RedhatPackageManage缩写，通过RPM的管理，用户可以把源代码包装成以 rpm为扩展名的文件形式，易于安装。
- i ,--install安装软件包
- v ,--verbose提供更多的详细信息输出
- h ,--hash软件包安装的时候列出哈希标记 (和 -v一起使用效果更好)，展示进度条

```
[root@atguigu01 opt]# ll
总用量 1237756
-rw-r--r--. 1 root root 8924465 7月 26 23:36 apache-tomcat-7.0.70.tar.gz
-rw-r--r--. 1 root root 567238341 7月 26 23:36 ideaIC-2019.2.4-no-jbr.tar.gz
-rw-r--r--. 1 root root 189784266 7月 26 23:36 jdk-8u152-linux-x64.tar.gz
-rw-r--r--. 1 root root 47810444 7月 27 22:59 mysql-community-client-8.0.25-1.el7.x86_64.rpm
-rw-r--r--. 1 root root 193616 7月 27 23:02 mysql-community-client-plugins-8.0.25-1.el7.x86_64.rpm
-rw-r--r--. 1 root root 628904 7月 27 22:59 mysql-community-common-8.0.25-1.el7.x86_64.rpm
-rw-r--r--. 1 root root 4240320 7月 27 22:59 mysql-community-libs-8.0.25-1.el7.x86_64.rpm
-rw-r--r--. 1 root root 448614076 7月 27 22:59 mysql-community-server-8.0.25-1.el7.x86_64.rpm
drwxr-xr-x. 2 root root 4096 9月 7 2017 .
```

2. 安装过程截图

```
[root@atguigu01 opt]# rpm -ivh mysql-community-client-plugins-8.0.25-1.el7.x86_64.rpm
警告: mysql-community-client-plugins-8.0.25-1.el7.x86_64.rpm: 头V3 DSA/SHA1 Signature, 密钥 ID 5072e1f5: NOKEY
准备中...
正在升级/安装...
1:mysql-community-client-plugins-8.0.25-1.el7.x86_64.rpm [100%]
[root@atguigu01 opt]# rpm -ivh mysql-community-libs-8.0.25-1.el7.x86_64.rpm
警告: mysql-community-libs-8.0.25-1.el7.x86_64.rpm: 头V3 DSA/SHA1 Signature, 密钥 ID 5072e1f5: NOKEY
准备中...
正在升级/安装...
1:mysql-community-libs-8.0.25-1.el7.x86_64.rpm [100%]
[root@atguigu01 opt]# rpm -ivh mysql-community-client-8.0.25-1.el7.x86_64.rpm
警告: mysql-community-client-8.0.25-1.el7.x86_64.rpm: 头V3 DSA/SHA1 Signature, 密钥 ID 5072e1f5: NOKEY
准备中...
正在升级/安装...
1:mysql-community-client-8.0.25-1.el7.x86_64.rpm [100%]
[root@atguigu01 opt]# rpm -ivh mysql-community-server-8.0.25-1.el7.x86_64.rpm
警告: mysql-community-server-8.0.25-1.el7.x86_64.rpm: 头V3 DSA/SHA1 Signature, 密钥 ID 5072e1f5: NOKEY
准备中...
正在升级/安装...
1:mysql-community-server-8.0.25-1.el7.x86_64.rpm [100%]
```

安装过程中可能的报错信息:

```
[root@localhost opt]# rpm -ivh mysql-community-libs-8.0.25-1.el7.x86_64.rpm
警告: mysql-community-libs-8.0.25-1.el7.x86_64.rpm: 头V3 DSA/SHA1 Signature, 密钥 ID 5072e1f5: NOKEY
错误: 依赖检测失败:
mariadb-libs 被 mysql-community-libs-8.0.25-1.el7.x86_64 取代
[root@localhost opt]# rpm -ivh mysql-community-libs-8.0.25-1.el7.x86_64.rpm
警告: mysql-community-libs-8.0.25-1.el7.x86_64.rpm: 头V3 DSA/SHA1 Signature, 密钥 ID 5072e1f5: NOKEY
```

一个命令: `yum remove mariadb-libs` 解决, 清除之前安装过的依赖即可

3. 查看 MySQL 版本

执行如下命令, 如果成功表示安装mysql成功。类似 `java -version` 如果打出版本等信息

```
1 mysql --version
2 # 或
3 mysqladmin --version
```

```
[root@atguigu01 opt]# mysqladmin --version
mysqladmin Ver 8.0.25 for Linux on x86_64 (MySQL Community Server - GPL)
```

执行如下命令, 查看是否安装成功。需要增加 `-i` 不用去区分大小写, 否则搜索不到。

```
1 rpm -qa | grep -i mysql
```

```
[root@atguigu01 opt]# rpm -qa|grep -i mysql
mysql-community-server-8.0.25-1.el7.x86_64
mysql-community-client-plugins-8.0.25-1.el7.x86_64
mysql-community-libs-8.0.25-1.el7.x86_64
mysql-community-client-8.0.25-1.el7.x86_64
mysql-community-common-8.0.25-1.el7.x86_64
```

4. 服务的初始化

为了保证数据库目录与文件的所有者为 mysql 登录用户, 如果你是以 root 身份运行 mysql 服务, 需要执行下面的命令初始化:


```
1 | mysql --initialize --user=mysql
```

说明: --initialize选项默认以“安全”模式来初始化, 则会为 root用户生成一个密码并将该密码标记为过期, 登录后你需要设置一个新的密码。生成的临时密码 会往日志中记录一份。

查看密码:

```
1 | cat /var/log/mysql.log
```

```
[root@atguigu01 log]# cat /var/log/mysql.log
2021-07-27T16:27:30.018766Z 0 [System] [MY-013169] [Server] /usr/sbin/mysqld (mysqld 8.0.25) initializing of server in progress as process 4532
2021-07-27T16:27:30.027795Z 1 [System] [MY-013576] [InnoDB] InnoDB initialization has started.
2021-07-27T16:27:30.245036Z 1 [System] [MY-013577] [InnoDB] InnoDB initialization has ended.
2021-07-27T16:27:31.054680Z 6 [Note] [MY-010454] [Server] A temporary password is generated for root@localhost: !E/_uZ_o.2sa
2021-07-27T16:27:33.490420Z 0 [System] [MY-010116] [Server] /usr/sbin/mysqld (mysqld 8.0.25) starting as process 4577
2021-07-27T16:27:33.500790Z 1 [System] [MY-013576] [InnoDB] InnoDB initialization has started.
2021-07-27T16:27:33.656675Z 1 [System] [MY-013577] [InnoDB] InnoDB initialization has ended.
2021-07-27T16:27:33.750754Z 0 [System] [MY-011323] [Server] X Plugin ready for connections. Bind-address: '::' port: 33060, socket: /var/run/mysql/mysql.sock
2021-07-27T16:27:33.902569Z 0 [Warning] [MY-010068] [Server] CA certificate ca.pem is self signed.
2021-07-27T16:27:33.902731Z 0 [System] [MY-013602] [Server] Channel mysql_main configured to support TLS. Encrypted connections are now supported for this channel.
2021-07-27T16:27:33.921842Z 0 [System] [MY-010931] [Server] /usr/sbin/mysqld: ready for connections. Version: '8.0.25' socket: '/var/lib/mysql/mysql.sock'
2021-07-27T16:27:33.921842Z 0 [System] [MY-011323] [Server] X Plugin ready for connections. Bind-address: '::' port: 33060, socket: /var/run/mysql/mysql.sock
```

root@localhost:后面就是初始化的密码

5. 启动 MySQL, 查看状态

```
1 | #加不加.service后缀都可以
2 | 启动: systemctl start mysqld.service
3 | 关闭: systemctl stop mysqld.service
4 | 重启: systemctl restart mysqld.service
5 | 查看状态: systemctl status mysqld.service
```

mysqld这个可执行文件就代表着 MySQL 服务器程序, 运行这个可执行文件就可以直接启动一个服务器进程。

```
[root@atguigu01 log]# systemctl start mysqld.service
[root@atguigu01 log]# systemctl status mysqld.service
● mysqld.service - MySQL Server
   Loaded: loaded (/usr/lib/systemd/system/mysqld.service; disabled; vendor preset: disabled)
   Active: active (running) since 2021-07-28 00:27:33 CST; 12s ago
     Docs: man:mysqld(8)
           http://dev.mysql.com/doc/refman/en/using-systemd.html
   Process: 4495 ExecStartPre=/usr/bin/mysqld_pre_systemd (code=exited, status=0/SUCCESS)
  Main PID: 4577 (mysqld)
    Status: "Server is operational"
     Tasks: 38
   CGroup: /system.slice/mysqld.service
           └─4577 /usr/sbin/mysqld

7月 28 00:27:22 atguigu01 systemd[1]: Starting MySQL Server...
7月 28 00:27:33 atguigu01 systemd[1]: Started MySQL Server.
```

查看进程:

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

```
[root@atguigu01 log]# ps -ef | grep -i mysql
mysql      4674      1  0 00:28 ?        00:00:00 /usr/sbin/mysqld
root       4729    3193  0 00:29 pts/1    00:00:00 grep --color=auto -i mysql
```

6. 查看 MySQL服务是否自启动

```
1 | systemctl list-unit-files | grep mysqld.service
```

```
[root@atguigu01 opt]# systemctl list-unit-files|grep mysqld.service
mysqld.service                                enabled
```


默认是enabled。

- 如果不是enabled可以运行如下命令设置自启动

```
1 | systemctl enable mysqld.service
```

```
[root@atguigu01 log]# systemctl enable mysqld.service
Created symlink from /etc/systemd/system/multi-user.target.wants/mysqld.service to /usr/lib/systemd/system/mysqld.service.
[root@atguigu01 log]# systemctl list-unit-files|grep mysqld.service
mysqld.service                                enabled
```

- 如果希望不进行自启动，运行如下命令设置

```
1 | systemctl disable mysqld.service
```

```
[root@atguigu01 log]# systemctl disable mysqld.service
Removed symlink /etc/systemd/system/multi-user.target.wants/mysqld.service.
[root@atguigu01 log]# systemctl list-unit-files|grep mysqld.service
mysqld.service                                disabled
```

3.MySQL登录

3.1首次登录

通过 `mysql -hlocalhost -P3306 -uroot -p` 进行登录，在Enter password: 录入初始化密码

```
[root@atguigu01 init.d]# mysql -hlocalhost -P3306 -uroot -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 8
Server version: 8.0.25 MySQL Community Server - GPL

Copyright (c) 2000, 2021, Oracle and/or its affiliates.

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.
```

3.2修改密码

- 因为初始化密码默认是过期的，所以查看数据库会报错
- 修改密码：

```
1 | ALTER USER 'root'@'localhost' IDENTIFIED BY 'new_password';
```

- 5.7版本之后（不含5.7），mysql加入了全新的密码安全机制。设置新密码太简单会报错。

```
mysql> ALTER USER 'root'@'localhost' IDENTIFIED BY 'HelloWorld';
ERROR 1819 (HY000): Your password does not satisfy the current policy requirements
mysql> ALTER USER 'root'@'localhost' IDENTIFIED BY 'HelloWorld123';
ERROR 1819 (HY000): Your password does not satisfy the current policy requirements
mysql> ALTER USER 'root'@'localhost' IDENTIFIED BY 'Hello_World';
ERROR 1819 (HY000): Your password does not satisfy the current policy requirements
```

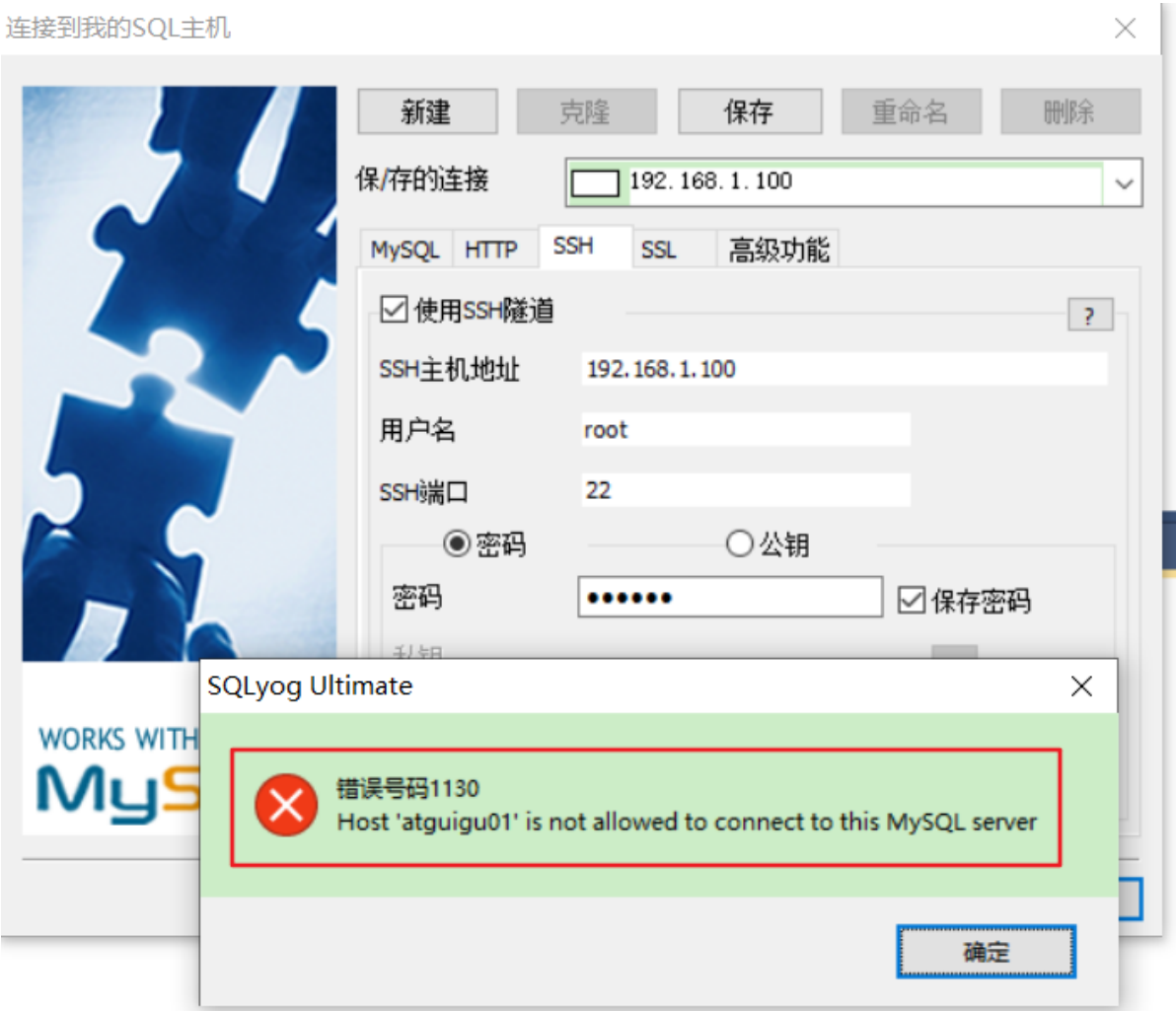
- 改为更复杂的密码规则之后，设置成功，可以正常使用数据库了

```
mysql> ALTER USER 'root'@'localhost' IDENTIFIED BY 'Hello_World123';
Query OK, 0 rows affected (0.01 sec)
```

3.3设置远程登录

1.当前问题

在用SQLyog或Navicat中配置远程连接Mysql数据库时遇到如下报错信息，这是由于 Mysql配置了不支持远程连接引起的。

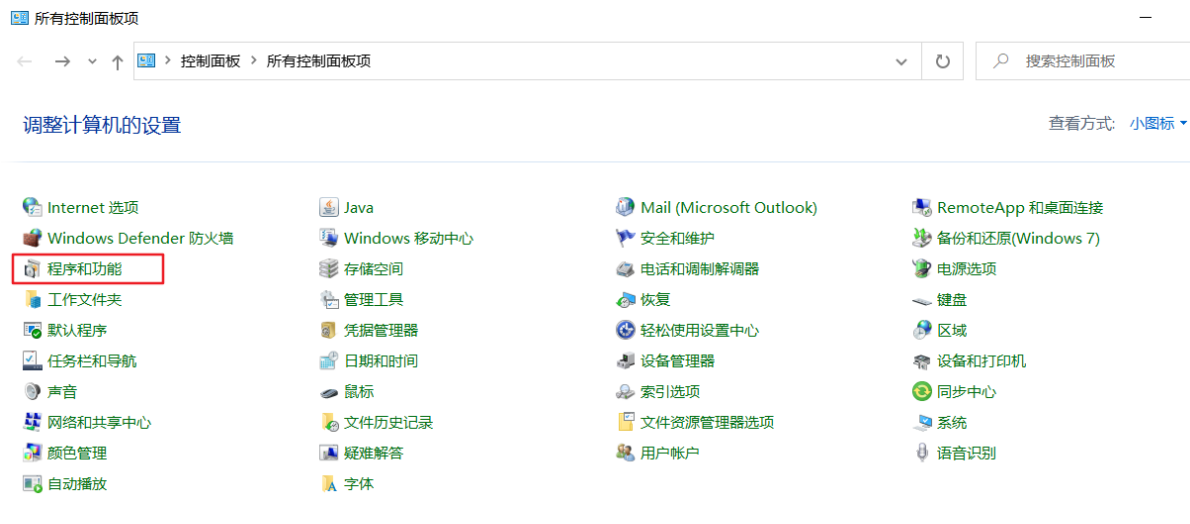


2.确认网络

1. 在远程机器上使用ping ip地址 保证网络畅通
2. 在远程机器上使用telnet命令 保证端口号开放 访问

```
1 | telnet ip地址 端口号
```

拓展：telnet命令开启：



控制面板主页

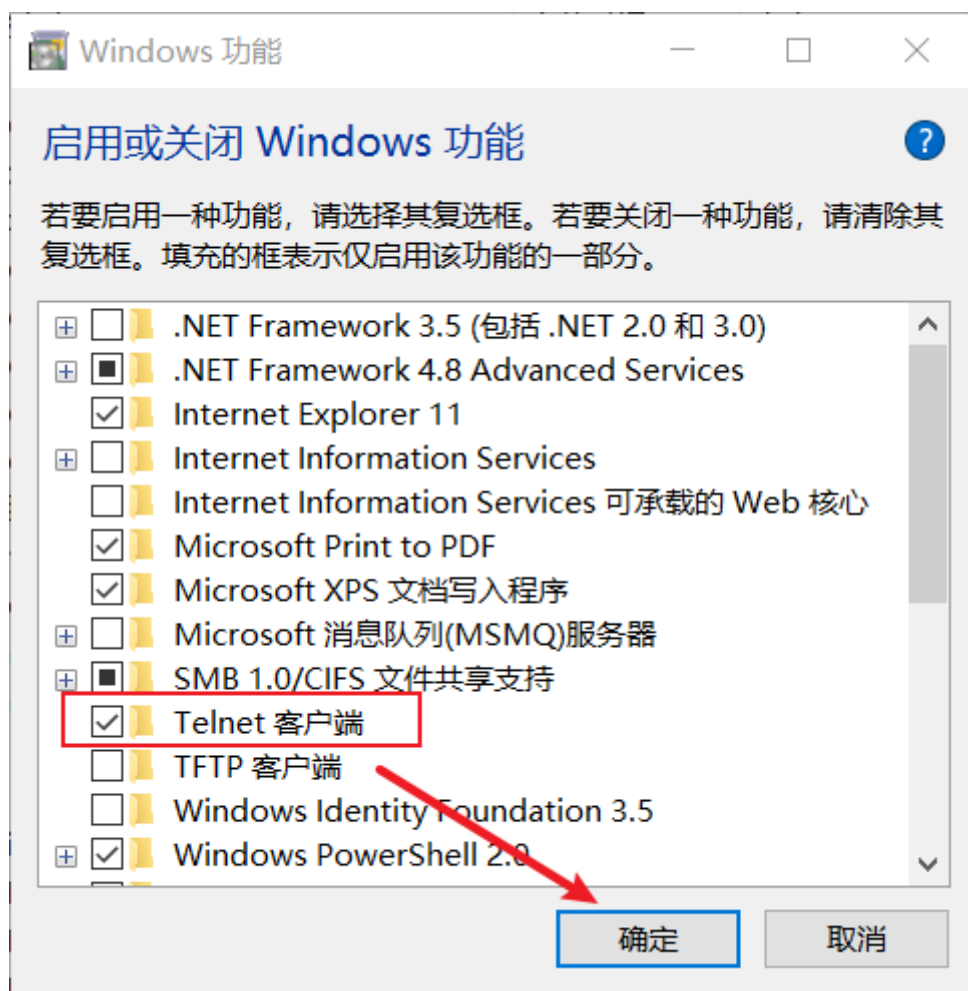
卸载或更改程序

查看已安装的更新

若要卸载程序，请从列表中将其选中，然后单击“卸载”、“更改”或“修复”。

启用或关闭 Windows 功能

组织 ▼



3.关闭防火墙或开放端口

方式一：关闭防火墙

- CentOS6:

```
1 | service iptables stop
```

- CentOS7:

```
1 systemctl start firewalld.service
2 systemctl status firewalld.service
3 systemctl stop firewalld.service
4 #设置开机启用防火墙
5 systemctl enable firewalld.service
6 #设置开机禁用防火墙
7 systemctl disable firewalld.service
```

方式二：开放端口

- 查看开放的端口号

```
1 firewall-cmd --list-all
```

- 设置开放的端口号

```
1 firewall-cmd --add-service=http --permanent
2 firewall-cmd --add-port=3306/tcp --permanent
```

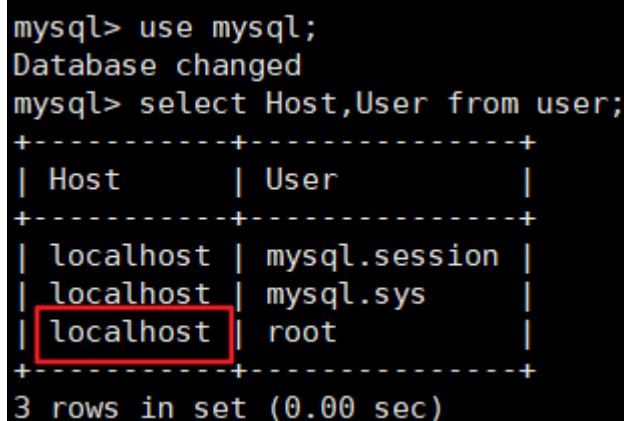
- 重启防火墙

```
1 firewall-cmd --reload
```

4.Linux下修改配置

在Linux系统MySQL下测试：

```
1 use mysql;
2 select Host,User from user;
```



```
mysql> use mysql;
Database changed
mysql> select Host,User from user;
+-----+-----+
| Host      | User          |
+-----+-----+
| localhost | mysql.session |
| localhost | mysql.sys     |
| localhost | root          |
+-----+-----+
3 rows in set (0.00 sec)
```

可以看到root用户的当前主机配置信息为localhost。

- 修改Host为通配符%

Host列指定了允许用户登录所使用的IP，比如user=rootHost=192.168.1.1。这里的意思就是说root用户只能通过192.168.1.1的客户端去访问。user=rootHost=localhost，表示只能通过本机客户端去访问。而 %是个 通配符，如果Host=192.168.1.%，那么就表示只要是IP地址前缀为“192.168.1.”的客户端都可以连接。如果 Host=%，表示所有IP都有连接权限。

注意：在生产环境下不能为了省事将host设置为 %，这样做会存在安全问题，具体的设置可以根据生产环境的IP进行设置。

```
1 | update user set host='%' where user='root';
```

Host设置了“%”后便可以允许远程访问。

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

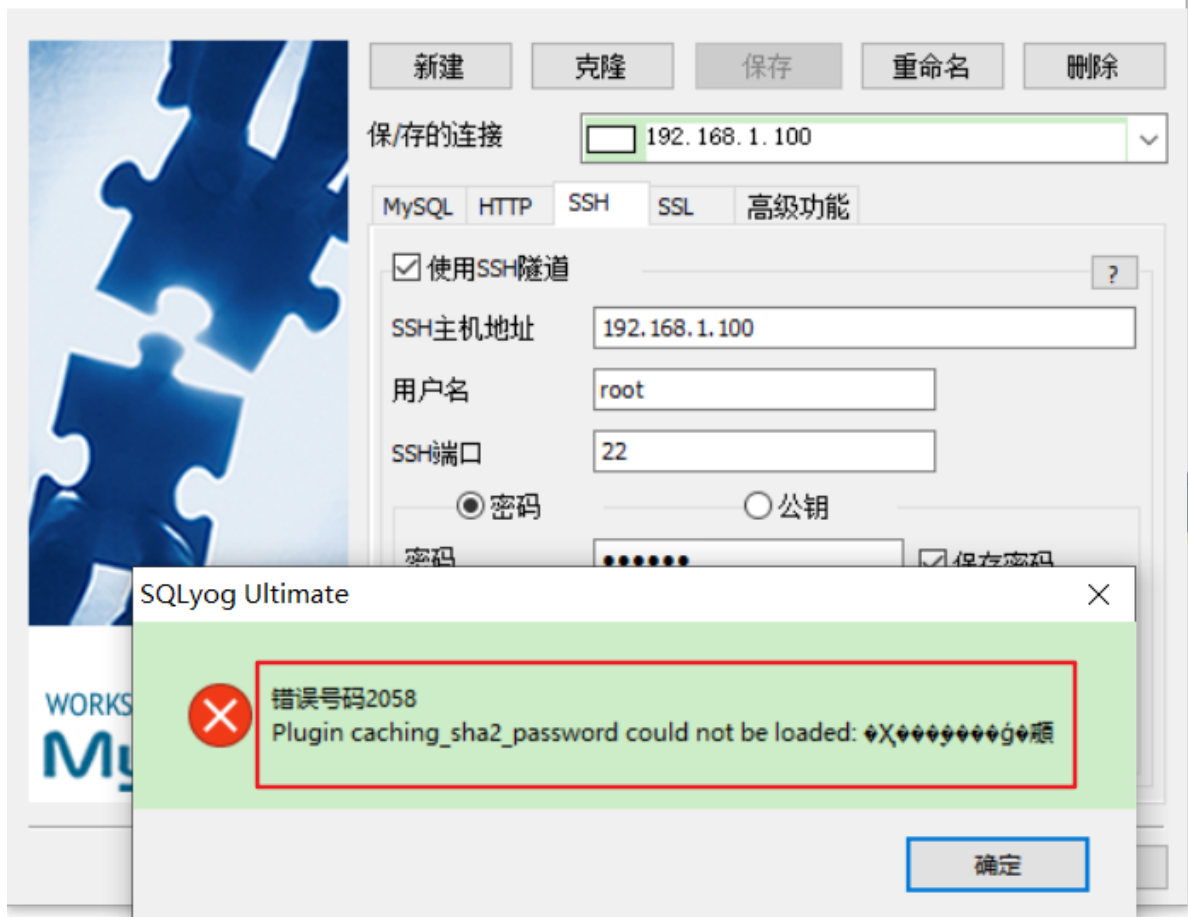
mysql> select Host,User from user;
+-----+-----+
| Host      | User      |
+-----+-----+
| %         | root      |
| localhost | mysql.session |
| localhost | mysql.sys  |
+-----+-----+
3 rows in set (0.00 sec)
```

Host修改完成后记得执行flush privileges使配置立即生效：

```
1 | flush privileges;
```

5.测试

- 如果是 MySQL5.7版本，接下来就可以使用SQLyog或者Navicat成功连接至MySQL了。
- 如果是 MySQL8版本，连接时还会出现如下问题：



配置新连接报错：错误号码 2058，分析是 mysql密码加密方法变了。

解决方法：Linux下 mysql -u root -p登录你的 mysql数据库，然后执行这条 SQL：

```
1 | ALTER USER 'root'@'%' IDENTIFIED WITH mysql_native_password BY 'abc123';
```

然后在重新配置SQLyog的连接，则可连接成功了，OK。

4.MySQL8的密码强度评估（了解）

4.1MySQL不同版本设置密码（可能出现）

- MySQL5.7中：成功

```
1 | mysql> alter user 'root' identified by 'abcd1234';
2 | Query OK, 0 rows affected (0.00 sec)
```

- MySQL8.0中：失败

```
1 | mysql> alter user 'root' identified by 'abcd1234'; # HelloWorld_123
2 | ERROR 1819 (HY000): Your password does not satisfy the current policy requirements
```


4.2MySQL8之前的安全策略

在MySQL8.0之前，MySQL使用的是validate_password插件检测、验证账号密码强度，保障账号的安全性。

安装/启用插件方式1：在参数文件my.cnf中添加参数

```
1 [mysqld]
2 plugin-load-add=validate_password.so
3 \#ON/OFF/FORCE/FORCE_PLUS_PERMANENT:是否使用该插件（及强制 /永久强制使用）
4 validate_password=FORCE_PLUS_PERMANENT
```

说明1：plugin library中的validate_password文件名的后缀名根据平台不同有所差异。对于Unix和Unix-like系统而言，它的文件后缀名是.so，对于Windows系统而言，它的文件后缀名是.dll。

说明2：修改参数后必须重启MySQL服务才能生效。

说明3：参数FORCE_PLUS_PERMANENT是为了防止插件在MySQL运行时的时候被卸载。当你卸载插件时就会报错。如下所示。

```
1 mysql> SELECT PLUGIN_NAME, PLUGIN_LIBRARY, PLUGIN_STATUS, LOAD_OPTION
2         -> FROM INFORMATION_SCHEMA.PLUGINS
3         -> WHERE PLUGIN_NAME = 'validate_password';
4 +-----+-----+-----+-----+
5 | PLUGIN_NAME | PLUGIN_LIBRARY | PLUGIN_STATUS | LOAD_OPTION |
6 +-----+-----+-----+-----+
7 | validate_password | validate_password.so | ACTIVE | |
8 | FORCE_PLUS_PERMANENT | | | |
9 +-----+-----+-----+-----+
10 1 row in set (0.00 sec)
11 mysql> UNINSTALL PLUGIN validate_password;
12 ERROR 1702 (HY000): Plugin 'validate_password' is force_plus_permanent and
13 can not be unloaded
14 mysql>
```

安装/启用插件方式2：运行时命令安装（推荐）

```
1 mysql> INSTALL PLUGIN validate_password SONAME 'validate_password.so';
2 Query OK, 0 rows affected, 1 warning (0.11 sec)
```

此方法也会注册到元数据，也就是mysql.plugin表中，所以不用担心MySQL重启后插件会失效。

4.3MySQL8的安全策略

1. validate_password说明

MySQL8.0，引入了服务器组件（Components）这个特性，validate_password插件已用服务器组件重新实现。8.0.25版本的数据库中，默认自动安装validate_password组件。

未安装插件前，执行如下两个指令，执行效果：

```

1 mysql> show variables like 'validate_password%';
2 Empty set (0.04 sec)
3 mysql> SELECT * FROM mysql.component;
4 ERROR 1146 (42S02): Table 'mysql.component' doesn't exist

```

安装插件后，执行如下两个指令，执行效果：

```

1 mysql> SELECT * FROM mysql.component;
2 +-----+-----+-----+
3 | component_id | component_group_id | component_urn |
4 +-----+-----+-----+
5 |          1 |          1 | file://component_validate_password |
6 +-----+-----+-----+
7 1 row in set (0.00 sec)

```

```

1 mysql> show variables like 'validate_password%';
2 +-----+-----+
3 | Variable_name | value |
4 +-----+-----+
5 | validate_password.check_user_name | ON |
6 | validate_password.dictionary_file | |
7 | validate_password.length | 8 |
8 | validate_password.mixed_case_count | 1 |
9 | validate_password.number_count | 1 |
10 | validate_password.policy | MEDIUM |
11 | validate_password.special_char_count | 1 |
12 +-----+-----+
13 7 rows in set (0.01 sec)

```

关于 `validate_password` 组件对应的系统变量说明：

选项	默认值	参数描述
<code>validate_password_check_user_name</code>	ON	设置为ON的时候表示能将密码设置成当前用户名。
<code>validate_password_dictionary_file</code>		用于检查密码的字典文件的路径名，默认为空
<code>validate_password_length</code>	8	密码的最小长度，也就是说密码长度必须大于或等于8
<code>validate_password_mixed_case_count</code>	1	如果密码策略是中等或更强的， <code>validate_password</code> 要求密码具有的小写和大写字符的最小数量。对于给定的这个值密码必须有那么多小写字符和那么多大写字符。
<code>validate_password_number_count</code>	1	密码必须包含的数字个数

选项	默认值	参数描述
<code>validate_password_policy</code>	MEDIUM	密码强度检验等级，可以使用数值0、1、2或相应的符号值LOW、MEDIUM、STRONG来指定。0/LOW：只检查长度。1/MEDIUM：检查长度、数字、大小写、特殊字符。2/STRONG：检查长度、数字、大小写、特殊字符、字典文件。
<code>validate_password_special_char_count</code>	1	密码必须包含的特殊字符个数

提示：

组件和插件的默认值可能有所不同。例如，MySQL5.7.validate_password_check_user_name的默认值为OFF。

2.修改安全策略

修改密码验证安全强度

```

1 SET GLOBAL validate_password_policy=LOW;
2 SET GLOBAL validate_password_policy=MEDIUM;
3 SET GLOBAL validate_password_policy=STRONG;
4 SET GLOBAL validate_password_policy=0; # For LOW
5 SET GLOBAL validate_password_policy=1; # For MEDIUM
6 SET GLOBAL validate_password_policy=2; # For HIGH
7 #注意，如果是插件的话，SQL为set global validate_password_policy=LOW

```

此外，还可以修改密码中字符的长度

```

1 set global validate_password_length=1;

```

3.密码强度测试

如果你创建密码是遇到“Your password does not satisfy the current policy requirements”，可以通过函数组件去检测密码是否满足条件：0-100。当评估在100时就是说明使用上了最基本的规则：大写+小写+特殊字符+数字组成的8位以上密码

```

1 mysql> SELECT VALIDATE_PASSWORD_STRENGTH('medium');
2 +-----+
3 | VALIDATE_PASSWORD_STRENGTH('medium') |
4 +-----+
5 |                                     25 |
6 +-----+
7 1 row in set (0.00sec)

```

```

1 mysql> SELECT VALIDATE_PASSWORD_STRENGTH('k354*45jkd5');
2 +-----+
3 | VALIDATE_PASSWORD_STRENGTH('k354*45jkd5') |
4 +-----+
5 |                                           100 |
6 +-----+
7 1 row in set (0.00sec)

```

注意：如果没有安装validate_password组件或插件的话，那么这个函数永远都返回 0。关于密码复杂度对应的密码复杂度策略。如下表格所示：

PasswordTest	Return Value
Length<4	0
Length≥ 4and<validate_password.length	25
Satisfiespolicy1(LOW)	50
Satisfiespolicy2(MEDIUM)	75
Satisfiespolicy3(STRONG)	100

当密码强度符合MEDIUM时，返回值是75。这个需要补充一个前提：

- 有字典文件，才可能会有75这个返回值
- 没有字典文件的时候，密码强度符合MEDIUM时，返回值是100

MEDIUM 验证长度 数字 大小写 特殊字符

STRONG 验证长度 数字 大小写 特殊字符 字典文件

什么是字典文件，可以理解成一个文本，上面有一行一行的单词

假如 密码长度 数字 大小写 特殊字符，都满足要求

但是包含了字典文件中的单词，那么返回值就是75

如果没有包含字典文件中的单词，那么返回值就是100

4.4卸载插件、组件(了解)

卸载插件

```

1 mysql> UNINSTALL PLUGIN validate_password;
2 Query OK, 0 rows affected, 1 warning (0.01 sec)

```

卸载组件

```

1 mysql> UNINSTALL COMPONENT 'file://component_validate_password';
2 Query OK, 0 rows affected (0.02sec)

```

5.字符集的相关操作

5.1修改 MySQL5.7字符集

1.修改步骤

在MySQL8.0版本之前，默认字符集为 `latin1`，utf8字符集指向的是 `utf8mb3`。网站开发人员在数据库设计的时候往往会将编码修改为utf8字符集。如果遗忘修改默认的编码，就会出现乱码的问题。从MySQL 8.0开始，数据库的默认编码将改为 `utf8mb4`，从而避免上述乱码的问题。

操作1：查看默认使用的字符集

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

- MySQL8.0中执行：

```
mysql> show variables like 'character%';
+-----+-----+
| Variable_name | Value |
+-----+-----+
| character_set_client | utf8mb4 |
| character_set_connection | utf8mb4 |
| character_set_database | utf8mb4 |
| character_set_filesystem | binary |
| character_set_results | utf8mb4 |
| character_set_server | utf8mb4 |
| character_set_system | utf8mb3 |
| character_sets_dir | /usr/share/mysql-8.0/charsets/ |
+-----+-----+
8 rows in set (0.01 sec)

mysql> show variables like '%char%';
+-----+-----+
| Variable_name | Value |
+-----+-----+
| character_set_client | utf8mb4 |
| character_set_connection | utf8mb4 |
| character_set_database | utf8mb4 |
| character_set_filesystem | binary |
| character_set_results | utf8mb4 |
| character_set_server | utf8mb4 |
| character_set_system | utf8mb3 |
| character_sets_dir | /usr/share/mysql-8.0/charsets/ |
| validate_password.special_char_count | 1 |
+-----+-----+
9 rows in set (0.00 sec)
```

- MySQL5.7中执行：

MySQL5.7默认的客户端和服务端都用了 `latin1`，不支持中文，保存中文会报错。MySQL5.7截图如下：

```
mysql> show variables like 'character%';
+-----+-----+
| 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/mysqlCharsets/ |
+-----+-----+
8 rows in set (0.00 sec)
```

在MySQL5.7中添加中文数据时，报错：

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

mysql> create database dbtest1;
Query OK, 1 row affected (0.00 sec)

mysql> use dbtest1;
Database changed
mysql> create table t_emp(id int,name varchar(15));
Query OK, 0 rows affected (0.01 sec)

mysql> insert into t_emp(id,name)values(1001,'Tom');
Query OK, 1 row affected (0.01 sec)

mysql> insert into t_emp(id,name)values(1002,'张三');
ERROR 1366 (HY000): Incorrect string value: '\xE5\xBC\xA0\xE4\xB8\x89' for column 'name' at row 1
```

因为默认情况下，创建表使用的是 `latin1`。如下：

```
Database changed
mysql> show create table t_emp;
+-----+-----+
| Table | Create Table |
+-----+-----+
| t_emp | CREATE TABLE `t_emp` (
  `id` int(11) DEFAULT NULL,
  `name` varchar(15) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1 |
+-----+-----+
1 row in set (0.00 sec)
```

操作2：修改字符集

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

在MySQL5.7或之前的版本中，在文件最后加上中文字符集配置：

```
1 | character_set_server=utf8
```



```
# For advice on how to change settings please see
# http://dev.mysql.com/doc/refman/5.7/en/server-configuration-defaults.html

[mysqld]
#
# 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
#
# 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 risks
symbolic-links=0

log-error=/var/log/mysql.log
pid-file=/var/run/mysql/mysql.pid
character_set_server=utf8

~
~
~
~
:wq!
```

操作3：重新启动MySQL服务

```
1 | systemctl restart mysqld
```

但是原库、原表的设定不会发生变化，参数修改只对新建的数据库生效。

2.已有库 &表字符集的变更

MySQL5.7版本中，以前创建的库，创建的表字符集还是 latin1。

```
mysql> show create table t_emp;
+-----+-----+
| Table | Create Table
+-----+-----+
| t_emp | CREATE TABLE `t_emp` (
  `id` int(11) DEFAULT NULL,
  `name` varchar(15) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1 |
+-----+-----+
1 row in set (0.00 sec)
```

修改已创建数据库的字符集

```
1 | alter database dbtest1 character set 'utf8';
```

修改已创建数据表的字符集

```
1 | alter table t_emp convert to character set 'utf8';
```

```
mysql> alter database dbtest1 character set 'utf8';
Query OK, 1 row affected (0.00 sec)

mysql> alter table t_emp convert to character set 'utf8';
Query OK, 1 row affected (0.04 sec)
Records: 1 Duplicates: 0 Warnings: 0

mysql> insert into t_emp(id,name)values(1002,'张三');
Query OK, 1 row affected (0.00 sec)
```

```
mysql> select * from t_emp;
```

id	name
1001	Tom
1002	张三

```
2 rows in set (0.00 sec)
```

```
mysql> show create table t_emp;
```

Table	Create Table
t_emp	CREATE TABLE `t_emp` (`id` int(11) DEFAULT NULL, `name` varchar(15) DEFAULT NULL) ENGINE=InnoDB DEFAULT CHARSET=utf8

```
1 row in set (0.00 sec)
```

注意：但是原有的数据如果是用非'utf8'编码的话，数据本身编码不会发生改变。已有数据需要导出或删除，然后重新插入。

5.2各级别的字符集

MySQL有4个级别的字符集和比较规则，分别是：

- 服务器级别
- 数据库级别
- 表级别
- 列级别

执行如下SQL语句：

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

```
mysql> show variables like 'character%';
+-----+-----+
| Variable_name | Value |
+-----+-----+
| character_set_client | utf8mb4 |
| character_set_connection | utf8mb4 |
| character_set_database | utf8mb4 |
| character_set_filesystem | binary |
| character_set_results | utf8mb4 |
| character_set_server | utf8mb4 |
| character_set_system | utf8mb3 |
| character_sets_dir | /usr/share/mysql-8.0/charsets/ |
+-----+-----+
8 rows in set (0.02 sec)
```

- character_set_server: 服务器级别的字符集
- character_set_database: 当前数据库的字符集
- character_set_client: 服务器解码请求时使用的字符集
- character_set_connection: 服务器处理请求时会把请求字符串从character_set_client转为character_set_connection
- character_set_results: 服务器向客户端返回数据时使用的字符集

1.服务器级别

- `character_set_server`: 服务器级别的字符集。

我们可以在启动服务器程序时通过启动选项或者在服务器程序运行过程中使用 `SET` 语句修改这两个变量的值。

比如我们可以在配置文件中这样写:

```
1 [server]
2 character_set_server=gbk # 默认字符集
3 collation_server=gbk_chinese_ci # 对应的默认的比较规则
```

当服务器启动的时候读取这个配置文件后这两个系统变量的值便修改了。

2.数据库级别

- character_set_database: 当前数据库的字符集

我们在创建和修改数据库的时候可以指定该数据库的字符集和比较规则，具体语法如下:

```
1 CREATE DATABASE 数据库名
2     [[DEFAULT] CHARACTER SET 字符集名称]
3     [[DEFAULT] COLLATE 比较规则名称];
4
5 ALTER DATABASE 数据库名
6     [[DEFAULT] CHARACTER SET 字符集名称]
7     [[DEFAULT] COLLATE 比较规则名称];
```

其中的DEFAULT可以省略，并不影响语句的语义。比如:

```

1 mysql> CREATE DATABASE charset_demo_db
2     -> CHARACTER SET gb2312
3     -> COLLATE gb2312_chinese_ci;
4 Query OK, 1 row affected (0.01 sec)

```

数据库的创建语句中也可以不指定字符集和比较规则，比如这样：

```

1 CREATE DATABASE 数据库名;

```

这样的话将使用服务器级别的字符集和比较规则作为数据库的字符集和比较规则。

3.表级别

我们也可以在创建和修改表的时候指定表的字符集和比较规则，语法如下：

```

1 CREATE TABLE 表名 (列的信息 )
2     [[DEFAULT] CHARACTER SET 字符集名称]
3     [[COLLATE 比较规则名称]]
4 ALTER TABLE 表名
5     [[DEFAULT] CHARACTER SET 字符集名称]
6     [[COLLATE 比较规则名称]

```

比方说我们在刚刚创建的 `charset_demo_db` 数据库中创建一个名为 `t` 的表，并指定这个表的字符集和比较规则：

```

1 mysql> CREATE TABLE t(
2     ->     col VARCHAR(10)
3     -> ) CHARACTER SET utf8 COLLATE utf8_general_ci;
4 Query OK, 0 rows affected (0.03 sec)

```

如果创建和修改表的语句中没有指明字符集和比较规则，将使用该表所在数据库的字符集和比较规则作为该表的字符集和比较规则。

4.列级别

对于存储字符串的列，同一个表中的不同的列也可以有不同的字符集和比较规则。我们在创建和修改列定义的时候可以指定该列的字符集和比较规则，语法如下：

```

1 CREATE TABLE 表名 (
2     列名 字符串类型 [[CHARACTER SET 字符集名称] [[COLLATE 比较规则名称],
3     其他列 ...
4 );
5 ALTER TABLE 表名 MODIFY 列名 字符串类型 [[CHARACTER SET 字符集名称] [[COLLATE 比较规则名称 ];

```

比如我们修改一下表 `t` 中列 `col` 的字符集和比较规则可以这么写：

```

1 mysql> ALTER TABLE t MODIFY col VARCHAR(10) CHARACTER SET gbk COLLATE
2     gbk_chinese_ci;
3 Query OK, 0 rows affected (0.04 sec)
4 Records: 0 Duplicates: 0 Warnings: 0

```

对于某个列来说，如果在创建和修改的语句中没有指明字符集和比较规则，将使用该列所在表的字符集和比较规则作为该列的字符集和比较规则。

提示

在转换列的字符集时需要注意，如果转换前列中存储的数据不能用转换后的字符集进行表示会发生错误。比方说原先列使用的字符集是utf8，列中存储了一些汉字，现在把列的字符集转换为ascii的话就会出错，因为ascii字符集并不能表示汉字字符。

5.小结

我们介绍的这4个级别字符集和比较规则的联系如下：

- 如果 创建或修改列时 没有显式的指定字符集和比较规则，则该列 默认用表的 字符集和比较规则
- 如果 创建表时 没有显式的指定字符集和比较规则，则该表 默认用数据库的 字符集和比较规则
- 如果 创建数据库时 没有显式的指定字符集和比较规则，则该数据库 默认用服务器的 字符集和比较规则

知道了这些规则之后，对于给定的表，我们应该知道它的各个列的字符集和比较规则是什么，从而根据这个列的类型来确定存储数据时每个列的实际数据占用的存储空间大小了。比方说我们向表 `t` 中插入一条记录：

```
1 mysql> INSERT INTO t(col) VALUES('我们');
2 Query OK, 1 row affected (0.00 sec)
3 mysql> SELECT * FROM t;
4 +-----+
5 | s      |
6 +-----+
7 | 我们   |
8 +-----+
9 1 row in set (0.00 sec)
```

首先列 `col` 使用的字符集是 `gbk`，一个字符 '我' 在 `gbk` 中的编码为 `0xCED2`，占用两个字节，两个字符的实际数据就占用4个字节。如果把该列的字符集修改为 `utf8` 的话，这两个字符就实际占用6个字节。

5.3字符集与比较规则（了解）

1.utf8与 utf8mb4

`utf8` 字符集表示一个字符需要使用1~4个字节，但是我们常用的一些字符使用1~3个字节就可以表示了。而字符集表示一个字符所用的最大字节长度，在某些方面会影响系统的存储和性能，所以设计MySQL的设计者偷偷的定义了两个概念：

- `utf8mb3`：阉割过的 `utf8` 字符集，只使用1~3个字节表示字符。
- `utf8mb4`：正宗的 `utf8` 字符集，使用1~4个字节表示字符。

在MySQL中 `utf8` 是 `utf8mb3` 的别名，所以之后在MySQL中提到 `utf8` 就意味着使用1~3个字节来表示一个字符。如果大家有使用4字节编码一个字符的情况，比如存储一些 `emoji` 表情，那请使用 `utf8mb4`。

此外，通过如下指令可以查看MySQL支持的字符集：

```
1 SHOW CHARSET;
2 # 或者
3 SHOW CHARACTER SET;
```

Charset	Description	Default collation	Maxlen
big5	Big5 Traditional Chinese	big5_chinese_ci	2
dec8	DEC West European	dec8_swedish_ci	1
cp850	DOS West European	cp850_general_ci	1
hp8	HP West European	hp8_english_ci	1
koi8r	KOI8-R Relcom Russian	koi8r_general_ci	1
latin1	cp1252 West European	latin1_swedish_ci	1
latin2	ISO 8859-2 Central European	latin2_general_ci	1
swe7	7bit Swedish	swe7_swedish_ci	1
ascii	US ASCII	ascii_general_ci	1
ujis	EUC-JP Japanese	ujis_japanese_ci	3
sjis	Shift-JIS Japanese	sjis_japanese_ci	2
hebrew	ISO 8859-8 Hebrew	hebrew_general_ci	1
tis620	TIS620 Thai	tis620_thai_ci	1
euckr	EUC-KR Korean	euckr_korean_ci	2
koi8u	KOI8-U Ukrainian	koi8u_general_ci	1
gb2312	GB2312 Simplified Chinese	gb2312_chinese_ci	2
greek	ISO 8859-7 Greek	greek_general_ci	1
cp1250	Windows Central European	cp1250_general_ci	1
gbk	GBK Simplified Chinese	gbk_chinese_ci	2
latin5	ISO 8859-9 Turkish	latin5_turkish_ci	1
armscii8	ARMSCII-8 Armenian	armscii8_general_ci	1
utf8	UTF-8 Unicode	utf8_general_ci	3
ucs2	UCS-2 Unicode	ucs2_general_ci	2
cp866	DOS Russian	cp866_general_ci	1
keybcs2	DOS Kamenicky Czech-Slovak	keybcs2_general_ci	1

2.比较规则

上表中，MySQL版本一共支持41种字符集，其中的 Default collation 列表示这种字符集中一种默认的比较规则，里面包含着该比较规则主要作用于哪种语言，比如 utf8_polish_ci 表示以波兰语的规则比较，utf8_spanish_ci 是以西班牙语的规则比较，utf8_general_ci 是一种通用的比较规则。

后缀表示该比较规则是否区分语言中的重音、大小写。具体如下：

后缀	英文释义	描述
_ai	accent insensitive	不区分重音
_as	accent sensitive	区分重音
_ci	case insensitive	不区分大小写
_cs	case sensitive	区分大小写
_bin	binary	以二进制方式比较

最后一列 Maxlen，它代表该种字符集表示一个字符最多需要几个字节。

这里把常见的字符集和对应的Maxlen显式如下：

字符集名称	Maxlen
ascii	1
latin1	1

字符集名称	Maxlen
gb2312	2
gbk	2
utf8	3
utf8mb4	4

常用操作1:

```

1  #查看 GBK字符集的比较规则
2  SHOW COLLATION LIKE 'gbk%';
3  #查看 UTF-8字符集的比较规则
4  SHOW COLLATION LIKE 'utf8%';

```

常用操作2:

```

1  #查看服务器的字符集和比较规则
2  SHOW VARIABLES LIKE '%_server';
3  #查看数据库的字符集和比较规则
4  SHOW VARIABLES LIKE '%_database';
5  #查看具体数据库的字符集
6  SHOW CREATE DATABASE dbtest1;
7  #修改具体数据库的字符集
8  ALTER DATABASE dbtest1 DEFAULT CHARACTER SET 'utf8' COLLATE
   'utf8_general_ci';

```

说明1:

utf8_unicode_ci和utf8_general_ci对中、英文来说没有实质的差别。

utf8_general_ci 校对速度快，但准确度稍差。

utf8_unicode_ci 准确度高，但校对速度稍慢。

一般情况，用utf8_general_ci就够了，但如果你的应用有德语、法语或者俄语，请一定使用utf8_unicode_ci。

说明2:

修改了数据库的默认字符集和比较规则后，原来已经创建的表格的字符集和比较规则并不会改变，如果需要，那么需单独修改。

常用操作3:

```

1  #查看表的字符集
2  show create table employees;
3  #查看表的比较规则
4  show table status from atguigudb like 'employees';
5  #修改表的字符集和比较规则
6  ALTER TABLE emp1 DEFAULT CHARACTER SET 'utf8' COLLATE 'utf8_general_ci';

```

5.4请求到响应过程中字符集的变化

我们知道从客户端发往服务器的请求本质上就是一个字符串，服务器向客户端返回的结果本质上也是一个字符串，而字符串其实是使用某种字符集编码的 二进制数据。这个字符串可不是使用一种字符集的编码方式一条道走到黑的，从发送请求到返回结果这个过程中伴随着 多次字符集的转换，在这个过程中会用到3个系统变量，我们先把它们写出来看一下：

系统变量	描述
character_set_client	服务器解码请求时使用的字符集
character_set_connection	服务器处理请求时会把请求字符串从 character_set_client转为 character_set_connection
character_set_results	服务器向客户端返回数据时使用的字符集

这几个系统变量在我的计算机上的默认值如下（不同操作系统的默认值可能不同）：

```
mysql> show variables like 'character%';
+-----+-----+
| Variable_name | Value |
+-----+-----+
| character_set_client | utf8mb4 |
| character_set_connection | utf8mb4 |
| character_set_database | utf8mb4 |
| character_set_filesystem | binary |
| character_set_results | utf8mb4 |
| character_set_server | utf8mb4 |
| character_set_system | utf8mb3 |
| character_sets_dir | /usr/share/mysql-8.0/charsets/ |
+-----+-----+
8 rows in set (0.02 sec)
```

为了体现出字符集在请求处理过程中的变化，我们这里特意修改一个系统变量的值：

```
1 | mysql> set character_set_connection = gbk;
2 | Query OK, 0 rows affected (0.00 sec)
```

现在假设我们客户端发送的请求是下边这个字符串：

```
1 | SELECT * FROM t WHERE s = '我';
```

为了方便大家理解这个过程，我们只分析字符 '我' 在这个过程中字符集的转换。

现在看一下在请求从发送到结果返回过程中字符集的变化：

1. 客户端发送请求所使用的字符集

一般情况下客户端所使用的字符集和当前操作系统一致，不同操作系统使用的字符集可能不一样，如下：

- 类 Unix 系统使用的是 utf8
- windows 使用的是 gbk

当客户端使用的是 `utf8` 字符集，字符 `'我'` 在发送给服务器的请求中的字节形式就是：

`0xE68891`

提示

如果你使用的是可视化工具，比如navicat之类的，这些工具可能会使用自定义的字符集来编码发送到服务器的字符串，而不采用操作系统默认的字符集（所以在学习的时候还是尽量用命令行窗口）。

2. 服务器接收到客户端发送来的请求其实是一串二进制的字节，它会认为这串字节采用的字符集是 `character_set_client`，然后把这串字节转换为 `character_set_connection` 字符集编码的字符。

由于我的计算机上 `character_set_client` 的值是 `utf8`，首先会按照 `utf8` 字符集对字节串 `0xE68891` 进行解码，得到的字符串就是 `'我'`，然后按照 `character_set_connection` 代表的字符集，也就是 `gbk` 进行编码，得到的结果就是字节串 `0xCED2`。

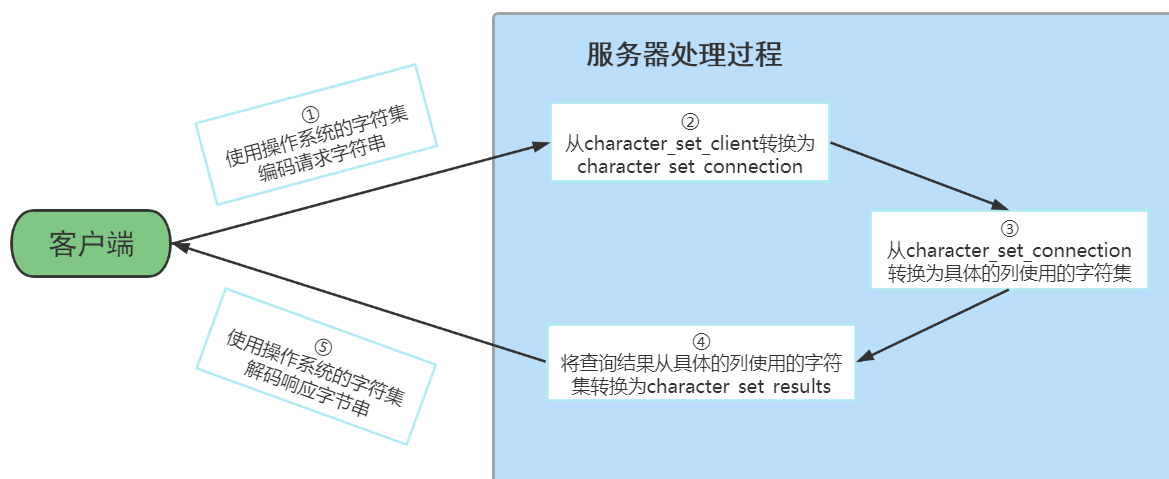
3. 因为表 `t` 的列 `col` 采用的是 `gbk` 字符集，与 `character_set_connection` 一致，所以直接到列中找字节值为 `0xCED2` 的记录，最后找到了一条记录。

提示

如果某个列使用的字符集和 `character_set_connection` 代表的字符集不一致的话，还需要进行一次字符集转换。

4. 上一步骤找到的记录中的 `col` 列其实是一个字节串 `0xCED2`，`col` 列是采用 `gbk` 进行编码的，所以首先会将这个字节串使用 `gbk` 进行解码，得到字符串 `'我'`，然后再把这个字符串使用 `character_set_results` 代表的字符集，也就是 `utf8` 进行编码，得到了新的字节串：`0xE68891`，然后发送给客户端。
5. 由于客户端是用的字符集是 `utf8`，所以可以顺利的将 `0xE68891` 解释成字符 `我`，从而显示到我们的显示器上，所以我们人类也读懂了返回的结果。

总结图示如下：



从这个分析中我们可以得出这么几点需要注意的地方：

- 服务器认为客户端发送过来的请求是用 `character_set_client` 编码的。
假设你的客户端采用的字符集和 `character_set_client` 不一样的话，这就会出现识别不准确的情况。比如我的客户端使用的是 `utf8` 字符集，如果把系统变量 `character_set_client` 的值设置为 `ascii` 的话，服务器可能无法理解我们发送的请求，更别谈处理这个请求了。
- 服务器将把得到的结果集使用 `character_set_results` 编码后发送给客户端。

假设你的客户端采用的字符集和 `character_set_results` 不一样的话，这就可能会出现客户端无法解码结果集的情况，结果就是在你的屏幕上出现乱码。比如我的客户端使用的是 `utf8` 字符集，如果把系统变量 `character_set_results` 的值设置为 `ascii` 的话，可能会产生乱码。

- `character_set_connection` 只是服务器在将请求的字节串从 `character_set_client` 转换为 `character_set_connection` 时使用，一定要注意，该字符集包含的字符范围一定涵盖请求中的字符，要不然会导致有的字符无法使用 `character_set_connection` 代表的字符集进行编码。

经验：

开发中通常把 `character_set_client`、`character_set_connection`、`character_set_results` 这三个系统变量设置成和客户端使用的字符集一致的情况，这样减少了很多无谓的字符集转换。为了方便我们设置，MySQL提供了一条非常简便的语句：

```
1 SET NAMES 字符集名;
```

这一条语句产生的效果和我们执行这3条的效果是一样的：

```
1 SET character_set_client=字符集名;
2 SET character_set_connection = 字符集名;
3 SET character_set_results = 字符集名;
```

比方说我的客户端使用的是 `utf8` 字符集，所以需要把这几个系统变量的值都设置为 `utf8`：

```
1 mysql> SET NAMES utf8;
```

另外，如果你想在启动客户端的时候就把 `character_set_client`、`character_set_connection`、`character_set_results` 这三个系统变量的值设置成一样的，那我们可以在启动客户端的时候指定一个叫 `default-character-set` 的启动选项，比如在配置文件里可以这么写：

```
1 [client]
2 default-character-set=utf8
```

它起到的效果和执行一遍 `SET NAMES utf8` 是一样一样的，都会将那三个系统变量的值设置成 `utf8`。

6.SQL大小写规范

6.1 Windows和Linux平台区别

在 SQL 中，关键字和函数名是不用区分字母大小写的，比如 `SELECT`、`WHERE`、`ORDER`、`GROUP BY` 等关键字，以及 `ABS`、`MOD`、`ROUND`、`MAX` 等函数名。

不过在 SQL 中，你还是要确定大小写的规范，因为在 Linux 和 Windows 环境下，你可能会遇到不同的大小写问题。`windows` 系统默认大小写不敏感，但是 `linux` 系统是大小写敏感的。

通过如下命令查看：

```
1 SHOW VARIABLES LIKE '%lower_case_table_names%'
```

- Windows 系统下：

```
mysql> SHOW VARIABLES LIKE '%lower_case_table_names%';
+-----+-----+
| Variable_name | Value |
+-----+-----+
| lower_case_table_names | 1      |
+-----+-----+
1 row in set, 1 warning (0.01 sec)
```

- Linux系统下:

```
mysql> SHOW VARIABLES LIKE '%lower_case_table_names%';
+-----+-----+
| Variable_name | Value |
+-----+-----+
| lower_case_table_names | 0      |
+-----+-----+
1 row in set (0.01 sec)
```

- lower_case_table_names参数值的设置:
 - 默认为 0, 大小写敏感。
 - 设置1, 大小写不敏感。创建的表, 数据库都是以小写形式存放在磁盘上, 对于sql语句都是转换为小写对表和数据库进行查找。
 - 设置2, 创建的表和数据库依据语句上格式存放, 凡是查找都是转换为小写进行。
- 两个平台上SQL大小写的区别具体来说:

MySQL在Linux下数据库名、表名、列名、别名大小写规则是这样的:

1. 数据库名、表名、表的别名、变量名是严格区分大小写的;
2. 关键字、函数名称在SQL中不区分大小写;
3. 列名 (或字段名) 与列的别名 (或字段别名) 在所有的情况下均是忽略大小写的;

MySQL在Windows的环境下全部不区分大小写

6.2Linux下大小写规则设置

当想设置为大小写不敏感时, 要在 `my.cnf` 这个配置文件 [mysqld]中加入

`lower_case_table_names=1`, 然后重启服务器。

- 但是要在重启数据库实例之前就需要将原来的数据库和表转换为小写, 否则将找不到数据库名。
- 此参数适用于MySQL5.7。在MySQL8下禁止在重新启动 MySQL服务时将 `lower_case_table_names` 设置成不同于初始化 MySQL服务时设置的 `lower_case_table_names` 值。如果非要将MySQL8设置为大小写不敏感, 具体步骤为:

1. 停止 MySQL服务
2. 删除数据目录, 即删除 `/var/lib/mysql` 目录
3. 在 MySQL配置文件 (`/etc/my.cnf`) 中添加 `lower_case_table_names=1`
4. 启动 MySQL服务

注意: 在进行数据库参数设置之前, 需要掌握这个参数带来的影响, 切不可盲目设置。

6.3 SQL编写建议

如果你的变量名命名规范没有统一，就可能产生错误。这里有一个有关命名规范的建议：

1. 关键字和函数名称全部大写；
2. 数据库名、表名、表别名、字段名、字段别名等全部小写；
3. SQL语句必须以分号结尾。

数据库名、表名和字段名在 LinuxMySQL环境下是区分大小写的，因此建议你统一这些字段的命名规则，比如全部采用小写的方式。

虽然关键字和函数名称在 SQL中不区分大小写，也就是如果小写的话同样可以执行。但是同时将关键词和函数名称全部大写，以便于区分数据库名、表名、字段名。

7.sql_mode的合理设置

7.1 介绍

sql_mode会影响MySQL支持的SQL语法以及它执行的 `数据验证检查`。通过设置sql_mode，可以完成不同严格程度的数据校验，有效地保障数据准确性。

MySQL服务器可以在不同的SQL模式下运行，并且可以针对不同的客户端以不同的方式应用这些模式，具体取决于sql_mode系统变量的值。

MySQL5.6和MySQL5.7默认的sql_mode模式参数是不一样的：

- 5.6的mode默认值为空（即：`NO_ENGINE_SUBSTITUTION`），其实表示的是一个空值，相当于没有什么模式设置，可以理解为 `宽松模式`。在这种设置下是可以允许一些非法操作的，比如允许一些非法数据的插入。
- 5.7的mode是 `STRICT_TRANS_TABLES`，也就是 `严格模式`。用于进行数据的严格校验，错误数据不能插入，报error（错误），并且事务回滚。

7.1 宽松模式 vs 严格模式

宽松模式：

如果设置的是宽松模式，那么我们在插入数据的时候，即便是给了一个错误的的数据，也可能被接受，并且不报错。

举例：我在创建一个表时，该表中有一个字段为name，给name设置的字段类型时 `char(10)`，如果我在插入数据的时候，其中name这个字段对应的有一条数据的 `长度超过了 10`，例如 `'1234567890abc'`，超过了设定的字段长度10，那么不会报错，并且取前10个字符存上，也就是说你这个数据被存为了 `'1234567890'`，而 `'abc'` 就没有了。但是，我们给的这条数据是错误的，因为超过了字段长度，但是并没有报错，并且mysql自行处理并接受了，这就是宽松模式的效果。

应用场景：通过设置sql mode为宽松模式，来保证大多数 sql符合标准的sql语法，这样应用在不同数据库之间进行 `迁移` 时，则不需要对业务sql进行较大的修改。

严格模式：

出现上面宽松模式的错误，应该报错才对，所以MySQL5.7版本就将sql_mode默认值改为了严格模式。所以在 `生产等环境` 中，我们必须采用的是严格模式，进而 `开发、测试环境` 的数据库也必须设置，这样在开发测试阶段就可以发现问题。并且我们即便是用的MySQL5.6，也应该自行将其改为严格模式。

开发经验：MySQL等数据库总想把关于数据的所有操作都自己包揽下来，包括数据的校验，其实开发中，我们应该在自己开发的项目程序级别将这些校验给做了，虽然写项目的时候麻烦了一些步骤，但是这样做之后，我们在进行数据库迁移或者在项目的迁移时，就会方便很多。

改为严格模式后可能会存在的问题：

若设置模式中包含了 `NO_ZERO_DATE`，那么MySQL数据库不允许插入零日期，插入零日期会抛出错误而不是警告。例如，表中含字段TIMESTAMP列（如果未声明为NULL或显示DEFAULT子句）将自动分配DEFAULT '0000-00-0000:00:00'（零时间戳），这显然是不满足 `sql_mode`中的`NO_ZERO_DATE`而报错。

7.2宽松模式再举例

宽松模式举例1：

```
1 select *from employees group by department_id limit 10;
2 set sql_mode = ONLY_FULL_GROUP_BY;
3 select * from employees groupby department_id limit10;
```

```
mysql> select * from emp group by ename limit 10;
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| id    | empno | ename  | job      | mgr | hiredate | sal      | comm     | deptno |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 262691 | 362692 | aAAaAA | SALESMAN | 1   | 2017-03-07 | 2000.00 | 400.00 | 100    |
| 105416 | 205417 | aaaaab | SALESMAN | 1   | 2017-03-07 | 2000.00 | 400.00 | 100    |
| 387366 | 487367 | AaAAbE | SALESMAN | 1   | 2017-03-07 | 2000.00 | 400.00 | 108    |
| 211538 | 311539 | aaaabf | SALESMAN | 1   | 2017-03-07 | 2000.00 | 400.00 | 104    |
| 255851 | 355852 | AAaABg | SALESMAN | 1   | 2017-03-07 | 2000.00 | 400.00 | 100    |
| 55039  | 155040 | aAAaBJ | SALESMAN | 1   | 2017-03-07 | 2000.00 | 400.00 | 105    |
| 163126 | 263127 | AaAAcI | SALESMAN | 1   | 2017-03-07 | 2000.00 | 400.00 | 102    |
| 160333 | 260334 | aAAaCJ | SALESMAN | 1   | 2017-03-07 | 2000.00 | 400.00 | 107    |
| 294502 | 394503 | aAAaCK | SALESMAN | 1   | 2017-03-07 | 2000.00 | 400.00 | 108    |
| 2696   | 102697 | aaaacI | SALESMAN | 1   | 2017-03-07 | 2000.00 | 400.00 | 109    |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
10 rows in set (0.00 sec)

mysql> set sql_mode =ONLY_FULL_GROUP_BY;
Query OK, 0 rows affected (0.00 sec)

mysql> select * from emp group by ename limit 10;
ERROR 1055 (42000): 'mytest.emp.id' isn't in GROUP BY
mysql>
```

宽松模式举例2：

```
mysql> desc t1;
+-----+-----+-----+-----+-----+-----+
| Field | Type          | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| id    | int(11)       | NO   | PRI | NULL    |       |
| name  | varchar(2)    | YES  |     | NULL    |       |
| age   | smallint(6)   | YES  |     | NULL    |       |
+-----+-----+-----+-----+-----+-----+
3 rows in set (0.01 sec)

mysql> set sql_mode = '';
Query OK, 0 rows affected (0.00 sec)
```

```
mysql> insert into t1 values(1,'aa','aaa');
Query OK, 1 row affected, 1 warning (0.01 sec)
```

```
mysql> show warnings;
```

Level	Code	Message
Warning	1366	Incorrect integer value: 'aaa' for column 'age' at row 1

1 row in set (0.00 sec)

```
mysql> select * from t1;
```

id	name	age
1	aa	0

1 row in set (0.00 sec)

设置 sql_mode 模式为 STRICT_TRANS_TABLES，然后插入数据：

```
mysql> set sql_mode = 'STRICT_TRANS_TABLES';
Query OK, 0 rows affected, 1 warning (0.00 sec)
```

```
mysql> insert into t1 values(2,'bb','bbb');
ERROR 1366 (HY000): Incorrect integer value: 'bbb' for column 'age' at row 1
mysql>
```

7.3 模式查看和设置

- 查看当前的 sql_mode

```
1 select @@session.sql_mode
2 select @@global.sql_mode
3 #或者
4 show variables like 'sql_mode';
```

```
mysql> select @@session.sql_mode;
+-----+
| @@session.sql_mode |
+-----+
| ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLES,NO_ZERO_IN_DATE,NO_ZERO_DATE,ERROR_FOR_DIVISION_BY_ZERO,NO_ENGINE_SUBSTITUTION |
+-----+
1 row in set (0.00 sec)
```

```
mysql> select @@global.sql_mode;
+-----+
| @@global.sql_mode |
+-----+
| ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLES,NO_ZERO_IN_DATE,NO_ZERO_DATE,ERROR_FOR_DIVISION_BY_ZERO,NO_ENGINE_SUBSTITUTION |
+-----+
1 row in set (0.00 sec)
```

- 临时设置方式：设置当前窗口中设置 sql_mode

```
1 SET GLOBAL sql_mode = 'modes...'; #全局
2 SET SESSION sql_mode = 'modes...'; #当前会话
```

举例：

```
1 #改为严格模式。此方法只在当前会话中生效，关闭当前会话就不生效了。
2 set SESSION sql_mode = 'STRICT_TRANS_TABLES';
```

```
1 #改为严格模式。此方法在当前服务中生效，重启 MySQL服务后失效。
2 set GLOBAL sql_mode='STRICT_TRANS_TABLES';
```

- 永久设置方式：在/etc/my.cnf中配置sql_mode

在my.cnf文件(windows系统是my.ini文件)，新增：

```
1 [mysqld]
2 sql_mode=ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLES,NO_ZERO_IN_DATE,NO_ZERO_DATE,E
  RROR_FOR _DIVISION_BY_ZERO,NO_ENGINE_SUBSTITUTION
```

然后 重启 MySQL。

当然生产环境上是禁止重启MySQL服务的，所以采用 临时设置方式 +永久设置方式 来解决线上的问题，那么即便是有一天真的重启了MySQL服务，也会永久生效了。

- 测试数据：

```
1 CREATE TABLE mytb12(id INT,NAME VARCHAR(16),age INT,dept INT);
2
3 INSERT INTO mytb12 VALUES(1,'zhang3',33,101);
4 INSERT INTO mytb12 VALUES(2,'li4',34,101);
5 INSERT INTO mytb12 VALUES(3,'wang5',34,102);
6 INSERT INTO mytb12 VALUES(4,'zhao6',34,102);
7 INSERT INTO mytb12 VALUES(5,'tian7',36,102);
8
9 #查询每个部门年龄最大的人
10 SELECT NAME,dept,MAX(age) FROM mytb12 GROUP BY dept ;
```

运行结果：报错

7.5 sql_mode常用值

下面列出MySQL中最重要的3种模式：

模式	含义
ONLY_FULL_GROUP_BY	对于GROUP BY聚合操作，如果在SELECT中的列，没有在GROUP BY中出现，那么这个SQL是不合法的，因为列不在GROUP BY从句中。
NO_AUTO_VALUE_ON_ZERO	该值影响自增长列的插入。默认设置下，插入0或NULL代表生成下一个自增长值。如果用户希望插入的值为0，而该列又是自增长的，那么这个选项就有用了。
STRICT_TRANS_TABLES	严格模式，在该模式下，如果一个值不能插入到一个事务表中，则中断当前的操作，对非事务表不做限制。
NO_ZERO_IN_DATE	在严格模式下，不允许日期和月份为零。
NO_ZERO_DATE	该值会使得MySQL数据库不允许插入零日期，插入零日期会抛出错误而不是警告。

模式	含义
ERROR_FOR_DIVISION_BY_ZERO	在INSERT或UPDATE过程中，如果数据被零除，则产生错误而非警告。如果未给出该模式，那么数据被零除时MySQL返回NULL
NO_AUTO_CREATE_USER	禁止GRANT创建密码为空的用户
NO_ENGINE_SUBSTITUTION	如果需要的存储引擎被禁用或未编译，那么抛出错误。不设置此值时，用默认的存储引擎替代，并抛出一个异常
PIPES_AS_CONCAT	将双竖线符号视为字符串的连接操作符而非“或”运算符，这个在Oracle中缺省支持通过来双竖线符号实现字符串拼接，不过在MySQL默认不支持，需要设置sql_mode模式为PIPES_AS_CONCAT才可以
ANSI_QUOTES	启用ANSI_QUOTES后，不能用双引号来引用字符串，因为它被解释为识别符