# MySQL环境搭建

## 安装

### Linux安装

由于数据库一般部署在服务端，而服务端的Linux一般是CentOS，所以本次笔记基于CentOS，在linux环境mysql的配置文件为：my.cnf。步骤：

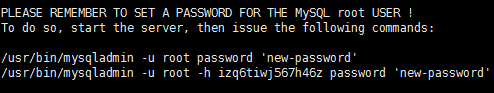
①先准备Mysql的服务端和客户端的安装包



②用rpm -ivh + 文件名 去安装mysql的服务端和客户端。安装服务端的时

候，会有这样一段提示，意思是让我们给Mysql的root用户设置一个密码，

初始是没有密码的



③测试是否安装成功。用mysqladmin --version命令去查看是否有如下的提示



### Window安装

推荐使用解压版，下载地址：<https://dev.mysql.com/downloads/mysql/>。

将其解压到一个目录，在window环境mysql的配置文件为my.ini，刚解压完是没有my.ini的，需要我们手动加上一个，配置文件例子为：

[mysql]

# 设置mysql客户端默认字符集

default-character-set=utf8

[mysqld]

#设置3306端口

port = 3306

# 设置mysql的安装目录(根据实际安装目录配置)

basedir=E:\database\mysql-5.7.19

# 设置mysql数据库的数据的存放目录(根据实际安装目录配置)

datadir=E:\database\mysql-5.7.19\data

# 允许最大连接数

max\_connections=200

# 服务端使用的字符集默认为8比特编码的latin1字符集

character-set-server=utf8

# 创建新表时将使用的默认存储引擎

default-storage-engine=INNODB

# 取消权限设置，即开启免密登陆

skip-grant-tables

接着进入到bin目录下，打开DOS窗口，执行命令：

mysqld --install mysql ##将mysql安装成服务，服务名为mysql

mysqld --initialize ##初始化mysql

net start mysql ##启动mysql

服务启动后，我们第一次登陆可以免密（my.ini配置了取消权限），登陆到mysql后使用命令修改root用户的命令：

mysql –u root -p ##免密登陆mysql

## 修改root用户的密码为root

update MySQL.user set authentication\_string=password('root') where user='root';

flush privileges; ##刷新权限

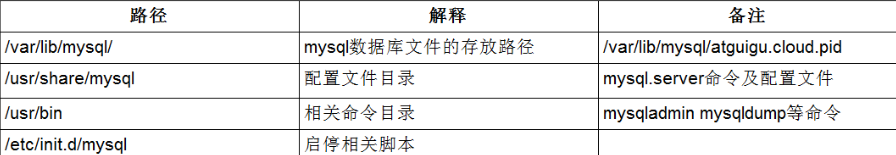
一切设置好后，修改my.ini配置文件，将skip-grant-tables配置删掉并重启。

## 配置

### 安装路径

在Linux系统中输入ps -ef| grep mysql命令查看mysql的安装路径

安装路径总结为以下的表格：



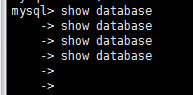
/var/lib/mysql ------建数据库，建表（实际的硬盘数据）都是保存在这个目录下

/usr/share/mysql---从这里拷贝出Mysql配置文件的副本，然后进行修改

/usr/bin --------------关于mysql的一些命令

### 自启动设置

使用which mysql命令可以查找mysql的安装位置。如果要让Mysql随机启动，可以使用chkconfig mysql on命令。在mysql命令行输入命令没反应，如下图所示，注意要在语句末尾加";"或者"/g"才行



### 修改配置文件的位置

一般不对Mysql自带的配置文件做任何修改，而是将其拷贝出来再修改，

保证原件不动，其实不只mysql这样做，其他框架或者系统都该这样做。

Mysql5.5版本，配置文件放在：/usr/share/mysql/my-huge.cnf，

Mysql5.6版本，配置文件放在：/usr/share/mysql/my-default.cnf；

如果是其他版本，配置文件不是以此命名的，自行谷歌查找

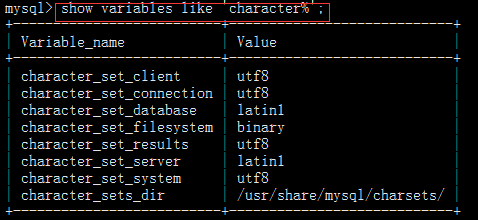
拷贝命令：cp /usr/share/mysql/my-huge.cnf /etc/my.cnf

这里有个问题：上面拷贝my-huge.cnf到/etc/my.cnf，为什么仅仅拷贝就可以了，都不用重新指定新的配置文件？~~2017.10.24不懂

### 修改默认字符集

Mysql原设置的字符集不支持中文的数据，插入的时候，会变成乱码，所以我们需要自己修改字符集，首先，查看Mysql此时的字符集为：

用命令： show variables like 'character%'; 或者show variables like '%char%';



可以看到，character\_set\_database和character\_set\_server的设置还是latin1，

character\_set\_database是Mysql存储数据到硬盘上的编码设置

character\_set\_server是Mysql服务端的字符集

因为之前已经拷贝了my-huge.cnf的副本，所以在副本文件/etc/my.cnf修改即可

黑色字体是配置文件自带的信息，红色字体是修改字符集需要的配置，总共要修改3个位置的配置(添加下面的红色字体到相应位置)：

[client]

#password= your\_password

port= 3306

socket= /var/lib/mysql/mysql.sock

default-character-set=utf8

[mysqld]

port= 3306

character\_set\_server=utf8

character\_set\_client=utf8

collation-server=utf8\_general\_ci

socket= /var/lib/mysql/mysql.sock

skip-external-locking

key\_buffer\_size = 384M

max\_allowed\_packet = 1M

table\_open\_cache = 512

sort\_buffer\_size = 2M

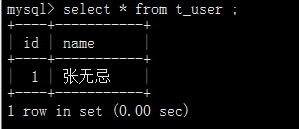
read\_buffer\_size = 2M

[mysql]

no-auto-rehash

default-character-set=utf8

修改完成后，保存配置并重启数据库，如果在此之前建立的数据库，因为字符集已经设置好了，所以就算修改配置文件的字符集，仍然是中文乱码。只有在修改配置文件之后建立的数据库，字符集修改才会起作用！



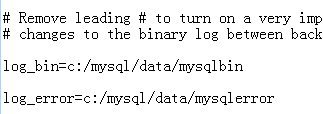
## 配置文件

在window系统下，配置文件是my.ini；在Linux系统下，配置文件是/etc/my.cnf

mysql的配置文件大体上分为：

①二进制日志log-bin--------默认关闭，记录主机的信息，用于主从复制

②错误日志log-error---------默认关闭，记录错误日志和启动、关闭的详细信息



③查询日志log-----------------默认关闭，记录查询的sql语句

④数据文件: 1、 .frm文件----存放表结构

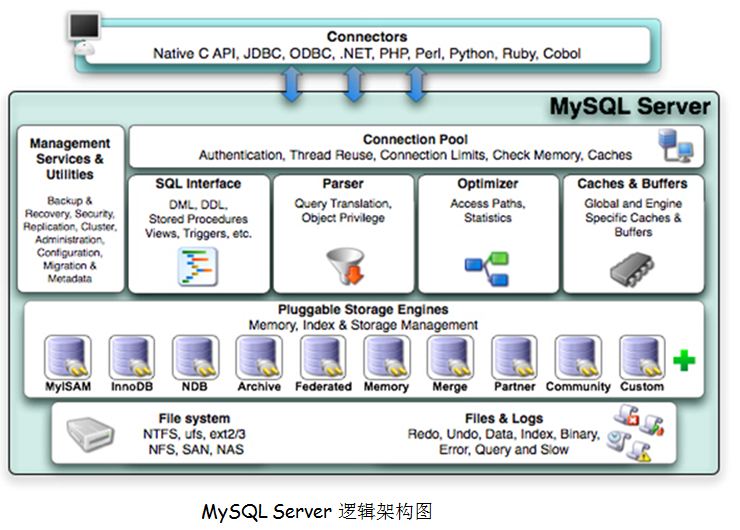
2、.myd文件----存放表数据结构

3、.myi文件-----存放表索引



# MySQL架构

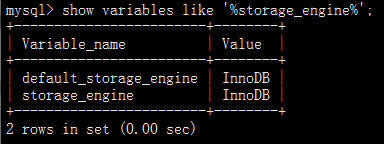
mysql的架构图： 最大特点，可拔插式的储存引擎



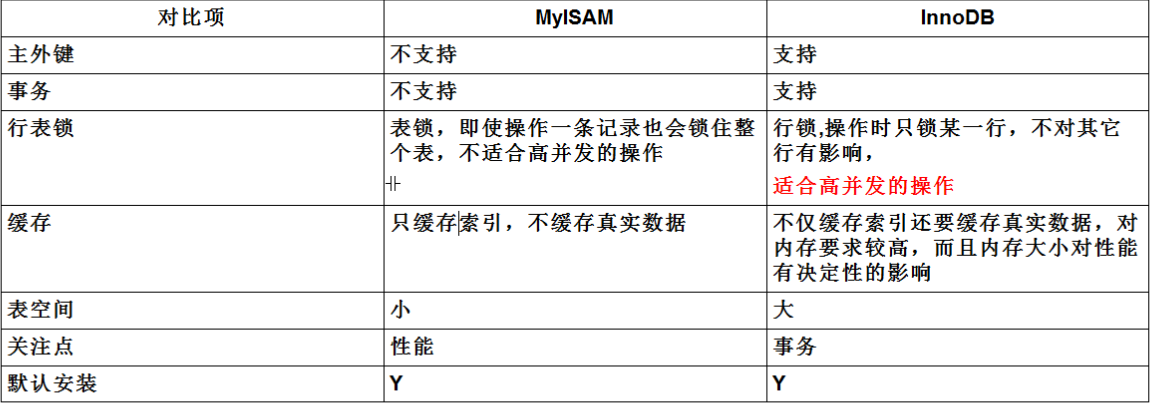
## 存储引擎

可以使用mysql的命令，查看当前数据库使用的引擎：

show variables like '%storage\_engine%'



在MySQL中用的最多的2种储存引擎：MyISAM和InnoDB，他们的区别：



# SQL语法

## 连接查询

这里讨论的是2个表之间的情况，3表及以上的连接，目测也是在2表的基础上，跟第3个表在进行连接 ，假设现在有员工表和部门表如下，

t\_emp表：

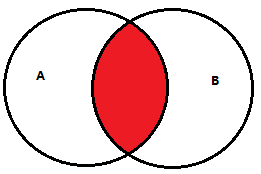
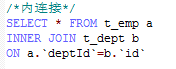
|  |  |  |
| --- | --- | --- |
| num | name | deptId |
| 1 | 小明 | 10 |
| 2 | 小棋 | 20 |
| 3 | 小军 | 30 |
| 4 | 小狗 | 40 |
| 5 | 小强 | 110 |

t\_dept表：

|  |  |
| --- | --- |
| Id | deptName |
| 10 | 研发部 |
| 20 | 人事部 |
| 30 | 财政部 |
| 40 | 测试部 |
| 50 | 后勤部 |

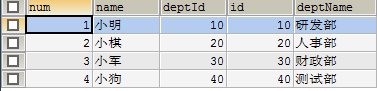
### 内连接

内连接，关键字是inner join，取左表和右表的共有部分，如下图的红色部分

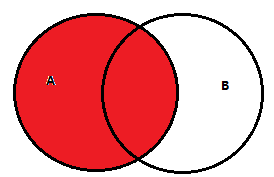
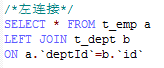
on关键字的作用是：两张表以什么条件关联起来，本例中是以部门ID关联起来

查询后的结果：



### 左连接

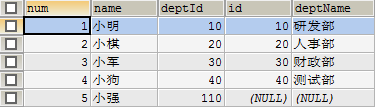
左连接，关键字是left join，关键字左边的表为主表，它的数据会全部查询出来，右表与其配对的数据就找出来，没有配对的就直接Null，如下面的红色部分

查询后的结果：

写在left join左边的表时t\_emp表，所以t\_emp表的数据会全部找出来，

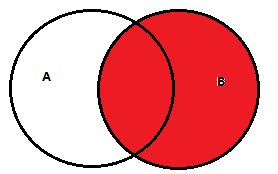
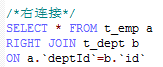
然后拿右表的数据来配对，符合on条件的都找出来，比如说部门号(id)为10的有2个，则这2个都会匹配，剩下A独有而B没有的，都以null展示



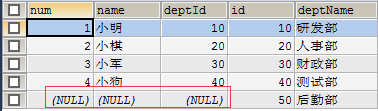
### 右连接

右连接，跟左连接相似，它的关键字是right join，它的主表是关键字右边的表，

所以它会把右表的数据全部找出来，然后左表与其配对，找出能匹配的

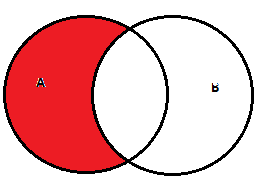
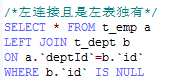
 

写在关键字right join右边的表时t\_dept表，所以会以这个表为主表，找出它的所有数据，然后t\_emp表与其配对，查询的结果为：



### 左连接且左表独有

左连接且左表独有，就是在左连接的基础上，去掉左表和右表共有的部分，如下面的红色部分：

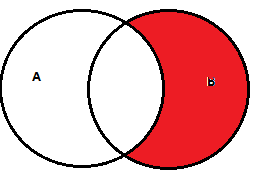
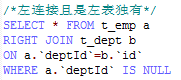
 

因为左连接的时候，会把左表的全部数据都展示，右表没有与其配对的就以

null存在，所以要找左表的独有，就是这些以Null展示的列，所以加一个where条件，查询结果为：



### 右连接且右表独有

同样的，在右连接的基础，加个where条件，找到右连接查询结果的那些null字段，如下面的红色部分所示：  
 

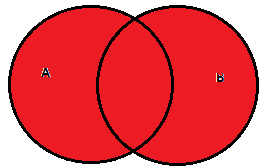
查询结果为：



### 全连接

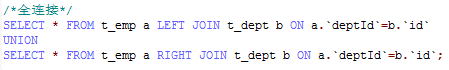
全连接，关键字是full outer join ，但是很不巧，这个关键字Mysql识别不了，

oracle可以识别。全连接=左表独有+右表独有+共有，如下图所示：

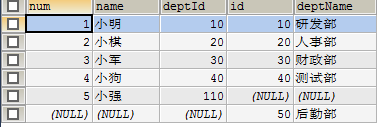


因为Mysql识别不了，我们采用左连接+右连接，然后去掉中间重复的数据，

就是全连接的结果，mysql中，2个结果集的并集且去掉重复的关键字是union

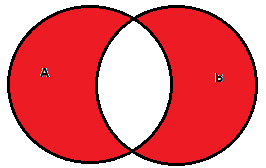


查询结果为：

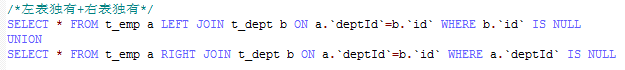


### 全连接去掉共有数据

同样的道理，全连接去掉共有数据，就是左表独有+右表独有，如下图所示：



说白了，就是：左表独有的+右表独有的，同样采用union关键字，SQL语句为：



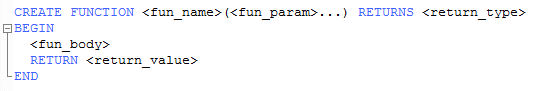
查询后的结果，就是



## 自定义函数

除了可以使用mysql提供的内置函数外， 也可以自定义函数

### 语法结构



<fun\_name>：是指函数名称 - 必须

<fun\_param>：是指函数的参数，可以指定多个 – 非必须

<return\_type>：指函数的返回类型(mysql的数据类型) – 必须，如：INT、

VARCHAR(150)...当返回VARCHAR类型需要指定大小

<fun\_body>：是指函数体，函数的处理逻辑 – 必须

<retun\_value>：是指返回值，mysql的函数必须都要有返回值

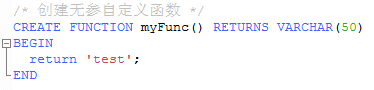
**PS：**

1、函数必须要定义返回值类型，必须要有返回值

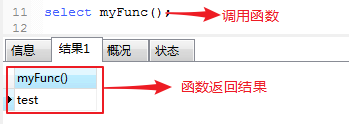
2、函数体每条语句完都要用";"结束

**例子：**

①定义无参mysql函数

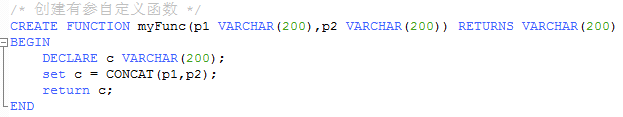


**调用此函数：**

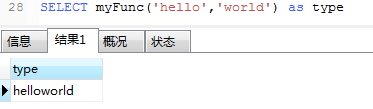


②定义有参的mysql函数

参数除了参数名以外，还需要指定参数的类型；DECLARE 关键字是定义一个变量；set关键字是为变量赋值



**调用此函数：**



### 声明变量

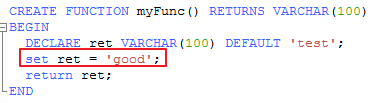
声明变量使用关键字" declare "(或大写版" DECLARE ")，变量为局部变量。

通过default 可以为声明的变量设置默认值；有两种方式可以为变量赋值：

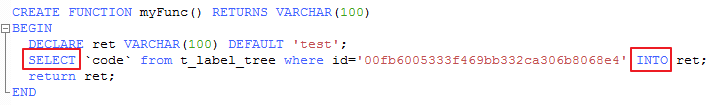
set和select...into...。

**例子：**

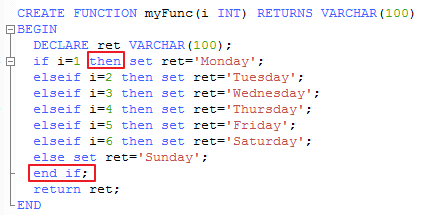
① 使用set为变量赋值



② 使用select...into为变量赋值



### if语句

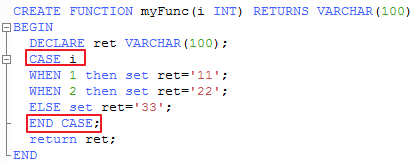


每个if语句判断后用 then 执行处理逻辑，并以";"结束

if语句执行完要以"end if"结尾。

**PS：**mysql中if判断条件是"="，不是"=="

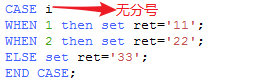
### case语句



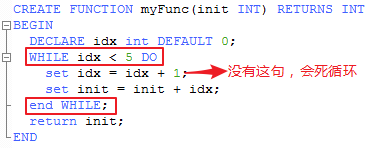
case语句比if语句简洁，它是用 when 判断，也用 then 处理函数逻辑。

case语句执行完需要用"end case"结尾。

**PS：**使用case语句，第一句不要使用";"



### while语句

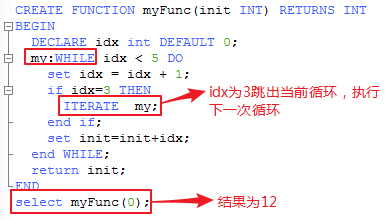


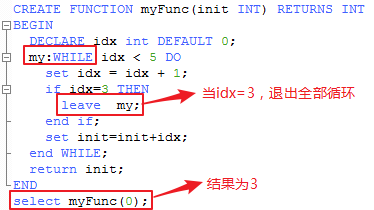
while + 循环条件 + DO，表示循环开始，while语句执行完用end while结尾。

一个while循环可以添加一个标签label，用来退出循环和跳出当前循环时使用。

退出循环使用"leave"，相当于"break"

跳出当前循环使用"iterate"，相当于"continue"





### 删除函数

**语法：**drop function <func\_name> ##<func\_name>是函数名称

**例子：**drop function myFunc

# MySQL驱动包

mysql驱动包版本5.x和6.x之间做了很大的改变:

①mysql-connector-java 5的驱动类时：com.mysql.jdbc.Driver

①mysql-connector-java 6的驱动类时：com.mysql.cj.jdbc.Driver

## 5.x配置

jdbc.driver=com.mysql.jdbc.Driver

jdbc.url=jdbc:mysql://localhost:3306/mybatis?characterEncoding=utf-8

jdbc.username=root

jdbc.password=

## 6.x配置

6.x引入时区的概念，所以配置中要指定时区。

jdbc.driver=com.mysql.cj.jdbc.Driver

jdbc.url=jdbc:mysql://localhost:3306/mybatis?useUnicode=true&characterEncoding=UTF-8&useJDBCCompliantTimezoneShift=true&useLegacyDatetimeCode=false&serverTimezone=UTC

jdbc.username=root

jdbc.password=

在设定时区的时候，如果设定serverTimezone=UTC，会比中国时间早8个小时，如果在中国，可以选择时区为Asia/Shanghai或者Asia/Hongkong，例如：

jdbc.driver=com.mysql.cj.jdbc.Driver

jdbc.url=jdbc:mysql://localhost:3306/mybatis?useUnicode=true&characterEncoding=UTF-8&useJDBCCompliantTimezoneShift=true&useLegacyDatetimeCode=false&serverTimezone=Asia/Shanghai

jdbc.username=root

jdbc.password=