



细细品味 Hadoop

——Hadoop 集群（第 10 期）

精
华
集
锦

csAxp

虾皮工作室

<http://www.cnblogs.com/xia520pi/>

2012 年 6 月 11 日

Hadoop 集群（第 10 期）

——MySQL 关系数据库

1、MySQL 安装

MySQL 下载地址：<http://www.mysql.com/downloads/>

1.1 Windows 平台

1) 准备软件

MySQL 版本：**mysql-5.5.21-win32.msi**

2) 安装环境：

操作系统：Windows 7 旗舰版

3) 开始安装

第一步：双击“msi”安装文件，出现如图 1.1-1 界面——“MySQL 安装向导”，按“Next”继续。

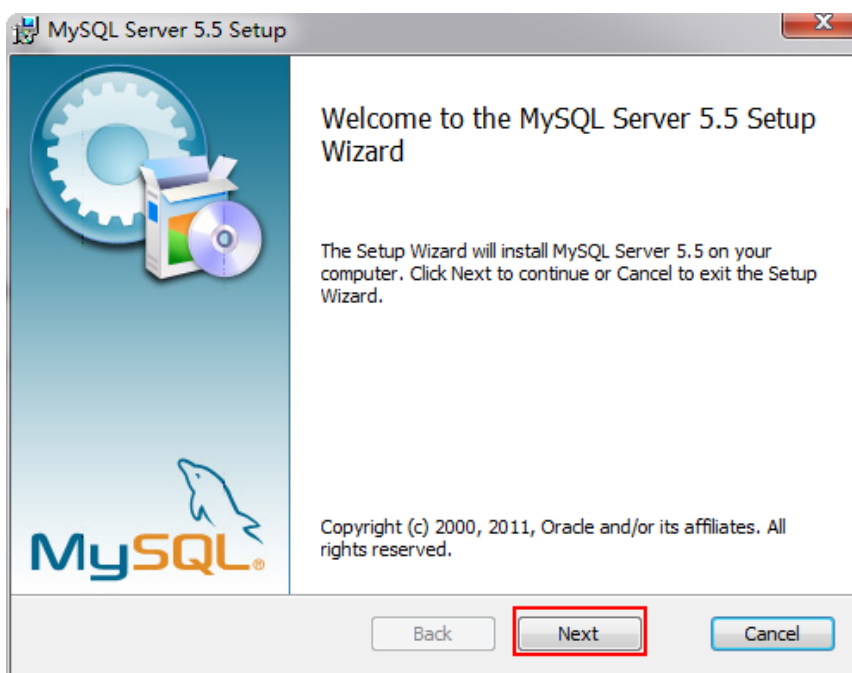


图 1.1-1 MySQL 安装向导

第二步：在“I accept”前面勾上，同意协议，按“Next”按钮继续。

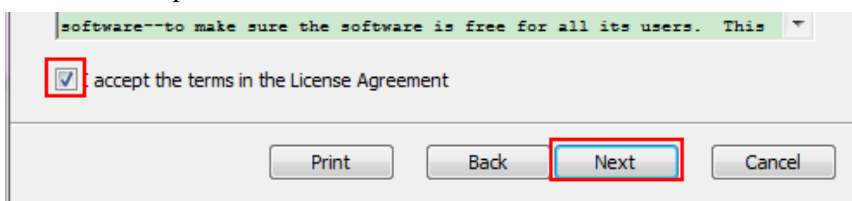


图 1.1-2 软件协议

第三步：选择安装类型，有“Typical（默认）”、“Custom（定制安装）”、“Complete（完全）”三个选项。

- **典型安装：**安装只安装 MySQL 服务器、mysql 命令行客户端和命令行实用程序。命令行客户端和实用程序包括 mysqldump、myisamchk 和其它几个工具来帮助你管理 MySQL 服务器。
- **定制安装：**安装允许你完全控制你想要安装的软件包和安装路径。
- **完全安装：**安装将安装软件包内包含的所有组件。完全安装软件包包括的组件包括嵌入式服务器库、基准套件、支持脚本和文档。

我们选择“**Custom**”，有更多的选项，也方便熟悉安装过程。

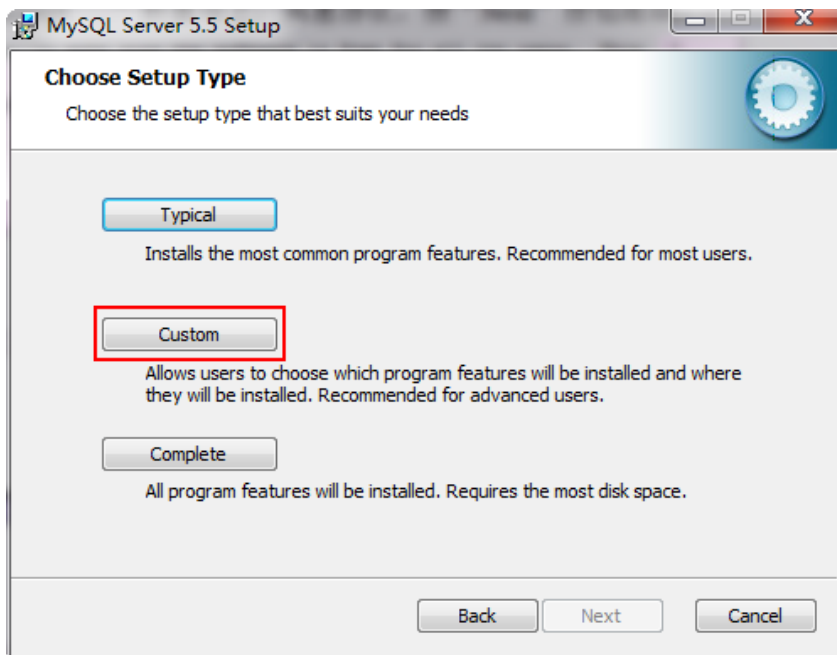


图 1.1-3 安装类型

第四步：选择组件及更改文件夹位置。

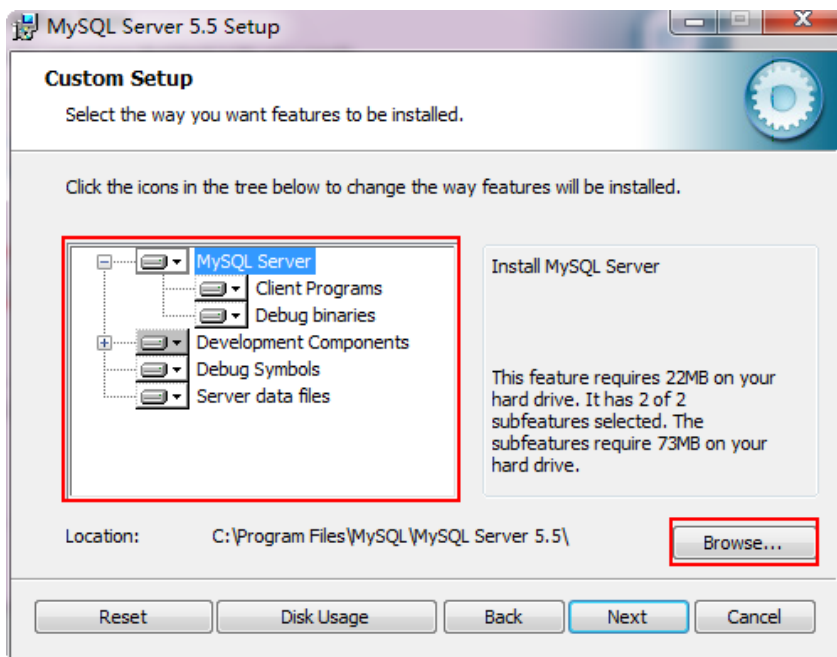


图 1.1-4 自定义界面

所有可用组件列入定制安装对话框左侧的树状视图内。未安装的组件用**红色 X** 图标表示；已经安装的组件有灰色图标。要想更改组件，点击该组件的图标并从下拉列表中选择新的选项。组件我选择了**默认安装**，**位置**我会**更改一下**，点击 Browse。

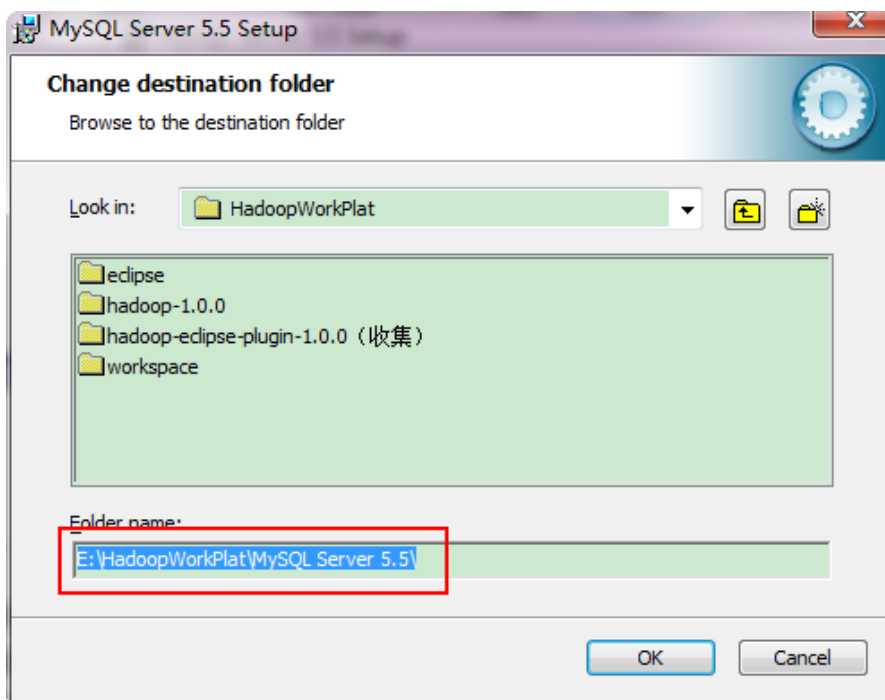


图 1.1-5 路径选择

按“OK”按钮返回，并按“Next”按钮继续。

备注：安装 mysql 的路径中，**不能**含有**中文**。

第五步：确认一下先前的设置，如果有误，按“Back”返回重做。按“Install”开始安装。

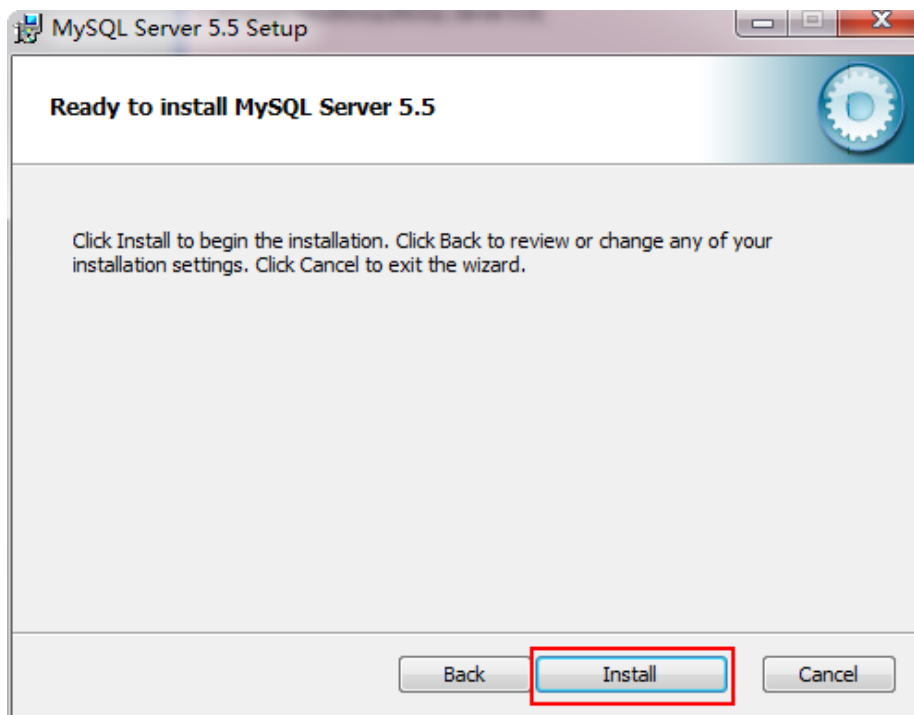


图 1.1-6 准备安装

第六步：正在安装中，请稍候.....

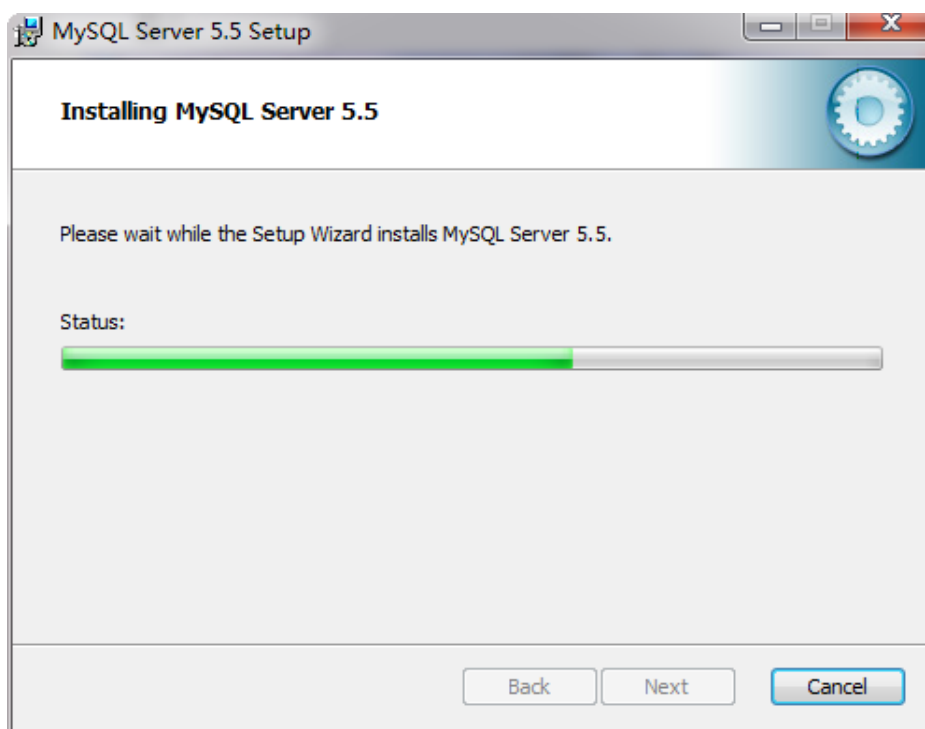


图 1.1-7 正在安装

第七步：弹出一个页面来，是关于介绍 MySQL 企业版的信息，没有什么可操作的，按“Next”按钮继续。



图 1.1-8 MySQL 企业版介绍

然后弹出一个类似界面，接着按“Next”按钮继续。

第八步：那个带复选框的是“MySQL 服务器实例配置向导”，保持默认勾选，按“Finish”按钮。至此，安装过程已经结束了，但是还需要配置一下。

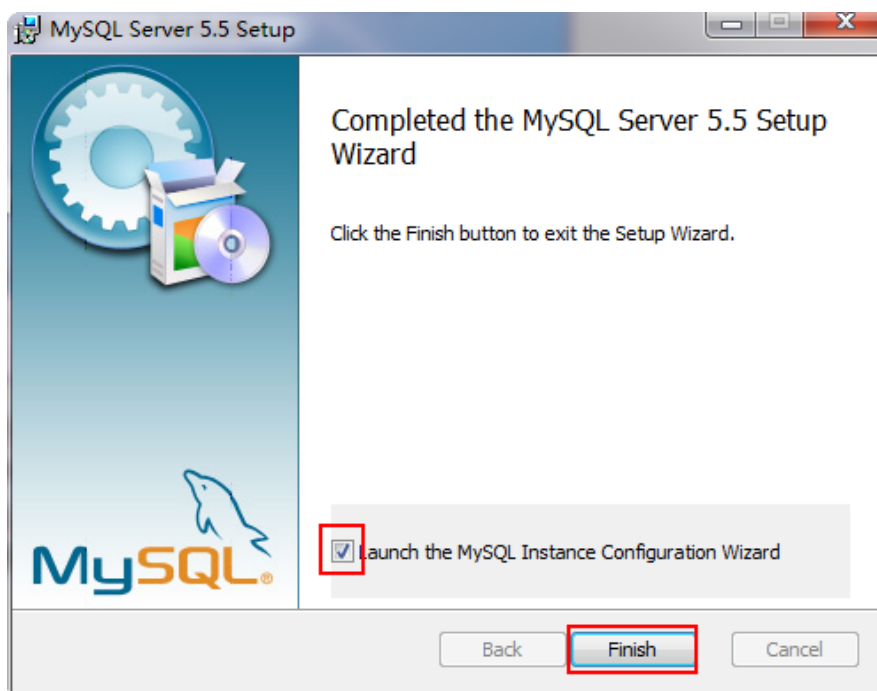


图 1.1-9 安装结束

第九步：MySQL 配置向导启动界面，按“Next”按钮继续。

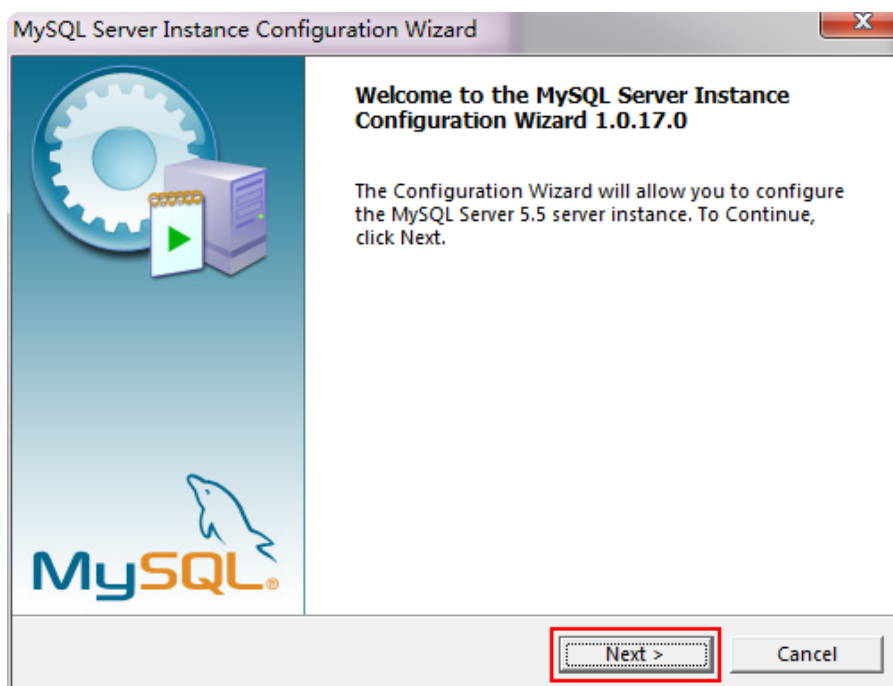


图 1.1-10 配置向导

MySQL Configuration Wizard（配置向导）可以帮助自动配置 Windows 中的服务器。MySQL Configuration Wizard（配置向导）问你一系列问题，然后将回答放到模板中生成一个 my.ini 文件，该文件与你的安装一致。目前只适用于 Windows 用户。

一般情况当 MySQL 安装帮助退出时，从 MySQL 安装帮助启动 MySQL Configuration Wizard（配置向导）。还可以点击 Windows 启动菜单中 MySQL 服务器实例配置向导条目中的 MySQL 部分来启动 MySQL Configuration Wizard（配置向导）。并且，还可以进入 MySQL 安装 bin 目录直接启动 MySQLInstanceConfig.exe 文件。

第十步：选择配置类型，可以选择两种配置类型：Detailed Configuration（详细配置）和 Standard Configuration（标准配置）。Standard Configuration（标准配置）选项适合想要快速启动 MySQL 而不必考虑服务器配置的新用户。详细配置选项适合想要更加细粒度控制服务器配置的高级用户。**我们选择“Detailed Configuration”，方便熟悉配置过程。**

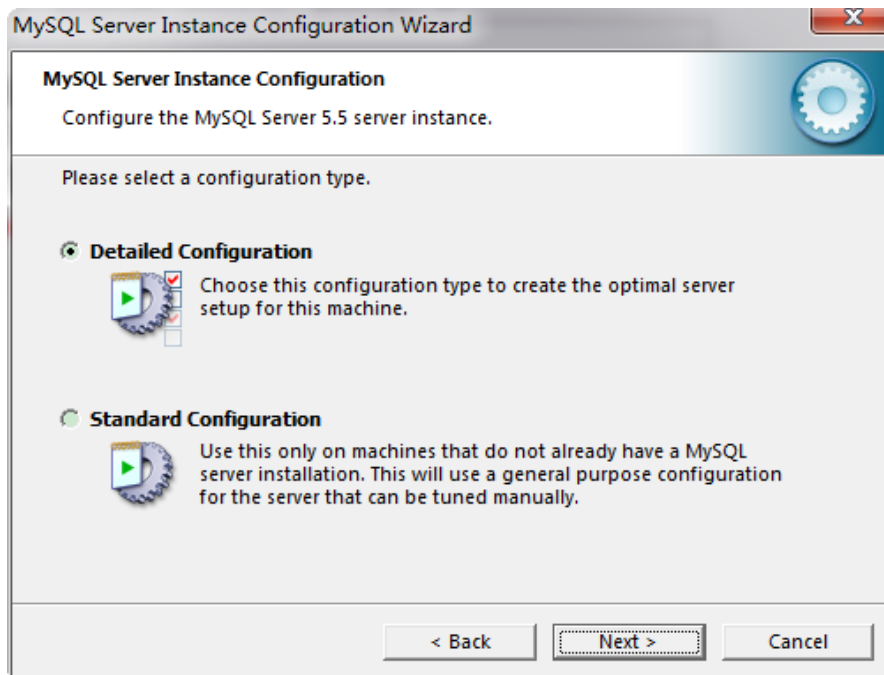


图 1.1-11 配置类型

备注：

如果你是 MySQL 的新手，需要配置为单用户开发机的服务器，Standard Configuration（标准配置）应当适合你的需求。选择 Standard Configuration（标准配置）选项，则 MySQL Configuration Wizard（配置向导）自动设置所有配置选项，但不包括服务选项和安全选项。

Standard Configuration（标准配置）设置选项可能与安装 MySQL 的系统不兼容。如果系统上已经安装了 MySQL 和你想要配置的安装，建议选择详细配置。

第十一步：选择服务器类型，可以选择 3 种服务器类型，选择哪种服务器将影响到 MySQL Configuration Wizard（配置向导）对内存、硬盘和过程或使用的决策。

- Developer Machine（开发机器）：该选项代表典型个人用桌面工作站。假定机器上运行着多个桌面应用程序。将 MySQL 服务器配置成使用最少的系统资源。
- Server Machine（服务器）：该选项代表服务器，MySQL 服务器可以同其它应用程序一起运行，例如 FTP、email 和 web 服务器。MySQL 服务器配置成使用适当比例的系统资源。
- Dedicated MySQL Server Machine（专用 MySQL 服务器）：该选项代表只运行 MySQL 服务的服务器。假定运行没有运行其它应用程序。MySQL 服务器配置成使用所有可用系统资源。

大家根据自己的类型选择了，一般选“Server Machine”，不会太少，也不会占满。**我们选择“Server Machine”，按“Next”按钮继续。**

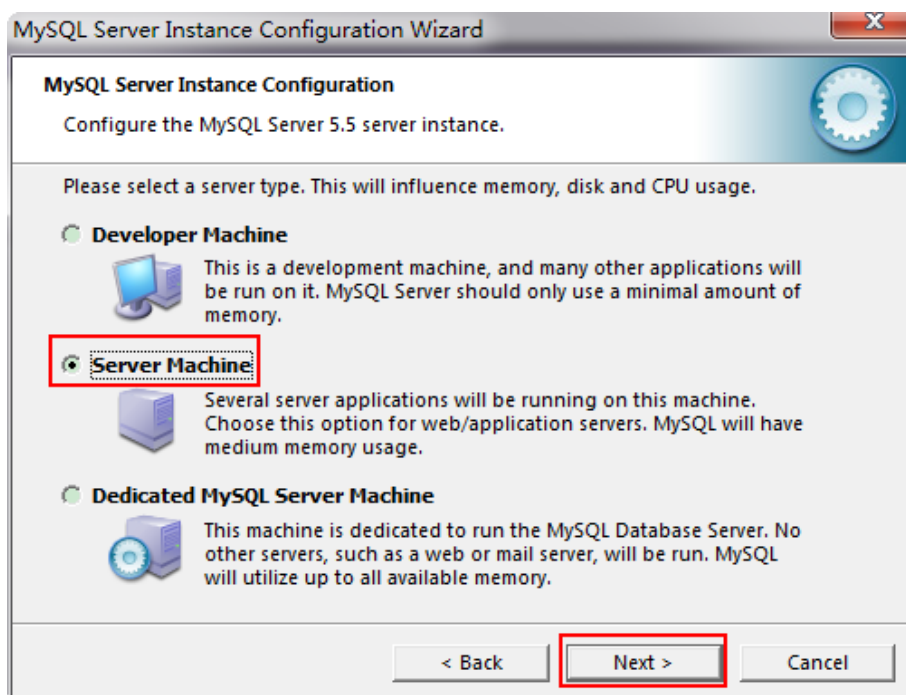


图 1.1-12 服务器类型

第十二步：选择数据库用途，通过 Database Usage（数据库使用）对话框，你可以指出创建 MySQL 表时使用的表处理器。通过该选项，你可以选择是否使用 InnoDB 储存引擎，以及 InnoDB 占用多大比例的服务器资源。“Multifunctional Database（通用多功能型，**好**）”、“Transactional Database Only（服务器类型，专注于事务处理，**一般**）”、“Non-Transactional Database Only（非事务处理型，较简单，主要做一些监控、记数用，对 MyISAM 数据类型的支持仅限于 non-transactional）”，随自己的用途而选择了，一般选择第一种多功能的。**我们选择“Multifunctional Database”，按“Next”按钮继续。**

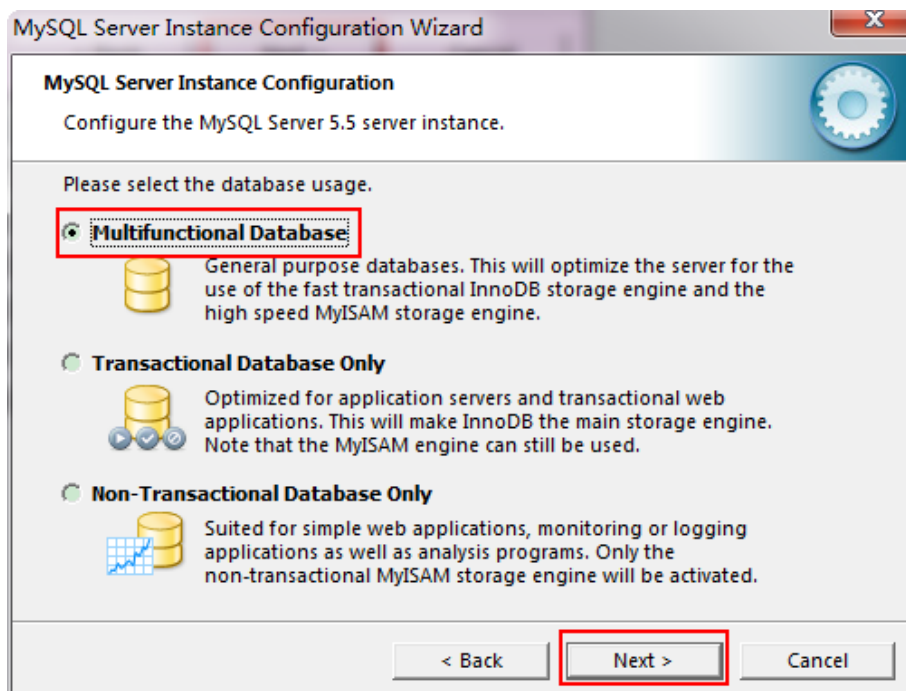


图 1.1-13 数据库用途

第十三步：对 InnoDB Tablespace 进行配置，就是为 InnoDB 数据库文件选择一个存储

空间, 如果修改了, 要记住位置, 重装的时候要选择一样的地方, 否则可能会造成数据库损坏, 当然, 对数据库做个备份就没问题了, 这里不详述。这里没有修改, **使用默认位置**, 按“Next”按钮继续。

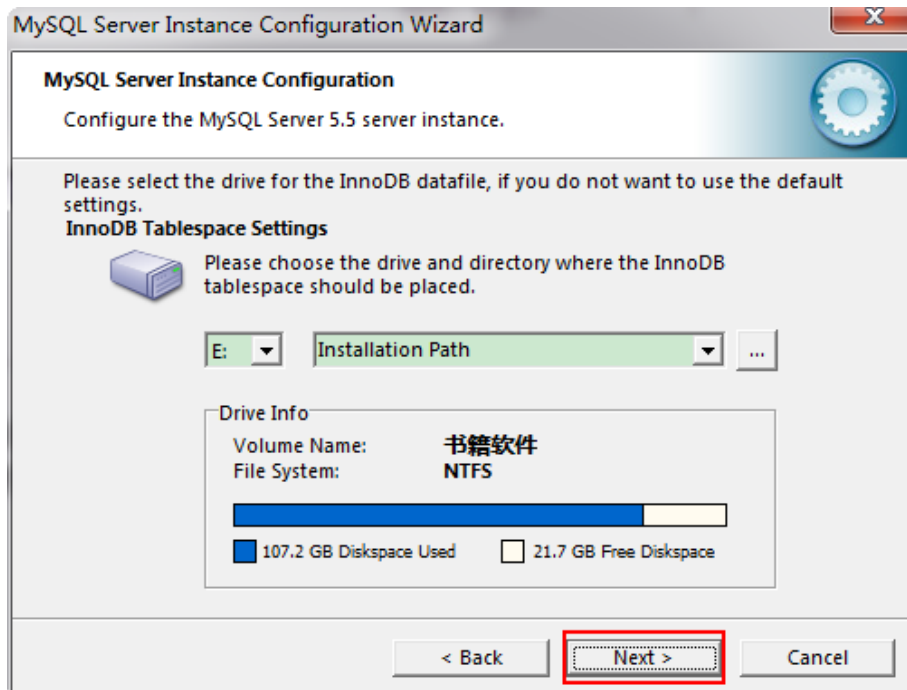


图 1.1-14 配置 InnoDB 表空间

第十四步: 选择 MySQL 允许的最大连接数, 限制所创建的与 MySQL 服务器之间的并行连接数量很重要, 以便防止服务器耗尽资源。在 Concurrent Connections (并行连接) 对话框中, 可以选择服务器的使用方法, 并根据情况限制并行连接的数量。还可以手动设置并行连接的限制。第一种是最大 20 个连接并发数, 第二种是最大 500 个并发连接数, 最后一种是自定义。我们选择 **“Online Transaction Processing(OLTP)”**, 按“Next”按钮继续。

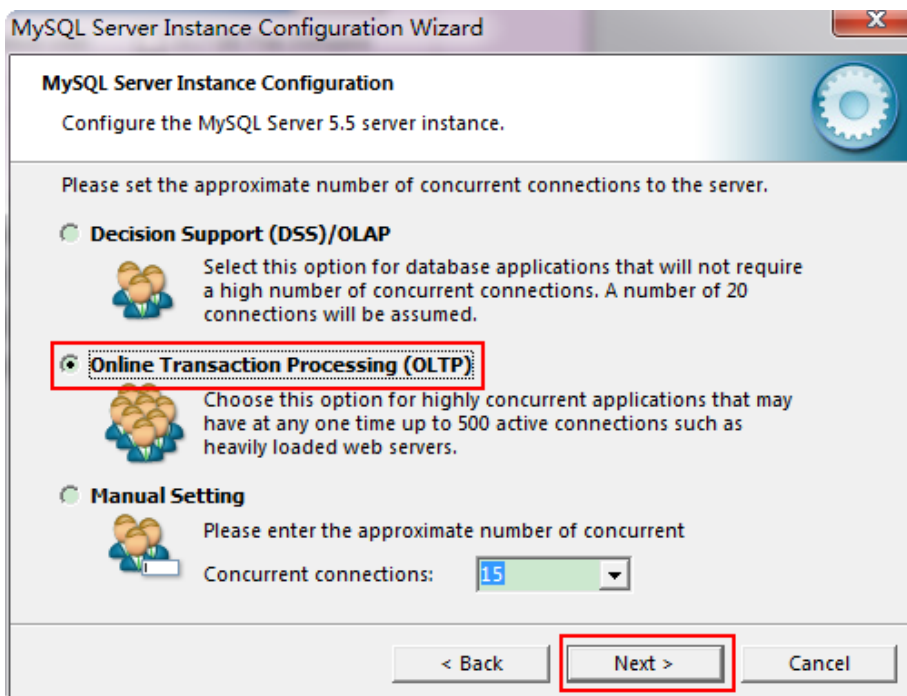


图 1.1-15 最大连接数

第十五步：进行网络配置，在 Networking Options（网络选项）对话框中可以启用或禁用 TCP/IP 网络，并配置用来连接 MySQL 服务器的端口号。默认情况启用 TCP/IP 网络。要想禁用 TCP/IP 网络，取消选择 Enable TCP/IP Networking 选项旁边的检查框。默认使用 3306 端口。要想更改 MySQL 使用的端口，从下拉框选择一个新端口号或直接向下拉框输入新的端口号。如果你选择的端口号已经被占用，将提示确认选择的端口号。**我们保持默认选项**，按“Next”按钮继续。

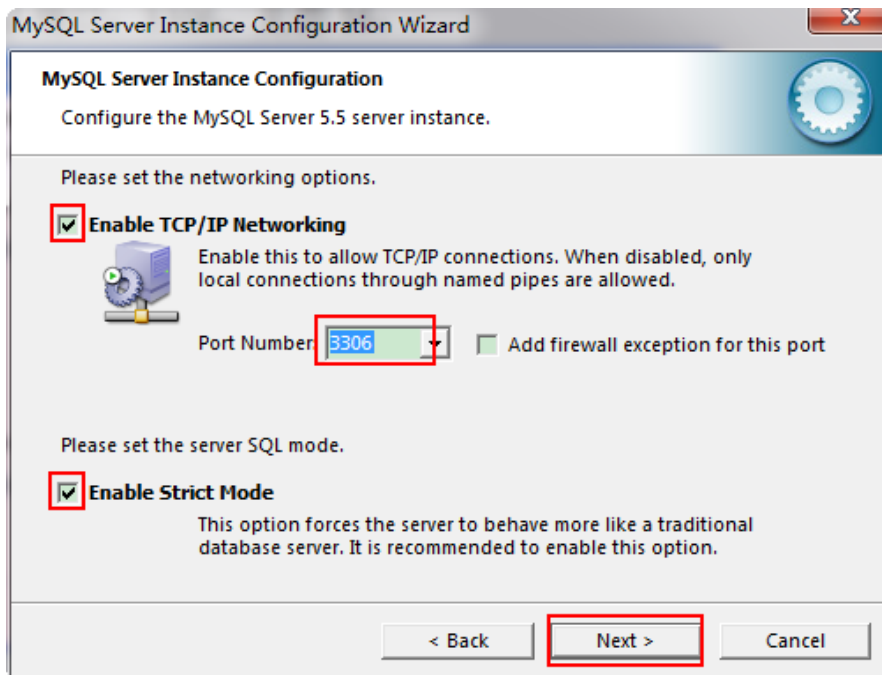


图 1.1-16 网络配置

第十六步：对数据库语言编码进行设置，非常重要，因为 Hadoop 里默认编码为 UTF-8，所以为了避免出现乱码，我们这里选择“UTF-8”作为 MySQL 数据库的语言编码。

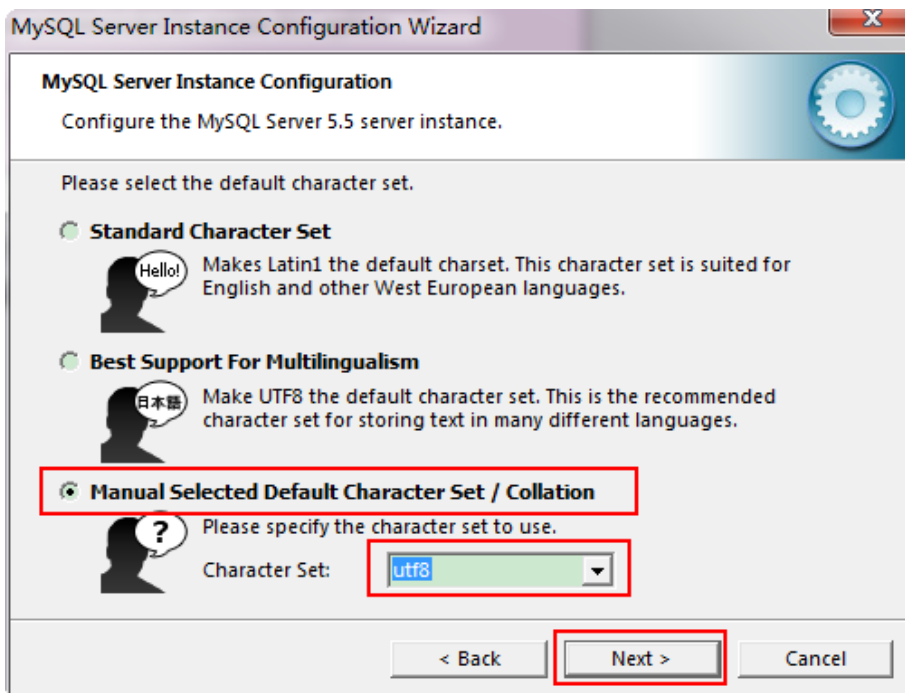


图 1.1-17 数据库编码

第十七步：是否要把 MySQL 设置成 Windows 的服务，一般选择设成服务，这样以后就可以通过服务中启动和关闭 mysql 数据库了。推荐：下面的复选框也勾选上，这样，在 cmd 模式下，不必非到 mysql 的 bin 目录下执行命令。**我们全部打上了勾**，Service Name 不变。按“Next”按钮继续。

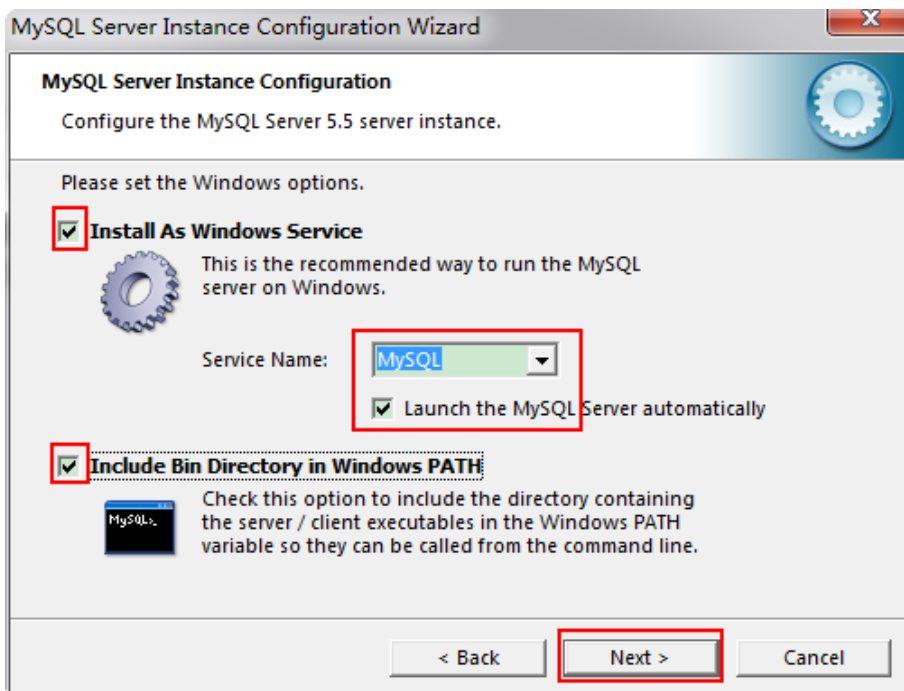


图 1.1-18 服务选项

第十八步：设置 MySQL 的超级用户密码，这个超级用户非常重要，对 MySQL 拥有全部的权限，请设置好并牢记超级用户的密码，下面有个复选框是选择是否允许远程机器用 root 用户连接到你的 MySQL 服务器上面，如果有这个需求，也请勾选。**我们这里的 root 用户密码**设置为“**hadoop**”，并**勾上“允许远程连接”复选框**，按“Next”按钮继续。

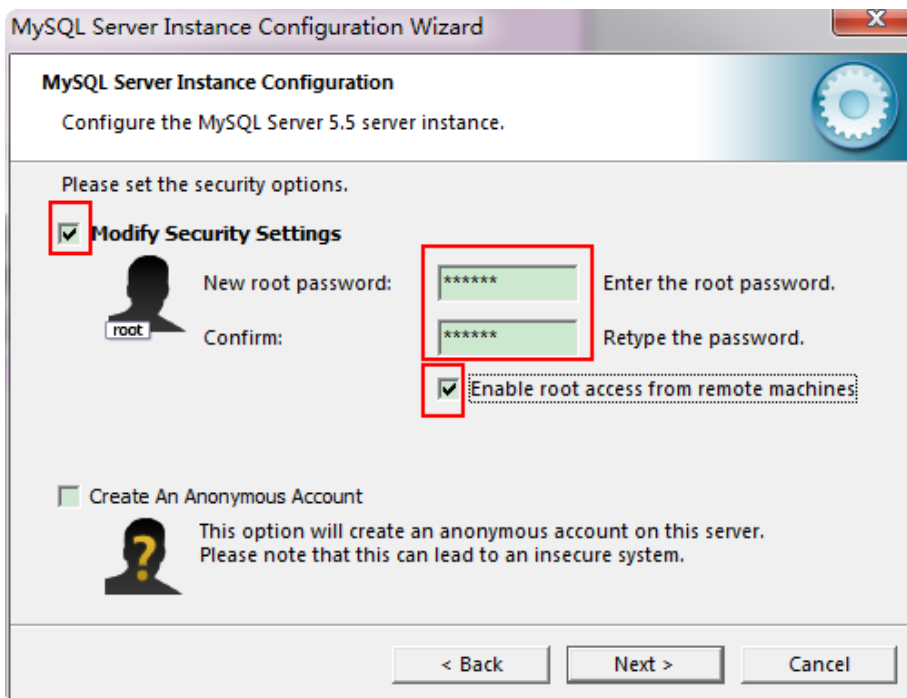


图 1.1-19 安全选项

备注：

- “Enable root access from remote machines（是否允许 root 用户在其它的机器上登陆，如果要安全，就不要勾上，如果要方便，就勾上它）”。
- “Create An Anonymous Account（新建一个匿名用户，匿名用户可以连接数据库，不能操作数据，包括查询）”，一般就不用勾了。

第十九步：确认设置无误，如果有误，按“Back”返回检查。如果没有，按“Execute”使设置生效。

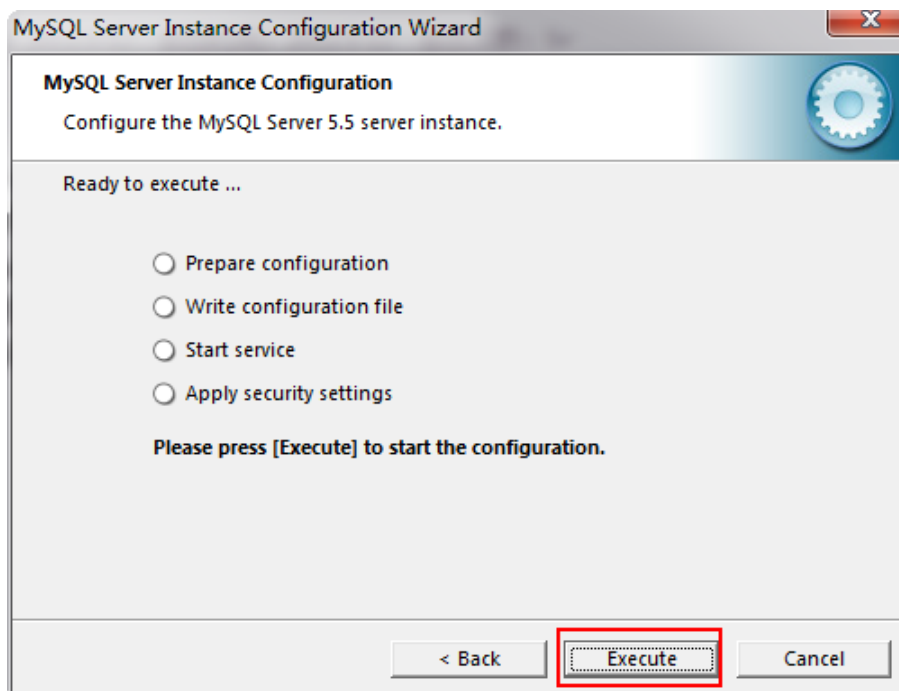


图 1.1-20 确认配置

第二十步：设置完毕，按“Finish”按钮结束 MySQL 的安装与配置。

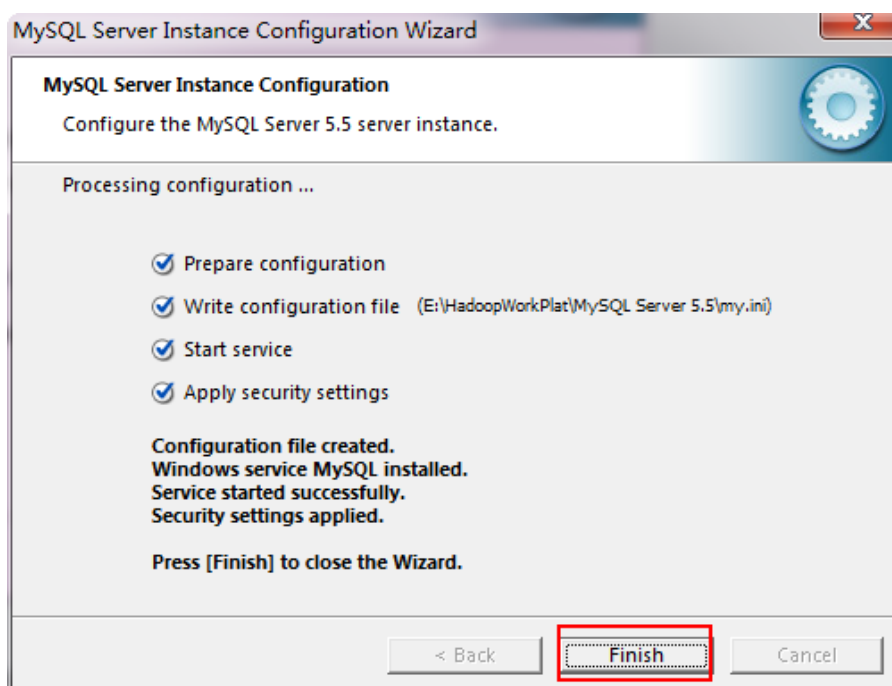


图 1.1-21 配置完成

备注: 这里有一个比较常见的错误, 就是不能 “Start service”, 一般出现在以前有安装 MySQL 的服务器上, 解决的办法, 先保证以前安装的 MySQL 服务器彻底卸载掉了; 不行的话, 检查是否按上面一步所说, 之前的密码是否有修改, 照上面的操作; 如果依然不行, 将 MySQL 安装目录下的 data 文件夹备份, 然后删除, 在安装完成后, 将安装生成的 data 文件夹删除, 备份的 data 文件夹移回来, 再重启 MySQL 服务就可以了, 这种情况下, 可能需要将数据库检查一下, 然后修复一次, 防止数据出错。

4) 验证成功

第一种: 打开任务管理器 看到 MySQL 服务是否已经启动。

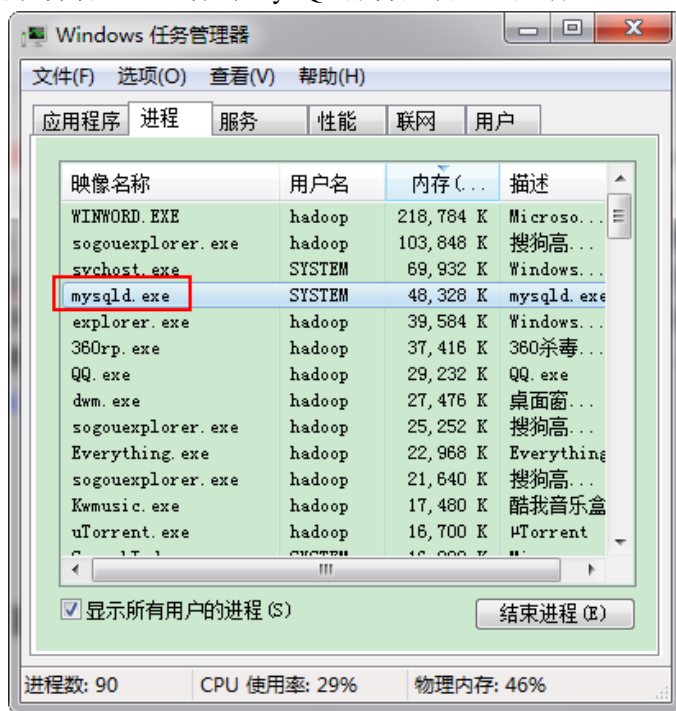


图 1.1-22 任务管理器

第二种: “开始→启动 cmd→开打 cmd 模式”, 输入 “mysql -u root -p” 连接数据库。

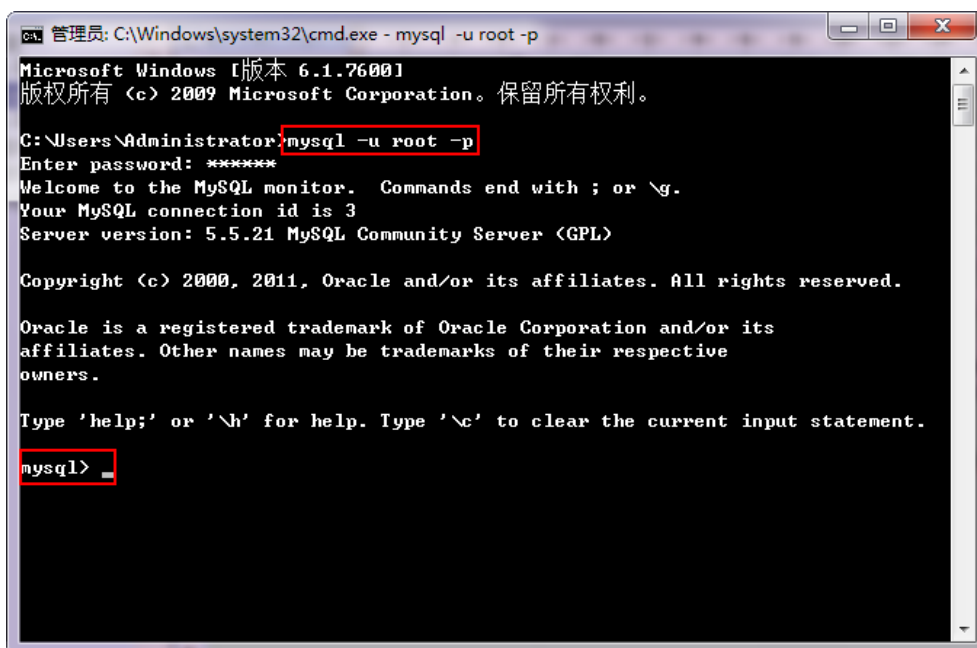


图 1.1-23 连接数据库

1.2 Linux平台

1) 准备软件

MySQL 数据库：**MySQL-server-5.5.21-1.linux2.6.i386.rpm**

MySQL 客户端：**MySQL-client-5.5.21-1.linux2.6.i386.rpm**

2) 安装环境：

操作系统：CentOS6.0 Linux

3) 检查安装

在安装 MySQL 之前，先检查 CentOS 系统中是否已经安装了一个 MySQL，如果已经安装先卸载，不然会导致安装**新**的 MySQL **失败**。

用下面命令查看系统之前是否已安装 MySQL。

```
rpm -qa | grep mysql
```

查看结果如下：

```
[hadoop@TMaster ~]$ rpm -qa | grep mysql
mysql-libs-5.1.47-4.el6.i686
[hadoop@TMaster ~]$
```

从上图得知，CentOS6.0 系统自带了一个 MySQL，我们需要删除这个老版本，用 **root** 用户执行下面语句。

```
rpm -e --nodeps mysql-libs-5.1.47-4.el6.i686
```

```
[hadoop@TMaster ~]$ su -
密码：
[root@TMaster ~]# rpm -e --nodeps mysql-libs-5.1.47-4.el6.i686
[root@TMaster ~]# rpm -qa | grep mysql
[root@TMaster ~]#
```

上图中，我们先切换到“**root**”用户下，然后执行删除语句，删除之后，我们再次查看，发现已经成功删除了 CentOS6.0 自带的旧 MySQL 版本。

在删除 MySQL 的 rpm 后，还要进行一些扫尾操作，网上有两种操作。（**备注**：我在这两种都没有用到，发现系统中**并没有**其他**残余**的 MySQL 信息。）

第一种善后处理：使用下面命令进行处理。

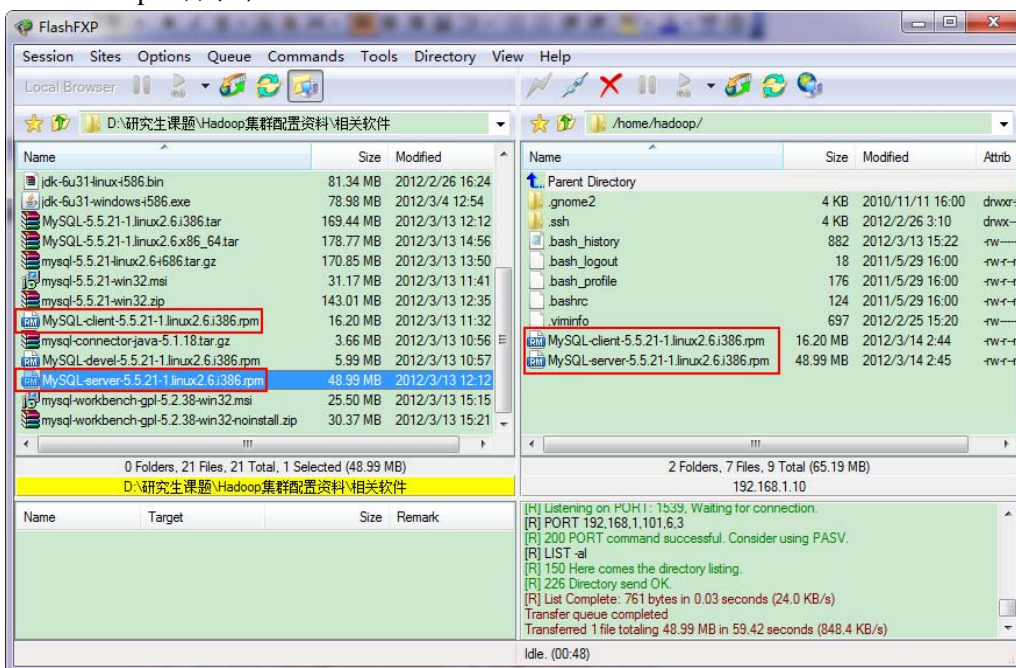
```
rm -rf /var/lib/mysql*
rm -rf /usr/share/mysql*
```

另一种善后处理：卸载后/var/lib/mysql 中的/etc/my.cnf 会重命名为 my.cnf.rpmsave，

/var/log/mysqld.log 会重命名为/var/log/mysqld.log.rpm.save, 如果确定没用后就手工删除。

4) 开始安装

第一步: 上传所需软件。通过“FlashFXP”软件使用“vsftpd”上传用到的两个软件到“/home/hadoop”目录下。



第二步: 安装 MySQL 服务端。用“root”用户运行如下命令进行安装: (备注: 以下步骤都是用“root”用户执行。)

```
rpm -ivh MySQL-server-5.5.21-1.linux2.6.i386.rpm
```

通过 SecureCRT 查看如下:

```
[hadoop@TMaster ~]$ su -
密码:
[root@TMaster ~]# cd /home/hadoop
[root@TMaster hadoop]# ll
总用量 66756
-rw-r--r--. 1 hadoop hadoop 16988103 3月 14 18:44 MySQL-client-5.5.21-1.linux2.6.i386.rpm
-rw-r--r--. 1 hadoop hadoop 51367247 3月 14 18:45 MySQL-server-5.5.21-1.linux2.6.i386.rpm
[root@TMaster hadoop]# rpm -ivh MySQL-server-5.5.21-1.linux2.6.i386.rpm
```

```
[root@TMaster hadoop]# rpm -ivh MySQL-server-5.5.21-1.linux2.6.i386.rpm
Preparing... ##### [100%]
1:MySQL-server ##### [100%]

PLEASE REMEMBER TO SET A PASSWORD FOR THE MySQL root USER !
To do so, start the server, then issue the following commands:

/usr/bin/mysqladmin -u root password 'new-password'
/usr/bin/mysqladmin -u root -h TMaster.Hadoop password 'new-password'

Alternatively you can run:
/usr/bin/mysql_secure_installation

which will also give you the option of removing the test
databases and anonymous user created by default. This is
strongly recommended for production servers.

See the manual for more instructions.

Please report any problems with the /usr/bin/mysqlbug script!
```


如出现如上信息，**服务端**安装完毕。

第三步：检测 MySQL **3306 端口**是否安打开。测试是否成功可运行 netstat 看 MySQL 端口是否打开，如打开表示服务已经启动，安装成功。MySQL 默认的端口是 3306。

```
netstat -nat
```

```
[root@TMaster hadoop]# netstat -nat
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 0.0.0.0:21              0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.1:25            0.0.0.0:*               LISTEN
tcp        0    52 192.168.1.10:22         192.168.1.101:1558      ESTABLISHED
tcp        0      0 :::22                   :::*                     LISTEN
tcp        0      0 :::1:25                  :::*                     LISTEN
[root@TMaster hadoop]#
```

从上图中发现并没有与“3306”有关的信息，说明“MySQL 服务器”没有启动。通过下面命令启动 MySQL。

```
service mysql start
```

```
[root@TMaster hadoop]# service mysql start
Starting MySQL.. SUCCESS!
[root@TMaster hadoop]# netstat -nat
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 0.0.0.0:3306            0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:21              0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.1:25            0.0.0.0:*               LISTEN
tcp        0    248 192.168.1.10:22         192.168.1.101:1558      ESTABLISHED
tcp        0      0 :::22                   :::*                     LISTEN
tcp        0      0 :::1:25                  :::*                     LISTEN
[root@TMaster hadoop]#
```

从上图中已经发现我们的 MySQL 服务器已经起来了。

第四步：安装 MySQL 客户端。用下面命令进行安装：

```
rpm -ivh MySQL-client-5.5.21-1.linux2.6.i386.rpm
```

执行命令显示如下：

```
[root@TMaster hadoop]# ll
总用量 66756
-rw-r--r--. 1 hadoop hadoop 16988103 3月 14 18:44 MySQL-client-5.5.21-1.linux2.6.i386.rpm
-rw-r--r--. 1 hadoop hadoop 51367247 3月 14 18:45 MySQL-server-5.5.21-1.linux2.6.i386.rpm
[root@TMaster hadoop]# rpm -ivh MySQL-client-5.5.21-1.linux2.6.i386.rpm
Preparing...##### [100%]
1:MySQL-client##### [100%]
[root@TMaster hadoop]#
```

从上图中显示 MySQL 客户端已经安装完毕。

第五步：MySQL 的几个重要目录。MySQL 安装完成后**不像 SQL Server**默认安装在一个目录，它的**数据库文件**、**配置文件**和**命令文件**分别在**不同的目录**，了解这些目录非常重

要，尤其对于 Linux 的初学者，因为 Linux 本身的目录结构就比较复杂，如果搞不清楚 MySQL 的安装目录那就无从谈起深入学习。

下面就介绍一下这几个目录。

a、数据库目录

/var/lib/mysql/

b、配置文件

/usr/share/mysql (mysql.server 命令及配置文件)

c、相关命令

/usr/bin(mysqladmin mysqldump 等命令)

d、启动脚本

/etc/rc.d/init.d/ (启动脚本文件 mysql 的目录)

如：/etc/rc.d/init.d/mysql start/restart/stop/status

下面就分别展示上面的几个目录内容：

● 数据库目录

```
[root@TMaster hadoop]# ll /var/lib/mysql
总用量 28700
-rw-rw----. 1 mysql mysql 18874368 3月 14 19:30 ibdata1
-rw-rw----. 1 mysql mysql 5242880 3月 14 19:32 ib_logfile0
-rw-rw----. 1 mysql mysql 5242880 3月 14 19:30 ib_logfile1
drwx--x--x. 2 mysql mysql 4096 3月 14 18:58 mysql
srwxrwxrwx. 1 mysql mysql 0 3月 14 19:32 mysql.sock
drwx-----. 2 mysql mysql 4096 3月 14 18:58 performance_schema
-rw-r--r--. 1 root root 130 3月 14 18:58 RPM_UPGRADE_HISTORY
-rw-r--r--. 1 mysql mysql 130 3月 14 18:58 RPM_UPGRADE_MARKER-LAST
drwxr-xr-x. 2 mysql mysql 4096 3月 14 18:58 test
-rw-rw----. 1 mysql root 3016 3月 14 19:32 TMaster.Hadoop.err
-rw-rw----. 1 mysql mysql 6 3月 14 19:32 TMaster.Hadoop.pid
```

● 配置文件

```
[root@TMaster hadoop]# ls /usr/share/mysql
binary-configure  fill_help_tables.sql  my-large.cnf          norwegian-ny
charsets          french                 my-medium.cnf         polish
config.huge.ini   german                my-small.cnf          portuguese
config.medium.ini greek                 mysqld_multi.server   romanian
config.small.ini  hungarian             mysql-log-rotate      russian
czech            italian               mysql.server          SELinux
danish           japanese              mysql_system_tables_data.sql  serbian
dutch            korean                mysql_system_tables.sql  slovak
english          magic                 mysql_test_data_timezone.sql  spanish
errmsg-utf8.txt  my-huge.cnf           ndb-config-2-node.ini  swedish
estonian         my-innodb-heavy-4G.cnf  norwegian              ukrainian
[root@TMaster hadoop]#
```

● 相关命令

```
[root@TMaster hadoop]# ls /usr/bin/mysql*
/usr/bin/mysql                /usr/bin/mysqlhotcopy
/usr/bin/mysqlaccess          /usr/bin/mysqlimport
/usr/bin/mysqlaccess.conf     /usr/bin/mysql_install_db
/usr/bin/mysqladmin           /usr/bin/mysql_plugin
/usr/bin/mysqlbinlog          /usr/bin/mysql_secure_installation
/usr/bin/mysqlbug             /usr/bin/mysql_setpermission
/usr/bin/mysqlcheck           /usr/bin/mysqlshow
/usr/bin/mysql_convert_table_format /usr/bin/mysqlslap
/usr/bin/mysqld_multi          /usr/bin/mysqltest
/usr/bin/mysqld_safe           /usr/bin/mysql_tzinfo_to_sql
/usr/bin/mysqldump             /usr/bin/mysql_upgrade
/usr/bin/mysqldumpslow         /usr/bin/mysql_waitpid
/usr/bin/mysql_find_rows       /usr/bin/mysql_zap
/usr/bin/mysql_fix_extensions
```

- 启动脚本

```
[root@TMaster hadoop]# ls /etc/rc.d/init.d
abrtid      cgroup      iptables    mdmonitor   network     sandbox     udev-post
acpid       cpuspeed    iptables    messagebus  postfix     saslauthd   vsftpd
atd         crond       irqbalance  microcode_ctl psacct      single
auditd      functions   kdump       mysql        rdisc       smartd
avahi-daemon haldaemon   killall     netconsole  restorecond sshd
cgconfig    halt        lvm2-monitor netfs        rsyslog     sysstat
```

第六步：更改 MySQL 目录。由于 MySQL 数据库目录占用磁盘比较大，而 MySQL 默认的数据文件存储目录为“var/lib/mysql”，所以我们要把目录移到“/”根目录下的“mysql_data”目录中。

需要以下几个步骤：

- “/”根目录下建立“mysql_data”目录

```
cd /
mkdir mysql_data
```

```
[root@TMaster hadoop]# cd /
[root@TMaster /]# ls
bin  cgroup  etc  lib  media  opt  root  selinux  sys  usr  yun
boot dev  home  lost+found  mnt  proc  sbin  srv  tmp  var
[root@TMaster /]# mkdir mysql_data
[root@TMaster /]# ls
bin  cgroup  etc  lib  media  mysql_data  proc  sbin  srv  tmp  var
boot dev  home  lost+found  mnt  opt  root  selinux  sys  usr  yun
```

- 把 MySQL 服务进程停掉

可以用两种方法：

```
service mysql stop
```

或者

```
mysqladmin -u root -p shutdown
```

```
[root@TMaster /]# mysqladmin -u root -p shutdown
Enter password:
[root@TMaster /]# netstat -nat
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 0.0.0.0:21              0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.1:25            0.0.0.0:*               LISTEN
tcp        0    52 192.168.1.10:22         192.168.1.101:1558      ESTABLISHED
tcp        0      0 :::22                   :::*                     LISTEN
tcp        0      0 :::1:25                  :::*                     LISTEN
[root@TMaster /]#
```

从上图我们得知“MySQL 服务进程”已经停掉。

备注：MySQL 默认用户名为“root”，此处的“root”与 Linux 的最高权限用户“root”不是一会儿，而且默认的用户“root”的密码为空，所以上图中让输入密码，直接点击回车即可。

- 把“/var/lib/mysql”整个目录移到“/mysql_data”

```
mv /var/lib/mysql /mysql_data
```

```
[root@TMaster /]# mv /var/lib/mysql /mysql_data
[root@TMaster /]# ll /mysql_data
总用量 4
drwxr-xr-x. 5 mysql mysql 4096 3月 14 20:14 mysql
[root@TMaster /]#
```

这样就把 MySQL 的数据文件移动到了“/mysql_data/mysql”下。

- 找到 my.cnf 配置文件

如果“/etc/”目录下没有 my.cnf 配置文件，请到“/usr/share/mysql/”下找到*.cnf 文件，拷贝其中一个合适的配置文件到“/etc/”并改名为“my.cnf”中。命令如下：

```
cp /usr/share/mysql/my-medium.cnf /etc/my.cnf
```

```
[root@TMaster ~]# ll /etc | grep my.cnf
[root@TMaster ~]# cp /usr/share/mysql/my-medium.cnf /etc/my.cnf
[root@TMaster ~]# ll /etc | grep my.cnf
-rwxr-xr-x. 1 root root 4682 3月 14 22:44 my.cnf
[root@TMaster ~]#
```

上图中，下查看“/etc/”下面是否有“my.cnf”文件，发现没有，然后通过上面的命令进行拷贝，拷贝完之后，进行查看，发现拷贝成功。

备注: “`/usr/share/mysql/`” 下有好几个结尾为 `cnf` 的文件, 它们的作用分别是。

```
[root@TMaster ~]# ll /usr/share/mysql | grep ".cnf$"
```

-rwxr-xr-x.	1	root	root	4697	1月	31	19:51	my-huge.cnf
-rwxr-xr-x.	1	root	root	19779	1月	31	19:51	my-innodb-heavy-4G.cnf
-rwxr-xr-x.	1	root	root	4671	1月	31	19:51	my-large.cnf
-rwxr-xr-x.	1	root	root	4682	1月	31	19:51	my-medium.cnf
-rwxr-xr-x.	1	root	root	2846	1月	31	19:51	my-small.cnf

```
[root@TMaster ~]#
```

a、**my-small.cnf**: 是为了小型数据库而设计的。不应该把这个模型用于含有一些常用项目的数据库。

b、**my-medium.cnf**: 是为中等规模的数据库而设计的。如果你正在企业中使用 RHEL, 可能会比这个操作系统的最小 RAM 需求(256MB)明显多得多的物理内存。由此可见, 如果有那么多 RAM 内存可以使用, 自然可以在同一台机器上运行其它服务。

c、**my-large.cnf**: 是为专用于一个 SQL 数据库的计算机而设计的。由于它可以为该数据库使用多达 512MB 的内存, 所以在这种类型的系统上将需要至少 1GB 的 RAM, 以便它能够同时处理操作系统与数据库应用程序。

d、**my-huge.cnf**: 是为企业中的数据库而设计的。这样的数据库要求专用服务器和 1GB 或 1GB 以上的 RAM。

这些选择高度依赖于内存的数量、计算机的运算速度、数据库的细节大小、访问数据库的用户数量以及在数据库中装入并访问数据的用户数量。随着数据库和用户的不断增加, 数据库的性能可能会发生变化。

备注: 这里我们根据实际情况, 选择了 “**my-medium.cnf**” 进行配置。

● 编辑 MySQL 的配置文件 “`/etc/my.cnf`”

为保证 MySQL 能够正常工作, 需要指明 “**mysql.sock**” 文件的产生位置, 以及默认编码修改为 **UTF-8**。用下面命令:

```
vim /etc/my.cnf
```

```
[root@TMaster ~]# vim /etc/my.cnf
```

需要修改和添加的内容如下:

【**client**】

```
socket                = /mysql_data/mysql/mysql.sock
default-character-set=utf8
```

【**mysqld**】

```
socket                = /mysql_data/mysql/mysql.sock
datadir               = /mysql_data/mysql
```

```
character-set-server=utf8
```

lower_case_table_names=1（注意 linux 下 mysql 安装完后是默认：区分表名的大小写，不区分列名的大小写；lower_case_table_names = 0 **0**：区分大小写，**1**：不区分大小写）

备注：【client】和【mysqld】设置的编码时前地名称不一样。

```
# The following options will be passed to all MySQL clients
[client]
#password      = your_password
port           = 3306
socket         = /mysql_data/mysql/mysql.sock
default-character-set=utf8
# Here follows entries for some specific programs

# The MySQL server
[mysqld]
port           = 3306
socket         = /mysql_data/mysql/mysql.sock
datadir        = /mysql_data/mysql
character-set-server=utf8
lower_case_table_names=1
skip-external-locking
key_buffer_size = 16M
max_allowed_packet = 1M
table_open_cache = 64
sort_buffer_size = 512K
net_buffer_length = 8K
read_buffer_size = 256K
read_rnd_buffer_size = 512K
myisam_sort_buffer_size = 8M
```

- 修改 MySQL 启动脚本 “/etc/rc.d/init.d/mysql”

最后，需要修改 MySQL 启动脚本/etc/rc.d/init.d/mysql，修改 **datadir=/mysql_data/mysql**。

```
vim /etc/rc.d/init.d/mysql
```

```
[root@TMaster ~]# vim /etc/rc.d/init.d/mysql
```

```
# If you change base dir, you must also change datadir. These may get
# overwritten by settings in the MySQL configuration files.

basedir=
datadir=/mysql_data/mysql
```

- 重新启动 MySQL 服务

```
service mysql start
```

```
[root@TMaster ~]# service mysql start
Starting MySQL. ERROR! The server quit without updating PID file
(/mysql_data/mysql/TMaster.Hadoop.pid).
[root@TMaster ~]#
```

正准备高兴时，发现 MySQL 启动不了了，网上搜了一下午，各种都没有解决。后来在一篇文章才得知又是“SELinux”惹得祸。解决办法如下：

打开/etc/selinux/config，把 SELINUX=enforcing 改为 SELINUX=disabled 后存盘退出重启机器试试，必须要重启，很关键。

```
# This file controls the state of SELinux on the system.
# SELINUX= can take one of these three values:
#     enforcing - SELinux security policy is enforced.
#     permissive - SELinux prints warnings instead of enforcing.
#     disabled - No SELinux policy is loaded.
#SELINUX=enforcing
SELINUX=disabled
# SELINUXTYPE= can take one of these two values:
#     targeted - Targeted processes are protected,
#     mls - Multi Level Security protection.
SELINUXTYPE=targeted
```

机器重启之后，在把“mysql 服务”启动。

```
[root@TMaster ~]# service mysql start
Starting MySQL SUCCESS!
[root@TMaster ~]#
```

第七步：修改登录密码。

MySQL 默认没有密码，安装完毕增加密码的重要性是不言而喻的。

- **修改前，直接登录**

```
[root@TMaster ~]# mysql
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 2
Server version: 5.5.21-log MySQL Community Server (GPL)

Copyright (c) 2000, 2011, 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>
```

在没有添加密码前，直接输入“mysql”就能登录到 MySQL 数据库里。

- 修改登录密码

用到的命令如下：

```
mysqladmin -u root password 'new-password'
```

格式：mysqladmin -u 用户名 -p 旧密码 password 新密码

我们这里设置 MySQL 数据库 “root” 用户的密码为 “hadoop”。执行的命令如下：

```
mysqladmin -u root password hadoop
```

```
[root@TMaster ~]# mysqladmin -u root password hadoop
[root@TMaster ~]#
```

- 测试是否修改成功

- (1) 不用密码登录

```
[root@TMaster ~]# mysql
ERROR 1045 (28000): Access denied for user 'root'@'localhost' (using password: NO)
[root@TMaster ~]#
```

此时显示错误，说明密码已经修改。

- (2) 用修改后的密码登录

```
[root@TMaster ~]# mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 5
Server version: 5.5.21-log MySQL Community Server (GPL)

Copyright (c) 2000, 2011, 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>
```

从上图中得知，我们已经成功修改了密码，并且用新的密码登录了 MySQL 服务器。

第八步：配置防火墙

第一种：修改防火墙配置文件 “/etc/sysconfig/iptables”，添加如下内容：

```
-A INPUT -m state --state NEW -m tcp -p tcp --sport 3306 -j ACCEPT
-A OUTPUT -m state --state NEW -m tcp -p tcp --dport 3306 -j ACCEPT
```

然后执行下面命令，使防火墙立即生效。

```
service iptables restart
```

第二种：关闭防火墙

通过下面两个命令使防火墙关闭，并且永远不起作用。

```
service iptables stop
chkconfig iptables off
```

我们在这里为了方便，采用**第二种**方法，执行效果如下。

```
[root@TMaster ~]# service iptables stop
iptables: 清除防火墙规则: [确定]
iptables: 将链设置为政策 ACCEPT: filter [确定]
iptables: 正在卸载模块: [确定]
[root@TMaster ~]# chkconfig iptables off
[root@TMaster ~]# chkconfig --list | grep iptables
iptables      0:关闭  1:关闭  2:关闭  3:关闭  4:关闭  5:关闭  6:关闭
[root@TMaster ~]#
```

第九步：验证 MySQL 数据库编码是否为 UTF-8。

连接上数据库之后，输入命令：“**SHOW VARIABLES LIKE '%char%';**”即可查看到现在你的数据库所使用的字符集了。

```
[root@TMaster ~]# mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 6
Server version: 5.5.21-log MySQL Community Server (GPL)

Copyright (c) 2000, 2011, 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 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 (0.01 sec)

mysql>
```

第十步：删除空用户，增强安全。

目前为止我们都是以“root”的身份进行的，但是当我们切换至普通用户登录 MySQL 时，直接输入“mysql”就进去了，我们刚才不是设置密码了吗？怎么就失效了呢？说明有空用户存在。先用命令“exit”退出，在按照下面命令进行修正。

```
Last login: Thu Mar 15 18:37:22 2012 from 192.168.1.102
[hadoop@TMaster ~]$ mysql
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 13
Server version: 5.5.21-log MySQL Community Server (GPL)

Copyright (c) 2000, 2011, 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
[hadoop@TMaster ~]$
```

解决步骤如下：

- 以 MySQL 用户“root”用密码形式登录。

```
mysql -u root -p
```

- 删除空用户，强烈建议。

```
mysql>delete from mysql.user where user="";
```

- 刷新权限表，以便可以使更改立即生效。

```
mysql>flush privileges;
```

- 输入“exit”，退出 MySQL。

```
mysql>exit
```

- 再重新以“mysql”登录测试

```
mysql
```

发现以“mysql”登录已经失效，必须以“mysql -u root -p”才能登录。

下面是执行效果截图：

```
[hadoop@TMaster ~]$ mysql -u root -p 1
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 14
Server version: 5.5.21-log MySQL Community Server (GPL)

Copyright (c) 2000, 2011, 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> delete from mysql.user where user=''; 2
Query OK, 2 rows affected (0.11 sec)

mysql> flush privileges; 3
Query OK, 0 rows affected (0.01 sec)

mysql> exit 4
Bye
[hadoop@TMaster ~]$ mysql 5
ERROR 1045 (28000): Access denied for user 'hadoop'@'localhost' (using password: NO)
[hadoop@TMaster ~]$
```

2、MapReduce与MySQL交互

MapReduce 技术推出后，曾遭到关系数据库研究者的挑剔和批评，认为 MapReduce 不具备有类似于关系数据库中的结构化数据存储和处理能力。为此，Google 和 MapReduce 社区进行了很多努力。**一方面**，他们设计了类似于关系数据中结构化数据表的技术（Google 的 BigTable，Hadoop 的 HBase）提供一些粗粒度的结构化数据存储和处理能力；**另一方面**，为了增强与关系数据库的集成能力，Hadoop MapReduce 提供了相应的访问关系数据库库的编程接口。

MapReduce 与 MySQL 交互的整体架构如下图所示。

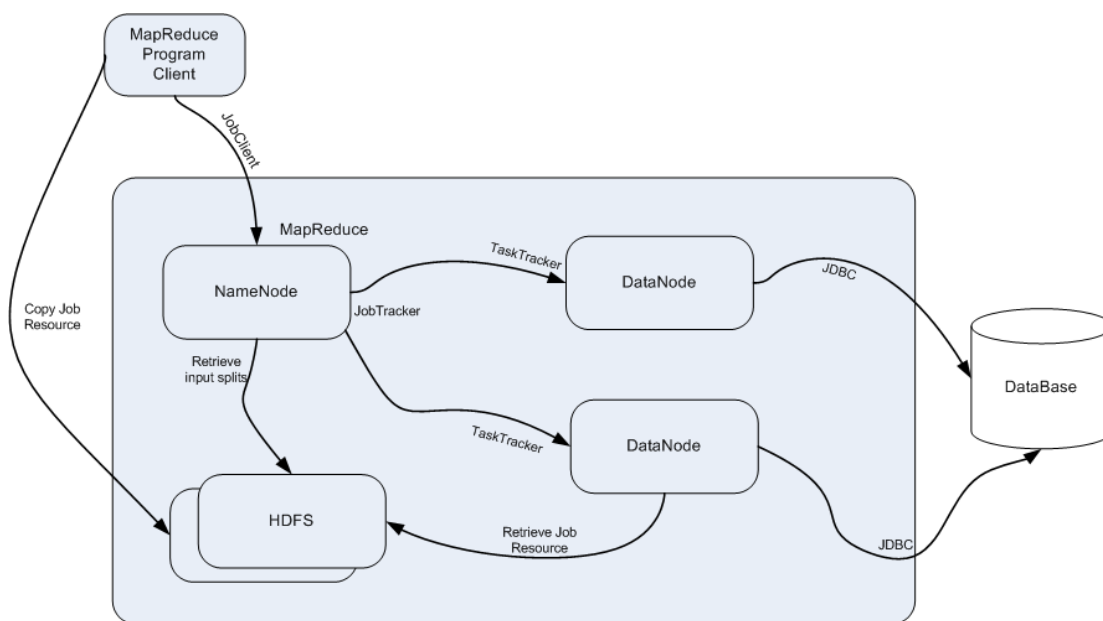


图 2-1 整个环境的架构

具体到 MapReduce 框架**读/写数据库**，有 2 个主要的程序分别是 **DBInputFormat** 和 **DBOutputFormat**，DBInputFormat 对应的是 SQL 语句 **select**，而 DBOutputFormat 对应的是 **insert/update**，使用 DBInputFormat 和 DBOutputFormat 时候需要实现 InputFormat 这个抽象类，这个抽象类含有 getSplits()和 createRecordReader()抽象方法，在 DBInputFormat 类中由 protected String getCountQuery() 方法传入结果集的个数，getSplits()方法再确定输入的切分原则，利用 SQL 中的 LIMIT 和 OFFSET 进行切分获得数据集的范围，请参考 DBInputFormat 源码中 public InputSplit[] getSplits(JobConf job, int chunks) throws IOException 的方法，在 DBInputFormat 源码中 createRecordReader()则可以按一定格式读取相应数据。

1) 建立关系数据库连接

- **DBConfiguration**：提供数据库配置和创建连接的接口。

DBConfiguration 类中提供了一个**静态方法**创建数据库连接：

```
public static void configureDB(Job job,String driverClass,String dbUrl,String userName,String Password)
```

其中，job 为当前准备执行的作业，driverClass 为数据库厂商提供的访问其数据库的驱动程序，dbUrl 为运行数据库的主机的地址，userName 和 password 分别为数据库提供访问地用户名和相应的访问密码。

2) 相应的从关系数据库查询和读取数据的接口

- **DBInputFormat**：提供从数据库读取数据的格式。
- **DBRecordReader**：提供读取数据记录的接口。

3) 相应的向关系数据库直接输出结果的编程接口

- **DBOutputFormat**：提供向数据库输出数据的格式。
- **DBRecordWrite**：提供数据库写入数据记录的接口。

数据库连接完成后，即可完成从 MapReduce 程序向关系数据库写入数据的操作。为了告知数据库将写入哪个表中的哪些字段，DBOutputFormat 中提供了一个静态方法来指定需要写入的数据表和字段：

```
public static void setOutput(Job job,String tableName,String ... fieldName)
```

其中，tableName 指定即将写入的数据表，后续参数将指定哪些字段数据将写入该表。

2.1 从数据库中输入数据

虽然 **Hadoop** 允许从数据库中**直接读取**数据记录作为 MapReduce 的输入，但**处理效率较低**，而且**大量频繁**地从 MapReduce 程序中**查询和读取**关系数据库**可能会大大增加数据库的访问负载**，因此 **DBInputFormat** 仅**适合读取少量数据记录的计算和应用**，**不适合数据库联机数据分析大量数据的读取处理**。

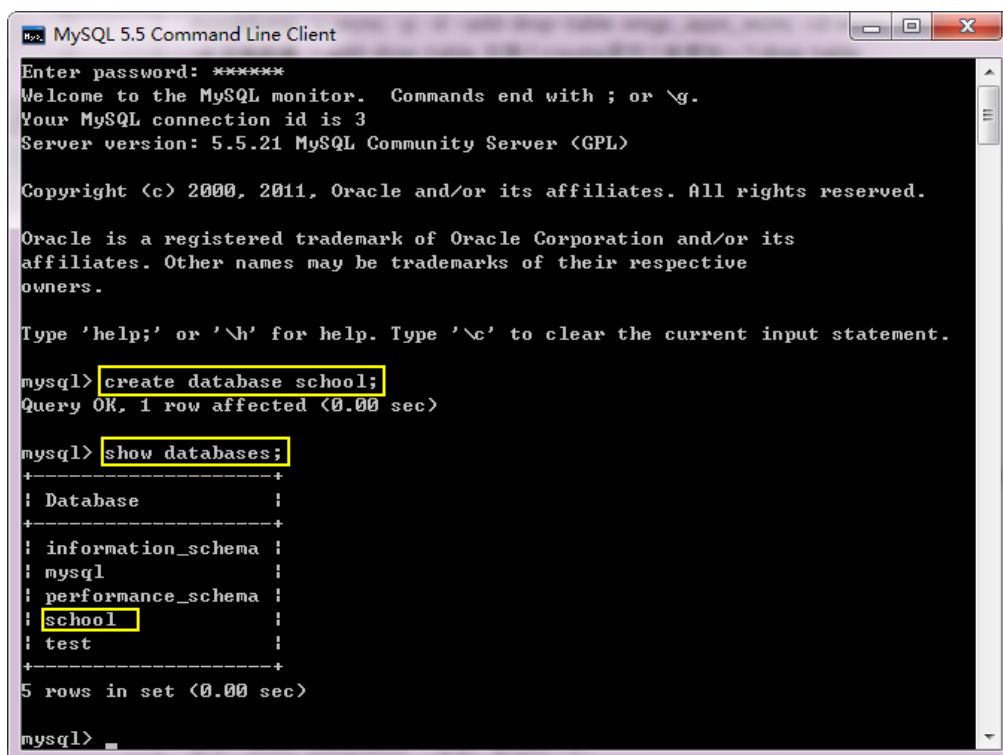
读取大量数据记录一个更好的解决办法是：用**数据库**中的 **Dump** 工具将大量**待分析数据输出**为**文本数据文件**，并上载到 HDFS 中进行处理。

1) 首先创建要读入的数据

- **Windows 环境**

首先创建数据库“school”，使用下面命令进行：

```
create database school;
```



然后通过以下几句话, 把我们事先准备好的 sql 语句 (student.sql 事先放到了 D 盘目录) 导入到刚创建的 “school” 数据库中。用到的命令如下:

```
use school;
source d:\student.sql
```

“student.sql” 中的内容如下所示:

```
DROP TABLE IF EXISTS `school`.`student`;

CREATE TABLE `school`.`student` (
  `id` int(11) NOT NULL default '0',
  `name` varchar(20) default NULL,
  `sex` varchar(10) default NULL,
  `age` int(10) default NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

INSERT INTO `student` VALUES ('201201', '张三', '男', '21');
INSERT INTO `student` VALUES ('201202', '李四', '男', '22');
INSERT INTO `student` VALUES ('201203', '王五', '女', '20');
```

```
INSERT INTO `student` VALUES ('201204','赵六','男','21');
INSERT INTO `student` VALUES ('201205','小红','女','19');
INSERT INTO `student` VALUES ('201206','小明','男','22');
```

执行结果如下所示：

```
mysql> use school;
Database changed
mysql> source d:\student.sql
Query OK, 0 rows affected, 1 warning (0.00 sec)

Query OK, 0 rows affected (0.08 sec)

Query OK, 1 row affected (0.03 sec)

Query OK, 1 row affected (0.02 sec)

Query OK, 1 row affected (0.01 sec)

Query OK, 1 row affected (0.03 sec)

Query OK, 1 row affected (0.09 sec)

Query OK, 1 row affected (0.02 sec)

mysql>
```

查询刚才创建的数据库表“student”的内容。

```
mysql> select * from student;
+----+-----+-----+-----+
| id  | name  | sex  | age  |
+----+-----+-----+-----+
| 201201 | 容独竿 | 男? | 21 |
| 201202 | 嫩衣凉 | 男? | 22 |
| 201203 | 黎嫫脾 | 男? | 20 |
| 201204 | 壁崧段 | 男? | 21 |
| 201205 | 灏伙孩 | 男? | 19 |
| 201206 | 灏吃霖 | 男? | 22 |
+----+-----+-----+-----+
6 rows in set (0.01 sec)

mysql>
```

结果发现显示是乱码，记得我当时是设置的 UTF-8，怎么就出现乱码了呢？其实我们使用的操作系统的系统为中文，且它的默认编码是 gbk，而 MySQL 的编码有两种，它们分别是：

【client】：客户端的字符集。客户端默认字符集。当客户端向服务器发送请求时，请求以该字符集进行编码。

【mysqld】：服务器字符集，默认情况下所采用的。

找到安装 MySQL 目录，比如我们的安装目录为：

```
E:\HadoopWorkPlat\MySQL Server 5.5
```


从中找到“my.ini”配置文件，最终发现 my.ini 里的 2 个 character_set 把 client 改成 gbk，把 server 改成 utf8 就可以了。

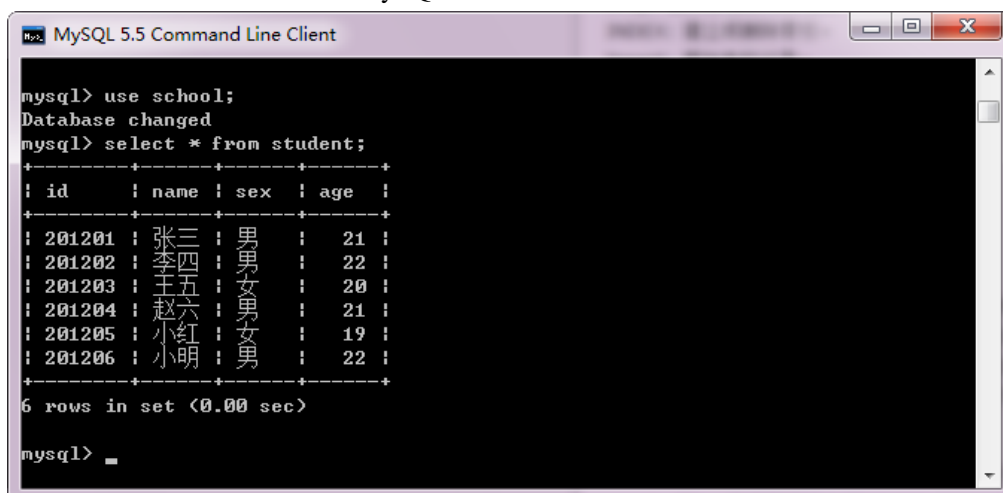
【client】端：

```
[client]
port=3306
[mysql]
default-character-set=gbk
```

【mysqld】端：

```
[mysqld]
# The default character set that will be used when a new schema or table is
# created and no character set is defined
character-set-server=utf8
```

按照上面修改完之后，重启 MySQL 服务。



```
MySQL 5.5 Command Line Client

mysql> use school;
Database changed
mysql> select * from student;
+----+-----+-----+-----+
| id  | name | sex  | age |
+----+-----+-----+-----+
| 201201 | 张三 | 男   | 21  |
| 201202 | 李四 | 男   | 22  |
| 201203 | 王五 | 女   | 20  |
| 201204 | 赵六 | 男   | 21  |
| 201205 | 小红 | 女   | 19  |
| 201206 | 小明 | 男   | 22  |
+----+-----+-----+-----+
6 rows in set (0.00 sec)

mysql>
```

此时在 Windows 下面的数据库表已经准备完成了。

● Linux 环境

首先通过“FlashFXP”把我们刚才的“student.sql”上传到“/home/hadoop”目录下，然后按照上面的语句创建“school”数据库。

```
[hadoop@TMaster ~]$ ll
总用量 208132
-rw-r--r-- 1 hadoop hadoop 59468784 3月 15 17:57 hadoop-1.0.0.tar.gz
-rw-r--r-- 1 hadoop hadoop 85292206 3月 15 17:58 jdk-6u31-linux-i586.bin
-rw-r--r-- 1 hadoop hadoop 16988103 3月 14 18:44 MySQL-client-5.5.21-1.linux2.6.i386.rpm
-rw-r--r-- 1 hadoop hadoop 51367247 3月 14 18:45 MySQL-server-5.5.21-1.linux2.6.i386.rpm
-rw-r--r-- 1 hadoop hadoop 678 3月 16 00:10 student.sql
[hadoop@TMaster ~]$
```

查看我们上传的“student.sql”内容：

```
[hadoop@TMaster ~]$ more student.sql
DROP TABLE IF EXISTS `school`.`student`;

CREATE TABLE `school`.`student` (
  `id` int(11) NOT NULL default '0',
  `name` varchar(20) default NULL,
  `sex` varchar(10) default NULL,
  `age` int(10) default NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

INSERT INTO `student` VALUES ('201201', '张三', '男', '21');
INSERT INTO `student` VALUES ('201202', '李四', '男', '22');
INSERT INTO `student` VALUES ('201203', '王五', '女', '20');
INSERT INTO `student` VALUES ('201204', '赵六', '男', '21');
INSERT INTO `student` VALUES ('201205', '小红', '女', '19');
INSERT INTO `student` VALUES ('201206', '小明', '男', '22');
[hadoop@TMaster ~]$
```

创建“school”数据库，并导入“student.sql”语句。

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

mysql> use school;
Database changed
mysql> source /home/hadoop/student.sql
Query OK, 0 rows affected, 1 warning (0.02 sec)

Query OK, 0 rows affected (0.18 sec)

Query OK, 1 row affected (0.11 sec)

Query OK, 1 row affected (0.09 sec)

Query OK, 1 row affected (0.08 sec)

Query OK, 1 row affected (0.09 sec)

Query OK, 1 row affected (0.08 sec)

Query OK, 1 row affected (0.08 sec)

mysql>
```

显示数据库“school”中的表“student”信息。

```
mysql> show tables;
+-----+
| Tables_in_school |
+-----+
| student          |
+-----+
1 row in set (0.01 sec)
```

显示表“student”中的内容。

```
mysql> select * from student;
```

id	name	sex	age
201201	张三	男	21
201202	李四	男	22
201203	王五	女	20
201204	赵六	男	21
201205	小红	女	19
201206	小明	男	22

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

到此为止在“Windows”和“Linux”两种环境下面都创建了表“student”表，并初始化了值。下面就开始通过 MapReduce 读取 MySQL 库中表“student”的信息。

2) 使 MySQL 能远程连接

MySQL 默认是允许别的机器进行远程访问地，为了使 Hadoop 集群能访问 MySQL 数据库，所以进行下面操作。

- 用 MySQL 用户“root”登录。

```
mysql -u root -p
```

- 使用下面语句进行授权，赋予任何主机访问数据的权限。

```
GRANT ALL PRIVILEGES ON *.* TO 'root'@'%' IDENTIFIED BY 'hadoop' WITH GRANT OPTION;
```

- 刷新，使之立即生效。

```
FLUSH PRIVILEGES;
```

执行结果如下图。

Windows 下面：

```
mysql> grant all privileges on *.* to 'root'@'%' identified by 'hadoop' with grant option;
Query OK, 0 rows affected (0.15 sec)

mysql> flush privileges;
Query OK, 0 rows affected (0.21 sec)

mysql>
```

Linux 下面：

```
mysql> grant all privileges on *.* to 'root'@'%' identified by 'hadoop' with grant option;
Query OK, 0 rows affected (0.00 sec)

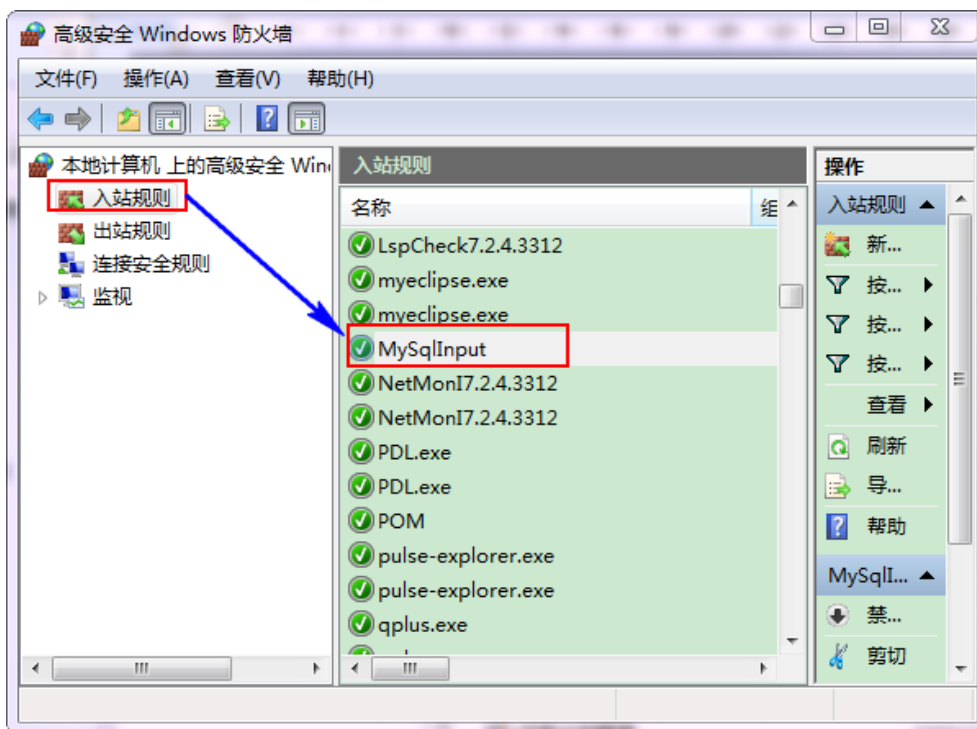
mysql> flush privileges;
Query OK, 0 rows affected (0.00 sec)
```

到目前为止, 如果连接 **Win7** 上面的 MySQL 数据库还不行, 大家还应该记得前面在 **Linux** 下面**关掉了防火墙**, 但是我们在 Win7 下对防火墙并没有做任何处理, 如果不对防火墙做处理, 即使执行了上面的远程授权, 仍然不能连接。下面是设置 Win7 上面的防火墙, 使远程机器能通过 3306 端口访问 MySQL 数据库。

解决方案: 只要在‘入站规则’上建立一个 3306 端口即可。

执行顺序: 控制面板→管理工具→高级安全的 Windows 防火墙→入站规则

然后**新建规则**→选择‘端口’→在‘特定本地端口’上输入一个‘**3306**’→选择‘允许连接’=>选择‘域’、‘专用’、‘公用’=>给个名称, 如: MySqlInput



3) 对 JDBC 的 Jar 包处理

因为程序虽然用 Eclipse 编译运行但最终要提交到 Hadoop 集群上, 所以 JDBC 的 jar 必须放到 Hadoop 集群中。有两种方式:

(1) 在每个节点下的\${HADOOP_HOME}/lib 下添加该包, 重启集群, 一般是比较原始的方法。

我们的 Hadoop 安装包在“**/usr/hadoop**”, 所以把 Jar 放到“**/usr/hadoop/lib**”下面, 然后重启, 记得是 Hadoop 集群中所有的节点都要放, 因为执行分布式程序是在每个节点本地机器上进行。

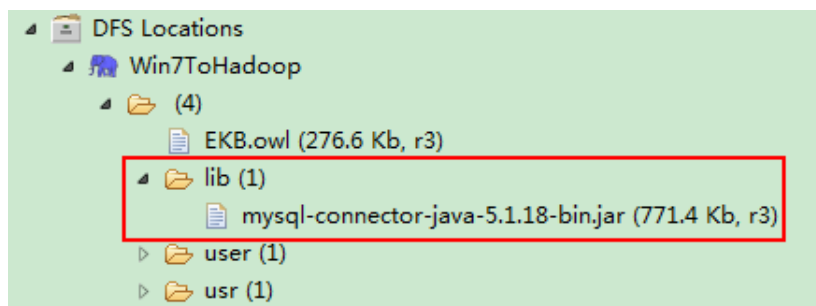
(2) 在 Hadoop 集群的分布式文件系统中创建“lib”文件夹, 并把我们的 JDBC 的 jar 包上传上去, 然后在主程序添加如下语句, 就能保证 Hadoop 集群中所有的节点都能使用这个 jar 包。因为这个 jar 包放在了 HDFS 上, 而不是本地系统, 这个要理解清楚。

```
DistributedCache.addFileToClassPath(new Path("/lib/mysql-connector-java-5.1.18-bin.jar"), conf);
```

我们用的 JDBC 的 jar 如下所示:

```
mysql-connector-java-5.1.18-bin.jar
```

通过 Eclipse 下面的 DFS Locations 进行创建 “/lib” 文件夹，并上传 JDBC 的 jar 包。执行结果如下：



备注：我们这里采用了第二种方式。

4) 源程序代码如下所示

```
package com.hebut.mr;

import java.io.IOException;
import java.io.DataInput;
import java.io.DataOutput;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;

import org.apache.hadoop.filecache.DistributedCache;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.io.Writable;
import org.apache.hadoop.mapred.JobClient;
import org.apache.hadoop.mapred.JobConf;
import org.apache.hadoop.mapred.MapReduceBase;
import org.apache.hadoop.mapred.Mapper;
import org.apache.hadoop.mapred.OutputCollector;
import org.apache.hadoop.mapred.FileOutputFormat;
import org.apache.hadoop.mapred.Reporter;
import org.apache.hadoop.mapred.lib.IdentityReducer;
import org.apache.hadoop.mapred.lib.db.DBWritable;
import org.apache.hadoop.mapred.lib.db.DBInputFormat;
import org.apache.hadoop.mapred.lib.db.DBConfiguration;

public class ReadDB {

    public static class Map extends MapReduceBase implements
```

```
Mapper<LongWritable, StudentRecord, LongWritable, Text> {

    // 实现 map 函数
    public void map(LongWritable key, StudentRecord value,
        OutputCollector<LongWritable, Text> collector, Reporter reporter)
        throws IOException {
        collector.collect(new LongWritable(value.id),
            new Text(value.toString()));
    }

}

public static class StudentRecord implements Writable, DBWritable {
    public int id;
    public String name;
    public String sex;
    public int age;

    @Override
    public void readFields(DataInput in) throws IOException {
        this.id = in.readInt();
        this.name = Text.readString(in);
        this.sex = Text.readString(in);
        this.age = in.readInt();
    }

    @Override
    public void write(DataOutput out) throws IOException {
        out.writeInt(this.id);
        Text.writeString(out, this.name);
        Text.writeString(out, this.sex);
        out.writeInt(this.age);
    }

    @Override
    public void readFields(ResultSet result) throws SQLException {
        this.id = result.getInt(1);
        this.name = result.getString(2);
        this.sex = result.getString(3);
        this.age = result.getInt(4);
    }

    @Override
    public void write(PreparedStatement stmt) throws SQLException {
```

```
        stmt.setInt(1, this.id);
        stmt.setString(2, this.name);
        stmt.setString(3, this.sex);
        stmt.setInt(4, this.age);
    }

    @Override
    public String toString() {
        return new String("学号: " + this.id + "_姓名: " + this.name
            + "_性别: " + this.sex + "_年龄: " + this.age);
    }
}

public static void main(String[] args) throws Exception {

    JobConf conf = new JobConf(ReadDB.class);

    // 这句话很关键
    conf.set("mapred.job.tracker", "192.168.1.2:9001");

    // 非常重要，值得关注
    DistributedCache.addFileToClassPath(new Path(
        "/lib/mysql-connector-java-5.1.18-bin.jar"), conf);

    // 设置输入类型
    conf.setInputFormat(DBInputFormat.class);

    // 设置输出类型
    conf.setOutputKeyClass(LongWritable.class);
    conf.setOutputValueClass(Text.class);

    // 设置 Map 和 Reduce 类
    conf.setMapperClass(Map.class);
    conf.setReducerClass(IdentityReducer.class);

    // 设置输出目录
    FileOutputFormat.setOutputPath(conf, new Path("rdb_out"));

    // 建立数据库连接
    DBConfiguration.configureDB(conf, "com.mysql.jdbc.Driver",
        "jdbc:mysql://192.168.1.24:3306/school", "root", "hadoop");

    // 读取“student”表中的数据
    String[] fields = { "id", "name", "sex", "age" };
```



```

DBInputFormat.setInput(conf, StudentRecord.class, "student", null,
    "id", fields);

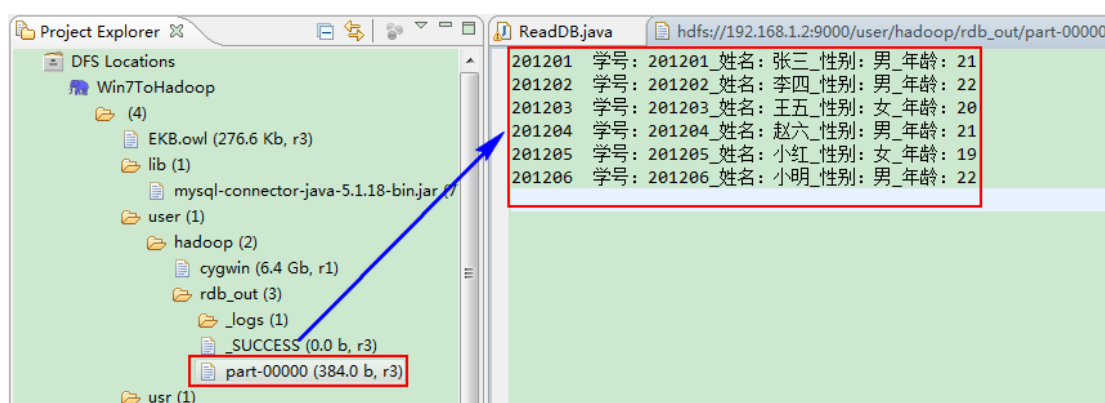
JobClient.runJob(conf);
}
}

```

备注: 由于 Hadoop1.0.0 新的 API 对关系型数据库 **暂不支持**, 只能用旧的 API 进行, 所以下面的“向数据库中输出数据”也是如此。

5) 运行结果如下所示

经过上面的设置后, 已经通过连接 Win7 和 Linux 上的 MySQL 数据库, 执行结果都一样。唯独变得就是代码中“DBConfiguration.configureDB”中 MySQL 数据库所在机器的 IP 地址。



2.2 向数据库中输出数据

基于**数据仓库**的**数据分析**和**挖掘输出结果**的数据量**一般不会太大**, 因而**可能适合于直接向数据库写入**。我们这里尝试与“WordCount”程序相结合, 把单词统计的结果存入到关系型数据库中。

1) 创建写入的数据库表

我们还使用刚才创建的数据库“school”, 只是在里添加一个新的表“wordcount”, 还是使用下面语句执行:

```

use school;
source sql 脚本全路径

```

下面是要创建的“wordcount”表的 sql 脚本。

```

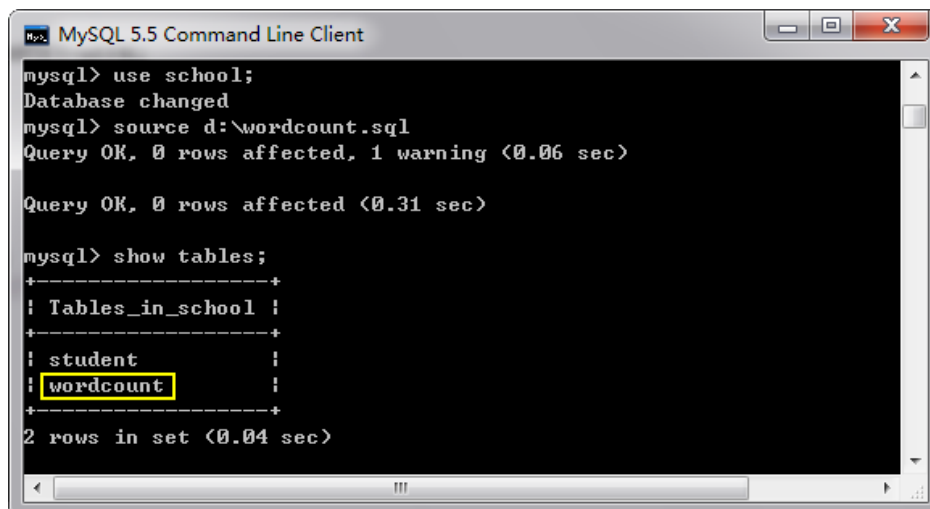
DROP TABLE IF EXISTS `school`.`wordcount`;

```

```
CREATE TABLE `school`.`wordcount` (  
  `id` int(11) NOT NULL auto_increment,  
  `word` varchar(20) default NULL,  
  `number` int(11) default NULL,  
  PRIMARY KEY (`id`)  
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

执行效果如下所示：

- Windows 环境



```
mysql> use school;  
Database changed  
mysql> source d:\wordcount.sql  
Query OK, 0 rows affected, 1 warning (0.06 sec)  
  
Query OK, 0 rows affected (0.31 sec)  
  
mysql> show tables;  
+-----+  
| Tables_in_school |  
+-----+  
| student          |  
| wordcount         |  
+-----+  
2 rows in set (0.04 sec)
```

- Linux 环境

```
mysql> use school;  
Database changed  
mysql> source /home/hadoop/wordcount.sql  
Query OK, 0 rows affected, 1 warning (0.00 sec)  
  
Query OK, 0 rows affected (0.11 sec)  
  
mysql> show tables;  
+-----+  
| Tables_in_school |  
+-----+  
| student          |  
| wordcount         |  
+-----+  
2 rows in set (0.00 sec)
```

2) 程序源代码如下所示

```
package com.hebut.mr;  
  
import java.io.IOException;  
import java.io.DataInput;  
import java.io.DataOutput;
```

```
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.Iterator;
import java.util.StringTokenizer;

import org.apache.hadoop.filecache.DistributedCache;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.io.Writable;
import org.apache.hadoop.mapred.FileInputFormat;
import org.apache.hadoop.mapred.JobClient;
import org.apache.hadoop.mapred.JobConf;
import org.apache.hadoop.mapred.MapReduceBase;
import org.apache.hadoop.mapred.Mapper;
import org.apache.hadoop.mapred.OutputCollector;
import org.apache.hadoop.mapred.Reducer;
import org.apache.hadoop.mapred.Reporter;
import org.apache.hadoop.mapred.TextInputFormat;
import org.apache.hadoop.mapred.lib.db.DBOutputFormat;
import org.apache.hadoop.mapred.lib.db.DBWritable;
import org.apache.hadoop.mapred.lib.db.DBConfiguration;

public class WriteDB {
    // Map 处理过程
    public static class Map extends MapReduceBase implements
        Mapper<Object, Text, Text, IntWritable> {

        private final static IntWritable one = new IntWritable(1);
        private Text word = new Text();

        @Override
        public void map(Object key, Text value,
            OutputCollector<Text, IntWritable> output, Reporter reporter)
            throws IOException {
            String line = value.toString();
            StringTokenizer tokenizer = new StringTokenizer(line);
            while (tokenizer.hasMoreTokens()) {
                word.set(tokenizer.nextToken());
                output.collect(word, one);
            }
        }
    }
}
```

```
// Combine 处理过程
public static class Combine extends MapReduceBase implements
    Reducer<Text, IntWritable, Text, IntWritable> {

    @Override
    public void reduce(Text key, Iterator<IntWritable> values,
        OutputCollector<Text, IntWritable> output, Reporter reporter)
        throws IOException {
        int sum = 0;
        while (values.hasNext()) {
            sum += values.next().get();
        }
        output.collect(key, new IntWritable(sum));
    }
}

// Reduce 处理过程
public static class Reduce extends MapReduceBase implements
    Reducer<Text, IntWritable, WordRecord, Text> {

    @Override
    public void reduce(Text key, Iterator<IntWritable> values,
        OutputCollector<WordRecord, Text> collector, Reporter reporter)
        throws IOException {

        int sum = 0;
        while (values.hasNext()) {
            sum += values.next().get();
        }

        WordRecord wordcount = new WordRecord();
        wordcount.word = key.toString();
        wordcount.number = sum;

        collector.collect(wordcount, new Text());
    }
}

public static class WordRecord implements Writable, DBWritable {
    public String word;
    public int number;

    @Override
```

```
public void readFields(DataInput in) throws IOException {
    this.word = Text.readString(in);
    this.number = in.readInt();
}

@Override
public void write(DataOutput out) throws IOException {
    Text.writeString(out, this.word);
    out.writeInt(this.number);
}

@Override
public void readFields(ResultSet result) throws SQLException {
    this.word = result.getString(1);
    this.number = result.getInt(2);
}

@Override
public void write(PreparedStatement stmt) throws SQLException {
    stmt.setString(1, this.word);
    stmt.setInt(2, this.number);
}
}

public static void main(String[] args) throws Exception {

    JobConf conf = new JobConf(WriteDB.class);

    // 这句话很关键
    conf.set("mapred.job.tracker", "192.168.1.2:9001");

    DistributedCache.addFileToClassPath(new Path(
        "/lib/mysql-connector-java-5.1.18-bin.jar"), conf);

    // 设置输入输出类型
    conf.setInputFormat(TextInputFormat.class);
    conf.setOutputFormat(DBOutputFormat.class);
    // 不加这两句，通不过，但是网上给的例子没有这两句。
    conf.setOutputKeyClass(Text.class);
    conf.setOutputValueClass(IntWritable.class);

    // 设置 Map 和 Reduce 类
    conf.setMapperClass(Map.class);
    conf.setCombinerClass(Combine.class);
}
```

```

conf.setReducerClass(Reduce.class);

// 设置输入目录
FileInputFormat.setInputPaths(conf, new Path("wdb_in"));

// 建立数据库连接
DBConfiguration.configureDB(conf, "com.mysql.jdbc.Driver",
    "jdbc:mysql://192.168.1.24:3306/school", "root", "hadoop");

// 写入“wordcount”表中的数据
String[] fields = { "word", "number" };
DBOutputFormat.setOutput(conf, "wordcount", fields);

JobClient.runJob(conf);
}
}

```

3) 运行结果如下所示

● Windows 环境

测试数据：

(1) file1.txt

```

hello word
hello hadoop

```

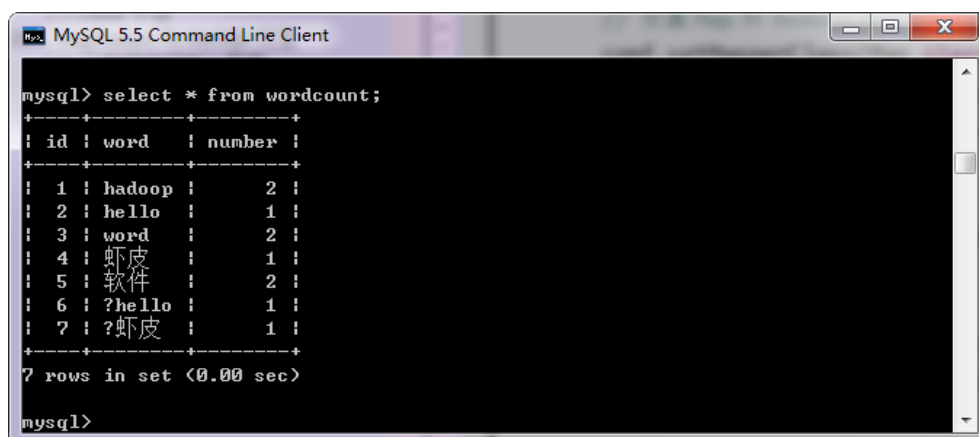
(2) file2.txt

```

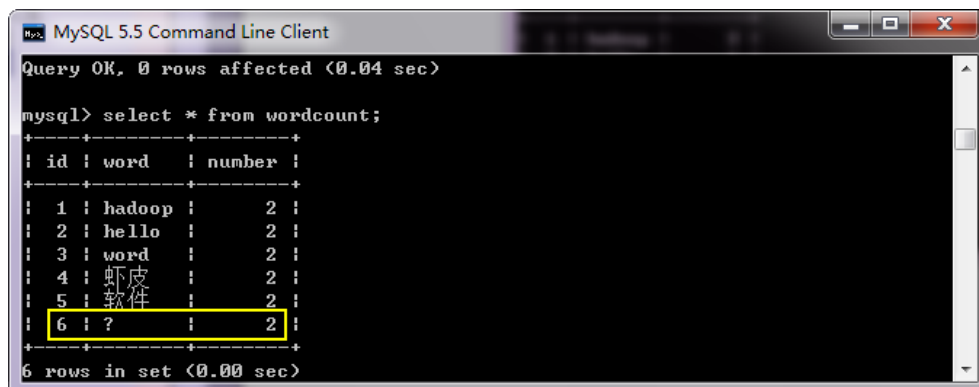
虾皮 hadoop
虾皮 word
软件 软件

```

运行结果：



我们发现上图中出现了“?”，后来查找原来是因为我的测试数据时在 Windows 用记事本写的然后保存为“UTF-8”，在保存时为了区分编码，自动在前面加了一个“**BOM**”，但是不会显示任何结果。然而我们的代码把它识别为“?”进行处理。这就出现了上面的结果，如果我们在每个要处理的文件前面的**第一行加一个空格**，结果就成如下显示：



接着又做了一个测试，在 **Linux** 上面用下面命令创建了一个文件，并写上中文内容。结果显示并没有出现“?”，而且网上说不同的记事本软件（EmEditor、UE）保存为“UTF-8”就**没有**这个问题。经过修改之后的 Map 类，就能够正常识别了。

```
// Map 处理过程
public static class Map extends MapReduceBase implements
    Mapper<Object, Text, Text, IntWritable> {

    private final static IntWritable one = new IntWritable(1);
    private Text word = new Text();

    @Override
    public void map(Object key, Text value,
        OutputCollector<Text, IntWritable> output, Reporter reporter)
        throws IOException {
        String line = value.toString();

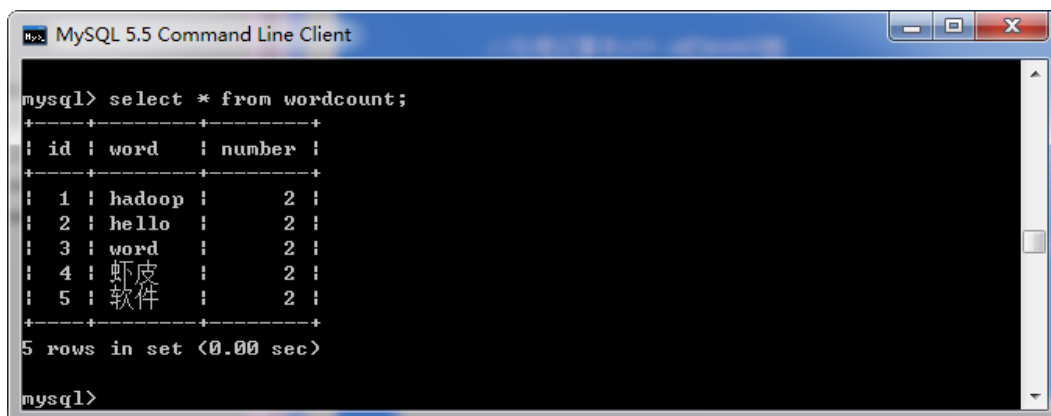
        //处理记事本 UTF-8 的 BOM 问题
        if (line.getBytes().length > 0) {
            if ((int) line.charAt(0) == 65279) {
                line = line.substring(1);
            }
        }

        StringTokenizer tokenizer = new StringTokenizer(line);
        while (tokenizer.hasMoreTokens()) {
            word.set(tokenizer.nextToken());
            output.collect(word, one);
        }
    }
}
```



```
}
}
```

处理之后的结果：



从上图中得知，我们的问题已经解决了，因此，在编辑、更改任何文本文件时，请务必使用不会乱加 BOM 的编辑器。Linux 下的编辑器应该都没有这个问题。Windows 下，请勿使用记事本等编辑器。推荐的编辑器是：Editplus 2.12 版本以上；EmEditor；UltraEdit（需要取消‘添加 BOM’的相关选项）；Dreamweaver（需要取消‘添加 BOM’的相关选项）等。

对于已经添加了 BOM 的文件，要取消的话，可以用以上编辑器另存一次。（Editplus 需要先另存为 gb，再另存为 UTF-8。）DW 解决办法如下：用 DW 打开指定文件，按 Ctrl+J→标题/编码→编码选择“UTF-8”，去掉“包括 Unicode 签名(BOM)”勾选→保存/另存为，即可。

国外有一个牛人已经把这问题解决了，使用“UnicodeInputStream”、“UnicodeReader”。

地址：<http://koti.mbnet.fi/akini/java/unicodereader/>

示例：[Java读带有BOM的UTF-8 文件乱码原因及解决方法](#)

代码：<http://download.csdn.net/detail/xia520pi/4146123>

● Linux 环境

测试数据：

(1) file1.txt

```
MapReduce is simple
```

(2) file2.txt

```
MapReduce is powerful is simple
```

(3) file2.txt

```
Hello MapReduce bye MapReduce
```

运行结果：

```
mysql> use school;
Database changed
mysql> select * from wordcount;
+-----+-----+-----+
| id | word      | number |
+-----+-----+-----+
| 1  | Hello     | 1      |
| 2  | MapReduce | 4      |
| 3  | bye       | 1      |
| 4  | is        | 3      |
| 5  | powerful  | 1      |
| 6  | simple    | 2      |
+-----+-----+-----+
6 rows in set (0.00 sec)
```

到目前为止，MapReduce 与关系型数据库交互已经结束，从结果中得知，目前新版的 API 还不能很好的支持关系型数据库的操作，上面两个例子都是使用的旧版的 API。关于更多的 MySQL 操作，具体参考“[Hadoop 集群_第 10 期副刊_常用 MySQL 数据库命令_V1.0](#)”。

本期历时五天，终于完成，期间遇到的关键问题如下：

- MySQL 的 JDBC 的 jar 存放问题。
- Win7 对 MySQL 防火墙的设置。
- Linux 中 MySQL 变更目录不能启动。
- MapReduce 处理带 BOM 的 UTF-8 问题。
- 设置 MySQL 可以远程访问。
- MySQL 处理中文乱码问题。

从这几天对 MapReduce 的了解，发现其实 Hadoop 对关系型数据库的处理还不是很强，主要是 Hadoop 和关系型数据做的事不是同一类型，各有所特长。下面几期我们将对 Hadoop 里的 HBase 和 Hive 进行全面了解。

编者简介

基本信息

姓 名：解耀伟 性 别：男
笔 名：虾皮 民 族：汉
学 历：研究生 专 业：计算机应用技术
电子信箱：xieyaowei1986@163.com
学 校：河北工业大学（211 工程）



求职意向

希望在 IT 行业从事软件开发等工作。

编程语言

Java、C#、C、ExtJS、Flex、汇编、PHP、VB，熟练程度由左到右逐级减弱。

个人经历

大学期间

- 1) 担任职务：学生会生活部部长、生活委员、团支书
- 2) 获得奖项：二等奖学金（2 次）、三好学生（1 次）

研究生期间

- 1) 担任职务：班长
- 2) 获得奖项：优秀班干部（1 次）

工作经历

实验室项目：国家 863 计划项目 1 项；国家技术基础专项 2 项；河北省技术专项 1 项。

研究生课题：基于 Hadoop 分布式搜索引擎研究

个人评价

性格开朗，善于与人沟通，上进心强，品德优秀，吃苦耐劳，喜欢团队合作，能积极服从上级的安排。

寄 言

相信您的信任与我的能力将为我们带来共同的成功。

参考文献

感谢以下文章的编写作者，没有你们的铺路，我或许会走得很艰难，参考不分先后，贡献同等珍贵。

【1】Hadoop 实战——陆嘉恒——机械工业出版社

【2】实战 Hadoop——刘鹏——电子工业出版社

【3】Linux 系统上安装 MySQL 5.5rpm

地址: <http://www.cnblogs.com/sunson/articles/2172086.html>

【4】linux 下 mysql(rpm)安装使用手册

地址: <http://www.cnblogs.com/sunwei2012/archive/2011/02/16/1956547.html>

【5】Linux 下如何更改 Mysql 默认的数据文件目录

地址: <http://hi.baidu.com/braveboyx/blog/item/38a169f2ea0237d57831aa5f.html>

【6】设置 MySql5.5 数据库的字符编码为 UTF8，解决中文乱码问题

地址: <http://kimho.iteye.com/blog/1141608>

【7】hadoop 与 mysql 数据库相连读出数据

地址: <http://blog.csdn.net/qwertyu8656/article/details/6426054>

【8】Hadoop 中 DBInputFormat 和 DBOutputFormat 使用

地址: <http://blog.csdn.net/dajuezhao/article/details/5799371>