MYSQL

write by Tube
zi.jiao.liu@gmail.com

前言

MYSQL 是时下非常流行的数据库软件。学习 MYSQL 的知识可以帮你打开数据库领域的大门。最好的 MYSQL 教材就是 MYSQL 官方的手册。但是，那本手册涉及的范围对于初学者来说有些过于繁杂。因此本书的定位是一本面向初学者的教材，试图以一个操作者的身份，带领您一步一步接触 MYSQL，并最终能够完成 MYSQL 的基本操作与维护。同时，需要注意的是，本书默认你是一名具有 Linux 操作基础的读者，因此当涉及到一些 Linux 操作的知识的时候，本书不会做过多讲解。

这本书一共分为十章。在第一章，主要介绍了 MySQL 的一些基本介绍以及如何获取相关技术文档。第二章，我们学习如何安装 MySQL 数据库。其中我们将介绍 Redhat RPM 包安装方式和 Linux 源代码安装方式。在学会如何安装 MySQL 之后，本书将在第三章带着大家去了解一些 MYSQL 的常用工具以及 MySQL 客户端的使用。第四章，SQL 语言基础。第五章，MYSQL 操作基础。第六章，MySQL 备份。第七章，使用 PHPMYADMIN 管理 MYSQL。第八章，MYSQL 优化。第九章，MYSQL 簇。第十章，用源代码包搭建 LAMP 环境。

刘子佼　　tube@uplooking.com
zi.jiao.liu@gmail.com

2008-2-3

本文目录

第1章 概述

感谢您学习“尚观科技”的MySQL课程。有些同学一听到数据库三个字就很头疼。总是认为，数据库是一门非常深奥的学问。只有那些智商120以上的家伙才能搞定。其实，不用那么担心。我们接下来所学习的MYSQL内容的主要目的是让大家能够用起来，能够达到基本使用、管理和维护一个MYSQL数据库的程度。因此，不要过于担心，这些都是比较简单的，只要你花点心思多做一些练习和实验，就没有什么你搞不定的。

MYSQL是个很好的小型数据库。把它当作数据库入门学习对象是很棒的选择，个人认为至少比FOXBASE或者ACCESS要好得多。那么什么是MySQL?它又有那些特点？它遵循什么标准？支持哪些操作系统？从哪里可以获取它呢？本章将针对这些问题给您答案。

第1.1节 什么是MYSQL？

MySQL是采用客户／服务器模型的开放源码关系型SQL数据库管理系统，它可以在多种操作系统上运行，它是由MySQL Ab公司开发、发布并支持的。

上面这段话是不是很拗口，读起来很迷糊？没关系，我们把这段话拆开来，一块一块来理解。

MySQL数据库软件是一种**客户端/服务器**系统，对于客户/服务器模型大家在前面的课程中应该已经有所了解了。MYSQL的客户端/服务器模型的体现是由一个支持不同后端的多线程SQL服务器，数种不同的客户端程序和库，众多管理工具和广泛的应用编程接口API组成。

所谓**关系型**是指，将数据保存在不同的表中，而不是将所有数据放在一个大的仓库内。这样就增加了速度并提高了灵活性。

MySQL中的**SQL**指的是“结构化查询语言”。SQL是访问数据库的最常用的标准化语言，它是由ANSI/ISO SQL标准定义的。SQL标准自1986年以来不断的演化和发展，有数种版本。我们常说的“SQL92”指的是1992年发布的SQL标准，“SQL99”则是1999年发布的SQL标准，目前最新的SQL标准为“SQL2003”。MySQL AB声称“我们致力于支持全套ANSI/ISO SQL标准，但不会以牺牲代码的速度和质量为代价。”

MySQL 是一种**开放源代码**的软件，它遵循 GPL 协议。您可以在以下网站中了解到关于 GPL 协议的详细内容：http://www.fsf.org/licenses/

MySQL AB 是由 MySQL 创始人和主要开发人创办的公司。MySQL AB 最初是由 David Axmark、Allan Larsson 和 Michael "Monty" Widenius 在瑞典创办的。

MySQL 可以在**多种操作系统**上运行，包括 Linux、Solaris、MacOS X、Windows。使用了 MyISAM 存储引擎的 MySQL ，最大表尺寸增加到了 65536TB（256^7 — 1 字节），但是对于不同的操作系统所支持的文件系统，MySQL 表的尺寸是受到限制的。因此，可以这样说：MySQL 数据库的最大有效表尺寸通常是由操作系统对文件大小的限制决定的，而不是由 MySQL 内部限制决定的。下表是部分操作系统的表大小限制。

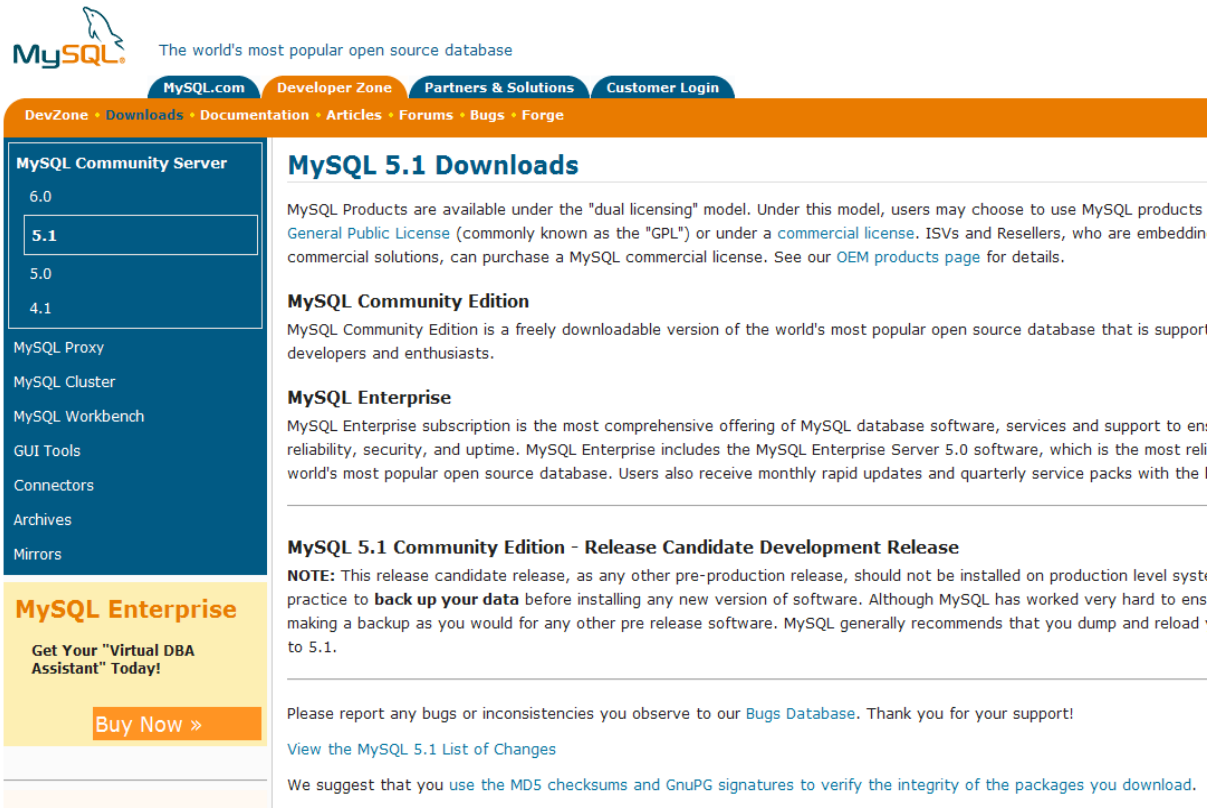
Linux 2.2-Intel 32-bit	2GB (LFS: 4GB)
Linux 2.4+	(using ext3 filesystem) 4TB
Solaris 9/10	16TB
NetWare w/NSS filesystem	8TB
win32 w/ FAT/FAT32	2GB/4GB
win32 w/ NTFS	2TB（可能更大）
MacOS X w/ HFS	2TB

第1.2节 获取 MYSQL 及获取帮助

MySQL 是开放源代码的软件，因此你可以很方便的直接在其官方网站上下载到它。官方网站的地址为：http://www.mysql.com （如图 1.1）

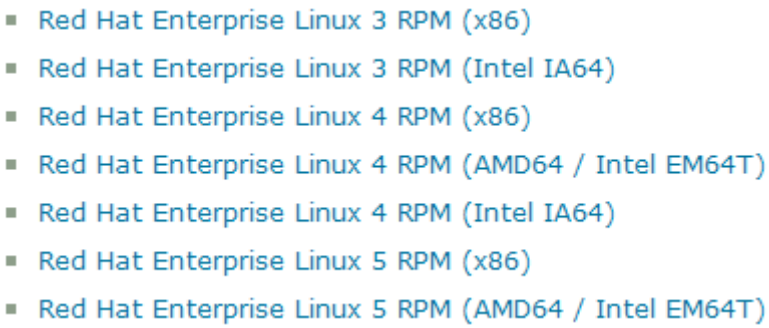
（图1.1）

下载MySQL的页面地址为http://dev.mysql.com/downloads/ 在该页面当中，你可以看到针对于很多不同操作系统的专门版本提供下载。（如图1.2）



（图 1.2）

Linux 的下载版本有针对于不同发行版本的安装包。比如，对于 RedHat 公司发布的 RedHat Enterprise 3、RedHat Enterprise4、RedHat Enterprise5 都有专门的 RPM 安装包，并且还详细区分了 x86、AMD64 / Intel EM64T、Intel IA64 三种不同选择。（如图 1.3）记住选择符合你的操作系统版本和机器情况的安装包进行下载。当然，你也可以选择源代码包下载，然后自己编译安装一个 MySQL。



（图 1.3）

获取 MySQL 的帮助非常简单，由于 MySQL 是时下非常流行的一款数据

库软件，因此你可以在互联网上搜索到很多的相关资料和文档。不仅如此，MySQL官方推出的中文手册也是相当不错的文档。MySQL官方手册中文版的浏览地址是：http://dev.mysql.com/doc/refman/5.1/zh/preface.html （如图1.4）

（图1.4）

第2章 安装 mysql

在 Windows 中安装一个 MYSQL 非常简单，双击图标然后看着办就行。在 Linux 当中，我们就有很多种选择了。首先要做的事情是，选择你应该使用的分发版本。简单的说，如果你使用的是 RHEL4 的操作系统，很显然你不应该去下载对应 RHEL5 的 RPM 分发版。如果你不确定你使用的操作系统或者 MYSQL 官方网站上没有提供对应你的操作系统的版本的“二进制预编译版”的话，你可以选择下载源代码包，然后自己编译安装 MYSQL。接下来我们主要讲解通过 RPM 包方式安装和源代码包方式两种方式安装 MYSQL。学会这两种方式之后，相信如果你遇到其他情况也难不倒你了。

第2.1节 RPM 包方式安装 MYSQL

大多数的 Linux 发行版都已经内含了 MYSQL 组件，比如 RedHat Enterprise4 内含的就是 MYSQL4.1。你可以通过 Linux 发行版光盘中附带的 RPM 包进行安装。如果你想获取到最新版本的 MYSQL 的话，当然也可以通过官方网站进行下载。仍然要再次提醒您：选择你应该选择的版本。

笔者当前使用的系统是 RedHat Enterprise4U4 x86，因此我们就以 RedHat Enterprise4 为例。打开 MYSQL 官方网站的下载页面，然后找到对应 RHEL4 的相关链接。（如图 2.1）在大多数情况，你只需要安装 MySQL-server 和 MySQL-client 软件包来安装 MySQL。在标准安装中不需要其它的包。

Red Hat Enterprise Linux 4 RPM (x86) downloads

Server	5.0.51a-0	17.4M	Download | Pick a mirror MD5: e7af4f5a666b1c21171ce6315f5cc136
Client	5.0.51a-0	6.0M	Download | Pick a mirror MD5: 94029f89c1365d738d777157c0229f3b
Shared libraries	5.0.51a-0	1.7M	Download | Pick a mirror MD5: d3bb2bee95d1c2aa325306480a612e21
Shared compatibility libraries (3.23, 4.x, 5.x libs in same package)	5.0.51a-0	3.3M	Download | Pick a mirror MD5: 7142d82da1b4ede5d00243ad99a1de50
Headers and libraries	5.0.51a-0	9.6M	Download | Pick a mirror MD5: 9b952d072a6412878d5ed02eee0af183
Test suite	5.0.51a-0	6.5M	Download | Pick a mirror MD5: 85785d9e968d19c065212b4fd6bbe667
Debug information	5.0.51a-0	54.5M	Download | Pick a mirror MD5: ce20b7088d64bd02176d26ef03fc886f
Cluster storage engine	5.0.51a-0	1.4M	Download | Pick a mirror MD5: 077a0964f394ad94336bda1330679ae7
Cluster storage engine management	5.0.51a-0	1003.9K	Download | Pick a mirror MD5: 33522a8d923891bce561bc12e31e998e
Cluster storage engine basic tools	5.0.51a-0	6.4M	Download | Pick a mirror MD5: a5d7d90f087ee4e921e569f22853984d
Cluster storage engine extra tools	5.0.51a-0	3.3M	Download | Pick a mirror MD5: 754890912c57d14053a0c600bac0393d

（图 2.1）

注释：通常由其它供应商提供 MySQL 的 RPM 分发版。其特征和功能与 MySQL AB 所构建的不同。

如果你使用的是对应 RHEL4 或者 RHEL5 的 RPM 发行版本，那么在安装结束之后你可以像控制其他 SYS V 服务一样控制 MYSQL 服务。对于其他 Linux 发行版本来说，你应该能够在/usr/share/mysql 目录中找到 mysql.server 这个脚本。这是 MYSQL 的启动和停止脚本。可以使用它配合“start”或“stop”参数来帮助我们启动和停止 MYSQL 服务器。

```
[root @server1 mysql ] #mysql.server start
[root @server1 mysql ] #mysql.server stop
```

为了让我们这个 MYSQL 和其他 RHEL5 自带的服务一样可以使用 service 和 chkconfig 等命令控制，我们还可以这样做：

```
[root @server1 mysql ] #cp  mysql.server  /etc/rc.d/init.d/mysqld
#将 mysql.server 脚本拷贝到 inid.d 目录中
[root @server1 mysql ] #chmod  +x  /etc/rc.d/init.d/mysqld
```

```
 #添加执行权限
[root @server1 mysql ] #service  mysqld  restart
#通过 service 控制 MYSQL
[root @server1 mysql ] #chkconfig  —add  mysqld
 #添加 MYSQL 服务
[root @server1 mysql ] #chkonfig  —level 345  mysqld on
#使 MYSQL 在 runleve13、4、5 中自动启动
```

之后不要忘记添加 MYSQL 的设置文件，mysql.server 将会调用这个设置文件进而决定 MYSQL 如何运行。你应该可以在/usr/share/mysql 目录中看到类似下面这样的文件：

my-huge.cnf

my-innodb-heavy-4G..cnf

my-large.cnf

my-medium.cnf

my-small.cnf

这些是 MYSQL 预设的对应不同服务器情况的一些设置文件模板。在这里 我们使用 my-medium.cnf 这个做为模板就好了。

```
[root @server1 mysql ] #cp  my-medium.cnf  /etc/my.cnf
```

到底为止，整个过程完毕。

第2.2节　源代码包方式安装 MYSQL

有的时候由于没有对应我们操作系统版本的 MYSQL 安装包或者我们想使用最新版本的 MYSQL 或者需要按照我们自己的需要进行个性化的定制的时候，类似 RPM 包这种安装形式显然已经不能满足我们的需要了。因此，我们会选择采用源代码包自己进行编译的方式安装 MYSQL。

首先，在官方网站中找到 MYSQL 源代码包的下载位置并下载它（如图 2.2）

Source downloads

Note that in the more recent MySQL 5.0 and 5.1 releases, Windows binaries are built from the same source as the Unix/Linux source TAR.

Compressed GNU TAR archive (tar.gz)	5.0.51a	26.3M	Download | Pick a mirror
			MD5: a83dbdbb91267daf73d2297a9c283dd1 | Signature
Source RPM	5.0.51a-0	26.1M	Download | Pick a mirror
			MD5: 82e7259733402e1ebb97bc096200684c
Generic Source RPM	5.0.51a-0	26.1M	Download | Pick a mirror
			MD5: 7373a78268833942c6238992d761f289
Red Hat Enterprise Linux 4	5.0.51a-0	26.1M	Download | Pick a mirror
			MD5: e4caa204b19129d64c7dcd4f59bd0b08
Red Hat Enterprise Linux 5	5.0.51a-0	26.2M	Download | Pick a mirror
			MD5: bfb45008373b00eeb79183b55ce9af9b
Red Hat Enterprise Linux 3	5.0.51a-0	26.1M	Download | Pick a mirror
			MD5: 1b0865ff4e804f0aa1d919bfc805c901
SuSE Linux Enterprise Server 9	5.0.51a-0	26.1M	Download | Pick a mirror
			MD5: 6c7600ddf33da761c9bd4a733ae462c8
SuSE Linux Enterprise Server 10	5.0.51a-0	26.1M	Download | Pick a mirror
			MD5: 987afe01f007d41d1217ac87d89dd397
Windows Source (zip)	5.0.51a	29.9M	Download | Pick a mirror
			MD5: 819a2dc91f5d4f5ac6e4eae8d3b78d85 | Signature
Windows Source (tar.gz)	5.0.45	20.6M	Download | Pick a mirror
			MD5: 02f97de162777ddf3f194a8344e667a3 | Signature

（图2.2）

在上图中你可以看到有很多源代码包的选择，其中包括GNU TAR包，公共SRPM包，SuSE，RHEL3、RHEL4、RHEL5的SRPM包，等等。这里我们选用GNU TAR包来进行举例。对应SRPM包的安装应该也没有什么难度，请大家自行尝试了。

在不同的硬件环境下，对应不同的需求，其编译过程多少都有些不同。我们没有办法一一举例，以下是一个编译安装MYSQL的标准过程：

```
[root @server1 mysql ] # groupadd mysql
[root @server1 mysql ] # useradd -g mysql mysql
[root @server1 mysql ] # tar -zxvf  mysql-VERSION.tar.gz
[root @server1 mysql ] # cd mysql-VERSION
[root @server1 mysql ] # ./configure --prefix=/usr/local/mysql
[root @server1 mysql ] # make
[root @server1 mysql ] # make install
[root @server1 mysql ] # cp support-files/my-medium.cnf /etc/my.cnf
[root @server1 mysql ] # cd /usr/local/mysql
[root @server1 mysql ] # bin/mysql_install_db --user=mysql
```

```
[root @server1 mysql ] # chown -R root  .
[root @server1 mysql ] # chown -R mysql var
[root @server1 mysql ] # chgrp -R mysql .
[root @server1 mysql ] # bin/mysqld_safe --user=mysql &
```

要查看 configure 支持的选择列表，可以运行命令：

```
[root @server1 mysql ] # ./configure --help
```

另外，在官方的手册当中提到了两个选项，使你可以编译静态的链接程序：

```
[root @server1 mysql ] # ./configure —with-client-ldflags=-all-static  --with-mysqld-
ldflags=-all-static
```

采取这样的方式进行编译的话，理论上可以提高大概 13%左右的性能。

另外，如果你正在使用 Unix 类的操作系统，你就可以使用 UNIX 套接字链接，同时你也可以将套接字放置于默认位置以外的某个地方(通常在目录“/tmp”或“/var/run”)，例如使用像这样的 configure 的命令：

```
[root @server1 mysql ] # ./configure\-- with-unix-socket-
path=/usr/local/mysql/tmp/mysql.sock
```

理论上说，这样大约可以提升 7.5%的性能。

第3章 MYSQL使用初步

有许多不同的MySQL客户端程序可以连接服务器以访问数据库或执行管理任务，比如我们之前安装的mysql-client。也可以使用其它工具，这些程序不与服务器进行通讯但可以执行MySQL相关的操作。

在本章中，让我们先简述其中一部分常用程序然后再详细描述了每个程序，以及如何调用这些程序和它们理解的选项。

一些重要的工具：

mysql

交互式输入SQL语句或从文件以批处理模式执行它们的命令行工具。

mysqladmin

执行管理操作的客户程序，例如创建或删除数据库，重载授权表，将表刷新到硬盘上，以及重新打开日志文件。mysqladmin还可以用来检索版本、进程，以及服务器的状态信息。

mysqlbinlog

从二进制日志读取语句的工具。在二进制日志文件中包含的执行过的语句的日志可用来帮助从崩溃中恢复。

mysqlcheck

检查、修复、分析以及优化表的表维护客户程序。

mysqldump

将MySQL数据库转储到一个文件（例如SQL语句或tab分隔符文本文件）的客户程序。

mysql import

使用LOAD DATA INFILE将文本文件导入相关表的客户程序。

第3.1节 mysql：MYSQL 命令行工具

mysql 是一个简单的“SQL SHELL”。它支持交互式和非交互式使用。当在系统提示符下直接使用 mysql，提示符将变为“mysql>”，这种状态我们称之为交互使用。在使用交互式时，查询结果采用 ASCII 表格式。当采用非交互式(例如，用作过滤器)模式时，结果为 tab 分割符格式。可以使用命令行选项更改输出格式。

第3.1.1节 Mysql 命令简介

使用 mysql 很简单，我们可以在操作系统命令行下直接调用它。例如：

```
[root @server1 mysql ] # mysql  db_name
#进入交互式模式并操作 db_name 数据库
```

或者

```
[root @server1 mysql ] # mysql --user=user_name --password=your_password db_name
```

这条命令的效果和刚才那个是一样的，只是我们使用指定的用户名和密码来访问。

```
[root @server1 mysql ] # mysql —help
```

执行上面这条命令，你可以查看 mysql 这个命令所支持的参数。你会发现参数有很多。没有必要把每一个参数都背下来。只要了解一下，并且在你需要用到的时候能够找到它们就可以了。

第3.1.2节 mysql 命令使用技巧

接下来我们来看看 mysql 命令使用的第一个技巧。你可以将 SQL 语句写入到一个文本文件中（我们称之为 SQL 脚本文件），然后使用输入/输出重定向让 mysql 从该文件读取输入。这样一来就好像你在使用 mysql 依次执行文本文件中的那些 SQL 命令一样：

```
[root @server1 mysql ] # mysql db_name < text_file
```

如果你正在以交互式方式运行 mysql，可以使用 source 或\.命令执行 SQL 脚本文件：

```
[root @server1 mysql ] # source filename
```

```
mysql> \. Filename
```

说到这里，我们需要暂时停下来一会，讨论一个在MYSQL中比较特例的问题。如果mysql客户程序发送查询时断开与服务器的连接，它立即并自动尝试重新连接服务器并再次发送查询。然而，即使mysql重新连接成功，你的第1个连接也已经结束，并且以前的会话对象和设定值被丢失：包括临时表、自动提交模式，以及用户和会话变量。该行为很危险，如下面的例子所示：

```
mysql> SET @a=1;
Query OK, 0 rows affected (0.05 sec)

mysql> INSERT INTO t VALUES(@a);
ERROR 2006: MySQL server has gone away
No connection. Trying to reconnect...
Connection id: 1
Current database: test

Query OK, 1 row affected (1.30 sec)

mysql> SELECT * FROM t;
+------+
| a    |
+------+
| NULL |
+------+
1 row in set (0.05 sec)
```

在这个例子当中，你可以看到，一开始设置了一个变量@a=1，然后连接断开，使得接下来进行insert操作的时候报告了“ERROR 2006: MySQL server has gone away

”的错误，然后mysql开始自动重新连接，连接恢复之后@a用户变量已经随连接丢失，而mysql仍然会重复执行之前的insert操作。最终你会看到，实际上该条语句虽然执行成功，但是在表t中什么都没有加进去。并且，糟糕

的是，如果你不查看一下的话，你根本不会发现发生了这样的问题！如果有必要在连接断开时终止 mysql 并提示错误，你可以用--skip-reconnect 选项启动 mysql 客户程序。

第3.2节 用于管理 MYSQL 服务器的客户端

mysqladmin 是一个执行管理操作的客户程序，它非常的强大。你可以用它来检查服务器的配置和当前的状态，创建并删除数据库等等。

使用 mysqladmin 的语法为：

mysqladmin [options] command [command-options] [command [command-options]] ...

mysqladmin 的应用方法有很多，在这里我们举一些常用的例子。

1、 创建名字为 db_name 的数据库

```
[root @server1 mysql ] # mysqladmin  create  db_name
```

2、 删除名字为 db_name 的数据库

```
[root @server1 mysql ] # mysqladmin  drop  db_name
```

3、 查看当前服务器状态信息

```
[root @server1 mysql ] # mysqladmin  extended-status
```

4、 重载授权表（类似 reload，通常跟随在重新权限操作后）

```
[root @server1 mysql ] # mysqladmin  flush-privileges
```

5、 给当前连接使用的用户设置一个新密码，密码更改为 new-password

```
[root @server1 mysql ] # mysqladmin  -u username —p  password  new-password
```

如果 new-password 包含空格或其它命令解释符的特殊字符，需要用引号将它引起来。例如：

```
[root @server1 mysql ] # mysqladmin  password  "my new password"
```

6、 关闭服务器

```
[root @server1 mysql ] # mysqladmin  shutdown
```

第3.3节 用于处理二进制日志文件的实用工具

二进制日志文件是记录对数据库操作记录的日志文件，采用二进制进行保存。简单的说，这类日志文件中保存的是我们对数据库的所有操作。它在我们恢复数据库的时候给予了我们很大的帮助。由于这一类文件使用的是二进制方式存储，因此不可以直接编辑和使用它们，因此我们要借助mysqlbinlog工具可以帮助我们检查这些文件。关于二进制日志文件的详细内容，我们等到第六章“mysql备份”的时候再做详细讨论。在此，我们简单看一mysqlbinlog的使用语法。

使用mysqlbinlog的语法：

```
mysqlbinlog [options] log-files...
```

例如，要想显示二进制日志binlog.000003的内容，使用下面的命令：

```
[root @server1 mysql ] # mysqlbinlog  binlog.0000003
```

可以将mysqlbinlog的输出传到mysql客户端以执行包含在二进制日志中的语句。如果你有一个旧的备份，该选项在崩溃恢复时也很有用：

```
[root @server1 mysql ] # mysqlbinlog  binlog.000001 | mysql
```

或：

```
[root @server1 mysql ] # mysqlbinlog  binlog.0000003
```

另一个方法是：

```
[root @server1 mysql ] # mysqlbinlog  binlog.000001  >   /tmp/statements.sql
[root @server1 mysql ] # mysqlbinlog  binlog.000002  >>  /tmp/statements.sql
[root @server1 mysql ] # mysql  -e  "source  /tmp/statements.sql"
```

这个方法可以避免当前一个二进制日志文件使用到临时表的时候，由于退出进程时自动删除临时表而造成的后一个二进制日志文件无法继续使用临时表的问题。

第3.4节 mysqlcheck：表维护和维修程序

mysqlcheck客户端可以检查和修复MyISAM表。它还可以优化和分析表。另

一个功能与它类似的程序是 myisamchk。它们之间的主要差别是当 mysqld 服务器在运行时必须使用 mysqlcheck，而 myisamchk 应用于服务器没有运行时。使用 mysqlcheck 的好处是不需要停止服务器就可以检查或修复表。

有三种方式来使用 mysqlcheck：

当你想检查某个数据的所有的表，或者某个数据库内的某张表的时候，你可以使用：

```
[root @server1 mysql ] # mysqlcheck  db_name  [table_name]
```

当你想一次性检查多个数据库的时候，你应该：

```
[root @server1 mysql ] # mysqlcheck  --databases  db_name1 [db_name2  db_name3  ….]
```

如果你想一次性检查所有的数据库的时候，那么这样做：

```
[root @server1 mysql ] # mysqlcheck  --all—database
```

如果在 mysqlcheck 后面什么都不跟，那么默认是检查所有的数据库。

mysqlcheck 默认跟随的操作是“mysqlcheck --check”，就是检查操作。你可以使用 “--repair”参数来进行修复动作。例如修复名字为 db_name 的数据库中所有的表：

```
[root @server1 mysql ] # mysqlcheck  --repair  db_name
```

或者：

```
[root @server1 mysql ] # mysqlcheck  --auto-repair  db_name
```

这句命令的含义是检查 db_name 这个数据库，如果发现有错误则自动修复它。

mysqlcheck 还有一个很有意思的特性，那就是你可以通过改变它的文件名来更改它的默认操作。比如说，将 mysqlcheck 复制为 mysqlrepair，那么当你执行 mysqlrepair 的时候，默认就采取了修复动作。

下面的表列出了一些可用来更改 **mysqlcheck** 默认行为的文件名：

mysqlrepair	默认选项为--repair
mysqlanalyze	默认选项为--analyze

mysqloptimize	默认选项为--optimize

第3.5节 数据库备份程序

mysqldump是一个非常好的数据库备份程序。它制作出来的备份可以跨平台、跨文件系统，甚至是跨版本的进行恢复。为什么会这么神奇呢？其实很简单，如果你打开一个由mysqldump生成的备份文件你会发现，实际上里面的内容其实是用SQL语言把数据库里所保存的数据表述出来，文件中充斥着create、drop、insert等SQL语句。既然大多数的数据库产品都支持SQL语言，那么mysqldump如此神奇也就不奇怪了吧。关于mysqldump的详细内容我们会在数据库备份那一章详细说到。不过这并不妨碍我们现在提前看看mysqldump的语法。

你可以以三种方式来调用mysqldump。

```
mysqldump [options] db_name [tables]
mysqldump [options] ---database DB1 [DB2 DB3...]
mysqldump [options] --all—database
```

是不是和刚才的mysqlcheck差不多？没错，聪明的你一定能看懂这三个用法的区别。[options]指的就是mysqldump所支持的选项。所有的选项你可以通过“--help”来查看。

第3.6节 mysqlimport：数据导入程序

我们刚刚看过了数据库的备份程序，那么当我们需要用到这些备份来恢复我们的数据库的时候该怎么做呢？那就是使用mysqlimport这个程序啦。

Mysqlimport的语法：

```
[root @server1 mysql ] # mysqlimport [options] db_name textfile1 [textfile2 ...]
```

使用举例：

```
[root @server1 mysql ] # mysqlimport  db_name  backup1.sql
```

使用 backup1.sql 备份文件恢复 db_name 数据库

和其他工具一样，mysqlimport 也有很多选项可以使用。你可以使用“--help”参数来查看它们。有时候你可能不得不用到它们，比如：

```
[root @server1 mysql ] # mysqlimport  -uroot  -p  --force  db_name  backup1.sql
```

使用 root 用户连接数据库，并且忽略恢复过程中可能出现的错误强制性的进行导入。

事实上，通过 mysql 的非交互式使用一样可以做到讲备份文件里的内容恢复进数据库。另外，你也可以使用其他各种具备类似功能客户端工具来做到这一点。他们所做的事情其实都是一样的，就是讲备份文件中的 SQL 语句录入到数据库当中。

第4章 SQL语言基础

SQL是英文Structured Query Language的缩写，意思为结构化查询语言。SQL语言的主要功能就是同各种数据库建立联系，进行沟通。按照ANSI（美国国家标准协会）的规定，SQL被作为关系型数据库管理系统的标准语言。SQL语句可以用来执行各种各样的操作，例如更新数据库中的数据，从数据库中提取数据等。目前，绝大多数流行的关系型数据库管理系统，如Oracle, Sybase, Microsoft SQL Server, Access，MySQL等都采用了SQL语言标准。虽然很多数据库都对SQL语句进行了再开发和扩展，但是包括Select, Insert, Update, Delete, Create, 以及Drop在内的标准的SQL命令仍然可以被用来完成几乎所有的数据库操作。

在这一章当中我们就来介绍一下SQL语言的基本知识。同时，还会涉及到一些MYSQL的基本管理，比如建立库、删除库。

第4.1节 数据库表格

在开始学习SQL语句之前，我们首先应该学习一下关系型数据库存储数据的时候所采取的方式。

一个典型的关系型数据库通常由一个或多个被称作表格的对象组成。数据库中的所有数据或信息都被保存在这些数据库表格中。数据库中的每一个表格都具有自己唯一的表格名称，都是由行和列组成，其中每一列包括了该列名称，数据类型，以及列的其它属性等信息，而行则具体包含某一列的记录或数据。下面是一个书店的库存数据库的范例：

ID	BOOKNAME	WRITER	BOOKDATE	PRICE	AMOUNT
1	Live with Linux	Tube	2007-1-25	75.00	50
2	Linux inside	Kevin	2008-2-15	83.00	50
3	L.A.M.P	Tom	2008-2-5	82.50	50
4	My way	Jam	2007-12-3	45.25	130
5	Open your heart	July	2007-3-18	35.00	20
6	Pro C	Todd	2007-8-25	85.00	25
7	Thinking in C	John	2006-6-13	65.00	30

在该表格中，编号、书名、作者、进货日期、价格、数量就是不同的列，而每一行保存的则是具体的数据。

我们接下来的目标就是在本章结束之前，建立一个名字为 bookshop 的库，在这个库中建立一张关于书籍库存名字为 reserve 的表，表的结构就参照上面的这个范例就可以了。

第4.2节　操作 MYSQL

操作 MySQL 数据库的标准界面是连接到 MySQL 服务软件。你可以使用 MySQL 客户端程序去连接你的 MySQL 服务器，你需要输入下面的命令：

```
[root @server1 mysql ] # mysql -h <hostname> -u <username> -p
```

你需要将<hostname>换成你的 MySQL 服务器正在其上运行的计算机的主机名或 IP 地址。如果你在运行服务的同一台计算机上运行客户端程序，你可以不使用-h <hostname>。<username>是你的用户名。 “-p”参数告诉程序提示你输入你的口令，这将在你输入上面的命令后立即显示。如果你什么都不追加，那么默认是使用 root 用户，密码为空去连接数据库的。如果你为 root 设置过密码的话，那么这样连接就会给你报错。正确的方法应该是：

```
[root @server1 mysql ] # mysql —u root —p
```

如果一切正常的话，MySQL客户端程序会连接上MySQL服务器并返回给你一个MySQL的命令行：

```
mysql>
```

事实上，一个MYSQL服务器可以同时维护着多个数据库。所以你的下一步应该是选择一个你所要操作的数据库。那么接下来我们查看一下当前这个MYSQL服务器中维护着哪些数据库。输入下面的命令（不要忘了分号！），然后打回车。

```
mysql> SHOW   DATABASES;
```

MySQL会显示给你服务器上的数据库列表。如果这是一个新安装的服务器（也就是说，这是你在第一章里自己安装的）。这个列表将会是这样：

```
   +----------+
| Database |
+----------+
| mysql    |
| test     |
+----------+
2 rows in set (0.11 sec)
```

MySQL服务器使用第一个被称之为mysql的数据库来管理用户及其口令和权限。现在我们暂时不会关心这个数据库，在以后的章节中，我们会讨论它。第二个叫test是一个空的数据库，正如它的名字一样，这个库仅仅是个模板。你可以删除这个数据库，在我们的教程中不会使用到它（我们会自己建立一些数据库）。一提到“删除”，大家的脑海里可能浮现出“delete”这个字眼，但是在数据库中，“delete”和“drop”是具有完全不同的含义的。这一点必须明确分别开来。删除什么东西在MySQL中被称之为“dropping”，意思为“丢弃”，也就是说我们不需要它了，要丢弃掉它。因此要删除test数据库，其正确的命令应该是：

```
mysql> DROP  DATABASE  test;
```

如果你输入了这个命令，并打了回车，MySQL会删除这个数据库，并返回Query OK。注意，并不会提示你诸如“这是否确定”这样的信息。整个过程非常简单，简单的非常可怕！所以你在MySQL输入这样的命令的时候，最好知道自己在做什么。

接下来我们重新查看一下数据库列表，看看是否成功删除了 test 这个库。在输入命令的时候我们还需要注意到一点：在 MySQL 中的几乎所有的命令都必须以分号“;”结束。如果在一条命令的结尾处没有加上分号，MySQL 会认为你还没有结束输入你的命令，并会让你这下一行中继续录入：

```
mysql> SHOW
-> DATABASES;
```

MySQL 在等待你输入命令中的剩余部分时，提示符会从 mysql>改变为->。如果你在输入一个非常长的 SQL 语句的时候，这个功能非常有用，同时也会让整条 SQL 语句显得非常规整。你应该练习，并且养成良好的断行输入习惯，合理的将你的命令分几行输入。

最后，当你要退出 MySQL 客户端程序时，你只需要输入 quit 或者 exit（这两个命令是完全一样的）。

```
mysql> quit
Bye
```

第4.3节　建立一个数据库

接下来，我们要开始建立我们的书店的数据库了。执行下面的命令，很容易地你就可以建立一个数据库了：

```
mysql> CREATE DATABASE bookshop;
```

现在我们已经有了一个数据库，名字叫做 bookshop。接下来我们要对于这样数据库开始一些操作，因此我们需要告诉 MYSQL，我们要使用这个数据库：

```
mysql> USE bookshop;
```

现在你可以开始使用你的数据库了。在你在其中添加数据表之前，这个数据库将是空的，我们的第一步工作应该是建立一个数据表来保存我们的记录。

第4.4节　建立一个表

我们之前遇到的 SQL 命令都是非常简单的，但是因为数据表是比较灵活的，相应地建立它们的命令就要复杂得多了。建立数据表的基本格式是这样的：

```
mysql> CREATE TABLE <table name> (
-> <column 1 name> <col. 1 type> <col. 1 details>,
-> <column 2 name> <col. 2 type> <col. 2 details>,
```

```
-> ...
-> );
```

按照我们的例子“reserve”表。这个表有六个数据列：ID（一个数字）、bookname（书的名字）、writer（书的作者）、date（出版日期）、cost（价格）和amount（数量）。那么，建立这个表的命令应该是这样的：

```
mysql> CREATE TABLE reserve (
-> id INT NOT NULL  PRIMARY KEY,
-> bookname TEXT,
-> writer TEXT,
-> bookdate DATE NOT NULL,
-> price float,
-> amount INT
-> );
```

看上去很复杂，是吧？让我们将它分解一下：

第一行是比较简单的；它说明我们想要建立一个新的名为reserve的数据表。

第二行说明我们需要一个数据列叫ID，这个列的类型应该是一个整数（INT）。这一行还定义了这个数据列的其他一些信息。首先，由于这一列中的数据是书的编号，每一本书都应该有一个编号，因此，这一行不允许为空（NOT NULL）。然后，由于编号是每一本书的唯一的标识符，并且书的编号是不应该重复的，所以这个数据列中的所有值都应该是不重复的（PRIMARY KEY）。

第三行很简单；这说明我们需要一个数据列叫bookname，这个列的类型应该是一个文本（TEXT），它保存的是书名。

第四行和第三行没有什么不同。它保存书的作者名字。

第五行定义了列名是date，这个列的类型是日期型（DATE）这个列也不能为空（NOT NULL）。

第六行和第七行分别定义了price列和amount列，分别是价格和数量。它们的类型分别是FLOAT和INT。小心！作为定义的最后一列，amount int后面可没有跟“;”。

请注意，我们在输入 SQL 命令时，大小写是完全自由的，但是，如果是在一个 UNIX 类系统下运行的 MySQL 服务的话，MYSQL 所创建出来的数据文件可是区分大小写的。因此我们必须与 MySQL 数据目录下的目录和文件一致，当遇到数据库名和表名时，我们必须区分大小写。否则，MySQL 是完全对大小写不敏感的，只有一种情况例外，在同一命令中多次出现的表名、列名以及其他名字必须在拼写上完全一致。

我们还应该注意到，我们为我们建立的每一列指定了一个指定的类型。 id 是一个整型，bookname 是一个文本型，date 是一个日期型。MySQL 允许我们为每一个列定义一个类型。这不仅仅可以帮助你组织数据，而且你可以利用它对数据进行比较（我们在下面会看到）。要想得到一个关于 MySQL 支持的数据类型的完整的列表，你可以参看 MySQL 用户手册。

总之，如果你正确输入了上面的命令，MySQL 会返回 Query OK 并会为你建立你的第一个数据表。如果你在输入中出了什么错误，MySQL 会告诉你输入的语名有问题，而且会给你一些提示，说明什么地方它不能理解。

如果之前的部分一切顺利，那么输入下面的命令可以查看你之前工作的成果：

```
mysql> SHOW  TABLES;
```

回显应该是这样的：

```
+-----------------------+
| Tables in bookshop    |
+-----------------------+
| reserve               |
+-----------------------+
1 row in set
```

看来一切都好了，使用 desc 命令再来看一下这个 reserve 表的结构：

```
mysql> desc reserve;
+-----------+-----------+------------+-----------+---------+-----------------+
| Field     | Type      | Null       | Key       | Default | Extra           |
+-----------+-----------+------------+-----------+---------+-----------------+
| id        | int(11)   | NO         | PRI       | NULL    | auto_increment  |
| bookname  | text      | YES        |           | NULL    |                 |
| writer    | text      | YES        |           | NULL    |                 |
```

```
| bookdate  | date    | NO    |        | NULL |            |
| price     | float   | YES   |        | NULL |            |
| amount    | int(11) | YES   |        | NULL |            |
+-----------+---------+-----------+-----------+--------+-------------------+
  6 rows in set (0.02 sec)
```

这儿提供了一个数据列的列表。正如我们看到的，这个表中有三个列，在返回的结果表中这被表示为三行。详细资料比较难以看懂，但是你认真看一下，应该还是能了解其大致的含义的。不要为这件事过于担心。我们还会继续学习，我们还会在这个表中添加一些书。

如果你要删除一个表。这和删除一个数据库一样简单得令人恐怖。命令也几乎一样：

```
mysql> DROP  TABLE  <tableName>;
```

第4.5节　向数据库中添加数据

我们的数据库和数据表都已经建立好了，接下来我们应该向这个数据库中录入一些数据了。向数据库中添加数据的命令被称之为 INSERT。这个命令有两种基本格式：

```
mysql> INSERT INTO <table name> SET
-> columnName1 = value1,
-> columnName2 = value2,
-> ...
-> ;
```

以及

```
    mysql> INSERT INTO <table name>
-> (columnName1, columnName2, ...)
-> VALUES (value1, value2, ...);
```

你可以根据自己的习惯，在这两种语法中选择一个你喜欢的使用。下面我选择前一种方法，演示了如何向数据库中插入一条新的图书记录。

```
mysql> insert into reserve SET
    -> id = 1,
    -> bookname = "Live with Linux",
```

```
    -> writer = "Tube",
    -> bookdate = "2007-01-25",
    -> price =75 ,
    -> amount = 50;
Query OK, 1 row affected (0.08 sec)
```

如果你选择第二种方式添加数据，需要注意的是，你给出的列的顺序必须与你给出的数据的顺序一致。

接下来，请大家参照本章一开始的例子，将那个表格里的所有内容都添加到你现在这个数据库中。希望你尝试使用两种不同的 insert 语法，找出你习惯和喜欢的方式。

当你成功完成上面的练习之后，让我们来看看如何显示表的内容。

第4.6节 查询数据库中的数据

查询你的数据库中表的存储数据的命令，被称之为 SELECT，这个命令无疑是 SQL 语言中最复杂的命令。说它复杂，是因为数据库的最主要的优点就是可以根据我们的需要，灵活的有选择性和关系性的展示我们所需要的数据。而这些特性就是通过 select 这个命令实现的。

首先，让我们从最简单的开始。使用下面的命令会列出存储在表 reserve 中的所有数据：

```
mysql> SELECT * FROM reserve;
```

这个命令意味着“从 reserve 中挑选所有的东西并显示出来”。下面是对这条命令的显示：

```
mysql> select * from reserve;
+----+-------------------+---------+----------------+-------+-----------+
| id  | bookname      | writer  | bookdate     | price | amount  |
+----+-------------------+---------+----------------+-------+-----------+
|  1 | Live with Linux | Tube  | 2007-01-25 |   75 |    50   |
|  2 | Tinking in JAVA | TOM  | 2006-03-15 |    75 |    300 |
+----+-------------------+---------+----------------+-------+-----------+
```

Select 命令中的那个“*”，可以理解为是一个通配符，表示的是“所

有的列”。如果你不像看所有列的信息，而只是想看看这个表中的某一部分的信息，比如说，我只想看一下当前库存里有哪些书，以及价格和库存数量多少。那么，这条查询语句应该这么去写：

```
mysql> SELECT bookname,price,amount FROM reserve;
```

另一个有用的函数是COUNT，通过这个函数，我们可以很简单地得到返回结果的个数。例如，如果我们想要找出在我们的表中我们存储了多少条记录，我们可以用下面这个命令：

```
mysql> select count(*) from reserve;
+----------+
| count(*) |
+----------+
|     2    |
+----------+
  1 row in set (0.03 sec)
```

到目前为止，我们的所有的例子都是针对表中的所有记录的。作为对SELECT命令的补充，我们可以使用“WHERE子句”，这样我们可以对返回的结果进行限制。看看下面这个例子：

```
mysql> SELECT COUNT(*) FROM reserve
-> WHERE bookdate >= "2006-01-01";
```

这个查询语句会数出所有日期“大于或等于”2006年1月1日的记录，对于一个日期来说“大于或等于”意味着“在当天或在此之后”。

有一种比较特殊的用法可以找出包含某一段文字的记录。你可以看看这个查询语句：

```
mysql> SELECT bookname FROM reserve
-> WHERE bookname LIKE "%Linux%";
```

这个查询语句显示了所有书名包含Linux的书。LIKE关键字告诉MySQL指定的列必须匹配给定的表达式。在这里，我们使用的表达式是"%Linux%"。这儿的%说明单词Linux可以出现在任何字符串的前面或后面。

条件也可以在 WHERE 子句中组合使用，这样可以做更复杂的查询。例如我们要找出所有 2006 年内出版的书名中带 Linux 的所有的书，我们可以这样做：

```
mysql> SELECT bookname,bookdate FROM reserve
WHERE
-> bookname LIKE "%Linux%" AND
-> bookdate >= "2006-01-01" AND
  -> bookdate < "2007-01-01";
```

我们还可以用 SELECT 语句做很多事，但是我们在这里将不再详细讨论它，我们在需要的时候才会再讲到其他一些高级的功能。如果你太好奇，没法再等下去的话，你可以去看 MySQL 用户手册。

第4.7节　修改表中存储的数据

一旦你已经向数据库的表中输入了一些数据，你可能想要进行一些修改。例如改正拼写错误，以及其他有关的数据，所有的这些改变都可以用 UPDATE 命令来完成。这个命令包含了一些 INSERT 命令（在设置列的数值方面）和 SELECT 命令（在选取改变对象方面）的基本原理。UPDATE 命令的基本格式是这样的：

```
mysql> UPDATE <tableName> SET
-> <col_name>=<new_value>, ...
-> WHERE <where clause>;
```

例如，如果你想要改变上面输入的书的出版日期，你需要输入下面的命令：

```
mysql> UPDATE reserve SET bookate="2007-04-01" WHERE ID=1;
```

这儿我们用到了 ID 列。通过它你可以很方便地指定你要改变的记录。WHERE 子句也可以用在这里，就象在 SELECT 命令中那样。下面的命令是改变所有正文中包含单词 Linux 的书的出版日期：

```
mysql> UPDATE reserve SET bookdate="2007-04-01"

-> WHERE bookname LIKE "%Linux%";
```

第4.8节　删除表中存储的数据

在 SQL 中删除一个内容是令人恐怖的简单，下面是这个命令的格式：

```
mysql> DELETE FROM <tableName> WHERE <where clause>;
```

要删除所有包含Linux的书，你只需要输入下面的命令：

```
mysql> DELETE FROM reserve WHERE bookname LIKE "%Linux%";
```

这里的WHERE子句是可选的，但是如果你不用它，你应该清楚你在干什么，因为这时其实你是在清空这个数据表。下面这个命令将清空数据表：

```
mysql> DELETE FROM reserve;
```

第4.9节 索引

第4.9.1节 什么是索引？

索引用来快速地寻找那些具有特定值的记录，所有MySQL索引都以B-树的形式保存。如果没有索引，执行查询时 MySQL必须从第一个记录开始扫描整个表的所有记录，直至找到符合要求的记录。表里面的记录数量越多，这个操作的代价就越高。如果作为搜索条件的列上已 经创建了索引，MySQL无需扫描任何记录即可迅速得到目标记录所在的位置。如果表有1000个记录，通过索引查找记录至少要比顺序扫描记录快100倍。

假设我们创建了一个名为people的表：

```
mysql> CREATE TABLE people ( peopleid SMALLINT NOT NULL, name CHAR(50) NOT NULL );
```

然后，假设我们完全随机把1000个不同name值插入到people表。

可以看到，在数据文件中name列没有任何明确的次序。如果我们创建了name列的索引，MySQL将在索引中排序name列：

对于索引中的每一项，MySQL在内部为它保存一个数据文件中实际记录所在位置的“指针”。因此，如果我们要查找 name等于“Mike”记录的peopleid（SQL命令为“SELECT peopleid FROM people WHERE name='Mike';”），MySQL能够在name的索引中查找“Mike”值，然后直接转到数据文件中相应的行，准确地返回该行的 peopleid（999）。在这个过程中，MySQL只需处理一个行就可以返回结果。如果没有“name”列的索引，MySQL要扫描数据文件中的所有 记录，即1000个记录！显然，需要MySQL处理的记录数量越少，则它完成任务的速度就越快。

第4.9.2节 索引的类型

MySQL提供多种索引类型供选择：

1、普通索引

这是最基本的索引类型，而且它没有唯一性之类的限制。普通索引可以通过以下几种方式创建：

创建索引，例如CREATE INDEX <索引的名字> ON tablename (列的列表);

例如：

```
mysql> create index index_id on reserv(id);
```

修改表，例如ALTER TABLE tablename ADD INDEX [索引的名字] (列的列表);

创建表的时候指定索引，例如CREATE TABLE tablename ([...], INDEX [索引的名字] (列的列表));

2、唯一性索引

这种索引和前面的“普通索引”基本相同，但有一个区别：索引列的所有值都只能出现一次，即必须唯一。唯一性索引可以用以下几种方式创建：

1）创建索引，例如CREATE UNIQUE INDEX <索引的名字> ON tablename (列的列表);

2）修改表，例如ALTER TABLE tablename ADD UNIQUE [索引的名字] (列的列表);

3）创建表的时候指定索引，例如CREATE TABLE tablename ([...], UNIQUE [索引的名字] (列的列表));

3、主键

主键是一种唯一性索引，但它必须指定为“PRIMARY KEY”。如果你曾经用过AUTO_INCREMENT类型的列，你可能已经熟悉主键之类的概念了。主键一般在创建表的时候指定，例如“CREATE TABLE tablename ([...], PRIMARY KEY (列的列表)); ”。但是，我们也可以通过修改表的方式加入主键，例如“ALTER TABLE tablename ADD PRIMARY KEY (列的列表); ”。每个表只能有一个主键。

4、全文索引

MySQL从3.23.23版开始支持全文索引和全文检索。在MySQL中，全文索引的索引类型为FULLTEXT。 全文索引可以在VARCHAR或者TEXT类型的列上创建。它可以通过CREATE TABLE命令创建，也可以通过ALTER TABLE或CREATE INDEX命令创建。对于大规模的数据集，通过ALTER TABLE（或者CREATE INDEX）命令创建全文索引要比把记录插入带有全文索引的空表更快。本文下面的讨论不再涉及全文索引，要了解更多信息，请参见官方手册。

第5章 MYSQL 用户与权限

在 MYSQL 中，你可以根据自己的需要定制不同的用户。比如，定制来自某些 IP 地址的用户可以访问，并且给这些用户设置密码。搭配权限的设置，你还可以进一步限定用户的权限范围，比如说仅仅让一个用户只能访问某一个库，并且只能对这个数据库拥有 insert 和 select 权限。

总的来说，MYSQL 中用户设置还算是很自由的。接下来，我们就看看如何对用户与权限进行管理和操作。

第5.1节 添加一个 MYSQL 用户

可以用两种方式来添加 MYSQL 用户：

1、 使用 GRANT 语句。

2、 使用 MYSQL 授权表。

事实上，MYSQL 中所有关于用户的信息都保存在名字为 mysql 的这个库中对应的表当中。使用 GRANT 语句添加用户，实际上是把你在语句中设定的内容写入 mysql 库中对应的表里面。因此，直接操作 mysql 库中对应的表也可以添加用户。不过推荐大家使用 GRANT 语句来创建新用户，因为这样更加方便。

首先来看看如何使用 GRANT 语句添加用户：

```
mysql> GRANT ALL PRIVILEGES ON *.* TO 'tube'@'192.168.0.254'
    ->     IDENTIFIED BY '123456' WITH GRANT OPTION;
```

这条语句看上去很复杂，我们来拆分开来，一段一段来解释。

首先是 GRANT 语句，跟在它后面的是权限内容。ALL PRIVILE 意思是“所有的权限”。

ON 语句指定这些权限应用于那些库和表。*.*代表的是所有的库的所有表。

TO 语句确定了这些权限给与哪个用户。在这里我把前面设置的权限赋予给'tube'@'192.168.0.254'这个用户。tube 是用户名，@符号后面确定了只允许该用户通过 192.168.0.254 这个 IP 地址访问。

IDENTIFIED BY 语句是设定用户密码，在这里用户密码设定为 123456.

WITH GRANT OPTION 的意思是允许 tube 这个用户把他拥有的权限再赋予其

他用户。

整条语句运行的结果就是创建了一个名字为tube的用户，限定他只能通过192.168.0.254这个IP地址访问服务器，这个用户拥有的权限为“除了WITH GRANT OPTION意外的全部权限”，然后又追加给他WITH GRANT OPTION权限以允许他可以将他自己拥有的权限赋予其他人。

第5.2节 删除一个用户

DROP USER语句用于删除一个或多个MySQL账户。要使用DROP USER，您必须拥有mysql数据库的全局CREATE USER权限或DELETE权限。

例：

```
mysql> DROP USER tube@192.168.0.254;
```

第5.3节 MYSQL用户权限管理

刚才，我们使用了GRANT语句来创建用户。实际上，GRANT语句本身是具有双重用途的。当GRANT语句中指定的用户不存在时，将创建该用户并赋予权限；当GRANT语句中指定的用户存在时，仅仅赋予该用户指定的权限。

在MYSQL中可以赋予用户很多种权限。通过刚才的例子，大家应该知道：想要赋予一个用户权利，只要在GRANT语句的后面跟上权限就可以了。在下面的表格中列出了MYSQL中可以为用户设定的权限和它们的含义：

权限	意义
ALL [PRIVILEGES]	设置除GRANT OPTION之外的所有简单权限
ALTER	允许使用ALTER TABLE
ALTER ROUTINE	更改或取消已存储的子程序
CREATE	允许使用CREATE TABLE
CREATE ROUTINE	创建已存储的子程序
CREATE TEMPORARY TABLES	允许使用CREATE TEMPORARY TABLE

CREATE USER	允许使用 CREATE USER, DROP USER, RENAME USER 和 REVOKE ALL PRIVILEGES。
CREATE VIEW	允许使用 CREATE VIEW
DELETE	允许使用 DELETE
DROP	允许使用 DROP TABLE
EXECUTE	允许用户运行已存储的子程序
FILE	允许使用 SELECT...INTO OUTFILE 和 LOAD DATA INFILE
INDEX	允许使用 CREATE INDEX 和 DROP INDEX
INSERT	允许使用 INSERT
LOCK TABLES	允许对您拥有 SELECT 权限的表使用 LOCK TABLES
PROCESS	允许使用 SHOW FULL PROCESSLIST
REFERENCES	未被实施
RELOAD	允许使用 FLUSH
REPLICATION CLIENT	允许用户询问从属服务器或主服务器的地址
REPLICATION SLAVE	用于复制型从属服务器（从主服务器中读取二进制日志事件）
SELECT	允许使用 SELECT
SHOW DATABASES	SHOW DATABASES 显示所有数据库
SHOW VIEW	允许使用 SHOW CREATE VIEW
SHUTDOWN	允许使用 **mysqladmin shutdown**
SUPER	允许使用 CHANGE MASTER, KILL, PURGE MASTER LOGS 和 SET GLOBAL 语句
UPDATE	允许使用 UPDATE
USAGE	“无权限”的同义词

GRANT OPTION	允许授予权限

那么如果我想让tube这个用户只可以访问bookshop数据库，并且拥有对这个库所有的操作权利。那么应该这么做：

```
mysql> grant all on bookshop.* to tube@192.168.0.254;
```

如果，我只想让tube这个用户只能查看和添加bookshop数据库里的内容，但是别的什么都干不了，那么应该这么做：

```
mysql> grant select,insert on bookshop.* to tube@192.168.0.254;
```

在赋予权限之后，不要忘记执行一下FLUSH PRIVILEGES命令刷新权限缓存，来让你的设置立即生效。

```
mysql> FLUSH PRIVILEGES;
```

如果你想收回某个用户的权限，你需要用到REVOKE语句。REVOKE语句的语法十分简单，用熟了GRANT语句之后，你会很容易理解REVOKE语句的用法。举例来说，如果你想收回之前赋予tube用户的select和insert权限的话：

```
mysql> revoke select,insert on bookshop.* from tube@192.168.0.254;
```

这样以来，你就收回了这些权限。是不是很简单？

第5.4节 修改用户密码

你可以使用多种方法更改一个用户的密码。我们首先介绍使用GRANT语句在IDENTIFIED By语句后面写入新的密码，例如：

```
mysql> grant usage on bookshop.* to tube@192.168.0.254 identified by
‘654321’;
```

这里我赋予tube了一个usage权限，而这个权限的含义是“无权限”。这样一来，我就仅仅更改了tube用户的密码。

另外一个方法就是使用UPDATE USER语句，使用这条语句必须先切换到mysql库。因为这条语句实际上是更新当前库下的user表内的数据。如果你位于其他库当中，就无法找到保存着MYSQL用户信息的user表，自然也就无法成功更改咯。下面的例子是将tube用户的密码更改为654321：

```
mysql> use mysql;
```

```
Database changed
mysql> update user set password=password('654321') where user="tube";
Query OK, 0 rows affected (0.00 sec)
Rows matched: 1  Changed: 0  Warnings: 0
```

第5.5节 忘记 ROOT 密码怎么办？

如果某些普通用户忘记了自己的密码的话，应对的方法很简单，给他重新设置一个密码就好了。可是如果你不小心忘记了 root 用户的密码，那真的是非常的糟糕。因为只有 root 用户自己才有权利更改 root 用户密码的权利。那么该怎么办呢？

首先你需要停掉 mysql 服务：

```
[root @server1 mysql ] # service mysqld stop
```

直接 kill 掉 mysql 服务进程也可以达到同样的目的：

```
[root @server1 mysql ] # killall -TERM mysqld
```

接下来使用 mysqld_safe 这个命令加上—skip-grant-tables 来重新启动 mysql。特别需要提醒你的是，这样启动 mysql 将允许任何人以 root 用户和空密码访问 mysql 服务器！因此，如果你不能断开网络连接的话，那么接下来的所有操作，你必须尽可能的快的去完成。

```
[root @server1 mysql ] # mysqld_safe   --skip-grant-tables &
[root @server1 mysql ] # mysql
mysql> use msyql;
mysql> update user set password=password( ‘new_pass’ ) where user=” root” ;
mysql> exit
[root @server1 mysql ] # service mysqld restart
```

第6章 MYSQL备份

服务器维护的工作永远都有一个不变的主题，那就是备份。正所谓，手里有粮，心里不慌。从事数据库相关工作一样永远都不要忘记一个重点：备份。因为，有了备份，我们就什么都不怕。出了任何问题（当然，我们善意的希望永远不要出什么问题），只要我们手上有完整的备份，那就有机会恢复回来。

我们可以通过多种方式做MYSQL的备份，但是需要根据不同的情况，不同的需要选择不同的方法。这些备份方法总的来说，分为三种类型：一种是使用操作系统命令进行文件系统层面上的备份。一种是通过外部工具制作备份。本章将会从这两种类型出发，详细介绍M

YSQL备份的方法与技巧。在讲述了基本的备份方法之后，我们将会更进一步学习使用二进制日志进行高级恢复的方法，并且带领大家做一个MYSQL 的AB复制实验。使用LVM快照方式进行备份。以及使用Xtrabackup进行备份和恢复。

第6.1节 通过操作系统命令备份

MySQL 预设的存储目录是/var/lib/mysql。打开这个目录你会发现有很多和你数据库同名的目录。目录里有很多文件，这些文件就保存着你数据库里的那些数据。

所谓的通过操作系统命令备份，就是把这些文件保存到其他地方去。在做这个事情之前，你首先必须停掉 MYSQL。因为，有可能在你备份的同时，有资料在进行写入，而 MYSQL 会 LOCK 正在使用的数据库文件。

```
[root @server1 mysql ] # /etc/rc.d/init.d/mysqld stop
   [root @server1 mysql ] # cd /var/lib/mysql/
[root @server1 mysql ] # tar zxcf mydb_backup.tgz mydb
[root @server1 mysql ] # /etc/rc.d/init.d/mysqld start
```

上面的这个例子首先是停止 MYSQL 的运行。然后进入/var/lib/mysql 目录。之后将目录下 mydb 这个库打包备份出一个压缩文件。之后再重新启动 MYSQL。就这么简单几个步骤，你就完成了 MYSQL 的备份。

如果发生了不幸的事情，你可以将这个备份解包覆盖回去，就算是恢复了。

这种备份方式的好处在于，它十分的简单，总共就这么几个步骤。但是缺点也显而易见。这样做，你必须首先停掉 MYSQL 才可以进行。另外，你制作的备份还受到 MYSQL 版本的限制。比如说如果源库运行在 MYSQL4.1 的环境里，你就不能把这样的备份文件在 MYSQL5.0 的环境里恢复。因为，不同版本的数据库文件是不可以混用的，它们的数据结构是不同的。

第6.2节 使用 mysqldump 进行 备份

正因为使用操作系统命令备份存在这么多的缺点，因此 MYSQL 官方推荐大家使用 mysqldump 这个程序来进行备份。mysqldump 的做法是将数据库中的每个表的结构和每一条记录都生成 SQL 语句，并把这些语句保存在一个文本文档中。使用 mysqldump 可以对整个 MYSQL 所有库进行备份，也可以详细至某一个库或某个库中的某个表。这样的备份文件不受数据库版本的限制。理论上，可以跨版本恢复（某些特殊情况除外），并且不受文件系统的限制。在进行备份的时候也无需停止 MYSQL。眞是好用又方便啊。以下是使用例子：

[root @server1 mysql] #mysqldump —user=root -p mydb > /backup/mydb.sql

以上指令，将mydb库备份到mysql.sql文件中。输入指令后，需要输入mysql的root密码。

恢复同备份一样很简单，只要一条命令就可以完成：

[root @server1 mysql] #mysql —user=root -p mydb < /backup/mydb.sql

如果你的MYSQL运行这不止一个数据库，而你希望一次性把所有的数据库都备份出来，那么你可以使用—all-databases参数，例如：

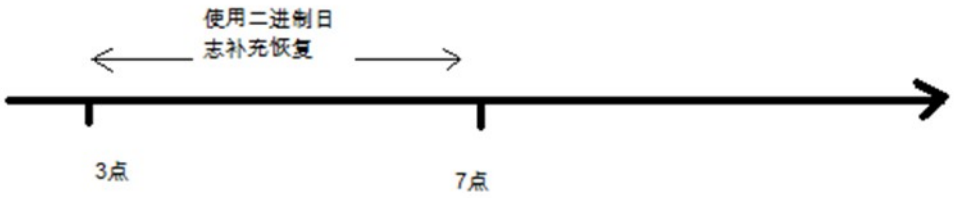
[root @server1 mysql] #mysqldump —user=root -p —all-databases > /backup/mysql.sql

第6.3节 二进制日志

二进制日志实际上是MYSQL日志的一种。二进制日志包含了所有更新了数据或者已经潜在更新了数据（例如，没有匹配任何行的一个DELETE）的所有语句。语句以“事件”的形式保存，它描述数据更改。

为什么把它放到备份这一章当中来说呢？那是因为，由于二进制日志的特殊性，它可以在备份工作上帮我们很大的忙。

前面我们说过的通过操作系统命令制作的备份和通过mysqldump制作的备份都存在着一个无法弥补的缺陷。那就是，这两种备份方法都只能备份“进行备份操作那一刻”的数据。举例来讲，如下图所示：



假如你定为凌晨3点进行日常备份。但是数据库在早上7点发生了崩溃。那么你手头上拥有的最新的数据库备份只能恢复到凌晨3点的状态。而3点~7点之间的数据你就恢复不回来了。在某些应用场景中，遗失一分钟的数据都是无法忍受的，何况是长达4个小时呢。

这个时候我们就需要请出二进制日志来帮忙了。回想一下，二进制日志不是保存的对数据库所有更改的记录么？那么，把日志中所有的记录再重新执行一次，就等于是把那些遗失的数据还原回来了。在刚才的例子中，我们就可以首先通过备份把数据库恢复到凌晨 3 点的状态，然后再通过二进制日志恢复从凌晨 3 点到 7 点之间所有的操作。这样，一个完整的数据库就又回来啦！

默认情况下 MYSQL 是没有打开二进制日志的。你需要在 my.cnf 文件的[mysqld]语句内添加 bin-log 参数，例如：

```
[mysqld]
bin-log
```

然后重新启动 MYSQL，你就会在 MYSQL 数据存储目录中发现很多类似 binlog.000001 这样的文件。这些文件就是二进制日志文件啦。由于这些数据是以二进制形式进行存储的，因此你无法直接查看它。那么怎么办呢？还记得我们之前介绍的 mysqlbinlog 这个程序么？对啦，它就是专门用来操作二进制日志文件的工具。

你可以在 mysqlbinlog 语句中通过—start-date 和--stop-date 选项指定 DATETIME 格式的起止时间。举例说明，假设在今天上午 10:00(今天是 2008-6-18）执行 SQL 语句来删除一个大表。要想恢复表和数据，你可以恢复前晚上的备份，并输入：

```
[root @server1 mysql ] #mysqlbinlog --stop-date="2008-06-18 9:59:59"
/var/log/mysql/bin.123456 | mysql -u root -pmypwd
```

该命令将恢复截止到在—stop-date 选项中以 DATETIME 格式给出的日期和时间的所有数据。如果你没有检测到几个小时后输入的错误的 SQL 语句，可能你想要恢复后面发生的活动。根据这些，你可以用起使日期和时间再次运行 mysqlbinlog：

```
[root @server1 mysql ] #mysqlbinlog --start-date="2008-06-18 10:01:00"
/var/log/mysql/bin.123456 | mysql -u root -pmypwd
```

在该行中，从上午 10:01 登录的 SQL 语句将运行。组合执行前夜的转储文件和 mysqlbinlog 的两行可以将所有数据恢复到上午 10:00 前一秒钟。你应检查日志以确保时间确切。

也可以不指定日期和时间，而使用 mysqlbinlog 的选项—start-position 和--stop-position 来指定日志位置。它们的作用与起止日选项相同，不同的

是给出了从日志起的位置号。使用日志位置是更准确的恢复方法，特别是当由于破坏性SQL语句同时发生许多事务的时候。要想确定位置号，可以运行mysqlbinlog寻找执行了不期望的事务的时间范围，但应将结果重新指向文本文件以便进行检查。操作方法为：

```
[root @server1 mysql ] #mysqlbinlog --start-date="2008-06-18 9:55:00" --stop-date="2008-06-18 10:05:00" /var/log/mysql/bin.123456 > /tmp/mysql_restore.sql
```

该命令将在/tmp目录创建小的文本文件，将显示执行了错误的SQL语句时的SQL语句。你可以用文本编辑器打开该文件，寻找你不要想重复的语句。如果二进制日志中的位置号用于停止和继续恢复操作，应进行注释。用log_pos加一个数字来标记位置。使用位置号恢复了以前的备份文件后，你应从命令行输入下面内容：

```
[root @server1 mysql ] #mysqlbinlog --stop-position="368312" /var/log/mysql/bin.123456 \
| mysql -u root -pmypwd

[root @server1 mysql ] #mysqlbinlog --start-position="368315" /var/log/mysql/bin.123456 \
| mysql -u root -pmypwd
```

上面的第1行将恢复到停止位置为止的所有事务。下一行将恢复从给定的起始位置直到二进制日志结束的所有事务。因为mysqlbinlog的输出包括每个SQL语句记录之前的SET TIMESTAMP语句，恢复的数据和相关MySQL日志将反应事务执行的原时间。

第6.4节 MYSQL的AB复制实验

MYSQL数据库没有增量备份的机制，当数据量太大的时候备份是一个很大的问题。还好MYSQL数据库提供了一种AB复制的机制。我们的从库所保存的数据必须和主库同步，也就是主库和从库的数据是一模一样的。因此，我们要做到：对主库进行任何的操作都可以引发从库进行相同的操作。这就是我们所说的MYSQL的AB复制。

下面的这个实验使用了我们上面刚刚介绍过的二进制日志。其原理是，从库去读取主库的二进制日志文件，并按照主库的二进制文件的记录对从库进行同样的操作，以达到从库与主库内容同步的效果。要想实现双机的热备首先要了解主从数据库服务器的版本的需求。首先，MYSQL的版本都要高于3.2，还有一个基本的原则就是作为从数据库的数据库版本可以高于主服务器数据库的版本，但是不可以低于主服务器的数据库版本。以下是实验文档：

环境介绍：主库 192.168.0.1 从库 192.168.0.2

1、主库创建/etc/my.cnf，修改[mysqld]里边的键值

```
server-id=1
log-bin = logname                    //定制日志文件名称前缀，如果不定制则按照系统默认定义
binlog-do-db=db_name          //记录日志的数据库
binlog-ignore-db=db_name            //不记录日志的数据库 ，不必须
以上的如果有多个数据库用","分割开
然后设置同步数据库的用户帐号
```

2、主库增加用户，用于从库读取主库日志。

```
mysql> grant  replication slave,reload,super  on  *.*  to  slave@192.168.0.2
identified  by  ' 123456'
```

MYSQL4.02 以前的版本由于不支持 replication 权限，要使用下面的语句来实现这个功能。

```
mysql> grant  file  on  *.*  to  slave@192.168.0.2  identified  by  'slavepass';
```

然后刷新授权表缓存。

```
mysql> flush  tables  with  read  lock;
```

3、在从库机器上连接主库进行测试，如果能够连接上表示之前用户名建立是成功的。

```
[root @server1 mysql ] # /opt/mysql/bin/mysql -u slave -p -h 192.168.0.1
```

完成这步之后，主服务器端的配置就完成了。接下来，你需要做的是使用 mysqldump 工具将主服务器的数据库进行一个完整的备份。再把相应的备份导入到从库中。我不推荐使用操作系统命令直接 COPY 数据文件。因为，可能在跨文件系统、跨操作系统和跨 MYSQL 版本的时候，这种备份方式往往会引起一些意外的不必要的麻烦。

```
[root @ server1 mysql] #mysqldump  --all-database  >  dbbackup.sql
```

然后，将这个备份文件复制到从服务器上，再恢复进从服务器。

```
[root @ server2 mysql] #mysql -uroot -p  < dbbackup.sql
```

4、停从库，修改从库/etc/my.cnf，增加选项：

```
[mysqld]
server-id=2
```

```
master-host=192.168.1.1
master-user=slave
master-password=123456
```

5、启动从库，进行主从库数据同步

```
[root @server1 mysql ] # /opt/mysql/share/mysql/mysql start
[root @server1 mysql ] # /opt/mysql/bin/mysql -u root -p
mysql> slave start;
mysql> load data from master;
```

6、进行测试：

主库创建表，

```
mysql>create database sampdb;
mysql>create table new (name char(20),phone char(20));
mysql>insert into new (' abc,' 0532555555' );
```

打开从库，察看：

```
[root @server1 mysql ] # /opt/mysql/bin/mysql -u root -p
mysql>show database;
mysql
sampdb
test
mysql>use sampdb;
mysql>show tables;
new
```

说明主从数据库创建成功。

7、主从数据库相关命令与相关文件：

slave stop; slave start ; 开始停止从数据库。

show slave status\G; 显示从库状态信息

show master status\G;显示主库状态信息

purge master logs to ' binlog.000004' ; 此命令非常小心，删除主数据库没用的二进制日志文件。如果误删除，那么从库就没有办法自动更新了。

change master；从服务器上修改参数使用

另外，如果你当前操作的从库以前曾经与其他服务器建立过主从关系，你

可能会发现即使你在 my.cnf 文件中更改了主服务器的位置，但是 MSQL 仍然在试图连接就旧的主服务器的现象。发生这种问题的时候，我们可以通过清除 master.info 这个缓存文件或者在 mysql 中通过命令来进行设置。方式如下：

a、删除 master.info 方法

这个文件位于数据文件存放目录里。默认是在/var/lib/mysql 中的。你可以直接将其删除，然后重新启动服务器。

b、mysql 命令方法

如果你不方便重新启动服务器的话，那么就只能使用 mysql 命令来帮助你做到。

首先登录到主服务器上，查看当前服务器状态：

```
mysql> show master status\G;
+---------------+----------+--------------+------------------+
| File | Position | Binlog_Do_DB | Binlog_Ignore_DB |
+---------------+----------+--------------+------------------+
| mysql-bin.003 | 73 | test | manual,mysql |
+---------------+----------+--------------+------------------+
```

记录下 File 和 Position 的值。然后登录从服务器，进行如下操作：

```
mysql> slave   stop;

mysql> CHANGE MASTER TO
-> MASTER_HOST='master_host_name',              //主服务器的 IP 地址
-> MASTER_USER='replication_user_name',              //同步数据库的用户
-> MASTER_PASSWORD='replication_password',           //同步数据库的密码
-> MASTER_LOG_FILE='recorded_log_file_name',  //主服务器二进制日志的文件名(前面要求记录的参数)
-> MASTER_LOG_POS=recorded_log_position;             //日志文件的开始位置(前面要求记录的参数)

mysql> slave  start;
```

第6.5节　使用 MYSQLHOTCOPY 进行备份和恢复

首先要说明的是，mysqlhotcopy 这个命令并不适用于所有情况。它仅仅是一个 perl 脚本，由 Tim Bunce 所编写。它使用 LOCK TABLES、FLUSH TABLES 和 cp 或 scp 来快速备份数据库。用单个命令上说，它是备份数据库或单个表的最快的途径，但它只能运行在数据库目录所在的机器上。并且，这个命令只用于备份 MyISAM。同时，由于它实际上是在调用 cp 或 scp 命令在做文件的拷贝，

因此只能运行在Unix、类Unix和NetWare中。不过，使用mysqlhotcopy的好处在于，备份过程中你不必停掉你的mysql服务，并且备份的速度的确挺快。

我们可以非常简单的备份某个数据库。比如，将bookshop数据库备份到/back目录中去。

```
[root @server1 mysql ] # mysqlhotcopy -h 192.168.0.254  -u root  -p uplooking  bookshop
/back
```

可以一次性将多个数据库备份到某个目录里去。比如,将bookshop和newdb两个数据库都备份到/back目录中去。

```
[root @server1 mysql ] # mysqlhotcopy -h 192.168.0.254  -u root  -p uplooking  bookshop
newdb   /back
```

令人兴奋的是，mysqlhotcopy命令也支持正则表达式。比如将bookshop数据库中所有叫做book的表都备份到/back目录中。

```
[root @server1 mysql ] # mysqlhotcopy -h 192.168.0.254  -u root  -p uplooking  bookshop
./book*/    /back
```

另外，你还可以使用—checkpoint这个参数将每次的备份记录到数据库中做备份记录。

```
mysql> create database hotcopy;

mysql> use hotcopy

mysql> create table checkpoint(time_stamp timestamp not null,src varchar(32),dest
varchar(60), msg varchar(255));

[root @server1 mysql ] #  mysqlhotcopy -h 192.168.0.254  -u root  -p uplooking
bookshop  /back --checkpoint hotcopy.checkpoint


mysql> use hotcopy

Database changed

mysql> select * from checkpoint;

+---------------------+------+------------+-----------+


| time_stamp | src | dest | msg |

+---------------------+----------+------------+-----------+
```

```
| 2009-05-11 04:12:50 | bookshop | /back/bookshop | Succeeded |
| 2009-05-11 04:23:27 | bookshop | /back/bookshop | Succeeded |
| 2009-05-11 04:27:03 | bookshop | /back/bookshop | Succeeded |
+---------------------+-----------+-------------+-----------+
```

第6.6节 使用 LVM 快照方式进行备份

事实上，MySQL 数据库的备份是一个让管理员一直很头疼的问题。各种工作虽然不少，但是各有优劣，想找到一个比较完美的方法却非常困难。Mysqldump 作为数据的逻辑备份工具，弱点在于无法进行在线热备，同时在数据库比较大的时候，备份和恢复的时间是在长得让人无法接受。Mysqlhotcopy 虽然克服了普通系统命令备份必须关闭 mysql 服务的尴尬，但是这东西只能用于备份使用 MYISAM 存储引擎的数据表，并且只能在类 UNIX 环境中使用。如果使用 mysql replication 的话，倒是可以解决热备问题。但是你需要承担增加一台服务器的成本。同时，如果数据被无意或恶意的篡改、删除，那么 slave 服务器上的数据同样不能幸免。

相对于以上方法，在中、大规模的 MySQL 应用环境中，我推荐使用 LVM 快照的方式来制作备份。为什么这种方式比较好呢？

原因如下：

1、 在大多数情况下，这种方式几乎算得上是热备。它无需关闭服务，只需要设置只读或者类似这样的限制。

2、 支持所有基于本地磁盘的存储引擎，比如 MYISAM、InnoDB 和 BDB，还支持 Solid、PrimeXT 和 Faction。

3、 备份速度最快，因为你只需要拷贝相关的二进制数据文件即可。

4、 由于只是简单的拷贝文件，因此对服务器开销非常低。

5、 保存方式多种多样，你可以备份到磁带上、FTP 服务器上、NFS 服务器上或者其他什么网络服务器，以及使用各种网络备份软件来备份。做到这些很简单，说到底就是拷贝文件而已。

6、 恢复速度很快。恢复所需要的时间等于你把数据拷贝回来的时间。

你可以想出更多的方法让这个时间变得更短。

7、 无需使用昂贵的商业软件。

当然，每个事物都有其两面性，它也存在一些缺点：

1、 很明显，你的系统需要支持快照。

2、 在一些公司里，系统管理员和DBA属于不同的团队。而使用快照需要系统root权限。因此，你可能需要做一些团队协调工作或者干脆在DBA Team里安插一个系统管理员。这种事在某些公司很容易，但也可能很麻烦。

3、 无法确切的预计服务停止时间。因为，这种方法到底什么时候算热备什么时候不算，完全取决于FLUSH TABLE WITH READ LOCK命令执行时间的长短。因此，我还是建议你在凌晨干这件事情。或者干脆定下一个维护时间段（比如某些网络游戏运营商的做法）。

4、 如果你把日志放在独立的设备上或者你的数据分布在多个卷上，那么就比较麻烦了。因为这样一来你就无法得到全部数据的一致性快照，这就是所谓的多卷上的数据问题。不过，有些系统可能自动做到多卷快照。

现在，我们来看看如果使用LVM的快照功能来制作MySQL备份。

当然，首先我们准备好相应的环境。配置一个LVM，并且划分合适大小的LV，并且将其挂载到MySQL的数据文件目录上。在这个例子中，你可以看到我已经建立好了一个名叫tube的数据库，并且里面包含一个叫做testdb的表，并且已经写入了一些数据。

```
[root@server1 mysql1]# df -h
Filesystem            Size  Used Avail Use% Mounted on
/dev/sda2             9.5G  3.6G  5.5G  40% /
/dev/sda1              99M   10M   84M  11% /boot
tmpfs                 506M     0  506M   0% /dev/shm
/dev/sda6              99M  5.6M   89M   6% /home
```

```
/dev/sda3             4.8G  238M  4.3G   6% /var
/dev/mapper/testvg-mysql1
                      194M   27M  158M  15% /var/lib/mysql
[root@server1 mysql]# mysql
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 4 to server version: 5.0.22-log

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql> use tube;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> select * from testdb;
+------+-------+
| id   | name  |
+------+-------+
|    1 | kevin |
|    2 | tube  |
+------+-------+
2 rows in set (0.00 sec)
```

接下来，连接到 MySQL 上，运行 FLUSH TABLES WITH READ LOCK。需要注意的是，如果你正在执行一个耗时比较长的查询，那么这条命令可能需要很长时间才能完成。因为，在这个时候，FLUSH TALBES WITH READ LOCK 需要等待在这一刻所有正在执行的查询执行完毕，甚至是对数据毫无改变的 select。所以，如果你正在执行一个较长时间的查询，那么要小心。推荐你在凌晨干这件事情，或者干脆定下一个维护的时间段（类似很多网络游戏公司那样）。如果你所有的表都只是用 innodb 存储引擎，并且不需要同步二进制日志的话，那么这一步可以省略。

```
mysql> flush tables with read lock;
Query OK, 0 rows affected (0.00 sec)
```

然后，我们开始创建快照。在下面的例子中，我们创建了一个名为dbbackup的快照，大小为100M。切记，一定要预估可能发生改变的数据量，并分配足够大小的undo space用于保存可能发生变化的部分。如果分配的空间不够大，快照就会无效，备份也就失败了。

```
[root@server1 mysql1]# lvcreate -L100M -s -n dbbackup /dev/testvg/mysql1
  Logical volume "dbbackup" created
[root@server1 mysql1]#
```

现在，快照已经做好，我们可以释放表锁了。不过在这之前，需要记录一下二进制日志的位置。运行SHOW MASTER STATUS就可以看到。这个值会在MySQL slave机上创建快照的时候用到（如果你有的话）。

```
mysql> show master status;
+------------------+----------+--------------+------------------+
| File             | Position | Binlog_Do_DB | Binlog_Ignore_DB |
+------------------+----------+--------------+------------------+
| mysql-bin.000002 |       98 |              |                  |
+------------------+----------+--------------+------------------+
1 row in set (0.00 sec)

mysql> unlock tables;
Query OK, 0 rows affected (0.00 sec)
```

现在，我们可以开始拷贝备份数据了。通常情况下，你可以忽略备份慢查询日志和错误日志，甚至是大部分的二进制日志。但是，假如你的slave可能由于延迟、处理能力不足或者其他各种原因没有同步到master的最新状态，甚至远远落后于master的话，就必须保存所需的二进制日志了。不管怎么说，我还是建议你采用一个合适的机制来备份你的二进制日志。因为，二进制日志非常重要，并且有的时候，它可以作为我们的增量备份。

```
[root@server1 mysql1]# mount /dev/testvg/dbbackup /mnt
[root@server1 mysql1]# mkdir /var/back
[root@server1 mysql1]# cp -R /mnt/* /var/back/
[root@server1 mysql1]# ls /var/back/
```

```
ibdata1      ib_logfile1  mysql-bin.000001  mysql-bin.index  test
ib_logfile0  mysql        mysql-bin.000002  mysql.sock       tube
[root@server1 mysql]#
```

备份完成后，卸掉挂载，并且删除快照。

```
[root@server1 mysql]# umount /mnt
[root@server1 mysql]# lvremove -f /dev/testvg/dbbackup
  Logical volume "dbbackup" successfully removed
```

到此，整个备份过程就算结束了。如果你想要将内容同步到 slave 机上的话，那么还需要多加几个步骤。

1、 把备份内容拷贝到 slave 的数据文件目录下。

2、 重启 MySQL 服务器，等待恢复完成。

3、 使用 CHANGE MASTER TO 命令告诉 slave 新的二进制日志位置，并从那里开始同步，（就是我们刚刚记录下来的那个）例如：

```
mysql> CHANGE master TO master_host="master", master_user="user",
master_password="password", master_log_file="host-bin.000002", master_log_pos=98;
```

大多数情况下，我们都希望能够自动化完成相应工作以减轻我们的负担。例如，上面的使用 LVM 快照方式备份步骤就比较多。我们都希望能够使用脚本来自动化的完成所有的步骤。不过，好的消息是我们无需自己去写了。Lenz Grimmer 开发了一个叫做 mylvmbackup 的工具，可以让我们快速完成以上工作。你可以在 http://www.lenzg.net/mylvmbackup/下载到这个工具。或者到 http://software.opensuse.org/search?baseproject=ALL&p=1&q=mylvmbackup 寻找更合适你的版本，发布方式有源代码包和 RPM 包两种。它的使用非常简单。下面是演示：

```
[root @server1 mysql ] # mysqlhotcopy -h 192.168.0.254  -u root  -p uplooking  bookshop
/back
```

```
[root@server1 ~]# wget http://www.lenzg.net/mylvmbackup/mylvmbackup-0.13.tar.gz
```

```
[root@server1 ~]# wget ftp://ftp.pbone.net/mirror/atrpms.net/el5-i386/atrpms/stable/perl-
TimeDate-1.16-3_2.0.el5.noarch.rpm
```

```
[root@server1 ~]# wget
ftp://ftp.pbone.net/mirror/ftp.pramberger.at/systems/linux/contrib/rhel5/i386/perl-Config-
IniFiles-2.51-1.el5.pp.noarch.rpm
```

```
[root@server1 ~]# rpm -ivh perl-Config-IniFiles-2.51-1.el5.pp.noarch.rpm
```

```
[root@server1 ~]# rpm -ivh perl-TimeDate-1.16-3_2.0.el5.noarch.rpm

[root@server1 ~]# rpm -ivh mylvmbackup-0.13-1.1.noarch.rpm

[root@server1 ~]# ln -s /usr/bin/mylvmbackup  /usr/local/bin/mylvmbackup

[root@server1 ~]# mylvmbackup --user=root   --mycnf=/etc/my.cnf --vgname=testvg
--lvname=mysql --backuptype=tar --lvsize=100M --backupdir=/root
```

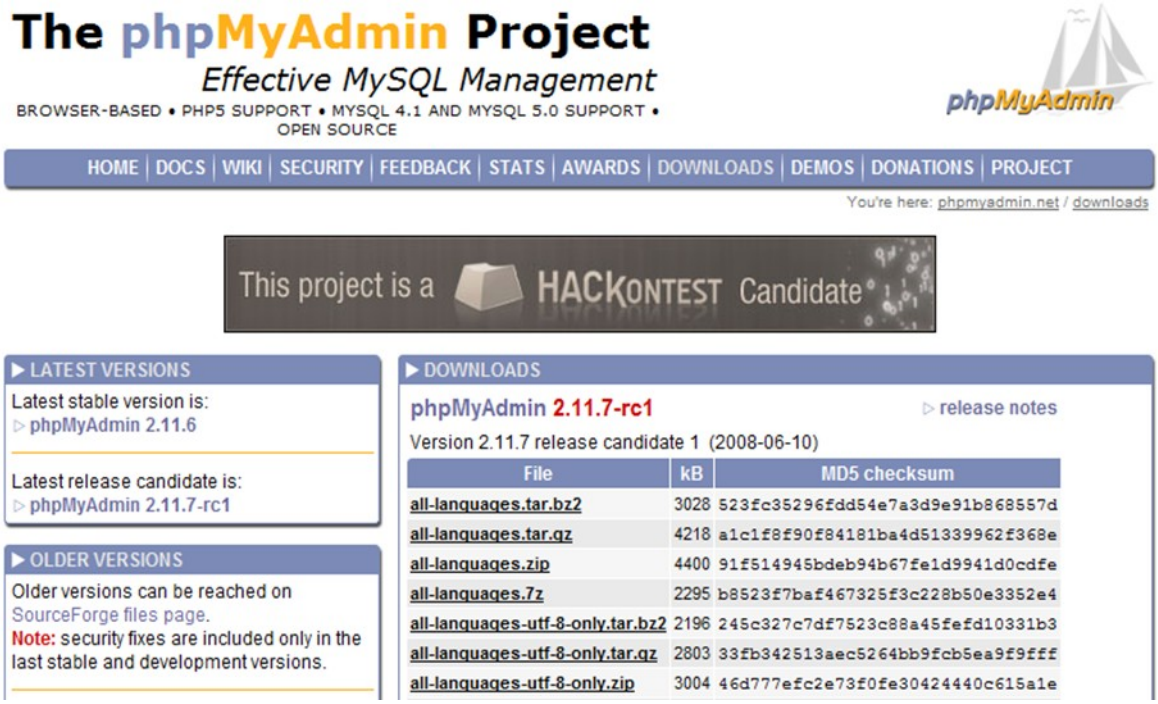
第7章 使用 PHPMYADMIN 管理 MYSQL

PHPMYADMIN 是一个使用 PHP 语言编写的，使用 web 管理 MYSQL 的组件。严格意义上说，它也是一种 MYSQL 的客户端。最近一段时间，出现的很多依靠网站连接 MYSQL 进行管理的产品，在这些“WEB GUI”中，PHPMYADMIN 是使用范围最为广泛的，同时也受到很多 MYSQL 数据库管理员的好评。通过它，你可以非常轻松，非常方便的管理 MYSQL 数据库。

第7.1节 获取 PHPMYADMIN

你可以到 PHPMYADMIN 的官方网站 http://www.phpmyadmin.net 下载最新的版本。

不要使用 quick downloads 里的快捷下载，点击标题栏里的 DOWNLOADS，进入下载页面，你会有更多的选择。我个人推荐下载 all-languages.tar.bz2 的包，毕竟你不知道自己什么时候会需要到其他语言方便的支持。如下图所示：



下载好相应的压缩包后，我们就可以开始安装 PHPMYADMIN 了。

第7.2节 安装 PHPMYADMIN

安装 PHPMYADMIN 的过程是非常简单的，只是后期的设置根据需要的不同，会有不同的调节方法。本书假设你是维护一个“私有的数据库”的管理员（相对于 WEB 空间提供商提供的公用数据库而言），进行的简单设置。

由于 PHPMYADMIN 是一个使用 PHP 语言编写的管理 MYSQL 数据库的产品。因此，如果你想使用它也就需要一个完善的 L.A.M.P 服务器环境。假设你已经安装好了一个 MYSQL，并且正确配置了你的 APACHE 服务器以及 PHP 环境，并且已经安装好了 php-mysql 这个包或者说已经在 APACHE 里配置了 php-mysql 模块。接下来你需要将下载下来的 PHPMYADMIN 解压到你的网站主目录下。默认是在/var/www/html 目录中。我们将原始目录改名为 pma，方便浏览：

```
[root @server1 mysql ] # cd /var/www/html
[root @server1 mysql ] # mv phpMyAdmin-2.11.0-all-languages pma
[root @server1 mysql ] # cd pma
```

然后，你会在目录中看到有一个 config.sample.inc.php 这样的文件，通

过名字就知道，这是 PHPMYADMIN 的配置文件的模板。我们需要把这个文件改个名字来使用。

```
[root @server1 mysql ] # mv config.sample.inc.php  config.inc.php
```

之后，我们需要编辑一下这个配置文件满足我们的使用需要

```
[root @server1 mysql ] # vi config,inc.php
```

我们现在采用 cookie 验证的方式登录 phpmyadmin。我个人不推荐将数据库的帐号与密码填入配置文件中。使用 cookie 方式，总让我更加有安全感一些。

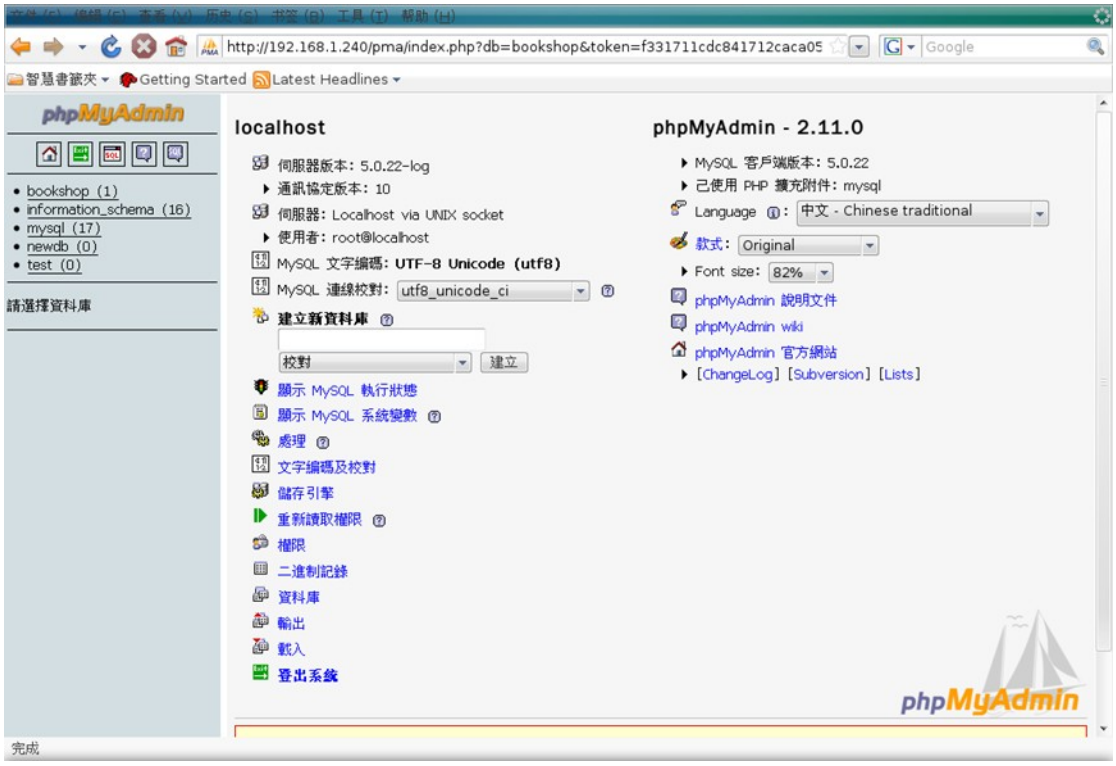
在配置文件找到这一行：

```
$cfg['blowfish_secret'] = ''; /* YOU MUST FILL IN THIS FOR COOKIE AUTH! */
```

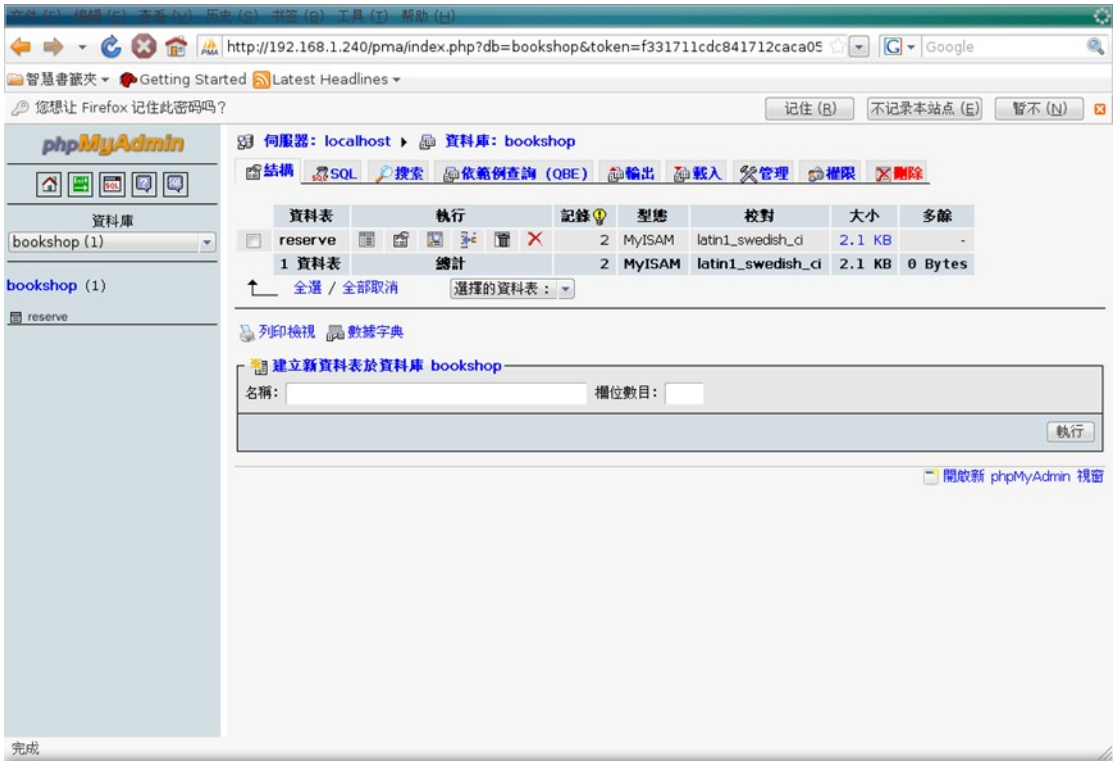
在中间加上你想要设置的 cookie 字符串，比如改成这样：

```
$cfg['blowfish_secret'] = 'uplooking.com'; /* YOU MUST FILL IN THIS FOR COOKIE AUTH! */
```

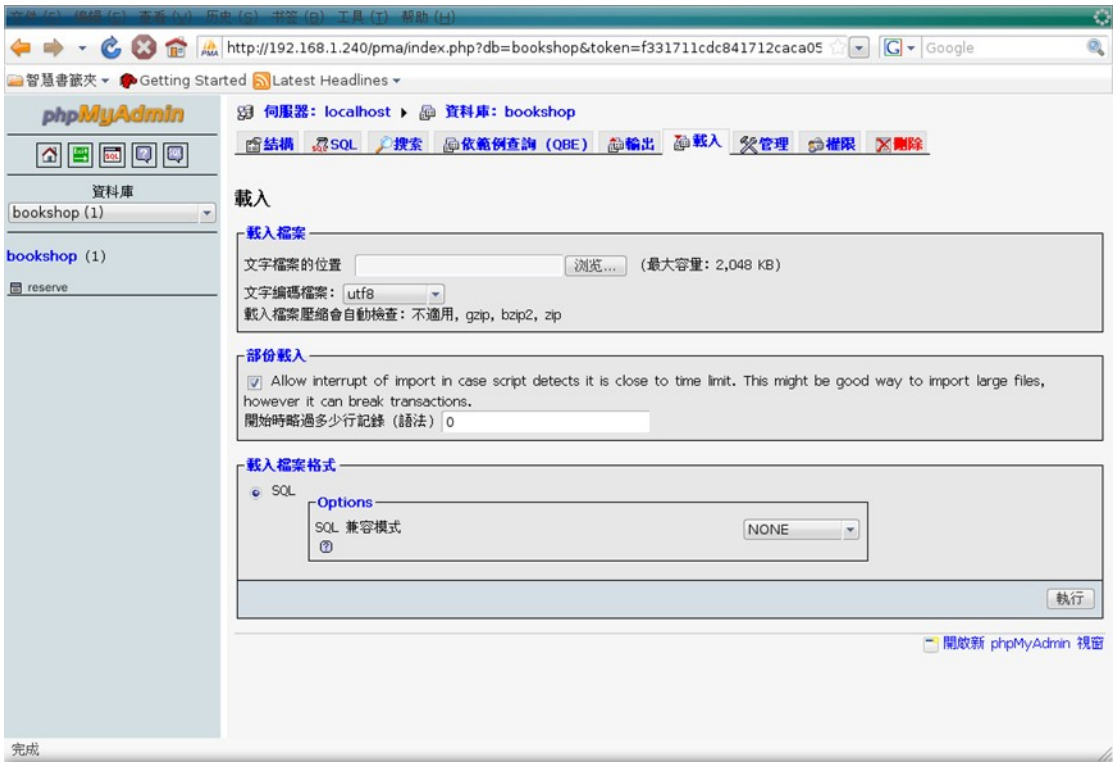
保存退出后，你就可以打开浏览器中输入 http://127.0.0.1/pma 查看 phpmyadmin 了。登录之后如果能看到类似下面这样的页面，则说明你安装是正确的。



当前的页面分为两个部分。左边是数据库选择列表，右边是服务器状态信息。在左边的列表中点选你需要操作的数据库，比如说我们之前建立的 bookshop 这个数据库，就会进入到下面的页面：



正如你所看到的，我们的库中现在只有 reserve 这个表。旁边的几个按钮可以很帮边的对表进行新增、插入、删除或者是查看结构以及浏览等等功能。页面上方的输出与载入。点击输出，则是备份当前数据库。点击输入，就是恢复或者说导入啦。如下图：



看，是不是非常的简单和方便？使用 PHPMYADMIN 可以让很多工作点几下鼠标就可以完成。不过一般来说，它还不能帮你完成所有的工作。作为一个合格的数据库管理员，仅仅依靠“WEB-GUI”类工具是远远不够的。所以，千万不要想着走捷径哦。

至此 PHPMYADMIN 的安装部分就已经完成了。当然，PHPMYADMIN 还有很多其他的功能，在这里就不详细阐述了。

第8章 MYSQL优化

简单安装一个未经调试与设置的MYSQL是不会工作在最佳状态下的。有时候我们需要对当前的MYSQL进行调试和优化来让它达到最佳的性能状态，为我们提供更高的效率，提高检索查询的速度，加强工作的稳定性等等。优化MYSQL分为硬件方面的优化和软件设置方面的优化。前者直接且效果显著，后者需要使用者具有一定的经验并且根据实际情况进行调节。本章分别对这两种途径的优化方法进行介绍。

第8.1节 MYSQL服务器硬件方面的优化

首先，所有的数据都是存储在物理磁盘上的。因此，磁盘性能是制约MYSQL性能的最大因素之一。物理磁盘由于受到磁盘转速的限制，其极限速度存在无法逾越的瓶颈。假设是一个日均PV在10万以上的中型网站，其MYSQL数据库的I／O操作将会非常繁忙的。在这种情况下，为了进一步增强物理磁盘的I／O性能，应该考虑使用磁盘阵列。而在磁盘阵列的选型上，相比较RAID5而言，我更加推荐选择RAID0＋1。

其次，内存大小对于MYSQL的I／O性能影响也很重要。一台MYSQL数据库服务器的内存最好不要小于2G，推荐最好在4G以上。同时，也不要这台服务器运行什么其他服务。

另外，在CPU的选择上，推荐使用S.M.P.架构的多路对称CPU，例如：可以使用两颗Intel Xeon系列的CPU。

如果，你的MYSQL真的非常非常非常的繁忙，单独一台服务器的提升成本／性能／价格比已经不那么划算的时候，还是使用MYSQL簇搭配服务器集群吧。关于MYSQL簇的相关内容会在下一章当中阐述。

第8.2节 MYSQL自身优化

当解决MYSQL服务器的硬件制约之后，我们来看看MYSQL自身的优化。对

MySQL 自身的优化主要是对其配置文件 my.cnf 中的各项参数进行优化调整。而这些参数数值与服务器硬件以及 MYSQL 运行状态信息息息相关。因此，不同的服务器需要根据实际情况进行相应的调整。在/usr/share/doc/mysql-server-VERSION 目录中保存着 my-huge.cnf my-innodb-heavy-4G.cnf my-large.cnf my-medium.cnf my-small.cnf 这样一些文件。这些文件是根据不同的服务器硬件情况的一些配置模板。我们采用 my-huge.cnf 来作范例。

首先，在 mysql 中输入 show status;查看输出的信息。对照这些信息以及服务器硬件配置，我们来一步步更改 my.cnf 里的配置。

skip-locking

这个选项是取消文件系统的外部锁，避免 MySQL 的外部锁定，减少出错几率增强稳定性。

skip-name-resolve

不进行域名反解析,注意由此带来的权限/授权问题使用这一选项可以消除 MySQL 进行 DNS 解析的时间。但需要注意，如果开启该选项，则所有远程主机连接授权都要使用 IP 地址方式，否则 MySQL 将无法正常处理连接请求！

key_buffer_size = 512M

key_buffer 被称为索引缓冲区。这个值控制缓冲区的大小。该值设定需要参照 SHOW STATUS 的输出，如下图所示:

```
mysql> show status like '%key_read%';
+-------------------+-----------+
| Variable_name     | Value     |
+-------------------+-----------+
| Key_read_requests | 163554268 |
| Key_reads         | 98247     |
+-------------------+-----------+
2 rows in set (0.00 sec)
```

Key_reads 代表对数据库的查询操作命中磁盘 buffer 的个数，Key_read_requests 是查询操作的总数。命中磁盘的数除以总数就是不中比率

—— 在本例中每 1,000 个请求，大约有 0.6 个没有命中内存。如果每 1,000 个请求中命中磁盘的数目超过 1 个，就应该考虑增大关键字缓冲区了。

调整这个值需要根据内存大小而定,如果是独立的db服务器,可以设置高达80%的内存总量指定用于索引的缓冲区大小，增加它可得到更好的索引处理性能。一般情况下对于内存在4GB左右的服务器该参数可设置为256M或384M。注意：该参数值设置的过大反而会是服务器整体效率降低！

back_log = 200

连接排队列表总数。指定MySQL连接队列数量。我们都知道在客户端连接MYSQL服务端的时候，服务端提供相应进程处理请求。当进程都在忙碌的时候，建立了TCP连接的请求暂时放入队列中等待处理。不同的操作系统在这个队列大小上有它自己的限制。 试图设定back_log高于你的操作系统的限制将是无效的。默认值为50。对于Linux系统推荐设置为小于512的整数。

table_cache = 512K

打开表缓存总数。如果你经常对一些表进行操作的话，那么每次操作都要重新读入表就显得不那么划算了。因此，我们将读过的表放入到table_cache当中减少对磁盘的读取次数。合理设定该值可以避免频繁的打开数据表产生的开销。

sort_buffer_size = 6M

当 MySQL 必须要进行排序时，就会在从磁盘上读取数据时分配一个排序缓冲区来存放这些数据行。如果要排序的数据太大，那么数据就必须保存到磁盘上的临时文件中，并再次进行排序。如下图所示： sort_merge_passes 状态的变化指示了磁盘的活动情况。

```
mysql> SHOW STATUS LIKE "sort%";
+-------------------+---------+
| Variable_name     | Value   |
+-------------------+---------+
```

```
| Sort_merge_passes | 1       |
| Sort_range        | 79192   |
| Sort_rows         | 2066532 |
| Sort_scan         | 44006   |
+-------------------+---------+
4 rows in set (0.00 sec)
```

如果 sort_merge_passes 很大，就表示需要注意 sort_buffer_size。例如， sort_buffer_size = 6M 将排序缓冲区设置为 6MB。在设置这些数字时要十分谨慎，因为这些设置针对于每个会话。该值乘以连接数才是最终的消耗！

read_buffer_size = 4M

MySQL 也会分配一些内存来读取表。理想情况下，查询应该先去找相应的索引，索引提供了足够多的信息，可以只读入所需要的行。但是有时候由于设计不佳或者数据本性使然，导致索引无法提供什么帮助，结果查询操作需要从表中读取大量的数据。观察这个问题需要知道运行了多少个 SELECT 语句，以及需要读取表中的下一行数据的次数（而不是通过索引直接访问）。通过下图，我们可以确定表扫描比率：

```
mysql> SHOW STATUS LIKE "com_select";
+---------------+--------+
| Variable_name | Value  |
+---------------+--------+
| Com_select    | 318243 |
+---------------+--------+
1 row in set (0.00 sec)

mysql> SHOW STATUS LIKE "handler_read_rnd_next";
+-----------------------+-----------+
| Variable_name         | Value     |
+-----------------------+-----------+
| Handler_read_rnd_next | 165959471 |
```

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

Handler_read_rnd_next / Com_select 得出了表扫描比率 —— 在本例中是 521:1。如果该值超过 4000，就应该查看 read_buffer_size，把这个值设的大一些。例如 read_buffer_size = 4M。如果这个数字超过了 8M 仍然不见效果，就应该与开发人员讨论一下对这些查询进行调优了。

join_buffer_size = 8M 跨表查询操作所能使用的缓冲区大小。

以上两个值和 sort_buffer_size 一样，该参数对应的分配内存也是对应每个连接的。

myisam_sort_buffer_size = 64M MyISAM 表发生变化时重新排序所需的缓冲。

tmp_table_size=

max_heap_table_size=

```
mysql> SHOW STATUS LIKE 'created_tmp%';
+-------------------------+-------+
| Variable_name           | Value |
+-------------------------+-------+
| Created_tmp_disk_tables | 30660 |
| Created_tmp_files       | 2     |
| Created_tmp_tables      | 32912 |
+-------------------------+-------+
3 rows in set (0.00 sec)
```

每次使用临时表都会增大 Created_tmp_tables；基于磁盘的表也会增大 Created_tmp_disk_tables。对于这个比率，并没有什么严格的规则，因为这依赖于所涉及的查询。长时间观察 Created_tmp_disk_tables 会显示所创建的磁盘表的比率，您可以确定设置的效率。 tmp_table_size 和 max_heap_table_size 都可以控制临时表的最大大小，因此请确保在 my.cnf 中对这两个值都进行了设置。

thread_cache = 128

每次 mysqld 需要创建一个新线程时，这个值都会增加。观察下图，此处重要的值是 Threads_created。如果这个数字在连续执行 SHOW STATUS 命令时快速增加，就应该尝试增大线程缓存。

```
mysql> SHOW STATUS LIKE 'threads%';
+-------------------+--------+
| Variable_name     | Value  |
+-------------------+--------+
| Threads_cached    | 27     |
| Threads_connected | 15     |
| Threads_created   | 838610 |
| Threads_running   | 3      |
+-------------------+--------+
4 rows in set (0.00 sec)
```

query_cache_size = 128M

我们肯定不希望每次想要查询什么的时候都需要重新对表进行一次读取。正确的处理方法应该是将查询缓存到 cache 里。但是，有了 cache 也不是说就是万事大吉了。我们还需要知道这个值设定的是否合适。通过观察缓存到 cache 中的这些查询结果到底是否被再次使用了，也就是命中率如何来得到答案。用以下命令观察，如果 Qcache_lowmem_prunes 的值非常大，则表明经常出现缓冲不够的情况：

```
mysql> SHOW VARIABLES LIKE '%query_cache%';
mysql> SHOW STATUS LIKE 'Qcache%';
```

Qcache_free_blocks	缓存中相邻内存块的个数。数目大说明可能有碎片。FLUSH QUERY CACHE 会对缓存中的碎片进行整理，从而得到一个空闲块。
Qcache_free_memory	缓存中的空闲内存。

Qcache_hits	每次查询在缓存中命中时就增大。
Qcache_inserts	每次插入一个查询时就增大。命中次数除以插入次数就是不中比率；用 1 减去这个值就是命中率。
Qcache_lowmem_prunes	缓存出现内存不足并且必须要进行清理以便为更多查询提供空间的次数。这个数字最好长时间来看；如果这个数字在不断增长，就表示可能碎片非常严重，或者内存很少。（上面的 free_blocks 和 free_memory 可以告诉您属于哪种情况）。
Qcache_not_cached	不适合进行缓存的查询的数量，通常是由于这些查询不是 SELECT 语句。
Qcache_queries_in_cache	当前缓存的查询（和响应）的数量。
Qcache_total_blocks	缓存中块的数量。

set-variable = wait_timeout=60 设置超时时间,能避免长连接。

max_connect_errors = 100

如果一个主机在连接到服务器时有问题，并重试很多次后放弃，那么这个主机就会被锁定，直到 FLUSH HOSTS 之后才能运行。默认情况下，10 次失败就足以导致锁定了。将这个值修改为 100 会给服务器足够的时间来从问题中恢复。如果重试 100 次都无法建立连接，那么使用再高的值也不会有太多帮助，可能它根本就无法连接。

wait_timeout = 10 终止空闲时间超过10秒的连接

thread_concurrency = 4

该值设定最大并发线程数,根据你的CPU数量*2来设定。

log-slow-queries = slow.log 慢查询的记录日志,查看日志然后对慢查询进行优化。

long_query_time = 5　　多长时间的查询算做是慢查询。

Log-queries-not-using-indexes

指定为没有使用索引的查询。你可以使用 mysqldumpslow 命令查看。

skip-innodb
skip-bdb

关闭不需要的表类型,如果你需要,就不要加上这个。

附：MYSQL 支持的表类型

DBD

Berkeley DB(DBD)表是支持事务处理的表，由 Sleepycat 软件公司(http://www.sleepycat.com)开发。它提供 MySQL 用户期待已久的功能-事务控制。事务控制在任何数据库系统中都是一个极有价值的功能，因为它们确保一组命令能成功地执行。

下面的表都属于非事务安全

HEAP

HEAP 表是 MySQL 中存取数据最快的表。这是因为他们使用存储在动态内存中的一个哈希索引。另一个要点是如果 MySQL 或服务器崩溃，数据将丢失。

ISAM

ISAM 表是早期 MySQL 版本的缺省表类型，直到 MyIASM 开发出来。建议不要再使用它。

MERGE

MERGE是一个有趣的新类型，在3.23.25之后出现。一个MERGE表实际上是一个相同MyISAM表的集合，合并成一个表，主要是为了效率原因。这样可以提高速度、搜索效率、修复效率并节省磁盘空间。

MyIASM

这是MySQL的缺省表类型。它基于IASM代码，但有很多有用的扩展。MyIASM比较好的原因：

1、MyIASM表小于IASM表，所以使用较少资源。

2、MyIASM表在不同的平台上二进制层可移植。

3、更大的键码尺寸，更大的键码上限。

你可在创建表时指定表的类型。下例创建一个HEAP表：

```
mysql>CREATE TABLE email_addresses TYPE=HEAP (
->email char(55) NOT NULL,
->name char(30) NOT NULL,
->PRIMARY KEY(email) );
```

第9章 MYSQL 簇

MySQL 簇是 MySQL 适合于分布式计算环境的高实用、高冗余版本。它采用了 NDB 簇存储引擎，允许在 1 个簇中运行多个 MySQL 服务器。NDB 是一种“内存中”存储引擎，它具有可用性高和数据一致性好的特点。

能够使用多种故障切换和负载平衡选项配置 NDB 存储引擎，但以簇层面上的存储引擎开始最简单。MySQL 簇的 NDB 存储引擎包含完整的数据集，仅取决于簇本身内的其他数据。

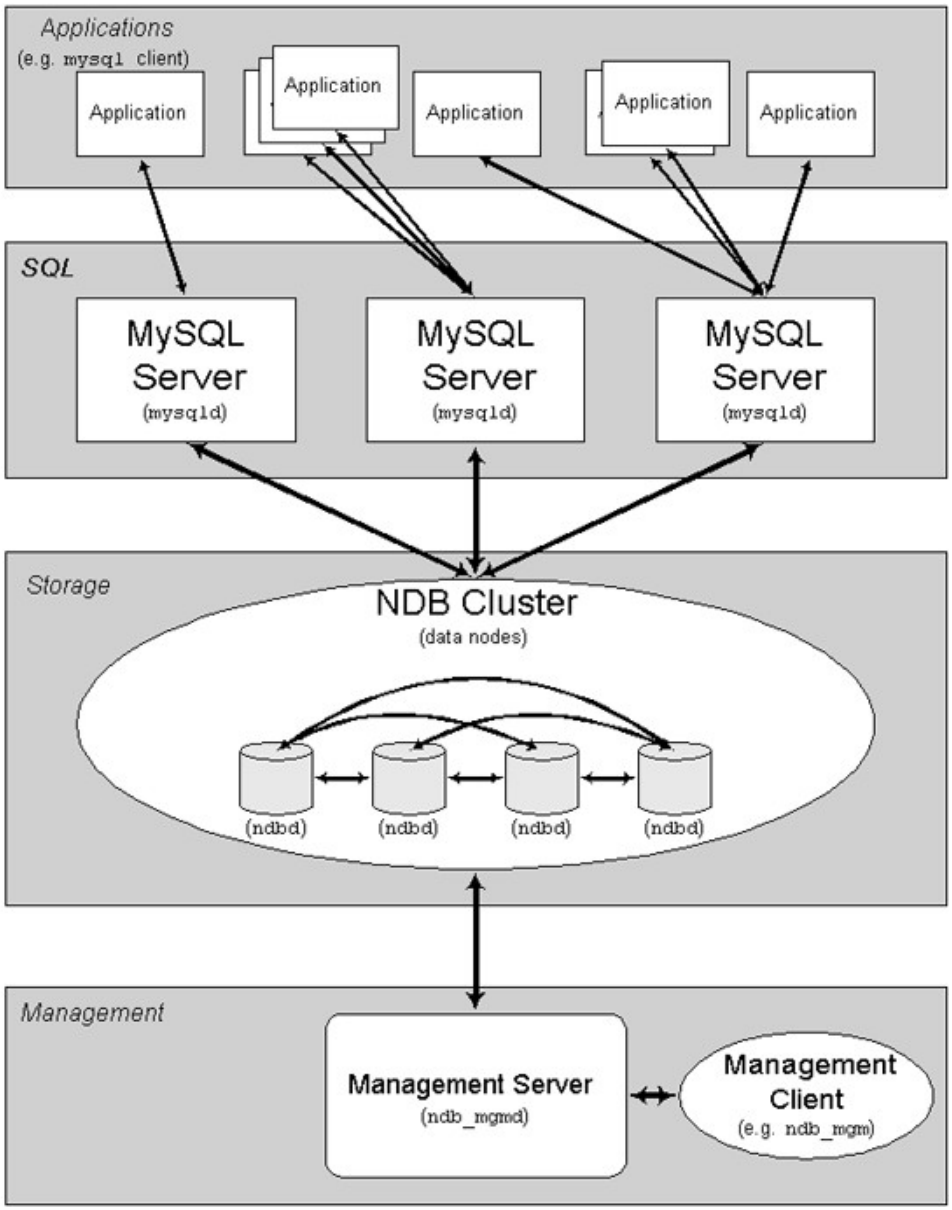
下面，我们介绍了设置由 NDB 存储引擎和一些 MySQL 服务器构成的 MySQL 簇的设置方法。

第9.1节 MYSQL 簇的基本概念

MySQL 簇是一种技术，该技术允许在无共享的系统中部署“内存中”数据库的簇。通过无共享体系结构，系统能够使用廉价的硬件，而且对软硬件无特殊要求。此外，由于每个组件有自己的内存和磁盘，不存在单点故障。

MySQL 簇将标准的 MySQL 服务器与名为 NDB 的“内存中”簇式存储引擎集成了起来。术语 NDB 指的是与存储引擎相关的设置部分，而术语“MySQL 簇”指的是 MySQL 和 NDB 存储引擎的组合。

MySQL 簇由一组计算机构成，每台计算机上均运行着多种进程，包括 MySQL 服务器，NDB 簇的数据节点，管理服务器，以及（可能）专门的数据访问程序。关于簇中这些组件的关系，请参见下图：



所有这些程序一起构成了MySQL簇。将数据保存到NDB簇存储引擎中时，表将保存在数据节点内。能够从簇中所有其他MySQL服务器直接访问这些表。因此，在将数据保存在簇内的工资表应用程序中，如果某一应用程序更新了1位雇员的工资，所有查询该数据的其他MySQL服务器能立刻发现这种变化。

对于MySQL簇，保存在数据节点内的数据可被映射，簇能够处理单独数据节点的故障，除了少数事务将因事务状态丢失而被放弃外，不会产生其他影响。由于事务性应用程序能够处理事务失败事宜，因而它不是问题源。

通过将MySQL簇引入开放源码世界，MySQL为所有需要它的人员提供了具有

高可用性、高性能和可缩放性的簇数据管理。

NDB 是一种“内存中”存储引擎，它具有可用性高和数据一致性好的特点。

能够使用多种故障切换和负载平衡选项配置 NDB 存储引擎，但以簇层面上的存储引擎开始最简单。MySQL 簇的 NDB 存储引擎包含完整的数据集，仅取决于簇本身内的其他数据。

目前，MySQL 簇的簇部分可独立于 MySQL 服务器进行配置。在 MySQL 簇中，簇的每个部分被视为 1 个节点。

有三类簇节点，在最低的 MySQL 簇配置中，至少有三个节点，这三类节点分别是：

·管理(MGM)节点：这类节点的作用是管理 MySQL 簇内的其他节点，如提供配置数据、启动并停止节点、运行备份等。由于这类节点负责管理其他节点的配置，应在启动其他节点之前首先启动这类节点。MGM 节点是用命令 ndb_mgmd 启动的。

·数据节点：这类节点用于保存簇的数据。数据节点的数目与副本的数目相关，是片段的倍数。例如，对于两个副本，每个副本有两个片段，那么就有 4 个数据节点。没有必要有一个以上的副本。数据节点是用命令 ndbd 启动的。

·SQL 节点：这是用来访问簇数据的节点。对于 MySQL 簇，客户端节点是使用 NDB 簇存储引擎的传统 MySQL 服务器。典型情况下，SQL 节点是使用命令 mysqld —ndbcluster 启动的，或将 ndbcluster 添加到 my.cnf 后使用 mysqld 启动。

簇配置包括对簇中单独节点的配置，以及设置节点之间的单独通信链路。对于目前设计的 MySQL 簇，其意图在于，从处理器的能力、内存空间和带宽来讲，存储节点是同质的，此外，为了提供单一的配置点，作为整体，簇的所有配置数据均位于 1 个配置文件中。

管理服务器(MGM节点)负责管理簇配置文件和簇日志。簇中的每个节点从管理服务器检索配置数据，并请求确定管理服务器所在位置的方式。当数据节点内出现有趣的事件时，节点将关于这类事件的信息传输到管理服务器，然后，将这类信息写入簇日志。

此外，可以有任意数目的簇客户端进程或应用程序。它们分为两种类型：

·标准MySQL客户端：对于MySQL簇，它们与标准的（非簇类）MySQL没有区别。换句话讲，能够从用PHP、Perl、C、C++、Java、Python、Ruby等编写的现有MySQL应用程序访问MySQL簇。

·管理客户端：这类客户端与管理服务器相连，并提供了优雅地启动和停止节点、启动和停止消息跟踪（仅对调试版本）、显示节点版本和状态、启动和停止备份等的命令。

第9.2节 MYSQL簇实验

实验目的：构建MYSQL簇，并测试MYSQL簇的工作性能。

实验所用到的软件包：

MySQL-client-community-5.0.51a-0.rhel4.i386.rpm

MySQL-clustermanagement-community-5.0.51a-0.rhel4.i386.rpm

MySQL-clusterstorage-community-5.0.51a-0.rhel4.i386.rpm

MySQL-clustertools-community-5.0.51a-0.rhel4.i386.rpm

MySQL-server-community-5.0.51a-0.rhel4.i386.rpm

perl-HTML-Template-2.9-1.el4.rf.noarch.rpm 或者

perl-HTML-Template-2.9-1.el5.rf.noarch.rpm

配置环境：

Server1: 192.168.0.1

Server2: 192.168.0.2

Server3: 192.168.0.3

Servers1 和 Server2 作为实际配置 MySQL 集群的服务器。Server3 作为管理节点，Server3 的系统进行很小的调整并且无需安装 MySQL。

[安装过程]

一、在 Server1 和 Server2 上安装

MySQL-client-community-5.0.51a-0.rhel4.i386.rpm

MySQL-server-community-5.0.51a-0.rhel4.i386.rpm

MySQL-clusterstorage-community-5.0.51a-0.rhel4.i386.rpm

安装完毕后，输入：

```
[root @server1 mysql ] # service mysql stop
```

暂时先把 MYSQL 停掉

二、安装并配置管理节点服务器(Server3)

作为管理节点服务器，Server3 需要 ndb_mgm 和 ndb_mgmd 两个文件，因此你需要安装：

MySQL-clustermanagement-community-5.0.51a-0.rhel4.i386.rpm

MySQL-clustertools-community-5.0.51a-0.rhel4.i386.rpm

由于在安装clustertools-community这个包的时候可能提示你需要安装per(HTML::Template)环境。那么就把下面这个包也装上。

perl-HTML-Template-2.9-1.el5.rf.noarch.rpm

三、开始为这台管理节点服务器建立配置文件：

```
[root @server1 mysql ] # mkdir /var/lib/mysql-cluster
[root @server1 mysql ] # cd /var/lib/mysql-cluster
[root @server1 mysql ] # vi config.ini
```

#在config.ini中添加如下内容：

```
[NDBD DEFAULT]
NoOfReplicas=2
[MYSQLD DEFAULT]
[NDB_MGMD DEFAULT]
[TCP DEFAULT]
# Managment Server
[NDB_MGMD]
HostName=192.168.0.3 #管理节点服务器Server3的IP地址
# Storage Engines
[NDBD]
HostName=192.168.0.1 #MySQL集群Server1的IP地址
DataDir= /var/lib/mysql-cluster
[NDBD]
HostName=192.168.0.2 #MySQL集群Server2的IP地址
DataDir=/var/lib/mysql-cluster
# 以下2个[MYSQLD]可以填写Server1和Server2的主机名。
```

```
# 但为了能够更快的更换集群中的服务器，推荐留空，否则更换服务器后必须对这个配置进行更改。
 [MYSQLD]
 [MYSQLD]
```

保存退出后，启动管理节点服务器 Server3：

```
[root @server1 mysql ] # ndb_mgmd -f /var/lib/mysql-cluster/config.ini
```

#启动管理节点后应该注意，这只是管理节点服务，并不是管理终端。因而你看不到任何#关于启动后的输出信息。

四、配置集群服务器并启动 MySQL

在 Server1 和 Server2 中都需要进行如下改动：

```
[root @server1 mysql ] # vi /etc/my.cnf

[mysqld]
 ndbcluster
 ndb-connectstring=192.168.0.3 #Server3 的 IP 地址
 [mysql_cluster]
 ndb-connectstring=192.168.0.3 #Server3 的 IP 地址
```

保存退出后，建立数据目录并启动 MySQL：

```
[root @server1 mysql ] #  mkdir /var/lib/mysql-cluster
[root @server1 mysql ] # cd /var/lib
[root @server1 mysql ] # chown mysql:mysql mysql-cluster
[root @server1 mysql ] #  ndbd —initial
[root @server1 mysql ] #  service mysql start
```

可以把/usr/local/mysql/bin/ndbd 加到/etc/rc.local 中实现开机启动。

注意：只有在第一次启动ndbd时或者对Server3的config.ini进行改动后才需要使#用--initial参数。平常就不需要使用--initial参数了。

五、检查工作状态

回到管理节点服务器Server3上，并启动管理终端：

```
[root @server1 mysql ] #  ndb_mgm
```

键入show命令查看当前工作状态：（下面是一个状态输出示例）

```
[root @server1 mysql ] #  ndb_mgm
-- NDB Cluster -- Management Client --
ndb_mgm> show
Connected to Management Server at: localhost:1186
Cluster Configuration
---------------------
[ndbd(NDB)]     2 node(s)
id=2    @192.168.0.2  (Version: 5.0.51, Nodegroup: 0, Master)
id=3    @192.168.0.3  (Version: 5.0.51, Nodegroup: 0)

[ndb_mgmd(MGM)] 1 node(s)
id=1    @192.168.0.1  (Version: 5.0.51)

[mysqld(API)]   2 node(s)
id=4    @192.168.0.2  (Version: 5.0.51)
id=5    @192.168.0.3  (Version: 5.0.51)
```

如果上面没有问题，现在开始测试MySQL：

#测试方法：在Server1中添加一个库，使用engine=nbdcluster建立指定可以被MYSQL-CLUSTER所使用的表。然后看看另外一个节点的效果。

```
[root @server1 mysql ] # mysql
```

```
mysql> use test;
mysql> create table testlll(id int,name char(10))engine=ndbcluster;
mysql> insert into table testlll(id,name) values (10,'test');
mysql> select * from testlll;
```

然后去另外一个节点看效果。

推荐使用一个大一点的数据库备份导入 。这样你可以顺便查看 MYSQL 簇的工作效率。根据之前的测试，MYSQL 的 MASTER/SLAVER 方式每秒可以录入 2500 条数据，现在你可以测试一下使用 MYSQL 簇每秒可以录入多少。

如果上述正常，则换到 Server2 上重复上面的测试，观察效果。

第10章 用源代码搭建LAMP环境

LAMP是一个缩写，它指一组通常一起使用来运行动态网站或者服务器的开源软件，包括:Linux操作系统，APACHE服务器，MYSQL，Perl，PHP或者Python编程语言。

虽然这些开放源代码程序本身并不是专门设计成同另外几个程序一起工作的，但由于它们都是影响较大的开源软件，拥有很多共同特点，这就导致了这些组件经常在一起使用。在过去的几年里，这些组件的兼容性不断完善，在一起的应用情形变得更加普遍。并且它们为了改善不同组件之间的协作，已经创建了某些扩展功能。目前，几乎在所有的Linux发行版中都默认包含了这些产品。Linux操作系统、Apache服务器、MySQL数据库和Perl、PHP或者 Python语言，这些产品共同组成了一个强大的Web应用程序平台。

本章的目标是完全通过源代码编译安装，组建一个LAMP的环境，并且在上面运行一个使用PHP语言编写的Discuz论坛。

第10.1.1节 构建安装环境

首先，我们可以通过rpm方式安装的软件包有以下几个：

gcc gcc-c++ flex bison autoconf automake bzip2-devel ncurses-devel libjpeg-devel libpng-devel libtiff-devel freetype-devel pam-devel

可以你使用如下命令一起检查它们是否安装：

```
[root @ server1 ~] # for i in gcc gcc-c++ flex bison autoconf automake bzip2-devel
ncurses-devel libjpeg-devel libpng-devel libtiff-devel freetype-devel pam-devel ; do rpm
-q $i ;done
```

接下来

第10.2节 编译安装MySQL

```
[root @ server1 ~] #tar xzvf mysql-5.0.27.tar.gz
[root @ server1 ~] # cd mysql-5.0.27
```

```
[root @ server1 mysql-5.0.27] # ./configure \
        "--prefix=/usr/local/mysql" \
        "--localstatedir=/var/lib/mysql" \
         "--with-comment=Source" \
        "--with-server-suffix=-Comsenz" \
        "--with-mysqld-user=mysql" \
        "--without-debug" \
        "--with-big-tables" \
         "--with-extra-charsets=all" \
        "--with-pthread" \
        "--enable-static" \
        "--enable-thread-safe-client" \
        "--with-client-ldflags=-all-static" \
        "--with-mysqld-ldflags=-all-static" \
        "--enable-assembler" \
        "--without-isam" \
        "--without-innodb" \
        "--without-ndb-debug"
[root @ server1 msyql-5.0.27] # make
[root @ server1 mysql-5.0.27] # make install
[root @ server1 mysql-5.0.27] # useradd mysql
[root @ server1 mysql-5.0.27] # cd /usr/local/mysql
[root @ server1 mysql] # bin/mysql_install_db --user=mysql
[root @ server1 mysql] # chown -R root:mysql .
[root @ server1 mysql] # chown -R mysql /var/lib/mysql
[root @ server1 mysql] # cp share/mysql/my-huge.cnf /etc/my.cnf
[root @ server1 mysql] # cp share/mysql/mysql.server /etc/rc.d/init.d/mysqld
[root @ server1 mysql] # chmod 755 /etc/rc.d/init.d/mysqld
[root @ server1 mysql] # chkconfig --add mysqld
[root @ server1 mysql] # chkconfig --level 3 mysqld on
[root @ server1 mysql] # /etc/rc.d/init.d/mysqld start
[root @ server1 mysql] # bin/mysqladmin -u root password 'password_for_root'
```

第10.3节 编译安装Apache

```
[root @ server1 src] # cd /usr/local/src
[root @ server1 src] # tar xjvf httpd-2.2.4.tar.bz2
[root @ server1 src] # cd httpd-2.2.4
[root @ server1 httpd-2.2.4] # ./configure \
        "--prefix=/usr/local/apache2" \
        "--with-included-apr" \
        "--enable-so" \
        "--enable-deflate=shared" \
        "--enable-expires=shared" \
        "--enable-rewrite=shared" \
        "--enable-static-support" \
        "--disable-userdir"
[root @ server1 httpd2.2.4] # make
[root @ server1 httpd2.2.4] # make install
[root @ server1 httpd2.2.4] # echo '/usr/local/apache2/bin/apachectl start ' >>
/etc/rc.local
```

第10.4节 编译安装PHP

```
[root @ server1 ~] # cd /usr/local/src
[root @ server1 src] # tar xjvf php-5.2.3.tar.bz2
[root @ server1 php-5.2.3] # cd php-5.2.3
[root @ server1 php-5.2.3] # ./configure \
        "--prefix=/usr/local/php" \
        "--with-apxs2=/usr/local/apache2/bin/apxs" \
        "--with-config-file-path=/usr/local/php/etc" \
        "--with-mysql=/usr/local/mysql" \
        "--with-libxml-dir=/usr/local/libxml2" \
        "--with-jpeg-dir" \
```

```
        "--with-png-dir" \
        "--with-bz2" \
        "--with-freetype-dir" \
        "--with-iconv-dir" \
        "--with-zlib-dir " \
        "--enable-soap" \
        "--enable-gd-native-ttf" \
        "--enable-memory-limit" \
        "--enable-ftp" \
        "--enable-mbstring" \
        "--enable-exif" \
        "--disable-ipv6" \
        "--disable-cgi" \
        "--disable-cli"
[root @ server1 php-5.2.3] # make
[root @ server1 php-5.2.3] # make install
[root @ server1 php-5.2.3] # mkdir /usr/local/php/etc
[root @ server1 php-5.2.3] # cp php.ini-dist /usr/local/php/etc/php.ini
```

第10.5节 安装 Zend Optimizer

```
[root @ server1 ~] # cd /usr/local/src
[root @ server1 src] # tar xzvf ZendOptimizer-3.2.8-linux-glibc21-i386.tar.gz
[root @ server1 src] # ./ZendOptimizer-3.2.8-linux-glibc21-i386/install.sh
```

第10.6节 整合 Apache 与 PHP

```
[root @ server1 src] # vi /usr/local/apache2/conf/httpd.conf
```

找到：

```
AddType application/x-gzip .gz .tgz
```

在该行下面添加

```
AddType application/x-httpd-php .php
```

找到：

```
<IfModule dir_module>
    DirectoryIndex index.html
</IfModule>
```

将该行改为

```
<IfModule dir_module>
    DirectoryIndex index.html index.htm index.php
</IfModule>
```

找到：

```
#Include conf/extra/httpd-mpm.conf
#Include conf/extra/httpd-info.conf
#Include conf/extra/httpd-vhosts.conf
#Include conf/extra/httpd-default.conf
```

去掉前面的“#”号，取消注释。

注意：以上 4 个扩展配置文件中的设置请按照相关原则进行合理配置！

修改完成后保存退出。

```
[root @ server1 src] # /usr/local/apache2/bin/apachectl restart
```

第10.7节 测试PHP并提高安全性

在网站根目录放置 phpinfo.php 脚本，检查phpinfo中的各项信息是否正确。

```
#vi phpinfo.php
<?php
```

```
phpinfo();
?>;
```

确认 PHP 能够正常工作后，在 php.ini 中进行设置提升 PHP 安全性。

```
[root @ server1 src] # vi /etc/php.ini
```

找到：

```
disable_functions =
```

设置为：

```
phpinfo,passthru,exec,system,chroot,scandir,chgrp,chown,escapeshellcmd,escapeshell
arg,shell_exec,proc_open,proc_get_status,error_log,ini_alter,ini_alter,ini_restore,dl,pfso
ckopen,openlog,syslog,readlink,symlink,leak,popepassthru,stream_socket_server
```

第10.8节 安装、配置 Discuz!6.0

首先，解压缩 Discuz 安装包到临时目录里。

```
[root@server1 LAMP]# tar zxvf discuz.tar.gz -C /tmp
```

之后，会在/tmp 目录中解压出一个 discuz 目录，在这个目录中有三个目录。将 upload 目录移动到你的 web 服务器的主目录中，并改名为 bbs。

```
[root@server1 LAMP]# mv   /tmp/discuz/upload   /var/www/html/bbs
```

然后，你需要在 MySQL 中创建用于安装 Discuz 的库和管理该库的用户。

```
mysql> create database discuz;
Query OK, 1 row affected (0.02 sec)

mysql> grant all on discuz.* to discuz@'localhost' identified by '1234';
Query OK, 0 rows affected (0.00 sec)

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

mysql>
```

然后，打开浏览器，输入你的主机地址，访问 install.php 这个页面，开

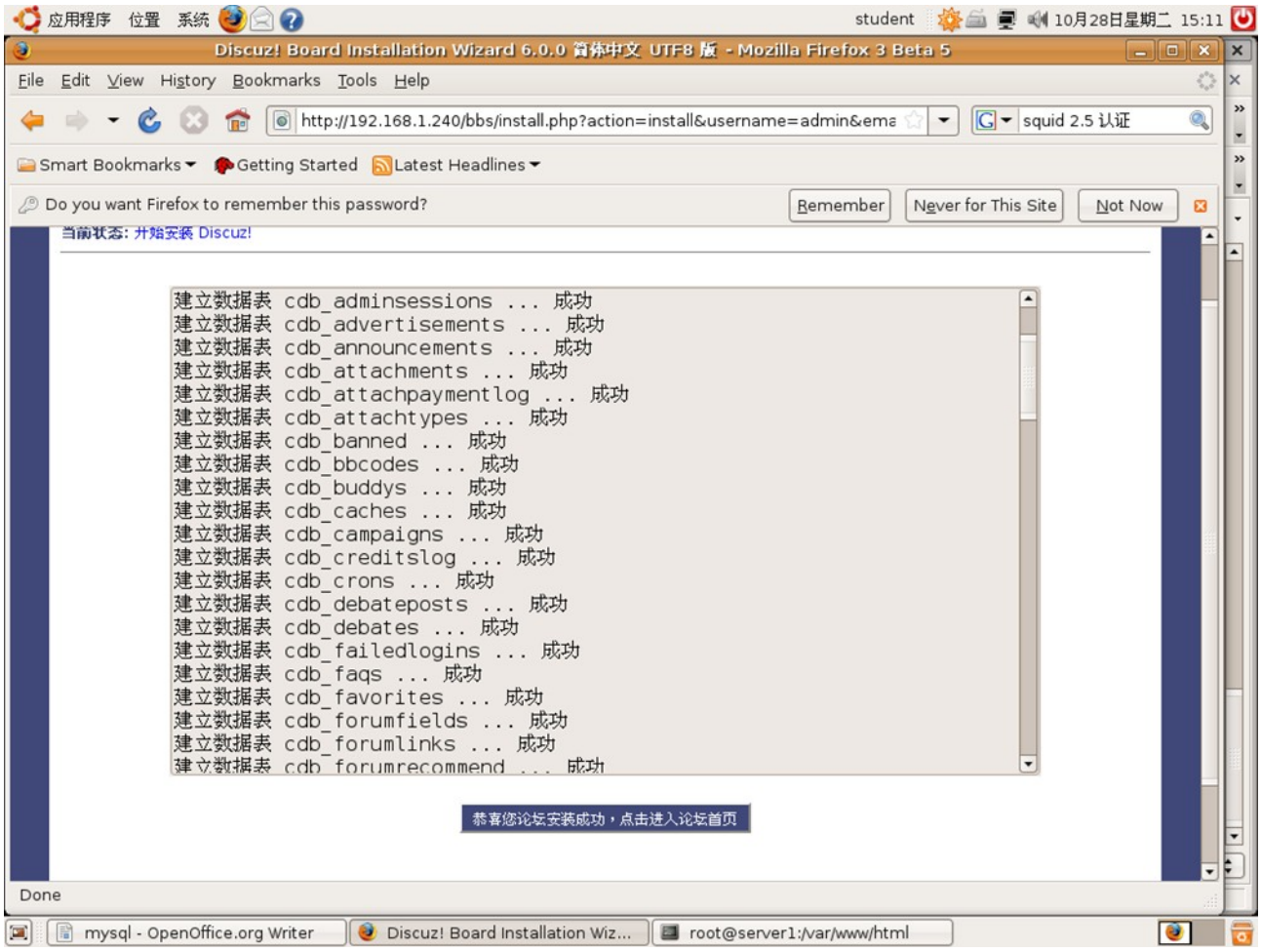
始安装 discuz。例如：http://yourhost/bbs/install.php

点击同意，进入下一步。这个时候安装脚本会检查你的系统配置是否适合安装 Discuz。并且，要求你将自定义头像目录(./customavatars)、附件目录(默认是 ./attachments)、数据目录(./forumdata)、数据缓存目录(./forumdata/cache)、数据缓存目录(./forumdata/threadcaches)、数据缓存目录(./forumdata/threadcaches)给予写入权限或者 777。在这里我强烈反对将任何一个目录赋予 777 这样的危险权限。你可以把这些目录的拥有者设置为 apache 用户。

```
[root@server1 LAMP]# cd   /var/www/html/
[root@server1 html]# chown -R  apache.   Bbs
```

接下来，安装脚本提示你输入 Discuz 所使用的数据库位置、用户名、密码、管理员 E-mail 和表名前缀。按照我们刚才的设置输入就可以了。

最后，设置论坛的管理员密码，就可以完成最后的安装了。



最后，点击进入论坛首页，就可以看到刚刚安装好的论坛了。

至此，整个实验结束。