第11章　常规的MySQL 管理

本章将讨论 MySQL 管理员为保持 MySQL 正常运行所需要完成的职责。这些职责包括确保服务器尽可能保持高性能地运行、设置用户账号以便客户机能够访问服务器、维护日志文件，以及执行数据库备份。为了得到更高的性能，管理员还可以运行多个服务器或修改服务器的操作参数。最后，由于 MySQL 正处于高速发展的状态，所以管理员必须能确定何对 MySQL 进行升级。其他重要的管理问题将在第 12章“安全性”和第13章“数据库维护和修复”介绍。

有几个对 MySQL 管理员有用的程序包括在这些章节中：

- mysqladmin 执行各种各样的管理操作。
- safe_mysqld 和 mysql.server 是启动 MySQL 服务器 mysqld 的脚本。
- mysqldump 用于数据库备份和拷贝操作。
- myisamchk 和 isamchk 用于表的完整性检查和修复操作。

有关这些程序的附加信息，请参阅附录 E“MySQL 程序参考”。

11.1　新的 MySQL 安装的安全性

您可能因为刚刚按照附录 A“获得和安装软件”一节的说明安装了 MySQL 而正在阅读本章。如果是这样的话，则需要为 MySQL root 用户设置口令——对于一个新的 MySQL 安装来说，服务器正在以不安全的权限运行着。笔者假定您已经初始化了数据目录和包含授权表的 mysql数据库。在 UNIX 中，可通过运行 mysql_install_db 脚本来进行。在 Windows 中，数据目录和 mysql 数据库通过运行服务器分发包中的 Setup 程序初始化。笔者还假定服务器正在运行中。

当初次在机器上安装 MySQL 时，mysql 数据库的授权表的初始权限如下：

- 可以从本地主机中以 root 进行连接，不带任何口令。root 用户拥有所有的权限（其中包括管理权限），因此能做许多事情（顺便说一句，MySQL 超级用户和 UNIX 超级用户二者都有名字 root，这一事实是巧合的。它们相互之间没有任何关系）。
- 匿名的访问授予从本地主机上连接的用户，该主机拥有名为 test 的数据库或带有以‘test_’开始的任何数据库。匿名用户可以利用这样的数据库做任何事，但没有管理权限。

无论连接的用户指定的是 localhost 主机名还是实际的主机名，从本地主机到服务器的连接都将是允许的。例如，如果服务器在 pit - viper.snake.net 上，则该主机上的客户机能够不使用口令而连接到该服务器，从而可利用下列之一的语句使用 test 数据库：

```
% mysql -h localhost test
% mysql -h pit-viper.snake.net test
```

您甚至可以不用口令就以 root 用户的身份连接到 MySQL，这一事实说明：初始的安装是不安全的。因此，作为 MySQL 管理员最初的行动之一应该是为 root 用户设置口令。然后，根据用来设置口令的方法，您还需要指示服务器重新加载授权的表，使服务器知道这些变化

（在服务器启动时，它将这些表加载到内存中，并且在没有意识到的情况下已经将它们改变。如果是这样的话，必须明确告诉它重新读取这些表）。

对于 MySQL 3.22 以上的版本，可以用 *mysqladmin* 程序按如下方式设置口令，可用您选择的口令来替代下列命令中的“ *your password* ”：

```
% mysqladmin -u root password "your password"
```

对于任何版本的 MySQL，您都可以使用 mysql 程序并直接更新 mysql 数据库中的用户授权的表：

```
% mysql -u root mysql
mysql> UPDATE user SET Password=PASSWORD("your password")
    -> WHERE User="root";
```

如果您有一个旧版本的 MySQL，则可使用 mysql 和一个 UPDATE 语句，这也包括 Windows 的共享软件版本。

在设置口令后，要看看是否需要通过运行下列命令指示服务器重新加载授权的表：

```
% mysqladmin -u root status
```

如果服务器仍然允许不使用口令就以 root 用户身份进行连接，则应告诉它重新读取授权的表（MySQL 3.22 以上的版本还允许用 mysqladmin flush.privileges 命令和 FLUSH PRIVILEGES 的SQL 语句重新加载表）：

```
% mysqladmin -u root reload
```

在设置了 root 的口令（并重新加载授权的表，如果必要的话）后，当以 root 的身份连接到服务器时，需要指定新的口令。

11.2 MySQL 服务器的启动和关闭

作为 MySQL 管理员，一个普通的目标就是确保服务器尽可能地处于运行状态，使得客户机能够随时访问它。但是，有时最好关闭服务器（例如，如果正在进行数据库的重定位，不希望服务器在该数据库中更新表）。保持服务器运行和偶尔关闭它的需求关系不是本书所解决的。但是我们至少可以讨论如何使服务器启动和停止，以便您具备进行这两个操作的能力。

本章的说明只用于 UNIX 系统。如果正在运行 Windows 系统，可以跳过本章，因为附录 A “ 获得和安装软件 ” 一节中包含了所有需要的启动和关闭命令。

> **调用本章给出的命令**
>
> 为了简洁，在大多数情况中，诸如 mysqladmin、mysqldump 等程序在本章中没有给出任何 -h、-u或-p选项。笔者假定您将会用连接服务器所需的任何选项调用这些程序。

11.2.1 用无特权的用户账号运行 MySQL 服务器

在讨论如何启动服务器之前，考虑一下在服务器启动时应该运行哪个账号。服务器可以手工和自动启动。如果手工启动，则服务器以 UNIX 用户身份运行（您恰好作为该用户进行了注册）。即，如果笔者以 paul 进行注册并启动服务器，则它将以 paul 身份运行。如果用 su 命令将用户切换到 root 然后启动服务器，则服务器以 root 身份运行。

但是，大多数时候可能都不会采用手工启动服务器。您很可能将安排服务器在系统引导时作为标准启动过程的一部分自动地运行。在 UNIX中，该启动过程由系统以 UNIX 的 root

用户的身份执行，该过程中启动的任何进程都用 root 的权限运行。

应该紧记 MySQL 服务器启动过程的两个目标：

- **要服务器以某些非 root 的用户身份启动。**通常，除非进程真的需要 root 访问权而 mysql 办不到，否则应限制任何进程的能力。
- **要服务器始终以同一个用户的身份运行。**服务器有时作为一个用户运行而有时又作为另一个用户运行时会产生矛盾。这将导致文件和目录以不同的所有权在该数据下被创建，甚至引起服务器不能访问数据库或表。以同一个用户的身份一致地运行服务器可以避免该问题。

为了以标准的、非特权的用户身份运行数据库，可按如下步骤执行该过程：

1) 选择用于运行服务器的账号。mysqld 可以以任何用户身份运行，但是很明显，它只为 MySQL 活动创建了一个单独的账号。您也可以为 MySQL 专门指定一个组。笔者将调用的这些用户和组的名字命名为 mysqladm 和 mysqlgrp。如果您使用了其他的名字，则在本书中有 mysqladm 和 mysqlgrp 的地方替换它们

如果您在自己的账号下安装了 MySQL 并且系统中没有特定的管理权限，则您可以在自己的 ID 用户下运行服务器。在这种情况下，应使用您自己的注册名和组名替代 mysqladm 和 mysqlgrp 。

如果您利用 RPM 文件在 RedHat Linux 下安装了 MySQL，则该安装程序将在 mysql 名下自动创建了一个账号。应使用该名字替换 mysqladm 。

2) 如果必要的话，可用系统常用的账号创建过程（ account-creation）来创建服务器账号。这需要以 root 身份进行操作。

3) 关闭服务器（如果它在运行）。

4) 修改数据目录以及任何子目录和文件的所有权，使 mysqladm 用户拥有它们。例如，如果数据目录是 /usr/local/var，则可按以下设置 mysqladm 用户的所有权：

```
# cd /usr/local/var                移动到数据目录
# chown -R mysqladm.mysqlgrp       设置所有目录和文件的所有权
```

5) 修改数据目录以及任何子目录和文件的许可权，使得只有 mysqladm 用户能够访问它们。设置该方式以避免其他人员访问是一种好得安全预防措施。如果数据目录是 /usr/local/var，则可通过 mysqladm 用户按下列操作设置应具有的一切（您需要以 root 身份运行这些命令）：

```
# cd /usr/local/var                移动到数据目录
# chmod -R go -rwx                 使所有一切只对 mysqladm 可访问
```

在设置数据目录及其内容的所有权和方式时，观察符号连接。您需要跟踪符号连接并修改所指向的文件或目录的所有权和方式。如果这些连接文件所定位的目录不属于您，则这样做可能会引起麻烦，因此您必须是 root 用户。

在完成前述过程后，应确保无论是作为 mysqladm 还是作为 root 用户注册都始终启动服务器。在后者中，要确保指定了 --user = mysqladm 的选项，使服务器可以将其用户 ID 切换到 mysqladm（该选项在系统启动过程中也可使用）。

--user 选项被增加到 MySQL 3.22 的 mysql 中。如果您的版本比 MySQL 3.22 旧，则在启动服务器并作为 root 用户运行时，可以使用 su 命令指示系统在指定账号下运行服务器。您需要阅读有关 su 的人工页，因为作为一个指定用户运行命令的语法被改变了。

11.2.2 启动服务器的方法

如果您已经确定了用来运行服务器的账号，则可以选择安排怎样启动服务器。可以从命令行手工运行，或在系统启动过程中自动运行服务器。有三种启动服务器的主要方法：

- **直接调用 mysqld。**这或许是最小的命令方法。除了说明 mysqld --help 是一个有用的命令（用它可以查找您可利用其他启动方法使用的选项）外，笔者不打算进一步讨论它。
- **调用 safe_mysqld 脚本。**safe_mysqld 试图确定服务器程序和数据目录的位置，然后利用反映这些位置的选项调用服务器。safe_mysqld 将服务器的标准错误输出重定向到数据目录的错误文件中，并以记录的形式出现。在启动服务器后，safe_mysqld 还监控服务器，并在其死机时重新启动。safe_mysqld 通常用于UNIX 的 BSD 风格的版本。

  如果您曾经作为 root 或在系统启动程序中启动 safe_mysqld，则错误日志将由 root 拥有。如果您试着以非特权的用户身份调用 safe_mysqld，则可能引起“所有权被拒绝”的错误。删除该错误文件再试一次。
- **调用 mysql.server 脚本。**通过运行 safe_mysqld.mysql.server，该脚本启动服务器。该脚本建议在使用 System V 启动/关闭系统的系统中使用。这个系统包括几个包含在机器登录或退出一个特定运行级时被调用的脚本的目录。它可以利用 start 或 stop 参数进行调用，以指明希望启动还是关闭服务器。

  safe_mysqld 脚本被安装在 MySQL 安装目录的 bin 目录下，或者在 MySQL 源程序分发包的 scripts 目录中。mysql.server 脚本安装在 MySQL 安装目录的 share/mysql 目录下，或者在 MySQL 源程序分发包的 support-files 目录中。如果要使用它，应将其拷贝到合适的启动目录中。

对于 BSD 风格的系统，在 /etc 目录中有几个文件相对应，它们在引导期间开始服务。这些文件的名字通常以‘ rc ’开始，因此很可能会有一个名为 rc.local （或类似的名字）的文件来启动本地的安装服务。在这样的系统中，您可能要按如下方法添加一些行到 rc.local 文件中以启动服务器（如果路径与您系统中的不同，可将其修改成 safe_mysqld）：

```
if [ -x /usr/local/bin/safe_mysqld ]; then
    /usr/local/bin/safe_mysqld &
fi
```

对于 System V 风格的系统，可以通过将其放置在 /etc 下的合适的启动目录中来安装 mysql.server。如果您运行 Linux 并从 RPM 文件中安装了 MySQL，那么这此操作可能已经完成了。否则，应该在主启动脚本目录中安装该脚本，并在合适的运行级目录中设置对它的连接。您还可使该脚本仅对 root 用户可执行。

启动文件目录的布局随系统而变化，因此将需要全面检查来弄清系统是怎样组织它们的。例如，在 LinuxPPC 中，这些目录为 /etc/rc.d/init.d 和 /etc/rc.d/rc3.d。应该按如下方法安装该脚本：

```
# cp mysql.server /etc/rc.d/init.d
# cd /etc/init.d
# chmod 500 mysql.server
# cd /etc/rc.d/rc3.d
# ln -s ../init.d/mysql.server S99mysql
```

在 Solaris 中，主脚本目录为 /etc/init.d，运行级目录为 /etc/rc2.d，因此上述命令将替换为：

```
# cp mysql.server /etc/init.d
# cd /etc/init.d
# chmod 500 mysql.server
# cd /etc/rc2.d
# ln -s ../init.d/mysql.server S99mysql
```

在系统启动期间，S99mysql 脚本利用 start 参数自动调用。

如果您拥有 chkconfig 命令（它在 Linux 中很常用），则可用其帮助安装 mysql.server 脚本来代替手工运行上述的命令。

1. 指定启动选项

在启动服务器时，如果想要指定附加的启动选项，可用两种方法进行操作。您可以修改所使用的启动脚本（safe_mysqld 或 mysql.server），并在调用服务器的命令行中直接指定这些选项。您还可以在选项文件中指定选项。笔者建议，如果可能的话，应在全局选项文件中指定服务器选项。通常该文件的位置是 UNIX 中的 /etc/my.cnf 和 Windows 中的 c:\my.cnf（有关使用选项文件的细节，请参阅附录 E）。

某些种类的信息不能作为服务器的选项指定。为了这些选项，您可能需要修改 safe_mysqld。例如，如果服务器不能正确地拾取 GMT 中的本地时区（local time zone）和返回时间值，可以设置 TZ 环境变量以给该变量一个提示。如果用 safe_mysqld 或 mysql.server 启动服务器，可以将时区设置增加到 safe_mysqld 中。找到启动服务器的命令行，并在该行之前增加下列命令：

```
TZ=US/Central
export TZ
```

这个命令将 TZ 设置为 US Central 时区。您需要使用合适位置的时区。该语法是 Solaris 的，您的系统可能会有所不同。例如，设置 TZ 变量的另一个常用语法为：

```
TZ=CST6CDT
export TZ
```

如果修改了启动脚本，当下次安装 MySQL 时（如，升级到更新的版本），将失去这些修改，除非在之前将该启动脚本拷贝到了其他地方。在安装新的版本之后，将您的脚本与新安装的脚本进行比较，以便看看重新建立还需要做什么改动。

2. 在启动期间检查表

除了在系统引导时安排服务器的启动外，您还可以安装一个脚本来运行 mysamchk 和 isamchk，以便在服务器启动前对表进行检查。您可能打算在服务器崩溃后重新启动，但表可能已经毁坏了。在服务器启动前检查这些表是发现问题的好办法。第 13 章包含了有关编写和安装这种脚本的细节。

11.2.3 关闭服务器

要想手工关闭服务器，可使用 mysqladmin：

```
% mysqladmin shutdown
```

要想自动关闭服务器，您不需要做特别的操作。 BSD 系统通常会通过给进程发送一个 TERM 信号来关闭服务。进程或者对其作出反应，或者被随便地取消。当 mysqld 接收到信号时，它会通过终止来响应。对于利用 mysql.server 启动服务器的 System V-风格的系统，该关闭进程将调用带有 stop 参数的脚本来指示服务器进行关闭——当然，这是在假定您已经安装了 mysql.server的情况下进行的。

11.2.4 在不连接时收回服务器的控制

在某些环境中，由于不能连接到服务器，您需要用手工重新启动它。当然，这有点荒谬，因为一般是通过连接到服务器然后告知服务器终止来手工关闭服务器的。那么这种情况是怎样出现的？

首先，MySQL 的root 口令可能得到了一个您不知道的值。这种情况可能是在修改口令时发生的——例如，如果在输入新的口令值时碰巧键入了一个不可见的控制字符。还有可能就是完全忘记了口令。

其次，对于 localhost 的连接通常是通过 UNIX 域的套接字文件进行的，它一般为/tmp/mysql.sock。如果该套接字文件被删除了，则本地客户机将不能进行连接。如果系统偶尔运行了一个删除 /tmp 中的临时文件的 cron 作业，这种情况就可能会发生。

如果因为失去套接字文件而不能进行连接，可以通过重新启动服务器简单地进行恢复，因为服务器在启动期间重新建立了该文件。这里应知道的是，不能用该套接字建立连接（因为它已经不存在）而必须建立 TCP/IP 连接。例如，如果服务器的主机是 pit - viper.snake.net，则可以按如下方法进行连接：

```
% mysqladmin -p -u root -h pit-viper.snake.net shutdown
```

如果此套接字文件被 cron 作业删除，则问题将复发，直到您修改 cron 作业或使用另一个套接字文件为止。您可以用全局选项文件指定另一个套接字文件。例如，如果数据目录为/usr/local/var，则可通过将以下行添加到 /etc/my.cnf 中来移动套接字文件到那里：

```
[mysqld]
socket=/usr/local/var/mysql.sock

[client]
socket=/usr/local/var/mysql.sock
```

路径名是为服务器和客户机程序二者所指定的，以便它们能使用相同的套接字文件。如果只对服务器设置路径名，客户机程序将仍然在旧的位置上查找套接字文件。在做出这个修改后应重新启动服务器，使它在新的位置创建套接字文件。

如果由于您忘记了 root 的口令或将其修改为一个您不知道的值而不能进行连接，则需要收回服务器的控制以便重新设置口令：

- **关闭服务器**。如果您以 root 用户的身份在服务器主机上进行登录，可用 kill 命令终止服务器。通过使用 ps 命令或通过查看服务器的 PID 文件（通常放在数据目录中）能找出服务器的 ID 进程。

  最好先试着用标准的 kill 命令取消服务器，该命令将一个 TERM 信号发送到服务器上，以查看服务器是否通过关闭信号来响应。也就是说，表和日志将被适当地刷新。如果服务器被堵塞并且没有响应正常的终止信号，可使用 kill -9 强制终止它。这是最后的一个方法，因为可能存在未刷新的更改，并且要承担非一致状态下将表保留下来的风险。

  如果用 kill -9 终止服务器，应确保在重新启动服务器之前利用 myisamchk 和 isamchk 对表进行检查（参见第13章）。
- **用 --skip-grant-tables 选项重新启动服务器**。该操作告诉服务器不要使用授权的表检查连接。这允许您作为 root 用户不用输入口令即可进行连接。在连接之后，修改 root 的口令。

■ **告诉服务器再利用 mysqladmin flush-privileges 使用授权表启动。**如果您的 mysqladmin 版本不识别 flush-privileges，试着进行重新加载。

11.3 用户账号管理

MySQL 管理员应该知道怎样通过指定哪些用户可连接到服务器、从哪里进行连接，以及在连接时做什么，来设置 MySQL 用户账号。MySQL 3.22.11 引入了两个更容易进行这项工作的语句：GRANT 语句创建MySQL 用户并指定其权限，REVOKE 语句删除权限。这两个语句充当 mysql 数据库中的授权表的前端，并提供直接操纵这些表内容的可选择的方法。GRANT 和 REVOKE 语句影响以下四个表：

授权表	内容
user	可连接到服务器的用户和他们拥有的任何全局特权
db	数据库级的特权
tables_priv	表级特权
columns_priv	列级特权

还有第五个授权表（host），但它不受 GRANT 或 REVOKE 的影响。

当您为某个用户发布 GRANT 语句时，应在 user 表中为该用户创建一个项。如果该语句指定了所有全局特权（管理权限或用于所有数据库的权限），则这些指定也被记录在 user 表中。如果指定了数据库、表或列的权限，它们将记录在 db、tables_priv 和 columns_priv 表中。

使用 GRANT 和 REVOKE 语句比直接修改授权表更容易。但是，建议您最好通过阅读第12章来补充本章的内容，第 12章中详细讨论了授权表。这些表非常重要，作为一位管理员应该了解这些表是怎样在 GRANT 和 REVOKE 语句级上工作的。

本节下面的部分将讨论如何设置 MySQL 用户的账号和授权，还将介绍如何取消权限以及从授权表中删除全部用户，并且将考虑一个困扰许多新的 MySQL 管理员的难题。

您还要考虑使用 mysqlaccess 和 mysql_setpermission 脚本，它们是 MySQL 分发包的组成部分。这些是 Perl 的脚本，它们提供了设置用户账号的 GRANT 语句的代用品。mysql_setpermission 需要具有 DBI 的支持环境。

11.3.1 创建新用户和授权

GRANT 语句的语法如下：

```
GRANT privileges (columns)
    ON what
    TO user IDENTIFIED BY "password"
    WITH GRANT OPTION
```

要使用该语句，需要填写以下部分：

■ privileges 分配给用户的权限。下表列出了可在 GRANT 语句中使用的权限说明符：

权限说明符权限允许的操作

ALTER	更改表和索引
CREATE	创建数据库和表
DELETE	删除表中已有的记录
DROP	删除数据库和表
INDEX	创建或删除索引
INSERT	插入新的记录到表中

REFERENCES	未用
SELECT	检索表中的已有记录
UPDATE	修改已有的表记录
FILE	读写服务器中的文件
PROCESS	查看有关线程在服务器中的执行或取消线程的信息
RELOAD	重新加载授权表或刷新日志、主机高速缓存或表高速缓存
SHUTDOWN	关闭服务器
ALL	任何事。ALL PRIVILEGES 是同义词
USAGE	一个特殊的“无权限”的权限

上表显示的第一组权限说明符适用于数据库、表和列。第二组说明符是管理特权。通常，这些权限的授予相当保守，因为它们会影响服务器的操作（例如， SHUTDOWN 特权不是按每天来分发的权限）。第三组说明符是特殊的。 ALL 的意思是“所有的权限”，而 USAGE 的意思是“无权限”——即创建用户，但不授予任何的权限。

- columns 权限适用的列。这是可选的，只来设置列专有的权限。如果命名多于一个列，则用逗号分开。
- what 权限应用的级别。权限可以是全局的（适用于所有数据库和所有的表）、数据库专有的（适用于某个数据库中的所有表），或表专有的。可以通过指定一个 COLUMNS 子句将权限授予特定的列。
- user 使用权限的用户。它由用户名和主机名组成。在 MySQL 中，不仅指定谁进行连接，还要指定从哪里连接。它允许您拥有两个带有相同名字的、从不同位置连接的用户。MySQL 允许在它们之间进行区别并相互独立地分配权限。

  MySQL 的用户名就是您在连接到服务器时指定的名字。该名字与您的 UNIX 注册名或 Windows 名的没有必然连系。缺省设置时，客户机程序将使用您注册的名字作为 MySQL 的用户名（如果您不明确指定一个名字的话），但这只是一个约定。有关将 root 作为可以操作一切MySQL 的超级用户名也是这样，就是一种约定。您也可以在授权表中将此名修改成 nobody，然后作为 nobody 用户进行连接，以执行需要超级用户特权的操作。
- password 分配给该用户的口令。这是可选的。如果您不给新用户指定 IDENTIFIED BY 子句，该用户不分配口令（是非安全的）。对于已有的用户，任何指定的口令将替代旧口令。如果不指定新口令，用户的旧口令仍然保持不变。当您确实要使用 IDENTIFIED BY 时，该口令串应该是直接量， GRANT 将对口令进行编码。当用 SET PASSWORD 语句时，不要使用 PASSWORD() 函数。

WITH GRANT OPTION 子句是可选的。如果包含该子句，该用户可以将 GRANT 语句授予的任何权限授予其他的用户。可以使用该子句将授权的能力授予其他的用户。

用户名、口令以及数据库和表的名称在授权表项中是区分大小写的，而主机名和列名则不是。

通过查询某些问题，通常可以推断出所需的 GRANT 语句的类型：

- 谁可以进行连接，从哪里连接？
- 用户应具有什么级别的权限，这些权限适用于什么？
- 允许用户管理权限吗？

让我们来提问这些问题，同时看一些利用 GRANT 语句设置 MySQL 用户账号的例子。

1. 谁可以进行连接，从哪里连接

您可以允许用户在特定的主机或涉及范围很宽的一组主机中进行连接。在一个极端，如果知道用户将仅从那个主机中进行连接，则可限定对单个主机的访问：

```
GRANT ALL ON samp_db.* TO boris@localhost IDENTIFIED BY "ruby"
GRANT ALL ON samp_db.* TO fred@ares.mars.net IDENTIFIED BY "quartz"
```

（符号 samp_db.* 含义是“在 samp_db 数据库中的所有表”）在另一个极端，您可能会有一个用户 max，他周游世界并需要能够从世界各地的主机中进行连接。在这种情况下，无论他从哪里连接您都将允许：

```
GRANT ALL ON samp_db.* TO max@% IDENTIFIED BY "diamond"
```

‘%’字符起通配符的作用，与 LIKE 模式匹配的含义相同，在上个语句中，它的意思是“任何主机”。如果您根本不给出主机名部分，则它与指定“ %”的含义相同。因此， max 和 max@% 是等价的。这是设置一个用户最容易的方法，但安全性最小。

要想采取妥协的办法，可允许用户在一组有限的主机中进行连接。例如，要使 mary 从 snake.net 域的任何主机中进行连接，可使用 %.snake.net 主机说明符：

```
GRANT ALL ON samp_db.* TO mary@%.snake.net IDENTIFIED BY "topaz"
```

该用户标识符的主机部分可用 IP 地址而不是主机名给出（如果愿意的话）。可以指定一个直接的 IP 地址或包含模式字符的地址。同样，自 MySQL 3.23 起，可以用一个网络掩码来指定 IP 号，网络掩码表明了用于该网络号的二进制位数：

```
GRANT ALL ON samp_db.* TO joe@192.168.128.3 IDENTIFIED BY "water"
GRANT ALL ON samp_db.* TO ardis@192.168.128.% IDENTIFIED BY "snow"
GRANT ALL ON samp_db.* TO rex@192.168.128.0/17 IDENTIFIED BY "ice"
```

第一条语句指明用户可进行连接的特定的主机。第二条语句指定 129.168.128 Class C 子网的 IP 模式。在第三条语句中， 192.168.128.0/17 指定一个 17 位二进制的网络号，并将任何主机与其 IP 地址的前 17 个二进制位中的 192.168.128.0/17 进行匹配。

如果 MySQL 抱怨您指定的用户值，则可能需要使用引号（但对用户名和主机名分别加引号）：

```
GRANT ALL ON samp_db.president TO "my friend"@"boa.snake.net"
```

2. 用户应具有什么级别的权限，这些权限适用于什么

您可授予不同级别的权限。全局权限的功能最强，因为它们适用于任何数据库。为了使 ethel 成为可以进行一切操作的超级用户（其中包括可以对其他用户授权），发布下列语句：

```
GRANT ALL ON *.* TO ethel@localhost IDENTIFIED BY "coffee"
    WITH GRANT OPTION
```

ON 子句中 *.* 说明符的意思是“所有数据库，所有的表”，为保险起见，我们已经指定 ethel 只能从本地主机中连接。限制超级用户从哪些主机上进行连接通常是明智的做法，因为它限制住了其他用户对口令进行试探。

有些权限（FILE、PROCESS、RELOAD 和 SHUTDOWN）是管理权限，只能用 NO *.* 全局权限说明符来授予。如果希望的话，也可以不用授予数据库级的权限来授予这些权限。例如，下列语句建立了一个 flush 用户，它除了发布 FLUSH 语句外不做其他任何事情。在管理脚本中这可能是有用的，因为需要在这些脚本中执行诸如在日志文件循环期间刷新日志的操作：

```
GRANT RELOAD ON *.* TO flush@localhost IDENTIFIED BY "flushpass"
```

通常授予管理权限应该是保守的，因为具有这些权限的用户可能影响服务器的操作。

数据库级的权限适用于特定数据库中的所有表。这些权限使用 ON db_name.* 子句进行授予：

```
GRANT ALL ON samp_db.* TO bill@racer.snake.net IDENTIFIED BY "rock"
GRANT SELECT ON menagerie.* TO ro_user@% IDENTIFIED BY "dirt"
```

第一条语句将 bill 的所有权限授予 samp_db 数据库的任何表。第二条语句创建一个限制访问的用户 ro_user（只读用户），它可以访问 menagerie 数据库的所有表，但只能读取。也就是说，该用户只能发布 SELECT 语句。

怎样在授权表项中指定本地主机名

如果您使用服务器的主机名而非 localhost，通常存在从该服务器主机连接的问题。这可能是由于在授权表中指定名字的方法和名字分解器例程（ name reslover routine）向程序报告名字的方法之间的错误匹配。如果分解器报告了一个非限定的名字（如 pit-viper），但授权表包含了具有全限定的名字的项（如 pit-viper.snake.net，反之亦然），则发生错误匹配。

为了确定这种情况是否正在系统中发生，可试着用 -h 选项连接到本地服务器，该选项指定了主机的名字。然后查看服务器的常规日志文件。它是怎样报告主机名的？是以非限定形式还是限定形式？不论它是哪种形式，都将告诉您在发布 GRANT 语句时需要怎样指定用户说明符的主机名部分。

可以同时列出许多被授予的单个权限。例如，如果想让用户能读取和修改已有表的内容，但又不允许创建新表或删除表，可按如下授权：

```
GRANT SELECT,INSERT,DELETE,UPDATE ON samp_db.* TO jennie@%
    IDENTIFIED BY "boron"
```

对于更小粒度（fine-grained）的访问控制，可以在单个表上授权，甚至在表的单个列上授权。当存在要对用户隐藏的表时，或者，当只允许用户修改特定列时，列专有的权限是有用的。假定历史同盟会中有一些志愿者利用您作为同盟会秘书应履行的职责来帮助您工作。这是一个好消息，但您决定首先给新的助手授予对 member 表只读的权限（该表中包含了会员资格的信息），然后再对他们增加授予该表的 expiration 列的列专有 UPDATE 权限。也就是说，您的助手可以在人们更新其会员资格时进行更改截止日期的工作。设置此 MySQL 用户的语句如下：

```
GRANT SELECT ON samp_db.member
    TO assistant@localhost IDENTIFIED BY "officehelp"
GRANT UPDATE (expiration) ON samp_db.member
    TO assistant@localhost
```

第一条语句授予对整个 member 表的读访问权并设置口令。第二条语句增加 UPDATE 权限，但只是对 expiration 列。此时不必要再指定口令，因为在第一条语句中已经完成了。

如果想要为多个列授予列专有的权限，可指定一个列清单，并用逗号将这些列分隔。例如，为了给 assistant 用户增加对 member 表地址列的 UPDATE 权限，可以使用下列语句。新的权限将被增加到对该用户来说已经存在的所有列上：

```
GRANT UPDATE (street,city,state,zip) ON samp_db.member
    TO assistant@localhost
```

通常，不要给用户授予比实际需要更大的权限。但是，当您想要使用户能够创建存储中间结果的临时表，而又不允许用户在包含有他们不能修改的数据的数据库中这样做时，就有

了要在数据库上授予相当多的许可权限的理由。您可以建立一个单独的数据库（笔者称它为tmp）并授予用户该数据库的所有权限。例如，如果想要 mars.net 域的主机中的任何用户都能够使用 tmp 数据库，可发布下列 GRANT 语句：

```
GRANT ALL ON tmp.* TO ""@%.mars.net
```

在完成这些之后，用户可使用 tmp.tbl_name 格式的名字创建和引用 tmp 数据库中的表（用户说明符中的“ ”创建一个匿名用户项，任何用户都与空白用户名相匹配）。

3. 允许用户管理权限吗

通过授予数据库所有者数据库的所有权限并在操作时指定 WITH GRANT OPTION，可以允许数据库所有者控制对该数据库的访问。例如，如果要让 alicia 能在 big.corp.com 域的所有主机中进行连接并管理 sales 数据库中所有表的权限，应使用下列 GRANT 语句：

```
GRANT ALL ON sales.*
    TO alicia@%.big-corp.com IDENTIFIED BY "applejuice"
    WITH GRANT OPTION
```

实际上，WITH GRANT OPTION 子句允许将访问的权利授予给另一个用户。要知道，具有 GRANT 权限的两个用户可以相互授予自己的权限。如果只给一个用户授予 SELECT 权限而给另一个用户除 SELECT 外还授予了 GRANT 和其他的权限，则第二个用户可以使第一个用户“强大”起来。

11.3.2 取消权限和删除用户

为了收回某个用户的权限，可使用 REVOKE 语句。除了要用 FROM 替换 TO 并且没有 IDENTIFIED BY 或 WITH GRANT OPTION 子句外，REVOKE 的语法与GRANT 语句非常相似：

```
REVOKE privileges (columns) ON what FROM user
```

user 部分必须与您想要取消其权限的用户的原始 GRANT 语句的 user 部分相匹配。*privileges* 部分不需要匹配，您可用 GRANT 语句授权，然后用 REVOKE 语句取消其中的一部分。REVOKE 语句只删除权限，不删除用户。用户的项仍然保留在 user 表中，即使您取消了该用户的所有权限也是如此。这意味着该用户仍然可连接到服务器上。要想删除整个用户，必须用 DELETE 语句将该用户的记录从 user 表中直接删除：

```
% mysql -u root mysql
mysql> DELETE FROM user
    -> WHERE User = "user_name" and Host = "host_name";
mysql> FLUSH PRIVILEGES;
```

DELETE 语句删除该用户的项，FLUSH 语句告诉服务器重新加载授权表（当使用 GRANT 或 REVOKE 语句，而不是直接修改授权表时，这些表将自动重新加载）。

11.4 节将讨论为什么删除 user 表项的原因。

一个权限难题，第一部分

下面是一个在 MySQL 邮件清单中反复出现的情况：一位新的 MySQL 管理员给某用户增加一个项，使用了主机名部分，该部分是用一个模式来指定的。例如：

```
GRANT ALL ON samp_db.* TO fred@%.snake.net IDENTIFIED BY "cocoa"
```

这里的意图是允许用户 fred 从 snake.net 域的所有主机中进行连接，并且具有对

samp_db 数据库的所有权限。事实上，fred 能够从那些主机中连接（除了服务器主机本身外）。当 fred 试着从服务器主机中进行连接时，该企图以“访问被拒绝”的消息而告失败。即使用户指定了正确的口令也是如此。

如果授权表中包含了由 mysql_install_db 安装脚本安装的缺省项，这种情况也会发生。其原因是，当服务器验证 fred 连接的企图时，一个匿名用户项（anonymous-user entry）比 fred 项优先。匿名用户项要求该用户不用口令来连接，并且一个口令错误匹配发生。该问题的另一个背景将在第 12章“权限难题，第二部分”中给出。目前，只要说修正此问题的方法是从 user 表中删除匿名用户项就足够了，我们不能用 REVOKE，因为该命令只删除权限。要想完全摆脱这些匿名项，执行如下操作：

```
% mysql -u root mysql
mysql> DELETE FROM user where User="";
mysql> FLUSH PRIVILEGES;
```

现在，当 fred 试图从本地主机连接时成功了。

11.4 日志文件维护

在 MySQL 服务器启动时，它检查其命令行的操作，来查看它是否应该执行登录并打开相应的日志文件（如果应该的话）。可以让服务器生成两种主要类型的日志文件：

- **常规日志文件。**它报告客户机的连接、查询和其他各种各样的事件。它对于跟踪服务器的活动很有用：谁正在连接、从哪里连接，以及他们正在做什么。
- **更新日志。**它报告修改数据库的查询。在此上下文中的术语“更新”不只涉及 UPDATE 语句，还涉及修改数据库的所有语句。由于这个原因，它包含了对 DELETE、INSERT、REPLACE、CREATE TABLE、DROP TABLE、GRANT 和 REVOKE 的查询记录。更新日志的内容以 SQL 语句的形式书写，这些语句用作对 mysql 的输入。如果在崩溃后必须恢复表的话，更新日志与备份是很有用的。您可以从备份文件中恢复数据库，然后通过将更新日志作为对 mysql 的输入，重新运行在该备份文件之后又修改数据库的任何查询。这样，可将表恢复到崩溃时刻的状态。

为了使日志有效，可使用 --log 选项开启常规日志，并用 --log-update 选项开启更新日志。可以在 mysqld.safe_mysqld 或 mysql.server 的命令行中，或在某个选项的 [mysqld] 组中指定这些选项。当日志有效时，日志文件在缺省时被写到服务器的数据目录中。

笔者建议在首次使用 MySQL 时应使两种日志类型都有效。在获得一些使用 MySQL 的经验后，可能会只用更新日志来对付，以便减少磁盘空间的需求。

在使日志有效后，要确保不用大量的日志信息将磁盘填满，尤其是如果服务器正在处理大量的查询话。可使用日志文件循环和截止时间，在避免日志文件无边界地增长的同时保持最近的几个日志是联机可用的。

日志文件循环工作如下。假定日志文件名为 log。在第一个循环中，log 被重新命名为 log.0，且服务器开始写新的 log文件。在第二次循环中，log.0 被重命名为 log.1，log 重命名为 log.0，服务器开始写另一个新的 log 文件。这样，每个文件循环通过名字 log.0、log.1，等等。当文件到达循环的某一点时，可以终止它。

更新日志和 LOAD DATA 语句

通常，当服务器执行 LOAD DATE 语句时，它只将该语句本身而不是被加载的行内容写到更新日志中。这意味着除非该数据文件仍然保持可访问，否则使用更新日志的恢复操作将是不完整的。为了确保这一点的安全，除非数据库已经备份，否则不应该删除数据文件。

系统备份

更新日志对于数据库恢复并不是任何时候都好，如果一个磁盘崩溃导致您失去了更新日志的话，应确保您执行定期的文件系统备份。将更新日志写到与存储数据库不相同的磁盘中也是一个好主意。有关重新加载日志文件的介绍，请参阅第 10 章的“ MySQL 数据目录 ”。

例如，如果您每天都循环日志，并且想保持一周的日志，则应保留 log.0 到 log.6。在下一个循环中，将通过令 log.5 覆盖 log.6 使其成为新的 log.6 来终止 log.6。这样，您就可以保留许多日志而又避免了它们超过磁盘的限度。

日志循环频率和保持的旧日志数量将依赖于服务器的繁忙程度（活动的服务器产生更多的日志信息）以及您希望为旧日志投入多少磁盘空间。当循环常规日志时，可以用 mysqladmin flush-logs 命令告诉服务器关闭当前的日志文件并打开新的日志文件。

执行常规日志循环的脚本类似如下（可修改它来反映您的日志基名和数据目录的位置，或许还有希望保留的旧日志的数量）：

```
#! /bin/sh

base=basename
datadir=DATADIR

cd $datadir
mv $base.5 $base.6
mv $base.4 $base.5
mv $base.3 $base.4
mv $base.2 $base.3
mv $base.1 $base.2
mv $base.0 $base.1
mv $base $base.0
mysqladmin flush-logs
```

最好从 mysqladm 账号中运行此脚本以确保日志文件属于那个用户。如果在 .my.cnf 选项文件中保留连接参数，您不需要在该脚本的 mysqladmin 命令中指定任何参数。如果您不这样做的话可以建立一个受限用户，它除了发布刷新命令外什么也不做。然后可以以最小的风险在该脚本中放置这个用户的口令。如果想这样做，则该用户应只有 RELOAD 权限。例如，要想调用用户 flush 并分配一个口令 flushpass，可使用下列 GRANT 语句：

```
GRANT RELOAD ON *.* TO flush@localhost IDENTIFIED BY "flushpass"
```

当需要在脚本中执行刷新操作时，可以这样做：

```
mysqladmin -u flush -pflushpass flush-logs
```

在 Linux 中，最好用 logrotate 来安装 MySQL 分发包中的 mysql - log - rotate 脚本，而不是自己编写脚本。如果 mysql-log-rotate 不通过 RPM 文件自动安装，应查看 MySQL 分发包的 support-files 目录。

由于服务器处理更新日志文件的方法不同，日志文件的循环在更新日志与常规日志之间稍有不同。如果告诉服务器使用没有扩展名的更新日志文件名（如 update）,则服务器将使用顺序的 update.001、update.002 等自动创建更新日志文件名。在服务器启动以及在日志刷新时，一个新的更新日志产生。如果您开启更新日志而没有指定文件名，服务器则使用主机名作为基名产生一个更新日志文件的序列。

当终止一个由这种方法生成的文件序列时，您或许想要根据其期限（最后被修改的时间）而非根据名字来终止它们。这样做的理由是由于您不知道 flush-log 命令将在何时发布，因此您不能指望在任何给定的时间周期内创建固定数量的更新日志。例如，如果用 mysqldump 备份表并使用 --flush-logs 选项，在该更新日志名序列中的一个新文件随每个备份一同创建。

对于带有由服务器自动产生的顺序文件名的更新日志，基于日志期限的终止脚本类似如下：

```
#! /bin/sh

find DATADIR -name "update.[0-9]*" -type f -mtime +6 | xargs rm -f
mysqladmin flush-logs
```

find 命令定位并删除修改时间超过一个星期的更新日志文件。重要的是使用 -name 参数来对一个数字的文件扩展名进行测试，以避免删除由错误的 update 所指定的表。

还可以告诉服务器使用固定的更新日志文件名（如果希望的话），如果想用与常规日志相同的方法循环更新日志，这是有用的。要想使用固定的更新日志名，应指定一个包含扩展名的名字。例如，可以用 --log-update=update.log 选项启动服务器来使用名字 update.log。服务器将一直关闭并在接收 flush-logs 命令时打开该日志，但是服务器并不是每次都产生新的文件。在这种情况下，用于更新日志的日志循环脚本和用于常规日志的脚本仅在循环的文件基名上有所不同。

如果想自动执行日志循环和终止，可使用 cron。假定循环常规日志和更新日志的脚本为 rotate-logs 和 rotate-update-logs，且安装在 /usr/user/mysql/bin 目录中。以 mysqlladm 用户进行注册，然后用以下命令编辑 mysqladm 用户的 crontab 文件：

```
% crontab -e
```

此命令允许编辑当前 crontab 文件的备份（如果之前没有这样做，则它可能为空）。按以下方法将行增加到该文件中：

```
0 4 * * * /usr/users/mysqladm/bin/rotate-logs
0 4 * * * /usr/users/mysqladm/bin/rotate-update-logs
```

这个项告诉 cron 在每天早上 4 点运行此脚本。您可以改变时间或按需要进行调度。有关说明请参见 crontab 的人工页。

11.5 备份和拷贝数据库

重要的是在表丢失和毁坏时备份数据库。如果系统发生崩溃，您就能够将表恢复到崩溃时刻的状态，并尽可能不丢失数据。同样，错发 DROP DATABASE 或 DROP TABLE 命令的用户可能会向您请求进行数据恢复。有时，这是由 MySQL 管理员引起的破坏，管理员试图通过使用像 vi 或 emacs 这样的编辑器直接编辑表文件而毁坏了它们。这样做对表来说肯定是干了坏事。

备份数据库的两种主要方法是使用 mysqldump 程序或直接拷贝数据库文件（如便用 cp、

tar 或cpio)。每种方法都有自己的优点和缺点：

- mysqldump 与 MySQL 服务器联合进行操作。直接拷贝方法与服务器相脱离，因此必须采取措施确保在进行拷贝时没有客户机在修改这些表。这个问题与利用文件系统备份来备份数据库的问题相同：如果数据库表在文件系统备份时进行更新，则进行备份的表文件处于非一致的状态，并且对于今后恢复该表没有意义。文件系统备份和直接拷贝文件的区别是：对于后者，您具有控制备份进度的权利，因此可以采取措施确保服务器使表处于静止状态。
- mysqldump 比直接拷贝技术要慢。
- mysqldump 产生可移植到其他机器、甚至具有不同硬件结构的机器上的文本文件。直接拷贝文件不能够移植到其他机器上，除非要拷贝的表使用 MyISAM 存储格式。ISAM 表只能在具有相同硬件结构的机器之间进行拷贝。例如，将文件从 SPARC 的 Solaris 机器拷贝到 Intel 的 Solaris 机器（或者相反）是行不通的。由 MySQL 3.23 引进的 MyISAM 表存储格式可以解决这个问题，因为该格式与机器独立。因此，如果以下两个条件都满足的话，直接拷贝文件可以移植到具有不同硬件结构的机器上：即另一台机器上也必须运行 MySQL 3.23 以上的版本，并且文件必须表示成 MyISAM 表，而不是 ISAM 表。

不论选择哪种备份方法，都有某些原则，您必须坚持这些原则，才能确保在需要恢复数据库内容时得到最好的结果：

- 定期执行备份。设置一个时间表并坚持使用它。
- 告诉服务器运行更新日志。更新日志在您需要恢复崩溃后的数据库时给予帮助。在使用备份文件将数据库恢复到备份时刻的状态后，可以通过运行更新日志中的查询，重新运行备份之后所做的改变。这个操作将数据库中的表恢复到了崩溃时刻的状态。
  在文件系统备份语言中，数据库备份文件表示完全转储（ full dump），而更新日志则表示增量转储。
- 使用一致和可理解的备份文件命名模式。像 backup1、backup2 等名字没有特殊的含义。当需要它执行恢复时，还得浪费时间去查看文件中的内容。您会发现使用数据库名和花时间去构造备份文件名是有好处的。例如：

```
% mysqldump samp_db > /usr/archives/mysql/samp_db.1999-10-02
% mysqldump menagerie > /usr/archives/mysql/menagerie.1999-10-02
```

  在产生备份文件后您可能需要将它们压缩。毕竟备份文件都比较大，所以您可能还需要终止备份文件以避免它们填满磁盘，这与终止日志文件类似。您可以用相同的技术终止备份文件：

- 用文件系统备份来备份您的备份文件。如果您遭受了一个完全崩溃，不仅毁坏了数据目录而且还破坏了包含数据库备份的磁盘驱动器，那将造成真正的麻烦。您还应该备份更新日志。
- 将备份文件放在与您的数据库不同的文件系统上。这将减少含有数据字典的文件系统被生成的备份文件填满的可能性。

创建备份的技术对于将数据库拷贝到另一个服务器上也是很有帮助的。将数据库转移到运行在另一个主机上的服务器是很平常的，但您还可以将数据转移到运行在相同主机上的另一个服务器。如果正为一个新版本的 MySQL 运行服务器，并且想用成品服务器上的某些真

实数据来测试它时，可能会这样做。还有一种可能，那就是您得到了一台新的机器并要将所有的数据库移动到新机器上。

11.5.1 用 mysqldump 备份和拷贝数据库

当使用 mysqldump 程序产生数据库备份文件时，缺省设置是该文件的内容由 CREATE TABLE 语句组成，这些语句创建被转储的表以及包含表中的行数据的 INSERT 语句。换句话说，mysqldump 创建在今后可作为对 mysql 的输入使用的输出结果，以重建数据库。

可以将整个数据库按以下命令转储到单独的文本文件中：

```
% mysqldump samp_db > /usr/archives/mysql/samp_db.1999-10-02
```

该输出文件的开始类似如下：

```
# MySQL dump 6.0
#
# Host: localhost    Database: samp_db
#--------------------------------------------------------
# Server version       3.23.2-alpha-log
#
# Table structure for table 'absence'
#
CREATE TABLE absence (
  student_id int(10) unsigned DEFAULT '0' NOT NULL,
  date date DEFAULT '0000-00-00' NOT NULL,
  PRIMARY KEY (student_id,date)
);
#
# Dumping data for table 'absence'
#
INSERT INTO absence VALUES (3,'1999-09-03');
INSERT INTO absence VALUES (5,'1999-09-03');
INSERT INTO absence VALUES (10,'1999-09-06');
...
```

该文件的其余部分由更多的 INSERT 和 CREATE TABLE 语句组成。

如果想在生成备份时进行压缩，可替换成类似下列的命令：

```
% mysqldump samp_db | gzip > /usr/archives/mysql/samp_db.1999-10-02.gz
```

如果您有一个超大数据库，则该输出文件也将是极大的且管理起来很困难。如果您喜欢的话，可以通过在 mysqldump 命令的数据库名之后命名单个的表来转储这些表的内容。这个操作将该转储文件分成更小的、更多的可管理的文件。下面的例子将说明如何将 samp_db 的一些表转储到单个文件中：

```
% mysqldump samp_db student score event absence > gradebook.sql
% mysqldump samp_db member president > hist-league.sql
```

如果您正在生成备份文件并打算用这些备份文件来定期刷新另一个数据库的内容，则可能要使用 --add-drop-table 选项。此选项告诉 mysqldump 将 DROP TABLE IF EXISTS 语句写到备份文件中。然后，当您取出该备份文件并将其加载到第二个数据库时，如果表已经存在将不会出现错误信息。如果您正在运行第二个数据库，可使用此技术利用从第一个数据库中的数据拷贝来定期地加载它。

如果您正在转储数据库使该数据库可以转换到另一个服务器上，则无须创建备份文件。应确保该数据库存在于另一台主机上，然后用一个管道使 mysql 直接读取 mysqldump 的输出

结果来转储数据库。例如，如果想要将 samp_db 数据库从 pit_viper.snake.net 拷贝到 boa.snake.net，操作如下：

```
% mysqladmin -h boa.snake.net create samp_db
% mysqldump samp_db | mysql -h boa.snake.net samp_db
```

稍后，如果想要在 boa.snake.net 中再次刷新该数据库，可跳过 mysqladmin 命令，但要将 --add-drop-table 增加到 mysqldump 中，以避免得到有关“表已经存在”的错误：

```
% mysqldump --add-drop-table samp_db | mysql -h boa.snake.net samp_db
```

mysqldump 的其他选项包括如下所示的几个：

- --flush-log 和 --lock-tables 的结合有助于检查数据库。--lock-table 锁定所有正在转储的表，而 --flush-log 关闭并重新打开更新日志文件。如果正在产生后续的更新日志，则新的更新日志将只包含从备份的那一点开始修改数据库的查询。这时检查对于该备份时间的更新日志的检查点（然而，锁定所有的表对于备份期间客户机访问来说不太好，如果您有需要执行更新操作的客户机的话）。

  如果用 --flush-logs 检查对于备份时间的更新日志检查点，最好转储整个数据库。如果转储单个文件，则将更新日志的检查点与备份文件同步是比较难的。在恢复操作中，您通常在总数据库（per-database）的基础上抽取更新日志的内容。对于抽取单个表的更新日志来说没有选项，因此您必须自己抽取它们。
- 缺省设置时，mysqldump 将表的全部内容在写之前读到内存中。这实际上不是必须的，事实上，如果您真的有大型表的话，这几乎是一个失败的方法。可以用 --quick 选项告诉 mysqldump 写每一行（只要是被检索的）。要想进一步优化该转储过程，可用 --opt 来代替 --quick。-- opt 选项开启其他的选项，这些选项将加快转储数据和读回数据的速度。

  由于快速备份的好处，使得用 --opt 执行备份成为最常用的方法。但是，要当心，--opt 选项有一个代价：--opt 所优化的是您的备份过程，而不是由其他客户机对数据库的访问。--opt 选项可防止任何人更新被锁定的正在转储的任何表。您会很容易地发现在常规数据库访问中在这一点上所做的努力。试着在一天中数据库通常最繁忙的时刻运行一个备份。这不会花费太多的时间。
- 与 --opt 作用有点相反的选项是 -delayed。该选项导致 mysqldump 写 INSERT DELAYED 语句而非 INSERT 语句。如果您将一个数据文件加载到另一个数据库中并且想要使该操作对其他查询（这些查询可能正在数据库中发生）造成的影响最小，则 --delayed将有助于达到这个目的。
- --compress 选项有助于将数据库拷贝到另一台机器上，因为它可以减少网络传输中的字节数量。这里有一个例子，请注意，为了使程序与远程主机上的服务器进行通信（而不是与本地主机通信），给出了 --compress 选项：

  ```
  % mysqldump --opt samp_db | mysql --compress -h boa.snake.net samp_db
  ```

mysqldump 有许多选项，详细信息请参考附录 E。

11.5.2 使用直接拷贝数据库备份和拷贝方法

不用 mysqldump 来备份数据库或表的另一种方法是直接拷贝表文件。通常可利用像 cp、tar 或 cpio 这样的实用程序来进行。本节的例子使用的是 cp。

当使用直接拷贝备份（direct-copy backup）方法时，必须确保没有使用这些表。如果在拷贝一个表的同时服务器正在修改它，则拷贝无效。

确保拷贝完整性的最好方法是关闭服务器，拷贝文件，然后重新启动服务器。如果不想关闭服务器，则应参考第 13 章，查阅有关在执行表检查点时锁定服务器的介绍。如果服务器在运行中，则相同的约束都适用于拷贝文件，您应该用同样的锁定协议使服务器保持静止状态。

假定服务器关闭，或者已经锁定了想要拷贝的表，下面的例子将说明怎样将整个 samp_db 数据库备份到备份目录中（DATADIR 代表服务器的数据目录）：

```
% cd DATADIR
% cp -r samp_db /usr/archive/mysql
```

单个表可按如下进行拷贝：

```
% cd DATADIR/samp_db
% cp member.* /usr/archive/mysql/samp_db
% cp score.* /usr/archive/mysql/samp_db
...
```

当完成备份时，可以重新启动服务器（如果已使它关闭），或者释放在表上施加的锁（如果保持服务器运行）。

要想用直接拷贝文件将数据库从一台机器拷贝到另一台机器，只要将这些文件拷贝到另一台服务器主机上的相应数据库上即可。应确保这些文件是对 MyISAM 表的或者两台机器都有相同的硬件结构。否则这些表在第二个主机上看起来好象有很奇怪的内容。还应该确保第二台主机的服务器不会在您安装这些表时去访问它们。

11.5.3 复制数据库

术语“复制”的含义简单地说有点像“拷贝数据库到另一个服务器”，或者是包含在主数据库的内容发生变化时次数据库的有效更新（live updating）的含义。如果想简单地将数据库拷贝到另一个服务器上，则可以使用在前面已经讨论的那些命令。自 MySQL 3.23 版本以来，就已经开始出现对基于有效更新的复制的支持。但它的功能仍未成熟，因此，在这方面笔者没有什么可讨论的，如果有兴趣，您可以注意一下当前的新版本，看看有些什么新的开发功能。

11.6 为数据恢复使用备份

数据库毁坏发生的原因有许多，且程度各不相同。如果幸运的话，可能是一两个表的小毁坏（例如，如果您的机器由于断电而暂时停机）。如果不是这样，可能需要置换整个的数据目录（例如，如果某个磁盘瘫痪而且数据目录在它上）。在其他情况下也需要恢复操作，例如，当用户错误地删除数据库或表时，或者错误地删除表的内容时。不论这些不幸的事件发生是由于什么原因，都需要恢复它们。

如果表被毁坏但没有丢失，可试着用 myisamchk 或 isamchk 来修复它们。如果修复实用程序能修复它们，就根本没有必要使用备份文件。表的修复过程将在第 13 章讨论。如果表被丢失或不能修复，则需要恢复它们。

恢复过程包括两个信息源：备份文件和更新日志。备份文件将表恢复到进行该备份时的状态。但是，在备份和故障发生这段时间中，表通常已经被修改。更新日志包含了用来完成

这些修改的查询。可以通过将更新日志作为对 mysql 的输入来重复这些查询（这就是为什么应该允许更新日志的原因。如果您还没有使更新日志有效，现在赶快做，并在进一步读取之前生成一个新的备份）。

恢复过程根据必须恢复的信息的多少而变化。事实上，恢复整个数据库比恢复单个的表要容易，因为对数据库应用更新日志比对表要容易。

11.6.1 恢复整个数据库

首先，如果要恢复的数据库是含有授权表的 mysql 数据库，将需要使用--skip-grant-tables 选项运行服务器。否则，服务器将抱怨无法找到授权表。在恢复表之后，执行 mysqladmin flush-privileges 来告诉服务器加载授权表，并用它们启动。

- 将原数据库目录的内容拷贝到其他的地方。例如，您可能会在稍后用它们进行崩溃表的事后分析检查（post-mortem examination）。
- 用最新的备份文件重新加载数据库。如果您打算使用由 mysqldump 加载的文件，则需要将它们作为 mysql 的输入。如果打算使用从数据库中直接拷贝的文件（如，用 tar 或 cp），则将它们直接拷贝回到该数据库目录中。但是，在这种情况下，应该在拷贝这些文件之前关闭服务器，然后再重新启动它。
- 用更新日志重做在进行备份后又修改了数据库表的查询。对于所有可用的更新日志，可使用它作为 mysql 的输入。指定 --one-database 选项，使 mysql 只对想要恢复的数据库执行查询。如果您知道需要使用所有的更新日志文件，可在包含日志的目录中使用下列命令：

```
% ls -t -r -1 update.[0-9]* | xargs cat | mysql --one-database db_name
```

ls 命令产生更新日志文件的单列列表，更新日志文件根据服务器生成的顺序进行排序（要知道，如果您修改了其中的任何文件，排序的顺序都将改变，这将导致更新日志按错误的顺序使用）。

您很可能必须使用某些更新日志。例如，如果自备份以来所产生的日志命名为 update.392、update.393 等等，可以重新运行它们中的命令：

```
% mysql --one-database db_name < update.392
% mysql --one-database db_name < update.393
...
```

如果正在运行恢复并打算使用更新日志恢复由于失策的 DROP DATABASE、DROP TABLE 或 DELETE 语句而丢失的信息，应确保先从更新日志中删除这些语句。

11.6.2 恢复单个的表

恢复单个表是很困难的。如果有通过 mysqldump 生成的备份文件并且它恰好不包含您想要的表数据，则需要抽取相关的行并用它们作为 mysql 的输入，这部分较容易。困难的是抽取应用于该表的更新日志的片段。您会发现：mysql_find_rows 实用程序对这方面有帮助，它可以从更新日志中抽取多行查询。

另一种可能性是用另一个服务器恢复整个数据库，然后将所要的该表的文件拷贝到原始数据库中。这实际很容易！在将文件拷贝回数据库目录时，应确保原始数据库的服务器关闭。

11.7 优化服务器

MySQL 服务器有几个影响其操作的参数（变量）。如果缺省的参数值不合适，可以将其修改为对服务器运行环境更合适的值。例如，如果您有大量的内存，可以告诉服务为磁盘和索引操作使用较大的缓冲区。这将使内存持有更多的信息并减少了必须进行的磁盘访问的数量。如果是一般的系统，可以告诉服务器使用较小的缓冲区，以防止它扰乱系统资源损害其他的进程。

系统变量的当前值可以通过执行 mysqladmin variables 命令来检查。变量可利用 --set-variable *var_name* = *value* 选项在命令行设置（-o *var_name* = *value* 是等价的）。如果要想设置几个变量，可使用多个 --set-variable 选项，还可以使用下列语法在一个选项文件的 [mysqld] 组中设置变量：

```
set-variable=var_name=value
```

在附录E的 mysql 程序的条款下给出了服务器变量的全部清单。有关性能优化比较常用的变量已在以下列表中给出。您还可以在 MySQL 参考手册的“从 MySQL 中获得最高性能”一章中找到该主题的附加讨论。

- back_log 引入的客户机连接请求的数量，这些请求在从当前客户机中处理时排队。如果您有一个很忙的站点，可以增加该变量的值。
- delayed_queue_size 此变量控制被排队的 INSERT DELAYED 语句中的行数。如果该队列已满，则更多的 INSERT DELAYED 将堵塞，直到队列有空间为止，这样可防止发布那些语句的客户机继续进行操作。如果您有许多执行这种 INSERT 的客户机，且发现它们正在堵塞，那么应增加该变量，使更多的客户机更快地进行工作（ INSERT DELAYED 在4.5节“调度与锁定问题”中讨论）。
- flush_time 如果系统有问题并且经常锁死或重新引导，应将该变量设置为非零值，这将导致服务器按 flush_time 秒来刷新表的高速缓存。用这种方法来写出对表的修改将降低性能，但可减少表讹误或数据丢失的机会。
  在 Windows 中，可以在命令行上用 --flush 选项启动服务器，以迫使表的修改在每次更新后被刷新。
- key_buffer_size 用于存放索引块缓冲区的大小。如果增加该变量值，将加快创建和修改索引操作的时间。值越大MySQL 就越有可能在内存中查找键值，这将减少索引处理所需的磁盘访问次数。
  在 MySQL 3.23 以前的版本中，该变量名为 key_buffer。MySQL 3.23 及后来的版本同时识别这两个名字。
- max_allowed_packet 客户机通信所使用的缓冲区大小的最大值。如果有客户机要发送大量的 BLOB 或 TEXT 的值，该服务器变量值可能需要增大。
  客户机目前使用大小为 24MB 的缺省缓冲区。如果有使用较小缓冲区的旧客户机。可能需要使该客户机的缓冲区大一些。例如， mysql 可以按如下调用来指定一个 24MB 信息包的限制值：

```
mysql --set-variable max_allowed_packet=24M
```

- max_connections 服务器允许的客户机同时连接的最大数量。如果服务器繁忙，可能

需要增加该值。例如，如果您的 MySQL 服务器被 Web 服务器使用来处理由 DBI 或 PHP 脚本产生的查询，并且还有大量的 Web 通信，如果该变量设置太低的话，则您站点的访问者会发现请求被拒绝。

- table_cache 表的高速缓存的大小。增加该值可以使 mysqld 保持更多的表，同时打开并减少必须进行的文件打开和关闭操作的次数。

如果增加了 max_connections 或 table_cache 值的大小，服务器将需要大量的文件描述符。这将引起有关操作系统对文件描述符总进程数量限定的问题，在这种情况下您需要增加该限制值或逐步解决它。由于增加文件描述符数量的限制值，过程会发生变化，所以您可能会在一个运行脚本中使用 ulimit 命令时来这样做，该脚本可用于启动服务器，或用于重新配置您的系统。有些系统可以通过编辑系统描述文件来简单地配置和重新引导。对于其他一些系统，则必须编辑一个内核描述文件并重建该内核。如何继续进行下去，请参考您系统的文档。

解决总进程文件描述符限制的一个方法是：将数据目录分离成多个数据目录并运行多个服务器。这样，通过运行多个服务器使可用的描述符数量成倍增长。但另一方面，其他的复杂因素可能会引起问题。由于命名了两个服务器，您不能从一个单个的服务器上访问不同数据目录中的数据库，并且还需要在不同服务器之间复制授权表的权限，以便用户需要访问一个以上的服务器。

有两个变量是管理员为提高性能时常增加的，它们是 record_buffer 和 sort_buffer。这些缓冲区在连接和分类操作中使用，但其值是属于每个连接的。也就是说，每个客户机都获得属于自己的缓冲区。如果使这些变量的值很大，性能可能会由于昂贵的系统资源的消耗而遭受实际的损失。如果想要修改这些变量，先执行 mysqladmin variables 查看一下它们当前的值，然后增量调整其值。这个操作使您能估计为减少严重的性能降低所进行的修改的效果。

11.8 运行多个服务器

大多数人们都在指定的机器上运行单个 MySQL 服务器，但在有些情况下，运行多个服务器是有好处的：

- 您可以在成品服务器运行的状态下测试新版本的服务器。在这种情况下，将运行不同的服务器。
- 操作系统通常将总进程限制施加于打开文件的描述符数量上。如果系统提高该限制值有困难，则运行多个服务器是解决该限制的办法（例如，提高限制可能需要重新编译该内核，并且，如果该机器不是您的管辖范围，可能还不能这样做）。在这种情况中，您可能会运行相同服务器的多个实例。
- 互联网服务经常提供给顾客他们自己的 MySQL 安装程序，它需要单独的服务器。在这种情况下，您可能会运行相同服务器或不同服务器的多个实例，如果不同的顾客需要不同 MySQL 版本的话。

当然，同时运行几个服务器比只运行一个更复杂。如果您打算安装多个版本，则不能将它们都安装在相同的位置。当服务器运行时，某些参数对于每个服务器必须是唯一的。其中有些参数包括在服务器安装的地方：数据目录的路径名、 TCP/IP 端口和 UNIX 域的套接字路径名，以及用于运行多个服务器的 UNIX 账号（如果不在相同的账号下运行所有的服务器的话）。如果决定运行多个服务器，则必须保持对正在使用的参数有良好说明，

以便不失去对运行的跟踪。

11.8.1 配置和安装多个服务器

如果打算运行不同版本的服务器而非相同服务器的多个实例，应该在不同的位置安装它们。如果安装二进制（不是 RPM）分发包，它们将在包含版本号的目录名下被安装。如果从源程序分发包中安装，使不同的分发包分开的最容易的办法是，当在每个版本的 MySQL 安装过程中运行 configure 时使用 --with-prefix 选项。这将导致在单独的目录下进行安装，并且可以连接该目录到该分发包的版本号上。例如，您可以按如下配置一个 MySQL 分发包，这里的 *version* 是 MySQL 的版本号：

```
% ./configure --with-prefix=/usr/local/mysql-version
```

-- with-prefix 选项还将确定服务器的唯一数据目录。您可能会增加选项来配置其他的服务器专有的值，如 TCP/IP 端口和套接字路径名（--with-tcp-port 和 --with-unix-socket）。

如果打算运行相同服务器的多个实例，则服务器专用的任何选项将需要在运行时指定。

11.8.2 多个服务器的启动过程

启动多个服务器比使用单个服务器复杂，因为 safe_mysqld 和 mysql.server 二者均在单个服务器上工作良好。笔者建议您仔细研究 safe_mysqld 并用它作为启动过程的基础，除非您使用按自己的需求修改得更精细的拷贝。

必须处理的一个问题是如何在选项文件中指定选项。对于多个服务器，不能对在总服务器基础上变化了的设置使用 /etc/my.cnf，只能为所有服务器都相同的设置使用该文件。如果每个服务器都有不同的数据目录位置，可以在每个服务器数据目录的 my.cnf 文件中指定服务器专有的参数。换句话说，应为所有服务器都使用的设置使用 /etc/my.cnf，并且为服务器专有的设置使用 *DATADIR*/my.cnf，这里的 *DATADIR* 随每个服务器变化。

指定服务器选项的另一种方法是：使用 --defaults-file=*path_name* 作为该命令行的第一个选项，以便告诉服务器从 *path_name* 指定的文件中读取选项。这样，可以将该文件中的服务器选项唯一地放置到那个服务器中，然后告诉服务器在启动时读取该文件。请注意，如果指定该选项，则没有任何通常的选项文件（如 /etc/my.cnf）将被使用。

11.9 更新 MySQL

MySQL 首次公开发行的是 3.11.1 版。目前流行的是 3.22 系列稳定版的发行版和 3.23 系列开发版的发行版。稳定版系列号总是比开发版系列的要小。在 3.23 稳定之后，将开始使用 3.24 作为开发系列。MySQL 开发者好像在夜以继日地工作着，版本更新相当频繁（一年有几次）。稳定版和开发版都发布更新版本。正在进行开发的速度给 MySQL 管理员带来了这样的问题：当新版本出现时是否应该对您已有的 MySQL 安装进行升级。本节提供了帮助您作出该项决定的一些指导。

在新版本出现时应该做的第一件事是查找它与前一个版本之间有什么不同。检查 MySQL 参考手册中的附录“变化说明”，使自己了解这些差别。然后给自己提出下列问题：

- 您在当前版本中遇到过新版本修正了的问题吗？
- 新版本中有您想要的附加功能吗？

- 对于您使用的某些类型的操作其性能提高了吗？

如果对所有这些问题的回答都是否定的，则没有任何强制升级的必要。如果所有的回答是肯定的，可能要继续进行下一步工作。这时，一般应等待几天，观察 MySQL 邮件清单，看看是否有关于其他人使用新版本的报道。

可帮助您做出决定的某些要考虑的其他因素如下：

- 稳定版系列的发行版通常都是对已有的错误进行修正，很少有新功能。通常在稳定版系列中升级的风险要比在开发版系列中的小（当然，如果正在运行开发版系列的服务器，可能根本不关心这个风险）。
- 如果对 MySQL 进行升级，可能还要对用其内部的 MySQL C 客户机库建立的其他程序进行升级。例如，在 MySQL 升级之后，还需要重新建立 PHP、Apache 和 Perl DBD::mysql 模块，将新的客户机库连接到那些程序中（当所有 MySQL 相关的 DBI 和 PHP 脚本在您升级 MySQL后开始转储核心时需要这样做）。重建这些程序通常不是什么了不起的事，但是，如果想避开它，最好别对 MySQL 进行。如果您使用静态连接程序而非动态连接程序，该问题的可能性将明显减少。但是，系统内存的需求将增加。

如果仍然不能肯定是否升级，则独立于当前的服务器来测试新的服务器总是可以的。可以通过或者将新服务器与成品服务器并行运行来测试，或者在另一台机器上安装新服务器进行测试。如果使用不同的机器，容易在服务器之间进行独立的维护。如果没有另一台用于测试的机器，可以在成品机上运行新的服务器。如果这样做，则必须用不同的参数值（如安装位置、数据目录以及服务器可以舰艇连接的网络端口和套接字）来运行新的服务器。

无论是哪种情况，您都可能会用已有数据库中的数据拷贝来测试新服务器。

如果作出了升级决定，应查看在 MySQL R 参考手册的“变化说明”附录中是否有关于升级必须采取的特殊步骤的说明。通常不会有，但总之最好检查一下。

不要害怕使用开发版的发行版

您可能不喜欢利用自己的成品数据库来使用开发的发行版，但是笔者鼓励您至少应试着用一下单独的测试服务器，或许再使用一下您的成品数据库备份。试用新的发行版的人越多，对发现错误就越有帮助。对于某些数据库产品，发现错误是一件可怕的事情。而对于 MySQL 来说，错误的报告是促进开发前进的重要因素，因为开发人员可根据用户团体所报告的问题进行实际修正。