第12章　MySQL高级管理

在本章中，我们将讨论关于MySQL的一些更高级话题，其中包括高级权限、安全和优化等问题。

本章将讨论的主要内容包括：

- 深入理解权限系统
- 提高数据库的安全性
- 获取关于数据库的更多信息
- 使用索引提高查询速度
- 优化数据库
- 备份和恢复
- 实现复制

12.1　深入理解权限系统

在第9章，我们学习了如何创建用户，以及使用GRANT命令实现用户权限的授予。如果要管理一个MySQL数据库，那么准确地理解GRANT命令的作用及其工作原理是非常有意义的。

当执行一条GRANT语句的时候，它会影响一个名为mysql的特殊数据库。权限信息就是保存在这个数据库的6个表中。这样，当对数据库授予权限的时候，就应该注意是否授予了访问mysql数据库的访问权限。

以管理员身份登录后，输入如下命令，我们就可以看到mysql数据库的内容：

```
use mysql;
```

执行上述命令后，如果想查看该数据库中的表，可以输入如下命令：

```
show tables;
```

上述命令的执行结果如下所示：

```
+---------------------------+
| Tables_in_mysql           |
+---------------------------+
| columns_priv              |
| db                        |
| event                     |
| func                      |
| general_log               |
| help_category             |
| help_keyword              |
| help_relation             |
| help_topic                |
```

```
| host                      |
| ndb_binlog_index          |
| plugin                    |
| proc                      |
| procs_priv                |
| servers                   |
| slow_log                  |
| tables_priv               |
| time_zone                 |
| time_zone_leap_second     |
| time_zone_name            |
| time_zone_transition      |
| time_zone_transition_type |
| user                      |
+---------------------------+
```

以上每个表都存储了关于系统的信息。保存权限信息的6个表分别是：user、host、db、tables_priv和columns_priv以及procs_priv。有时，这些表也称为*授权表*（grant tables）。

这些表的具体用途各不相同，但它们的基本用途都是相同的，都可以用来确定用户身份以及是否允许执行特定操作。这些表都包含两种类型的字段：范围字段，它可以指定用户、主机和数据库的一部分；权限字段，它可以指定该用户在该范围内可以执行哪些操作。

user表和host表用于确定一个用户是否可以连接MySQL服务器，以及该用户是否具有任何管理员权限。db表和host表确定用户可以访问哪些数据库。tables_priv表确定用户可以使用数据库中哪些表，而columns_priv表确定用户可以访问表中的哪些列，procs_priv表确定用户可以执行哪些过程。

12.1.1 user表

该表包含全局用户权限的详细信息。它可以确定一个用户是否有权连接MySQL数据库，是否具有全局级别的权限；即是否有使用系统中每个数据库的权限。

我们可以通过执行describe user;语句来浏览该表的结构。user表的模式如表12-1所示。

表12-1 mysql数据库中user表的模式

字 段	类 型
Host	varchar(60)
User	varchar(16)
Password	varchar(41)
Select_priv	enum('N', 'Y')
Insert_priv	enum('N', 'Y')
Update_priv	enum('N', 'Y')
Delete_priv	enum('N', 'Y')
Create_priv	enum('N', 'Y')
Drop_priv	enum('N', 'Y')
Reload_priv	enum('N', 'Y')

（续）

字　段	类　型
Shutdown_priv	enum('N', 'Y')
Process_priv	enum('N', 'Y')
File_priv	enum('N', 'Y')
Grant_priv	enum('N', 'Y')
References_priv	enum('N', 'Y')
Index_priv	enum('N', 'Y')
Alter_priv	enum('N', 'Y')
Show_db_priv	enum('N', 'Y')
Super_priv	enum('N', 'Y')
Create_tmp_table_priv	enum('N', 'Y')
Lock_tables_priv	enum('N', 'Y')
Execute_priv	enum('N', 'Y')
Repl_slave_priv	enum('N', 'Y')
Repl_client_priv	enum('N', 'Y')
Create_view_priv	enum('N', 'Y')
Show_view_priv	enum('N', 'Y')
Create_routine_priv	enum('N', 'Y')
Alter_routine_priv	enum('N', 'Y')
Create_user_priv	enum('N', 'Y')
Event_priv	enum('N', 'Y')
Trigger_priv	enum('N', 'Y')
ssl_type	enum('', 'ANY','X509','SPECIFIED')
ssl_cipher	blob
x509_issuer	blob
x509_subject	blob max_questions　　int(11) unsigned
max_updates	int(11) unsigned
max_connections	int(11) unsigned
max_user_connections	int(11) unsigned

上表中的每一行对应于一组用户权限，该用户来自一个主机并用密码Password进行登录。

这些都是该表的*范围字段*（scope field），因为它们可以描述其他字段的范围，其他字段则称为*权限字段*（privilege field）。

该表（以及其他接下来的表）中列出的权限对应于我们在第9章中用GRANT语句授予的权限。例如，Select_priv对应于运行SELECT命令的权限。

如果一个用户具有特定的权限，该列中的值就应该是Y。相反，如果用户没有被授予该权限，其值就应该是N。

user表中列出的所有权限都是全局的，也就是说，它们适用于系统中（包括mysql数据库）*所有数据库*。因此，在这里，管理员的一些权限为Y，但是大部分用户应该全部为N。普通用户应该有一些访问适当数据库但不是所有表的权限。

12.1.2　db表和host表

普通用户的权限大多数都保存在db表和host表中。

db表可以确定哪些用户可以访问哪些主机和哪些数据库。此表中给出的权限应用于在一个特定行中指定的任意数据库。

host表是db表的补充。如果一个用户从多个主机连接到数据库，在db表中将不会列出该用户的主机名。而与此项对应，该用户将在host表中获得一组记录，每个记录都指定了每个用户-主机对的权限。

这两个表的模式分别如表12-2和表12-3所示。

表12-2 mysql数据库中db表的模式

字 段	类 型	字 段	类 型
Host	char(60)	Index_priv	enum('N', 'Y')
Db	char(64)	Alter_priv	enum('N', 'Y')
User	char(16)	Create_tmp_tables_priv	enum('N', 'Y')
Select_priv	enum('N', 'Y')	Lock_tables_priv	enum('N', 'Y')
Insert_priv	enum('N', 'Y')	Create_view_priv	enum('N', 'Y')
Update_priv	enum('N', 'Y')	Show_view_priv	enum('N', 'Y')
Delete_priv	enum('N', 'Y')	Create_routine_priv	enum('N', 'Y')
Create_priv	enum('N', 'Y')	Alter_routine_priv	enum('N', 'Y')
Drop_priv	enum('N', 'Y')	Execute_priv	enum('N', 'Y')
Grant_priv	enum('N', 'Y')	Event_priv	enum('N', 'Y')
References_priv	enum('N', 'Y')	Trigger_priv	enum('N', 'Y')

表12-3 mysql数据库中host表的模式

字 段	类 型	字 段	类 型
Host	char(60)	References_priv	enum('N', 'Y')
Db	char(64)	Index_priv	enum('N', 'Y')
Select_priv	enum('N', 'Y')	Alter_priv	enum('N', 'Y')
Insert_priv	enum('N', 'Y')	Create_tmp_tables_priv	enum('N', 'Y')
Update_priv	enum('N', 'Y')	Lock_tables_priv	enum('N', 'Y')
Delete_priv	enum('N', 'Y')	Create_view_priv	enum('N', 'Y')
Create_priv	enum('N', 'Y')	Show_view_priv	enum('N', 'Y')
Create_view_priv	enum('N', 'Y')	Create_routine_priv	enum('N', 'Y')
Show_view_priv	enum('N', 'Y')	Alter_routine_priv	enum('N', 'Y')
Create_routine_priv	enum('N', 'Y')	Execute_priv	enum('N', 'Y')
Drop_priv	enum('N', 'Y')	Trigger_priv	enum('N', 'Y')
Grant_priv	enum('N', 'Y')		

12.1.3 tables_priv表，columns_priv表和procs_priv表

这三个表分别用于存储表级别的权限和列级别以及针对存储过程的权限。这与db表类似，但是它们分别为指定数据库中的表授予权限和为指定表中的列授予权限。

这些表的结构与user、db和host表结构有所不同。tables_priv表、columns_priv表和procs_priv表的模式分别如表12-4、表12-5和表12-6所示。

tables_priv表的Grantor列用来存储被授予权限的用户。两个表中的Timestamp列用来存储授权日期与时间。

表12-4　mysql数据库中tables_priv表的模式

字　段	类　型
Host	char(60)
Db	char(64)
User	char(16)
Table_name	char(60)
Grantor	char(77)
Timestamp	timestamp(14)
Table_priv	set('Select', 'Insert', 'Update', 'Delete', 'Create', 'Drop', 'Grant', 'References', 'Index', 'Alter', 'Create View', 'Show view', 'Trigger'))
Column_priv	set('Select', 'Insert', 'Update', 'References')

表12-5　mysql数据库中columns_priv表的模式

字　段	类　型
Host	char (60)
Db	char (64)
User	char (16)
Table_name	char (64)
Column_name	char (64)
Timestamp	timestamp (14)
Column_priv	set('Select', 'Insert', 'Update', 'References')

表12-6　mysql数据库中procs_priv表的模式

字　段	类　型
Host	char(60)
Db	char(64)
User	char(16)
Routine_name	char(64)
Routine_type	enum('FUNCTION', 'PROCEDURE')
Grantor	char(77)
Proc_priv	set('Execute','Alter Routine','Grant')
Timestamp	timestamp(14)

12.1.4　访问控制：MySQL如何使用grant表

MySQL使用grant表通过两个步骤来确定允许用户做什么：

1) 连接验证。在这里，MySQL基于如上所示的user表中的信息检查用户是否有权连接数据库。这是基于用户名、主机名和密码进行的验证。如果用户名为空，它将匹配所有用户。主机名可以用通配符%指定。通配符%可以用作整个主机名（也就是说，“%”符号匹配所有的主

机），或者用作主机名的一部分，例如，%.tangledweb.com.au匹配所有以.tangledweb.com.au结尾的主机。如果密码字段为空，则不要求密码。在主机名称中避免使用通配符，避免使用没有密码的用户名，避免空用户名，这样做会更安全。如果主机名为空，MySQL将引用host表，找到匹配的user和host对。

2) 请求验证。当建立一个连接之后，对于所发送的每一个请求，MySQL都会检查是否有执行该请求的权限级别。系统首先将检查全局权限（在user表中），如果这些还不够，系统将再检查db表和host表。如果仍然没有足够的权限，MySQL将检查tables_priv表，如果权限还不够，最后将检查columns_priv表。

12.1.5 更新权限：修改什么时候生效

MySQL服务器启动的时候以及使用GRANT和REVOKE语句的时候，服务器会自动读取grant表。但是，既然我们知道这些权限保存在什么地方以及它们是如何保存的，就可以手动修改它们。当手动更新它们的时候，MySQL服务器将不会注意到它们已经被修改了。

我们必须向服务器指出已经对权限进行了修改，有3种方法可以实现了这个任务。可以在MySQL提示符下（必须以管理员的身份登录进入）输入：

```
flush privileges;
```

这是更新权限最常使用的方法。

或者，还可以在操作系统中运行：

```
mysqladmin flush-privileges
```

或者

```
mysqladmin reload
```

此后，当用户下次再连接的时候，系统将检查全局级别权限；当下一个命令被执行时，将检查数据库级别的权限；而表级别和列级别权限将在用户下次请求的时候被检查。

12.2 提高MySQL数据库的安全性

安全性是非常重要的，特别是在开始连接MySQL数据库到网站的时候。在本节中，我们将讨论保护数据库应该采取的预防措施。

12.2.1 从操作系统角度来保护MySQL

如果正在运行类似于UNIX的操作系统，那么以root用户的身份运行MySQL服务器（mysqld）是一个糟糕的主意。因为，这可能赋予了一个MySQL普通用户读写操作系统任何地方的文件的权限。这一点是非常重要的，但是又容易被忽略，它是曾经用来攻击Apache网站的某种著名方法。（幸运的是，攻击该网站的黑客是“白帽子”（好人），他们采取的唯一行动是加强安全）。

创建一个专门用来运行mysqld的特定MySQL用户是一个好主意。此外，还可建立只能够由MySQL用户访问的目录（保存物理数据的地方）。在许多安装方式中，服务器都是设置为以MySQL组中的userid为mysql的用户来运行服务器的。

然而，更理想的办法是，应该将MySQL服务器建立于防火墙后。这样，MySQL服务器就可以终止来自未授权机器上的连接；检查一下是否可从服务器之外，以端口号为3306的方式连接MySQL。这是MySQL运行的默认端口，可以在防火墙中关闭它。

12.2.2 密码

必须确认所有用户都有密码（特别是root用户！）并且要选好这些密码，定期修改，与使用操作系统密码一样。这里要记住的基本原则是不要使用字典的单词作为密码，用字典的单词做密码是非常糟糕的想法。数字和字母的结合是最好的方案。

如果要在脚本文件中保存密码，请确认只有密码保存在该脚本中的用户可以阅读该脚本。用来连接数据库的PHP脚本需要能够访问的那个用户的密码。将登录名和密码保存在一个PHP脚本可能是安全的，例如dbconnect.php，这样可以在需要的时候包含这个文件。这个脚本必须小心地保存在Web文档树结构以外，并且只能由特定的用户进行访问。

请记住，如果要将这些细节信息保存在网络文档树中以.inc或其他一些扩展名结尾的文件中，请务必小心，检查Web服务器是否知道这一点，也就是说，这些文件必须解释为PHP，从而防止这些细节在另一个Web浏览器中被看到。

不要以纯文本的形式将密码保存于数据库中。MySQL密码不是这样保存的，但是在Web应用程序中通常也要保存网站用户的注册名和密码。可以用MySQL的SHA1()函数将密码加密以后（单向）再进行保存。请注意，如果在运行SELECT（试图登录一个用户）的时候，以这些格式之一使用INSERT插入一个密码到，必须再次使用相同的函数来检查用户输入的密码。

在本书的第五篇中，我们将使用这个功能来实现我们的项目。

12.2.3 用户权限

俗话说，知识就是力量。请确认你已经理解了MySQL的权限系统，以及授予特定权限的后果。请不要授予任何用户不必要的权限。应该查看grant表来确认这一点。

特别地，如果不是绝对需要，请不要将PROCESS、FILE、SHUTDOWN和RELOAD等权限授予任何非管理员的用户。PROCESS权限可用于观察其他用户正在做什么，输入了什么，包括他们输入的密码。FILE权限可以用来读写操作系统中的文件（例如，包括UNIX系统中的/etc/password）。

GRANT权限也要在授予的时候非常小心，因为它准许用户将他们的权限分享给其他用户。

请确认在建立用户的时候，只授予他们从当前连接的主机访问数据库的权限。如果有一个名为jane@localhost的用户，这没有问题，但是简单的jane是非常常见的，她可能从任何地方进行登录——这个jane可能不是你想象中的jane。同样的原因，我们应该避免在主机名中使用通配符。

可以通过在host表中使用IP地址而不是域名来提高安全性能。这可以在DNS位置避免错误问题或者黑客侵入。可以通过启动MySQL后台程序--skip-name-resolve选项加强它，该选项的意思是所有主机列的值必须是IP地址或本地主机。

此外，还应该防止非管理员用户访问Web服务器中的mysqladmin程序。因为这是从命令行

运行的，它是操作系统权限的问题。

12.2.4 Web问题

当连接MySQL数据库到网络的时候，一些特殊的安全问题就出现了。

建立一个专门用于网络连接的用户并不是一个坏主意。这样可以授予他们必要的、最少的权限，从而不给用户授予以下一些权限，例如DROP、ALTER或CREATE权限。我们可能只在类型表中授予SELECT权限，而只在订单表中授予INSERT权限。此外，这还是如何应用最少权限原则的例证。

提示 我们在上一章讨论了如何使用PHP的addslashes()函数和stripslashes()函数以去掉任何可能产生问题的字符。记住这样做和在将任何数据发送到MySQL数据库之前要执行一个常规的数据清理都是非常重要的。你可能记得，我们曾经使用了doubleval()函数来检查数字类型的数据是否真正是数字类型的数据。我们经常容易犯的一个错误就是会忘记它——人们往往记得使用addslashes()但是忘记检查数字型数据。

应该经常检查来自用户的所有数据。即使HTML表单中由选项框和按钮组成，一些人还是可能通过企图修改URL以进入脚本。此外，还应该检查用户数据的大小。

如果用户输入的密码或机密数据需要保存在数据库中，请注意，如果不使用SSL（Secure Sockets Layer，加密套接字层），这些数据将以纯文本的方式从浏览器传递到服务器。我们将在后面详细讨论SSL的应用。

12.3 获取更多关于数据库的信息

迄今为止，我们已经使用了SHOW和DESCRIBE来查询数据库的表和表中的数据行。下面，我们将简要了解它们是如何在其他方面应用的，以及使用EXPLAIN语句获取SELECT执行过程的详细信息。

12.3.1 使用SHOW获取信息

前面，我们已经使用了：

```
show tables;
```

语句来获取数据库中所有表。

如下语句：

```
show databases;
```

可以显示所有数据库。还可使用SHOW TABLES语句查看其中一个数据库的表：

```
show tables from books;
```

当使用SHOW TABLES的时候，如果没有指定特定的数据库，默认数据库为当前使用的数据库。

当知道数据库中都有哪些表的时候，可以获取该表的所有列：

```
show columns from orders from books;
```

如果没有给出数据库参数，在默认情况下，SHOW COLUMNS语句所操作的数据库是当前使用的数据库。此外，也可以使用table.column符号：

```
show columns from books.orders;
```

SHOW语句另一个非常有用的变化是查看用户所拥有的权限。例如，如果运行下列语句：

```
show grants for bookorama;
```

将得到如下所示的结果：

```
+-----------------------------------------------------------------------+
| Grants for bookorama@%                                                |
+-----------------------------------------------------------------------+
| GRANT USAGE ON *.* TO 'bookorama'@'%'                                 |
| IDENTIFIED BY PASSWORD '*1ECE648641438A28E1910D0D7403C5EE9E8B0A85'    |
| GRANT SELECT, INSERT, UPDATE, DELETE, CREATE, DROP, INDEX, ALTER      |
| ON `books`.* TO 'bookorama'@'%'                                       |
+-----------------------------------------------------------------------+
```

所示的GRANT语句未必是被执行并授予特定用户权限的所必需的语句，而更像是产生用户当前级别权限的等价语句。

此外，还可以使用SHOW语句的其他变体。事实上，SHOW语句总共有30多种变体。最常见的变体语句如表12-7所示。请参阅MySQL手册获得完整列表：http://dev.mysql.com/doc/refnan/5.1/show.html。在如下示例的[kike_or_where]例子中，你可以使用LIKE或WHERE表示式进行模式匹配。

表12-7　SHOW语句的语法

变化形式	描　述
SHOW DATABASES [like_or_where]	列出所有可供使用的数据库
SHOW [OPEN] TABLES [FROM database] [like_or_where]	列出当前（或名为database）数据库所使用的表
SHOW [FULL] COLUMNS FROM *table* [FROM *database*] [like_or_where]	列出当前被使用数据库（或者指定的数据库）中特定表的所有列。可以使用SHOW FIELDS语句代替SHOW COLUMNS
SHOW INDEX FROM *table* [FROM *database*]	显示当前被使用或指定（名为database）的数据库中特定表的所有索引。也可以使用SHOW KEYS语句
SHOW [GLOBAL | SESSION] STATUS [like_or_where]	给出系统项的信息，例如当前运行的线程数目。LIKE子句用来匹配这些项名称，例如，'Thread%' 用来匹配 'Threads_cached' 和 'Thread_connected'，'Threads created' 以及 'Threads running'
SHOW [GLOBAL SESSION] VARIABLES [like_or_where]	显示MySQL系统变量名称和值，例如版本号
SHOW [FULL] PROCESSLIST	显示系统当前运行的所有进程——也就是，当前执行的查询数。大多数用户可以看到他们自己的线程，但是如果他们具有PROCESS权限，他们可以看到每个人的进程——包括他们的密码。在默认情况下，这些查询将截断为100个字符。使用可选的关键字FULL可以显示完整的查询

（续）

变化形式	描 述
SHOW TABLE STATUS [FROM *database*] [like_or_where]	显示当前被使用（或名为database）的数据库所有表的信息，指定的数据库名称也可以具有通配符。该信息包括表类型以及每个表最后被更新的时间
SHOW GRANTS FOR *user*	显示要使用户具有当前权限级别的所需GRANT语句
SHOW PRIVILEGES	显示服务器所支持的不同权限
SHOW CREATE DATABASE *database*	显示可以创建指定数据库的CREATE DATABASE语句
SHOW CREATE TABLE *tablename*	显示可以创建指定表的 CREATE TABLE
SHOW [STORAGE] ENGINES	显示当前数据库安装中可供使用的默认存储引擎。（已经在第13章中介绍了存储引擎）
SHOW INNODB STATUS	显示关于InnoDB存储引擎的当前状态数据
SHOW WARNINGS [LIMIT [offset,] row_count]	显示上一个语句执行后的任何错误、警告或提示
SHOW ERRORS [LIMIT [offset,] row_count]	只显示上一个语句执行后的错误信息

12.3.2 使用DESCRIBE获取关于列的信息

作为SHOW COLUMNS命令的替换，可以使用DESCRIBE语句，它类似于Oracle（另一个RDBMS）中的DESCRIBE语句。其基本语法如下所示：

```
DESCRIBE table [column];
```

这将给出表中所有列的信息，或者如果指定了列，将给出该列的信息。也可以在列名中使用通配符。

12.3.3 用EXPLAIN理解查询操作的工作过程

有两种方式可以调用EXPLAIN语句。第一种，可以使用：

```
EXPLAIN table;
```

上述语句的输出结果非常类似于DESCRIBE table或SHOW COLUMNS FROM table语句的输出结果。

第二种，也是更有趣的方式，可以使用EXPLAIN语句来查看MySQL是如何来解释并执行一个SELECT查询。要使用这种方式，只要在SELECT语句前面加上单词EXPLAIN即可。

当试图使一个复杂的查询能够正常工作起来而查询语句写得不是很正确的时候，或者当一个查询的执行时间大大超出它应该需要的时间时，可以使用EXPLAIN语句。如果编写一个复杂的查询，可以在实际运行查询之前执行EXPLAIN命令以提前检查它，根据该语句的结果，如果有必要的话，可以修改SQL语句，从而对其进行优化。它也是方便的学习工具。

例如，尝试运行下列基于Book-O-Rama数据库的查询。

```
explain
select customers.name
from customers, orders, order_items, books
```

```
where customers.customerid = orders.customerid
and orders.orderid = order_items.orderid
and order_items.isbn = books.isbn
and books.title like '%Java%';
```

其运行结果如下所示（请注意，我们是以垂直的模式显示结果的，因为表行的宽度太大，无法在本书显示出来。可以通过在查询语句末尾处添加\G选项来实现）：

```
*************************** 1. row ***************************
           id: 1
  select_type: SIMPLE
        table: orders
         type: ALL
possible_keys: PRIMARY
          key: NULL
      key_len: NULL
          ref: NULL
         rows: 4
        Extra:
*************************** 2. row ***************************
           id: 1
  select_type: SIMPLE
        table: order_items
         type: ref
possible_keys: PRIMARY
          key: PRIMARY
      key_len: 4
          ref: books.orders.orderid
         rows: 1
        Extra: Using index
*************************** 3. row ***************************
           id: 1
  select_type: SIMPLE
        table: customers
         type: ALL
possible_keys: PRIMARY
          key: NULL
      key_len: NULL
          ref: NULL
         rows: 3
        Extra: Using where; Using join buffer
*************************** 4. row ***************************
           id: 1
  select_type: SIMPLE
        table: books
         type: eq_ref
possible_keys: PRIMARY
          key: PRIMARY
```

```
        key_len: 13
            ref: books.order_items.isbn
           rows: 1
          Extra: Using where
```

初看起来，这个输出结果有点乱，但是它却很有意义。下面，我们将逐个讨论该表中的列。

第一列id，给出的是该行所引用的查询SELECT语句的ID号。

select_type列，解释了所使用的查询类型。该列的值如表12-8所示。

表12-8 EXPLAIN输出结果所包含的可能Select类型

类 型	描 述
SIMPLE	就像这个例子中的，原来的SELECT语句
PRIMARY	使用了子查询和联合的外部（第一个）查询
UNION	联合中的第二个或后一个查询
DEPENDENT UNION	联合中的第二个或后一个查询，根据主查询而定
UNION RESULT	UNION查询的结果
SUBQUERY	内部子查询
DEPENDENT SUBQUERY	内部子查询，根据主查询而定（也就是，一个关联子查询）
DERIVED	在FROM子句中使用的子查询
UNCACHEABLE SUBQUERY	一个结果无法缓存的子查询，必须重新查看每行
UNCACHEABLE UNION	一个UNION中的第二个或后一个SELECT属于一个非缓存子查询

table列只是列出了用来完成查询所需的表。结果中的每一行将给出特定表在查询中如何使用的详细信息。在这个例子中，可以看到，所使用的表包括orders、order_items、customers和books。（通过查看这个查询就可以知道）。

type列解释了表在查询的关联中是如何使用的。该列的可能值如表12-9所示。这些值是按照由快至慢的查询速度排列。通过这个表，可以知道执行一个查询需要读入一个表的多少行。

表12-9 EXPLAIN结果中可能显示的连接类型

类 型	描 述
const或system	只从该表读取一次。发生在表正好只有一行时，当表是一个系统表时，将使用system类型，否则使用const类型
eq_ref	对来自关联中其他表的每组行，从该表读取一行。其使用条件是：结合使用了该表中索引所有部分，而该索引为UNIQUE或为一个主键
fulltext	使用fulltext索引执行了关联
ref	对来自关联中其他表的每组行，从该表中读取一组都匹配的行。其使用条件是：关联不能基于关联条件选择单行，也就是说，关联只使用的关键字一部分或者如果该关键字不是UNIQUE或者主键
ref_or_null	类似一个ref查询，但是MySQL也将查询为NULL的行（这种类型在大多数子查询中使用）
index_merge	特定的优化，Index Merge使用的
unique_subquery	这种关联类型可以用来在返回一个唯一行的IN子查询中代替ref
index_subquery	这种关联类型类似于unique_subquery，但是用作索引非唯一子查询
range	对来自关联中其他表的每组行，在属于某特定范围的表中读取一组行扫描整个索引
index	扫描整个索引
ALL	表中的每一行将被扫描

在前面的例子中，可以发现有两个表是通过eq_ref(orders和customers)来关联的，而其中一个表是使用index(order_items)来关联的，但其他一个表（books）是使用ALL；也就是说，通过查看表中每一行来连接的。

rows列列出（粗略地）执行连接所必须扫描的每个表的行数。这样，可以将这些行数相乘而得出查询执行时所检查的行的总数。我们将这些数字相乘是因为一个连接就像不同表中行的乘积。请参阅第10章获得关于连接的详细信息。请记住，行数是所查看的行数，而不是返回的行数，并且它只是一个估计——没有真正执行查询，MySQL不可能知道确切的数量。

很显然，这个数字越小越好。现在，我们的数据库中只有几乎可以忽略的数据量，但是当数据的容量开始增长的时候，该查询将耗费一定的执行时间。稍后，我们还将回过来讨论它。

与你理解的一样，possible_keys列列出MySQL可能用来连接表的关键字。在这个例子中，可以看出可能的关键字都是PRIMARY关键字。

key列，或者是MySQL实际使用的表中的关键字，或者为NULL，如果没有使用关键字的话。注意对books表尽管可以使用PRIMARY关键字，但是本查询并没有使用它们。

key_len列给出了所用关键字的长度。可以使用它来判别是否只使用了关键字的一部分。当我们使用包含多列的关键字时，这是有意义的。在这个例子中，需要使用关键字的地方，我们都使用了完整关键字。

ref列显示的是用来从表中选择列而必须与关键字一起使用的列。

最后，extra列告诉我们关于连接是如何执行的，以及其他所有信息。在该列中，可能出现的值如表12-10所示。

表12-10　EXPLAIN结果中的Extra列可能出现的值

值	意　义
Distinct	找到第1个匹配行后，MySQL停止查找
Not exists	查询已经用LEFT JOIN优化
Range checked for each record	对连接中的其他表组集合行中的每一行，试图找到要使用的最佳索引，如果有这样索引的话
Using filesort	要经过两个步骤才能分类数据（这显然是两倍长）
Using index	表中所有的信息来自索引——也就是说，实际上并不查找行
Using join buffer	使用联合缓冲将表格写入段，然后将行从缓冲提取出来以完成请求
Using temporary	要执行该查询，需要建立一个临时表
Using where	选择行时使用WHERE子句

可以用多种方法解决在EXPLAIN的结果中出现的问题。首先，可以检查列类型并且确认它们相同。这特别适用于列宽度。如果它们的列宽度不同，索引不能用于匹配这些列。

可以通过修改列类型以使其匹配，或者在设计的开始就建立类型匹配的列，从而解决这个问题。

其次，可以让关联优化器来检查关键字的分布，因此使用myisamchk或ANALYZE TABLE语句对关联进行优化，使得它效率更高。可以使用如下命令调用它：

```
myisamchk --analyze pathtomysqldatabase/table
```

可以通过在命令行上列出所有连接，对多个表进行检查，或者使用：

```
myisamchk --analyze pathtomysqldatabase/*.MYI
```

可以运行下列命令检查所有数据库的所有表：

```
myisamchk --analyze pathtomysqldatadirectory/*/*.MYI
```

或者，可以在MySQL监视程序中运行ANALYZE TABLE语句来列出所有表：

```
analyze table customers, orders, order_items, books;
```

除此之外，可能还要考虑对该表添加一个新索引。如果该查询是a）慢和b）一般，就应该认真考虑这个问题。如果它是一个使用一次就不再使用的查询，例如请求一次的模糊报告，它将不值得这样的努力，因为它将使其他东西慢下来。我们将在下一节讨论如何解决它。

如果EXPLAIN结果possible_keys列包含一些NULL值，可能需要对正在被讨论的表添加一个索引来提高查询性能。如果在WHERE子句中使用的列适合作为索引，可以使用ALTER TABLE语句为它创建一个新索引，例如：

```
ALTER TABLE表ADD INDEX (列);
```

12.4 数据库的优化

除了上述的优化技巧，通常，还有许多技巧可以用来提高MySQL数据库的性能。

12.4.1 设计优化

一般来说，我们可能会希望数据库中每一个数据越小越好。因此，可以使用最小化冗余的设计思想来实现此目的。当然，也可以通过使用最小的列数据类型来实现它。也应该尽可能使NULL最少，使主键尽可能短。

如果可能，尽量避免使用可变长度列（像VARCHAR、TEXT和BLOB）。如果字段长度固定，它们用起来将更快，但是要占用多一点的空间。

12.4.2 权限

除了使用前面所述的关于EXPLAIN的建议，还可以通过简化权限来提高查询速度。前面，我们讨论了在查询执行之前通过权限系统检查该查询的过程，该过程越简单，查询速度越快。

12.4.3 表的优化

如果一个表已经用了一段时间，随着更新和删除操作的发生，数据将会变得支离破碎。这样同样会增加在该表中查询所花的时间。可以使用如下语句修复它：

```
OPTIMIZE TABLE tablename;
```

或者在命令提示符下键入：

```
myisamchk -r table
```

也可以使用myisamchk工具根据索引对该表索引和数据进行排序，如下所示：

```
myisamchk --sort-index --sort-records=1 pathtomysqldatadirectory/*/*.MYI
```

12.4.4　使用索引

可以在需要提高查询速度的地方使用索引。简化索引，不要创建查询不使用的索引。运行前面所介绍的EXPLAIN命令可以检查该索引是否正在使用中。

12.4.5　使用默认值

在尽可能的地方使用列的默认值，只在与默认值不同的时候才插入数据。这样可以减少执行INSERT语句所花的时间。

12.4.6　其他技巧

在特殊的情况下或者有特别需要的时候，还可以使用许多其他小技巧来提高性能。

MySQL网站提供了一组技巧。可以在如下网址找到它：http://www.mysql.com

12.5　备份MySQL数据库

在MySQL中，可以通过几种不同的方法来执行数据库的备份。第一种方法是在复制数据文件时使用LOCK TABLES命令锁定这些表。该命令语法如下所示：

```
LOCK TABLES table lock_type [, table lock_type ...]
```

每一个表必须是表的名称，而锁定类型可以是READ或WRITE。对于备份来说，只需要READ锁。在执行备份之前，必须执行FLUSH TABLES；命令来确保对索引所做的任何修改将写入到磁盘。

在执行备份时，用户和脚本还可以运行只读查询。如果有大量可以修改数据库的查询，例如客户订单，这种解决方案并不实际。

第二种方法，也是比较好的方法是使用mysql_dump命令。该命令是在操作系统的命令行下使用的，该命令的典型用法如下所示：

```
mysqldump --opt  -all-databases > all.sql
```

上述命令将所有用来重新构建一个SQL数据库所需的内容都导出到一个名为all.sql的文件中。

稍后，可以停止mysqld进程，并且通过--log-bin[=logfile]命令选项重新启动它。保存在日志文件中的更新将给出上次备份后数据库发生的变化。（很明显，在任何常规文件的备份中，还应该备份所有日志文件。）

第三种方法是使用mysqlhotcopy脚本。可以使用如下命令调用：

```
mysqlhotcopy database /path/for/backup
```

然后，必须遵照前面介绍的启动和停止数据库的操作。

备份（故障转移）的最后一个方法是维护数据库的一个副本。数据库的复制将在本章稍后内容介绍。

12.6 恢复MySQL数据库

如果需要恢复MySQL数据库，也有许多方法可以实现。如果出现了一个破坏了的表，可以运行带有-r（修复）选项的myisamchk命令。

如果使用了第一种方法执行备份，可以将数据文件重新复制到安装MySQL的相同位置。

如果使用了第二种方法执行备份，数据库的恢复就需要执行一些操作。首先，必须在导出文件中运行查询。这可以将数据库重新构建至导出该文件时的状态。接着，还应该将数据库更新至保存在二进制日志文件中的状态。可以运行如下所示的命令：

```
mysqlbinlog hostname-bin.[0-9]* | mysql
```

可以在MySQL站点找到关于MySQL备份和恢复的更多信息： http://www.mysql.com

12.7 实现复制

复制是一个允许提供相同数据库的多个数据库服务器的技术。这样，可以载入共享并提高系统可靠性；如果有一个服务器停止运行，其他服务器还能继续工作。复制一旦设置成功。它也可以用作备份。

其基本思想就是拥有一个主服务器，并且为其添加几个从服务器。每一个从服务器都镜像了主服务器。当最初设置了从服务器后，可以在任何时候在主服务器上复制所有数据的快照。这样，从服务器将请求来自主服务器的更新。主服务器将传输通过其二进制日志而执行的查询细节，而从服务器可以重新将这些查询应用于它们的数据。

使用这种设置的常规方法是在主服务器应用写查询，而对从服务器应用读查询。这是通过应用程序逻辑来实现的。更复杂的架构也是可能的，例如具有多个主服务器，但是我们只介绍典型示例的设置。

我们必须意识到，通常从服务器的数据都没有主服务器的数据新。这发生在任何分布式数据库中。

要开始设置主服务器和从服务器架构，必须确认主服务器上启用了二进制日志记录。启用二进制日志记录将在附录A详细介绍。

必须在主服务器和从服务器上编辑my.ini或my.cnf文件。在主服务器上，需要如下所示的设置：

```
[mysqld]
log-bin
server-id=1
```

首要的操作就是开启二进制日志记录（这样就已经启动了二进制日志；如果没有，现在添加该语句）。第二个设置是为主服务器分配一个唯一的ID。每一个从服务器都需要一个ID，因此必须在每一个从服务器的my.ini/my.cnf文件中添加类似行。请确认，ID号是唯一的。例如，第一个从服务器可以设置为server-id = 2；而下一个可以为server-id = 3；等。

12.7.1 设置主服务器

在主服务器上，必须为从服务器创建一个用来连接主服务器的用户。从服务器的这个特殊的权限级别称作复制从服务器。根据如何策划实现初始数据传输的不同，可以临时授予某些额外的权限。

在大多数情况下，可以使用数据库快照来传输数据，而在这个例子中，只需要特殊的复制从服务器器权限。如果决定使用LOAD DATA FROM MASTER命令来传输数据（在下一节学习此内容），该用户还需要RELOAD、SUPER和SELECT权限，但是只用作初始的设置。根据第9章介绍的最少权限原则，在系统设置成功并运行起来后，必须撤销额外的权限。

在主服务器上创建一个用户。我们可以定义任意的用户名并且给定任意的密码，但是必须记住所选择的用户名和密码。在我们的例子中，用户名为rep_slave：

```
grant replication slave
on *.*
to 'rep_slave'@'%' identified by 'password';
```

很明显，我们必须修改该密码。

12.7.2 执行初始的数据传输

将数据从主服务器传输到从服务器有几种方法。最简单的方法是设置从服务器（将在下一节介绍），然后运行LOAD DATA FROM MASTER语句。这种方法的问题是在数据传输过程中，它将锁定主服务器上的表，而且这种传输需要一定的时间，因此我们并不建议使用这种方法（只有使用MyISAM表时，可以使用这个选项）。

通常，在当前时间获得数据库的一个快照是一个更好的办法。可以使用本章其他地方介绍的备份过程来获得快照。必须首先使用如下语句来清空表：

```
flush tables with read lock;
```

使用读锁定的原因是必须在获得快照时记录服务器在二进制日志中的位置。可以通过执行如下所示的语句来实现：

```
show master status;
```

应该看到类似于如下所示的输出结果：

```
+------------------+----------+--------------+------------------+
| File             | Position | Binlog_Do_DB | Binlog_Ignore_DB |
+------------------+----------+--------------+------------------+
| laura-ltc-bin.000003 |       95 |              |                  |
+------------------+----------+--------------+------------------+
```

请注意，File和Position的值；需要这些信息来设置从服务器。

现在，使用如下所示的语句获得快照并去除表的读锁定：

```
unlock tables;
```

如果所使用的是InnoDB表，最简单的方法是使用InnoDB Hot Backup工具，可以在

http://www.innodb.com获得该工具。这并不是一个免费软件，因此需要一定的许可费用。

或者，可以使用这里所介绍的步骤，在去除表锁定之前，关闭MySQL服务器，并且在重新启动服务器和去除表锁定之前，复制所希望复制的数据库整个目录。

12.7.3 设置一个/多个从服务器

设置一个/多个从服务器有两个选项。如果获得数据库的快照，可以在从服务器上安装它。

接下来，在从服务器上运行如下所示查询：

```
change master to
master-host='server',
master-user= user',
master-password='password',
master-log-file='logfile',
master-log-pos= logpos;
start slave;
```

必须填充以斜体字显示的数据。*server*是主服务器的名称。*user*和*password*来自在主服务器运行的GRANT语句。*logfile*和*logpos*来自在主服务器上运行的SHOW MASTER STATUS语句的输出。

现在，你应该已经设置并运行了从服务器。

如果没有获得快照，可以执行如下所示的语句在运行了以上查询后载入数据：

```
load data from master;
```

12.8 进一步学习

在介绍MySQL的这几章中，我们重点讨论了与Web开发相关的系统使用，以及在PHP中使用MySQL。要了解关于MySQL管理的详细信息，可以访问MySQL网站：http://www.mysql.com

也可以参阅MySQL出版的图书《MySQL Administrator's Guide》或Paul Dubois编写的《MySQL，Fourth Edition》一书，该书可以向Addison-Wesley出版社购买。

12.9 下一章

在第13章中，我们将介绍MySQL的一些高级特性，这些特性对于编写Web应用程序来说是非常有用的，例如如何使用不同的存储引擎、事务和存储过程。