第13章 数据库维护和修复

理想的情况是，MySQL 从首次安装以来始终平稳地运行。但有时确实会由于各种原因而出现问题，其范围可以从电源断电到硬件故障到不正常地关闭 MySQL 服务器（如用 kill -9 终止服务器或机器崩溃）。诸如这样的情况大部分都超出您的控制范围，它们会导致数据库表的毁坏，尤其是在对表进行修改且未完全写入时所引起的。

本章的重点是检测和解决表的问题，而不论问题是如何引起的。对于表的检查和修复，MySQL 管理员最好的朋友是 myisamchk 和 isamchk 实用程序。这两个程序有好几个功能，我们已经在第4章讨论了怎样使用它们执行索引键的分布分析和索引的释放与激活。还可以使用它们检查表和修复有问题的表。这使您能在表变坏之前（使表不能使用之前）修正故障。

myisamchk 和 isamchk 提供的全部选项的清单在附录 E 中。有关其他的背景，请参阅 MySQL 参考指南的“维护 MySQL 安装”一章。

13.1 检查和维护数据库表

表的故障检测和修正的一般过程如下：

1) 检查出错的表。如果该表检查通过，则完成任务，否则必须修复它。

2) 在开始修复之前对表文件进行拷贝，以防万一。

3) 试着修复表。

4) 如果修复操作失败，从数据库备份和更新日志中恢复此表。

上述过程的最后一步假定您已经执行了数据库备份并允许更新日志有效。如果不是这样的话，系统将有危险。参考第 11章查找一下怎样使用 mysqlaump 和怎样开启更新日志。您肯定不想不可挽回地丢失一个表，因此，应努力地做备份。

在使用 myisamchk 或 isamchk 检查或修复表之前，应该满足一些初步需求：

- 建立常规的数据库备份过程并允许更新日志，以防事情越来越糟使表的毁坏不能修复。笔者好像在以前提醒过这一点？
- 在开始试验之前应先仔细地阅读本章的内容。尤其是不应该在阅读“避免与 MySQL 服务器交互作用”之前进行操作，因为它将讨论当您试图在一个表上执行检查或修复过程时服务器正在使用这个表所引起的问题。它还讨论怎样在服务器运行时防止那些问题发生。
- 当运行表检查或修复时，您应该被注册在运行 mysql 的账号下，因为您需要对表文件读写访问。

13.1.1 myisamchk 和 isamchk 的调用语法

MySQL 的 myisamchk 和 isamchk 实用程序很类似，多数时候它们可以用同样的方式使用。它们之间的主要区别是它们所使用的表的类型。对于 MyISAM 表，使用 myisamchk，而对于 ISAM 表，则使用 isamchk。您可以通过表的索引文件的扩展名来告诉表使用哪种存储

格式。扩展名“ .MYI ”表明是一个 MyISAM 表，而“ .ISM ”表明是 ISAM 表。

为了使用任一个实用程序，应指明您所要检查或修复的表，以及指明要执行的操作类型的选项：

```
% myisamchk options tbl_name ...
% isamchk options tbl_name ...
```

tbl_name 参数可以是表名也可以是该表的索引文件名。如果指定多个表，可以很容易地使用文件名模式来拾取目录中所有相应的文件：

```
% myisamchk options *.MYI
% isamchk options *.ISM
```

不会因为告诉了错误的程序来检查某个表而使该表毁坏，但是除了发布一条警告消息外此程序不做任何事情。例如，下面的第一条语句将检查当前目录中的所有 MyISAM 表，而第二条语句只显示一条警告消息：

```
% myisamchk *.MYI          正确
% myisamchk *.ISM          不正确——文件类型错
```

不论是 myisamchk 还是 isamchk 都不对表所在的位置做任何判断，因此，应该或者在包含表文件的目录中运行程序，或者指定表的路径名。这允许您将表文件拷贝到另一个目录中并用该拷贝进行操作。

13.1.2 检查表

myisamchk 和isamchk 提供了表检查方法，这些方法在彻底检查表的程度方面有差异。通常用标准方法就足够了。如果标准检查报告没有发现错误而您仍然怀疑有毁坏（或许因为查询没有正常地工作），可能要执行更彻底的检查。要想用任意一个实用程序执行标准的表检查，则不用带任何选项直接调用即可：

```
% myisamchk tbl_name
% isamchk tbl_name
```

为了执行扩充检查，使用 --extend-check 选项。该选项非常慢，但检查极为彻底。对于该表的数据文件中的每个记录，索引文件中的每个索引的相关键都被检查以确保它真正指向正确的记录。myisamchk 还有一个中间选项 --medium-check，它不如扩展检查彻底，但速度快。

如果对于 --extend-check 检查不报告错误，则可以肯定表是好的。如果您仍然感觉表有问题，那原因肯定在其他地方。应重新检查任何好像有问题的查询以验证查询是正确书写的。如果您认为问题可能是 MySQL 服务器的原因，应考虑整理一份故障报告或升级到新的版本上。

如果 myisamchk 或 isamchk 报告表有错误，应用下节中的说明修复它们。

13.1.3 修复表

表的修复是一项可怕的工作，如果具体问题非常独特则更难进行。然而，有一些常规的指导思想和过程，可以遵循它们来增加修正表的机会。通常，开始时可以用最快的修复方法，看看是否能修正故障。如果发现不行的话，可以逐步升级到更彻底的（但更慢的）修复方法上，直到故障被修复或您不能继续升级为止（实际上，大多数问题不用更大规模的和更慢的方法就能修正）。如果表不能修复，则从备份中恢复该表。有关使用备份文件和更新日志进行

恢复的指导在已第11章中给出。

1. 执行标准的表修复

为了修复一个表，执行下列步骤：

1) 试着用 --recover 选项修正表，但也可以用 --quick 选项试图只根据索引文件的内容进行恢复。这样将不触及数据文件：

```
% myisamchk --recover --quick tbl_name
% isamchk --recover --quick tbl_name
```

2) 如果问题仍存在，再试一下上一步的命令，但忽略 --quick 选项，以允许 myisamchk 或 isamchk 前进并修改数据文件：

```
% myisamchk --recover tbl_name
% isamchk --recover tbl_name
```

3) 如果还不工作，试一试 --safe-recover 修复方法。这种方法比普通的恢复方法要慢，但能够修正 -recover 方法不能修正的几个问题：

```
% myisamchk --safe-recover tbl_name
% isamchk --safe-recover tbl_name
```

如果 myisamchk 或 isamchk 由于一个 “Can’t create new temp file: file_name” 的错误消息在任何一步中停止，应该重复这个命令并增加 --force 选项以迫使清除临时文件。这个临时文件可能是从上一次失败的修复中留下的。

在修复表之前拷贝它们

在执行表修复前应该遵循的一个常规的预防措施是做该表的新拷贝。这种情况未必出现，但如果发生，则可以从拷贝文件中做该表的新的拷贝并试试另一种恢复方法。

2. 标准表修复方法失败时怎么办

如果标准的修复过程未能修复表，则索引文件可能在修复时丢失或毁坏。尽管未必可能，但还是有可能使表的描述文件丢失。不论哪种情况，都需要替换受影响的文件，然后再试试标准修复过程。

为了重新生成索引文件，可以使用下列过程：

1) 定位到包含崩溃表的数据库目录中。

2) 将该表的数据文件移到安全的地方。

3) 调用 mysql 并通过执行下列语句重新创建新的空表，该语句使用表的描述文件 tbl_name.frm 重新开始生成新的数据和索引文件：

```
mysql> DELETE FROM tbl_name;
```

4) 退出 mysql，将原始的数据文件移回到数据库目录中，替换刚建立的新的空文件。

5) 再试试标准表修复方法。

为了恢复该表的描述文件，可先从备份文件中恢复，然后再试着用标准修复方法。如果由于某些原因没有备份，但知道建立表的 CREATE TABLE 语句，则仍可以恢复该文件：

1) 定位到包含崩溃表的数据库目录中。

2) 将该表的数据文件移动到安全的地方。如果想要使用索引的话，还需将索引文件移走。

3) 调用 mysql 并发布 CREATE TABLE 语句建立该表。

4) 退出 mysql，将原始数据文件移回数据库目录中，替换刚才新建的数据文件。如果在步骤2移动了索引文件，则也要将其移回数据库目录中。

5) 再试试标准表修复方法。

13.1.4 避免与 MySQL 服务器交互作用

当您正在运行表的检查/修复实用程序时，您或许不想让 MySQL 服务器和实用程序同时访问一个表。如果两个程序都向表中写数据显然是一件坏事，但是，当一个程序在写入时另一个程序在读取也不是件好事。如果表正由一个程序写入，同时进行读取的另一个程序会被搞乱。

如果您关闭服务器，就可以保证在服务器和 myisamchk 或isamchk 之间没有交互作用。但是管理员极不愿意使服务器完全地脱机，因为这使得没有故障的数据库和表也不可用。本节中讨论的过程将帮助您避免服务器和 myisamchk 或 isamchk 之间的交互作用。

服务器有两种类型的锁定方法。它使用内部锁定避免客户机的请求相互干扰——例如，避免客户机的 SELECT 查询被另一个客户机的 UPDATE 查询所干扰。服务器还使用外部锁定（文件级锁）来防止其他程序在服务器使用表时修改该表的文件。通常，在表的检查操作中服务器将外部锁定与 myisamchk 或 isamchk 组合使用。但是，外部锁定在某些系统中是禁用的，因为它不能可靠地进行工作。对运行 myisamchk 和 isamchk 所选择的过程取决于服务器是否能使用外部锁定。如果不使用，则必须使用内部锁定协议。

如果服务器用 --skip-locking 选项运行，则外部锁定禁用。该选项在某些系统中是缺省的，如Linux。可以通过运行 mysqladmin variables 命令确定服务器是否能够使用外部锁定。检查 skip_locking 变量的值并按以下方法进行：

- 如果 skip_locking 为 off，则外部锁定有效。您可以继续并运行任一个实用程序来检查表。服务器和实用程序将合作对表进行访问。但是，在运行任何一个实用程序之前，应该用 mysqladmin flush-tables 刷新表的高速缓存。为了修复表，应该使用表的修复锁定协议。
- 如果 skip_locking 为 on，则禁用外部锁定，但在 myisamchk 或isamchk 检查或修复一个表时服务器并不知道，最好关闭服务器。如果坚持使服务器保持开启状态，需要确保在您使用此表时没有客户机来访问它。必须使用恰当的锁定协议告诉服务器使该表独处，并阻塞客户机对其访问。

这里所描述的锁定协议使用服务器的内部锁定机制，以防止服务器在您利用 myisamchk 或isamchk 工作时访问表。通常的办法是调用 mysql 并对要检查或修复的表发布 LOCK TABLE 语句。然后，在 mysql 空闲时（即运行，但除了保持该表锁定外不用它做任何事情），运行 myisamchk 或 isamchk。在 myisamchk 或 isamchk 结束后，可以切换到 mysql 会话中并释放该锁以告诉服务器程序执行完毕此表可以再次使用了。

检查和修复的锁定协议有点区别。对于检查，您只需要获得读锁。在这种情况下，只能读取表，但不能修改它，因此它也允许其他客户机读取它。读锁足以防止其他客户机修改表。对于修复，您必须获得写锁以防止任何客户机在您对表进行操作时修改它。

锁定协议使用 LOCK TABLE 和 UNLOCK TABLE 语句获得并释放锁。协议还使用 FLUSH TABLES 告诉服务器刷新磁盘中任何未决的改变，并在通过表修复实用程序修改表后重新打开该表。您必须从单个 mysql 会话中执行所有 LOCK、FLUSH 和 UNLOCK 语句。如果锁定一个表然后退出 mysql，则该锁将释放，且运行 myisamchk 或isamchk 将不再是安全的！

如果保持打开两个窗口的状态，且一个运行 mysql，而另一个运行 myisamchk 或 isamchk，则运行锁定过程将会变得很容易。这样允许您很容易地在程序之间进行切换。如果不是运行在视窗环境中，当运行 myisamchk 或 isamchk 时，将需要使用外壳程序的作业控制工具暂停和恢复 mysql。下面的指导显示出对 myisamchk 或isamchk 的命令，可用与您正在使用的表相对应的那个命令。

1. 对检查操作锁定表

此过程只针对表的检查，不针对表的修复。在窗口 1中，调用 mysql 并发布下列语句：

```
% mysql db_name
mysql> LOCK TABLE tbl_name READ;
mysql> FLUSH TABLES;
```

该锁防止其他客户机在检查时写入该表和修改该表。 FLUSH 语句导致服务器关闭表的文件，它将刷新仍然在高速缓存中的任何未写入的改变。

当 mysql 空闲时，切换到窗口 2 并检查该表：

```
% myisamchk tbl_name
% isamchk tbl_name
```

当 myisamchk 或 isamchk 结束时，切换回到窗口 1 的 mysql 会话并释放该表锁：

```
mysql> UNLOCK TABLE;
```

如果 myisamchk 或isamchk 指出发现该表的问题，将需要执行表的修复。

2. 对修复操作锁定表

修复表的锁定过程类似于检查表的过程，但有两个区别。第一，您必须得到写锁而非读锁。由于您将要修改表，因此根本不允许客户机对其进行访问。第二，必须在执行修复之后发布 FLUSH TABLE 语句，因为 myisamchk 和 isamchk 建立了新的索引文件，除非再次刷新该表的高速缓存否则服务器将不会注意到它：

```
% mysql db_name
mysql> LOCK TABLE tbl_name WRITE;
mysql> FLUSH TABLES;
```

利用 mysql 的空闲切换到窗口2，做该表的数据库文件的拷贝，然后运行 myisamchk 或 isamchk：

```
% cp tbl_name.* /some/other/directory
% myisamchk --recover tbl_name
% isamchk --recover tbl_name
```

--recover 选项只是针对安装而设置的。这些特殊选项的选择将取决于您执行修复的类型。myisamchk 或 isamchk 运行完成后，切换回到窗口1的 mysql 会话，再次刷新该表的高速缓存并释放表锁：

```
mysql> FLUSH TABLES;
mysql> UNLOCK TABLE;
```

13.1.5 快速运行 myisamchk 和 isamchk

myisamchk 和 isamchk 的运行可能会花很长时间，尤其是您正在处理一个大表或使用一个更广泛的检查或修复方法时。通过告诉这些程序在运行时使用更多的内存，能够提高它们的速度。这两个实用程序都有几个可设置的操作参数。其中最重要的是控制程序使用的缓冲区大小的变量：

变 量	含 义
key_buffer_size	用于存放索引块的缓冲区大小
read_buffer_size	读操作用的缓冲区大小
sort_buffer_size	排序用的缓冲区大小
write_buffer_size	写操作用的缓冲区大小

要想查看任一个程序使用的这些变量的缺省值，可用 --help 选项运行该程序。要想指定其他的值，可在该命令上使用 --set-variable variable=value 或 -O variable=value。您可以将变量的名字简化成 key、read、sort 和 write。例如，可告诉 myisamchk 使用 16MB 的排序缓冲区和 1MB 的读写缓冲区，其调用如下：

```
% myisamchk -O sort=16M -O read=1M -O write=1M ...
```

sort_buffer_size 只能利用 --recover 选项来使用（而不是利用 --safe_recover）,在这种情况下，key_buffer 不能使用。

减少服务器的停机时间

防止服务器访问（您正在处理的）表的另一种方法是在数据目录的外面使用该表文件的拷贝。这样并不能消除交互作用的问题，因为仍然必须防止服务器访问（并可能修改）正在进行拷贝的表。但是，如果您不愿意使服务器脱机的话，该路线可能是使服务器停机时间最小化的一种方法，这对您是有吸引力的。在将该表的文件拷贝到另一个目录时关闭服务器，然后恢复服务器。

myisamchk 的未来打算

myisamchk 的表检查和修复功能打算在 MySQL 3.23 版本系列的某个时候被合并到服务器中。如果这种打算实现，对表的检查和修复将更容易，因为服务器与 myisamchk 的交互问题将不再会出现。

同样，您能够告诉服务器在启动时检查表，因此在启动服务器前将不需要设置任何特殊的命令在引导期间执行。该程序不对 ISAM 表进行操作，因此在服务器获得表的检查和修复能力时，应考虑将 ISAM 表转换成 MyISAM 表。请查看新发行版的 MySQL 参考指南，了解在此范围内有什么新进展。可以用 ALTER TABLE 语句转换表的类型：

```
ALTER TABLE tbl_name TYPE = MYISAM
```

13.2 安排预防性的维护

应该考虑建立一个预防性维护的时间表，以协助自动检测问题，使得您可以采取措施进行修正：

1) 执行常规的数据库备份并允许更新日志。

2) 安排定期的常规表检查。通过检查表，将减少使用备份的机会。这个工作使用 cron 作业（一般从运行服务器所使用的该账号的 crontab 文件中调用）并且很容易实现。例如，如果您作为 mysqladm 用户运行服务器，则可以从 mysqladm 的 crontab 文件中建立定期检查。如果您不知道如何使用 cron，应使用下列命令查看相关的 UNIX 人工页：

```
% man cron
% man crontab
```

3) 在服务器启动前的系统引导期间检查数据库表。机器可能会因早期的崩溃重新启动。如果是这样的话，数据库表可能已被毁坏，应该对它进行检查。

为了运行自动的表检查，可以编写一个脚本，将目录改变为服务器数据目录并对所有数据库表运行 myisamchk 和isamchk。我们将在下面讨论的脚本中同时使用这两个程序。如果您只有 MyISAM 表或只有 ISAM 表，则只需其中一个程序，可以将无关的那个程序从脚本中去除。

myisamchk 和 isamchk 都根据表检查的方式产生某些输出结果以便了解正在检查哪些表，甚至在没有问题时也是如此。对于 crontab 的项，除非表中有错误，否则通常将禁止输出结果（如果作业产生任何输出，cron 作业通常生成一个邮件消息，很少会收到没有任何问题的表检查作业的邮件）。如果用--silent 选项调用任一个实用程序，它仅当发现问题时才产生输出。另外，myisamchk 支持 --fast 选项，该选项允许程序跳过自上次检查以来没有被修改过的任何表。

在服务器数据目录中检查所有表的一个简单的脚本如下（ DATADIR 应该修改成对应您系统的值）：

```
#! /bin/sh
cd DATADIR
myisamchk --silent --fast */*.MYI
isamchk --silent */*.ISM
```

实用此脚本的一个潜在的问题是：如果有许多表，通配符模式‘ */*.MYI ’或‘ */*.ISM ’可能会由于“ too many arguments（过多的参数）”错误使外壳程序爆炸。另一个可选择的脚本如下（同样，将 DATADIR 改变为对应您系统的值）：

```
#! /bin/sh
datadir=DATADIR
find $datadir -name "*.MYI" -print | xargs myisamchk --silent --fast
find $datadir -name "*.ISM" -print | xargs isamchk --silent
```

不论您选择哪种形式的脚本，笔者都假定您调用的是 check_mysql_tables，应确保及时改变该方式使它可执行，然后您手工调用它以验证它工作是否正常：

```
% chmod +x check_mysql_tables
% check_mysql_tables
```

在理想情况下应该没有输出结果。如果系统不支持外部锁定，有可能服务器将在您检查表时改变它。此时，脚本可能会把实际没问题的表报告成有问题的。这有点不幸，但比出现相反的问题要好：当出现某些故障时脚本报告无问题。如果系统支持外部锁定，则该问题就不会出现。

以下部分将说明如何建立脚本，使它通过 cron 并在系统启动期间自动执行。在这些小节的例子中，笔者假定脚本安装在 /usr/users/mysaladm/bin 中。您将需要调整适合自己系统的值。

如果在您的系统上运行了多个服务器，将需要修改该过程来检查每个服务器数据目录中的表。您可以使用不同的 check_mysql_tables 拷贝来进行，或通过修改它来接收一个命令行参数进行，该参数指定了想要检查的数据目录。

13.2.1 用 cron 定期检查表

假定要想对 mysqladm 用户从 crontab 文件中调用脚本 check_mysql_tables。先以该用户的身份进行注册，然后用下列命令编辑 crontab 文件：

```
% crontab -e
```

该命令带您进入带有当前 crontab 文件拷贝的编辑器中（如果以前没有编辑过，此文件可能是空的）。增加一行到文件中：

```
0 3 * * 0 /usr/users/mysqladm/bin/check_mysql_tables
```

它告诉 cron 在每个星期日的上午 3 时运行此选项。可以按要求改变时间或安排。有关这些选项的格式，请参阅 crontab 人工页。

13.2.2 在系统启动期间检查表

如果您正在使用 BSD 风格的系统，并且已经将服务器的启动命令增加到 /etc/rc.local 或类似的其他文件中，则可以在启动服务器前从对应的文件中调用 check_mysql_tables。

如果正在使用 System V 风格的启动方法从 /etc/rc.d 目录之一中调用 mysql.server，则过程稍有点复杂。这些目录中的启动脚本应该理解 start 和 stop 参数的含义，以便对系统的启动和关闭采取相应的操作。为了执行表的检查，我们可以编写这样的脚本：当参数为 start 时调用 check_mysql_tables，而当该参数为 stop 时不做任何事情。让我们来调用这样的一个脚本 mysql.check，其内容如下：

```
#! /bin/sh

# See how we were called.
case "$1" in
  start)
    echo -n "Checking MySQL tables: "
    if [ -x /usr/users/mysqladm/bin/check_mysql_tables ]; then
        /usr/users/mysqladm/bin/check_mysql_tables
    fi
    ;;
  stop)
    # don't do anything
    ;;
  *)
    echo "Usage: $0 {start|stop}"
    exit 1
esac

exit 0
```

现在，您已经可以使用一个过程来安装 mysql.check了，该过程类似于在第 11 章所介绍的安装 mysql.server 的过程。必须给 mysql.check 一个运行级目录中较低的前缀号，才能使得它在 mysql.server 前运行。例如，如果在运行级目录中以 S99mysql.server 链接到mysql.server，则应该以 S98mysql.check 链接到 mysql.check 。