如何安装、配置、维护你的**MySQL**系统

徐超（xuchao@bj.tom.com）

前言

本文将介绍如何安装，配置MySQL，使其易维护。同时从一个MySQL DBA的角度，将数据库的日常维护、备份、恢复等事务与系统的安装配置联系起来，由于MySQL是一个OpenSource的系统，他的配置比较灵活。系统文件、数据文件在安装的时候如何指定？日志如何管理？如何备份数据？如何实现在线的完整数据备份？如何恢复数据？这些因素从安装、配置系统的时候就已经决定了。所以，对于MySQL系统来说，好的安装规划模式就是日后整个系统稳定的基础。同时将讲述在不同的情况下如何建立MySQL的同步数据库，如何实现在线的数据库备份。

下面，本文将以目前国内最流行的RedHat Linux(7.3)的系统为代表，讲述如何安装、配置一个容易维护的MySQL系统。

本文介绍的MySQL系统是以MyISAM数据库文件格式为基础的。

第一章 如何规划**Linux**系统

本章我首先讲述如何安装RedHat Linux 7.3，以及系统的分区规划。

第一节 *Linux*系统的安装

由于RedHat Linux 7.3的系统默认方式不支持ReiserFS系统，由于RedHat Linux支持的ext3文件系统的代码仍然处于测试阶段，而且在RedHat Linux 7.3的Linux-2.4.18-3的核心中包含的ext3代码存在问题，工作在多CPU环境中，会导致系统的当机，为了我们的数据安全，我们需要使用相对成熟的ReiserFS系统，如何让RedHat Linux支持ReiserFS系统呢?

一个非常方便的引导参数，可以让RedHat Linux系统支持ReiserFS，同时还可以支持其他更多的文件系统。

从安装光盘引导安装过程，出现以下提示:

```
Boot:
```

等待我们选择以何种方式安装系统的时候，我们输入：linux reiserfs text，选择字符安装界面，当然你也可以不输入text关键字，进入图形安装环境，然后我们按下回车。

Boot: linux reiserfs text

系统开始执行安装过程，显示欢迎安装界面、语言选择等界面，一一进行选择。这里我们选择Custom，自定义系统安装的组件。

第二节 系统的分区规划

由于使用Lilo + ReiserFS的系统必须给 /boot 分区使用ext2文件系统，所以我们需要设置一个70MB的/boot。
这里我架设我们的系统已经做过Raid，不考虑磁盘I/O的性能问题。
对于建立Raid系统的一点说明:

微型数据库系统使用2块磁盘做Raid 1（Mirror）
小型数据库系统使用4块磁盘做Raid 0+1
中小型系统使用6-8块磁盘做Raid 5 或者考虑I/O性能最大化使用Raid 0+1
中型系统就要使用外置的磁盘柜了，一般使用Raid 5。

此时数据库容量应该在1T左右了，超过此界限，建议不要使用MySQL作为数据库系统了，建议选用Oracle，DB2等系统，或者将当前数据库系统进行数据分割。

下面我给出一个简单的分区规划

/boot(ext2)	70MB
/	200MB
/usr	2000MB
/usr/local	2000MB
/var	2000MB
/tmp	1000MB
Swap1（Swap分区的总容量要等于内存的容量）	512MB
Swap2（每个swap分区的大小建议不超过512MB）	512MB

/data(数据库文件所在分区)	数据文件分区的大小建议配置为日至文件分区的2 倍
/log(日至文件所在分区)	

分区完成，继续安装Linux 系统。

第三节 我应该安装哪些*Linux* 组件?

对于一个专用的数据库服务器，我们不需要太多的应用，因此，只需要安装最简单的系统即可。一个简化的系统安装内容只需要包括：网络，开发工具，核心开发工具，兼容性软件包。

继续完成下面的安装。

进入系统，执行setup 命令：

```
#setup
```

察看当前系统开启的服务，对于一个专用的数据库系统来说，开启sshd2、mysqld 这两个应用服务足够了，另外你可能还需要打开几个系统服务：crond，network，syslog。其他的服务可以统统关掉了！

OK！现在我们应该已经有了一个不错的Redhat Linux7.3 的系统了！
让我们继续，开始配置MySQL

第二章 编译你的**MySQL**，并且对它进行配置、优化

本章我们将开始安装MySQL，配置它，定制MySQL，让它与我们的Linux 系统完全“兼容”。

第一节 取得最新的*MySQL* 稳定版本，并建立*MySQL* 账户

目前最新的MySQL 版本是3.23.54a，我们从 http://www.mysql.com 上下载mysql-3.23.54a.tar.gz 文件到 /tmp 目录。
使用最新稳定版本的MySQL，是系统将来稳定运行的基础。

建立专用的MySQL 账户：

```
#useradd mysql
```

使用专用的Mysql账户是为了保护系统的安全。

第二节 解压缩，开始编译

```
#cd /tmp
#tar –xzvf mysql-3.23.54a.tar.gz
#cd mysql-3.23.54a
#CFLAGS=-O6 ./configure \
 --prefix=/usr \
 --with-charset=gb2312 \
 --with-extra-charsets=all \
 --without-bench \
 --without-docs \
 --with-mysqld-user=mysql \
 --localstatedir=/data
#make
#make install
```

这里我们使用了一个CFLAGS=-O6的编译优化参数，由于MySQL的代码在过渡优化的情况会产生问题（详细的说明请参考 http://www.mysql.com 的说明）如果编译生成的MySQL不能正常启动，请去掉或者使用-O5或者更小的数值进行编译优化。

这个配置中我们把mysql的命令行工具指定存放在 /usr/bin 目录，mysqld 文件存放在/usr/libexec 目录，数据库文件存放在 /data 目录。基本字符集是GB2312，同时支持其他的字符集。mysqld 使用mysql用户权限启动。

如果上面的步骤没有任何问题的话，我们的MySQL就已经编译并且安装到系统了，接下来我们就要开始初始化MySQL的数据库了，建立基本的MySQL系统。

第三节 初始化数据库，建立配置文件

初始化数据库，并赋予mysql用户对目录/data的属主权限：

```
#cd scripts
#./ mysql_install_db
#cd ..
#chown –R mysql:mysql /data
```

让 Linux 系统启动的时候可以自动启动 MySQL

```
#cd support-files
#cp mysql.server /etc/rc.d/init.d/mysqld
#chmod +x /etc/rc.d/init.d/mysqld
#chkconfig –add mysqld
#cd ..
```

mysql 的一些参数可以在配置文件中进行指定，也可以在启动的时候在命令行上指定，这里我们选择比较方便的参数文件，mysql 的参数文件可以放在多个位置，位置不同，优先级也不一样，下表中列出了他们之间的关系：

文件名	作用
/etc/my.cnf	全局配置文件
DATADIR/my.cnf	库范围有效，如果你在一台机器安装多个 mysql，希望给不同的 mysql 使用不同的配置文件，这就非常有用了。
Defaults-extra-file	扩展配置文件，在安装系统的时候通过—defaults-extra-file=# 来指定的配置文件。
~/.my.cnf	在用户的个人目录中的配置文件，如果你要建立一个供学生学习 mysql 的实验室环境。或者提供虚拟主机用户一个独立的 mysql server。你可以尝试这个配置文件。

建立我们的 my.cnf 配置文件文件

```
#cd support-files
#cp my-huge.cnf /etc/my.cnf
```

在 support-files 目录中有多个系统预先设置好的建议配置文件，每个配置文件的前几行说明了这个配置文件适用于什么样的系统（CPU 数量，内存相关），请根据自己的机器配置进行选择，我们这里使用了 huge 的配置。

现在我们继续对缺省的配置文件进行进一步的优化：

如果你打算在 MySQL 中存入大量的二进制数据，或者是比较长的文本信息，需要设置 **max_allowed_packet** 这个参数，配置到 16MB 一般来说可以满足 90%以上的各种应用了。如果你没有这样的需求，请保留缺省值。

增加 **record_buffer** 数据记录缓存，**sort_buffer** 排序缓存，以提高 order by，group by 的效率。

统计你的系统的表的数量，粗略估计一下会被频繁的访问的表的数量，如果这个数量大于 512，你就应该考虑增加 **table_cache** 这个参数的值了。

如果你的系统是提供给一个 web 系统，作为数据存储的一部分，你就需要增加 **max_connections** 以保证你有足够的并发连接支持资源，响应来自 webserver 的请求。

把 thread_concurrency 设置成你 CPU 数量的两倍。

增加 **log-bin = /log/mysql-bin.log** 以支持 mysql 的日志系统，同时可以用它建立你的同步数据库。

打开 **log-slow-queries = /log/mysql-slow.log** 功能是十分明智的，这对我们查找、分析 SQL 语句的性能问题是十分有帮助的。系统对执行时间超过 8 秒的 SQL 语句进行记录，如果你发现这个文件每天都在大幅度的增长，就有必要进行分析一下了。等等，为什么这里是 8 秒？而不是其他的什么？哦，还记得有一个 8 秒钟规则吗？对对，就是这个了！当然，你也可以使用 long_query_time 参数来指定这个时间。

最后让我们来说说 key_buffer 这个参数，这个参数对系统的数据检索效率影响最大，尽可能的将这个参数设置的更大，但是不要超过 1G 以上。对于系统的优化调整我会在以后的时间专门讨论，因为这很庞大，涉及各个方面的东西。

更多、更详细的信息还是请参考权威的官方文档。

下面我们给出经过再次优化后，完整的/etc/my.cnf 的文件内容：

```
# Example mysql config file for very large systems.
#
# This is for large system with memory of 1G-2G where the system runs mainly
# MySQL.
#
# You can copy this file to
# /etc/my.cnf to set global options,
# mysql-data-dir/my.cnf to set server-specific options (in this
# installation this directory is /data) or
# ~/.my.cnf to set user-specific options.
#
# One can in this file use all long options that the program supports.
# If you want to know which options a program support, run the program
# with --help option.

# The following options will be passed to all MySQL clients
[client]
#password       = your_password
port            = 3306
socket          = /tmp/mysql.sock

# Here follows entries for some specific programs

# The MySQL server
[mysqld]
port            = 3306
socket          = /tmp/mysql.sock
skip-locking
set-variable    = key_buffer=384M
set-variable    = max_allowed_packet=16M
set-variable    = table_cache=512
set-variable    = sort_buffer=8M
set-variable    = record_buffer=8M
set-variable    = thread_cache=8
set-variable    = max_connections=2000
# Try number of CPU's*2 for thread_concurrency
set-variable    = thread_concurrency=4
set-variable    = myisam_sort_buffer_size=64M
log-bin         = /log/mysql-bin.log
server-id       = 1
log-slow-queries= /log/mysql-slow.log
```

OK！现在让我们启动MySQL
我发现如果指定了slow-log，系统不能自动创建这个文件，至少在3.23.54a这个版本是这样，所以我们需要手工创建这个文件。

```
#touch /data/mysql-slow.log
#chown mysql:mysql /data/mysql-slow.log
#/etc/rc.d/init.d/mysqld start
```

现在你可以建立你的数据库文件，库结构等所有的信息了，同时你会看到/data目录中产生了log文件。

第三章 **MySQL**的用户权限管理系统

本章我们将了解MySQL的用户权限管理系统，如何创建数据库的访问用户，以及如何分配权限，MySQL提供了哪些权限可以让我们使用。

第一节 *MySQL*的权限的分类

MySQL的权限大致分为4个等级：

全局级（Global Level）	对象是所有库，权限的定义在mysql.user表中体现
库级（Database Level）	对象为指定库中的所有表，权限的定义在mysql.db, mysql.host表中体现
表级（Table Level）	对象为指定表中的所有列，权限的定义在mysql.tables_priv表中体现
列级（Column Level）	对象为指定表中的单一列，权限的定义在mysql.columns_priv表中体现

MySQL的权限种类列表：

权限（**priv_type**）	权限说明
ALL [PRIVILEGES]	所有权限但是没有赋予 WITH GRANT OPTION
ALTER	允许使用 Alter
CREATE	允许使用 Create
DELETE	允许使用 Delete
DROP	允许使用 Drop Table
FILE	允许使用 SELECT ... INTO OUTFILE 与 LOAD DATA INFILE
INDEX	允许使用 CREATE INDEX 与 DROP INDEX
INSERT	允许使用 Insert
PROCESS	允许使用 Show [FULL] Processlist，同时可以不受最大连接数限制
RELOAD	允许使用 Flush Table
SELECT	允许使用 Select
SHUTDOWN	允许使用 mysqladmin shutdown
UPDATE	允许使用 Update
USAGE	允许连接 MySQL，相当于没有任何权限的登陆

第二节 用户的创建与权限的赋予

权限赋予指令:

```
GRANT priv_type [(column_list)] [, priv_type [(column_list)] ...]
    ON {tbl_name | * | *.* | db_name.*}
    TO user_name [IDENTIFIED BY [PASSWORD] 'encoded password']
        [, user_name [IDENTIFIED BY 'password'] ...]
    [WITH GRANT OPTION ]
```

如果使用了 WITH GRANT OPTION 参数，用户得到权限后可以将所获得的权限再赋予其他人。

现在创建一个 bbsuser 的用户，赋予它从 192.168.3.101 使用密码 bbstest 访问 bbs 数据库中所有表的权限。

```
mysql> grant select, insert, delete, update on bbs.* to bbsuser@192.168.3.101 identified by "bbstest";
```

缺省的 GRANT 赋予权限的用户，如果不存在，会自动创建，同时赋予 USAGE 权限

第三节 权限的收回

权限收回指令：

```
REVOKE priv_type [(column_list)] [, priv_type [(column_list)] ...]
    ON {tbl_name | * | *.* | db_name.*}
    FROM user_name [, user_name ...]
```

我们现在收回刚刚赋予权限的 bbsuser 用户的权限：

```
mysql> REVOKE select, insert, delete update on bbs.* from bbsuser@192.168.3.101;
```

第四节 查看权限

查看指定用户权限的指令：

```
SHOW GRANTS FOR user_name;
```

第五节 删除用户

删除用户需要直接操作系统库 mysql

```
mysql> use mysql;
mysql> delete from user where Host= "localhost" and User= "bbsuser";
mysql> flush privileges;
```

关于 Flush privileges，凡是直接操作系统库 mysql，只有执行 flush 操作后修改才能生效。在 MySQL 的权限分配系统里面 ALL 实际上是一个权限的集合。

第四章 建立同步备份数据库

MySQL 提供了数据库的同步备份功能，这对我们实现数据库的冗灾、备份/ 恢复、负载均衡等都是有极大帮助的。

第一节 配置一台同步数据库

同步数据库的硬件环境最好能够与主数据库完全一致，软件环境完全与主数据库相同的配置，可是按照前面的配置方法完成同步数据库的基本配置。

现在我们需要在 /etc/my.cnf 文件里面做些修改，使这台数据库成为同步数据库：

首先我们找到 server-id ，这个 id 是每一台数据库的唯一编号，在一个数据库同步系统里面编号不允许重复，我们现在把这个编号设置成 2 。

然后我们在下面增加几行：

master-host = 主数据库的 ip

master-user = 数据同步专用账户名

master-password = 密码

master-port = 主数据库的 TCP/IP 端口号

同时我们要去掉他的 log-bin 选项。

为了保护 SLAVE 端的 mysql 系统库，我们需要增加：

replicate-ignore-db = mysql

第二节 给予同步权限，启动同步数据库

我们现在回到主数据库，使用 mysql 工具连接到数据库：

```
#mysql –u localhost –u root –pxxxxxxxx
>grant FILE,SELECT on *.* to repl@<ip> identified by '<password>';
>quit
```

我们建立了一个用户repl，对所有的数据库有FILE，SELECT权限，但是他只能通过<ip>，并且使用指定的密码<password>来访问主数据库。这里的<ip>为同步数据库的ip地址。

这样我们就给予了同步数据库从主数据库获取同步数据的权限。

我们再次回到同步数据库，现在我们来启动同步数据库，然后连接上去：

```
#/etc/rc.d/init.d/mysqld start
#mysql –h localhost –u root –pxxxxxxxx
>show slave status;
+-----------+-----------+-----------+-------------+-------------+----
+-------------+...
|Master_Host|Master_User|Master_Port|Connect_retry|Log_File     |Pos |Slave_Running|
...
+-----------+-----------+-----------+-------------+-------------+----
+-------------+...
| x.x.x.x | repl    | 3306    |60       |mysql-bin.001|  4|Yes       |...
+-----------+-----------+-----------+-------------+-------------+----
+-------------+...
```

我们看到Slave_Running这一列的值是Yes，说明数据库的同步已经在运行了。
多次执行show slave status 指令，可以看到Pos这列的值不断的增加。

第三节 如果同步出现了问题，我该如何处理?

如果我们执行show slave status 的时候发现Slave_Running的列值为No，这个时候就说明MySQL的数据同步系统出现了问题，参考后面两列（Last_errno ，Last_error）的信息得知详细的问题报告。一般的Last_error列会给出出错的SQL语句，同时给出错误的原因。

一般的我们只需要执行：

```
> SET SQL_SLAVE_SKIP_COUNTER=1;
> SLAVE START;
> SHOW SLAVE STATUS;
```

就可以跳过出现问题的SQL 语句，并且启动同步进程，如果察看同步状态还是No，我们就需要重复执行上面的三行语句，至于数据是不是会有损失，那就要根据错误的信息、数据库里面的数据来进行判断了。
如果出现了大量的同步错误，而且你已经确认这些错误都可以忽略的时候，反复的执行这三条语句是非常麻烦的，有没有简单的方法呢?
这里我提供一个简单的脚本:

```
#!/bin/sh

while [ 1 ];
do
    while [ `echo "show slave status" | mysql | head -n2 | tail -1 | awk '{print $7}'` = 'No' ];
    do
        echo "SET SQL_SLAVE_SKIP_COUNTER=1; SLAVE START;" | mysql; echo "continue ...";
        sleep 1;
    done;
    sleep 60;
done;
```

你也可以把他们写在一行里面:

```
while [ 1 ]; do while [ `echo "show slave status" | mysql | head –n2 | tail -1 | awk '{print $7}'` = 'No' ]; do echo "SET SQL_SLAVE_SKIP_COUNTER=1; SLAVE START;" | mysql; echo "continue ..."; sleep 1; done; sleep 60; done;
```

这个脚本假设你是以root 身份运行，你的mysql 超级用户是 root，而且密码为空。如果不是这样，请在mysql 命令后面增加适当的参数。

这个脚本运行的时候会每秒执行一次操作，跳过同步错误的SQL 语句。如果长时间没有“continue ...”出现，就按下 <CTRL>+<C> 中止。

第四节 高级数据同步技术

在建立数据同步系统的时候我们可能会遇到以下的情况:

1.只有一台主数据库，我们将要增加的同步数据库是第一台同步库
对于这种情况，如何在不停机的情况下建立第一台同步数据库呢?
首先，我们要保证主数据库的binlog 的完整，也就是说我们在配置主数据库的时候，打开了binlog 功能，而且从数据库建立，一直到现在的所有binlog 都保存完整。
如果能保证以上的条件，我们就可以实现在线增加同步数据库了；如果没有binlog，那么增加第一台同步数据库的时候，很肯定就需要将主数据库停机了。

接下来，我们配置好同步数据库的环境，但是在my.cnf 文件中增加一行:
skip-slave-start
这样在我们启动mysql 后，同步进程不会自动启动，我们需要使用slave start 指令来启动。

我们现在启动同步库上的MySQL:

```
#/etc/rc.d/init.d/mysqld start
```

然后我们连接到主数据库，执行flush log 命令

```
#mysql –h <ip> -u <username> -p<password>
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 3529641 to server version: 3.23.54-log

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

mysql>show master status;
+--------------+----------+--------------+------------------+
| File       | Position  | Binlog_do_db | Binlog_ignore_db |
+--------------+----------+--------------+------------------+
| xxxx-bin.044 | 300054371 | xxxxxxxxxx  |            |
+--------------+----------+--------------+------------------+
mysql>flush log;
mysql>show master status;
+--------------+----------+--------------+------------------+
| File       | Position  | Binlog_do_db | Binlog_ignore_db |
+--------------+----------+--------------+------------------+
| db68-bin.045 | 000000004 | xxxxxxxxxx  |            |
+--------------+----------+--------------+------------------+
mysql>quit
```

我们看到在执行了 flush log 指令之后 File, Position 列出现了一些变化。
这个时候我们查看一下主数据库上的binlog

```
#cd /log
#ls –al
-rw-rw----   1 mysql   mysql   1073741951 Mar 13 21:53 mysql-bin.035
.
.
.
-rw-rw----   1 mysql   mysql   1073741916 Mar 17 05:49 mysql-bin.043
-rw-rw----   1 mysql   mysql   1073741912 Mar 18 17:35 mysql-bin.044
-rw-rw----   1 mysql   mysql     2491041 Mar 19 15:35 mysql-bin.045
```

我们看到系统生成了一个新的mysql-bin.045 的log 文件，我们现在把编号001 到044 的log，从主数据库复制到同步数据库。有可能我们将001-034 的binlog 转移到备份设备上去了，所以我们只看到了035-045 的binlog。然后登陆到同步数据库，利用binlog 还原数据库。

```
#mysqlbinlog mysql-bin.001 | mysql
#mysqlbinlog mysql-bin.002 | mysql
.
.
.
#mysqlbinlog mysql-bin.044 | mysql
```

一定要按照 binlog 的编号完成以上操作，对于编号为 045 的 binlog 我们采用下面的方式还原数据，登陆到同步库

```
#mysql –h <ip> -u <username> -p<password>
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 3529641 to server version: 3.23.54-log

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

mysql>CHANGE MASTER TO
  MASTER_HOST='x.x.x.x',
  MASTER_USER='repl',
  MASTER_PASSWORD='********',
  MASTER_PORT=3306,
  MASTER_LOG_FILE='mysql-bin.045',
  MASTER_LOG_POS=4;
mysql>slave start;
mysql>show slave status;
```

现在编号为 045 的 binlog 已经开始同步了。然后我们修改 my.cnf 文件，去掉 skip-slave-start 这一行，下次我们再次启动 mysql 的时候，就会生效了。

事实上，我们只需要将 001-034 的 binlog 文件从备份设备上直接复制到同步库，然后通过 mysqlbinlog 恢复 001-034 的部分，然后设置 MASTER_LOG_FILE='mysql-bin.035'就可以了，但是这样会增加一点主数据库的负担，而且恢复的速度会稍慢一些。

2.已经有一台同步数据库 A 了，这台 A 数据库可以停机，这个时候再增加一台同步数据库 B。这个时候我们就把数据库 A 的 mysql 停掉

```
#/etc/init.d/mysqld stop
```

然后将 /data 目录下的文件原封不动的复制到数据库 B，然后我们直接启动数据库 B 的 Mysql 就可以了！

3.我的binlog 不完整，而且我也没有同步数据库
这种情况就只好停机了，不能实现在线增加同步数据库了

第五章 日志的管理与维护

本章将讲述 MySQL 的日志管理系统，以及如何备份 mysql 的日志文件，为系统的数据安全提供保障。

第*1*节 *MySQL* 是如何维护日志文件的

在 MySQL 的系统中存在多种日志文件，主要是因为 MySQL 在不断的发展中衍生出来的产物，目前 MySQL 的官方文档中，建议我们采用最新的日志文件格式：bin-log。

BINLOG 日志文件的建立我们在数据库的同步这一章节已经讲过了。
下面我们对 BINLOG 日志系统的组成做一下说明:
BINLOG 日志系统由两部分组成：xxxx-bin.index xxxx-bin.nnn
index 文件是一个文本文件，每一行是一个文件名，记录一个 xxxx-bin.nnn 的文件所在的位置。
xxxx-bin.nnn 文件是 mysql 的 binlog 文件的实体，采用二进制方式存储数据的信息。

当一个 xxxx-bin.nnn 文件的大小达到 max_binlog_size 后 mysql 就会生成一个新的 xxxx-bin.nnn 文件。

index 文件中的最后一个 xxxx-bin.nnn 文件被称作 Online Binlog，其他的被称作 Archived Binlog。

Archived Binlog 在生成后就不再会有任何改动了，但是可以被其他的同步数据库通过数据同步机制进行访问（读取）。
Online Binlog 是 mysql 正在使用的 binlog，mysql 在不断的追加信息，同时其他同步数据库也可能在读取这些信息。

第二节 与日志文件相关的命令

1.Flush Log 指令
强制 binlog 翻滚。
有时候我们需要强制系统将 binlog 写入新的文件，或者有其他的需要，我们就可以使用 FLUSH LOG，强制系统切换 binlog

2.Reset Master 指令

清除所有index 文件中列出的binlog 文件，并且重新开始记录新的binlog。
有时候我们搭建了一个数据库的平台，后来我们这套数据库的数据转移走了，现在的这套MySQL 平台要处理新的数据库了，这个时候我们就要对binlog 进行初始化，这个时候就需要使用Reset Master 指令清除所有的binlog 文件，同时初始化index 文件。这个操作是一个破坏性的操作，执行后现有的binlog 文件都会被清除

3.Reset Slave 指令
重置同步状态
有时候我们的一台同步数据库会停止对主数据库A 做同步，转而对数据库B 做同步，这个时候我们就要执行：slave stop; reset slave; 这个指令的序列，重置同步状态

4.mysqlbinlog 命令行工具
将binlog 文件还原成可以被阅读的SQL 语句。

第六章 数据的备份

本章将讲述如何对MySQL 的数据进行备份，以及不同的备份方式会对系统产生的影响。

第一节 *MySQL* 数据备份的基本知识

1.MySQL 管理数据库文件的方式
数据库：每一个数据库都对应文件系统上的一个目录
表文件：每一个表文件对于MyISAM 格式的数据库文件来说，是由3 个文件组成：表结构定义文件（*.frm），数据文件（*.MYD），索引文件（*.MYI）

2.数据备份的方式
离线备份（冷备）：停止MySQL 系统，直接复制MySQL 的数据库文件
在线备份（热备）：表级完全备份，库级完全备份
通过同步数据库进行备份（热备）

第二节 如何进行离线备份

离线备份的方式非常简单，只需要停止MySQL 系统，直接复制数据库文件到备份设备即可，离线备份是库级完全备份，数据之间的逻辑关系都被完全的保留，逻辑上没有任何差错。

第三节 如何进行在线备份

1.表级完全备份
a.BACKUP 指令完成备份操作，系统会在备份表的时候增加只读锁。
backup table tablename to '<path>';
指令完成后，在<path>目录中会产生两个文件*.MYD, *.frm
b.mysqldump 工具完成备份操作，系统会在备份表的时候增加只读锁。

mysqldump -u <username> -p<password> database tablename;

2.库级完全备份

a.给库中所有的表增加只读锁，然后通过backup 指令备份数据

lock tables tablename1, tablename2, ... read;

backup table tablename1, tablename2, ... to '<path>';

unlock tables;

指令完成后，在<path>目录中会产生两类文件*.MYD, *.frm

b.使用 mysqldump 工具完成备份操作，系统会在备份期间增加只读锁到所有的表。

mysqldump -u <username> -p<password> -l database

第四节 如何利用同步数据库进行备份

这是在线备份 MySQL 数据库唯一不锁表的方式，通过对同步数据库的数据进行冷备方式得到数据库的一个完整备份。

首先我们应该有一个专门用来备份 MySQL 数据库的同步数据库，这将对我们来说是非常方便的意见事情。

需要对数据库进行备份的时候只需要对同步数据库进行冷备就可以了。

第七章 数据库的恢复

本章将讲述如何利用备份数据以及 binlog 回复全部或者部分数据，以挽回由于硬件问题、误操作问题造成的数据损失。

第一节 直接恢复

直接复制*.frm，*.MYD，*.MYI 直接恢复表文件。

第二节 如何从 *mysqldump* 生成的文件进行数据恢复

mysqldump 生成的数据文件是一条条 SQL 语句，可以直接使用命令倒入数据，一般的我们在使用 mysqldump 备份数据的时候都会增加 --add-drop-table 参数，这样在 mysqldump 生成的第一条 SQL 语句会首先删除老的 table，然后再建立 table，倒入数据。

```
#cat <mysqldump file> | mysql -u <username> -p<password> database
```

第三节 如何使用 *restore* 指令恢复 *backup* 指令备份的数据

首先将 backup 生成的*.frm, *.MYD 从备份设备上恢复到数据库系统的本地磁盘。

然后执行 restore 指令，恢复数据库。

```
mysql>drop table oldtable;
mysql>restore table tablename1, tablename2, ... from '<path>';
```

第四节 如何使用同步数据恢复数据

首先将主数据库的存在问题的表文件删除

```
mysql>drop table tablename;
```

停止同步数据库的MySQL，将需要的恢复的表的*.MYD，*.MYI，*.frm 文件从同步库复制到主数据库，然后对表文件进行一次检查。

```
mysql>check table tablename;
```

第五节 从*binlog* 中恢复数据库数据

有时候我们做了一部分误操作，导致大量的数据的某一列，或者某一个时间段的数据出现了问题，但是因为这个误操作没有被发现，导致我们无法通过以上的方式恢复数据库数据的时候，我们可以通过binlog 来恢复数据，当然，首先我们要保证binlog 的完整。

首先我们应该确定产生误操作的SQL 语句的大概的构成。例如我们执行了这样一个SQL

```
mysql>update table set flag=1;
```

这里由于操作人员疏忽，忘记写where 子句了。

这里我们判断这个产生误操作的SQL 语句，是一条update 语句，相关表是table，相关列是flag 列。

接下来我们要确定的是这条误操作语句执行的时间：

```
#mysqlbinlog xxxx-bin.001 | grep "update" | grep "table" | grep "flag"
#mysqlbinlog xxxx-bin.002 | grep "update" | grep "table" | grep "flag"
.
.
.
#mysqlbinlog xxxx-bin.nnn | grep "update" | grep "table" | grep "flag"
```

直到确定产生错误的SQL 语句所在的binlog，假设我们这里发现是在编号为040 的binlog

我们现在将xxxx-bin.040 倒出到文本文件，以确定执行的时间。

```
#mysqlbinlog xxxx-bin.040 > 040.sql
#vi 040.sql
```

然后我们查找误操作语句，我们会看到类似如下的信息：

```
# at 1137
#030308 0:16:14 server id 1 Query thread_id=451064579 exec_time=0
error_code=0
SET TIMESTAMP=1047053774;
UPDATE table SET flag=1;
```

这里我们就可以确定时间了，20030308 00:16:14。
然后我们将出现错误的这段SQL语句删除，存盘。

然后找到在20030308 00:16:14之前的最近一次数据完整备份，确定备份完成的时间，假设时间是20030301 03:00:00。

我们现在将 20030301 03:00:00 到 20030308 00:16:14 这段时间之间的SQL从binlog中抽取出来，通过binlog的日期，我们确定20030301 03:00:00这个时间点在编号为038的binlog里面。

```
#mysqlbinlog xxxx-bin.038 > 038.sql
#vi 038.sql
```

在vi编辑器里面我们删除20030301 03:00:00之前的所有信息，存盘。

我们将数据库使用20030301 03:00:00的备份恢复，然后执行

```
#cat 038.sql | mysql -u <user> -p<password>
#mysqlbinlog xxxx-bin.039 | mysql -u <user> -p<password>
# cat 040.sql | mysql -u <user> -p<password>
#mysqlbinlog xxxx-bin.041 | mysql -u <user> -p<password>
#mysqlbinlog xxxx-bin.042 | mysql -u <user> -p<password>
```

一直将所有编号大于041的binlog都倒入数据库，恢复宣告完成。

下面我们需要对系统做一些调整，让系统仍然是一个可以方便维护的系统
重置binlog日志系统：

```
mysql>reset master;
```

对数据进行冷备，作为以后binlog日至系统恢复根基

如果我们只想使用binlog恢复一张表，而不是整个库的时候，我们只需要使用下面的指令

```
#mysqlbinlog xxxx-bin.038 | grep "table" > 038.sql
#mysqlbinlog xxxx-bin.041 | grep "table" | mysql -u <user> -p<password> database
```

注意：一般的来说，使用binlog对数据进行恢复的时候mysql系统将不允许提供服务，否则将会出现恢复失败的可能。

第八章 **MySQL** 索引文件的维护

本章将讲述与MySQL数据检索效率联系紧密地索引文件，MySQL如何使用索引文件来加速搜索进程，以及索引文件的维护。

第一节 关于*MySQL*索引文件的基础知识

MySQL的索引分为 普通索引，唯一性索引，主键索引，全文索引。
每种索引又包括单列索引，以及多列索引。
MySQL的索引算法基本上采用B*树完成

第二节 索引文件的作用

索引文件在数据检索的过程中起着至关重要的作用，他可以加速数据检索的速度。但是索引文件一旦损坏，会导致数据检索效率大大降低，而且有可能比没有索引的时候还要低。

但是并不是所有的SQL语句都能够使用索引文件加快检索速度，如何利用索引文件加快检索速度将在SQL语句的优化中详细讲述。

索引文件虽然可以加快检索速度，但是由于数据的每次更新，都需要修改索引数据，所以，如果索引太多，就会降低数据更新、添加的速度。由于数据的更新、添加操作都会锁表，导致大量检索操作等待，系统就会表现繁忙。

因此高效的利用索引文件是非常重要的。

第三节 索引文件的检查与修复

如果我们发现数据库的负载出现异常，或者操作某张表文件的SQL 语句过量堆积，这个时候我们就应该对这样表的表文件进行检查了。
索引文件的检查分为下面的几步:
1.使用 myisamchk table.MYI 粗略检查。由于 MySQL 存在 Cache 的管理，所以在使用 Myisamchk 工具之前，应该使用 Flush table 指令刷新表缓存。
2.如果发现错误，或者希望使用 myisamchk -r or -o 进行进一步检查的时候，就说明索引文件出现了损坏。
3.如果粗略检查还是不能很好的确定问题，我们就要用到 check table tablename 指令来检查表文件了。
4.使用 repair table tablename 修复索引文件

粗略检查是不会锁表的，所以为了不影响在线业务的前提下，应首先粗略检查表文件。
使用 Check table 以及 Repair table 指令会锁定正在修理的表文件。