常见日志类型：

错误日志

二进制日志

普通查询日志

慢查询日志

中继日志(复制中的从服务器)

一般启动默认没有日志，应该启用二进制日志(非常关键)和错误日志

**错误日志：**

启动关闭过程中的信息

运行中的错误信息

事件调度运行一个事件时产生的信息

从服务器启动从服务进程时产生的信息

SHOW VARIABLES LIKE ‘%err%’ 查看错误日志信息

如果想把警告信息记录到错误日志中，**log\_warnings**选项启用即可，默认启用。

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**二进制日志**：类似于重做日志

可以做即时点(point-in-time)恢复；复制；一般记载DML、DDL等让数据库发生改变的语句。有时候这样的执行复制会带来不确定的结果，例如now();一般要二进制文件和数据文件分开,默认在数据目录。

SHOW VARIABLES LIKE ‘%bin%’

SHOW BINARY LOGS; //所有二进制日志

SHOW MASTER STATUS; // 当前正在使用的二进制文件

FLUSH LOG 就会分割新的二进制日志文件。

如果想要**改变二进制日志的文件路径**可以修改配置文件的mysqld段，重启服务器生效：

log-bin = /data/logs/binary/changelog //目录属主属组注意

log-bin-index = /data/logs/relay/binarylog.index //不指定为上面的默认路径

二进制日志的格式：(自动判断时是混合的)

基于语句的：

基于行：

配置文件中的binlog\_format指定

还有一个mysqlbinlog命令专门查看二进制日志的



使用样例：时间是绝对的 位置是相对的

* + mysqlbinlog mysql-bin.000001 \

--start-datetime="2011-03-05 17:34:40" \

--stop-datetime="2011-03-05 17:34:56" > sixteensecs.sql

* + mysqlbinlog --read-from-remote-server -uuser -p \

mysql-bin.000001 -h 192.168.0.182 -P 3306 \

--start-position=932 --stop-position=1132 > remote000001.sql



\*\*\*\*\*\*\*\*\*\*\*\*

普通(查询)日志：

因为压力比较大，所以一般查询日志不打开。

SHOW VARIABLES LIKE ‘general\_log’

慢查询日志：

SHOW VARIABLES LIKE ‘slow%’;

打开慢查询日志(短时间有效)，永久有效改配置文件：

SET GLOBAL VARIABLES slow\_query\_log=’ON’ 这种不能开启，应该是：

SET GLOBAL slow\_query\_log=’ON’

这种日志到还能记录到表中，SHOW GLOBAL VARIABLES LIKE ’log\_out%’查看记录到哪里。

日志滚动：一般对二进制日志有意义。不滚动会达到1G

FLUSH LOGS;

完全备份完成的二进制日志也可以给清除了，

mysql> PURGE BINARY LOGS TO ’mysql-bin.00007’;

mysql> PURGE BINARY LOGS BEFORE ’2011-3-5 23:00:00’;

###########################################################################

备份和恢复：

//最好使用一个单独的磁盘挂载到一个目录来备份；

备份类型：

热备(hot backup), 读/写操作都能正常进行

温备(warm backup), 读操作可以进行，写操作不行

冷备(cold backup), 关闭mysql服务

备份类型:

逻辑备份：备份的就是一些语句；纯文本信息, mysqldump

容易进行恢复

有很强的兼容性

独立于存储引擎

平台无关

CPU周期

备份的结果文件比较大

浮点数会丢失精度

物理备份：文件(数据文件，日志文件)

备份简单；速度快

备份的结果会更大

跨平台取决于存储引擎

备份类型：

完全备份：

增量备份：

差异备份：第一天的；前二天；前三天

备份什么：

二进制日志、事务日志

**innodb的事务日志文件ib\_logfile0**,新的版本中这个大小可以扩大。**早起版本这个如果过大会导致启动过慢**

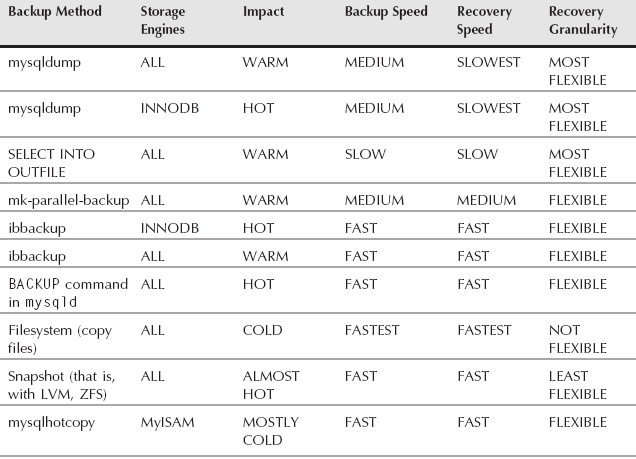
代码：触发器、存储例程

配置文件： my.cnf

从服务器的master.info文件；中继日志

cron任务、关联到MySQL的用户和组、管理脚本

备份工具：



mk-parallel-backup被收购后不提供了；

ibbackup很好，收费

BACKUP不支持了

主要讲mysqldump、select 和LVM这几种。

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

mysqldump：

mysqldump的逻辑备份也可以指定备份到哪里--where，便于读和文本处理

还可以远程备份

注意事项：

//非事物引擎备份前先锁定

//--master-data 记录备份在二进制文件中的位置

选项：

dbname\_ [tb1] [tb2] 指定库的指定表

--databases 备份指定的数据库

-- where WHERE条件(用的少) 备份指定的点

--event 数据库中的enevt也备份

--extended-insert 批量形式往里放，默认的不用选

--flush-logs 完全备份了就用不到二进制文件了，备份前最好flush一遍。

--lock-all-tables 锁定然后备份

--lock-tables

--no-data 只备份表结构不备份数据

--master-data=n 同时备份二进制文件的名字和位置，=2? 复制的时候还有

--replace 都保存replace语句而不是insert，用不多

--routines 备份存储过程，存储函数

--triggers 备份触发器

--single-transacton 这就是innodb热备的

小的库直接全部备份：

mysqldump -uroot -hlocalhost -p12345 --all-databases > /back/’date +%F-%H-%M-%S’.sql

mysqldump jiaowu courses > /backup/courses.sql

这样可能会造成不完整，myisam使用锁定再温备，innodb热备选项

FLUSH TABLES WITH READ LOCK;

mysqldump --databases jiaowu > /backup/courses.sql

UNLOCK TABLES;

直接指定就不用那么麻烦了：

mysqldump --databases jiaowu --lock-all-tables > /backup/courses.sql

innodb的热备选项：

mysqldump --databases jiaowu --single-transaction > /backup/a.sql

纯文本还可以压缩存放：

bzip2 alldatebase-2012-0101.sql

\*\*\*\*\*\*\*

恢复：

mysql -uroot -hlocalhost -p < jaiowu.sql

单张表的还原：指定还原的库：

#mysql jiaowu < backup.sql;

\*\*\*\*\*\*\*\*\*\*\*\*\*

SELECT方式：保存的纯数据

* + SELECT \* INTO OUTFILE ’/tmp/t1.txt’ FROM t1;
  + LOAD DATA INFILE ’/tmp/t1.txt’ INTO TABLE t1;

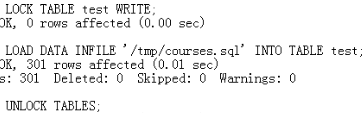
必须是有写入权限的位置。



这是纯文本文件

导入时先建表结构：





\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

增量备份和恢复：备份binlog

完全备份加binlog实现即时点恢复：

例如：某个表不小心执行了drop命令，如何恢复？(都是innodb的)

先备份完全备份，

mysqldump --databases jiaowu --single-transaction --flush-logs > /backup/jiaowu-bin.sql

这时是热备的，不影响备份；这时候如果插入了数据并且删除了表；

在增量备份时指定备份的位置：(即drop命令的位置)

先查看drop命令的位置：



在这里找到drop命令 //如果很多的话，怎么找到这条命令？



然后这之前的记录保存下来：



这样里面保存的就没有这一行了

恢复：



再恢复增量数据



就好了

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

恢复：

**恢复的时候最好把二进制日志给关掉**，否则恢复过程会记录到二进制日志中去。

修改全局变量无法实现，但是修改会话变量可以实现：SET SQL\_LOG\_BIN=0

然后再在当前会话恢复：>SOURCE /path/to/backup.sql

完成后使用SET SQL\_LOG\_BIN=1重新打开

所以到时候要考虑很多细小的问题。

物理备份：

离线：cp 这中对于myisa引擎有效，但是对于innodb不生效，所有的表都在同一个存储空间中

在线：LVM

先：mysql>FLUSH TABLES WITH READ LOCK; 这个施加的过程可能会花点时间

SHOW MASTER STATUS\G或者使用FLUSH LOGS 记录二进制日志的位置！

lvcreate -L大小 估计会变化的大小 -n名字 -s快照卷 -p权限 逻辑卷名

lvcreate -L 50M -n data-snap -s -p r /dev/myvg/mydata

mysql>UNLOCK TABLES;

mkdir /snmp

mount /dev/myvg/data-snap /snap

cp /mnt/ /backup

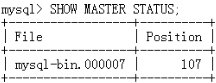
umount /mnt

lvremove /dev/myvg/data-snap

这个的即时点恢复还是需要借助二进制日志

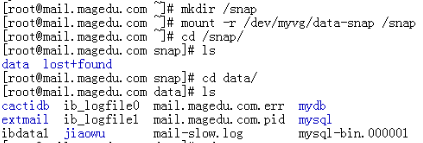
















建议备份的使用把二进制文件的名称和位置记录下来！

\*\*\*\*\*\*\*\*\*

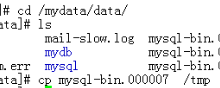
基于物理备份和二进制日志实现的还原：

如果不小心删除了，恢复：

还原只能冷还原，



将上次冷备份之后新生成的binlog专门复制到另外的地方，因为一还原都会被覆盖，这里实验最重要的新生成的是0007





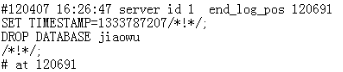
注意属组属主





这时再查看是恢复到冷备的记录。然后再用binlog恢复

这有一个二进制文件不能再进行一次，在指定时间找到drop所在的位置



然后就可以直接导入到mysql中了，



为了不再次生成二进制日志先进去关了





再次开启即可



