[**mysql数据库备份及恢复命令mysqldump,source的用法**](http://www.cppblog.com/niewenlong/archive/2014/08/07/207927.html)

**还原一个数据库:mysql -h localhost -u root -p123456 www<c:/www.sql**

**备份一个数据库:mysqldump -h localhost -u root -p123456 www > d:/www2008-2-26.sql**

**//以下是在程序中进行测试**

**//$command = "mysqldump --opt -h $dbhost -u $dbuser -p $dbpass $dbname | gzip > $backupFile";  
$command="mysqldump -h localhost -u root -p123456 guestbook > guestbook2-29.sql";  
system($command);  
echo "success";**

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

**备份MySQL数据库的命令**

mysqldump -hhostname -uusername -ppassword databasename > backupfile.sql

**备份MySQL数据库为带删除表的格式**   
备份MySQL数据库为带删除表的格式，能够让该备份覆盖已有数据库而不需要手动删除原有数据库。

mysqldump -–add-drop-table -uusername -ppassword databasename > backupfile.sql

**直接将MySQL数据库压缩备份**

mysqldump -hhostname -uusername -ppassword databasename | gzip > backupfile.sql.gz

**备份MySQL数据库某个(些)表**

mysqldump -hhostname -uusername -ppassword databasename specific\_table1 specific\_table2 > backupfile.sql

**同时备份多个MySQL数据库**

mysqldump -hhostname -uusername -ppassword –databases databasename1 databasename2 databasename3 > multibackupfile.sql

**仅仅备份数据库结构**

mysqldump –no-data –databases databasename1 databasename2 databasename3 > structurebackupfile.sql

**备份服务器上所有数据库**

mysqldump –all-databases > allbackupfile.sql

**还原MySQL数据库的命令**

mysql -hhostname -uusername -ppassword databasename < backupfile.sql

**还原压缩的MySQL数据库**

gunzip < backupfile.sql.gz | mysql -uusername -ppassword databasename

**将数据库转移到新服务器**

mysqldump -uusername -ppassword databasename | mysql –host=\*.\*.\*.\* -C databasename

**几个常用用例：**

1.导出整个数据库  
mysqldump -u 用户名 -p 数据库名 > 导出的文件名  
mysqldump -u root -p dataname >dataname.sql   
这个时候会提示要你输入root用户名的密码,输入密码后dataname数据库就成功备份在mysql/bin/目录中.

2.导出一个表  
mysqldump -u 用户名 -p 数据库名 表名> 导出的文件名  
mysqldump -u root -p dataname users> dataname\_users.sql

3.导出一个数据库结构  
mysqldump -u wcnc -p -d –add-drop-table smgp\_apps\_wcnc >d:/wcnc\_db.sql   
-d 没有数据 –add-drop-table 在每个create语句之前增加一个drop table

4.导入数据库  
常用source 命令  
进入mysql数据库控制台，  
如mysql -u root -p

mysql>use 数据库

然后使用source命令，后面参数为脚本文件（如这里用到的.sql）  
mysql>source d:/wcnc\_db.sql

**mysqldump支持下列选项：**   
–add-locks  
在每个表导出之前增加LOCK TABLES并且之后UNLOCK TABLE。(为了使得更快地插入到MySQL)。

–add-drop-table  
在每个create语句之前增加一个drop table。

–allow-keywords  
允许创建是关键词的列名字。这由表名前缀于每个列名做到。

-c, –complete-insert  
使用完整的insert语句(用列名字)。

-C, –compress  
如果客户和服务器均支持压缩，压缩两者间所有的信息。

–delayed  
用INSERT DELAYED命令插入行。

-e, –extended-insert  
使用全新多行INSERT语法。（给出更紧缩并且更快的插入语句）

-#, –debug[=option\_string]  
跟踪程序的使用(为了调试)。

–help  
显示一条帮助消息并且退出。

–fields-terminated-by=…

–fields-enclosed-by=…

–fields-optionally-enclosed-by=…

–fields-escaped-by=…

–fields-terminated-by=…

这些选择与-T选择一起使用，并且有相应的LOAD DATA INFILE子句相同的含义。  
LOAD DATA INFILE语法。

-F, –flush-logs  
在开始导出前，洗掉在MySQL服务器中的日志文件。

-f, –force,  
即使我们在一个表导出期间得到一个SQL错误，继续。

-h, –host=..  
从命名的主机上的MySQL服务器导出数据。缺省主机是localhost。

-l, –lock-tables.  
为开始导出锁定所有表。

-t, –no-create-info  
不写入表创建信息(CREATE TABLE语句）

-d, –no-data  
不写入表的任何行信息。如果你只想得到一个表的结构的导出，这是很有用的！

–opt  
同–quick –add-drop-table –add-locks –extended-insert –lock-tables。  
应该给你为读入一个MySQL服务器的尽可能最快的导出。

-pyour\_pass, –password[=your\_pass]  
与服务器连接时使用的口令。如果你不指定“=your\_pass”部分，mysqldump需要来自终端的口令。

-P port\_num, –port=port\_num  
与一台主机连接时使用的TCP/IP端口号。（这用于连接到localhost以外的主机，因为它使用 Unix套接字。）

-q, –quick  
不缓冲查询，直接导出至stdout；使用mysql\_use\_result()做它。

-S /path/to/socket, –socket=/path/to/socket  
与localhost连接时（它是缺省主机)使用的套接字文件。

-T, –tab=path-to-some-directory  
对于每个给定的表，创建一个table\_name.sql文件，它包含SQL CREATE 命令，和一个table\_name.txt文件，它包含数据。注意：这只有在mysqldump运行在mysqld守护进程运行的同一台机器上的时候才工 作。.txt文件的格式根据–fields-xxx和 –lines–xxx选项来定。

-u user\_name, –user=user\_name  
与服务器连接时，MySQL使用的用户名。缺省值是你的Unix登录名。

-O var=option, –set-variable var=option  
设置一个变量的值。可能的变量被列在下面。

-v, –verbose  
冗长模式。打印出程序所做的更多的信息。

-V, –version  
打印版本信息并且退出。

-w, –where=’where-condition’  
只导出被选择了的记录；注意引号是强制的！  
“–where=user=’jimf’” “-wuserid>1″ “-wuserid<1″

最常见的mysqldump使用可能制作整个数据库的一个备份：  
mysqldump –opt database > backup-file.sql

但是它对用来自于一个数据库的信息充实另外一个MySQL数据库也是有用的：  
mysqldump –opt database | mysql –host=remote-host -C database  
由于mysqldump导出的是完整的SQL语句，所以用mysql客户程序很容易就能把数据导入了：

shell> mysqladmin create target\_db\_name  
shell> mysql target\_db\_name < backup-file.sql  
就是  
shell> mysql 库名 < 文件名

mysqldump--导出工具

mysqldump备份还原和mysqldump导入导出语句大全详解

**mysqldump备份**：

mysqldump -u用户名 -p密码 -h主机 数据库 a -w "sql条件" --lock-all-tables > 路径

案例：

mysqldump -uroot -p1234 -hlocalhost db1 a -w "id in (select id from b)" --lock-all-tables > c:\aa.txt

**mysqldump还原**：

mysqldump -u用户名 -p密码 -h主机 数据库 < 路径

案例：

mysql -uroot -p1234 db1 < c:\aa.txt

**mysqldump按条件导出**：

mysqldump -u用户名 -p密码 -h主机 数据库  a --where "条件语句" --no-建表> 路径

mysqldump -uroot -p1234 dbname a --where "tag='88'" --no-create-info> c:\a.sql

**mysqldump按导入**：

mysqldump -u用户名 -p密码 -h主机 数据库 < 路径

案例：

mysql -uroot -p1234 db1 < c:\a.txt

**mysqldump导出表**：

mysqldump -u用户名 -p密码 -h主机 数据库 表

案例：

mysqldump -uroot -p sqlhk9 a --no-data

讲一下 mysqldump 的一些主要参数

**--compatible=name**  
它告诉 mysqldump，导出的数据将和哪种数据库或哪个旧版本的 MySQL 服务器相兼容。值可以为 ansi、mysql323、mysql40、postgresql、oracle、mssql、db2、maxdb、no\_key\_options、 no\_tables\_options、no\_field\_options 等，要使用几个值，用逗号将它们隔开。当然了，它并不保证能完全兼容，而是尽量兼容。   
**--complete-insert，-c**  
导出的数据采用包含字段名的完整 INSERT 方式，也就是把所有的值都写在一行。这么做能提高插入效率，但是可能会受到 max\_allowed\_packet 参数的影响而导致插入失败。因此，需要谨慎使用该参数，至少我不推荐。   
**--default-character-set=charset**  
指定导出数据时采用何种字符集，如果数据表不是采用默认的 latin1 字符集的话，那么导出时必须指定该选项，否则再次导入数据后将产生乱码问题。  
**--disable-keys**  
告诉 mysqldump 在 INSERT 语句的开头和结尾增加 /\*!40000 ALTER TABLE table DISABLE KEYS \*/; 和 /\*!40000 ALTER TABLE table ENABLE KEYS \*/; 语句，这能大大提高插入语句的速度，因为它是在插入完所有数据后才重建索引的。该选项只适合 MyISAM 表。  
**--extended-insert = true|false**默认情况下，mysqldump 开启 --complete-insert 模式，因此不想用它的的话，就使用本选项，设定它的值为 false 即可。   
**--hex-blob**  
使用十六进制格式导出二进制字符串字段。如果有二进制数据就必须使用本选项。影响到的字段类型有 BINARY、VARBINARY、BLOB。  
**--lock-all-tables，-x**  
在开始导出之前，提交请求锁定所有数据库中的所有表，以保证数据的一致性。这是一个全局读锁，并且自动关闭 --single-transaction 和 --lock-tables 选项。  
**--lock-tables**  
它和 --lock-all-tables 类似，不过是锁定当前导出的数据表，而不是一下子锁定全部库下的表。本选项只适用于 MyISAM 表，如果是 Innodb 表可以用 --single-transaction 选项。  
**--no-create-info，-t**  
只导出数据，而不添加 CREATE TABLE 语句。  
**--no-data，-d**  
不导出任何数据，只导出数据库表结构。  
**--opt**  
这只是一个快捷选项，等同于同时添加 --add-drop-tables --add-locking --create-option --disable-keys --extended-insert --lock-tables --quick --set-charset 选项。本选项能让 mysqldump 很快的导出数据，并且导出的数据能很快导回。该选项默认开启，但可以用 --skip-opt 禁用。注意，如果运行 mysqldump 没有指定 --quick 或 --opt 选项，则会将整个结果集放在内存中。如果导出大数据库的话可能会出现问题。  
**--quick，-q**  
该选项在导出大表时很有用，它强制 mysqldump 从服务器查询取得记录直接输出而不是取得所有记录后将它们缓存到内存中。  
**--routines，-R**  
导出存储过程以及自定义函数。  
**--single-transaction**  
该选项在导出数据之前提交一个 BEGIN SQL语句，BEGIN 不会阻塞任何应用程序且能保证导出时数据库的一致性状态。它只适用于事务表，例如 InnoDB 和 BDB。  
本选项和 --lock-tables 选项是互斥的，因为 LOCK TABLES 会使任何挂起的事务隐含提交。  
要想导出大表的话，应结合使用 --quick 选项。   
**--triggers**  
同时导出触发器。该选项默认启用，用 --skip-triggers 禁用它。  
其他参数详情请参考手册，我通常使用以下 SQL 来备份 MyISAM 表：  
/usr/local/mysql/bin/mysqldump -uyejr -pyejr "  
--default-character-set=utf8 --opt --extended-insert=false "  
--triggers -R --hex-blob -x db\_name > db\_name.sql  
使用以下 SQL 来备份 Innodb 表：  
/usr/local/mysql/bin/mysqldump -uyejr -pyejr "  
--default-character-set=utf8 --opt --extended-insert=false "  
--triggers -R --hex-blob --single-transaction db\_name > db\_name.sql  
另外，如果想要实现在线备份，还可以使用 --master-data 参数来实现，如下：  
**/usr/local/mysql/bin/mysqldump -uyejr -pyejr "**  
--default-character-set=utf8 --opt --master-data=1 "  
--single-transaction --flush-logs db\_name > db\_name.sql  
它只是在一开始的瞬间请求锁表，然后就刷新binlog了，而后在导出的文件中加入CHANGE MASTER 语句来指定当前备份的binlog位置，如果要把这个文件恢复到slave里去，就可以采用这种方法来做。

1.2 还原  
用 mysqldump 备份出来的文件是一个可以直接倒入的 SQL 脚本，有两种方法可以将数据导入。  
直接用 mysql 客户端   
例如：  
**/usr/local/mysql/bin/mysql -uyejr -pyejr db\_name < db\_name.sql**  
用 SOURCE 语法 （实验不成功！！！）   
其实这不是标准的 SQL 语法，而是 mysql 客户端提供的功能，例如：  
SOURCE /tmp/db\_name.sql;  
这里需要指定文件的绝对路径，并且必须是 mysqld 运行用户(例如 nobody)有权限读取的文件。

#导出某个数据库－－结构+数据

shell>mysqldump -h192.168.161.124 -uroot -pxxxxxx --opt db\_name |gzip -9 > /db\_bakup/db\_name.gz

#导出某个数据库的表－－结构+数据+函数+存储过程

shell>mysqldump -h192.168.161.124 -uroot -pxxxxxx --opt -R db\_name |gzip -9 > /db\_backup/db\_name.gz

#导出多个数据库

shell>mysqldump -h192.168.161.124 -uroot -pxxxxxx --opt --databases db\_name1 db\_name2 db\_name3 |gzip -9 > /db\_backup/mul\_db.gz

#导出所有的数据库

shell>mysqldump -h192.168.161.124 -uroot -pxxxxxx --opt --all-databases |gzip -9 > /db\_bak/all\_db.gz

#导出某个数据库的结构

shell>mysqldump -h192.168.161.124 -uroot -pxxxxxx --opt --no-data db\_name|gzip -9 > /db\_bak/db\_name.strcut.gz

#导出某个数据库的数据

shell>mysqldump -h192.168.161.124 -uroot -pxxxxxx --opt --no-create-info db\_name|gzip -9 > /db\_bak/db\_naem.data.gz

#导出某个数据库的某张表

shell>mysqldump -h192.168.161.124 -uroot -pxxxxxx --opt db\_name tbl\_name |gzip -9 > /db\_bak/db\_name.tal\_name.gz

# 导出某个数据库的某张表的结构

shell>mysqldump -h192.168.161.124 -uroot -pxxxxxx --opt --no-data db\_name tal\_name | gzip -9 > /db\_bak/db\_name.tal\_name.struct.gz

#导出某个数据库的某张表的数据

shell>mysqldump -h192.168.161.124 -uroot -pxxxxxx --opt --no-create-info db\_name tbl\_name | gzip -9 > /db\_bak/db\_name.tbl\_name.data.gz

##--opt==--add-drop-table + --add-locks + --create-options + --disables-keys + --extended-insert + --lock-tables + --quick + --set+charset

##默认使用--opt，--skip-opt禁用--opt参数