mysql 总结

1.1 数据库的种类

按照早期的数据库理论，比较流行的数据库模型有三种，分别为层次式数据库，网络式数据库和关系型数据库，而在当今的互联网中，最常用的数据库模型主要是两种，即关系型数据库和非关系型数据库

1.1.1 关系型数据库介绍

1、关系数据库冲突

虽然网状数据库和层次数据库已经很好的解决了数据集中和共享问题，但是在数据独

立性和抽象级别上有很大欠缺，用户在对这两种数据库进行存储时，仍然需要明确数据的存储结构，指出存储路径，而关系型数据库就可以较好的解决这些问题

2、关系型数据库介绍

关系型数据库模式是把复杂的数据结构归结为简单的二元关系（及即二维表格形式），例如老男孩教育某一期的学生关系就是一个二元关系，在关系型数据库中，对数据的操作几乎全部建立在一个或多个关系表格上，通过对这些关联的表格分类、合并、连接或选取等运算来实现数据的管理

3、关系型数据库表格之间的关系举例

表 1：学生表		
学号	姓名	年龄
S001	oldboy	28
S002	oldgirl	30
S003	张三	26
S004	李四	22

表 2：课程表		
课程号	课程名	学分
C1001	linux 运维	25
C1002	linux 架构	25
C1003	python 运维	25
C1004	mysql 数据库	25

表 3：学生选课表		
学号	课程号	成绩
S001	C1004	80
S002	C1002	90
S003	C1001	95
S004	C1003	70

1.1.2 非关系型数据库介绍

非关系型数据库也被称为 NOSQL 数据库，NOSQL 的本意是 NOT ONLY SQL，指的是非关系型数据库，NOSQL 是为了高性能，高并发而生的，NOSQL 典型产品 memcached（纯内存），redis（持久化缓存），mongodb

1.1.3 非关系型数据库种类

1 键值（key-value）存储数据库

键值数据库就类似传统语言中使用的哈希表，可以通过 key 来添加、查询或者删除数据，因此使用 key 主键访问，所以会获得很高的性能及扩展性

键值（key-value）数据库主要是使用一个哈希表，这个表中有一个特定的键和一个指针指向特定的数据，key/value 模型对于 IT 系统来说的优势在与简单、易部署、高并发

典型产品：memcached、redis

2、列存储（column-oriented）数据库

列存储数据将数据存储在列族（column family）中，一个列族存储经常被一起查询的相关数据，举个例子，如果我们有一个 person 类，我们通常会一起查询他们的姓名和年龄而不是薪资，这种情况下，姓名和年龄就会被放入一个列族中，而薪资则在另一个列族中

这部分数据库通常是用来应对分布式存储的海量数据，键仍然存在，但是他们的特点

是指向了多个列，这些列是由列家族安排的

典型产品：cassandra，hbase

3、面向文档（document-oriented）数据库

文档型数据库的灵感是来自于 lotus notes 办公室软件的，而且它同第一键值存储相类似，改类型的数据模式是版本化的文档，半结构化的文档以特定的格式存储，比如 JSON。文档型数据库可以看作是键值数据库的升级版，允许之间嵌套键值，而且文档型数据库比键值数据库的查询效率更高

面向文档数据库会将数据以文档的形式存储，每个文档都是自包含的数据单元，是一系列数据项的集合，每个数据项都有一个名称与对应的值，值即可以是简单的数据类型，如字符串、数字和日期等；也可以是复杂的类型。如有序列表和关联对象。数据存储的最小单位是文档，同一个表中存储的文档属性可以是不同的，数据可以使用 XML,JSON 或者 JSONB 等多种形式存储

典型产品：mongodb、couchdb

4、图形（graph）数据库

图形数据库允许我们将数据以图的方式存储，实体会被称作为顶点，而实体之间的关系会被作为边。

典型产品：neo4j、infogrid

1.1.4 关系型数据库产品介绍

1、oracle 数据库

主要应用范围：传统大企业，大公司，政府，金融，证券等等

2、mysql 数据库

主要应用范围：互联网领域，大中型网站，游戏公司，电商平台等等

3、mariadb 数据库

是 mysql 数据库的一个分支

4、SQL server 数据库

是微软公司开发的大型数据库系统，只能运行在 windows 平台下

1.1.5 常用非关系型数据库产品介绍

1、memcached（key-value）

memcached 是一个开源的、高性能的、具有分布式内存对象的缓存系统。通过它可以减轻数据库负载，加速动态的 web 应用。

缓存一般用来保存一些经常被存取的对象或数据（例如,浏览器会把经常访问的网页缓存起来一样），通过缓存来存取对象或数据要比在磁盘上存取快很多，前者是内存，后者是磁盘。memcached 是一种纯内存缓存系统，把经常存取的对象或数据缓存在 memcached 的内存中，这些被缓存的数据被程序通过 API 的方式存取，memcached 里面的数据就像一张巨大的 HASH 表，数据以 key-value 对的方式存在，memcached 通过缓存经常被存取的对象或数据，从而减轻频繁读取数据库的压力，提高网站的响应速度，构建出速度更快的可扩展的 web 应用。官方：http://memcached.org/

2、redis

和 memcached 类似，redis 也是一个 key-value 型存储系统，但 redis 支持的存储 value 类型相对更多，包括 string（字符串）、list（链表）、set（集合）和 zset（有序集合）等，redis 的数据都是缓存在内存中，区别是 redis 会周期性的把更新的数据写入磁盘或者把修改操作写入追加的记录文件

3、mongodb（document-oriented）

mongodb 是一个介于关系数据库和非关系数据库之间的产品，是非关系数据库当中

功能最丰富，最像关系数据库的。他支持的数据结构非常松散，类似 json 的 bjson 格式，因此可以存储比较复杂的数据类型，mongodb 最大的特点是支持的查询语言非常强大，其语法有点类似面向对象的查询语言，几乎可以实现类似关系数据库单表查询的绝大部分功能，而且还支持对数据建立索引，它的特点是高性能、易部署、易使用、存储数据非常方便

主要功能特性：

1、 面向集合存储，易存储类型的数据
2、 模式自由
3、 支持动态查询
4、 支持完全索引，包含内部对象
5、 支持查询
6、 支持复制和故障恢复
7、 使用高效的二进制数据存储，包括大型对象（如视频等）
8、 自动处理碎片，以支持云计算层次的扩展性
9、 支持 RUBY，PYTHON,JAVA,C++,PHP 等多种语言
10、文件存储格式为 BSON（一种 JSON 的扩展）
11、可通过网络访问

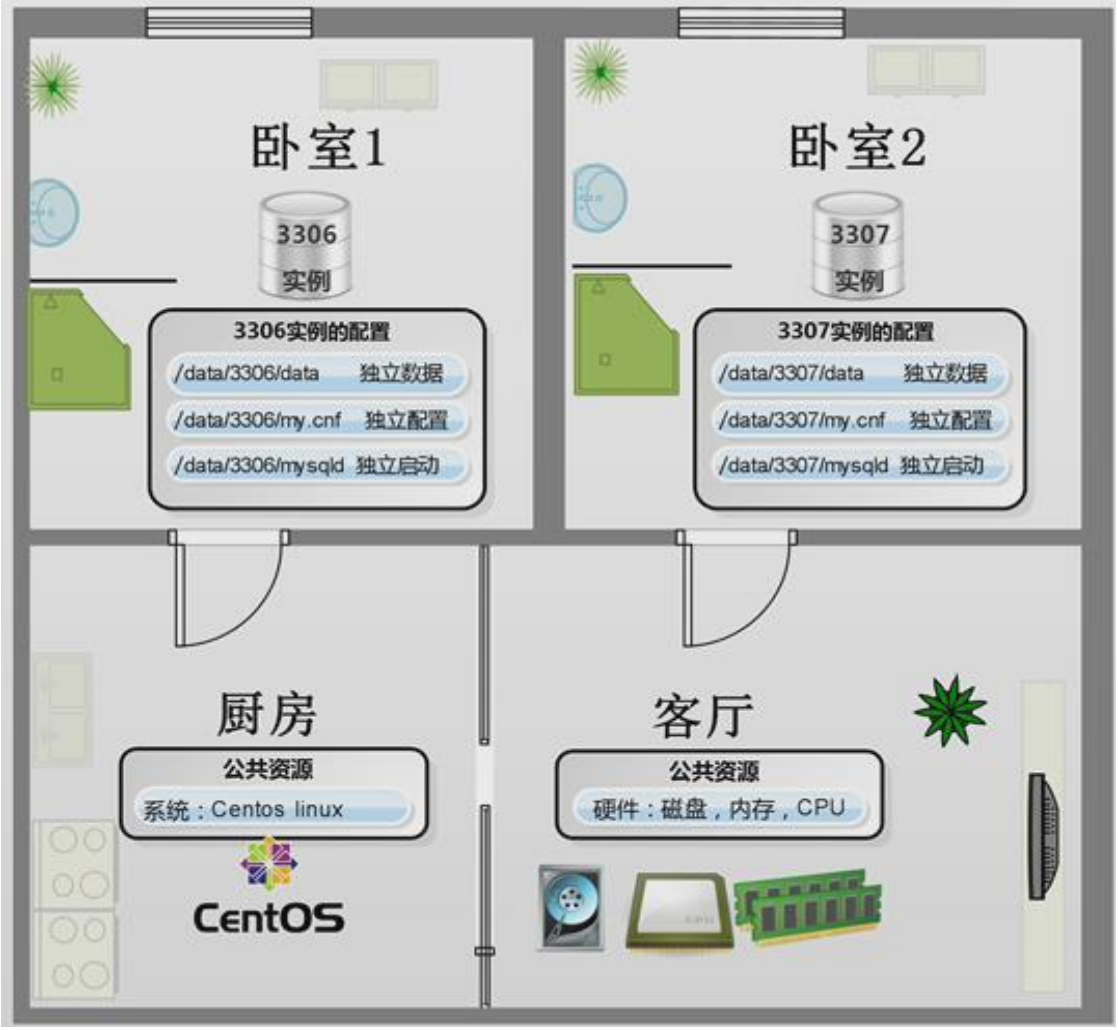
1.2 mysql 多实例安装实战

1.2.1 什么是 mysql 多实例？

简单的说，mysql 多实例就是在一台服务器上同时开启多个不同的服务端口（如 3306、3307），同时运行多个 mysql 服务进程，这些服务进程通过不同的 socket 监听不同的服务端口来提供服务

这些 mysql 多实例公用一套 mysql 安装程序，使用不同的 my.cnf（也可以相同）配置文件、启动程序（也可以相同）和数据文件，在提供服务时，多实例 mysql 在逻辑上看来是各自独立的，他们根据配置文件的对应设定值，获得服务器相应数量的硬件资源

1.2.2 mysql 的多实例结构图



1.2.3 实战安装 mysql 多实例需要的依赖包

```
[root@mysql ~]# yum install ncurses-devel  libaio-devel –y
[root@mysql ~]# rpm -qa  ncurses-devel  libaio-devel
libaio-devel-0.3.107-10.el6.x86_64
ncurses-devel-5.7-4.20090207.el6.x86_64
```

1.2.4 安装编译 mysql 需要的软件

```
[root@mysql tools]# pwd
/home/tools
[root@mysql tools]# ls
cmake-2.8.8.tar.zip
[root@mysql tools]# unzip cmake-2.8.8.tar.zip
Archive:  cmake-2.8.8.tar.zip
  inflating: cmake-2.8.8.tar.gz
[root@mysql tools]# ls
cmake-2.8.8.tar.gz  cmake-2.8.8.tar.zip
[root@mysql tools]# tar xfv  cmake-2.8.8.tar.gz
[root@mysql tools]# ls
cmake-2.8.8  cmake-2.8.8.tar.gz  cmake-2.8.8.tar.zip
```

```
[root@mysql tools]# cd cmake-2.8.8
[root@mysql cmake-2.8.8]# ./configure
[root@mysql cmake-2.8.8]# gmake
[root@mysql cmake-2.8.8]# gmake install
```

关于 cmake 软件也可以直接 yum 安装

例如：

```
yum install cmake –y
```

1.2.5 安装 mysql 软件

1、创建安装 mysql 软件的用户 mysql

```
[root@mysql ~]# useradd mysql -s /sbin/nologin –M
```

2、解压 mysql-5.5.27 的软件包

```
[root@mysql tools]# ls
cmake-2.8.8  cmake-2.8.8.tar.gz  cmake-2.8.8.tar.zip  mysql-5.5.27  mysql-5.5.27.tar.gz
```

3、编译安装 mysql（下面是编译参数）

```
/usr/local/bin/cmake . -DCMAKE_INSTALL_PREFIX=/application/mysql-5.5.27 \
-DMYSQL_DATADIR=/application/mysql-5.5.27/data \
-DMYSQL_UNIX_ADDR=/application/mysql-5.5.27/tmp/mysql.sock \
-DDEFAULT_CHARSET=utf8 \
-DDEFAULT_COLLATION=utf8_general_ci \
-DEXTRA_CHARSETS=gbk,gb2312,utf8,ascii \
-DENABLED_LOCAL_INFILE=ON \
-DWITH_INNOBASE_STORAGE_ENGINE=1 \
-DWITH_FEDERATED_STORAGE_ENGINE=1 \
-DWITH_BLACKHOLE_STORAGE_ENGINE=1 \
-DWITHOUT_EXAMPLE_STORAGE_ENGINE=1 \
-DWITHOUT_PARTITION_STORAGE_ENGINE=1 \
-DWITH_FAST_MUTEXES=1 \
-DWITH_ZLIB=bundled \
-DENABLED_LOCAL_INFILE=1 \
-DWITH_READLINE=1 \
-DWITH_EMBEDDED_SERVER=1 \
-DWITH_DEBUG=0
[root@mysql mysql-5.5.27]# make && make install
[root@mysql mysql-5.5.27]# ln -s /application/mysql-5.5.27  /application/mysql
[root@mysql ~]# echo "PATH=/application/mysql/bin:$PATH" >>/etc/profile
```

把 mysql 的命令放入至全局环境变量之中

```
[root@mysql ~]# .  /etc/profile
[root@mysql mysql-5.5.27]# cd  /application/mysql
[root@mysql mysql]# ls
bin          data  include              lib  mysql-test  scripts  sql-bench
COPYING  docs  INSTALL-BINARY  man  README          share      support-files
```

1.2.6 创建存放两个 mysql 实例的数据目录

```
[root@mysql ~]# mkdir  /mysqldata/{3306,3307}/data  -p
[root@mysql ~]# tree  /mysqldata
```

```
/mysqldata
├──── 3306
│     └──── data
└──── 3307
      └──── data

4 directories, 0 files
```

1.2.7 创建两个 mysql 多实例的配置文件

1.2.7.1 3306 的实例

实例 1、3306 的配置文件

```
[root@mysql 3306]# ls
data  my.cnf（配置文件）  mysql（程序启动文件）  mysql-bin.000001  mysql-bin.index
mysqld.pid  mysql_oldboy3306.err  mysql.sock
[root@mysql 3306]# cat my.cnf
[client]
port                = 3306
socket              = /mysqldata/3306/mysql.sock
[mysql]
no-auto-rehash
[mysqld]
user        = mysql
port        = 3306
socket      = /mysqldata/3306/mysql.sock
basedir = /application/mysql
datadir = /mysqldata/3306/data
open_files_limit        = 1024
back_log = 600
max_connections = 800
max_connect_errors = 3000
table_cache = 614
external-locking = FALSE
max_allowed_packet =8M
sort_buffer_size = 1M
join_buffer_size = 1M
thread_cache_size = 100
thread_concurrency = 2
query_cache_size = 2M
query_cache_limit = 1M
query_cache_min_res_unit = 2k
#default_table_type = InnoDB
thread_stack = 192K
#transaction_isolation = READ-COMMITTED
tmp_table_size = 2M
max_heap_table_size = 2M
```

```
long_query_time = 1
#log_long_format
#log-error = /data/3306/error.log
#log-slow-queries = /data/3306/slow.log
pid-file = /mysqldata/3306/mysql.pid
log-bin = /mysqldata/3306/mysql-bin
relay-log = /mysqldata/3306/relay-bin
relay-log-info-file = /mysqldata/3306/relay-log.info
binlog_cache_size = 1M
max_binlog_cache_size = 1M
max_binlog_size = 2M
expire_logs_days = 7
key_buffer_size = 16M
read_buffer_size = 1M
read_rnd_buffer_size = 1M
bulk_insert_buffer_size = 1M
#myisam_sort_buffer_size = 1M
#myisam_max_sort_file_size = 10G
#myisam_max_extra_sort_file_size = 10G
#myisam_repair_threads = 1
#myisam_recover
lower_case_table_names = 1
skip-name-resolve
slave-skip-errors = 1032,1062
replicate-ignore-db=mysql
server-id = 1
innodb_additional_mem_pool_size = 4M
innodb_buffer_pool_size = 32M
innodb_data_file_path = ibdata1:128M:autoextend
innodb_file_io_threads = 4
innodb_thread_concurrency = 8
innodb_flush_log_at_trx_commit = 2
innodb_log_buffer_size = 2M
innodb_log_file_size = 4M
innodb_log_files_in_group = 3
innodb_max_dirty_pages_pct = 90
innodb_lock_wait_timeout = 120
innodb_file_per_table = 0
[mysqldump]
quick
max_allowed_packet = 2M
[mysqld_safe]
log-error=/mysqldata/3306/mysql_oldboy3306.err
pid-file=/mysqldata/3306/mysqld.pid
```

实例 1、3306 的启动文件，并授权执行的权限

```
[root@mysql 3306]# chmod +x /mysqldata/3306/mysql
[root@mysql 3306]# cat mysql
#!/bin/sh
#############################################################
#this scripts is created by oldboy at 2007-06-09
#oldboy QQ:31333741
#site:http://www.etiantian.org
#blog:http://oldboy.blog.51cto.com
#oldboy trainning QQ group: 208160987 226199307    44246017
#############################################################

#init
port=3306
mysql_user="root"
mysql_pwd="oldboy"
CmdPath="/application/mysql/bin"
mysql_sock="/mysqldata/${port}/mysql.sock"
#startup function
function_start_mysql()
{
    if [ ! -e "$mysql_sock" ];then
       printf "Starting MySQL...\n"
       /bin/sh  ${CmdPath}/mysqld_safe  --defaults-file=/mysqldata/${port}/my.cnf  2>&1  >
/dev/null &
    else
       printf "MySQL is running...\n"
       exit
    fi
}

#stop function
function_stop_mysql()
{
    if [ ! -e "$mysql_sock" ];then
        printf "MySQL is stopped...\n"
        exit
    else
        printf "Stoping MySQL...\n"
        ${CmdPath}/mysqladmin        -u        ${mysql_user}        -p${mysql_pwd}        -S
/mysqldata/${port}/mysql.sock shutdown
   fi
}
```

```
#restart function
function_restart_mysql()
{
        printf "Restarting MySQL...\n"
        function_stop_mysql
        sleep 2
        function_start_mysql
}

case $1 in
start)
        function_start_mysql
;;
stop)
        function_stop_mysql
;;
restart)
        function_restart_mysql
;;
*)
        printf "Usage: /mysqldata/${port}/mysql {start|stop|restart}\n"
esac
```

更改 3306 实例目录的属主和属组权限，初始化数据库

```
[root@mysql 3306]# chown -R mysql.mysql /mysqldata/3306
[root@mysql mysql]# ./scripts/mysql_install_db --datadir=/mysqldata/3306/data --
basedir=/application/mysql --user=mysql
Installing MySQL system tables...
OK
Filling help tables...
OK

To start mysqld at boot time you have to copy
support-files/mysql.server to the right place for your system

PLEASE REMEMBER TO SET A PASSWORD FOR THE MySQL root USER !
To do so, start the server, then issue the following commands:

/application/mysql/bin/mysqladmin -u root password 'new-password'
/application/mysql/bin/mysqladmin -u root -h mysql password 'new-password'

Alternatively you can run:
/application/mysql/bin/mysql_secure_installation

which will also give you the option of removing the test
```

```
databases and anonymous user created by default.  This is
strongly recommended for production servers.

See the manual for more instructions.

You can start the MySQL daemon with:
cd /application/mysql ; /application/mysql/bin/mysqld_safe &

You can test the MySQL daemon with mysql-test-run.pl
cd /application/mysql/mysql-test ; perl mysql-test-run.pl

Please report any problems with the /application/mysql/scripts/mysqlbug script!
```

启动 3306 实例

```
[root@mysql 3306]# /mysqldata/3306/mysql start
[root@mysql 3306]# lsof -i :3306
COMMAND   PID  USER   FD   TYPE DEVICE SIZE/OFF NODE NAME
mysqld  31375 mysql   12u  IPv4  98252      0t0  TCP *:mysql (LISTEN)
```

1.2.7.2 3307 的实例

3307 实例配置文件

```
[root@mysql 3307]# ls
data  my.cnf（配置文件）  mysql（程序启动文件）  mysqld.pid  mysql_oldboy3307.err
mysql.sock
[root@mysql 3307]# cat my.cnf
[client]
port            = 3307
socket          = /mysqldata/3307/mysql.sock

[mysql]
no-auto-rehash

[mysqld]
user    = mysql
port    = 3307
socket  = /mysqldata/3307/mysql.sock
basedir = /application/mysql
datadir = /mysqldata/3307/data
open_files_limit    = 1024
back_log = 600
max_connections = 800
max_connect_errors = 3000
table_cache = 614
external-locking = FALSE
max_allowed_packet =8M
sort_buffer_size = 1M
```

```
join_buffer_size = 1M
thread_cache_size = 100
thread_concurrency = 2
query_cache_size = 2M
query_cache_limit = 1M
query_cache_min_res_unit = 2k
#default_table_type = InnoDB
thread_stack = 192K
#transaction_isolation = READ-COMMITTED
tmp_table_size = 2M
max_heap_table_size = 2M
#long_query_time = 1
#log_long_format
#log-error = /data/3307/error.log
#log-slow-queries = /data/3307/slow.log
pid-file = /mysqldata/3307/mysql.pid
#log-bin = /data/3307/mysql-bin
relay-log = /mysqldata/3307/relay-bin
relay-log-info-file = /mysqldata/3307/relay-log.info
binlog_cache_size = 1M
max_binlog_cache_size = 1M
max_binlog_size = 2M
expire_logs_days = 7
key_buffer_size = 16M
read_buffer_size = 1M
read_rnd_buffer_size = 1M
bulk_insert_buffer_size = 1M
#myisam_sort_buffer_size = 1M
#myisam_max_sort_file_size = 10G
#myisam_max_extra_sort_file_size = 10G
#myisam_repair_threads = 1
#myisam_recover

lower_case_table_names = 1
skip-name-resolve
slave-skip-errors = 1032,1062
replicate-ignore-db=mysql

server-id = 3

innodb_additional_mem_pool_size = 4M
innodb_buffer_pool_size = 32M
innodb_data_file_path = ibdata1:128M:autoextend
innodb_file_io_threads = 4
```

```
innodb_thread_concurrency = 8
innodb_flush_log_at_trx_commit = 2
innodb_log_buffer_size = 2M
innodb_log_file_size = 4M
innodb_log_files_in_group = 3
innodb_max_dirty_pages_pct = 90
innodb_lock_wait_timeout = 120
innodb_file_per_table = 0
[mysqldump]
quick
max_allowed_packet = 2M
`
[mysqld_safe]
log-error=/mysqldata/3307/mysql_oldboy3307.err
pid-file=/mysqldata/3307/mysqld.pid
```

3307 实例的启动配置文件

```
[root@mysql 3307]# cat /mysqldata/3307/mysql
#!/bin/sh
##############################################################
#this scripts is created by oldboy at 2007-06-09
#oldboy QQ:31333741
#site:http://www.etiantian.org
#blog:http://oldboy.blog.51cto.com
#oldboy trainning QQ group: 208160987 226199307    44246017
##############################################################

#init
port=3307
mysql_user="root"
mysql_pwd="oldboy"
CmdPath="/application/mysql/bin"
mysql_sock="/mysqldata/${port}/mysql.sock"
#startup function
function_start_mysql()
{
    if [ ! -e "$mysql_sock" ];then
       printf "Starting MySQL...\n"
       /bin/sh  ${CmdPath}/mysqld_safe  --defaults-file=/mysqldata/${port}/my.cnf  2>&1  >
/dev/null &
    else
       printf "MySQL is running...\n"
       exit
    fi
}
```

```
#stop function
function_stop_mysql()
{
    if [ ! -e "$mysql_sock" ];then
        printf "MySQL is stopped...\n"
        exit
    else
        printf "Stoping MySQL...\n"
        ${CmdPath}/mysqladmin -u ${mysql_user} -p${mysql_pwd} -S /mysqldata/${port}/mysql.sock shutdown
    fi
}

#restart function
function_restart_mysql()
{
    printf "Restarting MySQL...\n"
    function_stop_mysql
    sleep 2
    function_start_mysql
}

case $1 in
start)
    function_start_mysql
;;
stop)
    function_stop_mysql
;;
restart)
    function_restart_mysql
;;
*)
    printf "Usage: /mysqldata/${port}/mysql {start|stop|restart}\n"
esac
```

更改 3307 实例的目录权限，初始化数据库

```
[root@mysql 3307]# chown -R mysql.root /mysqldata/3307
[root@mysql mysql]# ./scripts/mysql_install_db --datadir=/mysqldata/3307/data --basedir=/application/mysql --user=mysql
Installing MySQL system tables...
OK
Filling help tables...
OK
```

To start mysqld at boot time you have to copy
support-files/mysql.server to the right place for your system

PLEASE REMEMBER TO SET A PASSWORD FOR THE MySQL root USER !
To do so, start the server, then issue the following commands:

/application/mysql/bin/mysqladmin -u root password 'new-password'
/application/mysql/bin/mysqladmin -u root -h mysql password 'new-password'

Alternatively you can run:
/application/mysql/bin/mysql_secure_installation

which will also give you the option of removing the test
databases and anonymous user created by default. This is
strongly recommended for production servers.

See the manual for more instructions.

You can start the MySQL daemon with:
cd /application/mysql ; /application/mysql/bin/mysqld_safe &

You can test the MySQL daemon with mysql-test-run.pl
cd /application/mysql/mysql-test ; perl mysql-test-run.pl

Please report any problems with the /application/mysql/scripts/mysqlbug script!
启动 3307 实例
[root@mysql mysql]# /mysqldata/3307/mysql start
Starting MySQL...
[root@mysql mysql]# lsof -i :3307
COMMAND PID USER FD TYPE DEVICE SIZE/OFF NODE NAME
mysqld 32195 mysql 11u IPv4 99276 0t0 TCP *:opsession-prxy (LISTEN)

1.2.7.3 多实例本地登录 mysql

多实例本地登录一般是通过 socket 文件来指定具体到那个实例的，此文件的具体位置是在 mysql 编译过程或者 my.cnf 文件里指定的，在本地登录数据库时，登录程序会通过 socket 文件来判断登录的是哪个数据库实例

例如：通过 mysql –uroot –ppcwangjixuan –S /mysqldata/3307/mysql.sock 可知，登录的是 3307 这个实例，mysql.sock 文件是 mysql 服务端与本地 mysql 客户端进行通信的 unix 套接字文件

1.2.7.4 远程连接登录 mysql 多实例

远程登录 mysql 多实例中的一个实例中，通过 TCP 端口（port）来指定所要登录的 mysql 实例，此端口的配置是在 mysql 配置文件 my.cnf 中指定的

例如：在 mysql –oldboy –ppcwangjixuan –h 10.0.0.171 –P 3307 中，-P 为端口参数，后面接具体的实例端口，端口是一种逻辑连接位置，是客户端程序被分派到计算机上特殊服

务程序的一种方式，强调提前在 10.0.0.171 上对 oldboy 用户做了授权

1.3 mysql 基础安全

1.3.1 启动程序设置 700，属主和用户组为 mysql
1.3.2 为 mysq 超级用户 root 设置密码
1.3.3 登录时尽量不要在命令行暴漏密码，备份脚本中如果有密码，给设置 700，属主和用户组为 mysql 或 root
1.3.4 删除默认存在的 test 库
1.3.5 初始删除无用的用户，只保留 root 127.0.0.1 和 root localhost
1.3.6 授权用户对应的主机不要用%，权限不要给 all，最小化授权，从库只给 select 权限
1.3.7 不要给一个用户管所有的库，尽量专库专用户
1.3.8 清理 mysql 操作日志文件~/mysql_history
1.3.9 禁止开发获取到 web 连接的密码，禁止开发连接操作生产对外的库
1.4.0 服务器禁止设置外网 IP
1.4.1 防 SQL 注入（WEB），php.ini 配置文件里面设置
1.4.2 mysql 的备份的脚本给 700 的权限，并且属主和属组为 root

1.4 关于 mysql 的启动与关闭

1.4.1 单实例 MySQL 启动与关闭方法

法一：常规启动关闭数据库方式（推荐）

1.启动 mysql 命令

```
[root@db01 ~]# /etc/init.d/mysqld start
Starting MySQL..                                           [  OK  ]
```

2.查看 MySQL 端口

```
[root@db01 ~]# lsof -i :3306
COMMAND   PID   USER   FD   TYPE DEVICE SIZE/OFF NODE NAME
mysqld  4430 mysql   11u  IPv4  15227      0t0  TCP *:mysql (LISTEN)
```

3.查看 MySQL 进程

```
[root@db01 ~]# ps -ef|grep mysql|grep -v grep
root      4165     1  0 16:45 pts/0    00:00:00 /bin/sh /application/mysql/bin/mysqld_safe --datadir=/mysqldata --pid-file=/mysqldata/db01.pid
mysql     4430  4165  0 16:45 pts/0    00:00:00 /application/mysql/bin/mysqld --basedir=/application/mysql --datadir=/mysqldata --plugin-dir=/application/mysql/lib/plugin --user=mysql --log-error=/mysqldata/db01.err --pid-file=/mysqldata/db01.pid --socket=/tmp/mysql.sock --port=3306
```

4.关闭 mysql 命令

```
[root@db01 ~]# /etc/init.d/mysqld stop
Shutting down MySQL.                                       [  OK  ]
```

1.4.2 多实例 MySQL 启动与关闭方法示例

启动：

```
/mysql/data/3306/mysql start
/mysql/data/3307/mysql start
```

关闭：

/mysql/data/3306/mysql stop
/mysql/data/3307/mysql stop
启动调用了这条命令：
/bin/sh ${CmdPath}/mysqld_safe --defaults-file=/mysqldata/${port}/my.cnf 2>&1 > /dev/null &
关闭调用了这条命令：
${CmdPath}/mysqladmin -u ${mysql_user} -p${mysql_pwd} -S /mysqldata/${port}/mysql.sock shutdown

1.5 生产环境关闭 mysql 的命令

1.5.1 生产环境不能用强制命令关闭 mysql 服务

关于强制关闭 mysql 服务的命令：
killall mysqld
pkill mysqld
killall -9 mysqld
kill -9 pid

1.5.2 下面来介绍优雅关闭数据库方法：

提示：推荐前两个方法，自己写脚本启动停止就用第一个
第一种：mysqladmin 方法
mysqladmin –uroot –ppcwangjixuan shutdown
第二种：自带脚本
/etc/init.d/mysqld stop
第三种：kill 信号的方法
kill –USR2 `cat path/pid`

1.6 登录 mysql 方法

1.6.1 单实例 MySQL 登录的方法

1、mysql <--刚装完系统无密码情况登录方式，不要密码
2、mysql –uroot <--刚装完系统无密码情况登录方式，不要密码
3、mysql –uroot –ppcwangjixuan <--非脚本里一般不这样用，密码明文会泄露密码，可以掩饰 history 功能解决

1.6.2 多实例 MySQL 的登录方法

[root@mysql ~]# mysql -uroot -ppcwangjixuan -S /mysqldata/3306/mysql.sock
提示：多实例通过 mysql 的-S 命令指定不同的 sock 文件登录不同的服务中
注意：多实例的远程连接无需指定 sock 路径
mysql –uroot –p –h 10.0.0.171 –P3306

1.7 关于 mysql 管理员设置

1.7.1 为管理员 root 用户设置密码并修改方法之一

mysqladmin –uroot password ‘pcwangjixuan’ <--没有密码的用户设置密码命令
mysqladmin –uroot –p123456 password ‘pcwangjixuan’ –S /data/3306/mysql.sock <--适合多实例修改密码

1.7.2 修改管理员 root 密码法二（sql 语句修改）

mysql> UPDATE mysql.user SET password=PASSWORD('123456') WHERE user='root' and host='localhost';

mysql> flush privileges;

1.7.3 找回丢失的 mysql root 用户密码（单实例和多实例）

单实例：

1、首先停止 mysql

[root@db01 ~]# /etc/init.d/mysqld stop

2、使用—skip-grant-tables 启动 mysql，忽略授权登录验证

mysqld_safe –skip-grant-tables –user=mysql &

mysql <--登录时空密码

提示：在启动时加—skip-grant-tables 参数，表示忽略授权表验证

3、登录 mysql，设置 root 用户密码

mysql> update mysql.user SET password=PASSWORD('pcwangjixuan') WHERE user='root' and host='lcoalhost';

4、停止 mysql 服务，重新启动 mysql

[root@db01 ~]# /etc/init.d/mysqld restart

多实例

1、 首先停止 mysql

[root@mysql ~]# /mysqldata/3306/mysql stop

Stoping MySQL...

2、使用—skip-grant-tables 启动 mysql，忽略授权登录验证

[root@mysql ~]# /application/mysql/bin/mysqld_safe --defaults-file=/mysqldata/3306/my.cnf --skip-grant-tables &

3、登录 mysql，设置 root 用户密码

[root@mysql ~]# mysql -u root -S /mysqldata/3306/mysql.sock

mysql> update mysql.user SET password=PASSWORD('pcwangjixuan1') WHERE user='root' and host='localhost';

Query OK, 1 row affected (0.10 sec)

Rows matched: 1 Changed: 1 Warnings: 0

mysql> flush privileges;

Query OK, 0 rows affected (0.04 sec)

4、停止服务，重新启动

[root@mysql ~]# /mysqldata/3306/mysql stop

[root@mysql ~]# /mysqldata/3306/mysql start

1.8 SQL 结构化查询语言

1.8.1 什么是 SQL？

SQL，英文全称 Structured Query Language，中文意思是结构化查询语言，它是一种对应关系数据中的数据进行定义和操作的语言方法，是大多数关系数据库管理系统所支持的工业标准

数据库查询语言 SQL 是一种数据库查询和程序设计语言，用于存取数据以及查询、更新和管理关系数据库系统；同时也是数据库脚本文件的扩展名，结构化查询语言是高级的非过程化编程语言，允许用户在高层数据结构上工作，它不要求用户指定对数据的存放方法，也不需要用户了解具体的数据存放方式，所以，具有完全不同于底层结构的，不同数据库系统可以使用相同的结构化查询语言作为数据输入与管理的接口，结构化查询语句可以嵌套，这使得它具有极大的灵活性和强大的功能，不同的数据库系统的 SQL 语言会有一些差别

1.8.2 SQL 语句最常见的分类一般就是 3 类

DDL（Data definition Language）--数据定义语言（CREATE,ALTER,DROP），管理基础数据，例如：库，表
DCL（Data Contol Language）--数据控制语言（GRANT,REVOKE,COMMIT，ROLLBACK），用户授权，权限回收，数据提交，回滚等
DML（Data Manipulation Language）--数据操作语言（SELECT,INSERT,DELETE,UPDATE），针对数据库里的表，记录

1.9 数据库的管理应用

1.9.1 创建数据库

命令语法：create database <数据库名> <==注意库名不能数字开头，大小写不敏感
在 MySQL 默认字符集情况下建立数据库测试如下：
建立一个名为 oldboy 的数据库

```
mysql> show databases like 'oldboy';
+------------------+
| Database (oldboy) |
+------------------+
| oldboy                    |
+------------------+
1 row in set (0.04 sec)
```

查看建库的语句 (注意:如果编译时指定字符集是 utf8，创建数据库时默认指定是 utf8)

```
mysql> show create database oldboy;
+----------+-----------------------------------------------------------------+
| Database | Create Database                                                 |
+----------+-----------------------------------------------------------------+
| oldboy     | CREATE DATABASE `oldboy` /*!40100 DEFAULT CHARACTER SET utf8 */ |
+----------+-----------------------------------------------------------------+
1 row in set (0.00 sec)
```

创建一个指定字符集的数据库

```
mysql> create database oldboy_gbk CHARACTER SET gbk COLLATE gbk_chinese_ci;
Query OK, 1 row affected (0.00 sec)

mysql> show create database oldboy_gbk;
+------------+-------------------------------------------------------------------+
| Database     | Create Database                                                   |
+------------+-------------------------------------------------------------------+
| oldboy_gbk | CREATE DATABASE `oldboy_gbk` /*!40100 DEFAULT CHARACTER SET gbk
*/ |
+------------+-------------------------------------------------------------------+
1 row in set (0.00 sec)
```

1.9.2 查看库的字符集及校队规则

```
mysql> SHOW CHARACTER SET;
+----------+---------------------------+--------------------+--------+
```

```
| Charset  | Description                | Default collation  | Maxlen |
+----------+----------------------------+--------------------+--------+
| big5     | Big5 Traditional Chinese   | big5_chinese_ci    |      2 |
| dec8     | DEC West European          | dec8_swedish_ci    |      1 |
| cp850    | DOS West European          | cp850_general_ci   |      1 |
| hp8      | HP West European           | hp8_english_ci     |      1 |
| koi8r    | KOI8-R Relcom Russian      | koi8r_general_ci   |      1 |
| latin1   | cp1252 West European       | latin1_swedish_ci  |      1 |
| latin2   | ISO 8859-2 Central European| latin2_general_ci  |      1 |
| swe7     | 7bit Swedish               | swe7_swedish_ci    |      1 |
| ascii    | US ASCII                   | ascii_general_ci   |      1 |
| ujis     | EUC-JP Japanese            | ujis_japanese_ci   |      3 |
| sjis     | Shift-JIS Japanese         | sjis_japanese_ci   |      2 |
| hebrew   | ISO 8859-8 Hebrew          | hebrew_general_ci  |      1 |
| tis620   | TIS620 Thai                | tis620_thai_ci     |      1 |
| euckr    | EUC-KR Korean              | euckr_korean_ci    |      2 |
| koi8u    | KOI8-U Ukrainian           | koi8u_general_ci   |      1 |
| gb2312   | GB2312 Simplified Chinese  | gb2312_chinese_ci  |      2 |
| greek    | ISO 8859-7 Greek           | greek_general_ci   |      1 |
| cp1250   | Windows Central European   | cp1250_general_ci  |      1 |
| gbk      | GBK Simplified Chinese     | gbk_chinese_ci     |      2 |
| latin5   | ISO 8859-9 Turkish         | latin5_turkish_ci  |      1 |
| armscii8 | ARMSCII-8 Armenian         | armscii8_general_ci|      1 |
| utf8     | UTF-8 Unicode              | utf8_general_ci    |      3 |
| ucs2     | UCS-2 Unicode              | ucs2_general_ci    |      2 |
| cp866    | DOS Russian                | cp866_general_ci   |      1 |
| keybcs2  | DOS Kamenicky Czech-Slovak | keybcs2_general_ci |      1 |
| macce    | Mac Central European       | macce_general_ci   |      1 |
| macroman | Mac West European          | macroman_general_ci|      1 |
| cp852    | DOS Central European       | cp852_general_ci   |      1 |
| latin7   | ISO 8859-13 Baltic         | latin7_general_ci  |      1 |
| utf8mb4  | UTF-8 Unicode              | utf8mb4_general_ci |      4 |
| cp1251   | Windows Cyrillic           | cp1251_general_ci  |      1 |
| utf16    | UTF-16 Unicode             | utf16_general_ci   |      4 |
| cp1256   | Windows Arabic             | cp1256_general_ci  |      1 |
| cp1257   | Windows Baltic             | cp1257_general_ci  |      1 |
| utf32    | UTF-32 Unicode             | utf32_general_ci   |      4 |
| binary   | Binary pseudo charset      | binary             |      1 |
| geostd8  | GEOSTD8 Georgian           | geostd8_general_ci |      1 |
| cp932    | SJIS for Windows Japanese  | cp932_japanese_ci  |      2 |
| eucjpms  | UJIS for Windows Japanese  | eucjpms_japanese_ci|      3 |
+----------+----------------------------+--------------------+--------+
39 rows in set (0.00 sec)
```

1.9.3 企业场景创建什么字符集的数据库呢？

1、根据开发的程序确定字符集（建议 UTF8）
2、可以编译的时候指定字符集，例如：
--DDEFAULT_CHARSET=utf8 \
--DDEFAULT_COLLATION=utf8_general_ci \
然后建库的时候默认创建即可，create database oldboy；
3、编译的时候没有指定字符集或者指定了和程序不同的字符集，如何解决？
指定字符集创建数据库即可
例如：
create database oldboy_gbk DEFAULT CHARACTER SET gbk COLLATE gbk_chinese_ci <== 创建 gbk 字符集数据库

1.9.4 显示数据库

```
mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| oldboy             |
| oldboy_gbk         |
| performance_schema |
+--------------------+
5 rows in set (0.05 sec)
```

1.9.5 删除数据库

```
mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| oldboy             |
| oldboy_gbk         |
| performance_schema |
+--------------------+
5 rows in set (0.00 sec)
mysql> drop database oldboy;
Query OK, 0 rows affected (0.05 sec)
mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| oldboy_gbk         |
```

```
| performance_schema |
+--------------------+
4 rows in set (0.00 sec)
```

1.9.6 进入数据库中指定的库

```
mysql> use mysql;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A
Database changed
mysql> select database();
+------------+
| database() |
+------------+
| mysql      |
+------------+
1 row in set (0.00 sec)
```

1.9.7 查看进入当前数据库的用户

```
mysql> select user();
+----------------+
| user()         |
+----------------+
| root@localhost |
+----------------+
1 row in set (0.05 sec)
```

1.9.8 删除数据库多余的账号

语法：drop user ‘user’@’主机域 0 ’ <--注意引号，可以是单或双引号，但是不能不加

```
mysql>drop user ‘root’@’oldboy’;
mysql>drop user ‘ ’@’localhost’; <--没有的部分就用两个单引号代替即可
```

如果 drop 删除不了（一般是特殊字符或大写），可以用下面方式删除（以 root 用户，oldboy 主机为例）

```
delete from mysql.user where user=’root’ and host=’oldboy’
flush privileges;
```

1.9.9 查看数据库的用户

```
mysql> select user,host    from mysql.user;
+------+-----------+
| user | host      |
+------+-----------+
| root | 127.0.0.1 |
| root | localhost |
+------+-----------+
2 rows in set (0.06 sec)
```

2.0.0 创建 MySQL 用户及赋予用户权限

2.0.1 使用语法:

通过在 mysql 中输入 help grant 得到如下帮助信息

```
CREATE USER 'jeffrey'@'localhost' IDENTIFIED BY 'mypass';
GRANT ALL ON db1.* TO 'jeffrey'@'localhost';
GRANT SELECT ON db2.invoice TO 'jeffrey'@'localhost';
GRANT USAGE ON *.* TO 'jeffrey'@'localhost' WITH MAX_QUERIES_PER_HOUR 90;
```

2.0.2 第一种创建用户及授权方法：

```
创建用户
mysql> create user oldboy@'localhost' identified by 'oldboy';
Query OK, 0 rows affected (0.00 sec)
查看用户其权限
mysql> show grants for oldboy@'localhost';
+-----------------------------------------------------------------------------------------------------------+
|                          Grants                          for                          oldboy@localhost
|
+-----------------------------------------------------------------------------------------------------------+
| GRANT USAGE ON *.* TO 'oldboy'@'localhost' IDENTIFIED BY PASSWORD
'*7495041D24E489A0096DCFA036B166446FDDD992' |
+-----------------------------------------------------------------------------------------------------------+
1 row in set (0.00 sec)
授权用户权限
mysql> grant all on oldboy_gbk.* to oldboy@'localhost';
Query OK, 0 rows affected (0.04 sec)
mysql> show grants for oldboy@'localhost';
+-----------------------------------------------------------------------------------------------------------+
|                          Grants                          for                          oldboy@localhost
|
+-----------------------------------------------------------------------------------------------------------+
| GRANT USAGE ON *.* TO 'oldboy'@'localhost' IDENTIFIED BY PASSWORD
'*7495041D24E489A0096DCFA036B166446FDDD992' |
|    GRANT ALL PRIVILEGES ON `oldboy_gbk`.* TO 'oldboy'@'localhost'
|
+-----------------------------------------------------------------------------------------------------------+
2       rows in set (0.00 sec)
```

2.0.3 第二种创建用户及授权方法：

```
mysql> grant all on oldboy_gbk.* to oldgirl@'localhost' identified by 'oldgirl';
Query OK, 0 rows affected (0.00 sec)
列表说明:
```

grant	all	on dbname.*	to username@’lcoalhost’	identified by ‘password’
授权命令	对应权限	目标：库和表	用户名和客户端主机	用户密码

```
mysql> show grants for oldgirl@'localhost';
+-----------------------------------------------------------------------------------------------------------+
|                          Grants                          for                          oldgirl@localhost
|
+-----------------------------------------------------------------------------------------------------------+
```

```
| GRANT USAGE ON *.* TO 'oldgirl'@'localhost' IDENTIFIED BY PASSWORD '*4FD27385BB43242FE02158144D4C211F75A03F76' |
| GRANT ALL PRIVILEGES ON `oldboy_gbk`.* TO 'oldgirl'@'localhost' |
+---------------------------------------------------------------------------------------------------------------+
2 rows in set (0.00 sec)
```

2.0.4 创建用户及授权哪个网段的主机可以连接 oldboy_gbk 库

提示：如果是 web 连接数据库的用户，尽量不要授权 all，而是 select，insert，update，delete

2.0.4.1 第一种方法：

```
mysql> grant all on oldboy_gbk.* to oldgirl@'172.16.1.%' identified by 'oldgirl';
Query OK, 0 rows affected (0.00 sec)
```

%表示 172.16.1.1-255 网段

2.0.4.2 第二种方法：

```
mysql> grant all on oldboy_gbk.* to oldgirl@'172.16.1.0/255.255.255.0' identified by 'oldgirl';
Query OK, 0 rows affected (0.00 sec)
```

提示:不能这样写 oldgirl@’172.16.1.0/24’

2.0.5 关于 mysql 回收某个用户权限

语法格式：

```
REVOKE
    priv_type [(column_list)]
      [, priv_type [(column_list)]] ...
    ON [object_type] priv_level
    FROM user [, user] ...
REVOKE ALL PRIVILEGES, GRANT OPTION
    FROM user [, user] ...
```

实例:

查看 oldboy 用户回收权限前的权限

```
mysql> show grants for oldboy@'localhost';
+---------------------------------------------------------------------------------------------------------------+
| Grants for oldboy@localhost |
+---------------------------------------------------------------------------------------------------------------+
| GRANT USAGE ON *.* TO 'oldboy'@'localhost' IDENTIFIED BY PASSWORD '*7495041D24E489A0096DCFA036B166446FDDD992' |
| GRANT ALL PRIVILEGES ON `oldboy_gbk`.* TO 'oldboy'@'localhost' |
+---------------------------------------------------------------------------------------------------------------+
3 rows in set (0.00 sec)
```

查看回收 oldboy 用户的 insert 权限之后的权限

```
mysql> REVOKE INSERT ON oldboy_gbk.* FROM 'oldboy'@'localhost';
Query OK, 0 rows affected (0.00 sec)
mysql> flush privileges;
Query OK, 0 rows affected (0.00 sec)
mysql> show grants for 'oldboy'@'localhost';
```

```
+-----------------------------------------------------------------------------------------------------------------------------+
| Grants for oldboy@localhost |
+-----------------------------------------------------------------------------------------------------------------------------+
| GRANT USAGE ON *.* TO 'oldboy'@'localhost' IDENTIFIED BY PASSWORD '*7495041D24E489A0096DCFA036B166446FDDD992' |
| GRANT SELECT, UPDATE, DELETE, CREATE, DROP, REFERENCES, INDEX, ALTER, CREATE TEMPORARY TABLES, LOCK TABLES, EXECUTE, CREATE VIEW, SHOW VIEW, CREATE ROUTINE, ALTER ROUTINE, EVENT, TRIGGER ON `oldboy_gbk`.* TO 'oldboy'@'localhost' |
+-----------------------------------------------------------------------------------------------------------------------------+
2 rows in set (0.00 sec)
```

2.0.6 企业生产环境如何授权用户权限（mysql 主库）

博客，CMS 等产品的数据库授权：

对于 web 连接用户授权尽量采用最小化原则，很多开源软件都是 web 界面安装，因此，在安装期间除了 select，insert，update，delete4 个权限外，还需要 create，drop 等比较危险的权限

```
    mysql> grant insert,delete,update,select on blog.* to blog@'172.16.1.%' identified by 'blog';
Query OK, 0 rows affected (0.00 sec)
mysql> show grants for blog@'172.16.1.%';
+-----------------------------------------------------------------------------------------------------------------------------+
| Grants for blog@172.16.1.% |
+-----------------------------------------------------------------------------------------------------------------------------+
| GRANT USAGE ON *.* TO 'blog'@'172.16.1.%' IDENTIFIED BY PASSWORD '*A5BA49C964C6DB89302E2EA293048E9224B33F34' |
| GRANT SELECT, INSERT, UPDATE, DELETE ON `blog`.* TO 'blog'@'172.16.1.%' |
+-----------------------------------------------------------------------------------------------------------------------------+
2 rows in set (0.00 sec)
```

常规情况下授权 select，insert，update，delete4 个权限即可，有的开源软件，例如 discuz bbs，还需要 create，drop 等比较危险的权限，生成数据库表后，要收回 create、drop 权限

```
    mysql> revoke drop,create on blog.*    from blog@'172.16.1.%';
Query OK, 0 rows affected (0.00 sec)
to your MySQL server version for the right syntax to use near 'from    blog@'172.16.1.%" at line 1
mysql> show grants for    blog@'172.16.1.%';
+-----------------------------------------------------------------------------------------------------------------------------+
| Grants for blog@172.16.1.% |
+-----------------------------------------------------------------------------------------------------------------------------+
```

```
| GRANT USAGE ON *.* TO 'blog'@'172.16.1.%' IDENTIFIED BY PASSWORD '*A5BA49C964C6DB89302E2EA293048E9224B33F34' |
| GRANT SELECT, INSERT, UPDATE, DELETE ON `blog`.* TO 'blog'@'172.16.1.%' |
+-----------------------------------------------------------------------------------------------------------+
2 rows in set (0.00 sec)
```

2.1 数据库表操作

2.1.1 以默认字符集 latin1 建库

由于我们并为特别设置数据库及客户端字符集（因为编译是二进制安装，默认是 latin1 字符集）

```
mysql> create database oldboy;
Query OK, 1 row affected (0.11 sec)
mysql> show create database oldboy;
+----------+-----------------------------------------------------------------+
| Database | Create Database                                                 |
+----------+-----------------------------------------------------------------+
| oldboy   | CREATE DATABASE `oldboy` /*!40100 DEFAULT CHARACTER SET latin1 */ |
+----------+-----------------------------------------------------------------+
1 row in set (0.00 sec)
```

2.1.2 建立表并查看表的结构

1、建表的基本命令语法：

```
create table <table_name> {
<字段名 1><类型 1>
.....
<字段名 n><类型 n>;
```

提示：其中 create table 是关键字，不能更改，但是大小可以变化

2、 建表语句

下面是人工写法设计的建表语句例子，表名 student

```
mysql> use oldboy
Database changed
mysql> create table student(
    -> id int(4) not null,
    -> name char(20) not null,
    -> age tinyint(2) not null default '0',
    -> dept varchar(16) default null
    ->);
mysql> show tables
+-----------------+
| Tables_in_oldboy |
+-----------------+
| student          |
+-----------------+
1 row in set (0.00 sec)
```

查看已建表的结构

```
mysql> show create table student\G;
*************************** 1. row ***************************
       Table: student
Create Table: CREATE TABLE `student` (
  `id` int(4) NOT NULL,
  `name` char(20) NOT NULL,
  `age` tinyint(2) NOT NULL DEFAULT '0',
  `dept` varchar(16) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8
1 row in set (0.07 sec)
ERROR:
No query specified
```

查看表结构

```
mysql> describe student;
+-------+-------------+------+-----+---------+-------+
| Field | Type        | Null | Key | Default | Extra |
+-------+-------------+------+-----+---------+-------+
| id    | int(4)      | NO   |     | NULL    |       |
| name  | char(20)    | NO   |     | NULL    |       |
| age   | tinyint(2)  | NO   |     | 0       |       |
| dept  | varchar(16) | YES  |     | NULL    |       |
+-------+-------------+------+-----+---------+-------+
4 rows in set (0.13 sec)
```

2.1.3 mysql 表的字符类型

2.1.3.1 数字类型

列类型	需要的存储量
TINYINT	1 字节
SMALLINT	2 字节
MEDIUMINT	3 字节
INT	4 字节
TNTEGER	4 字节
BIGINT	8 字节
FLOAT（X）	4 字节如果 X<=24 或 8 字节如果 25<=x<=53
FLOAT	4 字节
DOUBLE	8 字节
DOUBLE PRECISION	8 字节
REAL	8 字节
DECIMAL（M,D）	M 字节（D+2，如果 M<D）
NUMERIC(M,D)	M 字节（D+2，如果 M<D）

2.1.3.2 日期和时间类型

列类型	需要的存储量
DATE	3 个字节

DATETIME	8 个字节
TIMESTAMP	4 个字节
TIME	3 个字节
YEAR	1 个字节

2.1.3.3 字符串类型

列类型	需要的存储量
CHAR(M)	M 字节，1<=M<=255
VARCHAR(M)	L+1 字节，在此 L<=M 和 1<=M<=255
TINYBLOB,TINYTEXT	L+1 字节，在此 L<2^8
BLOB,TEXT	L+2 字节，在此 L<2^16
MEDIUMBLOB, MEDIUMTEXT	L+3 字节，在此 L<2^24
LONGBLOB, LONGTEXT	L+3 字节，在此 L<2^32
ENUM('value1','value2',....)	1 或 2 个字节，取决于枚举值的数目（最大值 65535）
SET ('value1','value2',....)	1,2,3,4 或 8 个字节，取决于集合成员的数量（最多 64 个成员）

2.1.3.4 关于字符类型小结

1. INT[(M)] 型： 正常大小整数类型

2. CHAR(M) 型：定长字符串类型，当存储时，总是用空格填满右边到指定的长度。

3. VARCHAR 型：变长字符串类型

有关 mysql 字段类型详细内容，请大家参考 mysql 手册。

1. INT[(M)] 型： 正常大小整数类型

2. DOUBLE[(M,D)] [ZEROFILL] 型： 正常大小(双精密)浮点数字类型

3. DATE 日期类型：支持的范围是 1000-01-01 到 9999-12-31。MySQL 以 YYYY-MM-DD 格式来显示 DATE 值，但是允许你使用字符串或数字把值赋给 DATE 列

4. CHAR(M) 型：定长字符串类型，当存储时，总是是用空格填满右边到指定的长度

5. BLOB TEXT 类型，最大长度为 65535(2^16-1)个字符。

6. VARCHAR 型：变长字符串类型

提示：见 3.2.2.1 mysql 字段类型

2.1.4 为表的字段创建索引

数据库索引就象书的目录一样，如果在字段上建立了索引，那么以索引列为查询条件时可以加快查询数据的速度

查询数据库，按主键查询是最快的，每个表只能有一个主键列，但是可以有多个普通索引列，主键列要求列的所有内容必须唯一，而普通索引列不要求内容必须唯一

主键就类似我们在学校学习时的学好一样，班级里是唯一的，整个表的每一条记录的主键值在表内都是唯一的，用来唯一标识一条记录

2.1.4.1 为表创建主键索引的方法

```
mysql> use oldboy
Database changed
mysql> create table student(
    -> id int(4) not null AUTO_INCREMENT,
    -> name char(20) not null,
```

```
    -> age tinyint(2) not null default '0',
    -> dept varchar(16) default null,
    ->primary key(id),
    ->KEY index_name(name)
    ->);
    提示：
    primary key(id)   <-主键
    KEY index_name(name) <-name 字段普通索引
    只有 int 类型且为 primary key 才可以使用 auto_increment
```

2.1.4.2 查看 student 表的结构

```
mysql> show create table student\G;
*************************** 1. row ***************************
       Table: student
Create Table: CREATE TABLE `student` (
  `id` int(4) NOT NULL AUTO_INCREMENT,
  `name` char(20) NOT NULL,
  `age` tinyint(2) NOT NULL DEFAULT '0',
  `dept` varchar(16) DEFAULT NULL,
  PRIMARY KEY (`id`),
  KEY `index_name` (`name`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8
1 row in set (0.00 sec)
ERROR:
No query specified
mysql> describe student;
+-------+-------------+------+-----+---------+----------------+
| Field | Type        | Null | Key | Default | Extra          |
+-------+-------------+------+-----+---------+----------------+
| id    | int(4)      | NO   | PRI | NULL    | auto_increment |
| name  | char(20)    | NO   | MUL | NULL    |                |
| age   | tinyint(2)  | NO   |     | 0       |                |
| dept  | varchar(16) | YES  |     | NULL    |                |
+-------+-------------+------+-----+---------+----------------+
4 rows in set (0.00 sec)
```

2.1.4.3 怎么删除一个表的主键

```
mysql> alter table student drop primary key;
提示:如果一个表中的 primary key 设置了 AUTO_INCREMENT（自动增加）的话，就删不掉
```

2.1.4.4 利用 alter 命令修改 id 列为自增主键列

```
mysql> alter table student change id id int primary key auto_increment;
```

2.1.4.5 建表后利用 alter 增加普通索引

```
删除建表时创建的 index_name 索引
mysql> select database();
+------------+
| database() |
```

```
+------------+
| oldboy      |
+------------+
1 row in set (0.00 sec)
mysql> show create table student;
+---------+-------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------
------------------------------+
|              Table                                             |              Create              Table
|
+---------+-------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------
------------------------------+
| student | CREATE TABLE `student` (
   `id` int(4) NOT NULL AUTO_INCREMENT,
   `name` char(20) NOT NULL,
   `age` tinyint(2) NOT NULL DEFAULT '0',
   `dept` varchar(16) DEFAULT NULL,
   PRIMARY KEY (`id`),
   KEY `index_name` (`name`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 |
+---------+-------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------
------------------------------+
1 row in set (0.00 sec)
删除普通索引
mysql> alter table student drop index index_name;
Query OK, 0 rows affected (0.08 sec)
Records: 0   Duplicates: 0   Warnings: 0

mysql> show create table student;
+---------+-------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------
---+
|              Table                                             |              Create              Table
|
+---------+-------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------
---+
| student | CREATE TABLE `student` (
   `id` int(4) NOT NULL AUTO_INCREMENT,
   `name` char(20) NOT NULL,
   `age` tinyint(2) NOT NULL DEFAULT '0',
   `dept` varchar(16) DEFAULT NULL,
```

```
  PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 |
+---------+-----------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------
---+
1 row in set (0.00 sec)
创建普通索引
mysql> use oldboy
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> alter table student add index index_name (name);
Query OK, 0 rows affected (0.47 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> show create table student;
+---------+-----------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------
--------------------------------+
|              Table                                 |              Create              Table
|
+---------+-----------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------
--------------------------------+
| student | CREATE TABLE `student` (
  `id` int(4) NOT NULL AUTO_INCREMENT,
  `name` char(20) NOT NULL,
  `age` tinyint(2) NOT NULL DEFAULT '0',
  `dept` varchar(16) DEFAULT NULL,
  PRIMARY KEY (`id`),
  KEY `index_name` (`name`)  提示:访问数据量很大的时候，不适合建立普通索引，会影响用户访问，尽量选择业务低估时建立索引
) ENGINE=InnoDB DEFAULT CHARSET=utf8 |
+---------+-----------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------
--------------------------------+
1 row in set (0.00 sec)
```

2.1.4.6 对表字段的前 n 个字符创建普通索引

当遇到表中比较大的列时，列内容的前 n 个字符在所有内容中已接近唯一时，这时可以对列的前 n 个字符建立索引，而无需对整个列建立本索引，这样可以节省创建索引占用的系统空间，以及降低读取和更新维护索引消耗的系统资源

对字段前 n 个字符创建普通索引的语法：

create index index_name on student(name(8));条件列前 N 个字符创建索引

```
下面来实战演示：
mysql> select database();
+------------+
| database() |
+------------+
| oldboy     |
+------------+
1 row in set (0.00 sec)
为 dept 列前八个字符创建普通索引
mysql> describe student;
+-------+-------------+------+-----+---------+----------------+
| Field | Type        | Null | Key | Default | Extra          |
+-------+-------------+------+-----+---------+----------------+
| id    | int(4)      | NO   | PRI | NULL    | auto_increment |
| name  | char(20)    | NO   | MUL | NULL    |                |
| age   | tinyint(2)  | NO   |     | 0       |                |
| dept  | varchar(16) | YES  |     | NULL    |                |
+-------+-------------+------+-----+---------+----------------+
4 rows in set (0.00 sec)
mysql> create index index_name_dept on student (dept(8));
Query OK, 0 rows affected (0.21 sec)
Records: 0  Duplicates: 0  Warnings: 0
mysql> describe student;
+-------+-------------+------+-----+---------+----------------+
| Field | Type        | Null | Key | Default | Extra          |
+-------+-------------+------+-----+---------+----------------+
| id    | int(4)      | NO   | PRI | NULL    | auto_increment |
| name  | char(20)    | NO   | MUL | NULL    |                |
| age   | tinyint(2)  | NO   |     | 0       |                |
| dept  | varchar(16) | YES  | MUL | NULL    |                |
+-------+-------------+------+-----+---------+----------------+
4 rows in set (0.00 sec)
mysql> show create table student;
+---------+------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------+
| Table | Create Table |
+---------+------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------+
| student | CREATE TABLE `student` (
  `id` int(4) NOT NULL AUTO_INCREMENT,
  `name` char(20) NOT NULL,
```

```
  `age` tinyint(2) NOT NULL DEFAULT '0',
  `dept` varchar(16) DEFAULT NULL,
  PRIMARY KEY (`id`),
  KEY `index_name` (`name`),
  KEY `index_name_dept` (`dept`(8))
) ENGINE=InnoDB DEFAULT CHARSET=utf8 |
+---------+----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------+
1 row in set (0.00 sec)
另外一种建立表后创建普通索引的方法：
mysql> alter table student add index index_name_dept (dept(8));
```

2.1.4.7 为表的多个字段创建联合索引

如何查询数据的条件时多列时，我们可以为多个查询的列创建联合索引，甚至，可以为多列的前 n 个字符列创建联合索引，实战演示如下:

```
mysql> create index index_name_and_dept on student (name,dept);
Query OK, 0 rows affected (0.11 sec)
Records: 0  Duplicates: 0  Warnings: 0
mysql> show create table student;
+---------+----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------+
| Table | Create Table |
+---------+----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------+
| student | CREATE TABLE `student` (
  `id` int(4) NOT NULL AUTO_INCREMENT,
  `name` char(20) NOT NULL,
  `age` tinyint(2) NOT NULL DEFAULT '0',
  `dept` varchar(16) DEFAULT NULL,
  PRIMARY KEY (`id`),
  KEY `index_name` (`name`),
  KEY `index_name_dept` (`dept`(8)),
  KEY `index_name_and_dept` (`name`,`dept`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 |
+---------+----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------+
1 row in set (0.00 sec)
```

2.1.4.8 为表的多个字段的前 n 个字符创建联合索引

```
mysql> create index index_name_and_dept on student (name(10),dept(10));
Query OK, 0 rows affected (0.03 sec)
```

```
Records: 0  Duplicates: 0  Warnings: 0
mysql> show create table student;
+---------+------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------------------
----+
|          Table                                        |           Create            Table
|
+---------+------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------------------
----+
| student | CREATE TABLE `student` (
  `id` int(4) NOT NULL AUTO_INCREMENT,
  `name` char(20) NOT NULL,
  `age` tinyint(2) NOT NULL DEFAULT '0',
  `dept` varchar(16) DEFAULT NULL,
  PRIMARY KEY (`id`),
  KEY `index_name` (`name`),
  KEY `index_name_dept` (`dept`(8)),
  KEY `index_name_and_dept` (`name`(10),`dept`(10))
) ENGINE=InnoDB DEFAULT CHARSET=utf8 |
+---------+------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------------------
----+
1 row in set (0.00 sec)
```

提示:按条件列查询数据时，联合索引是有前缀生效特性的
index(a,b,c)仅 a,ab,abc 三个查询条件列可以走索引，b,bc,ac,c 等无法使用索引了
尽量把最常用作为查询条件的列，放在第一位置

2.1.4.9 主键也可以联合多列做索引

```
mysql> show create  table  mysql.user\G;
*************************** 1. row ***************************
       Table: user
Create Table: CREATE TABLE `user` (
  `Host` char(60) COLLATE utf8_bin NOT NULL DEFAULT '',
  `User` char(16) COLLATE utf8_bin NOT NULL DEFAULT '',
  `Password` char(41) CHARACTER SET latin1 COLLATE latin1_bin NOT NULL DEFAULT '',
  `Select_priv` enum('N','Y') CHARACTER SET utf8 NOT NULL DEFAULT 'N',
  `Insert_priv` enum('N','Y') CHARACTER SET utf8 NOT NULL DEFAULT 'N',
  `Update_priv` enum('N','Y') CHARACTER SET utf8 NOT NULL DEFAULT 'N',
  `Delete_priv` enum('N','Y') CHARACTER SET utf8 NOT NULL DEFAULT 'N',
  `Create_priv` enum('N','Y') CHARACTER SET utf8 NOT NULL DEFAULT 'N',
  `Drop_priv` enum('N','Y') CHARACTER SET utf8 NOT NULL DEFAULT 'N',
```

```
  `Reload_priv` enum('N','Y') CHARACTER SET utf8 NOT NULL DEFAULT 'N',
  `Shutdown_priv` enum('N','Y') CHARACTER SET utf8 NOT NULL DEFAULT 'N',
  `Process_priv` enum('N','Y') CHARACTER SET utf8 NOT NULL DEFAULT 'N',
  `File_priv` enum('N','Y') CHARACTER SET utf8 NOT NULL DEFAULT 'N',
  `Grant_priv` enum('N','Y') CHARACTER SET utf8 NOT NULL DEFAULT 'N',
  `References_priv` enum('N','Y') CHARACTER SET utf8 NOT NULL DEFAULT 'N',
  `Index_priv` enum('N','Y') CHARACTER SET utf8 NOT NULL DEFAULT 'N',
  `Alter_priv` enum('N','Y') CHARACTER SET utf8 NOT NULL DEFAULT 'N',
  `Show_db_priv` enum('N','Y') CHARACTER SET utf8 NOT NULL DEFAULT 'N',
  `Super_priv` enum('N','Y') CHARACTER SET utf8 NOT NULL DEFAULT 'N',
  `Create_tmp_table_priv` enum('N','Y') CHARACTER SET utf8 NOT NULL DEFAULT 'N',
  `Lock_tables_priv` enum('N','Y') CHARACTER SET utf8 NOT NULL DEFAULT 'N',
  `Execute_priv` enum('N','Y') CHARACTER SET utf8 NOT NULL DEFAULT 'N',
  `Repl_slave_priv` enum('N','Y') CHARACTER SET utf8 NOT NULL DEFAULT 'N',
  `Repl_client_priv` enum('N','Y') CHARACTER SET utf8 NOT NULL DEFAULT 'N',
  `Create_view_priv` enum('N','Y') CHARACTER SET utf8 NOT NULL DEFAULT 'N',
  `Show_view_priv` enum('N','Y') CHARACTER SET utf8 NOT NULL DEFAULT 'N',
  `Create_routine_priv` enum('N','Y') CHARACTER SET utf8 NOT NULL DEFAULT 'N',
  `Alter_routine_priv` enum('N','Y') CHARACTER SET utf8 NOT NULL DEFAULT 'N',
  `Create_user_priv` enum('N','Y') CHARACTER SET utf8 NOT NULL DEFAULT 'N',
  `Event_priv` enum('N','Y') CHARACTER SET utf8 NOT NULL DEFAULT 'N',
  `Trigger_priv` enum('N','Y') CHARACTER SET utf8 NOT NULL DEFAULT 'N',
  `Create_tablespace_priv` enum('N','Y') CHARACTER SET utf8 NOT NULL DEFAULT 'N',
  `ssl_type` enum('','ANY','X509','SPECIFIED') CHARACTER SET utf8 NOT NULL DEFAULT
'',
  `ssl_cipher` blob NOT NULL,
  `x509_issuer` blob NOT NULL,
  `x509_subject` blob NOT NULL,
  `max_questions` int(11) unsigned NOT NULL DEFAULT '0',
  `max_updates` int(11) unsigned NOT NULL DEFAULT '0',
  `max_connections` int(11) unsigned NOT NULL DEFAULT '0',
  `max_user_connections` int(11) unsigned NOT NULL DEFAULT '0',
  `plugin` char(64) COLLATE utf8_bin DEFAULT '',
  `authentication_string` text COLLATE utf8_bin,
  PRIMARY KEY (`Host`,`User`)
) ENGINE=MyISAM DEFAULT CHARSET=utf8 COLLATE=utf8_bin COMMENT='Users and
global privileges'
1 row in set (0.00 sec)
ERROR:
No query specified
```

2.1.5.0 统计一个字段列的唯一值个数

```
mysql> select    user from mysql.user;
+-----------+
| user          |
```

```
+-----------+
| root      |
| blog      |
| oldgirl   |
| wordpress |
| oldgirl   |
| oldboy    |
| oldgirl   |
| root      |
+-----------+
8 rows in set (0.00 sec)
mysql> select count(distinct user) from mysql.user;
+----------------------+
| count(distinct user) |
+----------------------+
|                    5 |
+----------------------+
1 row in set (0.06 sec)
```

提示：尽量在唯一值多的大表上建立索引

2.1.5.1 创建唯一索引（非主键）

```
mysql> show create table student ;
+---------+-------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------
-------------------------------------+
|              Table                                                           |                  Create              Table
|
+---------+-------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------
-------------------------------------+
| student | CREATE TABLE `student` (
  `id` int(4) NOT NULL AUTO_INCREMENT,
  `name` char(20) NOT NULL,
  `age` tinyint(2) NOT NULL DEFAULT '0',
  `dept` varchar(16) DEFAULT NULL,
  PRIMARY KEY (`id`),
  UNIQUE KEY `index_age` (`age`),
  KEY `index_name` (`name`),
  KEY `index_name_dept` (`dept`(8)),
  KEY `index_name_and_dept` (`name`(10),`dept`(10))
) ENGINE=InnoDB DEFAULT CHARSET=utf8 |
+---------+-------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------
```

```
-------------------------------------------------------------------------------------------------------------------
-----------------------------------+
1 row in set (0.00 sec)
```

2.1.5.2 索引列的创建及生效条件

问题一：既然索引可以加快查询速度，那么就给所有的列建索引吧？

解答:因为索引不但占用系统空间，而且更新数据时还需要维护索引数据的，因此，索引是一把双刃剑，并不是越多越好，例如：数十到几百行的小表上无需建立索引，更新频繁，读取比较少的表要少建立索引

问题二：需要在哪些列上创建索引了?

解答：select user,host from mysql.user where password=...,索引一定要创建在 where 后的条件列上，而不是 select 后的选择数据的列上，另外，我们要尽量选择在唯一值多的大表上的列建立索引，例如:男女性别列唯一值，不适合建立索引

2.1.5 往表中插入数据

1、命令语法：

```
insert into <表名> [(<字段名 1>)[,.....<字段名>])]values(值 1)[,(值 n)]
```

2、新建一个简单的测试表 test

```
mysql> use oldboy
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A
Database changed
mysql>
mysql>
mysql> show tables ;
+-----------------+
| Tables_in_oldboy |
+-----------------+
| student                |
+-----------------+
1 row in set (0.00 sec)
mysql> create table test(
    -> id int(4) not null auto_increment,
    -> name char(20) not null,
    -> primary key(id)
    -> );
Query OK, 0 rows affected (0.07 sec)
mysql> show tables;
+-----------------+
| Tables_in_oldboy |
+-----------------+
| student                |
| test                   |
+-----------------+
2 rows in set (0.00 sec)
```

3、 往 test 表中插入数据

1、往 test 表中插入第一条数据

```
mysql> insert into test (id,name) values(1,'oldboy');
Query OK, 1 row affected (0.05 sec)
mysql> select * from test;
+----+--------+
| id | name   |
+----+--------+
|  1 | oldboy |
+----+--------+
1 row in set (0.00 sec)
```

2、由于 id 列为自增的，所以，可以只在 name 列插入值

```
mysql> insert into test(name) values('oldgirl');
Query OK, 1 row affected (0.00 sec)
mysql> select * from test;
+----+---------+
| id | name    |
+----+---------+
|  1 | oldboy  |
|  2 | oldgirl |
+----+---------+
2 rows in set (0.00 sec)
```

3、如果不指定列，就要按规矩为每列都插入恰当的值

```
mysql> select * from test;
+----+-----------+
| id | name      |
+----+-----------+
|  1 | oldboy    |
|  2 | oldgirl   |
|  3 | zhangxuan |
+----+-----------+
4       rows in set (0.00 sec)
```

4、 批量插入数据方法，提示效率

```
mysql> insert into test (id,name) values (4,'engchao'),(5,'geili');
Query OK, 2 rows affected (0.05 sec)
Records: 2    Duplicates: 0    Warnings: 0
mysql> select * from test;
+----+-----------+
| id | name      |
+----+-----------+
|  1 | oldboy    |
|  2 | oldgirl   |
|  3 | zhangxuan |
|  4 | engchao   |
|  5 | geili     |
```

```
+----+-----------+
5 rows in set (0.00 sec)
```

2.1.6 往表中删除一条数据

```
mysql> select * from test;
+----+------+-----------+-------------+
| id | age  | name      | shouji      |
+----+------+-----------+-------------+
|  1 | NULL | oldgirl   | NULL        |
|  2 | NULL | 老男孩    | NULL        |
|  3 | NULL | etiantian | NULL        |
|  4 |   24 | zhangxuan | 13511111111 |
|  5 |   22 | huangyan  | 13655555555 |
+----+------+-----------+-------------+
5 rows in set (0.00 sec)

mysql> delete from test where id=5;
Query OK, 1 row affected (0.04 sec)

mysql> select * from test;
+----+------+-----------+-------------+
| id | age  | name      | shouji      |
+----+------+-----------+-------------+
|  1 | NULL | oldgirl   | NULL        |
|  2 | NULL | 老男孩    | NULL        |
|  3 | NULL | etiantian | NULL        |
|  4 |   24 | zhangxuan | 13511111111 |
+----+------+-----------+-------------+
4 rows in set (0.00 sec)
```

2.1.7 查询数据

2.1.7.1 查询表的所有数据行

1、命令语法：select <字段 1，字段 2，..> from <表名> where <表达式>
其中，select，from，where 是不能随便改的，是关键字，支持大小写
2、查看表 test 中所有数据
a.进入指定库后查询

```
mysql> use oldboy
Database changed
mysql> show tables;
+-----------------+
| Tables_in_oldboy |
+-----------------+
| student          |
| test             |
+-----------------+
2 rows in set (0.00 sec)
```

```
mysql> select * from test;;
+----+-----------+
| id | name      |
+----+-----------+
|  1 | oldboy    |
|  2 | oldgirl   |
|  3 | zhangxuan |
|  4 | engchao   |
|  5 | geili     |
+----+-----------+
5 rows in set (0.00 sec)
ERROR:
No query specified
```

2.1.7.2 查看 mysql 库的用户

```
mysql> select user,host from mysql.user;
+-----------+-------------------------+
| user      | host                    |
+-----------+-------------------------+
| root      | 127.0.0.1               |
| blog      | 172.16.1.%              |
| oldgirl   | 172.16.1.%              |
| wordpress | 172.16.1.%              |
| oldgirl   | 172.16.1.0/255.255.255.0 |
| oldboy    | localhost               |
| oldgirl   | localhost               |
| root      | localhost               |
+-----------+-------------------------+
rows in set (0.00 sec)
```

2.1.7.3 根据指定条件查询表的部分数据

```
1、例:查看表 test 中前两行数据
执行命令：
mysql> select * from test limit 2;
+----+---------+
| id | name    |
+----+---------+
|  1 | oldboy  |
|  2 | oldgirl |
+----+---------+
2 rows in set (0.00 sec)
2、例：查看第一行的后两行数据
执行命令：
mysql> select * from test limit 1,2;
+----+-----------+
| id | name      |
```

```
+----+-----------+
|  2 | oldgirl   |
|  3 | zhangxuan |
+----+-----------+
2 rows in set (0.00 sec)
```

2.1.7.4 根据固定条件查数据

```
执行命令：
mysql> select * from test where id=1;
+----+--------+
| id | name   |
+----+--------+
|  1 | oldboy |
+----+--------+
1 row in set (0.05 sec)
mysql> select * from test where name='oldboy'; <-查询字符中要加引号
+----+--------+
| id | name   |
+----+--------+
|  1 | oldboy |
+----+--------+
1 row in set (0.00 sec)
```

2.1.7.5 指定固定条件范围查数据

```
执行命令：
mysql> select * from test where id>2 and id<5;
+----+-----------+
| id | name      |
+----+-----------+
|  3 | zhangxuan |
|  4 | engchao   |
+----+-----------+
2 rows in set (0.00 sec)
```

2.1.7.6 根据顺序查看列数据

```
正序
mysql> select * from test order by id asc;
+----+-----------+
| id | name      |
+----+-----------+
|  1 | oldboy    |
|  2 | oldgirl   |
|  3 | zhangxuan |
|  4 | engchao   |
|  5 | geili     |
+----+-----------+
5       rows in set (0.00 sec)
```

倒序

```
mysql> select * from test order by id desc;
+----+-----------+
| id | name      |
+----+-----------+
|  5 | geili     |
|  4 | engchao   |
|  3 | zhangxuan |
|  2 | oldgirl   |
|  1 | oldboy    |
+----+-----------+
5 rows in set (0.00 sec)
```

2.1.6.7 在表中根据条件导出数据至文件中

```
mysql> select * from test where id>2 and id<5 order by id desc into outfile '/tmp/id.txt';
Query OK, 2 rows affected (0.06 sec)
mysql> system cat /tmp/id.txt
4	engchao
3	zhangxuan
```

2.1.8 多表查询

建立几个关联表，实现多表连表查询，就需要有关联表及数据

2.1.8.1 创建学生表

```
mysql> create database oldboy;
Query OK, 1 row affected (0.01 s
mysql> use oldboy
Database changed
创建学生表
create table student(
Sno int(10) NOT NULL COMMENT '学号',
Sname varchar(16) NOT NULL COMMENT '姓名',
Ssex    char(2) NOT NULL COMMENT '性别',
Sage    tinyint(2) NOT NULL default '0' COMMENT '学生年龄',
Sdept    varchar(16) default NULL COMMENT '学生所在系别',
PRIMARY KEY    (Sno) ,
key index_Sname (Sname)
);
```

2.1.8.2 在学生表里插入数据

```
insert into student values(0001,'宏志','男',30,'计算机网络');
insert into student values(0002,'王硕','男',30,'computer application');
insert into student values(0003,'oldboy','男',28,'物流管理');
insert into student values(0004,'脉动','男',29,'computer application');
insert into student values(0005,'oldgirl','女',26,'计算机科学与技术');
insert into student values(0006,'莹莹','女',22,'护士');
```

2.1.8.3 创建课程表

```
create table course(
```

```
Cno int(10) NOT NULL COMMENT  '课程号',
Cname varchar(16) NOT NULL COMMENT  '课程名',
Ccredit tinyint(2) NOT NULL COMMENT  '学分',
PRIMARY KEY (Cno)
);
```

2.1.8.4 在课程表里插入数据

```
insert into course values(1001,'linux 中高级运维','3');
insert into course values(1002,'linux 中高级架构师','5');
insert into course values(1003,'mysql 高级 Dba','4');
insert into course values(1004,'python 运维开发','4');
insert into course values(1005,'java web 开发','3');
```

2.1.8.5 创建选课表

```
create table sc(
Scid int(12) not null auto_increment COMMENT '主键',
Cno int(10) not null  COMMENT '课程号',
Sno int(10) not null COMMENT '学号',
Grade tinyint(2) not null COMMENT '学生成绩',
PRIMARY KEY(Scid)
);
```

2.1.8.6 在选课表里加入数据

```
insert into sc(Sno,Cno,Grade) values(0001,1001,4);
insert into sc(Sno,Cno,Grade) values(0001,1002,3);
insert into sc(Sno,Cno,Grade) values(0001,1003,1);
insert into sc(Sno,Cno,Grade) values(0001,1004,6);

insert into sc(Sno,Cno,Grade) values(0002,1001,3);
insert into sc(Sno,Cno,Grade) values(0002,1002,2);
insert into sc(Sno,Cno,Grade) values(0002,1003,2);
insert into sc(Sno,Cno,Grade) values(0002,1004,8);

insert into sc(Sno,Cno,Grade) values(0003,1001,4);
insert into sc(Sno,Cno,Grade) values(0003,1002,4);
insert into sc(Sno,Cno,Grade) values(0003,1003,2);
insert into sc(Sno,Cno,Grade) values(0003,1004,8);

insert into sc(Sno,Cno,Grade) values(0004,1001,1);
insert into sc(Sno,Cno,Grade) values(0004,1002,1);
insert into sc(Sno,Cno,Grade) values(0004,1003,2);
insert into sc(Sno,Cno,Grade) values(0004,1004,3);

insert into sc(Sno,Cno,Grade) values(0005,1001,5);
insert into sc(Sno,Cno,Grade) values(0005,1002,3);
insert into sc(Sno,Cno,Grade) values(0005,1003,2);
insert into sc(Sno,Cno,Grade) values(0005,1004,9);
```

2.1.8.6 联表查询命令

```
mysql> select student.Sno,student.Sname,course.Cname,sc.Grade from student,course,sc where student.Sno=sc.Sno and course.Cno=sc.Cno order by Sno;
+-----+---------+------------------------+-------+
| Sno | Sname   | Cname                  | Grade |
+-----+---------+------------------------+-------+
|   1 | 宏志    | mysql 高级 Dba         |     1 |
|   1 | 宏志    | python 运维开发        |     6 |
|   1 | 宏志    | linux 中高级运维       |     4 |
|   1 | 宏志    | linux 中高级架构师     |     3 |
|   2 | 王硕    | linux 中高级运维       |     3 |
|   2 | 王硕    | linux 中高级架构师     |     2 |
|   2 | 王硕    | mysql 高级 Dba         |     2 |
|   2 | 王硕    | python 运维开发        |     8 |
|   3 | oldboy  | mysql 高级 Dba         |     2 |
|   3 | oldboy  | python 运维开发        |     8 |
|   3 | oldboy  | linux 中高级运维       |     4 |
|   3 | oldboy  | linux 中高级架构师     |     4 |
|   4 | 脉动    | linux 中高级运维       |     1 |
|   4 | 脉动    | linux 中高级架构师     |     1 |
|   4 | 脉动    | mysql 高级 Dba         |     2 |
|   4 | 脉动    | python 运维开发        |     3 |
|   5 | oldgirl | python 运维开发        |     9 |
|   5 | oldgirl | linux 中高级运维       |     5 |
|   5 | oldgirl | linux 中高级架构师     |     3 |
|   5 | oldgirl | mysql 高级 Dba         |     2 |
+-----+---------+------------------------+-------+
20 rows in set (0.00 sec)
```

2.1.9 使用 explain 查看 select 语句的执行计划

2.1.9.1 用查询语句查看是否使用索引情况

```
mysql> explain select * from test where name='oldgirl'\G;
*************************** 1. row ***************************
           id: 1
  select_type: SIMPLE
        table: test
         type: ALL
possible_keys: NULL    从查看的结果看出，查询的时候没有走索引
          key: NULL
      key_len: NULL
          ref: NULL
         rows: 4       总结查询了 4 行
        Extra: Using where
1 row in set (0.00 sec)
```

```
ERROR:
No query specified
mysql> show create table test;
+-------+-----------------------------------------------------------------------------------------------------------------------------------------------------------------+
| Table | Create Table |
+-------+-----------------------------------------------------------------------------------------------------------------------------------------------------------------+
| test | CREATE TABLE `test` (
  `id` int(4) NOT NULL DEFAULT '0',
  `age` tinyint(2) DEFAULT NULL,
  `name` varchar(16) DEFAULT NULL,
  `shouji` char(11) DEFAULT NULL,
  PRIMARY KEY (`id`)
) ENGINE=MyISAM DEFAULT CHARSET=utf8 |
+-------+-----------------------------------------------------------------------------------------------------------------------------------------------------------------+
1 row in set (0.00 sec)
```

2.1.9.2 为该列创建索引，再用查询语句查看是否走了索引

```
mysql> alter table test add index index_name (name);
Query OK, 4 rows affected (0.01 sec)
Records: 4  Duplicates: 0  Warnings: 0
mysql> explain select * from test where name='oldgirl'\G;
*************************** 1. row ***************************
           id: 1
  select_type: SIMPLE
        table: test
         type: ref
possible_keys: index_name   从下面结果看出语句查询的时候走了索引
          key: index_name
      key_len: 51
          ref: const
         rows: 1            总共查询了 1 行，效率更快
        Extra: Using where
1 row in set (0.00 sec)
ERROR:
No query specified
```

2.2.0 使用 explain 优化 SQL 语句（select 语句）的基本流程

2.2.1 用命令抓取慢 SQL 语句，然后用 explain 命令查看查询语句是否走的索引查询

```
1 在数据库命令行里面操作
mysql> show full processlist;
+----+------+-----------+--------+---------+------+-------+----------------------+
| Id | User | Host      | db     | Command | Time | State | Info                 |
```

```
+----+------+-----------+--------+---------+------+-------+----------------------+
| 13 | root | localhost | oldboy | Query   |    0 | NULL  | show full processlist |
+----+------+-----------+--------+---------+------+-------+----------------------+
1 row in set (0.00 sec)
2 在 linux 命令行操作
[root@mysql 3306]# mysql -u root -ppcwangjixuan -S /mysqldata/3306/mysql.sock -e "show full processlist"|grep -vi "sleep"
Id  User Host db  Command  Time  StateInfo
15  root localhost NULL  Query  0  NULL  show full processlist
```

2.2.2 设置配置参数记录慢查询语句

```
log_query_time = 2 <==查询时间超过 2 秒，记录到 log 里
log_queries_not_using_indexs <==没有走索引的语句，记录到 log 里
log-slow-queries = /data/3306/slow.log
```

2.2.3 对抓取到的慢查询语句用 explain 命令检查索引执行情况

```
例如：
explain select * from test where name='oldboy'\G;
```

2.2.4 对需要建索引的条件列建立索引

```
大表不能高峰值建立索引，300 万记录
```

2.2.5 切割慢查询日志，去重分析后发给大家

```
关于 mysql 的配置文件参数设置（my.cnf）
[mysqld]
long_query_time = 1
log-slow-queries = /data/3306/slow.log
log_queries_not_using_indexs
关于日志切割脚本
mv /data/3306/slow.log /opt/$(date +%F)_slow.log
mysqladmin –uroot –ppcwangjixuan –S /data/3306/mysql.sock flush-logs
```

2.2.1 修改表中数据

2.2.1.1 修改表中指定条件固定列的数据

```
1、命令语法：update <表名> set <字段>=<新值>,..... where <条件> (一定要注意条件)
2、修改指定的行字段内容
查看要修改的表
mysql> use oldboy
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A
Database changed
mysql> select * from test where id=4;  把 id 号为 4，name=zhangxuan 的更改为 name=pengchun
+----+------+-----------+-------------+
| id | age  | name      | shouji      |
+----+------+-----------+-------------+
|  4 |   24 | zhangxuan | 13511111111 |
+----+------+-----------+-------------+
1 row in set (0.00 sec)
mysql> update  test set name='pengchun' where id=4;
```

```
Query OK, 1 row affected (0.01 sec)
Rows matched: 1  Changed: 1  Warnings: 0
mysql> select * from test where id=4;
+----+------+----------+-------------+
| id | age  | name     | shouji      |
+----+------+----------+-------------+
|  4 |   24 | pengchun | 13511111111 |
+----+------+----------+-------------+
1 row in set (0.00 sec)
```

2.2.2 删除表中的数据

2.2.2.1 实践删除表中的数据

1、命令语法：delete from <表名> where <表达式> 提示：一定要加 where 条件，否则整张表都删了

实践 1 例如：删除表 test 中编号为 1 的记录

```
mysql> select * from test;
+----+------+-----------+-------------+
| id | age  | name      | shouji      |
+----+------+-----------+-------------+
|  1 | NULL | oldgirl   | NULL        |
|  2 | NULL | 老男孩    | NULL        |
|  3 | NULL | etiantian | NULL        |
|  4 |   24 | pengchun  | 13511111111 |
+----+------+-----------+-------------+
4 rows in set (0.00 sec)
mysql> delete from test where id=1;
Query OK, 1 row affected (0.00 sec)
mysql> select * from test;
+----+------+-----------+-------------+
| id | age  | name      | shouji      |
+----+------+-----------+-------------+
|  2 | NULL | 老男孩    | NULL        |
|  3 | NULL | etiantian | NULL        |
|  4 |   24 | pengchun  | 13511111111 |
+----+------+-----------+-------------+
3 rows in set (0.00 sec)
```

实践 2 删除表中 name=pengchun 的行

```
mysql> select * from test;
+----+------+-----------+-------------+
| id | age  | name      | shouji      |
+----+------+-----------+-------------+
|  2 | NULL | 老男孩    | NULL        |
|  3 | NULL | etiantian | NULL        |
|  4 |   24 | pengchun  | 13511111111 |
+----+------+-----------+-------------+
```

```
3 rows in set (0.00 sec)
mysql> delete from test where name='pengchun';
Query OK, 1 row affected (0.00 sec)
mysql> select * from test;
+----+------+-----------+--------+
| id | age  | name      | shouji |
+----+------+-----------+--------+
|  2 | NULL | 老男孩    | NULL   |
|  3 | NULL | etiantian | NULL   |
+----+------+-----------+--------+
2 rows in set (0.00 sec)
```

2.2.2.2 通过 update 伪删除数据

```
在开发人员开发程序时，页面显示，一般是通过状态来判断的，举例：test 表如下数据
mysql> select * from test;
+----+------+-----------+--------+
| id | age  | name      | state
+----+------+-----------+--------+
|  2 | NULL | 老男孩    | 1      |
|  3 | NULL | etiantian | 1      |
+----+------+-----------+--------+
2 rows in set (0.00 sec)
页面正常显示的数据：select * from test where state=1;
删除上述 oldgirl 的记录：update test set state=0 where name='oldgirl';
```

2.2.3 增删改表的字段

2.2.3.1 命令语法及默认添加演示

```
1、命令语法：alter table <表名> add <字段> <类型>  其他
2、测试表数据：
mysql> use oldboy;
Database changed
mysql> show create table test\G;
*************************** 1. row ***************************
       Table: test
Create Table: CREATE TABLE `test` (
  `id` int(4) NOT NULL DEFAULT '0',
  `age` tinyint(2) DEFAULT NULL,
  `name` varchar(16) DEFAULT NULL,
  `shouji` char(11) DEFAULT NULL,
  PRIMARY KEY (`id`),
  KEY `index_name` (`name`)
) ENGINE=MyISAM DEFAULT CHARSET=utf8
1 row in set (0.09 sec)
ERROR:
No query specified
mysql> desc test;
```

```
+--------+-------------+------+-----+---------+-------+
| Field  | Type        | Null | Key | Default | Extra |
+--------+-------------+------+-----+---------+-------+
| id     | int(4)      | NO   | PRI | 0       |       |
| age    | tinyint(2)  | YES  |     | NULL    |       |
| name   | varchar(16) | YES  | MUL | NULL    |       |
| shouji | char(11)    | YES  |     | NULL    |       |
+--------+-------------+------+-----+---------+-------+
4 rows in set (0.13 sec)
```

3、实践案例

例如：在表 test 中添加字段 sex

4、执行的命令演示

添加性别列，默认语句

```
mysql> desc test;
+--------+-------------+------+-----+---------+-------+
| Field  | Type        | Null | Key | Default | Extra |
+--------+-------------+------+-----+---------+-------+
| id     | int(4)      | NO   | PRI | 0       |       |
| age    | tinyint(2)  | YES  |     | NULL    |       |
| name   | varchar(16) | YES  | MUL | NULL    |       |
| shouji | char(11)    | YES  |     | NULL    |       |
| sex    | char(4)     | YES  |     | NULL    |       |
+--------+-------------+------+-----+---------+-------+
6       rows in set (0.06 sec)
```

4、指定添加年龄到 name 列后面的位置

```
mysql> desc test;
+--------+-------------+------+-----+---------+-------+
| Field  | Type        | Null | Key | Default | Extra |
+--------+-------------+------+-----+---------+-------+
| id     | int(4)      | NO   | PRI | 0       |       |
| name   | varchar(16) | YES  | MUL | NULL    |       |
| shouji | char(11)    | YES  |     | NULL    |       |
| sex    | char(4)     | YES  |     | NULL    |       |
+--------+-------------+------+-----+---------+-------+
4 rows in set (0.00 sec)
mysql> alter table test add age int(4) after name;
Query OK, 2 rows affected (0.07 sec)
Records: 2    Duplicates: 0    Warnings: 0
mysql> desc test;
+--------+-------------+------+-----+---------+-------+
| Field  | Type        | Null | Key | Default | Extra |
+--------+-------------+------+-----+---------+-------+
| id     | int(4)      | NO   | PRI | 0       |       |
| name   | varchar(16) | YES  | MUL | NULL    |       |
```

```
| age     | int(4)       | YES  |     | NULL    |       |
| shouji | char(11)     | YES  |     | NULL    |       |
| sex     | char(4)      | YES  |     | NULL    |       |
+--------+-------------+------+-----+---------+-------+
5 rows in set (0.00 sec)
```

5、 改变字段

语法：

```
alter table <表名> CHANGE [COLUMN] old_col_name new_col_name column_definition
```

6、 修改字段类型

```
mysql> desc test;
+--------+-------------+------+-----+---------+-------+
| Field  | Type        | Null | Key | Default | Extra |
+--------+-------------+------+-----+---------+-------+
| id     | int(4)      | NO   | PRI | 0       |       |
| name   | varchar(16) | YES  | MUL | NULL    |       |
| age    | int(4)      | YES  |     | NULL    |       |     修改前
| shouji | char(11)    | YES  |     | NULL    |       |
| sex    | char(4)     | YES  |     | NULL    |       |
+--------+-------------+------+-----+---------+-------+
5 rows in set (0.00 sec)
mysql> alter table test modify age char(4);
Query OK, 2 rows affected (0.06 sec)
Records: 2  Duplicates: 0  Warnings: 0
mysql> desc test;
+--------+-------------+------+-----+---------+-------+
| Field  | Type        | Null | Key | Default | Extra |
+--------+-------------+------+-----+---------+-------+
| id     | int(4)      | NO   | PRI | 0       |       |
| name   | varchar(16) | YES  | MUL | NULL    |       |
| age    | char(4)     | YES  |     | NULL    |       |     修改后
| shouji | char(11)    | YES  |     | NULL    |       |
| sex    | char(4)     | YES  |     | NULL    |       |
+--------+-------------+------+-----+---------+-------+
5 rows in set (0.00 sec)
```

2.2.4 更改表名

rename 语法:

```
RENAME TABLE old_table TO backup_table, new_table TO old_table;
mysql> show tables;
+-----------------+
| Tables_in_oldboy |
+-----------------+
| test            |
+-----------------+
1 row in set (0.00 sec)
```

```
mysql> rename table test to zhangxuan;
Query OK, 0 rows affected (0.06 sec)
mysql> show tables;
+-----------------+
| Tables_in_oldboy |
+-----------------+
| zhangxuan        |
+-----------------+
1 row in set (0.00 sec)
```

alter 语法：

```
Syntax:
ALTER [ONLINE | OFFLINE] [IGNORE] TABLE tbl_name | RENAME [TO] new_tbl_name
```

2.2.5 删除表名

```
DROP [TEMPORARY] TABLE [IF EXISTS]
    tbl_name [, tbl_name] ...
    [RESTRICT | CASCADE]
```

实例：

```
mysql> use oldboy;
Database changed
mysql> show tables;
+-----------------+
| Tables_in_oldboy |
+-----------------+
| zhangxuan        |
+-----------------+
1 row in set (0.00 sec)
mysql> drop table zhangxuan;
Query OK, 0 rows affected (0.06 sec)
mysql> show tables;
Empty set (0.00 sec)
```

2.2.6 mysql 数据库的备份与恢复

2.2.6.1 备份单个数据库练习多种参数使用

mysql 数据库自带了一个很好用的备份命令，就是 mysqldump，它的基本使用如下：

语法：mysqldump –u <用户名> -p <数据库名> 备份的文件名

范例一：

2.2.6.2 查看数据库 oldboy 的内容

```
mysql> use oldboy
Database changed
mysql> show tables;
+-----------------+
| Tables_in_oldboy |
+-----------------+
| test             |
+-----------------+
```

```
1 row in set (0.00 sec)
mysql> select * from test;
+----+------+-----------+-------------+
| id | age  | name      | shouji      |
+----+------+-----------+-------------+
|  1 | NULL | oldgirl   | NULL        |
|  2 | NULL | 老男孩    | NULL        |
|  3 | NULL | etiantian | NULL        |
|  4 |   24 | zhangxuan | 13511111111 |
|  5 |   22 | huangyan  | 13655555555 |
+----+------+-----------+-------------+
5 rows in set (0.05 sec)
```

2.2.6.3 执行备份的命令

```
[root@mysql ~]# mysqldump -u root -ppcwangjixuan -S /mysqldata/3306/mysql.sock oldboy >/opt/oldboy_bak.sql
```

2.2.6.4 查看备份的结果

```
[root@mysql ~]# grep -vE "#|\*|--|^$"  /opt/oldboy_bak.sql
DROP TABLE IF EXISTS `test`;
CREATE TABLE `test` (
  `id` int(4) NOT NULL DEFAULT '0',
  `age` tinyint(2) DEFAULT NULL,
  `name` varchar(16) DEFAULT NULL,
  `shouji` char(11) DEFAULT NULL,
  PRIMARY KEY (`id`),
  KEY `index_name_and_shouji` (`name`(6),`shouji`(8))
) ENGINE=MyISAM DEFAULT CHARSET=utf8;
LOCK TABLES `test` WRITE;
INSERT INTO `test` VALUES (1,NULL,'oldgirl',NULL),(2,NULL,'老男孩',NULL),(3,NULL,'etiantian',NULL),(4,24,'zhangxuan','13511111111'),(5,22,'huangyan','13655555555');
UNLOCK TABLES;
```

2.2.6.5 mysqldump 备份时加上-B 参数时的备份，然后比较不加-B 备份的不同

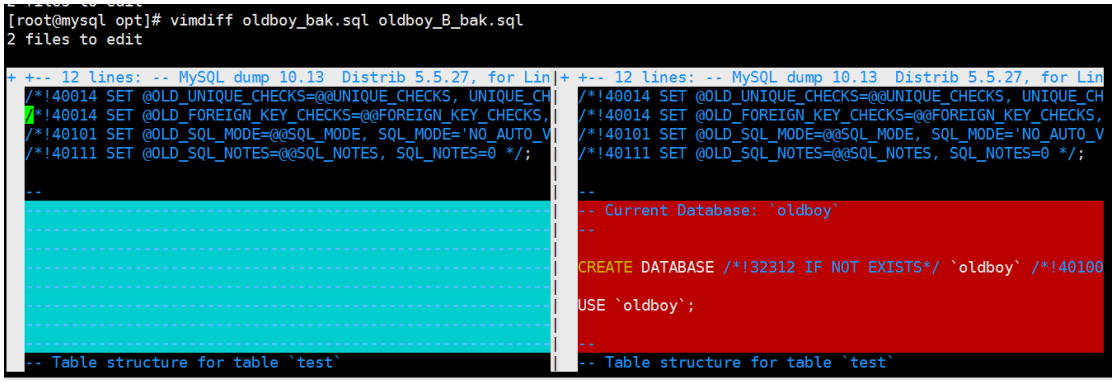
加上-B 备份的操作

```
[root@mysql opt]# mysqldump -u root -ppcwangjixuan -S /mysqldata/3306/mysql.sock -B oldboy >/opt/oldboy_B_bak.sql
```

比较两者备份的不同

```
[root@mysql opt]# vimdiff oldboy_bak.sql oldboy_B_bak.sql
```

从下面结果看出，加上-B 后，备份的时候，会有创建数据库并 use 库的过程



2.2.6.6 删除数据库中备份过的库 oldboy，然后将备份的数据重新导入数据库

删除数据库 oldboy

```
mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| oldboy             |
| oldboy_gbk         |
| performance_schema |
| wordpress          |
+--------------------+
6 rows in set (0.00 sec)
mysql> drop database oldboy;
Query OK, 1 row affected (0.44 sec)
mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| oldboy_gbk         |
| performance_schema |
| wordpress          |
+--------------------+
5 rows in set (0.00 sec)
```

用不同备份的数据分别导入查看其情况

```
[root@mysql ~]# mysql -u root -ppcwangjixuan -S /mysqldata/3306/mysql.sock </opt/oldboy_bak.sql
ERROR 1046 (3D000) at line 22: No database selected
```

从结果看出，没加-B 参数备份的数据，是不能导进去的，需要重新创建库，并且导入的时候指定库

```
[root@mysql ~]# mysql -u root -ppcwangjixuan -S /mysqldata/3306/mysql.sock </opt/oldboy_B_bak.sql
```

从结果看出，加了-B 参数备份的数据，可以直接导入数据库，查看

其导入结果

```
mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| oldboy             |
| oldboy_gbk         |
| performance_schema |
| wordpress          |
+--------------------+
7       rows in set (0.00 sec)
mysql> use oldboy
Database changed
mysql> select * from test;
+----+------+-----------+-------------+
| id | age  | name      | shouji      |
+----+------+-----------+-------------+
|  1 | NULL | oldgirl   | NULL        |
|  2 | NULL | 老男孩    | NULL        |
|  3 | NULL | etiantian | NULL        |
|  4 |   24 | zhangxuan | 13511111111 |
|  5 |   22 | huangyan  | 13655555555 |
+----+------+-----------+-------------+
5 rows in set (0.00 sec)
```

2.2.6.7 利用 mysqldump 命令对指定的库进行压缩备份

```
[root@mysql ~]# mysqldump -uroot -ppcwangjixuan -S /mysqldata/3306/mysql.sock -B oldboy|gzip>/opt/oldboy_B_bak.sql.gz
[root@mysql ~]# ls -l /opt/oldboy_B_bak.sql.gz
-rw-r--r--. 1 root root 881 Jun 15 21:04 /opt/oldboy_B_bak.sql.gz
```

2.2.6.8 利用 mysqldump 命令备份多个库（-B 参数后可以指定多个库）

备份多个库 oldboy 和 oldboy_gbk

```
[root@mysql ~]# mysqldump -uroot -ppcwangjixuan -S /mysqldata/3306/mysql.sock -B oldboy oldboy_gbk>/opt/oldboy_and_oldboy_gbk.bak.sql
```

登录数据库删除 oldboy 和 oldboy_gbk

```
mysql> drop database oldboy;
Query OK, 1 row affected (0.00 sec)
mysql> drop database oldboy_gbk;
Query OK, 0 rows affected (0.01 sec)
mysql> show databases;
+--------------------+
| Database           |
+--------------------+
```

```
| information_schema |
| mysql                          |
| performance_schema |
| wordpress                  |
+--------------------+
4 rows in set (0.00 sec)
```

将备份的数据直接导入数据库查看其结果（从下面结果看出，数据恢复过来）

```
[root@mysql ~]# mysql -u root -ppcwangjixuan -S /mysqldata/3306/mysql.sock </opt/oldboy_and_oldboy_gbk.bak.sql
[root@mysql ~]# mysql -u root -ppcwangjixuan -S /mysqldata/3306/mysql.sock
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 38
Server version: 5.5.27-log Source distribution
Copyright (c) 2000, 2011, Oracle and/or its affiliates. All rights reserved.
Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.
mysql> show databases;
+--------------------+
| Database                   |
+--------------------+
| information_schema |
| mysql                          |
| oldboy                       |
| oldboy_gbk                 |
| performance_schema |
| wordpress                  |
+--------------------+
6 rows in set (0.00 sec)
```

2.2.6.9 分库备份（对 mysql、oldboy、oldboy_gbk、wordpress 库进行备份）

利用命令行分库备份

```
[root@mysql ~]# mysql -uroot -ppcwangjixuan -S /mysqldata/3306/mysql.sock -e "show databases"|egrep -v "Data|inf|per"|sed -r 's#(.*)#mysqldump -uroot -ppcwangjixuan -S /mysqldata/3306/mysql.sock -B &#g'|bash
```

2.2.7.0 对一个库的多个表备份

利用命令对单个库的多个表备份

```
[root@mysql ~]# mysql -u root -ppcwangjixuan -S /mysqldata/3306/mysql.sock -e "use oldboy;show tables"
+------------------+
| Tables_in_oldboy |
+------------------+
| pengchun                 |
| test                         |
```

| zhangxuan |
+------------------+
备份命令
[root@mysql ~]# mysql -u root -ppcwangjixuan -S /mysqldata/3306/mysql.sock -e "use oldboy;show tables"|sed '1'd|sed -r 's#(.*)#mysqldump -u root -ppcwangjixuan -S /mysqldata/3306/mysql.sock oldboy & >/opt/&.sql#g'|bash

2.2.7.1 备份多个表

语法：mysqldump –u 用户名 –p 数据库名 表名 1 表名 2 >备份的文件名
操作结果：
mysql> use oldboy;
Database changed
mysql> show tables;
+------------------+
| Tables_in_oldboy |
+------------------+
| pengchun |
| test |
| zhangxuan |
+------------------+
3 rows in set (0.09 sec)
[root@mysql ~]# mysqldump -u root -ppcwangjixuan -S /mysqldata/3306/mysql.sock oldboy pengchun test zhangxuan >/opt/oldboy.$(date +%F).sql

2.2.7.2 备份单个表

语法：mysqldump –u 用户名 –p 数据库名 表名 >备份的文件名
执行结果：
[root@mysql ~]#mysqldump –u root –ppcwangjixuan oldboy student >opt/table.sql
提示：不能加-B 参数了，因为库 oldboy 后面就是 oldboy 表了，指定-B 就表示后面的都是库

2.2.7.3 关于 mysqldump 的参数说明

1、-B 备份多个库（添加 create 和 use 库的语句）
2、-d 只备份库表结构
3、-t 只备份数据（sql 语句形式）
4、-T 分离库表和数据不同的文件，数据是文本，非 SQL 语句
5、-A 备份数据库中所有的数据
6、--compact 去掉注释，适合调试输出，生产不使用
7、-F 刷新 binglog 日志，生成新文件，将来增量恢复从这个文件开始
8、--master-data 增加 binlog 日志文件名及对应的位置点（即 CHANGE MASTER 语句）
--master-data=1 不注释，--master-data=2 注释
9、-x –lock-all-tables 锁表，当某一时刻备份数据时需要加入此参数，以确定备份数据时是从某一时刻开始的
10、-l –lock-tables 对读锁表
11、--single-transaction 适合 innodb 事务数据库备份
innodb 表在备份时，通常启动选项—single-transaction 来保证备份的一致性，实际上它的工作原理是设定本次会话的隔离级别为 REPEATABLE READ,以确保本次会话（dump）时，不会看到其他会话已经提交了的数据

12、-q 不做缓冲查询，直接导入输出

2.2.7.4 刷新 binglog 的参数

binglog 是什么，记录数据库更新的 SQL 语句
mysqldump 用于定时对某一时刻的数据的全备，例如：00 点进行备份 bak.sql
增量备份：当有数据写入到数据库时，还会同时把更新的 SQL 语句写入到对应的文件里，这个文件就叫做 binglog 文件
10 点丢失数据需要恢复数据
1、00 点时刻备份的 bak.sql 数据还原到数据库，这个时候数据恢复到了 00 点
2、00 点-10:00 数据，就要从 binglog 里恢复
binglog 文件生效需要一个参数：log-bin
[root@mysql ~]# grep log-bin /mysqldata/3306/my.cnf
log-bin = /mysqldata/3306/mysql-bin
binglog 文件：
[root@mysql ~]# ll /mysqldata/3306/mysql-bin.*
-rw-rw----. 1 mysql mysql 1434 Jun 13 07:06 /mysqldata/3306/mysql-bin.000001
-rw-rw----. 1 mysql mysql 622 Jun 13 07:14 /mysqldata/3306/mysql-bin.000002
-rw-rw----. 1 mysql mysql 434 Jun 13 18:37 /mysqldata/3306/mysql-bin.000003
-rw-rw----. 1 mysql mysql 126 Jun 13 18:43 /mysqldata/3306/mysql-bin.000004
-rw-rw----. 1 mysql mysql 2194 Jun 14 03:39 /mysqldata/3306/mysql-bin.000005
-rw-rw----. 1 mysql mysql 483 Jun 14 03:48 /mysqldata/3306/mysql-bin.000006
-rw-rw----. 1 mysql mysql 529082 Jun 14 21:38 /mysqldata/3306/mysql-bin.000007
-rw-rw----. 1 mysql mysql 150 Jun 14 21:39 /mysqldata/3306/mysql-bin.000008
-rw-rw----. 1 mysql mysql 126 Jun 14 21:39 /mysqldata/3306/mysql-bin.000009
-rw-rw----. 1 mysql mysql 150 Jun 14 21:40 /mysqldata/3306/mysql-bin.000010
-rw-rw----. 1 mysql mysql 126 Jun 14 21:41 /mysqldata/3306/mysql-bin.000011
-rw-rw----. 1 mysql mysql 366 Jun 14 21:43 /mysqldata/3306/mysql-bin.000012
-rw-rw----. 1 mysql mysql 1032064 Jun 14 23:18 /mysqldata/3306/mysql-bin.000013
-rw-rw----. 1 mysql mysql 444 Jun 14 23:20 /mysqldata/3306/mysql-bin.000014
-rw-rw----. 1 mysql mysql 524680 Jun 15 21:41 /mysqldata/3306/mysql-bin.000015
-rw-rw----. 1 mysql mysql 495 Jun 14 23:20 /mysqldata/3306/mysql-bin.index
binglog 日志切割：确定全备和增量的临界点
-F 刷新 binglog 日志，生成新文件，将来增量恢复从这个文件开始
--master-data 在备份语句里添加 CHANGE MASTER 语句及 binglog 文件及位置点信息
值为 1，为可执行的 CHANGE MASTER 语句
值为 2，注释的—CHANGE MASTER 语句
--master-data 除了增量恢复确定临界点外，做主从复制时作用更大

2.2.7.5 生产场景不同引擎 mysqldump 备份命令

myisam 引擎企业生产备份命令（适合所有引擎或混合引擎）
mysqldump –uroot –ppcwangjixuan –A –B –F –R –master-data=2 –x –events|gzip >/opt/all.sql.gz
提示-F 也可以不用，与—master-data 有些重复
innodb 引擎企业生产备份命令：推荐使用的
mysqldump –uroot –ppcwangjixuan –A –B –F –R --master-data=2 –events –single-transaction|gzip>/opt/all.sql.gz
提示:-F 也可以不用，与—master-data 有些重复

其中—master-data 作用：
使用—master-data=2 进行备份文件会增加如下内容，适合普通备份增量恢复
-- CHANGE MASTER TO MASTER_LOG_FILE='mysql-bin.000024', MASTER_LOG_POS=107;
额外补充：
50G 以内的数据
1、mysqldump 逻辑备份，缺点：效率不是特别高。优点：简单、方便、可靠、迁移，适用与数据量不是特别大的场景
超过 50G 的数据
1、 xtrabackup 物理备份工具：全量和增量物理备份方案：
2、 从库停止 SQL 线程，打包，cp
什么时候会使用 mysqldump 的数据？
1、 恢复数据到测试库
2、 人为通过 SQL 将数据删除的时候
3、 主从复制

2.2.8 恢复数据库实践

2.2.8.1 数据库恢复事项

提示：
数据恢复和字符集关联很大，如果字符集不正确会导致恢复的数据乱码
mysql 命令以及 source 命令恢复数据库的原理就是把文件的 SQL 语句，在数据库里重新执行的过程

2.2.8.2 利用 source 命令恢复数据库

进入 mysql 数据库控制台，mysql –uroot –p 登陆后
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 oldboy_db.sql 这个文件是系统路径，默认是登录 mysql 前的系统路径
下面来做个例子利用 source 命令来对数据库中的 oldboy 库进行恢复
1、 删除库 oldboy 前对数据库中的 oldboy 库进行备份
[root@mysql ~]# mysqldump -u root -ppcwangjixuan -B oldboy -S /mysqldata/3306/mysql.sock >/tmp/oldboy_bak.sql
2、 进入数据库控制台，删除数据库 oldboy，然后进行恢复
mysql> show databases;
+--------------------+
| Database |
+--------------------+
| information_schema |
| mysql |
| oldboy |
| oldboy_gbk |
| performance_schema |
| wordpress |
+--------------------+
6 rows in set (0.00 sec)
mysql> drop database oldboy;

```
Query OK, 3 rows affected (0.63 sec)

mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| oldboy_gbk         |
| performance_schema |
| wordpress          |
+--------------------+
5 rows in set (0.00 sec)
mysql> source /tmp/oldboy_bak.sql
Query OK, 0 rows affected (0.00 sec)
Query OK, 0 rows affected (0.00 sec)
Query OK, 0 rows affected (0.00 sec)
Query OK, 0 rows affected (0.00 sec)
Query OK, 0 rows affected (0.00 sec)
Query OK, 0 rows affected (0.00 sec)
mysql> show databases;   （从下面结果看出，数据库恢复完毕）
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| oldboy             |
| oldboy_gbk         |
| performance_schema |
| wordpress          |
+--------------------+
6 rows in set (0.00 sec)
```

2.2.8.3 利用 mysql 命令恢复（标准）

```
mysql> drop database oldboy;
Query OK, 3 rows affected (0.07 sec)
mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| oldboy_gbk         |
| performance_schema |
| wordpress          |
```

```
+--------------------+
5 rows in set (0.00 sec)
mysql> drop database oldboy;
Query OK, 3 rows affected (0.07 sec)
mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| oldboy_gbk         |
| performance_schema |
| wordpress          |
+--------------------+
5 rows in set (0.00 sec)
mysql> quit
Bye
[root@mysql ~]# mysql -u root -ppcwangjixuan -S /mysqldata/3306/mysql.sock </tmp/oldboy_bak.sql
[root@mysql ~]# mysql -u root -ppcwangjixuan -S /mysqldata/3306/mysql.sock
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 87
Server version: 5.5.27-log Source distribution
Copyright (c) 2000, 2011, Oracle and/or its affiliates. All rights reserved.
Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.
mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| oldboy             |
| oldboy_gbk         |
| performance_schema |
| wordpress          |
+--------------------+
6 rows in set (0.00 sec)
```

注意:

假定开发人员让我们插入数据到数据库（可能是邮件发给我们的，内容可能是字符串或者下面的文件）

sql 文件里没有 use db 这样的字样时，在导入时就要指定数据库名了

2.2.8.4 针对压缩的备份数据恢复

删除源文件的备份方法：

```
gzip –d    /opt/mysql_bak.sql.gz
mysql –uroot –ppcwangjixuan </opt/mysql_bak.sql
```

不删除源文件的备份发法：

```
gzip –cd 01.sql.gz >02.sql
```

2.2.9 实现和 mysql 非交互式对话

2.2.9.1 利用 mysql -e 参数查看 mysql 数据库的库名

```
[root@mysql ~]# mysql -u root -ppcwangjixuan -S /mysqldata/3306/mysql.sock -e "show databases"
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| oldboy             |
| oldboy_gbk         |
| performance_schema |
| wordpress          |
+--------------------+
```

2.2.9.2 利用 mysql -e 参数查看 mysql 数据库的线程状态

```
[root@mysql ~]# mysql -u root -ppcwangjixuan -S /mysqldata/3306/mysql.sock -e "show full processlist"
+----+------+-----------+------+---------+------+-------+----------------------+
| Id | User | Host      | db   | Command | Time | State | Info                 |
+----+------+-----------+------+---------+------+-------+----------------------+
| 89 | root | localhost | NULL | Query   |    0 | NULL  | show full processlist |
+----+------+-----------+------+---------+------+-------+----------------------+
```

2.2.9.3 mysql sleep 线程过多的问题案例

```
[root@mysql ~]# mysql -u root -ppcwangjixuan -S /mysqldata/3306/mysql.sock -e "show full processlist"
+----+------+-----------+------+---------+------+-------+----------------------+
| Id | User | Host      | db   | Command | Time | State | Info                 |
+----+------+-----------+------+---------+------+-------+----------------------+
| 89 | root | localhost | NULL | Query   |    0 | NULL  | show full processlist |
+----+------+-----------+------+---------+------+-------+----------------------+
```

解决办法：

```
mysql> show variables like '%_timeout%';
+----------------------------+----------+
| Variable_name              | Value    |
+----------------------------+----------+
| connect_timeout            | 10       |
| delayed_insert_timeout     | 300      |
| innodb_lock_wait_timeout   | 120      |
```

```
| innodb_rollback_on_timeout | OFF          |
| interactive_timeout        | 28800        |
| lock_wait_timeout          | 31536000 |
| net_read_timeout           | 30           |
| net_write_timeout          | 60           |
| slave_net_timeout          | 3600         |
| wait_timeout               | 28800        |
+----------------------------+----------+
10 rows in set (0.00 sec)
```

解决办法：

第一步：mysql 命令行

```
mysql> set global interactive_timeout = 60;
Query OK, 0 rows affected (0.04 sec)
mysql> set global wait_timeout = 60;
Query OK, 0 rows affected (0.00 sec)
```

第二步：

配置文件里面修改：

```
[mysqld]
interactive_timeout = 60
wait_timeout = 60
```

第三步：其它方法

1、 php 程序中，不使用持久链接，即使用 mysql_connect 而不是 pconnect（JAVA 调整连接池）

2、 php 程序执行完毕，应该显示调用 mysql_close

3、 逐步分析 mysql 的 sql 查询及慢查询日志，找到查询国漫的 SQL 优化之

2.2.9.4 查看 mysql 配置文件有没有在数据库中生效

查看 slow_log 有没有开启

```
[root@mysql ~]# mysql -u root -ppcwangjixuan -S /mysqldata/3306/mysql.sock -e "show variables like '%slow%';"
+---------------------+-----------------------------------+
| Variable_name       | Value                             |
+---------------------+-----------------------------------+
| log_slow_queries    | OFF                               |
| slow_launch_time    | 2                                 |
| slow_query_log      | OFF                               |
| slow_query_log_file | /mysqldata/3306/data/mysql-slow.log |
+---------------------+-----------------------------------+
```

查看 mysql 的 server_id

```
[root@mysql ~]# mysql -u root -ppcwangjixuan -S /mysqldata/3306/mysql.sock -e "show variables like '%server_id%';"
+---------------+-------+
| Variable_name | Value |
+---------------+-------+
| server_id     | 1     |
```

```
+---------------+-------+
```

2.2.9.5 不重启数据库修改数据库参数

不重启数据库修改数据库参数，但是要求重启后还能生效

举例说明：（以慢查询参数是否开启为例）

第一步：

```
[root@mysql ~]# mysql -u root -ppcwangjixuan -S /mysqldata/3306/mysql.sock -e "show variables like '%slow%';"
+---------------------+-----------------------------------+
| Variable_name | Value |
+---------------------+-----------------------------------+
| log_slow_queries | OFF |
| slow_launch_time | 2 |
| slow_query_log | OFF |
| slow_query_log_file | /mysqldata/3306/data/mysql-slow.log |
+---------------------+-----------------------------------+
[root@mysql ~]# mysql -u root -ppcwangjixuan -S /mysqldata/3306/mysql.sock -e "set global slow_query_log =ON;set global log_slow_queries=ON"
[root@mysql ~]# mysql -u root -ppcwangjixuan -S /mysqldata/3306/mysql.sock -e "show variables like '%slow%';"
+---------------------+-----------------------------------+
| Variable_name | Value |
+---------------------+-----------------------------------+
| log_slow_queries | ON |
| slow_launch_time | 2 |
| slow_query_log | ON |
| slow_query_log_file | /mysqldata/3306/data/mysql-slow.log |
+---------------------+-----------------------------------+
```

第二步：

```
[root@mysql ~]# sed -i 's/#long_query_time = 1/long_query_time = 1/g' /mysqldata/3306/my.cnf
[root@mysql ~]# sed -i 's&#log-slow-queries = /data/3306/slow.log&log-slow-queries = /data/3306/slow.log&g' /mysqldata/3306/my.cnf
```

举例说明：（以 myisam 引擎缓冲大小为例）

第一步：

```
[root@mysql ~]# mysql -u root -ppcwangjixuan -S /mysqldata/3306/mysql.sock -e "show variables;"|grep key_buffer
key_buffer_size 16777216
[root@mysql ~]# mysql -u root -ppcwangjixuan -S /mysqldata/3306/mysql.sock -e "set global key_buffer_size = 1024*1024*32"
[root@mysql ~]# mysql -u root -ppcwangjixuan -S /mysqldata/3306/mysql.sock -e "show variables;"|grep key_buffer
key_buffer_size 33554432
```

第二步：

```
[root@mysql ~]# sed -i 's#key_buffer_size = 16M#key_buffer_size = 32M#g' /mysqldata/3306/my.cnf
```

2.2.9.6 不重启数据库更改数据库参数小结

1、mysql -u root -ppcwangjixuan -S /mysqldata/3306/mysql.sock -e "set global key_buffer_size = 1024*1024*32" <==及时生效，重启 mysql 失效

2、配置文件也要改，编辑/etc/my.cnf,修改 key_buffer_size = 32 M

2.3.0 查看 mysql 状态的信息（利用 zabbix 可以监控其状态信息）

```
mysql> show global status;
+-----------------------------------------+-------------+
| Variable_name                           | Value       |
+-----------------------------------------+-------------+
| Aborted_clients                         | 0           |
| Aborted_connects                        | 0           |
| Binlog_cache_disk_use                   | 0           |
| Binlog_cache_use                        | 0           |
| Binlog_stmt_cache_disk_use              | 0           |
| Binlog_stmt_cache_use                   | 0           |
| Bytes_received                          | 1113        |
| Bytes_sent                              | 21450       |
| Com_admin_commands                      | 0           |
| Com_assign_to_keycache                  | 0           |
| Com_alter_db                            | 0           |
| Com_alter_db_upgrade                    | 0           |
| Com_alter_event                         | 0           |
| Com_alter_function                      | 0           |
| Com_alter_procedure                     | 0           |
| Com_alter_server                        | 0           |
| Com_alter_table                         | 0           |
| Com_alter_tablespace                    | 0           |
| Com_analyze                             | 0           |
| Com_begin                               | 0           |
| Com_binlog                              | 0           |
| Com_call_procedure                      | 0           |
| Com_change_db                           | 0           |
| Com_change_master                       | 0           |
| Com_check                               | 0           |
| Com_checksum                            | 0           |
| Com_commit                              | 0           |
| Com_create_db                           | 0           |
| Com_create_event                        | 0           |
| Com_create_function                     | 0           |
| Com_create_index                        | 0           |
| Com_create_procedure                    | 0           |
| Com_create_server                       | 0           |
| Com_create_table                        | 0           |
| Com_create_trigger                      | 0           |
```

Com_create_udf	0
Com_create_user	0
Com_create_view	0
Com_dealloc_sql	0
Com_delete	0
Com_delete_multi	0
Com_do	0
Com_drop_db	0
Com_drop_event	0
Com_drop_function	0
Com_drop_index	0
Com_drop_procedure	0
Com_drop_server	0
Com_drop_table	0
Com_drop_trigger	0
Com_drop_user	0
Com_drop_view	0
Com_empty_query	0
Com_execute_sql	0
Com_flush	0
Com_grant	0
Com_ha_close	0
Com_ha_open	0
Com_ha_read	0
Com_help	0
Com_insert	0
Com_insert_select	0
Com_install_plugin	0
Com_kill	0
Com_load	0
Com_lock_tables	0
Com_optimize	0
Com_preload_keys	0
Com_prepare_sql	0
Com_purge	0
Com_purge_before_date	0
Com_release_savepoint	0
Com_rename_table	0
Com_rename_user	0
Com_repair	0
Com_replace	0
Com_replace_select	0
Com_reset	0
Com_resignal	0

Com_revoke	0
Com_revoke_all	0
Com_rollback	0
Com_rollback_to_savepoint	0
Com_savepoint	0
Com_select	7
Com_set_option	4
Com_signal	0
Com_show_authors	0
Com_show_binlog_events	0
Com_show_binlogs	0
Com_show_charsets	0
Com_show_collations	0
Com_show_contributors	0
Com_show_create_db	0
Com_show_create_event	0
Com_show_create_func	0
Com_show_create_proc	0
Com_show_create_table	0
Com_show_create_trigger	0
Com_show_databases	0
Com_show_engine_logs	0
Com_show_engine_mutex	0
Com_show_engine_status	0
Com_show_events	0
Com_show_errors	0
Com_show_fields	0
Com_show_function_status	0
Com_show_grants	0
Com_show_keys	0
Com_show_master_status	0
Com_show_open_tables	0
Com_show_plugins	0
Com_show_privileges	0
Com_show_procedure_status	0
Com_show_processlist	0
Com_show_profile	0
Com_show_profiles	0
Com_show_relaylog_events	0
Com_show_slave_hosts	0
Com_show_slave_status	0
Com_show_status	1
Com_show_storage_engines	0
Com_show_table_status	0

Com_show_tables	0
Com_show_triggers	0
Com_show_variables	3
Com_show_warnings	0
Com_slave_start	0
Com_slave_stop	0
Com_stmt_close	0
Com_stmt_execute	0
Com_stmt_fetch	0
Com_stmt_prepare	0
Com_stmt_reprepare	0
Com_stmt_reset	0
Com_stmt_send_long_data	0
Com_truncate	0
Com_uninstall_plugin	0
Com_unlock_tables	0
Com_update	0
Com_update_multi	0
Com_xa_commit	0
Com_xa_end	0
Com_xa_prepare	0
Com_xa_recover	0
Com_xa_rollback	0
Com_xa_start	0
Compression	OFF
Connections	8
Created_tmp_disk_tables	0
Created_tmp_files	6
Created_tmp_tables	4
Delayed_errors	0
Delayed_insert_threads	0
Delayed_writes	0
Flush_commands	1
Handler_commit	0
Handler_delete	0
Handler_discover	0
Handler_prepare	0
Handler_read_first	3
Handler_read_key	0
Handler_read_last	0
Handler_read_next	0
Handler_read_prev	0
Handler_read_rnd	0
Handler_read_rnd_next	687

Handler_rollback	0
Handler_savepoint	0
Handler_savepoint_rollback	0
Handler_update	0
Handler_write	658
Innodb_buffer_pool_pages_data	288
Innodb_buffer_pool_pages_dirty	0
Innodb_buffer_pool_pages_flushed	0
Innodb_buffer_pool_pages_free	1759
Innodb_buffer_pool_pages_misc	0
Innodb_buffer_pool_pages_total	2047
Innodb_buffer_pool_read_ahead_rnd	0
Innodb_buffer_pool_read_ahead	0
Innodb_buffer_pool_read_ahead_evicted	0
Innodb_buffer_pool_read_requests	2068
Innodb_buffer_pool_reads	289
Innodb_buffer_pool_wait_free	0
Innodb_buffer_pool_write_requests	0
Innodb_data_fsyncs	3
Innodb_data_pending_fsyncs	0
Innodb_data_pending_reads	0
Innodb_data_pending_writes	0
Innodb_data_read	6918144
Innodb_data_reads	299
Innodb_data_writes	3
Innodb_data_written	1536
Innodb_dblwr_pages_written	0
Innodb_dblwr_writes	0
Innodb_have_atomic_builtins	ON
Innodb_log_waits	0
Innodb_log_write_requests	0
Innodb_log_writes	1
Innodb_os_log_fsyncs	3
Innodb_os_log_pending_fsyncs	0
Innodb_os_log_pending_writes	0
Innodb_os_log_written	512
Innodb_page_size	16384
Innodb_pages_created	0
Innodb_pages_read	288
Innodb_pages_written	0
Innodb_row_lock_current_waits	0
Innodb_row_lock_time	0
Innodb_row_lock_time_avg	0
Innodb_row_lock_time_max	0

Innodb_row_lock_waits	0
Innodb_rows_deleted	0
Innodb_rows_inserted	0
Innodb_rows_read	0
Innodb_rows_updated	0
Innodb_truncated_status_writes	0
Key_blocks_not_flushed	0
Key_blocks_unused	26792
Key_blocks_used	0
Key_read_requests	0
Key_reads	0
Key_write_requests	0
Key_writes	0
Last_query_cost	0.000000
Max_used_connections	1
Not_flushed_delayed_rows	0
Open_files	23
Open_streams	0
Open_table_definitions	34
Open_tables	27
Opened_files	86
Opened_table_definitions	34
Opened_tables	34
Performance_schema_cond_classes_lost	0
Performance_schema_cond_instances_lost	0
Performance_schema_file_classes_lost	0
Performance_schema_file_handles_lost	0
Performance_schema_file_instances_lost	0
Performance_schema_locker_lost	0
Performance_schema_mutex_classes_lost	0
Performance_schema_mutex_instances_lost	0
Performance_schema_rwlock_classes_lost	0
Performance_schema_rwlock_instances_lost	0
Performance_schema_table_handles_lost	0
Performance_schema_table_instances_lost	0
Performance_schema_thread_classes_lost	0
Performance_schema_thread_instances_lost	0
Prepared_stmt_count	0
Qcache_free_blocks	1
Qcache_free_memory	2079912
Qcache_hits	0
Qcache_inserts	0
Qcache_lowmem_prunes	0
Qcache_not_cached	7

```
| Qcache_queries_in_cache                 | 0                |
| Qcache_total_blocks                     | 1                |
| Queries                                 | 21               |
| Questions                               | 21               |
| Rpl_status                              | AUTH_MASTER      |
| Select_full_join                        | 0                |
| Select_full_range_join                  | 0                |
| Select_range                            | 0                |
| Select_range_check                      | 0                |
| Select_scan                             | 4                |
| Slave_heartbeat_period                  | 0.000            |
| Slave_open_temp_tables                  | 0                |
| Slave_received_heartbeats               | 0                |
| Slave_retried_transactions              | 0                |
| Slave_running                           | OFF              |
| Slow_launch_threads                     | 0                |
| Slow_queries                            | 0                |
| Sort_merge_passes                       | 0                |
| Sort_range                              | 0                |
| Sort_rows                               | 0                |
| Sort_scan                               | 0                |
| Table_locks_immediate                   | 37               |
| Table_locks_waited                      | 0                |
| Tc_log_max_pages_used                   | 0                |
| Tc_log_page_size                        | 0                |
| Tc_log_page_waits                       | 0                |
| Threads_cached                          | 0                |
| Threads_connected                       | 1                |
| Threads_created                         | 1                |
| Threads_running                         | 1                |
| Uptime                                  | 1220             |
| Uptime_since_flush_status               | 1220             |
+-----------------------------------------+------------+
287 rows in set (0.00 sec)
```

2.3.1 mysqladmin 的命令

```
mysqladmin password oldboy123  为 mysql 用户添加密码
mysqadmin –uroot –ppcwangjixuan password oldboy  为 mysql 用户修改密码
mysqladmin –uroot –ppcwangjixuan status  查看 mysql 的状态
mysqladmin -u root -ppcwangjixuan -S /mysqldata/3306/mysql.sock -i 1 status 每秒 mysql 的状态
mysqladmin -u root -ppcwangjixuan -S /mysqldata/3306/mysql.sock flush-logs 刷新 binlog 日志
mysqladmin -u root -ppcwangjixuan -S /mysqldata/3306/mysql.sock processlist 查看 mysql 的线程状态
mysqladmin -u root -ppcwangjixuan -S /mysqldata/3306/mysql.sock processlist –i 1 每秒查看 mysql 的线程状态
```

mysqladmin -u root -ppcwangjixuan -S /mysqldata/3306/mysql.sock extended-status 查看 mysql 的所有状态信息
mysqladmin -u root -ppcwangjixuan -S /mysqldata/3306/mysql.sock shutdown 优雅关闭 mysql 服务
mysqladmin -u root -ppcwangjixuan -S /mysqldata/3306/mysql.sock variables 查看 mysql 配置文件的信息

2.3.2 mysql 工具 mysqlbinlog

mysqlbinlog 工具的作用是解析 mysql 的二进制 binlog 日志内容，把二进制日志解析成可以再 mysql 数据库执行的 SQL 语句

2.3.2.1 mysql 的 binlog 日志是什么？

mysql 数据目录下的如下文件就是 mysql 的 binlog 日志

```
[root@mysql ~]# ls -l /mysqldata/3306/mysql-bin.*
-rw-rw----. 1 mysql mysql      1434 Jun 13 07:06 /mysqldata/3306/mysql-bin.000001
-rw-rw----. 1 mysql mysql       622 Jun 13 07:14 /mysqldata/3306/mysql-bin.000002
-rw-rw----. 1 mysql mysql       434 Jun 13 18:37 /mysqldata/3306/mysql-bin.000003
-rw-rw----. 1 mysql mysql       126 Jun 13 18:43 /mysqldata/3306/mysql-bin.000004
-rw-rw----. 1 mysql mysql      2194 Jun 14 03:39 /mysqldata/3306/mysql-bin.000005
-rw-rw----. 1 mysql mysql       483 Jun 14 03:48 /mysqldata/3306/mysql-bin.000006
-rw-rw----. 1 mysql mysql    529082 Jun 14 21:38 /mysqldata/3306/mysql-bin.000007
```

提示：要打开 log-bin 功能，才能生成上面文件

2.3.2.2 mysql 的 binlog 日志作用是什么？

mysql 的 binlog 日志作用是用来记录 mysql 内部增删改等对 mysql 数据库有更新的内容的记录（对数据库的改动），对数据库查询的语句如 show，select 开头的语句，不会被 binlog 日志记录，用于数据库的增量恢复，以及主从复制

2.3.2.3 mysqlbinlog 工具解析 binlog 日志实践

默认情况 binlog 日志是二进制格式的，不能使用查看文本工具的命令查看，例如：cat,vi

```
[root@mysql ~]# file /mysqldata/3306/mysql-bin.000001
/mysqldata/3306/mysql-bin.000001: MySQL replication log
```

2.3.2.4 解析指定库的 binlog 日志

范例：利用 mysqlbinlog –d 参数解析指定库的 binlog 日志（如果想获取某张表，就从库中 grep 表名）

```
[root@mysql ~]# mysqlbinlog -d oldboy /mysqldata/3306/mysql-bin.000001 -r oldboy.sql (-r 文件名，相当于重定向“>”)
[root@mysql ~]# ls -l oldboy.sql
-rw-r--r--. 1 root root 1508 Jun 17 01:45 oldboy.sql
```

结论：mysqlbinlog 工具分库导出 binlog，如果使用-d 参数，那更新数据时，必须有 use database，才能分出指定库的 binlog，例如：

```
use oldboy
insert into student values(1,’oldboy’)
```

下面的写法就不行：

```
insert into oldboy.student values(2,’oldgirl’)
```

按照位置截取：精确

```
mysqlbinlog mysqlbin.000020 –start-position=365 –stop-position=456 –r pos.sql
```

按照时间点截取：不准确，模糊

mysqlbinlog mysqlbin.000020 –start-datetime=’2014-10-16 17:14:15’ –stop-datetime=’2014-10-16 17:15:15’ –r time.sql

2.3.3 mysql 数据库的服务日志

2.3.3.1 错误日志（error log）介绍与调整

1、错误日志（error log）介绍
mysql 的错误日志记录 mysql 服务进程 mysqld 在启动/关闭或运行过程中遇到的错误信息：
2、错误日志（error log）实践
法一：在配置文件中调整发法：当然可以在启动时加入启动参数
[mysqld_safe]
log-error=/data/3306/mysql_oldboy3306.err
法二：启动 mysql 命令里加入：
mysqld_safe –defaults-file=/data/3306/my.cnf –log-error=/data/3306/mysql_oldboy.err &

2.3.3.2 普通查询日志（general query log）介绍与调整（生产环境中不用）

1、普通查询日志介绍
普通查询日志，记录客户端连接信息和执行的 SQL 语句信息
2、普通查询日志调整

```
mysql> show variables like 'general_log%';
+-----------------+-------------------------------+
| Variable_name       | Value                                    |
+-----------------+-------------------------------+
| general_log           | OFF                                      |
| general_log_file | /mysqldata/3306/data/mysql.log |
+-----------------+-------------------------------+
2     rows in set (0.04 sec)
```

3、临时调整生效普通查询日志

```
mysql> show variables like 'general_log%';
+-----------------+-------------------------------+
| Variable_name       | Value                                    |
+-----------------+-------------------------------+
| general_log           | OFF                                      |
| general_log_file | /mysqldata/3306/data/mysql.log |
+-----------------+-------------------------------+
2 rows in set (0.04 sec)
mysql> set global general_log_file = "/mysqldata/3306/MySQL_oldboy.log";
Query OK, 0 rows affected (0.00 sec)
mysql> show variables like 'general_log%';
+-----------------+---------------------------------+
| Variable_name       | Value                                      |
+-----------------+---------------------------------+
| general_log           | OFF                                        |
| general_log_file | /mysqldata/3306/MySQL_oldboy.log |
+-----------------+---------------------------------+
2 rows in set (0.00 sec)
mysql> set global general_log = "ON";
```

```
Query OK, 0 rows affected (0.00 sec)
mysql> show variables like 'general_log%';
+------------------+---------------------------------+
| Variable_name        | Value                                          |
+------------------+---------------------------------+
| general_log           | ON                                              |
| general_log_file | /mysqldata/3306/MySQL_oldboy.log |
+------------------+---------------------------------+
2 rows in set (0.00 sec)
```

2.3.3.3 慢查询日志介绍与调整

```
1、慢查询日志介绍
慢查询日志：记录执行时间超出指定值的 SQL 语句
2、慢查询日志调整
log_query_time =1
log-slow-queries = /data/3306/slow.log
log_queries_not_using_indexes
3、慢查询日志切割
[root@mysql scripts]# cat /server/scripts/cut_slow.sh
#!/bin/sh
cd /data/3306 && \
/bin/mv slow.log slow.log.$(date +%F) && \
mysqladmin -uroot -ppcwangjixuan -S /data/3306/mysql.sock flush-logs
定时任务
00   0 * * * /bin/sh /server/scripts/cut_slow.sh &>/dev/null
3、使用工具 mysqlsla 分析慢查询，定时发送给相关人员信箱
4、使用 explain 命令优化 SQL 语句（select 语句）的基本流程（上面以提到）
```

2.3.3.4 二进制日志介绍与调整

```
1、二进制日志介绍
二进制日志：记录数据被修改的相关信息，用于主从复制及增量恢复
2、二进制日志调整
mysql> show variables    like 'log_bin%';
+---------------------------------+-------+
| Variable_name                          | Value |
+---------------------------------+-------+
| log_bin                                  | ON       |    记录 binlog 开关
| log_bin_trust_function_creators | OFF      |
+---------------------------------+-------+
2 rows in set (0.00 sec)
```

2.3.4 mysql 的 binlog 有三种模式

2.3.4.1 row level

```
日志中会记录成每一行数据被修改的形式，然后在 slave 端再对相同的数据进行修改
      优点：在 row level 模式下，bin-log 中可以不记录执行的 sql 语句的上下文相关的信息，仅仅只需要记录哪一条记录被修改了，修改成什么样了，所以 row level 的日志内容会非常清楚的记录下每一行数据修改的细节，非常容易理解，而且不会出现某些特定情况下的存储
```

过程或 function，以及 trigger 的调用和触发无法被正常复制的问题

缺点：row level 下，所有的执行的语句当记录到日志中的时候，都将以每行记录的修改来记录，这样可能会产生大量的日志内容，比如有这样一条 update 语句：update product set owner_member_id = ‘b’ where owner_member_id= ’a’,执行之后，日志中记录的不是这条 update 语句对应额事件（mysql 以事件的形式来记录 bin-log 日志），二十这条语句所更新的每一条记录的变化情况，这样就记录成很多条记录被更新的很多个事件，自然，bin-log 日志的量就会很大，尤其是当执行 alter table 之类的语句的时候，产生的日志量是惊人的，因为 mysql 对于 alter table 之类的表结构变更语句的处理方式是整个表的每一条记录都需要变动，实际上就是重建了整个表，那么该表的每一条记录都会被记录到日志中

检验 ROW 模式下 binlog 日志记录效果

```
[root@mysql ~]# mysqlbinlog --base64-output="decode-rows" --verbose mysql-bin.000001
```

2.3.4.2 statement level(默认)

每一条会修改数据的 sql 都会记录到 master 的 bin-log 中，slave 在复制的时候 sql 进程会解析成和原来 master 端执行过的相同的 sql 来再次执行

优点：statement level 下的优点首先就是解决了 row level 下的缺点，不需要记录每一行数据的变化，减少 bin-log 日志量，节约 IO，提高性能，因为它只需要记录在 master 上所执行的语句的细节，以及执行语句时候的上下文的信息

缺点：由于它是记录的执行语句，所以，为了让这些语句在 slave 端也能正确执行，那么它还必须记录每条语句在执行的时候的一些相关信息，也就是上下文信息，以保证所有语句在 slave 端执行的时候能够得到和在 master 端执行时候相同的结果，另外就是，由于 mysql 现在发展比较快，很多的新功能不断的加入，是 mysql 的复制遇到了不小的挑战，自然复制的时候涉及到越复杂的内容，bug 也就越容易出现，在 statement level 下，目前已经发现就有不少情况会造成 mysql 的复制出现问题：主要是修改数据的时候使用了某些特定的函数或者功能的时候会出现

2.3.4.3 Mixed

实际上就是前两中模式的结合，在 mixed 模式下，mysql 会根据执行的每一条具体的 sql 语句来区分对待记录的日志形式，也就是在 statement 和 row 之间选择一种，新版本中 statement level 还是和以前的一样，仅仅记录执行的语句，而新版本的 mysql 中队 row level 模式也被做了优化，并不是所有的修改都会以 row level 来记录，像遇到表结构变更的时候就会以 statement 模式来记录，如果 sql 语句确实就是 update 或者 delete 等修改数据的语句，那么还是会记录所有行的变更

2.3.5 企业场景如何选择 binlog 的模式

1、互联网公司，使用 mysql 的功能相对少（存储过程、触发器、函数），使用默认的语句模式，statementlevel（默认）

2、公司如果用到使用 mysql 的特殊功能（存储过程、触发器、函数），则选择 Mixed 模式

3、公司如果用到使用 mysql 的特殊功能（存储过程、触发器、函数），又希望数据最大化一致，则选择 row 模式

2.3.6 设置 mysql binlog 的格式

查看当前 binlog 的模式

```
mysql> show global variables like 'binlog_format%';
+---------------+-----------+
| Variable_name | Value     |
+---------------+-----------+
| binlog_format | STATEMENT |
```

```
+---------------+-----------+
1 row in set (0.01 sec)
在配置文件修改参数：
[mysqld]
log-bin=mysql-bin
binlog_format=”STATEMENT”
#binlog_format=”ROW”
#binlog_format=”MIXED”
运行时在线修改：调整为 row 模式
mysql> show global variables like 'binlog_format%';
+---------------+-----------+
| Variable_name | Value         |
+---------------+-----------+
| binlog_format | STATEMENT |
+---------------+-----------+
1 row in set (0.01 sec)
mysql> set global binlog_format = ROW;
Query OK, 0 rows affected (0.00 sec)
mysql> show global variables like 'binlog_format%';
+---------------+-------+
| Variable_name | Value |
+---------------+-------+
| binlog_format | ROW     |
+---------------+-------+
1 row in set (0.00 sec)
```

2.3.7 mysql 生产备份实战应用指南

2.3.7.1 全量备份

```
全量备份
全量数据就是数据库中所有的数据，全量备份就是把数据库中所有的数据进行备份
例：
备份所有库
[root@mysql ~]# mysqldump -uroot -ppcwangjixuan -S /mysqldata/3306/mysql.sock    -A -B -R --events|gzip>/tmp/all.backup.$(date +%F).sql.gz
```

2.3.7.2 增量备份

```
增量数据是从上次全量备份之后，更新的新数据，对于 mysql 来说，binlog 日志就是 mysql 的增量数据
例：
按天全备情况
优点：恢复时间：短，维护成本：低
缺点：占用空间：多，占用系统资源多，经常锁表影响用户体验
```

周一 00 点全量备份	周二 00 点全量备份
01. sql.gz 周一增量备份	02. sql.gz 周二增量数据
mysql-bin.000024	mysql-bin.000037

mysql-bin.000025 mysql-bin.000026 ..... mysql-bin.index (binlog 的索引文件)	mysql-bin.000038 mysql-bin.000039 ..... mysql-bin.index (binlog 的索引文件)

按周全备情况

缺点：维护成本高，恢复麻烦，时间长

优点：占用空间：少，占用系统资源少，无需锁表，用户体验好一些

周六 00 点全量备份
03. sql.gz 周六增量备份
mysql-bin.000024 mysql-bin.000025 mysql-bin.000026 ..... mysql-bin.index (binlog 的索引文件)

2.3.7.3 企业场景和增量的频率是怎么做的？

1）中小公司，全量一般是每天一次，业务流量低谷执行全备，备份时会锁表

增量备：

定时推 binlog 增量，例如每分钟推一次增量

rsync –avz /data/3306/mysql-bin000* rsync_backup@10.0.0.18 –password-file=/etc/rsync.password

2）大公司周备，每周六 00 点一次全备，周日-下周六 00 点前都是增量

优点：节省备份时间，减少备份压力，缺点：增量的 binlog 文件副本太多，还原会很麻烦

3）一主多从环境，主从复制本省就是实时远程备份，可以解决服务器物理故障

4）一主多从环境，可采取一个从库服务器上专门用 mysqldump，cp，tar，xtrabackup 做备份，延迟同步

2.3.7.4 mysql 增量恢复必备条件

1、开启 mysql log-bin 日志功能

mysql 数据库开启了 log-bin 参数记录 binlog 日志功能如下：

[root@mysql ~]# grep 'log-bin' /mysqldata/3306/my.cnf

log-bin = /mysqldata/3306/mysql-bin

提示：主库和备份的从库都要开启 binlog 记录功能

小结：增量恢复的条件：

存在一份全备加上全备之后的时刻到出问题时刻的所有增量 binlog 文件备份

2.3.7.5 实战模拟凌晨 00 点对 oldboy 库做个全备，早上 10 点左右删除了 oldboy 库，下面是其恢复过程

备份过程

[root@mysql ~]# date -s '2016/6/17 00:00'

Fri Jun 17 00:00:00 CST 2016

[root@mysql ~]# mysqldump -u root -ppcwangjixuan -S -x /mysqldata/3306/mysql.sock -F -B oldboy>/server/backup/oldboy_$(date +%F).sql

[root@mysql ~]# ls -l /server/backup/oldboy_2016-06-17.sql

-rw-r--r--. 1 root root 3300 Jun 17 00:00 /server/backup/oldboy_2016-06-17.sql

删除过程

[root@mysql ~]# date -s '2016/6/17 10:00'

Fri Jun 17 10:00:00 CST 2016
[root@mysql ~]# mysql -u root -ppcwangjixuan -S /mysqldata/3306/mysql.sock
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 21
Server version: 5.5.27-log Source distribution
Copyright (c) 2000, 2011, Oracle and/or its affiliates. All rights reserved.
Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.
mysql> drop database oldboy ;
Query OK, 3 rows affected (0.06 sec)
mysql> show databases;
+--------------------+
| Database |
+--------------------+
| information_schema |
| mysql |
| oldboy_gbk |
| performance_schema |
| wordpress |
+--------------------+
5 rows in set (0.00 sec)
恢复过程
找到 00 点到 10 点的 bin-log 日志，并且用 mysqlbinlog 命令将二进制文件解析成文本数据，然后在文件中找出删除 drop database oldboy 的 sql 语句，并删除
[root@mysql 3306]# mysqlbinlog /mysqldata/3306/mysql-bin.000028 -d oldboy -r /server/backup/oldboy.sql
[root@mysql backup]# sed -i 's#drop database oldboy##g' /server/backup/oldboy.sql
导入过程
[root@mysql backup]# mysql -u root -ppcwangjixuan -S /mysqldata/3306/mysql.sock </server/backup/oldboy_2016-06-17.sql
[root@mysql backup]# mysql -u root -ppcwangjixuan -S /mysqldata/3306/mysql.sock </server/backup/oldboy.sql
登录数据库查看其恢复情况
[root@mysql backup]# mysql -u root -ppcwangjixuan -S /mysqldata/3306/mysql.sock
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 24
Server version: 5.5.27-log Source distribution
Copyright (c) 2000, 2011, Oracle and/or its affiliates. All rights reserved.
Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

```
mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| oldboy             |
| oldboy_gbk         |
| performance_schema |
| wordpress          |
+--------------------+
6 rows in set (0.00 sec)
mysql> use oldboy;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A
Database changed
mysql> show tables;
+-----------------+
| Tables_in_oldboy |
+-----------------+
| pengchun        |
| test            |
| zhangxuan       |
+-----------------+
3 rows in set (0.00 sec)
```

2.3.7.6 实战模拟凌晨 00 点对 oldboy 库做个全备，早上 10 点左右更新了 oldboy 库的 test 表中所有字段数据，下面是其恢复过程（update 表中的数据的时候，把表中的字段换成了一个相同的内容，这时候要停库）

备份过程

```
[root@mysql 3306]# date -s '2016/6/18 00:00'
Sat Jun 18 00:00:00 CST 2016
[root@mysql 3306]# mysqldump -u root -ppcwangjixuan -x -F -S /mysqldata/3306/mysql.sock -B oldboy >/server/backup/oldboy.$(date +%F).sql
[root@mysql 3306]# mysqlbinlog /mysqldata/3306/mysql-bin.000002 -d oldboy -r /server/backup/oldboy.sql
[root@mysql backup]# ls
oldboy.2016-06-18.sql  oldboy.sql
```

更新过程

```
[root@mysql backup]# date -s '2016/6/18 10:00'
Sat Jun 18 10:00:00 CST 2016
[root@mysql backup]# mysql -u root -ppcwangjixuan -S /mysqldata/3306/mysql.sock
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 17
Server version: 5.5.27-log Source distribution
```

```
Copyright (c) 2000, 2011, Oracle and/or its affiliates. All rights reserved.
Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.
mysql> use oldboy
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A
Database changed
mysql> update test set name='zhangxuan';
Query OK, 4 rows affected (0.00 sec)
Rows matched: 5  Changed: 4  Warnings: 0
mysql> select * from test;
+----+------+-----------+-------------+
| id | age  | name      | shouji      |
+----+------+-----------+-------------+
|  1 | NULL | zhangxuan | NULL        |
|  2 | NULL | zhangxuan | NULL        |
|  3 | NULL | zhangxuan | NULL        |
|  4 |   24 | zhangxuan | 13511111111 |
|  5 |   22 | zhangxuan | 13655555555 |
+----+------+-----------+-------------+
5 rows in set (0.00 sec)
```

恢复过程

```
[root@mysql backup]# mysqlbinlog /mysqldata/3306/mysql-bin.000002 -r /server/backup/oldboy.sql
```

找到/server/backup/oldboy.sql 文件,把 update test set name='zhangxuan'去掉

导入过程

```
[root@mysql backup]# mysql -u root -ppcwangjixuan -S /mysqldata/3306/mysql.sock oldboy </server/backup/oldboy.sql
```

查看恢复情况

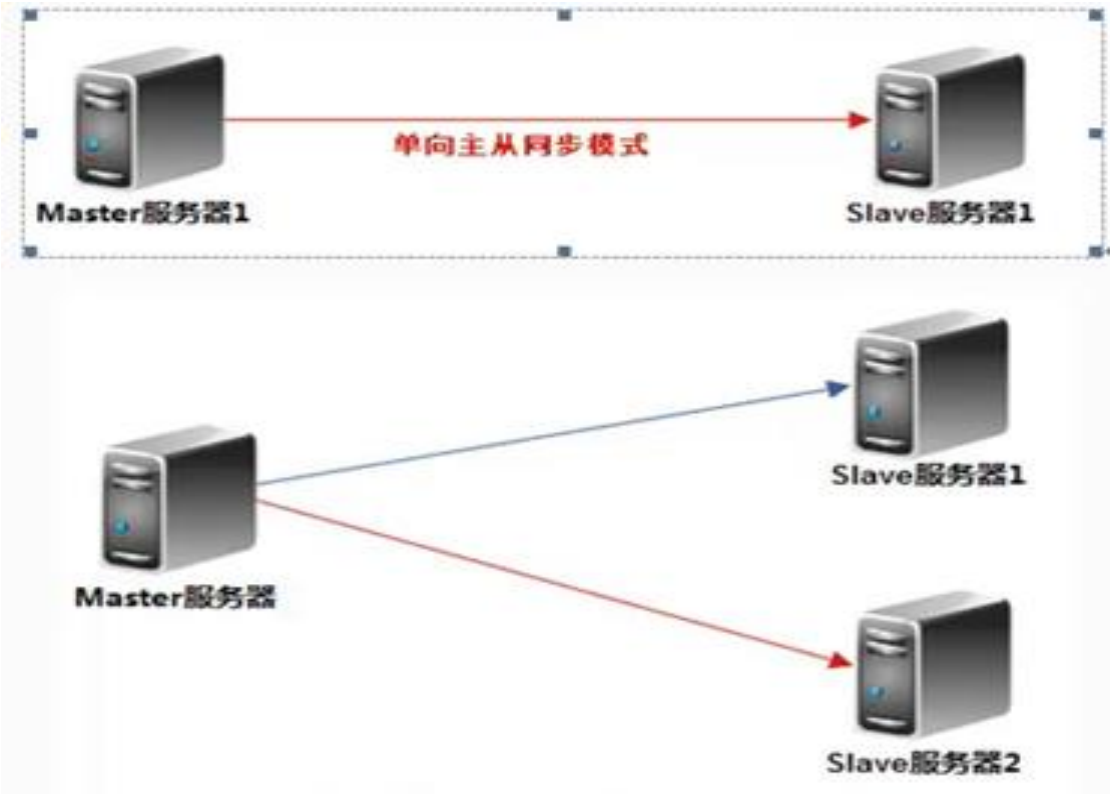
```
[root@mysql 3306]# mysql -u root -ppcwangjixuan -S /mysqldata/3306/mysql.sock
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 36
Server version: 5.5.27-log Source distribution
Copyright (c) 2000, 2011, Oracle and/or its affiliates. All rights reserved.
Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.
mysql> use oldboy
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A
Database changed
```

```
mysql> select * from test;
+----+------+-----------+-------------+
| id | age  | name      | shouji      |
+----+------+-----------+-------------+
|  1 | NULL | oldgirl   | NULL        |
|  2 | NULL | 老男孩    | NULL        |
|  3 | NULL | etiantian | NULL        |
|  4 |   24 | zhangxuan | 13511111111 |
|  5 |   22 | huangyan  | 13655555555 |
+----+------+-----------+-------------+
5 rows in set (0.00 sec)
```

2.3.8 mysql 的主从复制的结构图

mysql 的主从复制，和文件及文件系统级别同步是类似的，都是数据的传输，只不过 mysql 无需借助第三方工具，而是其自带的同步复制功能，另外一点，mysql 的主从复制并不是磁盘文件直接同步，而是逻辑的 binlog 日志同步到本地在应用执行的过程

2.3.8.1 单向的主从复制图，此架构只能在 master 端进行数据写入（生产环境可以使用）



2.3.8.2 双向的主主同步逻辑图，此架构可以在 master1 端或 master2 端进行数据写入（生产环境不建议使用）

2.3.8.3 线性级联单向双主同步逻辑图，此架构只能在 master1 端进行数据写入（生产环境可以使用）

2.3.8.4 环状级联单向多主同步逻辑图，任意一个都可以写入数据（生产环境不建议使用）

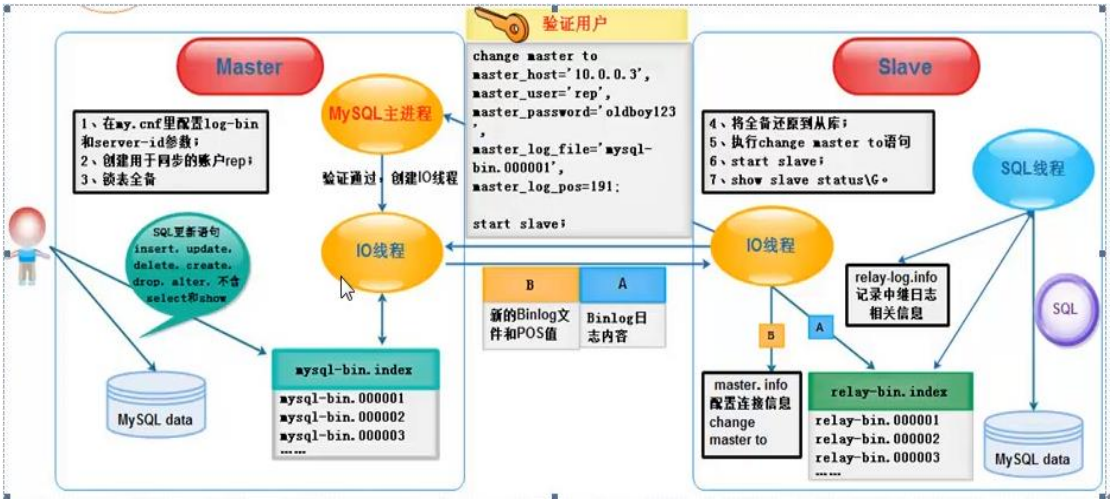
2.3.8.5 环状级联单向多主多从同步逻辑图，此架构只能在任意一个 master 端进行数据写入（生产环境不建议使用）

2.3.9 mysql 主从复制的原理

mysql 的主从复制是一个异步的复制过程（虽然一般情况下感觉是实时的），数据将从一个 mysql 数据库（我们称之为 master）复制到另一个 mysql 数据库（我们称之为 slave），

在 master 与 slave 之间实现整个主从复制的过程是由三个线程参与完成的，其中有两个线程（SQL 线程和 IO 线程）在 slave 端，另外一个线程（I/O 线程）在 master 端，要实现 mysql 的主从复制，首先必须打开 master 端的 binlog 记录功能，否则就无法实现，因为整个复制过程实际上就是 slave 从 master 端获取 binlog 日志，然后在 slave 上以相同顺序执行获取的 binlog 日志中所记录的各种 SQL 操作

2.4.0 mysql 主从复制的原理图



2.4.1 mysql 主从复制的实践

2.4.1.1 环境准备

mysql 多实例，其中向外提供服务的端口分别为 3306，3307，其中 3306 为主库，开启了 binlog 功能，3307 为从库，并且两个数据库中的 server-id 是不同

```
[root@mysql ~]# netstat -lntup|grep 330
tcp        0      0 0.0.0.0:3306                0.0.0.0:*                   LISTEN
16986/mysqld
tcp        0      0 0.0.0.0:3307                0.0.0.0:*                   LISTEN
18616/mysqld
```

2.4.1.2 分别查看 3306 和 3307 不同数据库有哪些库

3306 数据库上

```
[root@mysql ~]# mysql -u root -ppcwangjixuan -S /mysqldata/3306/mysql.sock
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 37
Server version: 5.5.27-log Source distribution
Copyright (c) 2000, 2011, Oracle and/or its affiliates. All rights reserved.
Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.
mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
```

```
| oldboy             |
| oldboy_gbk         |
| performance_schema |
| wordpress          |
+--------------------+
6 rows in set (0.05 sec)
```

3307 数据库上

```
[root@mysql ~]# mysql -uroot -ppcwangjixuan -S /mysqldata/3307/mysql.sock
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 2
Server version: 5.5.27 Source distribution
Copyright (c) 2000, 2011, Oracle and/or its affiliates. All rights reserved.
Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.
mysql> show databases ;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
+--------------------+
3 rows in set (0.00 sec)
```

2.4.1.3 全量备份 3306 数据库的库，然后到 3307 数据库中

全量备份 3306 的数据库

```
[root@mysql ~]# mysqldump -uroot -ppcwangjixuan -A -B -R --events -x --master-data=2 -S /mysqldata/3306/mysql.sock|gzip>/tmp/all.$(date +%F).sql.gz
```

将全量备份的数据导入到 3307 数据库中，并查看数据库中的变化

```
[root@mysql ~]# gzip -d /tmp/all.2016-06-18.sql.gz
[root@mysql ~]# mysql -u root -ppcwangjixuan -S /mysqldata/3307/mysql.sock </tmp/all.2016-06-18.sql
[root@mysql ~]# mysql -uroot -ppcwangjixuan -S /mysqldata/3307/mysql.sock
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 4
Server version: 5.5.27 Source distribution
Copyright (c) 2000, 2011, Oracle and/or its affiliates. All rights reserved.
Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.
mysql> show databases;
+--------------------+
```

```
| Database           |
+--------------------+
| information_schema |
| mysql              |
| oldboy             |
| oldboy_gbk         |
| performance_schema |
| wordpress          |
+--------------------+
6 rows in set (0.00 sec)
```

2.4.1.4 在 3306 数据库上授权用户可以到 3306 数据库上复制 binlog

```
[root@mysql ~]# mysql -uroot -ppcwangjixuan -S /mysqldata/3306/mysql.sock
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 42
Server version: 5.5.27-log Source distribution
Copyright (c) 2000, 2011, Oracle and/or its affiliates. All rights reserved.
Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.
mysql> grant replication slave on *.* to slave@'172.16.1.%' identified by 'slave';
Query OK, 0 rows affected (0.09 sec)
```

2.4.1.5 在 3307 数据库上开启复制 3306 的 binlog 开关，并查看是否处于同步状态

```
[root@mysql ~]# mysql -uroot -ppcwangjixuan -S /mysqldata/3307/mysql.sock
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 5
Server version: 5.5.27 Source distribution
Copyright (c) 2000, 2011, Oracle and/or its affiliates. All rights reserved.
Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.
mysql> CHANGE MASTER TO
    -> MASTER_HOST='172.16.1.171 ',
    -> MASTER_PORT=3306,
    -> MASTER_USER='slave',
    -> MASTER_PASSWORD='slave',
    -> MASTER_LOG_FILE='mysql-bin.000002',
    -> MASTER_LOG_POS=4710;
```

注意：其中 MASTER_LOG_FILE='mysql-bin.000002', MASTER_LOG_POS=4710;这两个 binlog 位置是从/tmp/all.2016-06-18.sql（全备 3306 数据库文件）中找到的，因为备份时指定了--master-data=2 会把备份时的 binlog 位置点记录下来

```
Query OK, 0 rows affected (0.09 sec)
mysql> start slave;
```

```
Query OK, 0 rows affected (0.07 sec)
mysql> show slave status\G;  从下面查看 Slave_IO_Running: Yes，Slave_SQL_Running: Yes，Seconds_Behind_Master: 0 得知，mysql 处于主从同步状态
*************************** 1. row ***************************
               Slave_IO_State: Waiting for master to send event
                  Master_Host: 172.16.1.171
                  Master_User: slave
                  Master_Port: 3306
                Connect_Retry: 60
              Master_Log_File: mysql-bin.000002
          Read_Master_Log_Pos: 520241
               Relay_Log_File: relay-bin.000004
                Relay_Log_Pos: 520387
        Relay_Master_Log_File: mysql-bin.000002
             Slave_IO_Running: Yes
            Slave_SQL_Running: Yes
              Replicate_Do_DB:
          Replicate_Ignore_DB: mysql
           Replicate_Do_Table:
       Replicate_Ignore_Table:
      Replicate_Wild_Do_Table:
  Replicate_Wild_Ignore_Table:
                   Last_Errno: 0
                   Last_Error:
                 Skip_Counter: 0
          Exec_Master_Log_Pos: 520241
              Relay_Log_Space: 520580
              Until_Condition: None
               Until_Log_File:
                Until_Log_Pos: 0
           Master_SSL_Allowed: No
           Master_SSL_CA_File:
           Master_SSL_CA_Path:
              Master_SSL_Cert:
            Master_SSL_Cipher:
               Master_SSL_Key:
        Seconds_Behind_Master: 0
Master_SSL_Verify_Server_Cert: No
                Last_IO_Errno: 0
                Last_IO_Error:
               Last_SQL_Errno: 0
               Last_SQL_Error:
  Replicate_Ignore_Server_Ids:
             Master_Server_Id: 1
```

```
1 row in set (0.00 sec)
ERROR:
No query specified
```

2.4.1.6 在 3306 上创建数据库 zhangxuan，看 3307 上是否同步过来

在 3306 数据库上创建 zhanxguan 库

```
[root@mysql ~]# mysql -uroot -ppcwangjixuan -S /mysqldata/3306/mysql.sock
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 46
Server version: 5.5.27-log Source distribution
Copyright (c) 2000, 2011, Oracle and/or its affiliates. All rights reserved.
Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.
mysql> create database zhangxuan;
Query OK, 1 row affected (0.01 sec)
mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| oldboy             |
| oldboy_gbk         |
| performance_schema |
| wordpress          |
| zhangxuan          |
+--------------------+
8 rows in set (0.00 sec)
```

在 3307 数据库上查看 zhangxuan 库是否同步过来（从下面结果看出，zhangxuan 库时同步过来的）

```
[root@mysql ~]# mysql -uroot -ppcwangjixuan -S /mysqldata/3307/mysql.sock
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 11
Server version: 5.5.27 Source distribution
Copyright (c) 2000, 2011, Oracle and/or its affiliates. All rights reserved.
Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.
mysql> show databases;
+--------------------+
| Database           |
+--------------------+
```

```
| information_schema |
| mysql              |
| oldboy             |
| oldboy_gbk         |
| performance_schema |
| wordpress          |
| zhangxuan          |
+--------------------+
7 rows in set (0.00 sec)
```

2.4.2 关于主从复制出现故障怎么解决

下面以这个故障为例：

```
    show slave status;报错:且 show slave status\G:

    Slave_IO_Running: Yes

    Slave_SQL_Running: NO

    Seconds_Behind_Master: NULL

                    Last_Error: Error 'Can't create database 'xiaoliu'; database exists' on query.
Default database: 'xiaoliu'. Query: 'create database xiaoliu'
```

解决办法 1：

登录从库上操作：

1、stop slave； #<==临时停止同步开关

2、set global sql_slave_skip_counter =1; #<==将同步指针向下移动一个，也可以多个，如果多次不同步，可以重复操作；

3、start slave；#<==重启主从复制开关

解决办法 2：

在配置文件中加入参数（/etc/my.cnf）

slave-skip-errors = 1032,1062,1007

其他可能引起复制故障的问题：

1、mysql 自身的原因及人为重复插入数据

2、不同的数据库版本会引起不同步，低版本到高版本可以，但是高版本不能往低版本同步

3、mysql 的运行错误或者程序 BUG

4、binlog 记录模式，例如：row level 模式就比默认的语句模式更好

2.4.3 主从复制延迟问题原因及解决方案

2.4.3.1 一个主库的从库太多，导致复制延迟

建议从库数量 3-5 个为宜，要复制的从节点数量过多，会导致复制延迟

2.4.3.2 从库硬件比主库差，导致复制延迟

查看 master 和 slave 的系统配置，可能会因为机器配置的问题，包括磁盘 IO、CPU、内存等各方面因素造成复制的延迟，一般发生在高并发大数据量写入场景

2.4.3.3 慢 SQL 语句过多

假如一条 SQL 语句，执行时间是 20 秒，那么执行完毕，到从库上能查到数据也至少是 20 秒，这样就延迟 20 秒了

SQL 语句的优化一般要作为常规工作不断的监控和优化，如果是单个 SQL 的写入时间长，可以修改后分 多次写入，通过查看慢查询日志或 show full processlist 命令找出执行

时间长的查询语句或者大的事务

2.4.3.4 主从复制的设计问题

例如，主从复制单线程，如果主库写并发太大，来不及传送到从库就会导致延迟。更高版本的 mysql 可以支持多线程复制，门户网站则会自己开发多线程同步功能

2.4.3.5 主从库之间的网络延迟

主从库的网卡，网线，连接的交换机等网络设备都可能成为复制的瓶颈，导致复制延迟，另外，跨公网主从复制很容易导致主从复制延迟

2.4.3.6 主库读写压力大，导致复制延迟

主库硬件要搞好一点，架构的前端要加 buffer 以及缓存层

2.4.4 通过 read-only 参数让从库只读访问

read-only 参数选项可以让从服务器只允许来自从服务器线程或具有 super 权限的数据库用户进行更新，可以确保从服务器不接受来自用户端的非法用户更新

read-only 参数允许数据更新的条件为：

1、具有 super 权限的用户可以更新，不受 read-only 参数影响，例如：管理员 root

2、来自从服务器线程可以更新，不受 read-only 参数影响，例如：前文的 rep 用户

在生产环境中，可以在从库 slave 中使用 read-only 参数，确保从库数据不被非法更新

read-only 参数配置方法如下：

方法一：启动数据库时直接带--read-only 参数启动或重启，使用

```
mysqladmin –uroot –ppcwangjixuan –S /data/3307/mysql.sock shutdown
mysqld_safe –defaults-file=/data/3307/my.cnf --read-only &
```

方法二：在 my.cnf 里[mysqld]模块下加 read-only 参数，然后重启数据库，配置如下

```
[mysqld]
read-only
```

2.4.5 web 用户专业设置方案：mysql 主从复制读写分离集群

专业的运维人员提供给开发人员的读写分离的账号设置方法如下：

1） 访问主库和从库时使用一套用户密码，例如,用户为 web，密码为 oldboy123

2） 即使访问 IP 不同，端口也尽量相同（3306），例如：写库 VIP 为 10.0.0.7，读库 VIP 为 10.0.0.8

除了 IP 没办法修改之外，要尽量为开发人员提供方便，如果是数据库前端有 DAL 层（DBPROXY）,还可以只给开发人员一套用户、密码、IP、端口，这样就更专业了，剩下的都由运维人员搞定。

下面是授权 web 连接用户访问的方案：mysql 主从复制读写分离集群。

方法 1：主库和从库使用不同的用户，授权不同的权限

主库上对 web_w 用户授权如下：

用户：web_w 密码：oldboy123 端口：3306 主库 VIP：10.0.0.7

权限：select，insert，update，delete

命令：grant select,insert,update,delete on web.* to web_w@’172.16.1.%’ identified by ‘oldboy123’

从库上对 web_r 用户授权如下：

用户：web_r 密码：oldboy123 端口：3306 从库 VIP：10.0.0.8

权限：select

命令：grant select on web.* to web_r@’172.16.1.%’ identified by ‘oldboy123’

提示：此法显得不够专业，但是可以满足开发需求

方法 2：主库和从库使用相同的用户，但授予不同的权限

主库上对 web 用户授权如下：

主库上对 web 用户授权如下：
用户：web 密码：oldboy123 端口：3306 主库 VIP：10.0.0.7
权限：select，insert，update，delete
命令：grant select,insert,update,delete on web.* to web@'172.16.1.%' identified by 'oldboy123'
从库上对 web 用户授权如下：
用户：web 密码：oldboy123 端口：3306 从库 VIP：10.0.0.8
权限：select
命令：grant select on web.* to web@'172.16.1.%' identified by 'oldboy123'
提示：由于主库和从库都是同步复制的，所以从库上的 web 用户会自动和主库的一致，即无法实现只读 select 的授权，要想实现方法 2 中的授权方案，有两个方法：
一是在主库上创建完用户和权限，从库上 revoke 收回对应更新权限（insert，update，delete）
二是忽略授权表 mysql 同步，主库的配置参数如下：
[mysqld]
binlog-ignore-db = mysql
replicate-ignore-db = mysql
提示：注意上面参数等号两边必须要有空格
方法三：在从库上设置 read-only 参数，让从库只读
主库从库：主库和从库使用相同的用户，授予相同的权限（非 ALL 权限）
用户：web 密码：oldboy123 端口：3306 主库 VIP：10.0.0.7 从库 VIP：10.0.0.8
权限：select，insert，update，delete
命令：grant select,insert,update,delete on web.* to web@'172.16.1.%' identified by 'oldboy123'
由于从库设置了 read-only，非 super 权限是无法写入的，因此，通过 read-only 参数就可以很好的控制用户非法将数据写入从库

2.4.6 让 mysql 从库记录 binlog 日志方法

从库需要记录 binlog 的应用场景为：当前的从库还要作为其他从库的主库，例如：级联复制或双主互为主从的场景的情况下，下面介绍一下从库记录 binlog 日志的方法

在从库的 my.cnf 中加入如下参数，然后重启服务生效即可

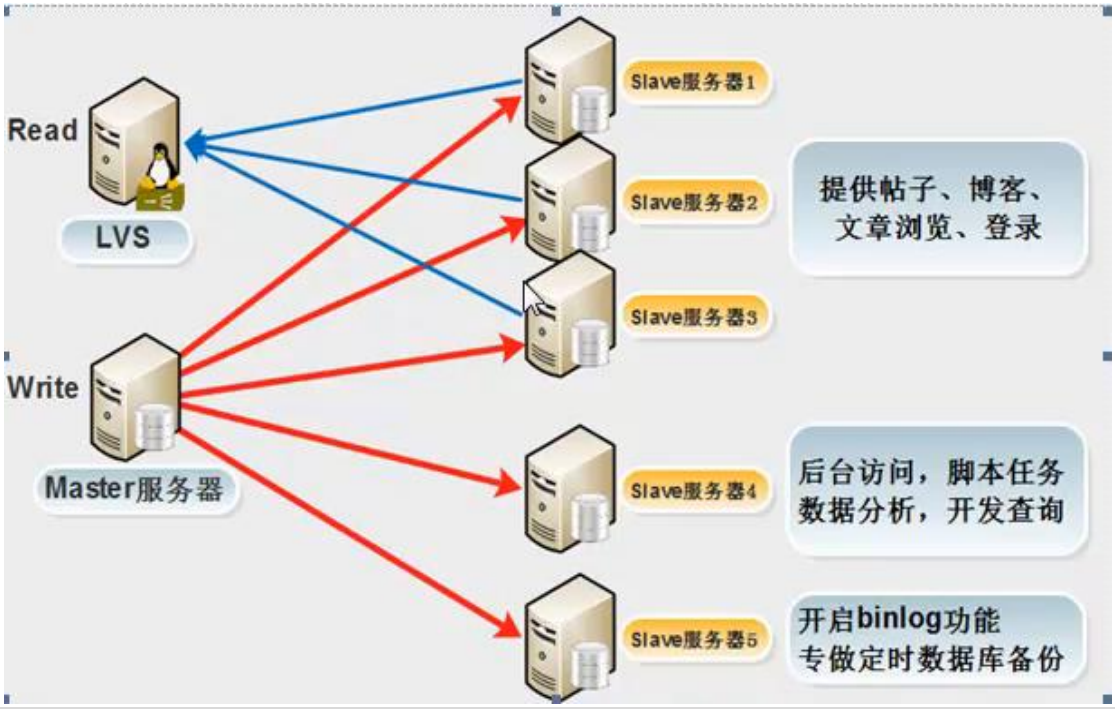
log-slave-updates #<== 必须要有这个参数

log-bin = /data/3307/mysql-bin

expire_logs_days = 7 #<==相当于 find /data/3307/ -type f –name "mysql-bin.000*" –mtime +7|xargs rm –f

2.4.7 mysql 主从复制集群架构的数据备份策略

有主从复制了，还需要做定时全量加增量备份么？
因为，如果主库有语句级误操作（例如：drop database oldboy），从库也会执行 drop database oldboy；这样 mysql 主从库就都删除了该数据
把从库做为数据库备份服务器时，备份策略如下:
高并发业务场景备份时，可以选择在一台从库上备份（slave5），把从库作为数据库备份服务器时需要在从库开启 binlog 功能，其逻辑图如下所示



步骤如下：
1）选择一个不对外提供服务的从库，这样可以确保和主库更新最接近，专门做数据备份
2）开启从库的 binlog 功能
备份时可以选择只停止 SQL 线程，停止应用 SQL 语句到数据库，IO 线程保留工作状态，执行命令为 stop slave sql_thread;备份方式可以采取 mysqldump 逻辑备份或者直接物理备份，例如：物理备份使用 cp、tar（针对/data 目录）或 xtrabackup（第三方的物理备份软件）进行备份，逻辑备份根据总的备份数据量的多少进行选择，数据量低于 20G，建议选择 mysqldump 逻辑备份方案，安全稳定，最后把全备和 binlog 数据发送到备份服务器留存

2.4.8 mysql 一主多从，主库宕机，从库怎么接管

2.4.8.1 半同步从库（谷歌半同步插件 5.5 版本自带）

----S1 作为太子
第一：主库插入数据后，同时写入到 S1，成功返回
优点：两台库同时写入数据
缺点：写入会慢，网络不稳定，主库持续等待
解决措施：
1、 连不上 S1 的时候自动转为异步
2、 设置 10 秒超时，超过 10 秒转为异步
3、 S1 网络，硬件要好，不提供服务，干等接管

2.4.8.2 S1，啥也不干只做同步的从库，500 台服务器，百度

2.4.8.3 皇帝驾崩现选（耽误事，容易被篡位）

主库宕机有两种情况：
1、 如果主库可以 SSH 连接，binglog 数据没丢，要把主库的 binlog 补全到所有从库
第一步：调整 S1 为 M1 的操作
调整配置文件 read-only，授权用户 select，变成增删改查，开启 binlog
rm –f master.info relay-log*
登录数据库 reset master
重启数据库

第二步：其余所有从库的操作：

CHANGE MASTER TO '172.16.1.51',MASTER_LOG_FILE='mysql-bin.000004',MASTER_LOG_POS=107;

2. 如果主库连不上（事先没有指定从库为主库）

选主的方法确保 IO 和 SQL 都读取了自己的 LOG，并应用到了数据库

第一步: 登录所有从库 show processlist，看两个线程的更新状态

```
mysql> show processlist\G;
*************************** 1. row ***************************
     Id: 9
   User: system user
   Host:
     db: NULL
Command: Connect
   Time: 49931
  State: Waiting for master to send event
   Info: NULL
*************************** 2. row ***************************
     Id: 10
   User: system user
   Host:
     db: NULL
Command: Connect
   Time: 49727
  State: Slave has read all relay log; waiting for the slave I/O thread to update it
   Info: NULL
*************************** 3. row ***************************
     Id: 12
   User: root
   Host: localhost
     db: NULL
Command: Query
   Time: 0
  State: NULL
   Info: show processlist
3 rows in set (0.00 sec)
ERROR:
No query specified
```

查看是否更新完毕，

第二步：分别登录从库查看其 master.info 文件

cat /data/3306/data/master.info

确保更新完毕后，查看 master.info 确定哪个从库 POS 最快，经过查看没有延迟的情况 POS 差距最少，提升为主，然后提升为主的从库（S1）补全所有中继日志至每个从库

第三步：切换角色

假如 S1 延迟的差距最少，提升 S1 为主库

调整 S1 为 M1 的操作

调整配置文件 read-only，授权用户 select，变成增删改查，开启 binlog

rm –f master.info relay-log*

登录数据库 reset master

重启数据库

其余所有从库的操作：

CHANGE MASTER TO '172.16.1.51',MASTER_LOG_FILE='mysql-bin.000004',MASTER_LOG_POS=107;

2.4.9 事务介绍

数据库事务介绍

简单的说，事务就是指逻辑上的一组 SQL 语句操作，组成这组操作的各个 SQL 语句，执行时要么全成功要么全失败

例如：oldboy 给 oldgirl 转账 5 块钱，流程如下

a、 从 oldboy 银行卡取出 5 块，计算式 money-5

b、 把上面 5 块钱打入 oldgirl 的账户上，oldgirl 收到 5 块，money+5

上述转账的过程，对应的 sql 语句为

update oldboy_accout set money=money-5 where name='oldboy';

update oldboy_accout set money=money+5 where name='oldgirl';

上述两条 SQL 操作，在事务中的操作就是要么都执行，要么都不执行，这就是事务的原子性，mysql5.5 支持事务的引擎：innodb/ndb

2.4.9.1 事务的四大特性（ACID）

1、原子性（atomicity）

事务是一个不可分割的单位，事务中的所有 SQL 等操作要么都发生，要么都不发生

2、 一致性（consistency）

事务发生前和发生后，数据的完整性必须保持一致

3、 隔离性（isolation）

当并发访问数据库时，一个正在执行的事务在执行完毕前，对于其他的会话是不可见的，多个并发事务之间的数据是相互隔离的

4、 持久性（durability）

一个事务一旦被提交，它对数据库中的数据改变就是永久性的，如果出了错误，事务也不允许撤销，只能通过“补偿性事务”

2.4.9.2 事务的开启

数据库默认事务是自动提交的，也就是发一条 sql 它就执行一条，如果想多条 sql 放在一个事务中执行，则需要使用事务进行处理，当我们开启一个事务，并且没有提交，mysql 会自动回滚事务，或者我们使用 rollback 命令手动回滚事务

数据库开启事务命令：

mysql 默认是自动提交的，也就是你提交一个 query，它就直接执行，我们可以通过

```
mysql> show variables like '%autocommit%';
+---------------+-------+
| Variable_name | Value |
+---------------+-------+
| autocommit    | ON    |
+---------------+-------+
1 row in set (0.00 sec)
```

要想关闭自动提交
在 my.cnf 配置文件里面加入
[mysqld]
autocommit = OFF
然后重启数据库

2.4.9.3 事物的实现

像其他数据库一样，MySQL在做事务的时候使用的日志先行的方式保证事务的快速和持久。
当开始一个事务时，会记录该事务的的一个LSN日志序列号；当执行事务时，会往innodb_log_buffer日志缓冲区里插入事务日志；当事务
个动作是由innodb_flush_log_at_trx_commit这个参数控制的。
innodb_flush_log_at_trx_commit=0，每个事务提交时候，每隔一秒，把事务日志缓存区的数据写到日志文件中，以及把日志文件的数
innodb_flush_log_at_trx_commit=1，每个事务提交时候，把事务日志从缓存区写到日志文件中，并且刷新日志文件的数据到磁盘上；
innodb_flush_log_at_trx_commit=2，每事务提交的时候，把事务日志数据从缓存区写到日志文件中；每隔一秒，刷新一次日志文件，
度；
innodb_flush_log_at_trx_commit=0，性能是最好的，同样安全性也是最差的。当系统宕机时，会丢失1秒钟的数据。

2.5.0 mysql 引擎概述

2.5.0.1 什么是存储引擎？

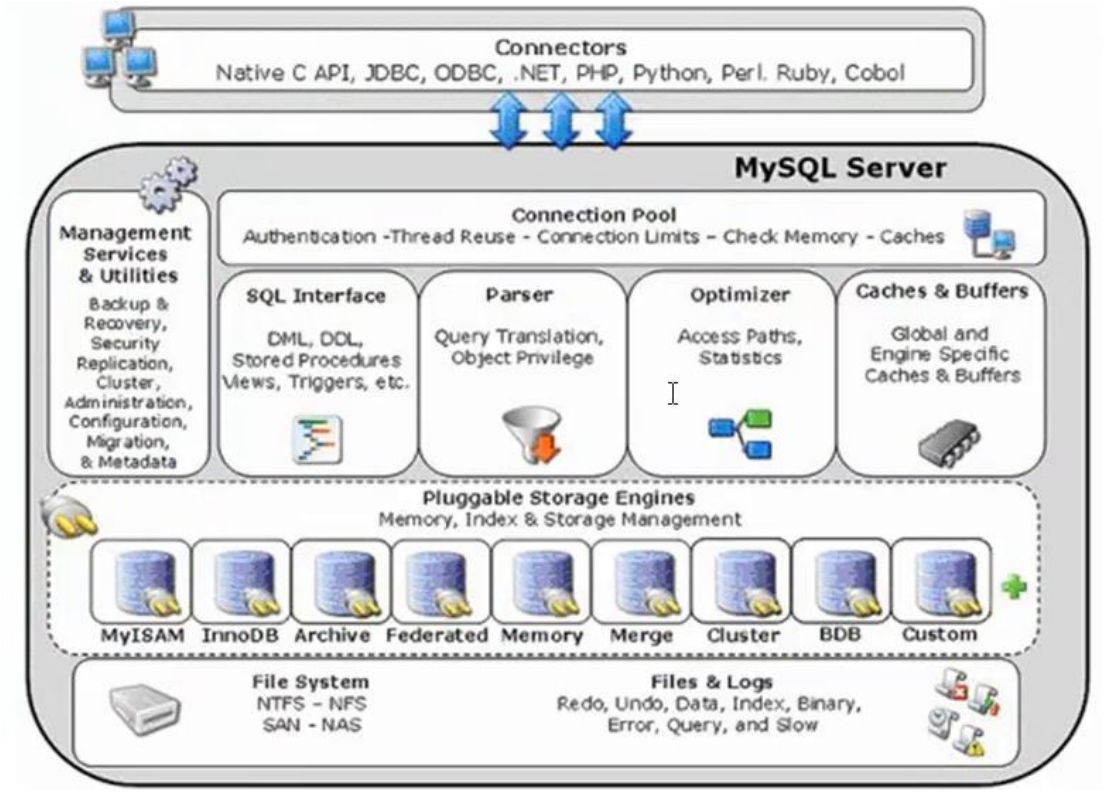
在讲清楚什么是存储引擎之前，我们先来个比喻，我们都知道录制一个视频文件，可以转换成不同的格式如 mp4，avi，wmv 等，而存在我们电脑磁盘上也会存在于不同类型的文件系统中如 windows 里常见的 ntfs，fat32，存在于 linux 里最常见的 ext3，ext4，xfs，但是，给我们或者用户看到实际内容都是一样的，直观区别是，占用系统的空间大小与清晰程度可能不一样

那么，数据库里表的数据存储在数据库里及磁盘上和上述的视频格式及存储磁盘文件系统格式特征类似，也有很多存储方式

但是，对于用户和应用程序来说同样一张表的数据，无论用什么引擎来存储，用户看到的数据都是一样的，不用的引擎存储，引擎功能，占用的空间大小，读取性能等可能有区别

mysql 最常见存储引擎为:MyISAW 和 InnoDB，目前 5.5 版本 myisaw 和 innodb 都已经支持

2.5.0.2 mysql 存储引擎的架构



2.5.0.3 myisaw 引擎介绍

myisam 引擎是 mysql 关系数据库管理系统的默认存储引擎（mysql5.5.5 以前），这种 mysql 表存储结构从旧的 ISAM 代码扩展出许多有用的功能，在新版本的 mysql 中，innodb 引擎由于其对事务参照完整性，以及更高的并发性等优点开始逐步的取代 myisaw 引擎

每一个 myisaw 的表都对应于硬盘上的三个文件，这三个文件有一样的文件名，但是有不同的扩展名知识其类型用途: .frm 文件保存表的定义，这个文件并不是 myisaw 引擎的一部分，而是服务器的一部分，.MYD 保存表的数据，.MYI 是表的索引文件

范例：

```
[root@mysql ~]# ls -l /mysqldata/3306/data/mysql
-rw-rw----. 1 mysql mysql     8820 Jun 18 15:02 columns_priv.frm #<==表的定义
-rw-rw----. 1 mysql mysql        0 Jun 18 15:02 columns_priv.MYD #<==data
-rw-rw----. 1 mysql mysql     4096 Jun 18 15:02 columns_priv.MYI #<==index
```

2.5.0.4 myisaw 引擎特点

1、不支持事务（事务是指逻辑上的一组 SQL 操作，组成这组操作的各个单元，要么全成功要么全失败）

2、表级锁定，数据更新时锁定了整个表，其锁定机制是表级锁定，这虽然可以让锁定的实现成本很小但是也同时大大降低了其并发性能

3、读写互相阻塞，不仅会在写入的时候阻塞读取，myisaw 还会在读取的时候阻塞写入，但读本身并不会阻塞另外的读

4、只会缓存索引，myisaw 可以通过 key_buffer_size 缓存索引，以大大提高访问性能减少磁盘 IO，但是这个缓存区只会缓存索引，而不会缓存数据

5、读取速度较快，占用资源相对少

6、不支持外键约束，但支持全文索引

7、myisaw 引擎是 mysql5.5.5 前缺省的存储引擎

2.5.1 myisaw 引擎调优精要

1、设置合适的索引（缓存机制）
2、调整读写优先级，根据实际需求确保重要操作更优先执行
3、启用延迟插入改善大批量写入性能（降低写入频繁，尽可能多条数据一次性写入）
4、尽量顺序操作让 insert 数据都写入到尾部，减少阻塞时间
5、分解大的时间长的 SQL 操作，降低单个操作的阻塞时间
6、降低并发数（减少对 mysql 访问），某些高并发场景通过应用进行排队队列机制 Q 队列
7、对于相对静态（更改不频繁）的数据库数据，充分利用 query cache 或 memcached/redis 缓存服务可以极大的提高访问效率，网站动态内容静态化，减少对数据库的访问
8、MyISAW 的 count 只有在全表扫描的时候特别高效，带有其他条件的 count 都需要进行实际的数据访问
9、可以把主从同步的主库使用 innodb，从库使用 MyISAW 引擎（不推荐）

2.5.2 innodb 引擎

2.5.2.1 什么是 innodb 引擎？

innodb 引擎是 mysql 数据库的另一个重要的存储引擎，正成为目前 mysqlAB 所发行新版的标准，被包含在所有二进制安装包里，和其它的存储引擎相比，innodb 引擎的优点是支持兼容 ACID 的事务，以及参数完整性（即对外键的支持）

mysql5.5.5 以后数据库的默认存储引擎是 innodb

```
mysql> select    version();
+------------+
| version()    |
+------------+
| 5.5.5-log |
+------------+
1 row in set (0.07 sec)
mysql>
mysql> show create database oldboy;
+----------+-----------------------------------------------------------------+
| Database | Create Database                                                 |
+----------+-----------------------------------------------------------------+
| oldboy     | CREATE DATABASE `oldboy` /*!40100 DEFAULT CHARACTER SET utf8 */ |
+----------+-----------------------------------------------------------------+
1 row in set (0.25 sec)
mysql> use oldboy
Database changed
mysql> show create table test;
+-------+--------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------
----------------------------------------------+
|                    Table                    |                    Create                    Table
|
+-------+--------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------
----------------------------------------------+
```

```
| test    | CREATE TABLE `test` (
   `id` int(4) NOT NULL DEFAULT '0',
   `age` tinyint(2) DEFAULT NULL,
   `name` varchar(16) DEFAULT NULL,
   `shouji` char(11) DEFAULT NULL,
   PRIMARY KEY (`id`),
   KEY `index_name_and_shouji` (`name`(6),`shouji`(8))
) ENGINE=InnoDB DEFAULT CHARSET=utf8 |
+-------+------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------
----------------------------------------------+
1 row in set (0.09 sec)
关于 innodb 引擎的表的数据结构
[root@mysql ~]# ls -l /mysqldata/3306/data/oldboy*
/mysqldata/3306/data/oldboy:
total 24
-rw-rw----. 1 mysql mysql     61 Jun 17 00:13 db.opt
-rw-rw----. 1 mysql mysql 8648 Jun 18 15:02 test.frm
-rw-rw----. 1 mysql mysql   124 Jun 18 15:02 test1. frm
-rw-rw----. 1 mysql mysql 3072 Jun 18 15:02 student. frm
提示：只有 test1.frm 没有 MyISAW 对应的数据文件和索引文件了，这是为什么了？
-rw-rw----. 1 mysql mysql 134217728 Jun 19 08:23 ibdata1 这里是存放 Innodb 数据的文件
-rw-rw----. 1 mysql mysql     4194304 Jun 19 08:23 ib_logfile0
-rw-rw----. 1 mysql mysql     4194304 Jun 12 15:31 ib_logfile1
-rw-rw----. 1 mysql mysql     4194304 Jun 12 15:31 ib_logfile2
-rw-rw----. 1 mysql mysql            57 Jun 19 08:23 master.info
drwx------. 2 mysql mysql        4096 Jun 18 15:02 mysql
-rw-rw----. 1 mysql mysql          975 Jun 17 00:00 mysql-slow.log
drwx------. 2 mysql mysql        4096 Jun 18 15:02 oldboy
drwx------. 2 mysql mysql        4096 Jun 15 21:16 oldboy_gbk
drwx------. 2 mysql mysql        4096 Jun 12 15:23 performance_schema
drwx------. 2 mysql mysql        4096 Jun 14 05:11 wordpress
drwx------. 2 mysql mysql        4096 Jun 18 15:14 zhangxuan
```

2.5.2.2 innodb 引擎特点

1、支持事务：支持 4 个事务隔离级别，支持多版本读

2、行级锁定（更新时一般是锁定当前行）：通过索引实现，全表扫描仍然会表锁，注意间隙锁的影响

3、读写阻塞与事务隔离级别相关

4、具有非常高效的缓存特性：能缓存索引，也能缓存数据

5、整个表和主键以 Cluster 方式存储，组成一颗平衡树

6、所欲 secondary index 都会保存主键信息

7、支持分区、表空间

8、支持外键约束，5.5 以前不支持全文索引，以后就支持了

9、和 MyISAW 引擎比，innodb 对硬件资源要求比较高

2.5.2.3 innodb 引擎适应的生产业务场景

1、需要事务支持的业务（具有较好的事务特性）
2、行级锁定对高并发有很好的适应能力，但需要确保查询是通过索引完成
3、数据读写及更新都比较频繁的场景
4、数据一致性要求比较高的业务，例如：充值转账，银行卡转账
5、硬件设备内存较大，可以利用 innodb 较好的缓存能力来提高内存利用率，尽可能减少磁盘 IO

2.5.2.4 关于 innodb 引擎的一些参数设置

```
[root@mysql ~]# grep -i innodb /mysqldata/3306/my.cnf
default_table_type = InnoDB
innodb_additional_mem_pool_size = 4M
innodb_buffer_pool_size = 32M
innodb_data_file_path = ibdata1:128M:autoextend
innodb_file_io_threads = 4
innodb_thread_concurrency = 8
innodb_flush_log_at_trx_commit = 2
innodb_log_buffer_size = 2M
innodb_log_file_size = 4M
innodb_log_files_in_group = 3
innodb_max_dirty_pages_pct = 90
innodb_lock_wait_timeout = 120
innodb_file_per_table = 0
```

2.5.2.5 innodb 引擎调忧精要

1、主键尽可能小，避免给 secondary index 带来过大的空间负担
2、建立有效索引避免全表扫描，因为会使用表锁
3、尽可能缓存所有的索引和数据，提高响应速度，减少磁盘 IO 消耗
4、在大批量小插入的时候，尽量自己控制事务而不要使用 autocommit 自动提交，有开关可以控制提交方式
5、合理设置 innodb_flush_log_at_trx_commit 参数值，不要过度追求安全性
如果 innodb_flush_log_at_trx_commit 的值为 0，log buffer 每秒就会刷写日志文件到磁盘，提交事务的时候不做任何操作
6、避免主键更新，因为这会带来大量的数据移动

2.5.3 有关 mysql 引擎特别说明

Feature	MyISAM	Memory	InnoDB	Archive	NDB
Storage limits	256TB	RAM	64TB	None	384EB
Transactions	No	No	Yes	No	Yes
Locking granularity	Table	Table	Row	Table	Row
MVCC	No	No	Yes	No	No
Geospatial data type support	Yes	No	Yes	Yes	Yes
Geospatial indexing support	Yes	No	No	No	No
B-tree indexes	Yes	Yes	Yes	No	No
T-tree indexes	No	No	No	No	Yes
Hash indexes	No	Yes	No [a]	No	Yes
Full-text search indexes	Yes	No	Yes [b]	No	No
Clustered indexes	No	No	Yes	No	No
Data caches	No	N/A	Yes	No	Yes
Index caches	Yes	N/A	Yes	No	Yes
Compressed data	Yes [c]	No	Yes [d]	Yes	No
Encrypted data [e]	Yes	Yes	Yes	Yes	Yes
Cluster database support	No	No	No	No	Yes
Replication support [f]	Yes	Yes	Yes	Yes	Yes
Foreign key support	No	No	Yes	No	No
Backup / point-in-time recovery [g]	Yes	Yes	Yes	Yes	Yes
Query cache support	Yes	Yes	Yes	Yes	Yes
Update statistics for data dictionary	Yes	Yes	Yes	Yes	Yes

2.5.4 MySQL 数据库字符集介绍

简单的说，字符集就是一套文字符号及其编码，比较规则的集合。mysql 数据库字符集包括字符集（character）和校队规则（collation）两个概念，其中，字符集用来应以 mysql 数据库字符串的存储方式，而校队规则则是定义比较字符串的方式

2.5.4 关于 mysql 的字符集

2.5.4.1 mysql 常见的字符集？

常用字符集	一个汉字长度	说明
GBK	2	不是国际标准，对中文环境支持的很好
UTF8	3	中英文混合的环境，建议使用此字符集，用的比较多的
latin1	1	mysql 的默认字符集
utf8mb4	4	UTF-8 Unicode，移动互联网

2.5.4.2 mysql 如何选择合适的字符集？

1、如果处理各种各样的文字，发布到不同语言国家地区，应选 Unicode 字符集，对 mysql 来说就是 UTF-8(每个汉字三字节)，如果应用需处理英文，仅用少量汉字 UTF-8 更好
2、如只需支持中文，并且数据量很大，性能要求也很高，可选 GBK（定长 每个汉字占双字节，英文也占双字节），如果需要大量运算，比较排序等，定长字符集，更快，性能高
3、处理移动互联网业务，可能需要使用 utf8mb4 字符集
提示：没有特别的要求选择 utf8

2.5.4.3 如何查看字符集

```
mysql> show character set ;
+----------+-----------------------------+---------------------+--------+
| Charset  | Description                 | Default collation   | Maxlen |
+----------+-----------------------------+---------------------+--------+
| big5     | Big5 Traditional Chinese    | big5_chinese_ci     |      2 |
| dec8     | DEC West European           | dec8_swedish_ci     |      1 |
```

```
| cp850      | DOS West European                  | cp850_general_ci       |          1 |
| hp8        | HP West European                   | hp8_english_ci         |          1 |
| koi8r      | KOI8-R Relcom Russian              | koi8r_general_ci       |          1 |
| latin1     | cp1252 West European               | latin1_swedish_ci      |          1 |
| latin2     | ISO 8859-2 Central European | latin2_general_ci      |          1 |
| swe7       | 7bit Swedish                       | swe7_swedish_ci        |          1 |
| ascii      | US ASCII                           | ascii_general_ci       |          1 |
| ujis       | EUC-JP Japanese                    | ujis_japanese_ci       |          3 |
| sjis       | Shift-JIS Japanese                 | sjis_japanese_ci       |          2 |
| hebrew     | ISO 8859-8 Hebrew                  | hebrew_general_ci      |          1 |
| tis620     | TIS620 Thai                        | tis620_thai_ci         |          1 |
| euckr      | EUC-KR Korean                      | euckr_korean_ci        |          2 |
| koi8u      | KOI8-U Ukrainian                   | koi8u_general_ci       |          1 |
| gb2312     | GB2312 Simplified Chinese          | gb2312_chinese_ci      |          2 |
| greek      | ISO 8859-7 Greek                   | greek_general_ci       |          1 |
| cp1250     | Windows Central European           | cp1250_general_ci      |          1 |
| gbk        | GBK Simplified Chinese             | gbk_chinese_ci         |          2 |
| latin5     | ISO 8859-9 Turkish                 | latin5_turkish_ci      |          1 |
| armscii8 | ARMSCII-8 Armenian                   | armscii8_general_ci |          1 |
| utf8       | UTF-8 Unicode                      | utf8_general_ci        |          3 |
| ucs2       | UCS-2 Unicode                      | ucs2_general_ci        |          2 |
| cp866      | DOS Russian                        | cp866_general_ci       |          1 |
| keybcs2    | DOS Kamenicky Czech-Slovak         | keybcs2_general_ci     |          1 |
| macce      | Mac Central European               | macce_general_ci       |          1 |
| macroman | Mac West European                    | macroman_general_ci |          1 |
| cp852      | DOS Central European               | cp852_general_ci       |          1 |
| latin7     | ISO 8859-13 Baltic                 | latin7_general_ci      |          1 |
| utf8mb4    | UTF-8 Unicode                      | utf8mb4_general_ci     |          4 |
| cp1251     | Windows Cyrillic                   | cp1251_general_ci      |          1 |
| utf16      | UTF-16 Unicode                     | utf16_general_ci       |          4 |
| cp1256     | Windows Arabic                     | cp1256_general_ci      |          1 |
| cp1257     | Windows Baltic                     | cp1257_general_ci      |          1 |
| utf32      | UTF-32 Unicode                     | utf32_general_ci       |          4 |
| binary     | Binary pseudo charset              | binary                 |          1 |
| geostd8    | GEOSTD8 Georgian                   | geostd8_general_ci     |          1 |
| cp932      | SJIS for Windows Japanese          | cp932_japanese_ci      |          2 |
| eucjpms    | UJIS for Windows Japanese          | eucjpms_japanese_ci |          3 |
+----------+----------------------------+--------------------+--------+
39 rows in set (0.39 sec)
```

2.5.4.4 不同字符集参数的含义如下（要想数据库字符不乱码，下面几个字符集要相同）

```
mysql> show variables like '%character_set%';
+--------------------------+----------------------------------------+
| Variable_name            | Value
+--------------------------+----------------------------------------+
```

```
| character_set_client     | utf8                       客户端字符集
| character_set_connection | utf8                       客户端连接字符集
| character_set_database   | utf8                       数据库字符集，配置文件指定或建库建表指定
| character_set_filesystem | binary                     文件系统字符集
| character_set_results    | utf8                        客户端返回结果字符集
| character_set_server     | utf8                        服务器字符集，配置文件指定或建库建表指定
| character_set_system     | utf8                        系统字符集
| character_sets_dir       | /application/mysql-5.5.27/share/charsets/ |
+--------------------------+-------------------------------------------+
8 rows in set (0.14 sec)
```

2.5.4.5 set names 字符集名，此命令有什么作用

控制客户端的字符集

```
mysql> show variables like '%character_set%';
+--------------------------+-------------------------------------------+
| Variable_name                | Value                                                 |
+--------------------------+-------------------------------------------+
| character_set_client     | utf8                                                   |
| character_set_connection | utf8                                                   |
| character_set_database   | utf8                                                    |
| character_set_filesystem | binary                                                |
| character_set_results    | utf8                                                    |
| character_set_server     | utf8                                                    |
| character_set_system     | utf8                                                     |
| character_sets_dir       | /application/mysql-5.5.27/share/charsets/ |
+--------------------------+-------------------------------------------+
8 rows in set (0.14 sec)
mysql> set names gbk;
Query OK, 0 rows affected (0.03 sec)
mysql> show variables like '%character_set%';
+--------------------------+-------------------------------------------+
| Variable_name                | Value                                                 |
+--------------------------+-------------------------------------------+
| character_set_client     | gbk                                                    |
| character_set_connection | gbk                                                    |
| character_set_database   | utf8                                                   |
| character_set_filesystem | binary                                                |
| character_set_results    | gbk                                                    |
| character_set_server     | utf8                                                   |
| character_set_system     | utf8                                                    |
| character_sets_dir       | /application/mysql-5.5.27/share/charsets/ |
+--------------------------+-------------------------------------------+
8 rows in set (0.04 sec)
```

2.5.4.6 根据配置文件更改客户端字符集

```
[client]
```

default-character-set=latin1
注意：多实例此步骤字符集的修改，要修改/etc/my.cnf

2.5.4.7 更改 mysql 服务端的字符集

法 1、按如下要求更改 my.cnf 参数
[mysqld]
default-character-set=latin1 <-适合 5.1 及以前版本
character-set-server=latin1 <-适合 5.5
法 2、编译的时候指定字符集
-DDEFAULT_CHARSET=utf8
-DDEFAULT_COLLATION=utf8_general_ci
-DEXTRA_CHARSETS=gbk,gb2312,utf8,ascii

2.5.4.8 怎么解决乱码问题

1、建库建表字符集统一
2、程序字符集统一
3、连接数据库的 linux 客户端字符集统一
4、编译时字符集统一
5、系统字符集统一
6、服务端字符集

2.5.4.9 插入数据不乱码的方法

1、尽量不在 mysql 命令行直接插入数据（SSH 客户端影响），SQL 语句文件形式
2、sql 文件的格式统一”utf8 没有签名”（在 windows 中文件保存的方式）
3、可以 mysql 命令行中用 source 执行 sql 文件
4、命令方式导入数据 mysql –uroot –ppcwangjixusn oldboy <test.sql
5、sql 文件里 set names utf8,或 mysql –uroot –ppcwangjixuan oldboy –default-character-set=utf8<test.sql

2.5.5.0 更改数据库的字符集

```
mysql> show create database oldboy;
+----------+-----------------------------------------------------------------+
| Database | Create Database                                                 |
+----------+-----------------------------------------------------------------+
| oldboy   | CREATE DATABASE `oldboy` /*!40100 DEFAULT CHARACTER SET utf8 */ |
+----------+-----------------------------------------------------------------+
1 row in set (0.01 sec)
mysql> alter database oldboy character set gbk collate = gbk_chinese_ci;
Query OK, 1 row affected (0.13 sec)
mysql> show create database oldboy;
+----------+-----------------------------------------------------------------+
| Database | Create Database                                                 |
+----------+-----------------------------------------------------------------+
| oldboy   | CREATE DATABASE `oldboy` /*!40100 DEFAULT CHARACTER SET gbk */ |
+----------+-----------------------------------------------------------------+
1 row in set (0.00 sec)
```

2.5.5.1 更改表的字符集

```
mysql> show create table test\G;
```

```
*************************** 1. row ***************************
       Table: test
Create Table: CREATE TABLE `test` (
  `id` int(4) NOT NULL DEFAULT '0',
  `age` tinyint(2) DEFAULT NULL,
  `name` varchar(16) DEFAULT NULL,
  `shouji` char(11) DEFAULT NULL,
  PRIMARY KEY (`id`),
  KEY `index_name_and_shouji` (`name`(6),`shouji`(8))
) ENGINE=InnoDB DEFAULT CHARSET=utf8
1 row in set (0.00 sec)
mysql> alter table test character set gbk;
Query OK, 5 rows affected (0.88 sec)
Records: 5  Duplicates: 0  Warnings: 0
mysql> show create table test\G;
*************************** 1. row ***************************
       Table: test
Create Table: CREATE TABLE `test` (
  `id` int(4) NOT NULL DEFAULT '0',
  `age` tinyint(2) DEFAULT NULL,
  `name` varchar(16) CHARACTER SET utf8 DEFAULT NULL,
  `shouji` char(11) CHARACTER SET utf8 DEFAULT NULL,
  PRIMARY KEY (`id`),
  KEY `index_name_and_shouji` (`name`(6),`shouji`(8))
) ENGINE=InnoDB DEFAULT CHARSET=gbk
1 row in set (0.00 sec)
```

2.5.6 模拟将 latin1 字符集的数据库修改成 UTF8 字符集的实际过程

1、导出表结构

mysqldump –uroot –ppcwangjixuan –default-set = latin1 –d oldboy >alltable.sql

–default-set = utf8 表示以 UTF8 字符集进行连接 –d 只导表结构

2、编辑表结构语句 alltable.sql 将所有 latin1 字符串改成 UTF8（用 sed 命令）

3、确保数据库不再更新，导出所有数据（不带表结构）

mysqldump –uroot –ppcwangjixuan –quick --no-create-info –extended-insert –default-character-set=latin1 oldboy >alldata.sql

参数说明：

--quick：用于存储大的表，强制 mysqldump 从服务器一次一行的检索数据而不是检索所有行，并输出前 CACHE 到内存中

--no-create-info:不创建 create table 语句

--extended-insert：使用包括几个 values 列表的多行 insert 语法，这样文件更小，IO 也小，导入数据时会非常快

--default-character-set =latin1 按照原有字符集导出数据，这样导出的文件中，所有中文都是可见的，不会保存乱码

4、修改 my.cnf 配置调整客户端及服务端字符集，重启生效

5、通过 utf8 建库

删除原库，然后 create database oldboy default charset utf8；
6、导入表结构（更改过字符集的表结构）
mysql –u root –ppcwangjixuan oldboy <alltabl.sql
7、 导入数据
mysql –uroot –ppcwangjixuan oldboy<alldata.sql
注意：选择目标字符集时，要注意最好大于等于源字符集（字库更大），否则，可能会丢失不被支持的数据

2.5.7 工作中 mysql 集群图