关系型数据库与非关系型数据库

1、关系型数据库

2、非关系型数据库

3、非关系型数据库产生背景---帮助关系型数据库缓解压力

1) High performance--对数据库高并发读写需求

2) Huge storage--对海量数据库高效存储与访问需求

3) High Scalability && High Availability---对数据库高扩展性与高可用性需求

Redis

1.Redis简介

2.Redis的工作原理

3.Redis的优点

4.Redis与memcached及其他类型数据库的对比---面试

5.Redis安装部署

1.下载redis源码包，解压并进入解压路径

2.执行make编译安装

3.生成启动脚本及配置文件信息

4.启动服务

5.配置参数

6.redis命令工具

7.redis-cli的语法

8.Redis-cli中help命令的使用方法

9.redis-benchmark 测试工具　　--了解

关系型数据库与非关系型数据库

数据库按照其结构可以分为关系型数据库与其他数据库,而这些其他数据库我们]将统称为非关系型数据库。

1、关系型数据库

关系型数据库是一个结构化的数据库

创建在关系模型的基础上，一般面向记录

他借助与集合代数等数学概念和方法来处理数据库中的数据，关系模型是二维表格模型，因而一个关系型数据库就是由二维表及其之间的联系组成的一个数据一个数据组织。

现实世界中，各种实体与实体之间的各种联系都可以用关系模型来表示。

SQL语句(Structured OueryLanage，结构化查询语言)就是一种基于关系型数据库的语言，用于执行对关系型数据库中据的检索和操作

主流的关系型数据库(mysgl、 oracle、 sgl server、Microsoft Access、DB2) 等

2、非关系型数据库

NoSQL (Not Only SQL)其意思是不仅仅是SQL,是非关系型数据库的总称，

主流的非关系型数据库

Memcached、Redis、 MongoDB、Hbase、CouhDB 等以上的这些数据库它们的存储方式，存储结构，以及使用场景是完全不同的，

所以我们认为他是一个非关系型数据库的集合，而不是像关系型数据库一样是一 个统称。

换言之，主流关系型数据库以外的数据库都是非关系型数据库

NoSQL数据库凭借着其非关系型，分布式，开源的横向扩展等优势，

被认为是下一代数据库产品

3、非关系型数据库产生背景---帮助关系型数据库缓解压力

关系型数据库已经诞生很久了,而且我们一直在使用,没有什么问题，面对这样的情况，为什么还会产生非关系型数据库?“下面我们就来介绍一下非关系型数据库产生的背景

随着Web2.0网站的兴起，关系型数据库在应对Web2.0网站，

特别是海量数据库和高并发的SNS (Social Networking Services，即社交网络服务)类型的Web 2.0纯动态网站时，

暴露出很多难以解决的问题，例如以下三高问题

1) High performance--对数据库高并发读写需求

Web2.0网站会根据用户的个性化信息来实时生成动态页面和提供动态信息，因为，无法使用动态页面静态化技术，所以数据库的并发负载非常高，一般会达到10000次/s以上的读写请求。关系型数据库对于上万次的查询请求还是可以勉强支撑的。当出现上万次的写数据请求时，硬盘I/O就已经无法承受了，对于普通的BBS网站，往往也会存在高并发的读数据请求，如明星鹿晗在微博上公布恋情，结果因为流量过大而引发微博瘫痪。

2) Huge storage--对海量数据库高效存储与访问需求

类似于Facebook、Friendfeed 这样的SNS网站，每天会产生大量的用户动态信息，例如Friendfeed一个月就会产生2.5亿条用户动态信息，对于关系型数据库来说，在- -个包含2.5

亿条记录的表中执行SQL查询，效率是非常低的。

3) High Scalability && High Availability---对数据库高扩展性与高可用性需求

在Web架构中，数据库是最难进行横向扩展的，当应用系统的用户量与访问量与日俱增时，数据库是没办法像Web服务一样，

简单地通过添加硬件和服务节点其性能和负载均衡能力的，
尤其对于一些需要24h对外提供服务的网站来说，
数据库的升级与扩展性往往伴随着停机维护与数据迁移，其工作量是非常庞大的
关系型数据库和非关系型数据库都有各自的特点与应用场景
再者的紧密结合将会给Web2.0的数据库发展带来新的思路
让关系型数据库关注在关系上，非关系数据库关注在存储上

Redis

！！！！！

1.Redis简介

Redis是一个开源的，使用C语言编写，支持网络，
可基于内存工作亦可持久化(AOF, RDB)的日志型，
key-values (键值对)数据库　　类似变量名和变量值
一个速度极快的非关系性数据库，
也就是我们所说的NoSQL数据库，
他可以存储(key)与5种不同类型的值(value) 之间的映射(mapping)，
可以将存储在内存的键值对数持久化到硬盘，
可以使用复制特性来扩展读性能，
还可以使用客户端分片来扩展性能，并且它还提供了多种语言的API（给很多开发语言保留了接口）

Redis的所有数据都是保存在内存中，（特点：快）
然后不定期的通过异步方式保存到磁盘上(这个称为半持久化RDB);类似于mysqldump
也可以把每一次数据 变化都写入到一个append onlyfile (aof)里面(这称为“全持久化”)。
类似于binlog
全持久化与半持久化的方式在工作中适当的使用，
在如今的生产环境中，如果使用全持久化的方式可能会造成append onlyfile的文件过大，
如果是半持久化的话又没有办法保证数据的安全性，
所以希望大家适当的使用

机械硬盘：150M/s　　----mysql
固态硬盘：400M/s
内存速度：15000M/s　　----Redis

2.Redis的工作原理

Redis服务器程序是一个单进程模型，
也就是说在一台服务 器上可以开启多个redis 进程（多实例），
而redis的实际处理速度则完全依靠于主进程的执行速率，
若在服务器上只运行一个redis进程，
当多个客户端同时访问时，服务器处理能力会有一定程度的下降，
若在一个服务器上开启多个redis进程，
redis在提高并发处理能力的同时也会给CPU造成很大的压力，
所以在实际生产环境中，结合实际服务器环境来决定如何使用

redis　　--缓存（缓解数据库压力），解决session问题

3.Redis的优点

- 具有极高的数据读写速度数据读取速度最高可达11万次/s，
  数据写入速度最高可达8万1千次/s
- 支持丰富的数据类型，支持丰富的数据类型不仅支持key-values 数据类型，
  还支持Strings，Lists， Hashes， Sets，及Ordered Sets 等数据类型操作
- 支持数据的持久化，
  在这一点上redis远远强于memcached,
  redis 可以将数据保存到磁盘中，重启后还可以继续加载使用
- 原子性redis的所有操作都是原子性的
- 支持数据备份及master-slave模式的数据备份

4.Redis与memcached及其他类型数据库的对比---面试

面试题----同类型产品的对比
Redis与memcached
Apache与nginx
LVS+nginx　与　haproxy

Redis经常被拿来与memcached进行比较
两者都可用于存储(键值)数据
性能也相差无几
但是redis能够自动以两种不同的方式将数据写入硬盘

而且Redis.除了能存储普通的字符串键

还能存储其它四种数据结构

使得reids可以用于解决更为广泛的问题

并且即可以作为主主数据库使用，又可以作为其他存储系统的辅助数据库

对比学习（集群），效率更高

名称	类型	数据存储选项	查询类型	附加功能
Redis	使用内存存储的非关系性数据库	字符串，列表，集合，散列表，有序集合	每种数据类型专属的命令，以及批量操作和不完全的事物支持	发布与订阅，主从复制，持久化，脚本
Memcached	使用内存存储的键值缓存	键值之间的映射	创建，读取，删除，更新等命令	多线程服务器用于提升性能
MySQL	关系型数据库	每个数据库可以包含多个表，每个表可以包含多个行;可以处理多个表的视图;支持空间和第三方扩展	Select, insert，update, delete, drop,函数，存储过程-	支持ACID性质(需要使用InnoDB)，主从复制，主主复制
MongoDB	使用硬盘存储(on-disk)的非关系文档存储	每个数据库可以包含多个表，每个表可以包含多个schema的BSON文档	创建，读取，更新删除，条件查询等命令口	支持map-reduce操作，主从复制，分片，空间索引

5.Redis安装部署

原理是用来在面试中聊天的~

也可以用yum安装，但一般用源码的多

1.下载redis源码包，解压并进入解压路径

官方下载：wget http://download.redis.io/releases/redis 4.0.10.tar.gz

[root@localhost ~]# tar xf redis-4.0.10.tar.gz -C /usr/src/

[root@localhost ~]# cd /usr/src/redis-4.0.10/

2.执行make编译安装

在我们之前的学习中，我们都会执行./configure进行编译从而生成Makefile,

redis 的源码包中则直接提供了Makefile. 文件所以在解压完成之后直接进行make编译即可)

```
[root@localhost redis-4.0.10]# make && make install
```

redis其实可以不用执行make install

但是为了后面方便我们执行所以执行make install

在安装过程中，若想更改默认的安装路径，可使用以下命令格式来进行安装操作。

```
make PREFIX=安装路径 install
```

3.生成启动脚本及配置文件信息

makeinstall只是生成了二进制文件到系统中，并没有启动脚本和配置文件，

所以此时使用redis默认提供的install server.sh 脚本进行生成启动脚本和配置文件

```
[root@localhost redis-4.0.10]# cd utils/                    #在解压路径下的utils中
[root@localhost utils]# ./install_server.sh                 #执行安装脚本
Welcome to the redis service installer
This script will help you easily set up a running redis server
# 路回车即可
Please select the redis port for this instance: [6379]            # 设置服务端口号
Selecting default: 6379
Please select the redis config file name [/etc/redis/6379.conf]  #设置主配置文件存放位置
Selected default - /etc/redis/6379.conf
Please select the redis log file name [/var/log/redis_6379.log]   #设置reids日志存放位置
Selected default - /var/log/redis_6379.log
Please select the data directory for this instance [/var/lib/redis/6379]   #设置数据目录
Selected default - /var/lib/redis/6379
Please select the redis executable path [/usr/local/bin/redis-server]    #设置执行命令
Selected config:
Port           : 6379                                       //服务端口
Config file    : /etc/redis/6379.conf                 //配置文件
Log file       : /var/log/redis_6379.log               //日志文件
Data dir       : /var/lib/redis/6379                 //数据目录
Executable     : /usr/local/bin/redis-server          //服务端命令
Cli Executable : /usr/local/bin/redis-cli               //客户端命令
Is this ok? Then press ENTER to go on or Ctrl-C to abort.
Copied /tmp/6379.conf => /etc/init.d/redis_6379
```

```
Installing service...
Successfully added to chkconfig!
Successfully added to runlevels 345!
Starting Redis server...
Installation successful!
```

4.启动服务

在执行安装脚本后redis服务其实已经启动了

```
[root@localhost ~]# netstat -lnpt|grep 6379
tcp        0      0 127.0.0.1:6379          0.0.0.0:*               LISTEN
80856/redis-server
```

我们已经为redis生成了启动脚本及配置文件

所以此时我们已经可以控制redis的启动及停止了

```
[root@localhost ~]# service redis_6379 stop
```

或者：

```
[root@localhost ~]# /etc/init.d/redis_6379 stop
Stopping ...
Redis stopped
[root@localhost ~]# /etc/init.d/redis_6379 start
Starting Redis server...
[root@localhost ~]# netstat -lnpt|grep redis
tcp        0      0 127.0.0.1:6379          0.0.0.0:*               LISTEN
81007/redis-server
```

5.配置参数

redis主配置文件为/etc/redis/6379.conf

由注释行和设置行两部分组成

#开头代表的是注释行

```
[root@localhost ~]# vim /etc/redis/6379.conf
  70 bind 192.168.200.107                        //监听的主机地址
  93 port 6379                                          //端口
 137 daemonize yes                                 //启用守护进程
 159 pidfile /var/run/redis_6379.pid          //指定PID文件
 167 loglevel notice                               //日志级别
 172 logfile /var/log/redis_6379.log            //指定日志文件
[root@localhost ~]# /etc/init.d/redis_6379 restart
Stopping ...
Redis stopped
```

Starting Redis server...

tcp 0 0 192.168.200.107:6379 0.0.0.0:* LISTEN 81139/redis-server

常用配置项的解释:

1. Redis默认不是以守护进程的方式运行，

可以通过该配置项修改，使用yes启用守护进程

daemonize no

2.当Redis以守护进程方式运行时，

Redis 默认会把pid写入/var/run/redis. pid文件，可以通过pidfile指定

pidfile /var/run/redis. pid

3.指定Redis监听端口，默认端口为6379

作者在自己的一篇博文中解释了为什么选用6379作为默认端口，

因为6379在手机按键上MERZ对应的号码，而MERZ取自意大利歌女

Alessia Merz的名字

port 6379

4.绑定的主机地址

bind 127.0. 0.1

5.当客户端闲置多长时间后关闭连接，如果指定为0，表示关闭该功能

timeout 300

6.指定日志记录级别，Redis 总共支持四个级别:

debug、 verbose、 notice、warning, 默认为verbosed

7.日志记录方式，默认为标准输出，

如果配置Redis为守护进程方式运行，而这里又配置为日志记录方式为标准输出，

则日志将会发送给/dev/null

logfile stdout

8. 设置数据库的数量，默认数据库为0，

可以使用SELECT <dbid>命令在连接上指定数据库id

databases 16

9.指定在多长时间内，有多少次更新操作，就将数据同步到数据文件，

可以多个条件配合 save <seconds> <changes>

Redis默认配置文件中提供了三个条件:

save 900 1

save 300 10

save 60 10000

分别表示900秒(15 分钟)内有1个更改，300秒(5 分钟)内有10个更改以及

60秒内有10000个更改

10.指定存储至本地数据库时是否压缩数据，默认为yes, Redis 采用LZF压缩，

如果为了节省CPU时间，可以关闭该选项，但会导致数据库文件变的巨大

rdbcompression yest

11.指定本地数据库文件名，默认值为dump. rdb

dbfi l ename dump. rdb

[root@localhost ~]# ls /var/lib/redis/6379

dump.rdb

12.指定本地数据库存放目录

dir ./

13.设置当本机为slave服务时，设置master服务的IP地址及端口，

在Redis启动时，它会自动从master进行数据同步

slaveof <masterip> <masterport>

14.当master服务设置了密码保护时，slave 服务连接master的密码

masterauth <master- password>

15.设置Redis连接密码，如果配置了连接密码，客户端在连接Redis时需要通过AUTH

<password>命令提供密码，默认关闭

requi repass foobared

16.设置同一时间最大客户端连接数，默认无限制，

Redis 可以同时打开的客户端连接数为Redis进程可以打开的最大文件描述符数，

如果设置maxclients 0，表示不作限制。

当客户端连接数到达限制时，Redis 会关闭新的连接

并向客户端返回max number of clientsreached错误信息

17.指定Redis最大内存限制，Redis 在启动时会把数据加载到内存中，

达到最大内存后，Redis 会先尝试清除已到期或即将到期的Key，

当此方法处理后，仍然到达最大内存设置，将无法再进行写入操作，

但仍然可以进行读取操作。

Redis 新的ym机制，会把Key存放内存，Value 会存放在swap区

maxmemory <bytes>

18.指定是否在每次更新操作后进行日志记录，

Redis 在默认情况下是异步的把数据写入磁盘，

如果不开启，可能会在断电时导致一段时间内的数据丢失。

因为redis 本身同步数据文件是按上面save条件来同步的，

所以有的数据会在一段时间内 只存在于内存中。默认为no

appendonly no

19.指定更新日志文件名， 默认为appendonly. aof

appendfilename appendonly. aof

20.指定更新日志条件，共有3个可选值:

no:表示等操作系统进行数据缓存同步到磁盘(快)

always:表示每次更新操作后手动调用fsync()将数据写到磁盘(慢，安全)

everysec:表示每秒同步一次(折衷，默认值)。

appendfsync everysec

6.redis命令工具

Redis提供了多个命令工具，这些命令工具的作用分别如下所示:

➢redis-server: 用于启动redis的工具

➢redis-benchmark: 用于检测redis在本机的运行效率；压力测试

➢redis-check-aof: 修复apf持久化文件

➢redis-check-rdb: 修复rdb持久化文件

➢redis-cli: redis client 命令工具

➢redis-setinel: redis-server文件的软连接--哨兵

Redis数据库系统也是一个典型的C/S (客户端/服务器)架构的应用，

要访问redis数据库需要使用专门的客户端软件，

redis-cli 是redis.自带的命令行工具，

使用redis-cli连接进入redis数据库后进入提示符为“远程主机IP:端口号>”的数据库操作环境

7.redis-cli的语法

redis-cli -h 远程连接主机 -p 指定端口 -a 指定密码

注:

若未设置数据库密码-a选项可以忽略退出数据库操作环境可以执行quit 或exit

就可以返回到原来的shell文件

执行ping命令可以检测Redis服务是否启动

```
[root@localhost ~]# redis-cli
Could not connect to Redis at 127.0.0.1:6379: Connection refused
Could not connect to Redis at 127.0.0.1:6379: Connection refused
not connected> exit
[root@localhost ~]# redis-cli -h 192.168.200.107
```

```
192.168.200.107:6379> ping
PONG
192.168.200.107:6379>

[root@localhost ~]# vim /etc/redis/6379.conf
bind 127.0.0.1 192.168.200.107
[root@localhost ~]# rm -rf /var/run/redis_6379.pid
[root@localhost ~]# /etc/init.d/redis_6379 restart
[root@localhost ~]# netstat -lnpt
tcp        0      0 192.168.200.107:6379    0.0.0.0:*               LISTEN
81659/redis-server
tcp        0      0 127.0.0.1:6379          0.0.0.0:*               LISTEN
81659/redis-server
[root@localhost ~]# redis-cli
127.0.0.1:6379> ping
PONG
```

8.Redis-cli中help命令的使用方法

➢help @<group> :获取group命令列表

➢help <command>: 获取某个命令的帮助

➢help<tab>:获取可能帮助的主题列表

使用方法help 命令字<tab键>通过使

用tab查看命令

9.redis-benchmark 测试工具　　--了解

Redis-benchmark是redis官方自带的redis性能测试工具，

可以有效的测试redis服务的性能

基本的测试语法为:

reids-benchmark [option] [option value]

常用选项:

常用下面几个

-h:指定服务器名

-p:指定服务器端口

-S:指定服务器socket

-C:指定并发连接数

-n:指定请求连接数

-d: 以字节(B)的形式指定SET/GET值的数据大小

-k: 1=keep alive 0=reconnect

-r: SET/GET/INCR使用随机key, SADD 使用的随机值

-P:通过管道传输<numrea>请求

-q:强制退出redis。 仅显示query/sec 值

--csv :以CSV格式输出

-l :生成循环，永久执行测试

-t :仅运行以逗号分隔的测试命令列表

-I :idle模式。仅打开/v个idle连接并等待

10.Redis- benchmark应用实例

1、测试并发数为10请求连接数为100000个请求的性能

[root@localhost ~]# redis-benchmark -h localhost -p 6379 -c 10 -n 100000

2、测试存取大小为100B的数据包时redis的性能

11.Redis数据库常用命令

Redis数据库采用key-values (键值对)的数据存储形式

所使用的命令是set与get命令

➢set: 用于redis数据库中存放数据命令格式为set key value

➢get:用于redis数据库中获取数据命令格式为get keye

Redis简单命令使用

[root@localhost ~]# redis-cli

127.0.0.1:6379> ping

PONG

127.0.0.1:6379> info

set get 应用案例

127.0.0.1:6379> set name sofia

OK

127.0.0.1:6379> get name

"sofia"

12.key相关命令

keys: 在使用keys命令可以取符合规则的键值列表，

通常情况可以结合*，?等选项来使用

? :表示任意一位数据

*:表示任意数据

案例:

配置如下列信息键值列表

```
127.0.0.1:6379> set k1 1
OK
127.0.0.1:6379> set k2 2
OK
127.0.0.1:6379> set k3 3
OK
127.0.0.1:6379> set s1 4
OK
127.0.0.1:6379> set v55 5
OK
127.0.0.1:6379> keys *          #  最好别在工作中使用，容易拖死机器，万一有几十万个keys
 1) "v55"
 2) "mylist"
 3) "counter:__rand_int__"
 4) "k2"
 5) "k3"
 6) "s1"
 7) "key:__rand_int__"
 8) "name"
 9) "myset:__rand_int__"
10) "k1"
127.0.0.1:6379> keys k?    # 尽量用这个可以定位的
1) "k2"
2) "k3"
3) "k1"
```

exists命令

```
127.0.0.1:6379> exists name
(integer) 1
127.0.0.1:6379> exists names
(integer) 0
```

del 命令

```
127.0.0.1:6379> del k3
(integer) 1
127.0.0.1:6379> keys k*
1) "k2"
2) "key:__rand_int__"
3) "k1"
127.0.0.1:6379> get k3
(nil)
```

type命令

```
127.0.0.1:6379> type k1              查看类型
string
127.0.0.1:6379> get k1
"1"             # 字符串
```

rename命令

作用:对已有的key进行重命名

命令格式:rename 源key 目标key

注意:

使用rename命令进行重命名时，无论目标key是否存在都会进行重命名，

在实际使用过程中建议先使用exists 查看目标key是否存在，

再决定是否执行rename命令，以免覆盖重要的数据

```
127.0.0.1:6379> exists k1
(integer) 1
127.0.0.1:6379> keys k*
1) "k2"
2) "key:__rand_int__"
3) "k1"
127.0.0.1:6379> get k2
"2"
127.0.0.1:6379> rename k2 k1
```

```
OK
127.0.0.1:6379> get k1
"2"
```

renamenx命令

作用:对已有的key进行重命名，并检测新名是否存在

格式:renamenx 源key 目标key

注意:

使用renamenx 进行重命名时renamenx 会检查新名是否存在，

如目标key存在则个进行从命名

```
127.0.0.1:6379> keys k*
1) "k2"
2) "key:__rand_int__"
3) "k1"
127.0.0.1:6379> renamenx k2 k1
(integer) 0
127.0.0.1:6379> keys k*
1) "k2"
2) "key:__rand_int__"
3) "k1"
127.0.0.1:6379> renamenx k2 k5
(integer) 1
127.0.0.1:6379> keys k*
1) "k5"
2) "key:__rand_int__"
3) "k1"
```

dbsize命令

作用:查看当前数据库中key的数目

```
127.0.0.1:6379> dbsize
(integer) 9
```

13.多数据库常用命令

1.多数据库之间切换

Redis支持多数据库，redis在默认没有任何改动的情况下包含16个数据库，

数据库的名称是使用数值0~15来依次命名的，

而我们通过redis-cli打开的是默认的第一个库其显示为“<ip地址: 6379>” 的形式，

通过select命令进行切换后其格式会变为“<ip 地址:6379[n]>”

n表示select后面的数字

```
[root@localhost ~]# redis-cli
127.0.0.1:6379> select 5
OK
127.0.0.1:6379[5]> select 10
OK
127.0.0.1:6379[10]> select 0
OK
127.0.0.1:6379> get name
"sofia"
127.0.0.1:6379> select 1
OK
127.0.0.1:6379[1]> get name
(nil)
127.0.0.1:6379[1]> keys *
(empty list or set)
```

2.多数据库之间移动数据

Reids数据库中虽然包含0-15即16个库但是彼此之间在一定程度 上是相互独立的

从上面的例子我们可以看出在0库上创建的键在1上是没办法查看的所以redis为了方便维护及管理提供了一个不同库之间进行数据移动的命令move

```
127.0.0.1:6379[1]> select 0
OK
127.0.0.1:6379>
127.0.0.1:6379> keys *
1) "v55"
2) "mylist"
3) "counter:__rand_int__"
4) "s1"
5) "k5"
6) "key:__rand_int__"
7) "name"
8) "myset:__rand_int__"
```

```
9) "k1"
127.0.0.1:6379> move name 1
(integer) 1
127.0.0.1:6379> select 1
OK
127.0.0.1:6379[1]> keys *
1) "name"
127.0.0.1:6379[1]> get name
"sofia"
127.0.0.1:6379[1]> select 0
OK
127.0.0.1:6379> keys *
1) "v55"
2) "mylist"
3) "counter:__rand_int__"
4) "s1"
5) "k5"
6) "key:__rand_int__"
7) "myset:__rand_int__"
8) "k1"
127.0.0.1:6379> get name
(nil)
```

3.清除数据库内数据　　---谨慎使用，最好不用

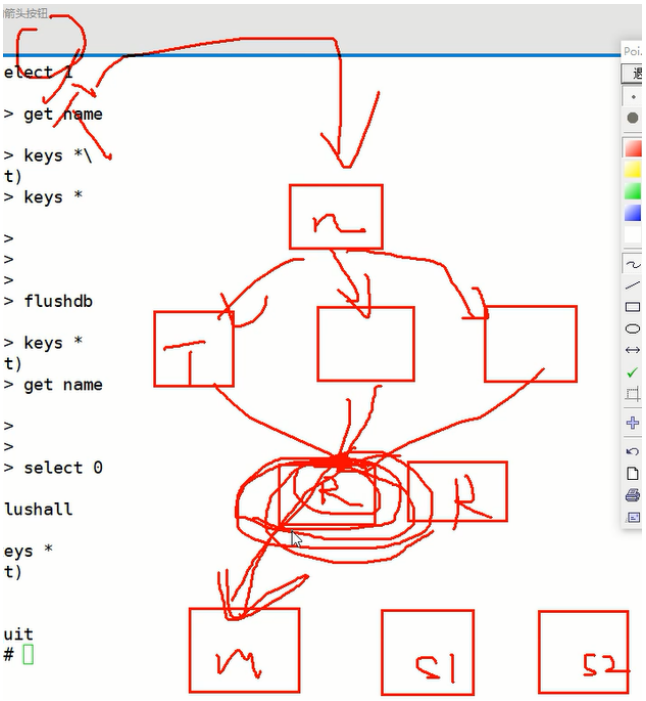
Redis清除数据库一般分为两部分

1. 清除当前数据库: flushdb
2. 清除所有数据库文件: flushall

注意：2操作较为危险不建议在生产环境下使用

```
127.0.0.1:6379> select 1
OK
127.0.0.1:6379[1]> get name
"sofia"
127.0.0.1:6379[1]> keys *
1) "name"
```

```
127.0.0.1:6379[1]> flushdb
OK
127.0.0.1:6379[1]> keys *
(empty list or set)
127.0.0.1:6379[1]> get name
(nil)
127.0.0.1:6379> keys *
(empty list or set)
```



数据的穿透，会对数据库造成很大的压力

redis的读取速度是mysql的差不多100倍