# Redis-day01-note

### Redis介绍

- 特点及优点

```
1、开源的，使用C编写，基于内存且支持持久化
2、高性能的Key-Value的NoSQL数据库
3、支持数据类型丰富，字符串strings，散列hashes，列表lists，集合sets，有序集合sorted sets 等等
4、支持多种编程语言（C C++ Python Java PHP ... ）
```

- 与其他数据库对比

```
1、MySQL ：关系型数据库，表格，基于磁盘，慢
2、MongoDB：键值对文档型数据库，值为JSON文档，基于磁盘，慢，存储数据类型单一
3、Redis的诞生是为了解决什么问题？？
   # 解决硬盘IO带来的性能瓶颈
```

- 应用场景

```
1、使用Redis来缓存一些经常被用到、或者需要耗费大量资源的内容，通过这些内容放到redis里面，程序可以快速读取这些内容
2、一个网站，如果某个页面经常会被访问到，或者创建页面时消耗的资源比较多，比如需要多次访问数据库、生成时间比较长等，我们可以使用redis将这个页面缓存起来，减轻网站负担，降低网站的延迟，比如说网站首页等
```

- redis版本

```
1、最新版本：5.0
2、常用版本：2.4、2.6、2.8、3.0、3.2、3.4、4.0、5.0
3、图形界面管理工具(写的一般)
    RedisDesktopManager
```

- 诞生历程

```
# 1、历史
LLOOGG.com 帮助别的网站统计用户信息，各个网站发送的浏览记录都会存储到存储队列，5-10000条记录，多余5条需要收费

# 2、原理
FIFO机制，先进先出，满了进一条就出一条，网站越多，队列越多，推入和弹出操作越多

# 3、技术及问题
开始使用MySQL进行硬盘读写，速度很慢，导致无法实时显示，所以自己写了一个列表结构的内存数据库，程序性能不会受到硬盘IO的限制，加了持久化的功能

# 4、redis数据库戛然而生
# 为了解决负载问题，所以发明了redis
```

- Redis附加功能

```
1、持久化
   将内存中数据保存到磁盘中，保证数据安全，方便进行数据备份和恢复
2、过期键功能
    为键设置一个过期时间，让它在指定时间内自动删除
    <节省内存空间>
    # 音乐播放器，日播放排名，过期自动删除
3、事务功能
    原子的执行多个操作
4、主从复制
5、Sentinel哨兵
```

# 安装

- Ubuntu

```
# 安装
sudo apt-get install redis-server
# 服务端启动

# 客户端连接

```

- Windows

```
1、下载安装包
   https://github.com/ServiceStack/redis-windows/blob/master/downloads/redis-64.3.0.503.zip
2、解压
3、启动服务端
   双击解压后的 redis-server.exe
4、客户端连接
   双击解压后的 redis-cli.exe

    # 问题：关闭终端后服务终止
# 解决：将Redis服务安装到本地服务
1、重命名 redis.windows.conf 为 redis.conf,作为redis服务的配置文件
```

```
2、cmd命令行，进入到redis-server.exe所在目录
3、执行：redis-server --service-install redis.conf --loglevel verbose
4、计算机-管理-服务-Redis-启动

# 卸载
到 redis-server.exe 所在路径执行：
1、redis-server --service-uninstall
2、sc delete Redis
```

# 配置文件详解

- 配置文件所在路径

```
1、Ubuntu
    /etc/redis/redis.conf

2、windows 下载解压后的redis文件夹中
    redis.windows.conf
    redis.conf
```

- 设置连接密码

```
1、requirepass 密码
2、重启服务
   sudo /etc/init.d/redis-server restart
3、客户端连接
   redis-cli -h 127.0.0.1 -p 6379 -a 123456
   127.0.0.1:6379>ping
```

- 允许远程连接

```
1、注释掉本地IP地址绑定

2、关闭保护模式

3、重启服务

```

- 远程连接测试

  **Windows连接Ubuntu的Redis服务**

```
# cmd命令行
1、e:
2、cd Redis3.0
3、redis-cli -h x.x.x.x -a 123456
4、x.x.x.x:6379>ping
```

# 数据类型

## *字符串类型(string)*

### 特点

```
1、字符串、数字，都会转为字符串来存储
2、以二进制的方式存储在内存中
```

### 字符串常用命令-必须掌握

```
# 1. 设置一个key-value
set key value
# 2. 获取key的值
get key
# 3. key不存在时再进行设置(nx)
set key value nx
# 4. 设置过期时间(ex)
set key value ex seconds

# 5. 同时设置多个key-value
mset key1 value1 key2 value2 key3 value3
# 6. 同时获取多个key-value
mget key1 key2 key3
```

### 字符串常用命令-作为了解

```
# 1.获取长度
strlen key
# 2.获取指定范围切片内容
getrange key start stop
# 3.从索引值开始，value替换原内容
setrange key index value
# 4.追加拼接value的值
append key value
```

### 数值操作-字符串类型数字(必须掌握)

```
# 整数操作
INCRBY key 步长
DECRBY key 步长
INCR key : +1操作
DECR key : -1操作
# 应用场景：抖音上有人关注你了，是不是可以用INCR呢，如果取消关注了是不是可以用DECR

# 浮点数操作：先转为数字类型，然后再进行相加减，不能使用append
incrbyfloat key step
```

### 键的命名规范

mset wang:email wangweichao@tedu.cn

```
127.0.0.1:6379> mset wang:email wangweichao@tedu.cn guo:email guods@tedu.cn
OK
127.0.0.1:6379> mget wang:email guo:email
1) "wangweichao@tedu.cn"
2) "guods@tedu.cn"
127.0.0.1:6379>
```

**string命令汇总**

```
# 字符串操作
1、set key value
2、set key value nx
3、get key
3、mset
4、mget
5、set key value ex seconds
6、strlen key
# 数字操作
7、incrby key 步长
8、decrby key 步长
9、incr key
10、decr key
11、incrbyfloat key number
# 设置过期时间的两种方式
# 方式一
1、set key value ex 3
# 方式二
1、set key value
2、expire key 5 # 秒
3、pexpire key 5 # 毫秒
# 查看存活时间
ttl key
# 删除过期
persist key
```

- **通用命令 适用于所有数据类型**

```
# 切换库
select number
# 查看键
keys *
# 键类型
TYPE key
# 键是否存在
exists key
# 删除键
del key
# 键重命名
rename key newkey
# 返回旧值并设置新值（如果键不存在，就创建并赋值）
getset key value
# 清除当前库中所有数据（慎用）
flushdb
# 清除所有库中所有数据（慎用）
```

```
flushall
```

**string数据类型注意**

```
# key值取值原则
1、key值不宜过长，消耗内存，且在数据中查找这类键值的计算成本高
2、不宜过短，可读性较差
# 值
1、一个字符串类型的值最多能存储512M内容
```

**练习**

```
1、查看 db0 库中所有的键
2、设置键 trill::username 对应的值为 user001，并查看
3、获取 trill::username 值的长度
4、一次性设置 trill::password 、trill::gender、trill::fansnumber 并查看（值自定义）
5、查看键 trill::score 是否存在
6、增加10个粉丝
7、增加2个粉丝（一个一个加）
8、有3个粉丝取消关注你了
9、又有1个粉丝取消关注你了
10、思考、思考、思考...,清除当前库
11、一万个思考之后，清除所有库
```

# *列表数据类型（List）*

- 特点

```
1、元素是字符串类型
2、列表头尾增删快，中间增删慢，增删元素是常态
3、元素可重复
4、最多可包含2^32 -1个元素
5、索引同python列表
```

- **头尾压入元素（LPUSH | RPUSH）**

1、LPUSH key value

2、RPUSH key value

```

```

- **查看|设置 列表元素**

  查看（LRANGE）

  ```

  ```

  获取指定位置元素（LINDEX）

```
1
```

设置指定位置元素的值（LSET）

```
1
```

获取列表长度（LLEN)

```
1
```

- **头尾弹出元素（LPOP | RPOP）**

  LPOP key : 从列表头部弹出一个元素

  RPOP key : 从列表尾部弹出一个元素

  RPOPLPUSH source destination : 从一个列表尾部弹出元素压入到另一个列表头部

```
1
```

- **移除指定元素（LREM）**

  LREM key count value

```
1 count>0: 表示从头部开始向表尾搜索，移除与value相等的元素，数量为count
2 count<0: 表示从尾部开始向表头搜索，移除与value相等的元素，数量为count
3 count=0: 移除表中所有与value相等的值
```

  示例

```
1
```

- **去除指定范围外元素（LTRIM）**

  LTRIM key start stop

```
1
```

  应用场景: 保存微博评论最后500条

```
1
```

- **列表中插入值（LINSERT）**

  LINSERT key BEFORE|AFTER pivot value

  key和pivot不存在，不进行任何操作

  示例代码

```
1
```

- **阻塞弹出（BLPOP | BRPOP）**

  BLPOP key timeout

  BRPOP key timeout

```
1、如果弹出的列表不存在或者为空，就会阻塞
2、超时时间设置为0，就是永久阻塞，直到有数据可以弹出
3、如果多个客户端阻塞再同一个列表上，使用First In First Service原则，先到先服务
```

  示例

```

```

**列表常用命令总结**

```
# 增
1、LPUSH key value1 value2
2、RPUSH key value1 value2
3、RPOPLPUSH source destination
4、LINSERT key after|before value newvalue
# 查
5、LRANGE key start stop
6、LLEN key
# 删
7、LPOP key
8、RPOP key
9、BLPOP key timeout
10、BRPOP key timeout
11、LREM key count value
12、LTRIM key start stop
# 改
13、LSET key index newvalue
```

**练习**

```
1、查看所有的键
2、向列表 spider::urls 中以RPUSH放入如下几个元素：01_baidu.com、02_taobao.com、03_sina.com、04_jd.com、05_xxx.com
3、查看列表中所有元素
4、查看列表长度
5、将列表中01_baidu.com 改为 01_tmall.com
6、在列表中04_jd.com之后再加1个元素 02_taobao.com
7、弹出列表中的最后一个元素
8、删除列表中所有的 02_taobao.com
9、剔除列表中的其他元素，只剩前3条
```

# 与python交互

- 模块

Ubuntu

```
sudo pip3 install redis
```

Windows

```
python -m pip install redis
```

- 使用流程

```
import redis
# 创建数据库连接对象
r = redis.Redis(host='127.0.0.1',port=6379,db=0,password='123456')
```

- 通用命令代码示例

```

```

**字符串命令代码示例**

```

```

**python操作list**

```

```

**位图操作bitmap（重要）**

位图不是真正的数据类型，它是定义在字符串类型中 一个字符串类型的值最多能存储512M字节的内容，位上限：2^32

**强势点**

```
可以实时的进行统计，极其节省空间。官方在模拟1亿2千8百万用户的模拟环境下，在一台MacBookPro上，典型的统计如“日用户数”的时间消耗小于50ms，占用16MB内存
```

**设置某一位上的值**

```
setbit key offset value
# offset是偏移量，从0开始
```

**示例**

```
# 默认扩展位以0填充
127.0.0.1:6379> set mykey ab
OK
127.0.0.1:6379> get mykey
"ab"
127.0.0.1:6379> SETBIT mykey 0 1
(integer) 0
127.0.0.1:6379> get mykey
"\xe1b"
127.0.0.1:6379>
```

**获取某一位上的值**

GETBIT key offset

```
127.0.0.1:6379> GETBIT mykey 3
(integer) 0
127.0.0.1:6379> GETBIT mykey 0
(integer) 1
127.0.0.1:6379>
```

**bitcount**

统计键所对应的值中有多少个 1

```
127.0.0.1:6379> SETBIT user001 1 1
(integer) 0
127.0.0.1:6379> SETBIT user001 30 1
(integer) 0
127.0.0.1:6379> bitcount user001
(integer) 2
127.0.0.1:6379>
```

**应用场景案例**

网站用户的上线次数统计（寻找活跃用户）

用户名为key，上线的天作为offset，上线设置为1

示例: 用户名为 user001 的用户，今年第1天上线，第30天上线

SETBIT user001 1 1

SETBIT user001 30 1

BITCOUNT user001

**代码实现**

```

```

**list案例: 一个进程负责生产url，一个进程负责消费url**

进程1: 生产者

```
1
```

进程2: 消费者

```
1
```