

**数据存储类型介绍**

**作为缓存使用**

1. 原始业务功能设计
   1. 秒杀
   2. 618 、双11
   3. 12306
2. 运营平台监控到的突发高频访问数据

突发时政要闻，被强势关注围观

1. 高频、复杂的统计数据

在线人数

投票排行榜

**附加功能**

1. 系统功能优化或升级

单服务器升级集群

Session管理

Token管理

Redis数据类型（5种常用）

string

hash

list

set

sorted\_set

redis数据存储格式

redis本身是一个map，其中所有的数据都是采用key：value的形式存储的

**注意：数据类型指的是存储数据的类型，也就是value部分的类型，key部分永远都是字符串**

1. **string**

**1.特点：**

存储的数据：单个数据，最简单的数据存储类型，也是最常用的数据存储类型

存储数据的格式：一个存储空间存一个数据

存储内容：通常使用字符串，如果字符串以整数的形式展示，可以作为数字操作使用，但本质是字符串

1. **基本操作**

set get

del

操作结果为成功/失败为标准就是以下两个返回值

成功：(integer) 1

失败：(integer) 0

**添加/修改多个数据**

mset key1 value1 key2 value2 …

**获取多个数据**

mget key1 key2

**获取字符个数（字符串长度）**

strlen key

**追加信息到原始信息尾部（如果原始信息存在就追加，否则就创建）**

appen key value -- 返回值（追加后的字符串长度）

**单指令多指令区别：**

假设要插入3个数据

单指令：3条请求 + 3条应答 + 3条运行时间

多指令：1条请求 + 1条应答 + 3条运行时间

1. **扩展操作**
2. **场景**

企业级应用中，分表操作是基本操作，使用多张表存储同类型数据，但是对应的主键id必须保证统一性，不能重复。Oracle数据库具有sequence设定，可以解决该问题，但是MySQL没有类似机制，如何解决？

解决方案：

1. 设置数值数据增加指定范围的值

incr key

incrby key increment 指定增加值

incrbyfloat key increment

1. 设置数值数据减少指定范围的值

decr key

decrby key increment

1. **string作为数值操作**

1）string在redis内部存储默认就是一个字符串，当遇到增减类操作incr，decr时会转成数值型进行计算。

2）redis所有的操作都是原子性的，采用单线程处理所有任务，故无需考虑并发带来的数据影响

**注意：按数值进行操作的数据，如果原始数据不能转成数值，或者超过了redis数值范围上限，将会报错**

**9223372036854775807（java中long型数据最大值，long.MAX\_VALUE）**

**ps1:**

1. **redis用于控制数据库表主键id，为数据库表主键提供生成策略，保障数据库表的主键唯一性**
2. **此方案适用于所有数据库，且支持数据库集群**
3. **场景**

数据时效性控制 -- 每四小时只能投一票

解决方案

设置数据具有指定的生命周期

setex key seconds value 秒

psetex key milliseconds value 毫秒

**该命令会被set重置而导致失效**

**ps2：**

**redis控制数据的生命周期，通过数据是否失效控制业务行为，适用于所有具有时效性限定控制的操作**

1. **string数据类型操作的注意事项**
2. 数据操作成功与否的反馈

成功：(integer)1 -- true

失败：(integer)0 -- false

1. 表示运行结果的值

(integer)3 -- 3 3个

(integer)1 -- 1 1个

具体表达的意思看使用的命令是什么

2）数据未获取到

nil -- null

3）数据最大存储量

512MB

4）数值计算最大范围（java中long的最大值）

1. **场景**

微博大V主页信息：粉丝、关注、博客

解决方案：

redis中为大V用户设定用户信息，以用户主键和属性值作为key，后台设定定时刷新策略即可

eg：

user:id:0000001:fans -- 1000000

user:id:0000001:blogs -- 100

user:id:0000001:focus -- 13

在redis中以json格式存储大V用户信息，定时刷新

user:id:000001 {id:000001,blogs:100,fans:12345678,focus:13}

**ps3:**

**redis应用于各种结构型和非结构性高热度数据访问加速**

数据库中的热点数据key命名惯例

表名：主键名：主键值：字段名

1. **hash类型**
2. **结构：**

前言：json格式，对象类数据的存储如果具有较频繁的更新需求会显得笨重

主键与属性分离

**主键 -- 属性键值对集合（哈希）**

1. 新的存储需求：对一系列存储的数据进行编组，方便管理，典型应用存储对象信息
2. 需要的存储结构：一个存储空间保存多个键值对数据
3. hash类型：底层使用哈希表结构实现数据存储

hash存储结构优化：

1. 如果field（属性）数量较少，存储结构优化为类数组结构
2. 如果field（属性）数量较多，采用hashmap结构
3. **基本操作**

**添加/修改数据**

hset key field value

**获取数据**

hget key field

hgetall key

**删除数据**

hdel key field1 …

**添加/修改多个数据**

hmset key field1 value1 field2 value2 …

**获取多个数据**

hmget key fileld1 field2 …

**获取哈希表中字段的数量**

hlen key

**获取哈希表中是否存在指定的字段**

hexists key field

1. **hash类型数据扩展操作**

**获取哈希表中所有的字段名或字段值**

hkeys key

hvals key

**设置指定字段的数值数据增加指定范围的值**

hincrby key field increment

hincrbyfloat key field increment

1. **hash类型数据操作的注意事项**

1）hash类型下的value只能存储字符串，不存在嵌套现象。如果数据未获取到，对应的值为（nil）

2）每个hash可以存储2^32 - 1个键值对

3）hash类型十分近对象的数据存储形式，并且可以灵活添加删除对象属性。但是hash设计初衷不是为了存储大量对象设计的，更不可以将hash作为对象列表使用

4）hgetall操作可以获取全部属性，如果内部field过多，遍历整体数据效率就会很低，有可能成为数据访问瓶颈

**5.hash使用场景**

**1）电商网站购物车设计与实现**

key：用户id

field：用户购买的商品编号，value：购买数量

解决方案：

1. 添加商品：追加全新的field和value
2. 浏览：遍历hash
3. 更改数量：自增/自减，设置value值
4. 删除商品：删除field
5. 清空：删除key

是否加速了购物车的呈现？--没有，呈现的只是商品编号

·每条购物车中的商品记录保存成两条field

·field1专用于保存购买数量

命名格式：商品id：nums

保存数据：数值

·field2专用于保存购物车中显示的信息，包含文字描述，图片地址，所属商家信息等 -- **独立hash**

命名格式：商品id：info

保存数据：json

hsetnx key field value 在运行前先检查商品有没有已经存在于公共的商品库了。防止每次商品本身信息没有改变，但是对于不同的用户都要重新加载一次

**ps4：redis应用于购物车数据存储设计**

**2）双11，优惠券，30，50，100元，抢购上限各1000张**

解决方案：

1. 以商家id作为key
2. 将参与抢购的商品id作为field
3. 将参与抢购商品数量作为对应的value
4. 抢购时使用降值的方式控制产品数量

**ps5：**

**redis应用于抢购，限购类，限量发放优惠券，激活码等业务的数据存储设计**

string存储对象（json）与hash存储对象

string：整体性强，但是修改起来较为困难 -- 主要读操作为主

hash：更新操作灵活（群组概念，对外产生唯一的key）