【第46话：Redis你真是让我又爱又恨，为什么你有那么多数据类型】

Hello 小伙伴们，这节课给大家讲一下Redis中非常常见的一个面试题：“请说一下Redis支持的数据类型”

Redis作为Java项目中缓存工具的一哥，目前已经发布了Redis 7版本。其高效读写性能，备受开发者喜爱。可以说在绝大多数互联网类型项目中都少不了Redis的身影。所以Redis中的面试题也比较多，我们先说说里面问的最多的几个问题之一“Redis支持的数据类型”

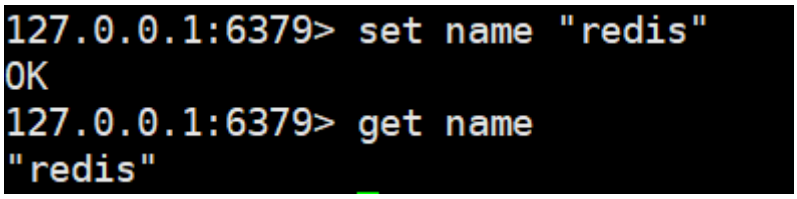
完全可以把Redis当做具备持久化能力的Map集合看待，其数据是key-value形式。Key在实际开发中多使用字符串，Value支持下面十种类型：

1. String 字符串
2. Hash 哈希：每个key可以包含多组属性值
3. List 列表：存储有序可重复数据
4. Set 集合：存储无序不可重复数据
5. Sorted Set 有序集合：每个值带有score分数，根据分数进行排序。
6. Stream类型（Redis5新出现的类型）：带有id的流元素
7. Geospatial 地理位置：存储地理位置，可以搜索指定半径内的内容。
8. HyperLogLog 超级日志：统一内容中存在多少个唯一值，属于一种概率算法。
9. bitmaps 位图：位向量的实现。
10. bitfields 位字段：做数字内容递增的。

字符串值(String)

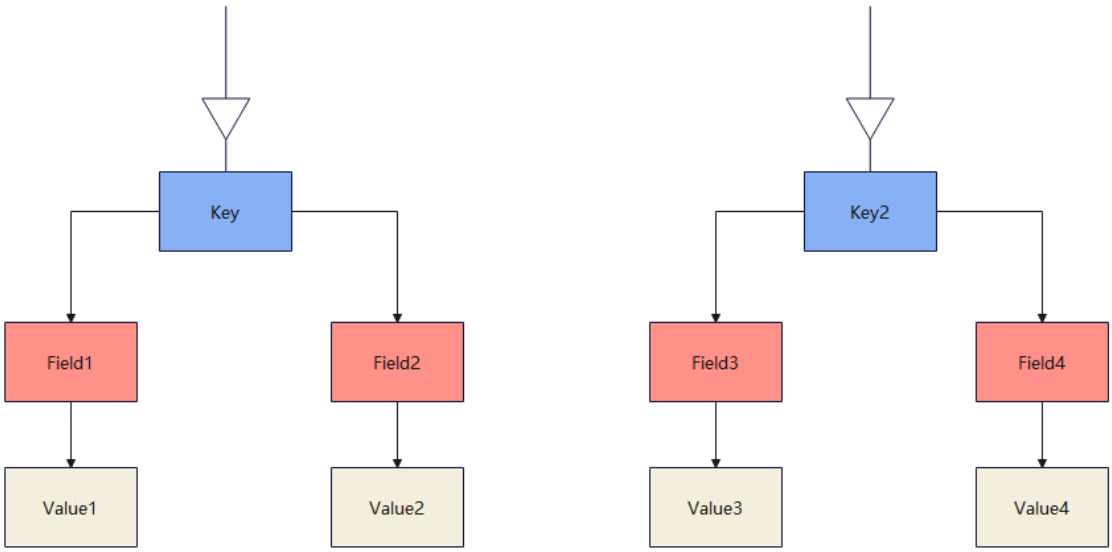
字符串值，这个对于小伙伴们好理解，就是文字内容。

字符串类型的值在项目开发过程中使用频率较高。例如Spring Data Redis里面自带六种序列化器，可以把Object类型值转换为字符串类型后存储到Redis中。



哈希表(Hash)

Hash类型的值中包含多组field value。给我们的感觉好像每个key的value又是一个Map<String,Map>



命令示例：

```
127.0.0.1:6379> hset student name "smallming"
(integer) 1
127.0.0.1:6379> hset student age 18
(integer) 1
127.0.0.1:6379> hgetall student
1) "name"
2) "smallming"
3) "age"
4) "18"
```

列表（List）

List类型表示多个有序可重复的值。和Java中List类似。

尤其是和Java中的LinkedList类似。因为Redis中的List就是一个双向链表。

命令示例：

```
127.0.0.1:6379> rpush hovers "study"
(integer) 1
127.0.0.1:6379> rpush hovers "play"
(integer) 2
127.0.0.1:6379> lrange hovers 0 -1
1) "study"
2) "play"
```

集合(Set)

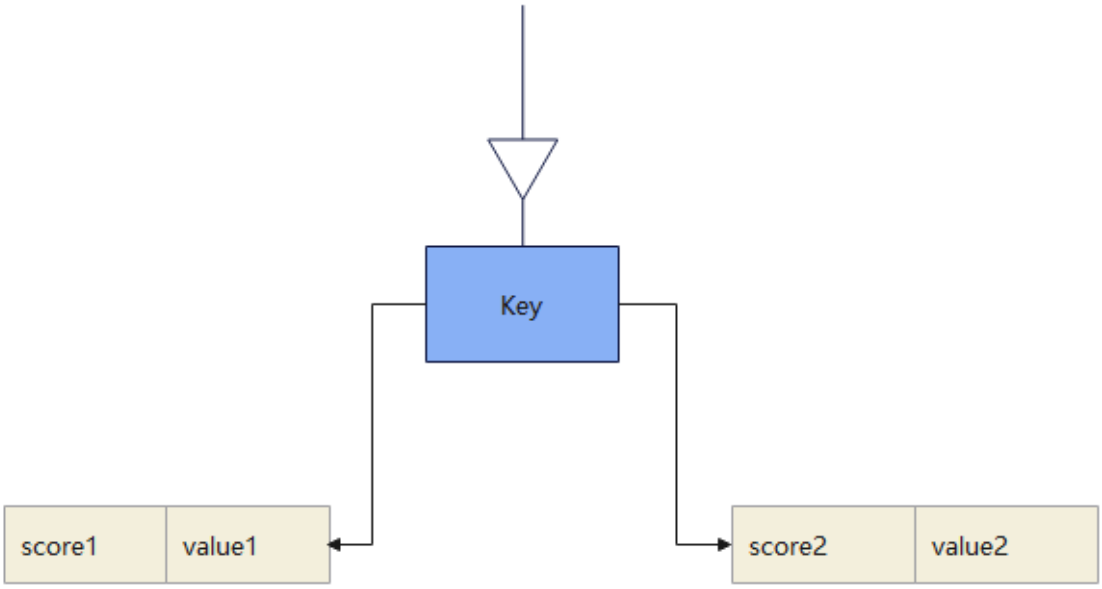
set集合类型可以存储多个值，这些值存放顺序和添加顺序没有关系。如果新增的值已经存在返回值为0，如果不存在返回1.

命令示例：

```
127.0.0.1:6379> sadd record "excellent"
(integer) 1
127.0.0.1:6379> sadd record "good"
(integer) 1
127.0.0.1:6379> sadd record "excellent"
(integer) 0
127.0.0.1:6379> smembers record
1) "good"
2) "excellent"
```

有序集合（Sorted Set）

有序集合中每个value都有一个分数（score），根据score进行排序。



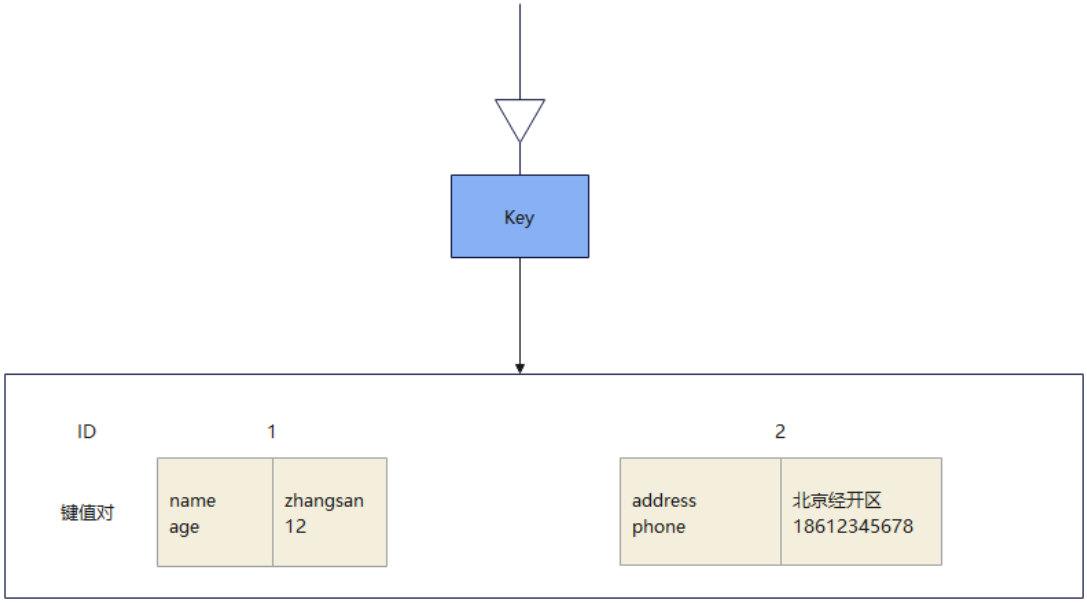
命令示例：

```
127.0.0.1:6379> zadd result 99 chinese 100 maths
(integer) 2
127.0.0.1:6379> zrange result 0 -1 withscores
1) "chinese"
2) "99"
3) "maths"
4) "100"
```

流类型（Stream）

Stream类型从Redis 5 出现的。里面可以包含任意个流元素的有序队列。每个元素里面必须包含一个ID和任意多个键值对。这个感觉有点像包含主键的数据库表中每行数据。

使用stream类型时一定要注意，ID值可以自动生成，但是后添加的ID必须大于与存在所有数据中最大ID的值。



命令示例：

如果希望自动生成ID，按照xadd语法，ID位置使用星号。

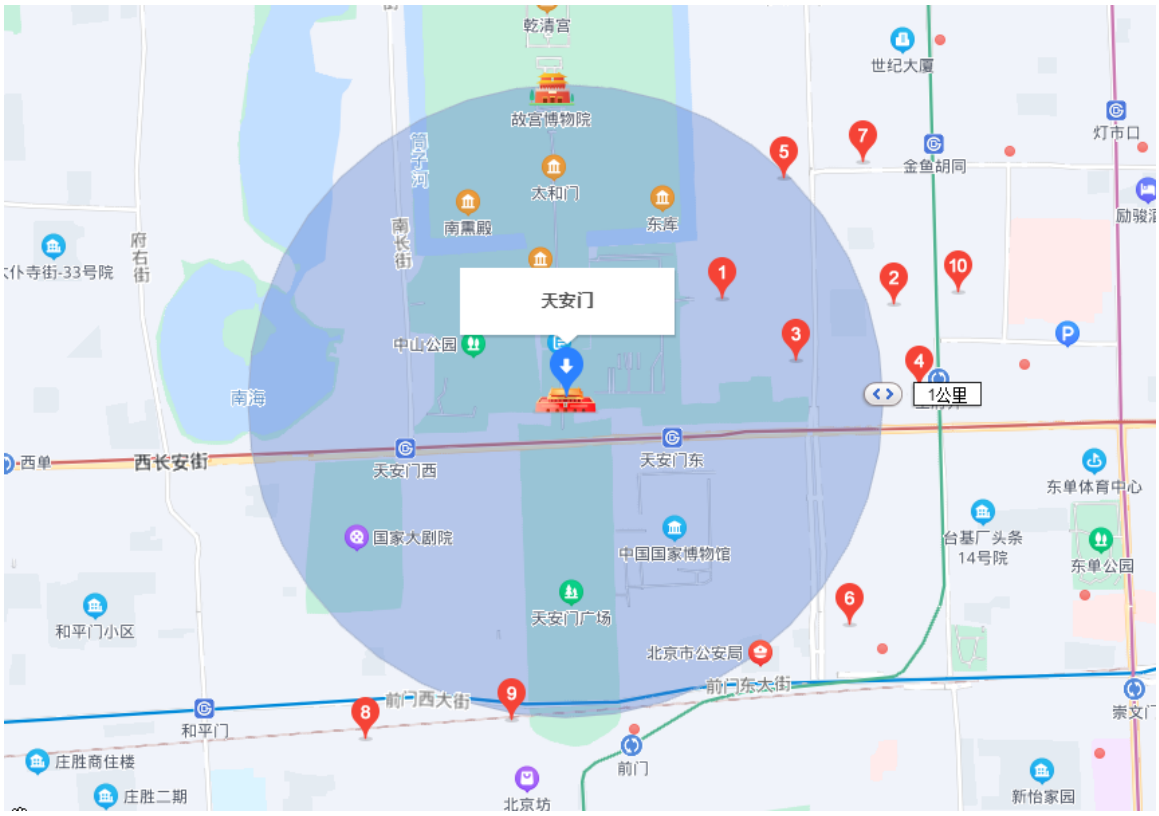
如果希望查询全部，- +表示最小ID到最大ID

```
127.0.0.1:6379> xadd teacher 1 name "zhangsan" age 18
"1-0"
127.0.0.1:6379> xadd teacher 2 address "beijing" phone "18612345678"
"2-0"
127.0.0.1:6379> xadd teacher * name "lisi" address "shanghai"
"1653198830422-0"
127.0.0.1:6379> xrange teacher - +
1) 1) "1-0"
   2) 1) "name"
      2) "zhangsan"
      3) "age"
      4) "18"
2) 1) "2-0"
   2) 1) "address"
      2) "beijing"
      3) "phone"
      4) "18612345678"
3) 1) "1653198830422-0"
   2) 1) "name"
      2) "lisi"
      3) "address"
      4) "shanghai"
```

Geospatial 地理位置

地理位置类型允许存储地理位置坐标，然后可以通过地理位置坐标来搜索。

这个效果类似我们平时所使用的百度地图或高德地图，搜索附近酒店或充电桩等。



命令示例：

通过geoadd录入两个地理位置

geodist查看两个位置的距离

georadius 以某个坐标点为原点，查询半径内的地理位置

```
127.0.0.1:6379> geoadd location 13 13 a 14 14 b
(integer) 2
127.0.0.1:6379> geodist location a b
"155138.7665"
127.0.0.1:6379> georadius location 13 13 100 km
1) "a"
127.0.0.1:6379> georadius location 13 13 160 km
1) "a"
2) "b"
```

HyperLogLog 超级日志

HyperLogLog属于一种随机概率算法。它可以用一种特别少的内存空间来提供一个集合元素的近似值。

在Redis实现中，每个密钥仅使用12KB进行计数，标准错误为0.81%，并且可以计数的项目数量没有限制。

可以使用超级日志实现一些唯一数字统计：

- 网站流量中IP数量统计。每个IP可能访问多次，但是每个访问者都有自己唯一的IP。
- 用户搜索过的不同内容次数。

```
127.0.0.1:6379> pfadd members 1
(integer) 1
127.0.0.1:6379> pfadd members 2
(integer) 1
127.0.0.1:6379> pfadd members 1
(integer) 0
127.0.0.1:6379> pfcount members
(integer) 2
```

bitmaps 位图

bitmaps属于对字符串类型（String）的扩展，可以将字符串作为一个位向量（bit vector：只包含0或1的数组）。

通俗理解：有一个长度为N的数组，数组的key为字符串值。数组每个元素取值只能是0或1.可以通过offset（偏移量，取值0到2的32次方）来设置数组某个脚标的值。

所以位向量能出现的场景，Redis的bitmaps就能出现。

具体场景：

有1000台客户端服务器，每个客户端服务器都有自己的编号，分别是0~999.当客户端向服务器发送消息服务器就记录一下。最后可以统计出来服务器是否发送过消息。

```
127.0.0.1:6379> setbit client 99 1
(integer) 0
127.0.0.1:6379> setbit client 123 1
(integer) 0
127.0.0.1:6379> getbit client 123
(integer) 1
127.0.0.1:6379> getbit client 456
(integer) 0
```

bitfields 位字段

bitfileds 主要是为了实现对数字做递增的，重点是数字的位数支持设置，可以通过i16、u8来设置数字所占的bit位数。这点类似Java中提供short、int、long是一个道理。

可以通过位字段来做计数统计。

命令示例：

一款游戏。需要统计用户杀怪的数量和获取金币的数量。

设定游戏和杀怪数量可能比较大，所以使用u32存储。

金币的offset为123、杀怪数量的offset为456

```
127.0.0.1:6379> bitfield player set u32 123 0
1) (integer) 0
127.0.0.1:6379> bitfield player set u32 456 0
1) (integer) 0
127.0.0.1:6379> bitfield player incrby u32 123 15
1) (integer) 15
127.0.0.1:6379> bitfield player incrby u32 456 2
1) (integer) 2
127.0.0.1:6379> bitfield player get u32 123 get u32 456
1) (integer) 15
2) (integer) 2
```