Redis Basic 基本概念 - 03课

1.总结什么是set以及什么是sorted set，并完成对set以及sorted set的增删改查（查需要至少4种方式，比如列表，单个节点等）

```
127.0.0.1:6380> sadd s01 s1 s2 s3
(integer) 3
127.0.0.1:6380>
127.0.0.1:6380> smembers s01
1) "s3"
2) "s2"
3) "s1"
127.0.0.1:6380>
127.0.0.1:6380> spop s01
"s3"
127.0.0.1:6380> smembers s01
1) "s2"
2) "s1"
127.0.0.1:6380>
127.0.0.1:6380> sadd s01 s3
(integer) 1
127.0.0.1:6380> smembers s01
1) "s3"
2) "s2"
3) "s1"
127.0.0.1:6380> srem s01 s2
(integer) 1
127.0.0.1:6380>
127.0.0.1:6380> smembers s01
1) "s3"
2) "s1"
127.0.0.1:6380>
127.0.0.1:6380> smembers s01
1) "s3"
2) "s1"
127.0.0.1:6380> smove s01 s02 s1
(integer) 1
127.0.0.1:6380> smembers s01
1) "s3"
127.0.0.1:6380> smembers s02
1) "s1"
127.0.0.1:6380>
127.0.0.1:6380> smembers s01
1) "s2"
2) "s3"
127.0.0.1:6380> smembers s02
1) "s3"
2) "s1"
127.0.0.1:6380> sdiff s01 s02
1) "s2"
127.0.0.1:6380>
127.0.0.1:6380> sinter s01 s02
1) "s3"
127.0.0.1:6380>
127.0.0.1:6380> sunion s01 s02
1) "s1"
2) "s3"
3) "s2"
```

```
127.0.0.1:6380>


127.0.0.1:6380> zadd ss01 1 11 2 22 3 31 4 30
(integer) 4
127.0.0.1:6380> zrange ss01 0 10
1) "11"
2) "22"
3) "31"
4) "30"
127.0.0.1:6380>

127.0.0.1:6380> zrange ss01 0 10
1) "11"
2) "22"
3) "31"
4) "30"
5) "33"
127.0.0.1:6380>
127.0.0.1:6380> zadd ss01 XX 4 33
(integer) 0
127.0.0.1:6380>
127.0.0.1:6380> zrange ss01 0 10
1) "11"
2) "22"
3) "31"
4) "30"
5) "33"
127.0.0.1:6380>
127.0.0.1:6380> zadd ss01 NX 3 32
(integer) 1
127.0.0.1:6380> zrange ss01 0 10
1) "11"
2) "22"
3) "31"
4) "32"
5) "30"
6) "33"
127.0.0.1:6380>
127.0.0.1:6380> zscan ss01 0
1) "0"
2)  1) "11"
    2) "1"
    3) "22"
    4) "2"
    5) "31"
    6) "3"
    7) "32"
    8) "3"
    9) "30"
   10) "4"
   11) "33"
   12) "4"
127.0.0.1:6380>

127.0.0.1:6380> zadd ss02 1 10 2 20 3 30 4 40
(integer) 4
127.0.0.1:6380>
```

```
127.0.0.1:6380> zrange ss02 0 100
1) "10"
2) "20"
3) "30"
4) "40"
127.0.0.1:6380>
127.0.0.1:6380> zrevrangebyscore ss02 100 0
1) "40"
2) "30"
3) "20"
4) "10"
127.0.0.1:6380>
127.0.0.1:6380> zremrangebyscore ss02 2 3
(integer) 2
127.0.0.1:6380> zrange ss02 0 100
1) "10"
2) "40"
127.0.0.1:6380>
```

2.总结redis的事务特征，并且实际操作事务的提交 丢弃以及乐观锁

MULTI: 开启事务
EXEC： 执行事务所有命令，类似commit
DISCARD：类似回滚，不执行MULTI后的命令
WATCH：如果观察的key发生变化，就不执行事务
UNWATCH: 取消乐观锁

Redis事务特征：单线程，使用乐观锁
A: 要么全部执行要不不执行，如果发现key变化了，事务不提交
C：单个线程，其他命令修改不影响当前执行，只是没有回滚机制，一个命令失败了，下面的命令还会继续进行
I: 单线程，隔离线强
D: 持久性 依赖于redis持久化机制，需要设置相关参数 appendfsync（always）

丢弃：

```
127.0.0.1:6380> multi
OK
127.0.0.1:6380> set msg01 'tr01'
QUEUED
127.0.0.1:6380> get msg01
QUEUED
127.0.0.1:6380> get
(error) ERR wrong number of arguments for 'get' command
127.0.0.1:6380> exec
(error) EXECABORT Transaction discarded because of previous errors.
127.0.0.1:6380> get msg01
(nil)
127.0.0.1:6380>
```

正常：

```
127.0.0.1:6380> multi
OK
127.0.0.1:6380> set msg01 'tr02'
QUEUED
127.0.0.1:6380> get msg01
QUEUED
127.0.0.1:6380> exec
1) OK
2) "tr02"
```

```
127.0.0.1:6380> get msg01
"tr02"
127.0.0.1:6380>

127.0.0.1:6380> set tkey 0001
OK
127.0.0.1:6380> get tkey
"0001"
127.0.0.1:6380>
```

窗口1

```
127.0.0.1:6380> multi
OK
127.0.0.1:6380> set tkey 00002
QUEUED
127.0.0.1:6380> exec
1) OK
127.0.0.1:6380> get tkey
"00002"
127.0.0.1:6380>
```

窗口2

```
127.0.0.1:6380> get tkey
"0001"
127.0.0.1:6380> set tkey 0002
OK
127.0.0.1:6380> get tkey
"00002" ----> 并不是0002
127.0.0.1:6380>
```

窗口1：

```
127.0.0.1:6380> watch tkey
OK
127.0.0.1:6380> multi
OK
127.0.0.1:6380> set tkey 0003
QUEUED
127.0.0.1:6380> get tkey
QUEUED
127.0.0.1:6380> get tkey
QUEUED
127.0.0.1:6380> exec
(nil)
127.0.0.1:6380> get tkey
"0004" ----> 是窗口2的结果
127.0.0.1:6380>
```

窗口2：

```
127.0.0.1:6380> get tkey
"00002"
127.0.0.1:6380> set tkey 0004
OK
127.0.0.1:6380> get tkey
"0004"
127.0.0.1:6380>
```