小象科技 ChinaHadoop.cn

Redis 从入门到精通

黄健宏（huangz）

散列

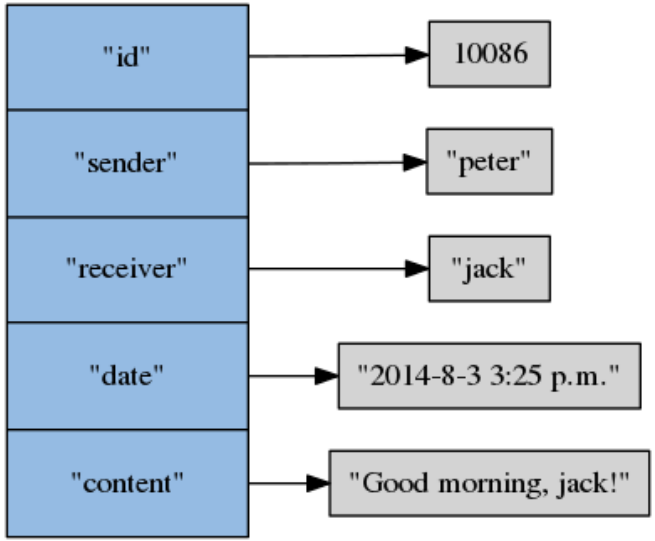
储存多个域值对。

散列(hash)

一个散列由多个域 值对(field-value pair)组成，散列的域和值都可以是文字、整数、浮点数或者二 进制数据。

同一个散列里面的每个域必 须是独一无二、各不相同的，而域的值则没有这一要求，换句话说，不同域的值可以是重复的。

通过命令，用户可以对散列执行设置域值对、获取域的值、检查域是否存在等操作，也可以 让 Redis 返回散列包含的所有域、所有 值或者所有域 值对。



一个储存了消息(message)信息的散列。

基本操作

关联域值对、获取域的值、检查域是否存在、获取域值对的数量，等等。

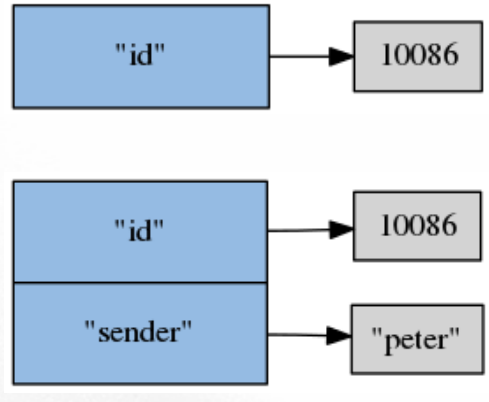
关联域值对

HSET key field value

在散列键 key 中关联给定的域值对 field 和 value 。
如果域 field 之前没有关联值，那么命令返回 1 ；
如果域 field 已经有关联值，那么命令用新值覆盖旧值，并返回 0 。

复杂度为 O(1) 。

```
redis> HSET message "id" 10086
(integer) 1
redis> HSET message "sender" "peter"
(integer) 1
redis> HSET message "receiver" "jack"
(integer) 1
```



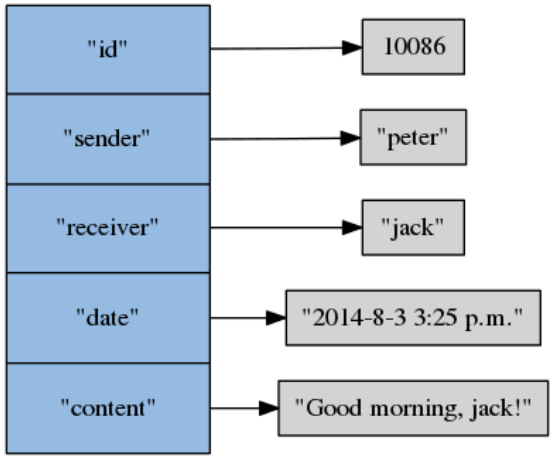
获取域关联的值

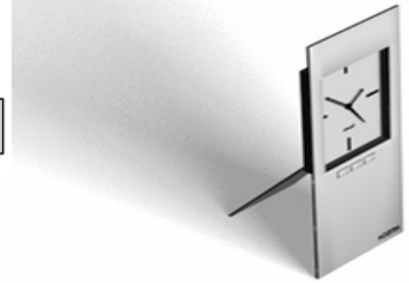
HGET key field

返回散列键 key 中，域 field 所关联的值。如果域 field 没有关联值，那么返回 nil 。

复杂度为 O(1) 。

```
redis> HGET message "id"
"10086"
redis> HGET message "sender"
"peter"
redis> HGET message "content"
"Good morning, jack!"
redis> HGET message "NotExistsField"
(nil)
```





仅当域不存在时，关联域值对

小象科技 ChinaHadoop.cn

HSETNX key field value

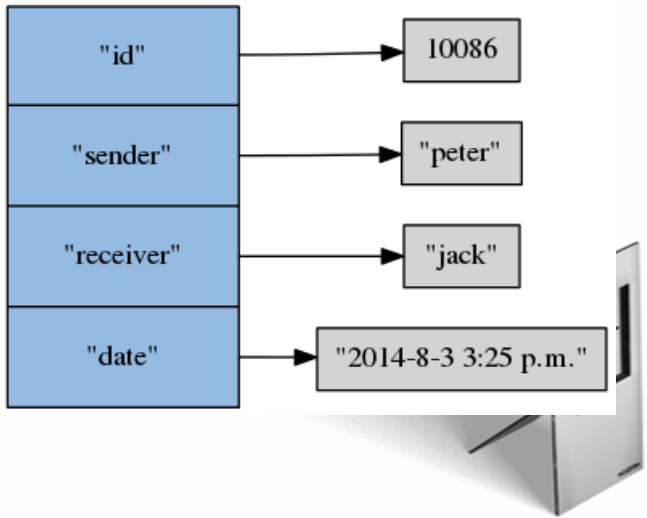
如果散列键 key 中，域 field 不存在（也即是，还没有与之相关联的值），那么关联给定的域值对 field 和 value 。

如果域 field 已经有与之相关联的值，那么命令不做动作。

复杂度为 O(1) 。

redis> HSETNX message "content" "Good morning, jack!"
(integer) 1

redis> HSETNX message "content" "Good morning, jack!"
(integer) 0



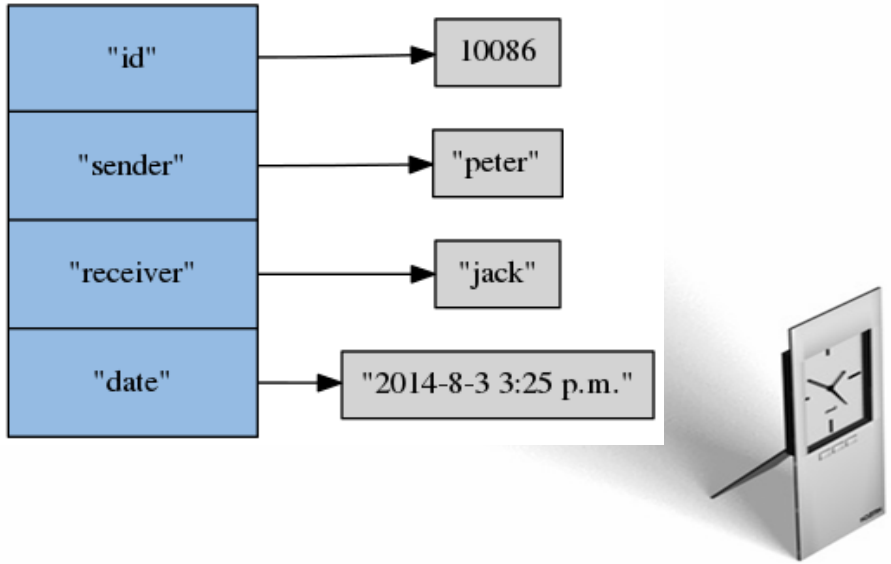
检查域是否存在

HEXISTS key field

查看散列键 key 中，给定域 field 是否存在：存在返回 1，不存在返回 0 。

复杂度为 O(1) 。

```
redis> HEXISTS message "id"
(integer) 1
redis> HEXISTS message "sender"
(integer) 1
redis> HEXISTS message "content"
(integer) 0
redis> HEXISTS message "NotExistsField"
(integer) 0
```



删除给定的域值对

HDEL key field [field ...]

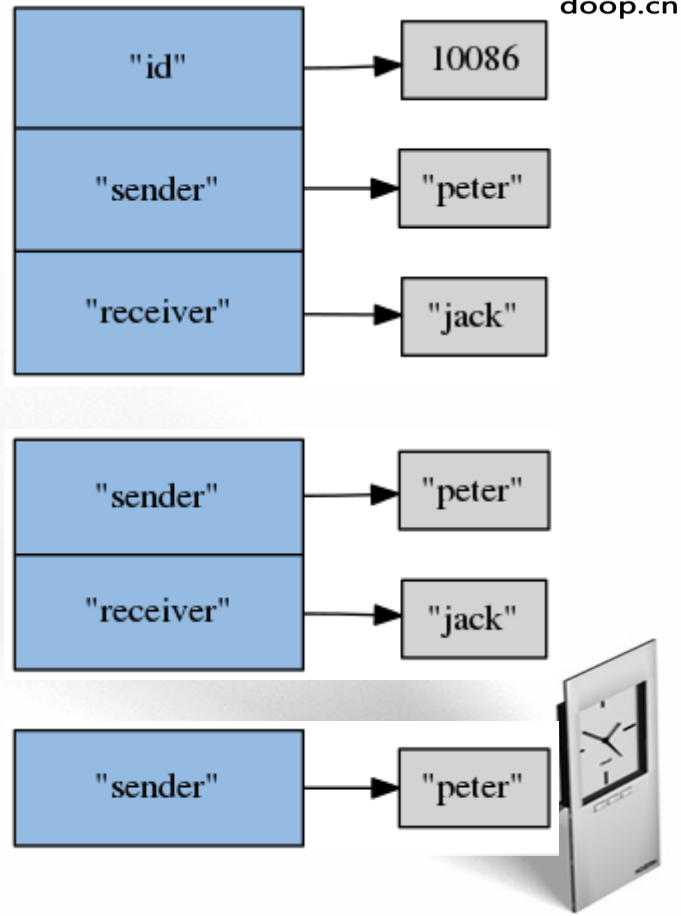
删除散列键 key 中的一个或多个指定域，以及那些域的 值。不存在的域将被忽略。命令返回被成功 删除的域值对数量。

复杂度为 O(N)，N 为被删除的域值对数量。

redis> HDEL message "id"
(integer) 1

redis> HDEL message "receiver"
(integer) 1

redis> HDEL message "sender"
(integer) 1



获取散列包含的键值对数量

小象科技 ChinaHadoop.cn

HLEN key

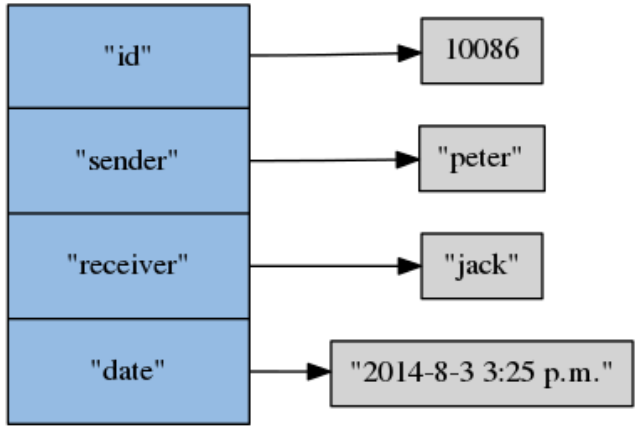
返回散列键 key 包含的域值对数量。

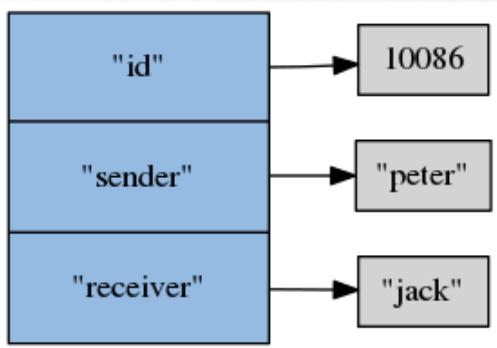
复杂度为 O(1) 。

redis> HLEN message
(integer) 4

redis> HDEL message "date"
(integer) 1

redis> HLEN message
(integer) 3





批量操作

一次对多个域、多个值或者多个域值对进行操作。

一次设置或获取散列中的多个域值对

小象科技 ChinaHadoop.cn

命令	效果	复杂度
HMSET key field value [field value ...]	在散列键 key 中关联多个域值对，相当于同时执行多个 HSET 。	O(N)，N 为输入的域值对数量。
HMGET key field [field ...]	返回散列键 key 中，一个或多个域的值，相当于同时执行多个 HGET 。	O(N)， N 为输入的域数量。

```
redis> HMSET message "id" 10086 "sender" "peter" "receiver" "jack"
OK

redis> HMGET message "id" "sender" "receiver"
1) "10086"
2) "peter"
3) "jack"
```

获取散列包含的所有域、值、或者域值对

小象科技 ChinaHadoop.cn

命令	效果	复杂度
HKEYS key	返回散列键 key 包含的所有域。	O(N)，N 为被返回域的数量。
HVALS key	返回散列键 key 中，所有域的值。	O(N)，N 为被返回值的数量。
HGETALL key	返回散列键 key 包含的所有域值对。	O(N)，N 为被返回域值对的数量。

为什么命令叫 HKEYS 而不是 HFIELDS ？

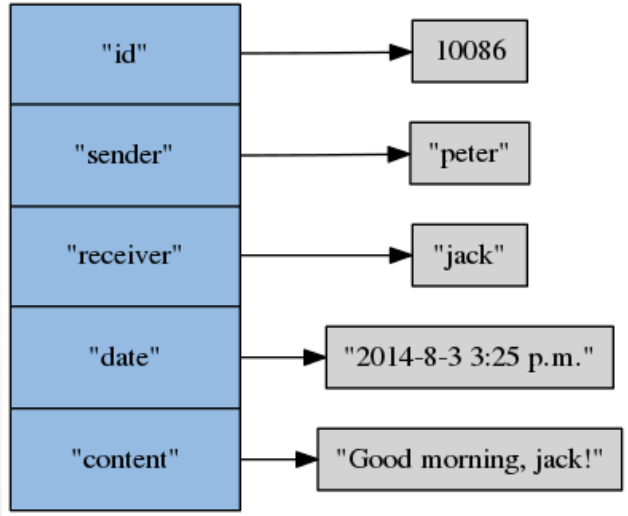
对于散列来说，key 和 field 表示的是同一个意思，并且 key 比 field 更容易拼写，所以 Redis 选择使用 HKEYS 来做命令的名字，而不是 HFIELDS 。

HKEYS、HVALS 和 HGETALL 示例


```
redis> HKEYS message
1) "id"
2) "sender"
3) "receiver"
4) "date"
5) "content"
```

```
redis> HVALS message
1) "10086"
2) "peter"
3) "jack"
4) "2014-8-3 3:25 p.m."
5) "Good morning, jack!"
```

```
redis> HGETALL message
1) "id"            # 域
2) "10086"     # 值
3) "sender"    # 域
4) "peter"       # 值
5) "receiver"
6) "jack"
7) "date"
8) "2014-8-3 3:25 p.m."
9) "content"
10) "Good morning, jack!"
```



数字操作

和字符串键的值一样，在散列里面，域的值也可以被解释为数字，并执行相应的数字操作。

对域的值执行自增操作

小象科技 ChinaHadoop.cn

命令	作用	复杂度
HINCRBY key field increment	为散列键 key 中， 域 field 的值加上整数增量 increment 。	O(1)
HINCRBYFLOAT key field increment	为散列键 key 中， 域 field 的值加上浮点数增量 increment 。	O(1)

虽然 Redis 没有提供与以上两个命令相匹配的HDECRBY 命令和 HDECRBYFLOAT 命令，但我们同样可以通过将 increment 设为负数来达到做减法的效果。

```
redis> HINCRBY numbers x 100    # 域不存在，先将值初始化为 0 ，然后再执行 HINCRBY 操作
(integer) 100
redis> HINCRBY numbers x -50    # 传入负值，做减法
(integer) 50
redis> HINCRBYFLOAT numbers x 3.14    # 浮点数计算
"53.14"
```

散列键和字符串键

有一种似曾相识的感觉.....

效果类似的命令

小象科技 ChinaHadoop.cn

散列命令	字符串/数据库命令
HSET	SET
HGET	GET
HSETNX	SETNX
HDEL	DEL（删除一个键，不仅限于字符串键）
HMSET	MSET
HMGET	MGET
HINCRBY	INCRBY
HINCRBYFLOAT	INCRBYFLOAT
HEXISTS	EXISTS（检查一个键是否存在，不仅限于字符串键）

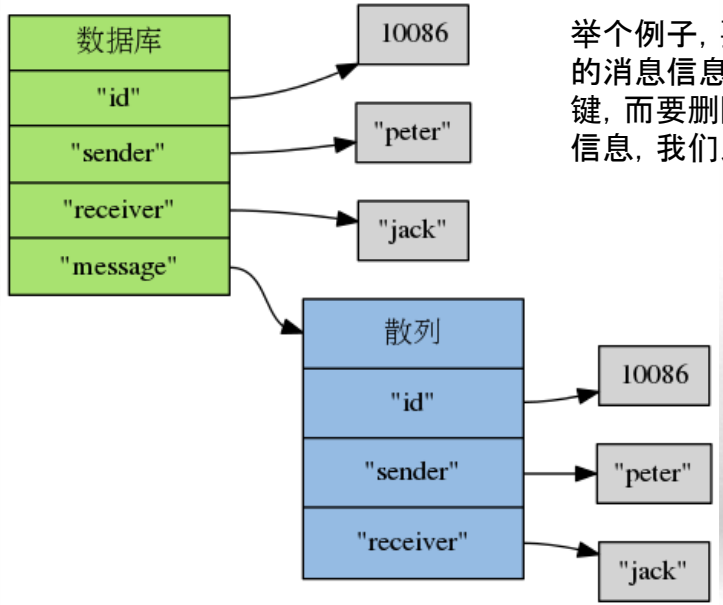
如果散列键能做的事情，字符串键也能做，那么我们为什么不直接使用字符串键呢？

使用散列的好处(1):将数据放到同一个地方

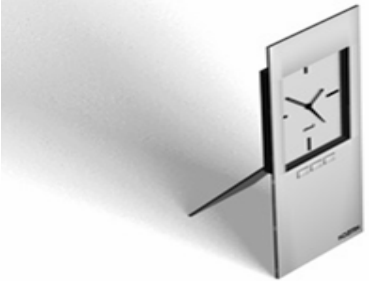
散列可以让我们将一些相关的信息储存在同一个地方，而不是直接分散地储存在整个数据库里面，这不仅方便了数据管理，还可以尽量避免误操作发生。

```
redis> MSET "id" 10086
                "sender" "peter"
                "receiver" "jack"
OK

redis> HMSET message
                "id" 10086
                "sender" "peter"
                "receiver" "jack"
OK
```



举个例子，要删除字符串键记录的消息信息，我们需要输入三个键，而要删除散列键储存的消息信息，我们只要输入一个键。



使用散列的好处(2):避免键名冲突

在介绍字符串键的时候，我们说过，可以在命名键的时候，使用分割符来避免命名冲突，但更好的办法是直接使用散列键来储存键值对数据。

举个例子，如果我们需要储存多条消息，那么我们可能会按照格式 message::<id>::<field> 的方式来创建字符串键并储存数据，比如使用 message::12345::receiver 键来储存 id 为 12345 的消息的接收者，使用 message::10086::sender 键来储存 id 为 10086 的发送者，诸如此类。

但更好的办法是直接使用 message:<id> 散列键来表示消息信息，并将与消息有关的各项信息储存到散列的各个域里面，比如创建 message::12345 散列，并将该消息的各项信息储存分别储存到这个散列的 id 域、receiver 域和 sender 域里面。

这保证了数据库中每个键的作用都是固定的、单一的，储存的信息都是被隔离的，从而最大限度地避免键名冲突。

对比

以下两个数据库分别以字符串键和散列键两种形式保存了相同的信息，但是很明 显，使用字符串键的数据库创建的键数量多，而使用散列 键的数据库创建的键数量则少。

随着域数量的增加，使用散列会比使用字符串少 创建很多数据库键。

数据库
"message::10086::id"
"message::10086::sender"
"message::10086::receiver"
"message::12345::id"
"message::12345::sender"
"message::12345::receiver"
"message::15000::id"
"message::15000::sender "
"message::15000::receiver"

数据库
"message::10086"
"message::12345"
"message::15000"

使用散列的好处(3):减少内存占用

在一般情况下，保存相同数量的 键值对信息，使用散列 键比使用字符串 键更节约内存。

因为在数据库里面创建的每个键都带有数据库附加的管理信息(比如 这个键的类型、最后一次被 访问的时间等等)，所以数据 库里面的键越多，服 务器在储存附加管理信息方面耗 费的内存就越多，花在管理数据库键上的 CPU 也会越多。

除此之外，当散列包含的域 值对数量比较少的时候，Redis 会自动使用一种占用内存非常少的数据 结构来做散列的底 层实现，在散列的数量比 较多的时候，这一措施对减少内存有很大的帮助。

结论

只要有可能的话，就尽量使用散列键而不是字符串键来储存键值对数据，因为散列键管理方便、能够避免键名冲突、并且还能够节约内存。

一些没办法使用散列键来代替字符串键的情况：

1. 使用二进制位操作命令：因为 Redis 目前支持对字符串键进行 SETBIT、GETBIT、BITOP 等操作，如果你想使用这些操作，那么只能使用字符串键。
2. 使用过期功能：Redis 的键过期功能目前只能对键进行过期操作，而不能对散列的域进行过期操作，因此如果你要对键值对数据使用过期功能的话，那么只能把键值对储存在字符串里面。关于过期键的详细信息会在之后介绍。

示例:使用散列重新实现计数器

API	效果	实现
Counter(name, client)	设置计数器的名字以及客户端。	
Counter.incr()	将计数器的值增一，然后返回计数器的值。	调用 HINCRBY 命令。
Counter.get()	返回计数器当前的值。	调用 HGET 命令。
Counter.reset(n=0)	将计数器的值重置为 n，默认重置为 0 。	调用 HGET 命令和 HSET 命令。

```
c = Counter('page-counter', redis_client)  # 创建一个名为 page-counter 的计数器
c.incr()  # => 1
c.incr()  # => 2
```

这个计数器的功能和 API，跟之前用字符串键实现的计数器完全一样，不同之处在于，这个实现会将所有计数器都储存在同一个散列里面，一个域值对就是一个计数器。

这个计数器的完整代码请查看 hash_counter.py 文件。

复习

回顾一下前面对散列键的介绍。

复习(1/2)

一个散列由多个域 值对(field-value pair)组成，散列的键和值都可以是文字、整数、浮点数或者二 进制数据。

同一个散列里面的各个域必 须是独一无二的，但不同域的 值可以是重复的。

尽量使用散列 键而不是字符串 键来储存键值对数据，因为散列键管理方便、能 够避免键名冲突、并且还能够节约内存。

复习(2/2)

基本操作	批量操作	数字操作
HSET 和 HGET HSETNX HEXISTS HDEL HLEN	HMSET 和 HMGET HGETALL HKEYS 和 HVALS	HINCRBY HINCRBYFLOAT