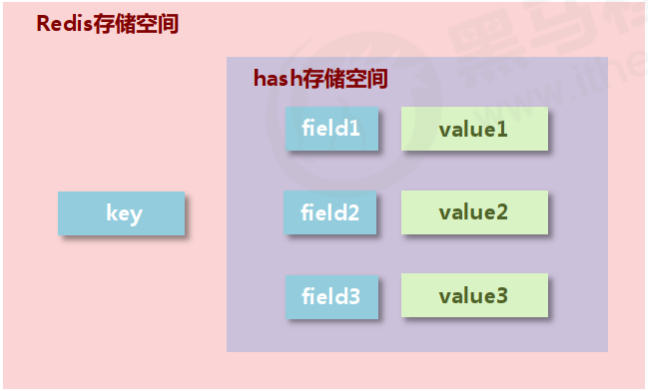
Hash类型

- 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
- 需要的存储结构：一个存储空间保存多个键值对数据
- 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

- 如果field数量较少，存储结构优化为类数组结构
- 如果field数量较多，存储结构使用HashMap结构

- 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 [field2]
```

- 添加/修改多个数据

```
hmset key field1 value1 field2 value2 …
```

- 获取多个数据

```
hmget key field1 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
```

- 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
```

Hash类型数据操作的注意事项

- hash类型下的value只能存储字符串，不允许存储其他数据类型，不存在嵌套现象。如果数据未获取到，对应的值为（nil）
- 每个hash可以存储2^{32}-1个键值对
- hash类型十分贴近对象的数据存储类型，并且可以灵活添加删除对象属性。但hash设计初衷不是为了存储大量对象而设计的，切忌不可滥用，更不可以将hash作为对象列表使用
- 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的瓶颈。