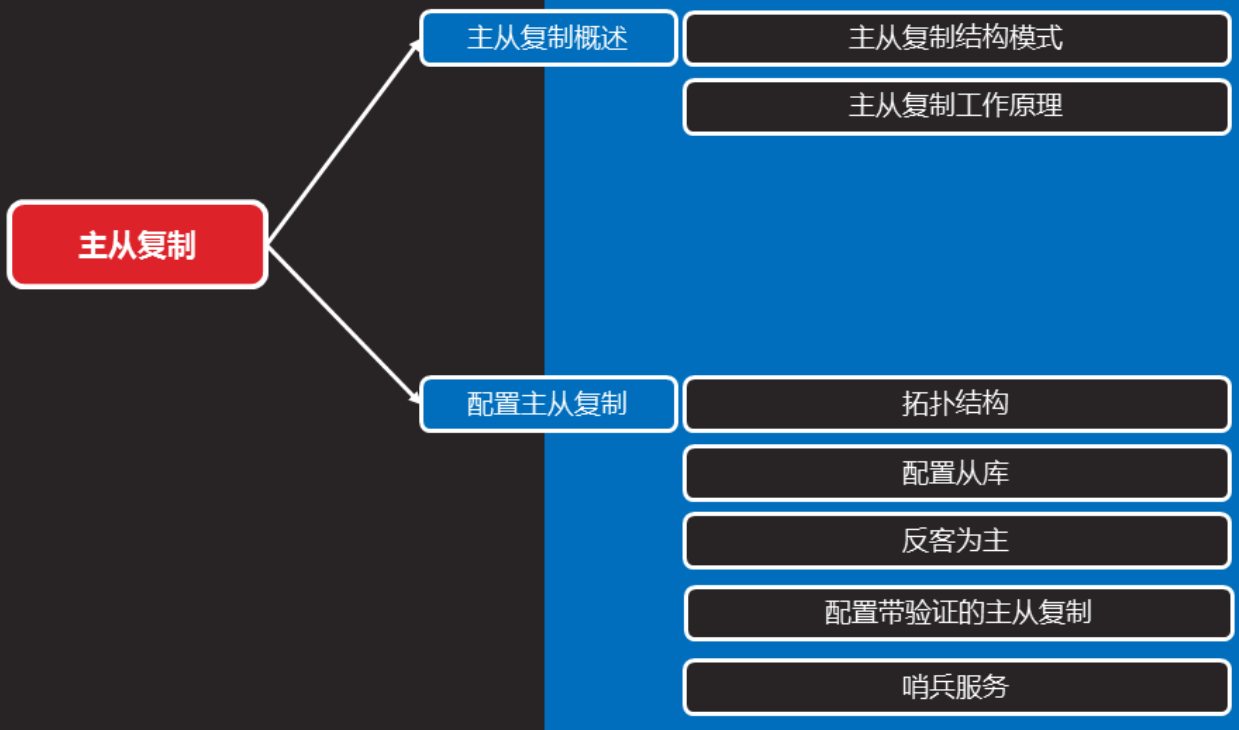
NOSQL DAY03

# 内容

| | | |
|---|---|---|
| 上午 | 09:00 ~ 09:30 | 作业讲解和回顾 |
| | 09:30 ~ 10:20 | 主从复制 |
| | 10:30 ~ 11:20 | |
| | 11:30 ~ 12:00 | 持久化 |
| 下午 | 14:00 ~ 14:50 | |
| | 15:00 ~ 15:50 | 数据类型 |
| | 16:10 ~ 17:00 | |
| | 17:10 ~ 18:00 | 总结和答疑 |

## 主从复制

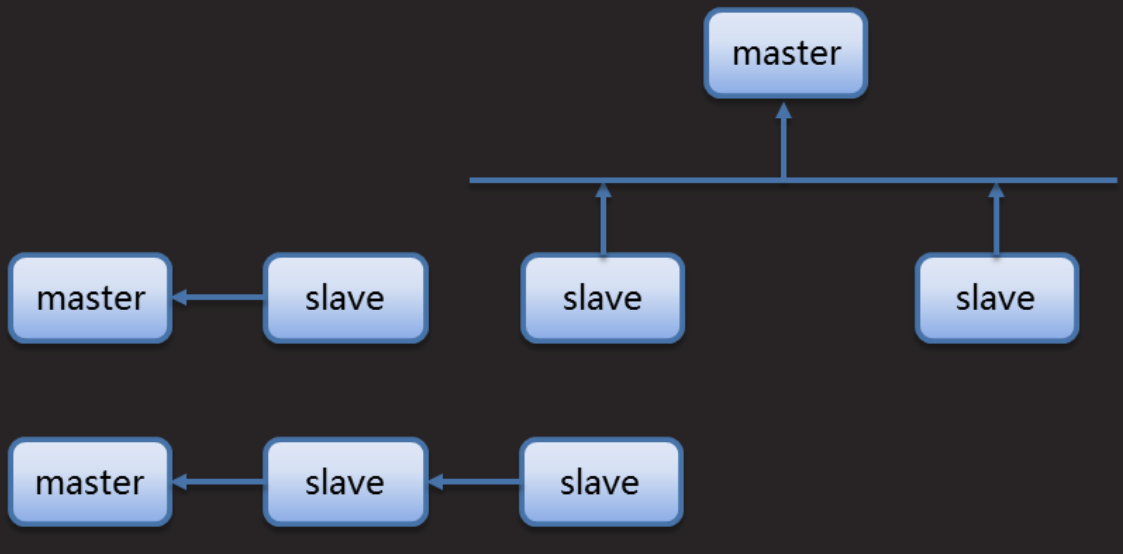


# 主从复制概述

## 主从复制结构模式

- 结构模式
  - 一主一从、一主多从、主从从

知识讲解



# 主从复制工作原理

知识讲解

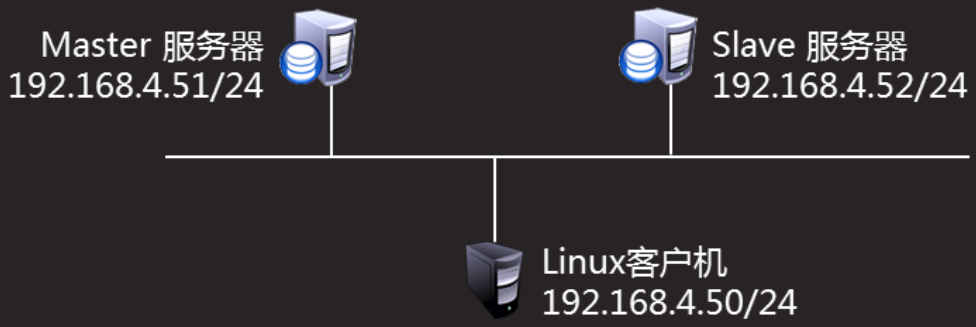
- 工作原理
  - slave向master发送sync命令
  - master启动后台存盘进程，并收集所有修改数据命令
  - master完成后台存盘后，传送整个数据文件到slave
  - slave接收数据文件，加载到内存中完成首次完全同步
  - 后续有新数据产生时，master继续收集数据修改命令依次传给slave，完成同步

# 配置主从复制

# 拓扑结构

- 一主一从

知识讲解




# 配置从库

- 配置从库
  - redis服务运行后，默认都是master 服务器

知识讲解

```
192.168.4.52:6379>  info  replication  //查看复制信息
# Replication
role:                  //角色
master_host:           //主库ip地址
master_port:           //主库端口号
master_link_status: //与主库连接状态

[root@redis52 ~]# redis-cli  -h 192.168.4.52
192.168.4.52:6379>  slaveof  192.168.4.51  6351  //命令行配置
OK
```

slaveof 主库IP地址 端口号

```
]# vim  /etc/redis/6379.conf
   slaveof  192.168.4.51 6351          //永久配置
```

# 反客为主

知识讲解

- 反客为主
  - 将从库恢复为主库

```
[root@redis52 ~]# redis-cli  -h 192.168.4.52

192.168.4.52:6379> slave  no  one      //命令行临时设置
OK


]# vim   /etc/redis/6379.conf
  #slaveof     192.168.4.51  6351  //永久设置
```

# 案例1：主从复制

课堂练习

具体要求如下：

- 将主机192.168.4.51配置为主库
- 将主机192.168.4.52配置为192.168.4.51的从库
- 测试配置

# 配置带验证的主从复制

知识讲解

- 配置master
  - 设置连接密码 ,重启服务

```
]# sed  -n  '501p'  /etc/redis/6379.conf
    requirepass  123456                    //定义连接密码
```

- 配置slave
  - 设置连接密码 ，重启服务

```
]#  sed  -n  '282p;289p'  /etc/redis/6379.conf
   masterauth  123456                      //主库密码
```

# 案例2：配置带验证的主从复制

课堂练习

具体要求如下：

- 基于案例1的配置
- 设置主库192.168.4.51 连接密码123456
- 配置从库192.168.4.52
- 测试配置

# 哨兵服务

知识讲解

- 哨兵服务介绍
  - 监视master服务器
  - 发现master宕机后，将从库升级为主库
  - 主配置文件 sentinel.conf
  - 模板文件：redis-4.0.8/sentinel.conf


# 哨兵服务(续1)

知识讲解

- 配置哨兵服务
  - 创建主配置文件
  - 启动服务

```
]# vim  /etc/sentinel.conf
    sentinel  monitor  server51  192.168.4.51  6351  1
    bind  0.0.0.0                                  //服务地址
    sentinel auth-pass  server51  123456  //连接服务密码

]# redis-sentinel  /etc/sentinel.conf        //启动服务
```

**sentinel monitor 主机名 ip地址 端口 票数**
主机名：自定义
IP地址：master主机的IP地址
端　口：master主机 redis服务使用的端口
票　数： 发现主库宕机的哨兵服务器个数

# 案例3：哨兵服务

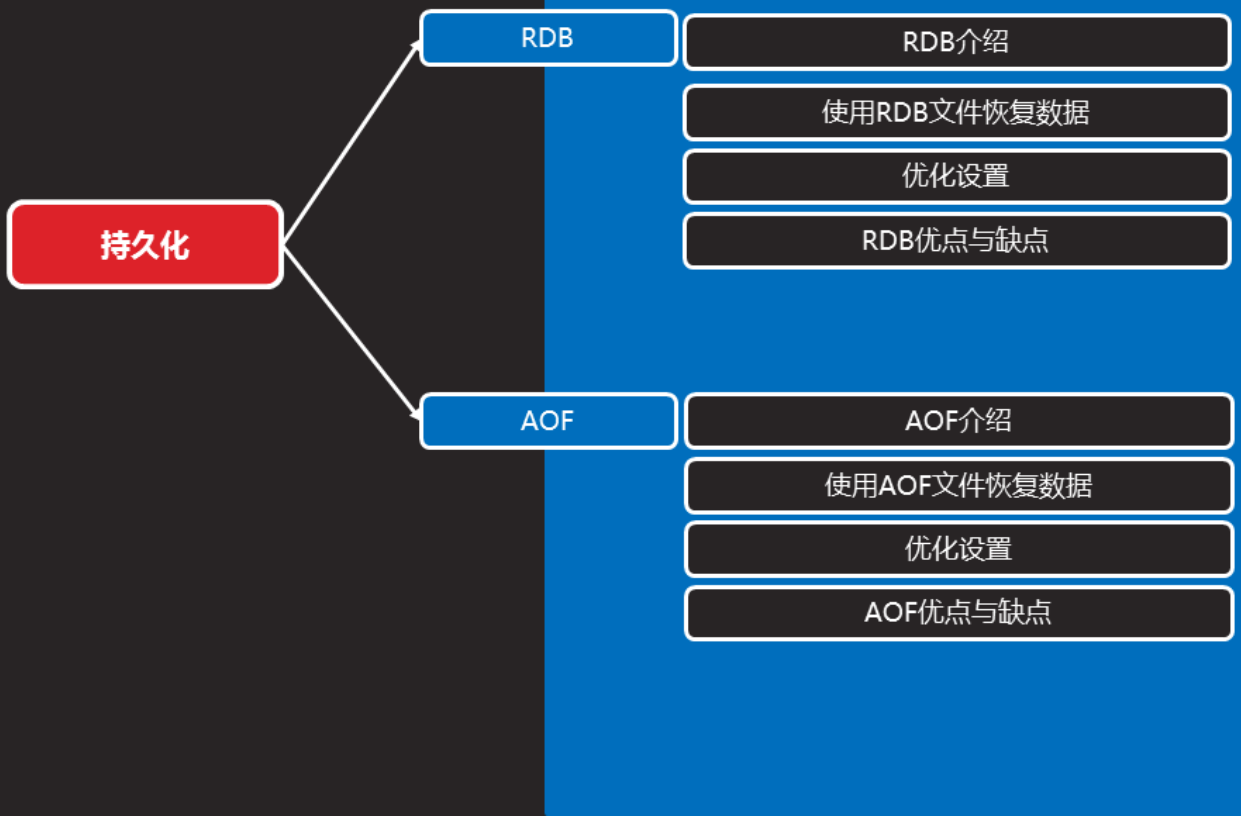
Tedu.cn 达内教育

具体要求如下：

- 基于案例2的配置
- 配置哨兵服务
- 测试配置

课堂练习


## 持久化



# RDB

## RDB介绍

知识讲解

- Redis数据库文件，全称 Redis DataBase
  - 数据持久化方式之一
  - 数据持久化默认方式
  - 按照指定时间间隔，将内存中的数据集快照写入硬盘
  - 快照术语叫Snapshot
  - 恢复时，将快照文件直接读入内存
- 定义RDB文件名
  - dbfilename “dump.rdb” //文件名

# 使用RDB文件恢复数据

知识讲解

- 备份数据
  - 备份dump.rdb 文件到其他位置

```
]# cp  数据库目录/dump.rdb    备份目录
```

- 恢复数据
  - 拷贝备份文件到数据库目录，重启redis服务

```
]# cp  备份目录/dump.rdb   数据库目录/
```

# 优化设置

知识讲解

- 数据从内存保存到硬盘的频率
  - save 900 1　　//15分钟且有1个key改变
  - save 300 10　　//5分钟且有10个key改变
  - save 60 10000　　//1分钟且有10000个key改变
- 手动存盘
  - save　　//阻塞写存盘
  - bgsave　　//不阻塞写存盘

# RDB优点与缺点

知识讲解

- RDB优点
  - 高性能的持久化实现 —— 创建一个子进程来执行持久化，先将数据写入临时文件，持久化过程结束后，再用这个临时文件替换上次持久化好的文件；过程中主进程不做任何IO操作
  - 比较适合大规模数据恢复，且对数据完整性要求不是非常高的场合
- RDB的缺点
  - 意外宕机时，丢失最后一次持久化的所有数据

# 案例4 ：使用RDB文件恢复数据

课堂练习

要求如下：

- 启用RDB
- 设置存盘间隔为120秒 10个key改变即存盘
- 备份RDB文件
- 删除数据
- 使用RDB文件恢复数据

# AOF

## AOF介绍

知识讲解

- Append Only File
  - 追加方式记录写操作的文件
  - 记录redis服务所有写操作
  - 不断的将新的写操作，追加到文件的末尾
  - 默认没有启用
  - 使用cat命令可以查看文件内容
- 启用AOF

```
> config set appendonly yes    //启用
> config rewrite               //写进配置文件
```

# 使用AOF文件恢复数据

Tedu.cn 达内教育

知识讲解

- 备份数据
  - 备份appendonly.aof文件到其他位置

```
]# cp    数据库目录/applendonly.aof     备份目录
```

- 恢复数据
  - 拷贝备份文件到数据库目录
  - 重启redis服务

```
]# cp  备份目录/appendonly.aof   数据库目录

]# /etc/redis/redis_端口  start
```

# 优化配置

Tedu.cn 达内教育

知识讲解

- 定义文件名
  - appendonly  yes  //启用aof ，默认no
  - appendfilename  "appendonly.aof"  //指定文件名
- AOF文件记录写操作的方式
  - appendfsync  always  //时时记录，并完成磁盘同步
  - appendfsync  everysec  //每秒记录一次，并完成磁盘同步
  - appendfsync  no  //写入aof ，不执行磁盘同步

# 优化配置（续2）

知识讲解

- 日志文件会不断增大，何时触发日志重写？
  - auto-aof-rewrite-min-size 64mb //首次重写触发值
  - auto-aof-rewrite-percentage 100 //再次重写，增长百分比
- 修复AOF文件
  - 把文件恢复到最后一次的正确操作

```
[root@redis53 6379]# redis-check-aof --fix appendonly.aof
0x 83: Expected \r\n, got: 6166
AOF analyzed: size=160, ok_up_to=123, diff=37
This will shrink the AOF from 160 bytes, with 37 bytes, to 123 bytes
Continue? [y/N]: y
Successfully truncated AOF
```

# AOF优点与缺点

知识讲解

- AOF优点
  - 可以灵活设置持久化方式
  - 出现意外宕机时，仅可能丢失1秒的数据
- AOF缺点
  - 持久化文件的体积通常会大于RDB方式
  - 执行fsync策略时的速度可能会比RDB方式慢

# 案例5 ：使用AOF文件恢复数据

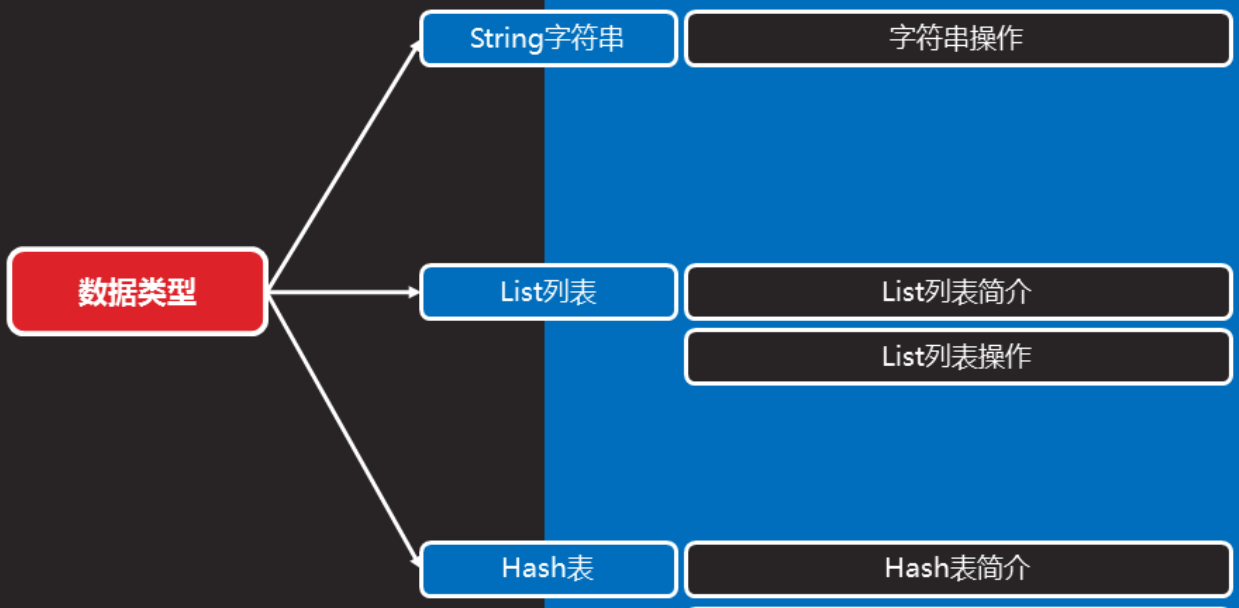
Tedu.cn 达内教育

要求如下：

- 启用AOF
- 备份AOF文件
- 删除数据
- 使用AOF文件恢复数据

课堂练习

## 数据类型



# String字符串

## 字符串操作

知识讲解

- set key value [ex seconds] [px milliseconds] [nx|xx]
  - 设置key及值，过期时间可以使用秒或毫秒为单位
- setrange key offset value
  - 从偏移量开始复写key的特定位的值

```
> set  first  "hello world"
> setrange  first  6  "Redis"                    //改写为hello Redis
```

- strlen key，统计字串长度

```
> strlen first
```

# 字符串操作（续1）

Tedu.cn 达内教育

知识讲解

- append key value
  - 存在则追加，不存在则创建key及value，返回key长度

```
> append myname jacob
```

- setbit key offset value
  - 对key所存储字串，设置或清除特定偏移量上的位(bit)
  - value值可以为1或0，offset为0~2^32之间
  - key不存在，则创建新key

```
> setbit  bit  0  1          //设置bit第0位为1
> setbit  bit  1  0          //设置bit第1位为0
```

# 字符串操作（续2）

Tedu.cn 达内教育

知识讲解

- bitcount key
  - 统计字串中被设置为1的比特位数量

```
> setbit  bits 0 1           //0001
> setbit  bits 3 1           //1001
> bitcount  bits             //结果为2

> setbit  peter  100  1      //网站上线100天用户登录了一次
> setbit  peter  105  1      //网站上线105天用户登录了一次
> bitcount  peter
```

**场景说明**：
记录网站用户上线频率，如用户A上线了多少天等类似的数据
如用户在某天上线，则使用setbit，以用户名为key，将网站上线日为offset，并在该offset上设置1，最后计算用户总上线次数时，使用bitcount用户名即可。
这样，即使网站运行10年，每个用户仅占用10*365比特位即456字节。

# 字符串操作（续3）

知识讲解

- decr key
  - 将key中的值减1，key不存在则先初始化为0，再减1

```
> set test 10
> decr test
```

- decrby key decrement
  - 将key中的值，减去decrement

```
> set count 100
> decrby count 20
```

- get key
  - 返回key存储的字符串值，若key不存在则返回null
  - 若key的值不是字串，则返回错误，get只能处理字串

# 字符串操作（续4）

知识讲解

- getrange key start end
  - 返回字串值中的子字串，截取范围为start和end
  - 负数偏移量表示从末尾开始计数，-1表示最后一个字符，-2表示倒数第二个字符

```
> set first "hello,the world"
> getrange first -5 -1
> getrange first 0 4
```

# 字符串操作（续5）

Tedu.cn 达内教育

知识讲解

- incr key
  - 将key的值加1，如果key不存在，则初始为0后再加1
  - 主要应用为计数器

```
> set  page 20
> incr  page
```

- incrby key increment
  - 将key的值增加increment

# 字符串操作（续6）

Tedu.cn 达内教育

知识讲解

- incrbyfloat key increment
  - 为key中所储存的值加上浮点数增量 increment

```
> set  num  16.1
> incrbyfloat  num  1.1
```

- mget key [key...]
  - 获取一个或多个key的值，空格分隔，具有原子性
- mset key value [key value ...]
  - 设置多个key及值，空格分隔，具有原子性

## 案例6：string 字符串

练习字符串类型命令的使用，具体命令如下：

课堂练习

- set mset setrange get mget
- getrange strlen append
- setbit bitcount
- decr decrby
- incr incrby
- incrbyfloat

# List列表

# List列表简介

Tedu.cn 达内教育

知识讲解

- Redis的list是一个字符队列
- 先进后出
- 一个key可以有多个值

# List列表操作

Tedu.cn 达内教育

知识讲解

- lpush key value [value...]
  - 将一个或多个值value插入到列表key的表头
  - Key不存在，则创建key

```
> lpush list a b c          //list1值依次为c、b、a
```

- lrange key start stop
  - 从开始位置读取key的值到stop结束

```
> lrange list 0 2           //从0位开始，读到2位为止
> lrange list 0 -1          //从开始读到结束为止
> lrange list 0 -2          //从开始读到倒数第2位为止
```

# List列表操作（续1）

知识讲解

- lpop key
  - 移除并返回列表头元素数据，key不存在则返回nil

```
> lpop  list                    //删除表头元素，可以多次执行
```

- llen key
  - 返回列表key的长度

# List列表操作（续2）

知识讲解

- lindex key index
  - 返回列表中第index个值

```
> lindex  key  0 ;   lindex  key  2;   lindex  key  -2
```

- lset key index value
  - 将key中index位置的值修改为value

```
> lset  list  3  test                    //将list中第3个值修改为test
```

# List列表操作（续3）

知识讲解

- rpush key value [value...]
  - 将value插入到key的末尾

```
> rpush  list3  a b c          //list3值为a b c
> rpush  list3  d              //末尾插入d
```

- rpop key
  - 删除并返回key末尾的值

```
> rpush  list4   a  b  c       //list4值为a b c
> rpop  list4                  //删除末尾的c，并返回删除的值
```

# 案例7：list 列表

课堂练习

练习列表类型命令的使用，具体命令如下：

- lpush llen lrange
- lpop lindex lset
- rpush rpop

# Hash表

## Hash表简介

知识讲解

- Redis hash
  - 是一个string类型的field和value的映射表
  - 一个key可对应多个field，一个field对应一个value
  - 将一个对象存储为hash类型，较于每个字段都存储成string类型更能节省内存

# Hash表操作

知识讲解

- hset key field value
  - 将hash表中field值设置为value

```
> hset site google 'www.g.cn'
> hset site baidu 'www.baidu.com'
```

- hget key filed
  - 获取hash表中field的值

```
> hget site google
```

# Hash表操作（续1）

知识讲解

- hmset key field value [field value...]
  - 同时给hash表中的多个field赋值

```
> hmset site google www.g.cn baidu www.baidu.com
```

- hmget key field [field...]
  - 返回hash表中多个field的值

```
> hmget site google baidu
```

- hkeys key
  - 返回hash表中所有field名称

```
> hmset site google www.g.cn baidu www.baidu.com
> hkeys site
```

# Hash表操作（续2）

知识讲解

- hgetall key
  - 返回hash表中所有field的值
- hvals key
  - 返回hash表中所有filed的值

```
> hvals key
```

- hdel key field [field...]
  - 删除hash表中多个field的值，不存在则忽略

```
> hdel site google baidu
```

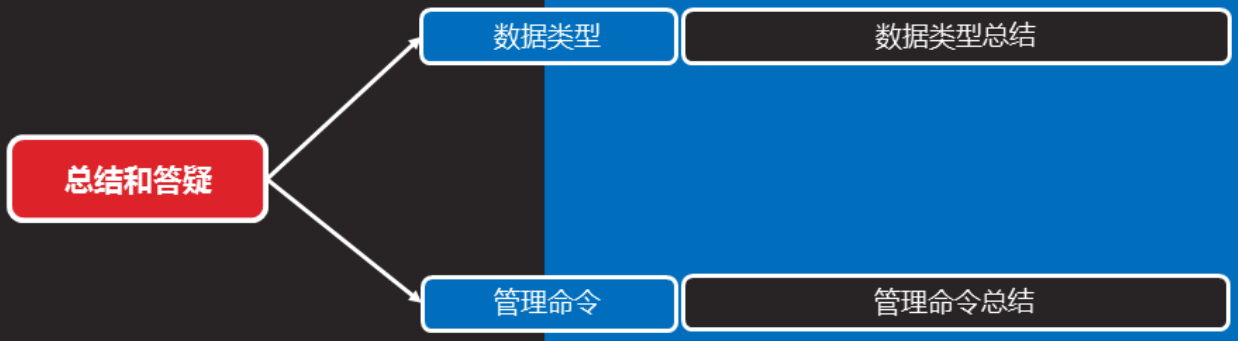
# 案例8：Hash表

课堂练习

练习hash表类型命令的使用，具体命令如下：

- hset hmset
- hgetall hkeys hvals
- hget hmget
- hdel

## 总结和答疑




Tedu.cn 达内教育

# 数据类型

# 数据类型总结

- 字符类型
- hash表类型
- List列表类型

知识讲解

# 管理命令

# 管理命令总结

Tedu.cn 达内教育

知识讲解

- del key [key...]
  - 删除一个或多个key
- exists key
  - 测试一个key是否存在
- expire key seconds
  - 设置key的生存周期
- persist key
  - 设置key永不过期
- ttl key
  - 查看key的生存周期