# Redis：

## 常见数据结构：

|  |  |  |
| --- | --- | --- |
| 数据结构 | 描述及用法 | 案例 |
| 字符串-Strings | 描述：最基本的类型，可以是简单的字符串、二进制数据或数字。  用法：可以用于存储用户信息、缓存网页内容等 | SET key "value"  GET key |
| 哈希-Hashes | 描述：用于存储键值对集合，适合存储对象。  用法：可以用于存储用户信息（如用户名、年龄、邮箱等） | HSET user:1000 name "John"  HGET us  er:1000 name |
| 列表-Lists | 描述：有序的字符串列表，可以从两端插入和移除元素。  用法：可以用于消息队列、任务列表等 | LPUSH tasks "task1"  RPUSH tasks "task2"  LPOP tasks  RPOP tasks |
| 无序集合-Sets | 描述：无序的字符串集合，支持集合操作如交集、并集和差集。  用法：可以用于标签、共同朋友等 | SADD tags "redis"  SADD tags "database"  SMEMBERS tags |
| 有序集合-Sorted Sets | 描述：类似集合，但每个元素都有一个分数，根据分数排序。  用法：可以用于排行榜、优先级队列等 | ZADD leaderboard 100 "user1"  ZADD leaderboard 200 "user2"  ZRANGE leaderboard 0 -1 |

## 不常见的数据结构

|  |  |  |
| --- | --- | --- |
| 位图-Bitmaps | 描述：操作字符串的二进制位，可以用于高效存储和处理位数据。  用法：可以用于统计、标记用户活跃状态等 |  |
| HyperLogLog | 描述：用于基数统计的概率性数据结构，适合大数据集的基数估算。  用法：可以用于统计独立访问IP、点击数等 |  |
| 流-Streams | 描述：日志型数据结构，适合高效处理时间序列数据。  用法：可以用于日志系统、实时消息系统等 |  |
| 空间索引-Geospatial | 描述：用于存储地理位置的数据结构，支持地理半径查询。  用法：可以用于存储和查询地理位置信息 |  |

## 持久化机制

### 4.1， RDB

#### 4.1.1，概述

也叫快照，将某一时刻的数据全部存放到硬盘上。

redis会调用fork来创建子线程去负责生成快照，主线程继续负责处理请求。

缺点是系统发生奔溃，则最新一次快照文件之后的数据将无法恢复

#### 4.1.2，触发方式

主动触发：redis客户端发送BGSAVE指令，fork创建子线程去创建快照文件

条件触发：配置文件：save 60 10000, 60秒内有10000次写入，则触发快照文件生成。

### 4.2，AOF

#### 4.2.1，概述

全称：Append Only File，将每一个写操作指令追加到AOF文件中，AOF是纯文本文件，通过重放AOF文件来恢复数据。

有三种同步策略：

1. always，每次有写操作，都追加到AOF文件，性能低，影响磁盘寿命
2. everysec，默认的，每秒将写操作追加到AOF文件，意思就是最多只会丢失一秒内的数据
3. no，不同步，以来操作系统的缓存机制进行写入，性能最高，但最不安全。

AOF重写：时间越往后，AOF文件越大，通过创建新的AOF文件，里面仅包含恢复数据库的最小操作集，而其实异步的，既有注定触发，也可以条件触发。

### 4.3，两种对比以及使用

数据持久性：

AOF有更高的数据持久性，几乎可以实时同步

RDF有一定的风险，可能会丢失最新一次快照文件之后的所有写操作

性能：

AOF，频繁写入磁盘，性能低

RDB，快照保存，性能高

文件大小和恢复速度：

AOF，文件大，恢复速度慢

RDB，文件小，恢复速度快

一般情况下两个同时使用：

RDB提供快速冷启动恢复

AOF提供更高的数据持久性，确保极端情况下的数据安全。

## **缓存设计中的考虑因素：**

### 4.1，缓存策略

缓存失效：为每个缓存设置TTL（Time to live，失效时间）时间，确保数据到期后重新加载最新数据

主动更新：在数据更新时，主动删除缓存或者更新缓存。确保和数据库中的数据一致

### 4.2，缓存穿透：

指的是特定的请求会绕过缓存直接访问数据库，从而导致缓存失去意义，数据库的负载大幅增加。通常情况下，缓存穿透发生在请求数据在数据库中不存在，因为这种情况下，缓存中不会保存这些无效数据。

解决办法：布隆过滤器

### **4.3，缓存雪崩**

### 4.4，缓存预热

### **4.5，缓存与数据库一致性**

策略：写通过，写回，写旁路

# Java

# Mysql

# Linux

# JVM

# Spring框架

# Git

# Maven