# REDIS

①字符串（strings），

②散列（hashes），

③列表（lists），

④ 集合（sets），

⑤有序集合（sorted sets），

与范围查询， bitmaps， hyperloglogs 和 地理空间（geospatial） 索引半径查询。 Redis 内置了 复制（replication），LUA脚本（Lua scripting）， LRU驱动事件（LRU eviction），事务（transactions） 和不同级别的 磁盘持久化（persistence）， 并通过 Redis哨兵（Sentinel）和自动 分区（Cluster）提供高可用性（high availability）。

Redis是内存中的数据结构存储系统，它可以用作数据库、缓存和消息中间件

数据库

Redis本质是内存数据库，所以自然可以当做数据库来使用，但要注意的是内存空间是极其有限的，可不像硬盘那样浩瀚无垠，所以大多数情况下我们还是用关系型数据库+Redis缓存的方式运用Redis

缓存

比如Mysql，可承担的并发访问量有多大呢？答案是几百左右就会扛不住了，所以我们为了支持更高的并发，会使用缓存，为数据库筑起一道护盾，让大多数请求都发生在缓存这一层。Redis是把数据存储在内存上的，访问数据速度相当快，很适合做缓存。

消息中间件

Redis支持发布/订阅消息，当然真正的MQ我们一般在Rabbit，Rocket，kakfa之间选一个，这并不是Redis的强项

# 4.1 Redis五大数据类型–String

Redis 的字符串是动态字符串，是可以修改的字符串，内部结构实现上类似于 Java 的 ArrayList，采用预分配冗余空间的方式来减少内存的频繁分配，，内部为当前字符串实际分配的空间 capacity 一般要高于实际字符串长度 len。当字符串长度小于 1M 时，扩容都是加倍现有的空间，如果超过 1M，扩容时一次只会多扩 1M 的空间。需要注意的是字符串最大长度为 512M。

SDS 结构使用了范型 T，为什么不直接用 int 呢，这是因为当字符串比较短时，len 和 capacity 可以使用 byte 和 short 来表示，Redis 为了对内存做极致的优化，不同长度的字符串使用不同的结构体来表示。所以别人才那么快嘛，太精细了！

注意：创建字符串时 len 和 capacity 一样长，不会多分配冗余空间，这是因为绝大多数场景下我们不会使用 append 操作来修改字符串

# 4.2 Redis五大数据类型–list

单键多值

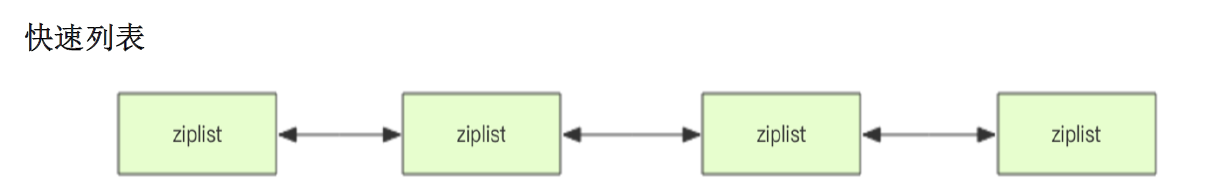
Redis 列表是简单的字符串列表，按照插入顺序排序。你可以添加一个元素导列表的头部（左边）或者尾部（右边）。

它的底层实际是个双向链表，对两端的操作性能很高，通过索引下标的操作中间的节点性能会较差。操作的时间复杂度为 O(1)，但是索引定位很慢，时间复杂度为 O(n).

其实底层用的是ZipList（压缩链表）和QuickList（快速链表）。

首先在列表元素较少的情况下会使用一块连续的内存存储，这个结构是 ziplist，也即是压缩列表。它将所有的元素紧挨着一起存储，分配的是一块连续的内存。（感觉好像数组啊，连续的内存空间）

当数据量比较多的时候才会改成 quickList。那么什么是quickList呢？Redis 将链表和 ziplist 结合起来组成了 quickList。



# 4.3 Redis五大数据类型–set

Redis set对外提供的功能与list类似是一个列表的功能，特殊之处在于set是可以自动排重的，当你需要存储一个列表数据，又不希望出现重复数据时，set是一个很好的选择，并且set提供了判断某个成员是否在一个set集合内的重要接口，这个也是list所不能提供的。

Redis的Set是string类型的无序集合。它底层其实是一个value为null的hash表,所以添加，删除，查找的复杂度都是O(1)

# 4.4 Redis五大数据类型–hash

Redis hash 是一个键值对集合。

Redis hash是一个string类型的field和value的映射表，hash特别适合用于存储对象。

类似Java里面的Map

# 4.5 Redis五大数据类型–zset (sorted set)

Redis有序集合zset与普通集合set非常相似，是一个没有重复元素的字符串集合。不同之处是有序集合的每个成员都关联了一个评分（score） ，这个评分（score）被用来按照从最低分到最高分的方式排序集合中的成员。集合的成员是唯一的，但是评分可以是重复了 。

因为元素是有序的, 所以你也可以很快的根据评分（score）或者次序（position）来获取一个范围的元素。访问有序集合的中间元素也是非常快的,因此你能够使用有序集合作为一个没有重复成员的智能列表。

Redis 两大持久化策略：RDB 与AOF

Redis 提供了不同级别的持久化方式:

RDB（RedisDataBase）持久化方式能够在指定的时间间隔能对你的数据进行快照存储.

AOF（AppendOnlyFile）持久化方式记录每次对服务器写的操作,当服务器重启的时候会重新执行这些命令来恢复原始的数据,AOF命令以redis协议追加保存每次写的操作到文件末尾.Redis还能对AOF文件进行后台重写,使得AOF文件的体积不至于过大.

如果你只希望你的数据在服务器运行的时候存在,你也可以不使用任何持久化方式.

你也可以同时开启两种持久化方式, 在这种情况下, 当redis重启的时候会优先载入AOF文件来恢复原始的数据,因为在通常情况下AOF文件保存的数据集要比RDB文件保存的数据集要完整.

最重要的事情是了解RDB和AOF持久化方式的不同,让我们以RDB持久化方式开始:

# 分布式锁

1. 数据库级别乐观锁，悲观锁，乐观锁就是在更新时添加version字段的校验，如果version
2. 悲观锁：共享锁（SELECT）、更新锁(FOR UPDATE, )、排他锁(INSERT/DELETE/UPDATE)
3. Redis : a.利用redis 单线程优势，SETNX + EXPIRE控制， NX代表 对象不存在才能set成功（即获得锁），通过设置 EXPIRE，即设置一个缓存失效时间，预防因程序错误锁没释放导致的死锁，释放锁就是把这个对象delete掉
4. Redission ： redission.getLock(“xxx”)
5. Zookeeper：watcher机制，“临时顺序队列”