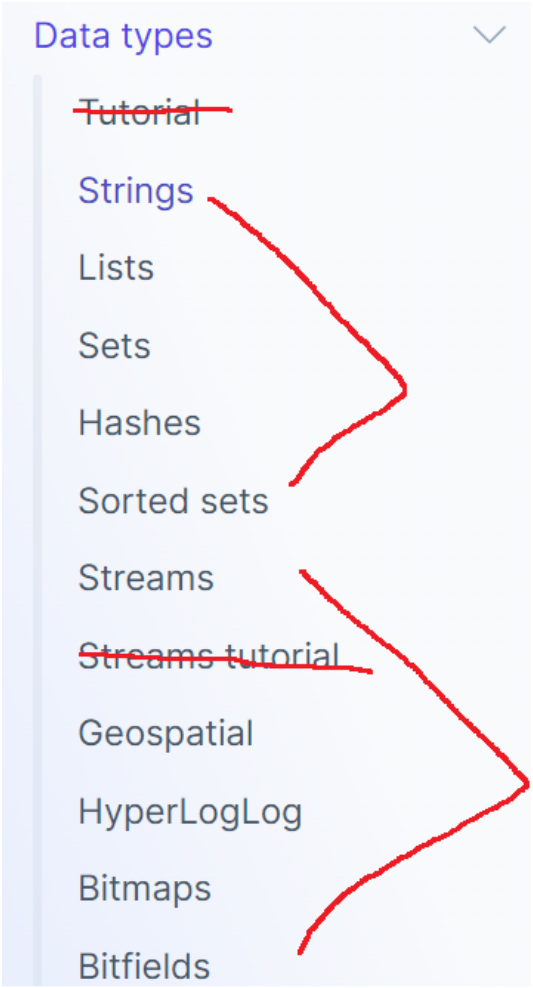
Redis 高级数据类型详解：Stream、Geospatial、HyperLogLog、Bitmaps、Bitfields（134 ~ 139）

Redis 除了常用的 String、List、Set、ZSet、Hash 外，还提供了多种**针对特定场景优化的数据类型**，用于处理消息队列、地理位置、基数统计、位操作等高性能需求。



一、Redis Stream 类型：高可靠消息队列与事件流

1. 核心定义与特性

Redis Stream 是从 **Redis 5.0** 引入的日志型数据结构，设计目标是高可靠的消息队列。

- **数据结构特性**：

- 每条消息都有唯一 **ID**，格式为 `时间戳-序号`（如 `1695882334567-0`），保证全局唯一并严格递增。
- 消息是有序的，天然支持时间排序。
- 支持**持久化**，即使 Redis 重启消息不会丢失。
- 支持多生产者、多消费者模式。

- **底层实现**：
  - 基于 **quicklist**（ziplist + 双向链表）
  - 消息存储内存紧凑，支持快速插入和范围查询

---

2. 核心设计思想

- **事件驱动模型**：
  - 消费者可以通过 `XREAD` / `XREADGROUP` 阻塞监听 Stream
  - Redis 内部利用 `epoll` 等 I/O 多路复用机制唤醒等待的消费者，避免轮询浪费资源
- **消费组（Consumer Group）机制**：
  - 每个消费组可包含多个消费者
  - 消息分配给组内空闲消费者，实现**负载均衡**
  - 支持“至少一次”的消息投递，消费者需通过 `XACK` 确认消息
- **典型应用场景**：
  - 订单支付回调队列
  - 系统日志收集与分析
  - 实时数据同步
  - 异步任务队列（如消息推送、任务调度）

---

3. 示例命令

代码块

```
XADD mystream * user "Alice" action "login"
XREAD COUNT 10 STREAMS mystream 0
XGROUP CREATE mystream mygroup $ MKSTREAM
XREADGROUP GROUP mygroup consumer1 COUNT 10 STREAMS mystream >
XACK mystream mygroup 1695882334567-0
```

- **解释**：
  - `XADD`：生产消息
  - `XREAD`：读取消息（可阻塞）
  - `XGROUP`：创建消费组
  - `XREADGROUP`：消费组消费消息
  - `XACK`：确认消费

二、Redis Geospatial：地理位置数据类型

1. 核心功能

Geospatial 用于存储和查询地理坐标，底层通过 **Geohash** 实现高效范围查询。

- **核心命令**：

```
代码块
1  GEOADD locations 116.4074 39.9042 "Beijing"
2  GEODIST locations Beijing Shanghai km
3  GEORADIUS locations 116.4 39.9 50 km WITHDIST WITHCOORD COUNT 5
```

- **命令说明**：
  - `GEOADD`：添加经纬度坐标
  - `GEODIST`：计算两点间距离
  - `GEORADIUS`：范围查询附近位置，支持返回距离、坐标、数量限制
- **底层实现**：
  - 经度/纬度 -> Geohash 字符串
  - 使用字典存储 member -> Geohash
  - 范围查询通过 Geohash 前缀匹配实现，时间复杂度约 `O(logN)`

---

2. 典型场景

- 社交 App：附近用户推荐
- 外卖平台：附近商家查询
- LBS 服务：物流轨迹跟踪、营销推送

- **工程实践**：
  - 精度与性能折中：Geohash 位数越多，精度越高，但索引更大
  - 可结合 **ZSet** 排序距离，实现“最近距离排序”

三、Redis HyperLogLog：概率型基数统计

1. 核心功能

- **目的**：在固定内存下统计大规模数据唯一元素数量
- **特性**：
  - 内存固定约 12 KB
  - 误差率约 0.81%
  - 非精确，但足够用于 UV 统计或大数据去重
- **核心命令**：

```
代码块
1   PFADD uv2026 user1 user2 user3
2   PFCOUNT uv2026
3   PFMERGE uvTotal uv2026 uv2025
```

---

2. 典型场景

- 网站 UV / DAU / MAU 统计
- 日志分析中唯一事件统计
- 用户行为分析
- **工程实践**：
  - 误差可接受 → 用 HyperLogLog
  - 需要精确计数 → 使用 Set 或数据库

四、Redis Bitmaps：位操作类型

1. 核心功能

- 基于字符串的位级操作，每个位仅占 1 bit

- 支持高效统计和状态标记
- **核心命令**：

```
代码块
1   SETBIT user:1:sign 0 1
2   GETBIT user:1:sign 0
3   BITCOUNT user:1:sign
4   BITOP AND user:active1 user:active2
```

- **典型用途**：
  - 用户签到统计（每位表示日期）
  - 用户在线状态（1 表示在线，0 离线）
  - 多天活跃用户交集（连续活跃用户统计）
- **底层实现**：
  - 基于 SDS（简单动态字符串）
  - 支持动态扩容

五、Redis Bitfields：多位整数操作

1. 核心功能

- 可在单个字符串中存储多个小整数
- 支持读/写/增量/溢出处理
- **核心命令**：

```
代码块
1   BITFIELD user:stats
2     SET u4 0 5
3     INCRBY u4 4 1
4     GET u4 0
5   BITFIELD user:stats OVERFLOW SAT INCRBY u4 4 10
```

- **解释**：
  - `SET`：设置某个位段整数
  - `INCRBY`：增加整数值

- `OVERFLOW`：处理溢出（WRAP 循环 / SAT 饱和 / FAIL 报错）

- **底层实现**：
  - 使用 SDS 存储
  - 二进制位紧凑排列，提高内存利用率

---

2. 典型场景

- 游戏属性存储：等级、金币、经验
- 状态位聚合：多个布尔状态/小整数状态聚合
- 优势：
  - 减少键数量
  - 内存占用极低

六、核心设计思想与最佳实践

1. **场景驱动选型**：
   - 消息队列 → Stream
   - 地理位置 → Geospatial
   - 基数统计 → HyperLogLog
   - 位状态 → Bitmaps
   - 多整数聚合 → Bitfields
2. **内存与精度权衡**：
   - HyperLogLog：极小内存 + 可接受误差
   - Bitmaps / Bitfields：位级存储，极高效率
3. **性能与可靠性平衡**：
   - Stream：消费组 + Ack，企业级可靠投递
   - Geospatial：Geohash 编码，精度与性能折中
4. **工程实践建议**：
   - 针对不同业务选择最合适数据类型
   - 对大规模数据考虑存储压缩、分片或批量处理
   - 结合 Redis 持久化策略，确保可靠性

✅ 总结

Redis 提供了丰富的数据类型，不仅支持基本 KV 操作，还提供了针对特定业务场景的高性能解决方案：

- **Stream** → 高可靠消息队列
- **Geospatial** → 地理位置查询
- **HyperLogLog** → 海量基数统计
- **Bitmaps** → 位操作与状态存储
- **Bitfields** → 多整数紧凑存储

通过合理选择和组合，可以实现高性能、低内存占用的复杂业务逻辑，如实时排行榜、连续活跃用户统计、基于位置的推荐等。