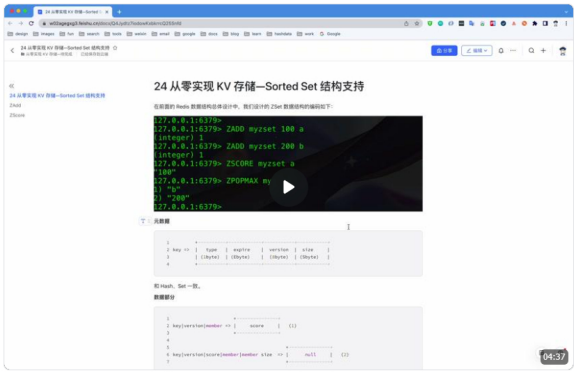
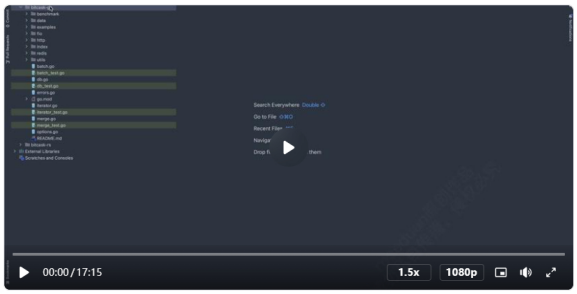
24 从零实现 KV 存储—Sorted Set 结构支持

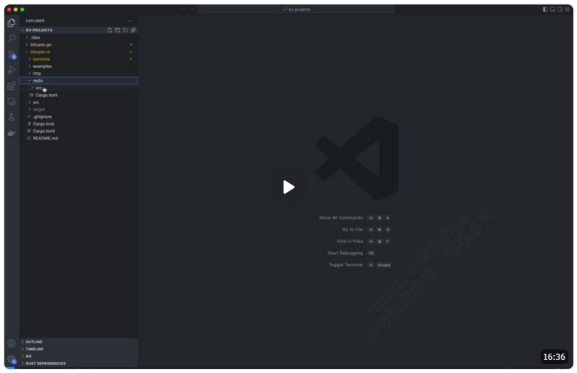
理论讲解



Go 编码



Rust 编码



在前面的 Redis 数据结构总体设计中，我们设计的 ZSet 数据结构的编码如下：

```
127.0.0.1:6379>
127.0.0.1:6379> ZADD myzset 100 a
(integer) 1
127.0.0.1:6379> ZADD myzset 200 b
(integer) 1
127.0.0.1:6379> ZSCORE myzset a
"100"
127.0.0.1:6379> ZPOPMAX myzset 1
1) "b"
2) "200"
127.0.0.1:6379>
```

元数据

```
1         +----------+------------+-----------+-----------+
2 key =>  |   type   |   expire   |  version  |   size    |
3         | (1byte)  | (8byte)    |  (8byte)  | (8byte)   |
4         +----------+------------+-----------+-----------+
```

和 Hash、Set 一致。

数据部分

```
1                       +---------------+
2 key|version|member =>  |     score     |   (1)
3                       +---------------+
4
5                                         +---------------+
6 key|version|score|member|member size  =>  |      null     |   (2)
7                                         +---------------+
```

分为两部分存储，第一部分主要是通过 key + member 获取到对应的 score，第二部分将 member 按照 score 分值进行排列，主要是为了能够按照 score 的顺序获取 member。

ZAdd

首先根据 key 获取元数据，如果没拿到的话则初始化新的元数据结构体。

然后再获取这个 member 是否已经存在了，如果存在，并且其 score 和用户设置的 score 是一样的话，那么不需

要做任何操作。

否则的话需要更新这个值，构造数据部分的两个 key，调用存储引擎的接口写入数据，并且更新元数据。

ZScore

首先根据 key 获取元数据，如果没获取到的话，则说明 key 不存在则直接返回。

否则根据元数据的信息构造数据部分的 key：`key|version|member`，调用存储引擎的接口获取 score 的值。