資料結構 – HW4 報告 *2024 DEC*

姓名：徐崇恆
學號：112503023
系級：通訊二

1 摘要

本作業旨在 Linux 環境中以 C 語言實作類 Redis 的 NoSQL 資料庫。本作業重點是為資料庫支援有序集合 (Sorted Set) 的基本操作。本作業效仿 Redis 使用跳表 (Skip List) 和雜湊表 (Hash Table) 的結合實現有序集合。

2 需求分析

Sorted Set 是一種集合類型的資料結構，它的每個元素包含一個字串成員 (member) 與一個分數 (score)，score 的型別為 double 浮點數。集合的元素按照分數進行排序，一個集合中所有元素的 member 在該集合中都是唯一的。 在排序時當出現 score 相同的元素時則以 member 進行比較。當對非 Sorted Set 資料形態的鍵使用 Sorted Set 指令時，資料庫將會報錯。

在本作業中，需要支援的 Sorted Set 操作包括：

1. **ZADD**
   句法：ZADD key score1 member1 [score2 member2 …]
   向有序集合中添加一個或多個成員，回傳成功新增的成員數量。若成員已存在則更新該成員的分數。
2. **ZCARD**
   句法：ZCARD key
   回傳有序集合的成員數。若有序集合不存在則回傳 0。
3. **ZCOUNT**
   句法：ZCOUNT key min max
   計算在有序集合中指定區間分數的成員數，區間包含 min 和 max。可以用 -inf 和 +inf 分別表示負無窮和正無窮。若 min 和 max 設置不正確會報錯。
4. **ZINTERSTORE**
   句法：ZINTERSTORE destination numkeys key [key …] [WEIGHTS weight [weight ...]] [AGGREGATE SUM|MIN|MAX]
   例子：ZINTERSTORE sum_set 3 set_a set_b set_c WEIGHTS 4 3 2
   計算多個有序集合的交集並儲存到新集合，回傳新集合的大小。簡單來說，就是把不同有序集合裡面交集的 member 的分數加總，並儲存為一個新的有序集合。Weight 是權重，即該集合的 score 會被乘上的數值，預設為 1。Aggregate 為聚合方式，預設是 sum。
5. **ZUNIONSTORE**
   句法：ZUNIONSTORE destination numkeys key [key …] [WEIGHTS weight [weight ...]] [AGGREGATE SUM|MIN|MAX]
   與 *ZINTERSCORE* 類似，但它處理的是聯集。
6. **ZRANGE**
   句法：ZRANGE key [start] [stop] [WITHSCORES]（Redis 原句法：ZRANGE key start stop [WITHSCORES]）

回傳指定範圍內根據分數順序排列的元素。索引從 0 開始，並以 -1 表示最後一個元素。回傳範圍包含的位於 start 和 stop 的元素。與原版 Redis 不同的是，我們分別給予 start 和 stop 0 和 -1 的預設值。傳入 WITHSCORES 參數時，回傳列表為 member score [member2 score2 …] 形式的列表，否則只會傳一個 member 列表。若 min 和 max 設置不正確會報錯。

7. **ZRANGEBYSCORE**
   句法：ZRANGEBYSCORE key min max [WITHSCORES] [LIMIT offset limit]
   與 *ZRANGE* 類似，但它是以 score 作為範圍。
   1. 理解 [LIMIT offset limit]
      假設一個 Sorted Set 中有 a (1)、b (2)、c (3)、d (4)、e (5)、f (6)、g (7)
      我們使用 ZRANGEBYSCORE myset 2 6 WITHSCORES LIMIT 1 3
      分數範圍為 [2, 6]，暫存結果為：b (2)、c (3)、d (4)、e (5)、f (6)。
      LIMIT 1 3 的意思是跳過前 1 個元素，因此跳過 b (2) 。
      所以最終返回的 3 個元素是 c (3)、d (4)、e (5)。
   2. 表示小於
      min max 若以數字表示例如 5 10 即表示查詢 5 ≤ score ≤ 10 的元素。而以半角左括號標示的數值可以表示小於 <，例如 (5 10 表示查詢 5 < score ≤ 10， (5 (10 表示查詢 5 < score < 10。
8. **ZRANK**
   句法：ZRANK key member [WITHSCORE]
   回傳指定 Sorted Set 中的 member 的排名。以 0 為基數，即 score 最小的 member 排名為 0。傳入 WITHSCORE 會回傳一個 [rank, score] 的列表，否則只會傳一個整數的 rank。若 member 不存在，則返回 NULL 。
9. **ZREM**
   句法：ZREM key member [member …]
   從有序集合中移除一個或多個成員，回傳成功移除的成員數量。
10. **ZREMRANGEBYSCORE**
    句法：ZREMRANGEBYSCORE key min max
    從有序集合中移除指定範圍內的成員，其範圍選取機制與 *ZRANGEBYSCORE* 相同。

3 設計

Sorted Set

在本作業中，使用跳表 (Skip List) 和雜湊表 (Hash Table) 的結合實現有序集合。我們使雜湊表能夠儲存 Skip List 的元素，可以使分數的查詢時間複雜度為 O(1)。同時，結合跳表的特性使得範圍查詢的時間複雜度為 O(log n)。

Skip List

Skip List 是一種高效的資料結構，用於加速有序元素的搜尋、插入和刪除操作。它基於有序鏈結串列，透過引入多層索引，使得搜尋可以跳過部分節點，大幅提升效率。每個節點除了指向下一個節點外，還可能包含指向更高層級的索引指針，整體結構形成類似金字塔的多層鏈結串列。搜尋從最高層開始，逐層向下，插入則根據隨機概率決定新節點的高度，刪除過程與搜尋相似。Skip List 在效率和簡單性上優於許多平衡樹結構，並且具備動態平衡能力，適合用於資料庫索引等需要頻繁搜尋和插入的場景。

4 實作

結構體

DBZSetElement 將作為 skip list 中的元素，也是負責儲存資料的單位。而 DBZSet 則是 skip list 本體結構體與雜湊表以及哨站節點的結合。

```
121  typedef struct DBZSetElement
122  {
123    db_double_t score;
124    char *member;
125    db_uint8_t level;
126    struct DBZSetElement *backward;
127    struct DBZSetElement **forward;
128  } DBZSetElement;
129
130  typedef struct DBZSet
131  {
132    DBHash *dict;
133    db_uint8_t level;
134    DBZSetElement **sentinel_forward;
135    DBZSetElement *tail;
136  } DBZSet;
137
```

5 測試與結果

本作業中我們設計了一些測試案例以測試 Sorted Set 相關函式的功能。

```
root@pi:/home/cch137/hw4-cch137# bash build.sh
root@pi:/home/cch137/hw4-cch137# ./test
[PASS] zset_test_zadd: zcard == 3 (Expected: 3, Got: 3)
[PASS] zset_test_zadd: 'a' exists (Expected: true, Got: true)
[PASS] zset_test_zadd: 'e' exists (Expected: true, Got: true)
[PASS] zset_test_zadd: 'b' exists (Expected: true, Got: true)
[PASS] zset_test_zscore: score of 'a' == 1 (Expected: 1.00, Got: 1.00)
[PASS] zset_test_zscore: no_such_member is null (Expected: true, Got: true)
[PASS] zset_test_zcard: empty zset card == 0 (Expected: 0, Got: 0)
[PASS] zset_test_zcard: after adding 2 elements == 2 (Expected: 2, Got: 2)
[PASS] zset_test_zcount: [1,5] should be 5 (Expected: 5, Got: 5)
[PASS] zset_test_zcount: (1,5) should be 3 (Expected: 3, Got: 3)
[PASS] zset_test_zcount: (2,5] should be 3 (Expected: 3, Got: 3)
[PASS] zset_test_zrange: [1,2] == {b,c} (Expected: "{b,c}", Got: "{b,c}")
[PASS] zset_test_zrangebyscore: [2,3] == {b,c} (Expected: "{b,c}", Got: "{b,c}")
[PASS] zset_test_zrank: rank of 'b' == 1 (Expected: 1, Got: 1)
[PASS] zset_test_zrem: after removing 'b', zcard == 2 (Expected: 2, Got: 2)
[PASS] zset_test_zrem: 'b' removed (Expected: true, Got: true)
[PASS] zset_test_zremrangebyscore: removed count == 1 (Expected: 1, Got: 1)
[PASS] zset_test_zremrangebyscore: 'b' removed (Expected: true, Got: true)
[PASS] zset_test_zremrangebyscore: others remain (zcard==3) (Expected: 3, Got: 3)
[PASS] zset_test_zinterstore: zcard == 2 (Expected: 2, Got: 2)
[PASS] zset_test_zinterstore: 'b' score == 6 (Expected: 6.00, Got: 6.00)
[PASS] zset_test_zinterstore: 'c' score == 6 (Expected: 6.00, Got: 6.00)
[PASS] zset_test_zunionstore: zcard == 3 (Expected: 3, Got: 3)
[PASS] zset_test_zunionstore: 'a' score == 1 (Expected: 1.00, Got: 1.00)
[PASS] zset_test_zunionstore: 'b' score == 5 (Expected: 5.00, Got: 5.00)
[PASS] zset_test_zunionstore: 'c' score == 4 (Expected: 4.00, Got: 4.00)
DONE!
```

6 討論

與 Redis 的區別

在 Redis 中的設計中，score 的是以字串形式儲存的浮點數，只有在運算時被轉換為 double 浮點數。而在此作業的實作中，我們並沒有效仿 Redis，而是直接以 double 浮點數儲存 score。

沒有實現的需求

這次作業的函式有點多，而且需要函式的形式與之前的有些不同，需要更複雜的指令解析，所以尚未支援在終端以指令的形式使用這些功能。

7 結論

本次作業透過 C 語言實作類 Redis 的 NoSQL 資料庫，成功完成對有序集合 (Sorted Set) 的核心功能實現，包括多種操作指令如 ZADD、ZRANGE、ZINTERSTORE 等，並且利用跳表 (Skip List) 和雜湊表 (Hash Table) 的結合實現高效的資料處理。本實作驗證了相關資料結構的效率與適用性，並展示了其在有序資料處理中的優勢。然而，與 Redis 的實作相比，本作業仍有改進空間，特別是在指令解析及終端支援方面。此作業加深了對資料結構和資料庫設計的理解，為後續相關研究奠定了基礎。

8 參考文獻與資料

1. Redis - Docs
   https://redis.io/docs/latest/
2. Redis sorted set | Docs
   https://redis.io/docs/latest/develop/data-types/sorted-sets/
3. Skip Lists 跳表 - HackMD
   https://hackmd.io/@mam8t5W3TIuMwYqnndsBQQ/rJ_Tyq5wq

9 附錄

1. 函式庫 cJSON GitHub 倉庫 - DaveGamble/cJSON: Ultralightweight JSON parser in ANSI C
   https://github.com/DaveGamble/cJSON