

# 資料結構 HW1 Report

電機 3A 110501521 徐松廷

## 1. 作業要求:

於 Linux 環境，以 C 語言開發簡單的 NoSQL 資料庫引擎，支援 key-value pairs 進行存儲和查詢。你需要設計資料庫引擎，能夠有效地處理大量鍵值對數據，並支援基本的 CRUD ( 創建、讀取、更新、刪除 ) 操作，同時使用指位器與動態記憶體配置來管理數據。

## 2. Code 註解

作業一是希望可以建立一個具有插入、讀取、刪除、更新功能的資料庫，因為資料庫最常使用的功能就是查找，所以我在這邊使用 hash Table 來實現，Hash Table 的優點是在未產生碰撞的情況下，查找的時間複雜度都是  $O(1)$ 。

- a. Struct hash node: 一開始定義一個 hash\_node 的資料型態，一個 hash\_node 就是包含一對 key-value，並且為了處理 hash Table 的碰撞問題，多開一個 pointer 來實現 linked list，例如當 key 1-value1 和 key7-value7 對應到同一個 index 時，key7-value7 的 hash\_node 就插在 key 1-value1 的 hash\_node 後面。

```
struct hash_node {
    char* key;
    char* value;
    struct hash_node* ptr_to_next_hash_node;
};
```

- b. Struct hash\_TB\_array: Hash Table 的本體就是一個一百格的 array，每一格都儲存一個指向 struct hash\_node 結構的 pointer。

```
struct hash_TB_array {
    struct hash_node* TB_array[100];
    // 開一個一百格的array，每一格儲存的是指向struct hash_node結構的Pointer
};
```

- c. Insert function(C):

插入新的 key-value pair 需要動態配置出新的記憶體，並且在 insert function 中使用 strcpy 來對 C 語言中的字串賦值比較不會出錯。為了處理 Hash Table 的碰撞問題，這邊在插入時一樣也是會使用和 Linked list 相同的新增 node 的方法(如這個 function 的倒數兩行)。

```
void insert(struct hash_TB_array* ptr_to_TB_array, char* key, char* value) {
    struct hash_node* ptr_to_new_hash_node = (struct hash_node*)malloc(sizeof(struct hash_node));

    // 為key和value分配新的内存空间，并复制字符串
    ptr_to_new_hash_node->key = (char*)malloc(strlen(key) + 1);
    strcpy(ptr_to_new_hash_node->key, key);

    ptr_to_new_hash_node->value = (char*)malloc(strlen(value) + 1);
    strcpy(ptr_to_new_hash_node->value, value);

    int index = hash(key);

    // 為了處理碰撞問題，Array存放的Pointer會指向Linked list的開頭，在往後接下去成文linked list
    ptr_to_new_hash_node->ptr_to_next_hash_node = ptr_to_TB_array->TB_array[index];
    ptr_to_TB_array->TB_array[index] = ptr_to_new_hash_node;
}
```

d. Update function(U):

在更新一對 key-value pair 時，會先確認當前的 key 是否存在 hash table，如果不存在就不更新，存在的話，就是藉由 hash function 直接找到對應的 array index，並且把那一格對應到的 linked list 中含有該 key 的 node 更新(下圖中 while loop 中的動作)。

```
void updata(struct hash_TB_array* ptr_to_TB_array, char* key, char* value) {
    int index = hash(key);
    struct hash_node* cur_node = ptr_to_TB_array->TB_array[index];
    if (cur_node == NULL) {
        printf("The Key %s doesn't exist ", key);
        return;
    }
    while (1) {
        if (strcmp(cur_node->key, key) == 0) {
            // 找到了，為新的value分配内存并复制字符串
            free(cur_node->value); // 释放旧的内存
            cur_node->value = (char*)malloc(strlen(value) + 1);
            strcpy(cur_node->value, value);
            printf("Successfully updated\n");
            break;
        } else if (cur_node->ptr_to_next_hash_node == NULL) {
            break;
        } else {
            cur_node = cur_node->ptr_to_next_hash_node;
        }
    }
}
```

e. Delete function(D):

根據給定的 key 找到對應 index 並刪除，如果 key-value pair 不在鏈表中 (即 cur\_node 為 NULL)，它會發出一條找不到的消息。如果要刪除的 key-value pair 位於鏈表的開頭 (即 prev\_node 為 NULL)，它會更新索引位置的指針，並釋放相關內存。如果要刪除的 key-value pair 位於鏈表的中間或末尾，它會更新前一個節點的指針，跳過當前節點，並釋放相關內存。

```
void delete(struct hash_TB_array* ptr_to_TB_array, char* key){
    int index=hash(key);
    struct hash_node* cur_node=ptr_to_TB_array->TB_array[index];
    struct hash_node* prev_node=NULL;
    while(1){
        if(cur_node==NULL){
            printf("We didn't find the value to %s \n",key);
            break;
        }
        else{
            if(strcmp(cur_node->key,key)==0){//找到了
                if(prev_node!=NULL){
                    prev_node->ptr_to_next_hash_node=cur_node->ptr_to_next_hash_node;
                    free(cur_node);
                }
                else{//要刪除的在linked list的開頭
                    // cur_node->ptr_to_next_hash_node=NULL;
                    ptr_to_TB_array->TB_array[index] = cur_node->ptr_to_next_hash_node;
                    cur_node->key=NULL;
                }
            }
        }
    }
}
```

```

        cur_node->value=NULL;
        cur_node=NULL;
        free(cur_node);
    }
    printf("Successfully delete the value to %s \n",key);
    break;
}
else{//
    prev_node=cur_node;
    cur_node=cur_node->ptr_to_next_hash_node;//沒有就往下一個找
}
}
}

```

#### f. Read function(R):

get\_value 函數的作用是根據給定的 key 從 hash table 中檢索對應的值。它首先使用 hash 函數計算 key 的索引位置，如果找到了，它會返回該鍵值對的值；如果找不到，則返回 NULL 表示未找到。

```

char* get_value(struct hash_TB_array* ptr_to_TB_array,char* key){
    int index=hash(key);// 使用哈希函數計算鍵的索引位置
    struct hash_node* cur_node=ptr_to_TB_array->TB_array[index];// 取得該索引位置
    while(1){
        if(cur_node==NULL){// 如果鏈表為空，表示找不到指定的鍵值對
            break;
        }
        else{
            if(strcmp(cur_node->key,key)==0){// 找到了指定的鍵值對
                return cur_node->value;// 返回該鍵值對的值
            }
            else{// 如果當前節點不是要找的節點，繼續搜索下一個節點
                cur_node=cur_node->ptr_to_next_hash_node;
            }
        }
    }
    return NULL;
}

```

#### g. Hash function:

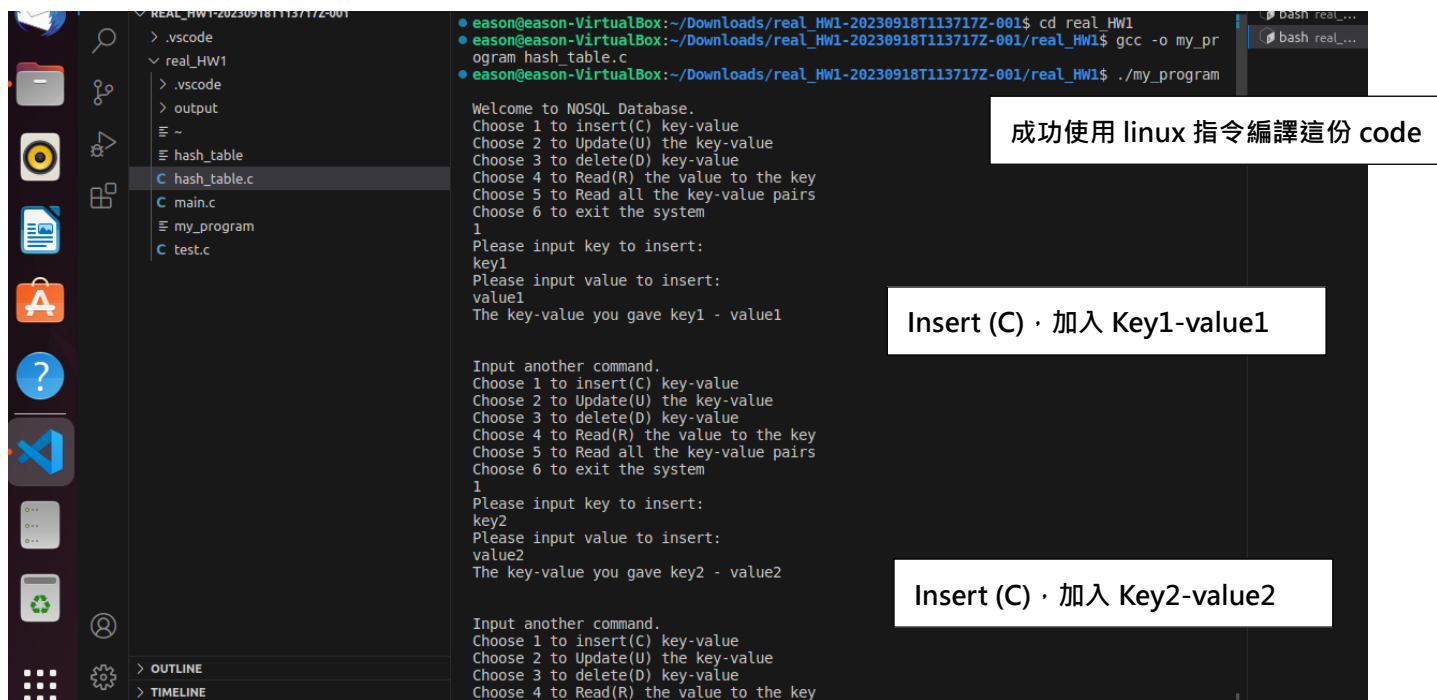
我使用的 hash function 首先初始化一個名為 hash 的無符號長整數並設置初始值為 5381，遍歷 key 中的每個字符，對 hash 值進行一系列位家法以生成一個唯一的哈希碼。最後使用 % 100 運算將哈希碼限制在範圍 0 到 99 之間，將 key 映射到哈希表中的某個 index，這個索引位置在 TB\_array 中用於存儲相關的鍵值對。這個哈希函數的目標是均勻地將不同的鍵分散到不同的索引位置，以提高查找效率(O(1))。

```

int hash(char* key) {
    unsigned long hash = 5381;
    int c;
    while ((c = *key++)) {
        hash = ((hash << 5) + hash) + c;
    }
    return hash % 100;
}

```

### 3. 在 Linux Ubuntu 上的編譯執行結果:



```
eason@eason-VirtualBox:~/Downloads/real_HW1-20230918T113717Z-001$ cd real_HW1
eason@eason-VirtualBox:~/Downloads/real_HW1-20230918T113717Z-001/real_HW1$ gcc -o my_program hash_table.c
eason@eason-VirtualBox:~/Downloads/real_HW1-20230918T113717Z-001/real_HW1$ ./my_program

Welcome to NOSQL Database.
Choose 1 to insert(C) key-value
Choose 2 to Update(U) the key-value
Choose 3 to delete(D) key-value
Choose 4 to Read(R) the value to the key
Choose 5 to Read all the key-value pairs
Choose 6 to exit the system
1
Please input key to insert:
key1
Please input value to insert:
value1
The key-value you gave key1 - value1

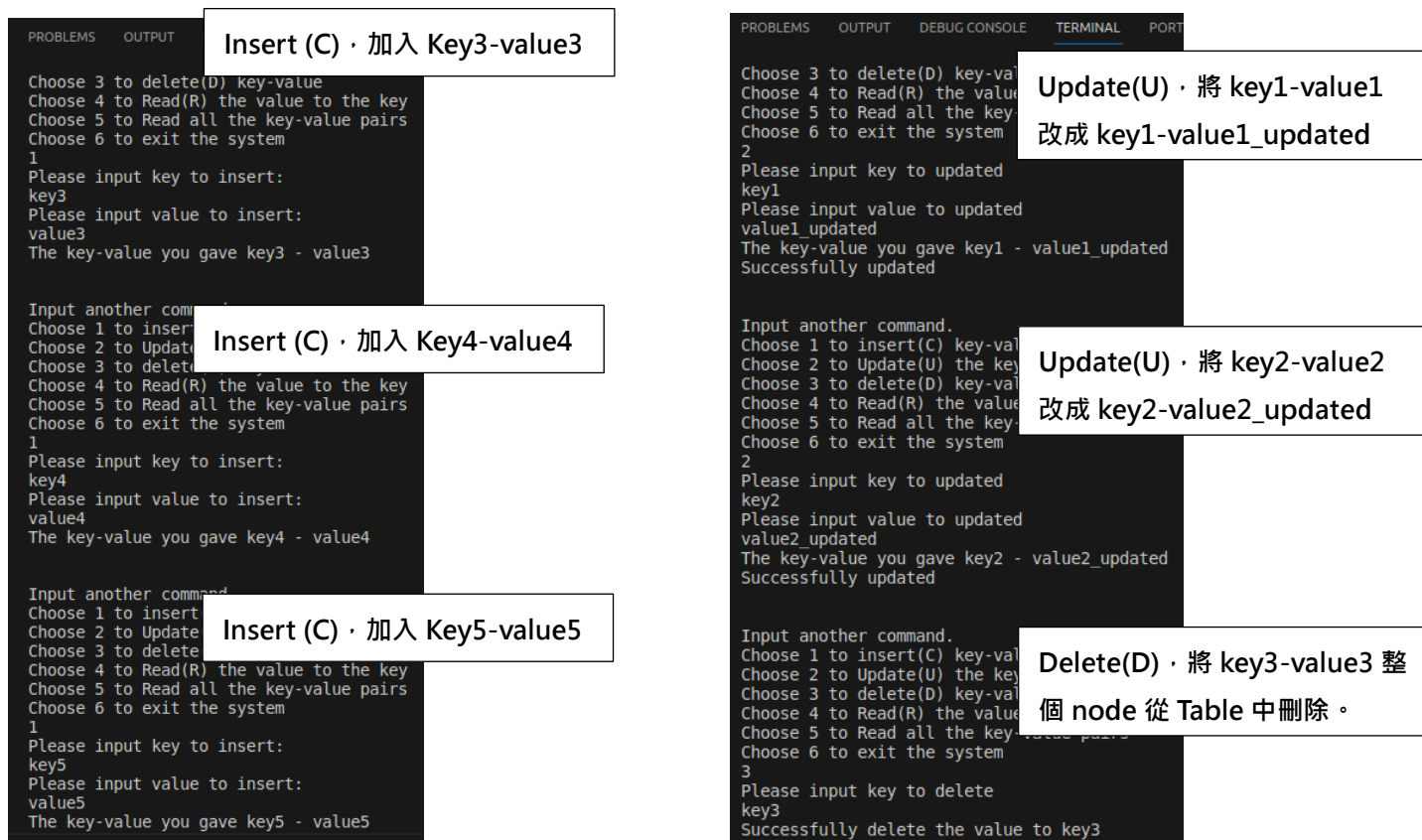
Input another command.
Choose 1 to insert(C) key-value
Choose 2 to Update(U) the key-value
Choose 3 to delete(D) key-value
Choose 4 to Read(R) the value to the key
Choose 5 to Read all the key-value pairs
Choose 6 to exit the system
1
Please input key to insert:
key2
Please input value to insert:
value2
The key-value you gave key2 - value2

Input another command.
Choose 1 to insert(C) key-value
Choose 2 to Update(U) the key-value
Choose 3 to delete(D) key-value
Choose 4 to Read(R) the value to the key
```

成功使用 linux 指令編譯這份 code

Insert (C) · 加入 Key1-value1

Insert (C) · 加入 Key2-value2



```
Choose 3 to delete(D) key-value
Choose 4 to Read(R) the value to the key
Choose 5 to Read all the key-value pairs
Choose 6 to exit the system
3
Please input key to insert:
key3
Please input value to insert:
value3
The key-value you gave key3 - value3

Input another command.
Choose 1 to insert(C) key-value
Choose 2 to Update(U) the key-value
Choose 3 to delete(D) key-value
Choose 4 to Read(R) the value to the key
Choose 5 to Read all the key-value pairs
Choose 6 to exit the system
2
Please input key to updated:
key1
Please input value to updated:
value1_updated
The key-value you gave key1 - value1_updated
Successfully updated

Input another command.
Choose 1 to insert(C) key-value
Choose 2 to Update(U) the key-value
Choose 3 to delete(D) key-value
Choose 4 to Read(R) the value to the key
Choose 5 to Read all the key-value pairs
Choose 6 to exit the system
2
Please input key to updated:
key2
Please input value to updated:
value2_updated
The key-value you gave key2 - value2_updated
Successfully updated

Input another command.
Choose 1 to insert(C) key-value
Choose 2 to Update(U) the key-value
Choose 3 to delete(D) key-value
Choose 4 to Read(R) the value to the key
Choose 5 to Read all the key-value pairs
Choose 6 to exit the system
3
Please input key to delete:
key3
Successfully delete the value to key3
```

Insert (C) · 加入 Key3-value3

Update(U) · 將 key1-value1 改成 key1-value1\_updated

Update(U) · 將 key2-value2 改成 key2-value2\_updated

Delete(D) · 將 key3-value3 整個 node 從 Table 中刪除。

PROBLEMS OUTPUT DEBUG

```
Choose 3 to delete(D) key-value
Choose 4 to Read(R) the value to the key
Choose 5 to Read all the key-value pairs
Choose 6 to exit the system
3
Please input key to delete
key7
We didn't find the value to key7
```

Delete(D) · 將 key7-value7 刪除，但這個 node 不存在所以顯示不存在

```
Input another command.
Choose 1 to insert(C) key-value
Choose 2 to Update(U) the key-value
Choose 3 to delete(D) key-value
Choose 4 to Read(R) the value to the key
Choose 5 to Read all the key-value pairs
Choose 6 to exit the system
4
Please input key to read its value
key1
The value to key1 is value1_updated
```

Read(R) · 查看 key1 對應到的 value，結果為 value1\_updated

```
Input another command.
Choose 1 to insert(C) key-value
Choose 2 to Update(U) the key-value
Choose 3 to delete(D) key-value
Choose 4 to Read(R) the value to the key
Choose 5 to Read all the key-value pairs
Choose 6 to exit the system
4
Please input key to read its value
key4
The value to key4 is value4
```

Read(R) · 查看 key4 對應到的 value，結果為 value4，結果正確

```
Input another command.
Choose 1 to insert(C) key-value
Choose 2 to Update(U) the key-value
Choose 3 to delete(D) key-value
```

The value to key4 is value4

```
Input another command.
Choose 1 to insert(C) key-value
Choose 2 to Update(U) the key-value
Choose 3 to delete(D) key-value
Choose 4 to Read(R) the value to the key
Choose 5 to Read all the key-value pairs
Choose 6 to exit the system
5
All the key-value pairs
key1 value1_updated
key2 value2_updated
key4 value4
key5 value5
```

查看所有 pair，測試結果正確

```
Input another command.
Choose 1 to insert(C) key-value
Choose 2 to Update(U) the key-value
Choose 3 to delete(D) key-value
Choose 4 to Read(R) the value to the key
Choose 5 to Read all the key-value pairs
Choose 6 to exit the system
4
Please input key to read its value
key5
The value to key5 is value5
```

Read(R) · 查看 key5 對應到的 value，結果為 value5，結果正確

```
Input another command.
Choose 1 to insert(C) key-value
Choose 2 to Update(U) the key-value
Choose 3 to delete(D) key-value
Choose 4 to Read(R) the value to the key
Choose 5 to Read all the key-value pairs
Choose 6 to exit the system
6
```

eason@eason-VirtualBox:~/Downloads/real\_HW1-2

感謝助教用心批改!!!!