Project I Report

By Devinesh Singh

Introduction

Project I consist of two parts: (1) provide the user with metadata about the schema of the table and (2) reverse the order of the results from a query (ORDER BY) – results are returned in ascending order, but instead we want them to be returned in descending order by default.

Part 1

Insert.c was the file that required changes to the source code. Inside Insert.c, there is a function sqliteInsert that contains the following error message: "table %S has %d columns but %d values were supplied" – a simple search helped find this line. At first, when I wrote the print statement above the error message, the console printed it below the error message. Sqlite3Insert provided comments to help understand which variables/pointers would allow us to see all the column names and table. Code is provided below:

```
printf("Metadata help: %s{", pTab->zName);
for (i = 0; i < pTab->nCol; i++)
    {
     if (i < pTab->nCol - 1)
     {
       printf("%s, ", pTab->aCol[i].zName);
     }
     else
     {
       printf("%s", pTab->aCol[i].zName);
     }
    }
    printf("}\n");
```

Part II:

Part II was a bit tricky – upon initial review, it seemed the issue lied within the aSortOrder variable, but after several changes, this didn't seem to make a difference. In class, Professor Gill provided a hint: we must make changes to a header file. So, I found that sqliteInt contained the following variables: SQLITESO_ASC and SQLITESO_DESC which are for

SortOrder. Additionally, the clue on the project description said we need to be able to work with defines. After a quick search, I found that these variables control the sort order – and a simple swap between 0 and 1 was the solution to this problem.

```
#define SQLITE_SO_ASC 1       /* Sort in ascending order */
#define SQLITE_SO_DESC 0      /* Sort in ascending order */
#define SQLITE_SO_UNDEFINED -1 /* No sort order specified */
```

Results:

Below, you can find results for both parts (1) and (2).

```
Use ".open FILENAME" to reopen on a persistent database.
sqlite> create table cs157a(sjsu_id int, name varchar(5));
sqlite> insert into cs157a values(1050,'Mary');
sqlite> insert into cs157a values(1001,'Alice');
sqlite> insert into cs157a values(1200,'Trudy');
sqlite> insert into cs157a values(1001,'John');
sqlite> insert into cs157a values(2050,'Judy', 'A+');
Metadata help: cs157a{sjsu_id, name}
Error: table cs157a has 2 columns but 3 values were supplied
sqlite> select * from cs157a order by sjsu_id;
1200|Trudy
1050|Mary
1001|Alice
1001|John
sqlite>
```

Build Process:

SQLite Source Repository website provided details on how to process a custom build of the code. The process was as follows:

1. Make a directory called 'bld'
    a. mkdir bld
2. go into the bld directory
    a. cd bld
3. run ../sqlite/configure
4. run make

upon completing the commands above, the code was ready to run with ./sqlite3