CS-157A-Sec 01: Database Management System:- Spring 2021

Project 1 Report: -

Prepared By: Team 23
Nimay Patel (013720616),
Date: 04/01/2021

Research: -

First of all, I watched the instructor's demo on how to build sqlite3 from source code.Then, I followed his instruction and intended to do the project on my Mac environment without any virtual machine. - I downloaded the source code from the website shown in the demo video and extracted it. After extracting it, I used the files in the src folder to complete the tasks.

In there, I found lots of C files and I just happened to see the insert.c file. I started working on it from that point to the end of the project since the change requested in problem 1 relates to how errors are handled when insertion goes wrong.

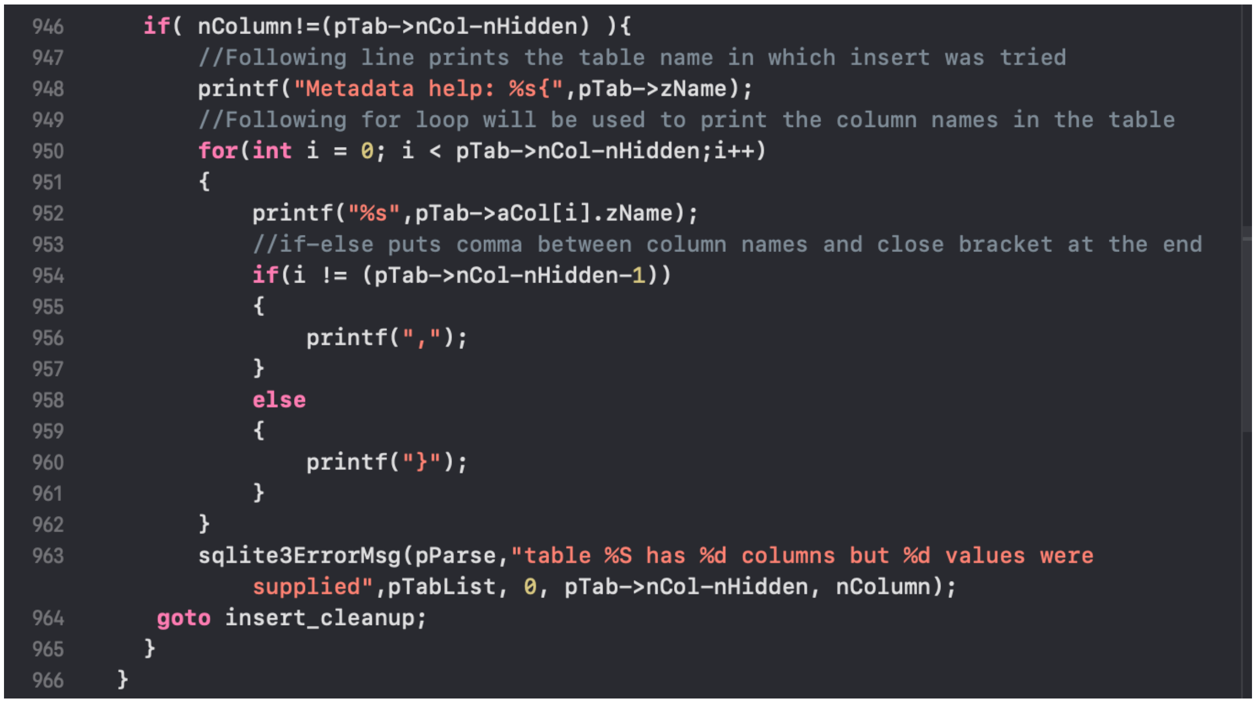
After spending some time researching, I finally found the exact line that pops the error message in standard sqlite. On lines 947 and 948, I found `sqlite3ErrorMsg(pParse,"table %S has %d columns but %d values were supplied", pTabList, 0, pTab->nCol-nHidden, nColumn);`

Based on variables, pTab,nCol and pTabList, I used Cmd + F to search these terms and was able to find the values that these variables carried. SInce the file was too long, I did not focus on understanding every line of code but I skimmed through the whole file and mostly focused on variables and parameters defined in those particular struct that were useful to me in this project.

For problem 2, I tried to find #define statements and C structures that define order by and its functionalities. Fortunately, because of my CS-149 class of Operating Systems, I already knew how effectively we can make user defined header files and include them in normal C files and I was 95% sure that since this is a built in very commonly used functionality, it must've been implemented in a header file. I spent a lot of time going through most of them and eventually found a couple of references to this in sqlitInt.h file.

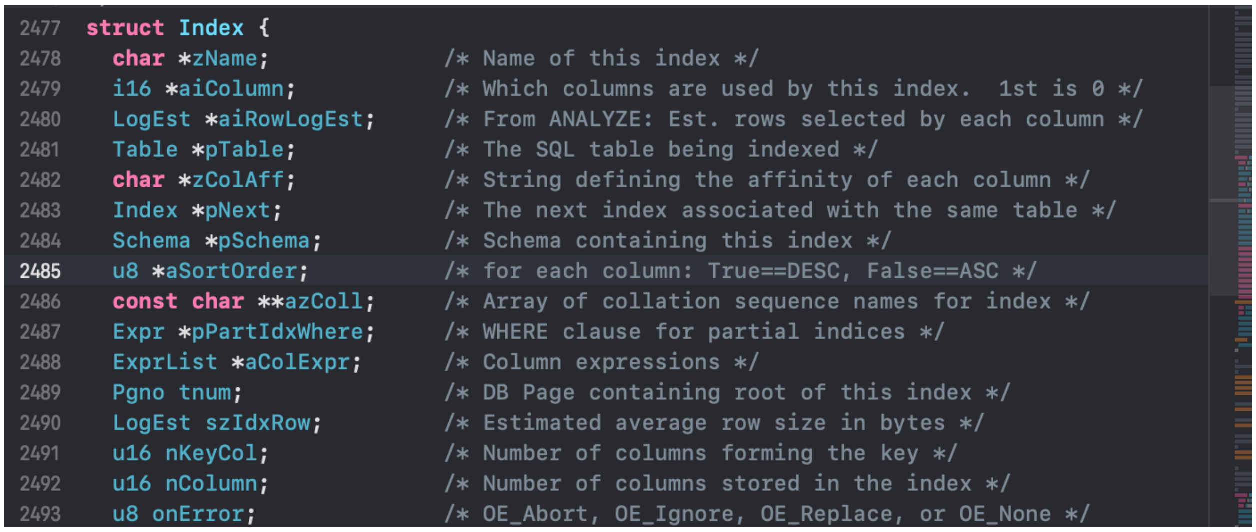
Code Changes: -

For problem 1, the name is stored in variable zName which gets referenced by pointer pTab whereas for column name, pointer *aCol is used which is of Column data type and refers to a list of all columns in a particular table pointed by pTab. So the change that I made for problem 1 is:



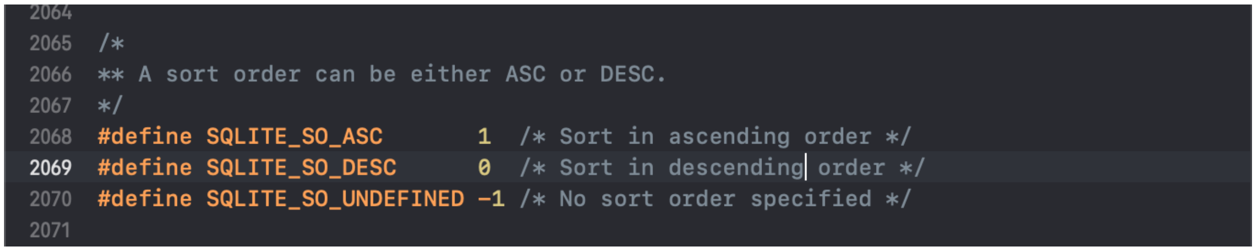
```
     if( nColumn!=(pTab->nCol-nHidden) ){
       //Following line prints the table name in which insert was tried
       printf("Metadata help: %s{",pTab->zName);
       //Following for loop will be used to print the column names in the table
       for(int i = 0; i < pTab->nCol-nHidden;i++)
       {
         printf("%s",pTab->aCol[i].zName);
         //if-else puts comma between column names and close bracket at the end
         if(i != (pTab->nCol-nHidden-1))
         {
           printf(",");
         }
         else
         {
           printf("}");
         }
       }
       sqlite3ErrorMsg(pParse,"table %S has %d columns but %d values were supplied",pTabList, 0, pTab->nCol-nHidden, nColumn);
      goto insert_cleanup;
     }
    }
```

For solving problem 2, changes were made to the sqlInt.h file. While researching, I found that line 2485 of sqliteInt.h file declares a pointer which will carry a value of true if order is descending and false if the value is ascending.



```
2477 struct Index {
2478   char *zName;             /* Name of this index */
2479   i16 *aiColumn;           /* Which columns are used by this index.  1st is 0 */
2480   LogEst *aiRowLogEst;     /* From ANALYZE: Est. rows selected by each column */
2481   Table *pTable;           /* The SQL table being indexed */
2482   char *zColAff;           /* String defining the affinity of each column */
2483   Index *pNext;            /* The next index associated with the same table */
2484   Schema *pSchema;         /* Schema containing this index */
2485   u8 *aSortOrder;          /* for each column: True==DESC, False==ASC */
2486   const char **azColl;     /* Array of collation sequence names for index */
2487   Expr *pPartIdxWhere;     /* WHERE clause for partial indices */
2488   ExprList *aColExpr;      /* Column expressions */
2489   Pgno tnum;               /* DB Page containing root of this index */
2490   LogEst szIdxRow;         /* Estimated average row size in bytes */
2491   u16 nKeyCol;             /* Number of columns forming the key */
2492   u16 nColumn;             /* Number of columns stored in the index */
2493   u8 onError;              /* OE_Abort, OE_Ignore, OE_Replace, or OE_None */
```

Since binary value 0 means False and 1 means True in C, I tried to find lines of code that predefine ascending order as 1 and descending order as 0 which can only be changed by manually specifying otherwise. After spending a lot of time, I found that lines 2068 and 2069 do the exact same. Hence, I exchanged both of their values and predefined Descending order as a defacto sorting order for all select statements unless specified otherwise. Following is the change I made in sqliteInt.h header file.



```
/*
** A sort order can be either ASC or DESC.
*/
#define SQLITE_SO_ASC       1  /* Sort in ascending order */
#define SQLITE_SO_DESC      0  /* Sort in descending order */
#define SQLITE_SO_UNDEFINED -1 /* No sort order specified */
```

By doing this, I made sure that when aSortOrder becomes true, the sorting happens in ascending order rather than descending and vice versa when aSortOrder becomes false.

Build Process: -

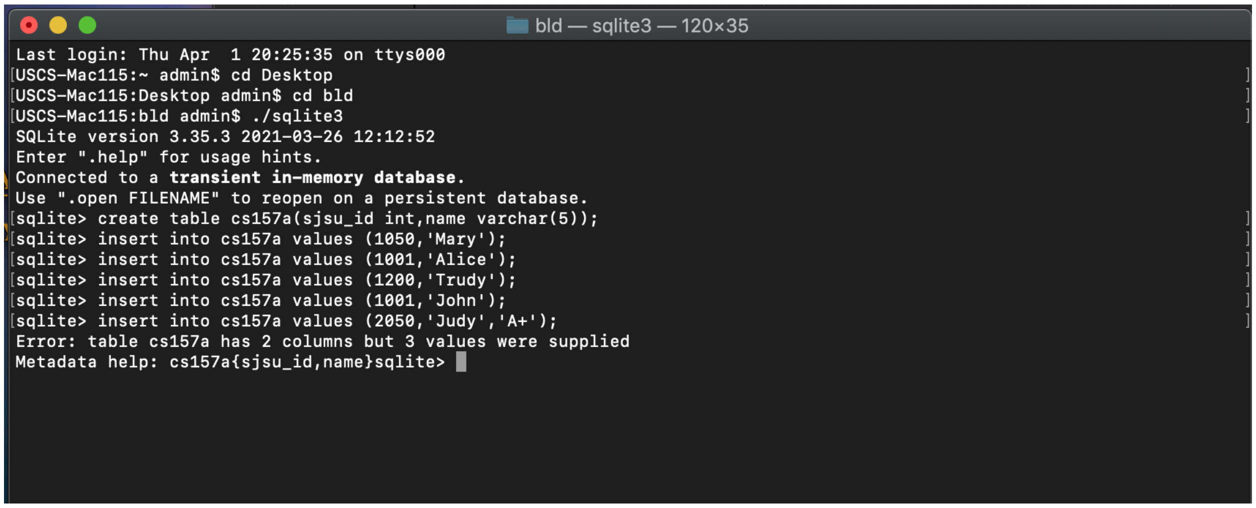
For the build process, I completely relied on the professor's demo and followed all steps from his demo. I first made a directory called 'bld' on Desktop where my unzipped sqlite folder was also there using commands:
mkdir bld
cd bld
Then, I configured the sqlite by
../sqlite/configure
make
When I tried to make them, I ran into an error because of a syntax error. So I went back and solved them and then I reconfigured and remade the files.
For opening the custom built sqlite version, I went to 'bld' directory and typed: ./sqlite3.
After this, the latest modified version with today's date started and that's where I performed the tests.

Unit Test Results:

For problem 1, I created a table called cs157a with column names: sjsu_id and name. Then, I performed some insertion operations and entered legitimate values in the table. Then, I tried to pass an insert statement with 3 values while the table only had 2 columns and that is when I was able to see the following output which is exactly what it was supposed to be for the test case to pass.

For problem 2, I tried to pass a select statement, without specifying that I wanted sorting to happen in descending order. Yet, just because we are using the modified version of sqlite, it sorted all of them by sjsu_id in descending order by default. Now, whenever that version of sqlite is used and someone wants sorting in ascending order, they will have to specify "ASC".

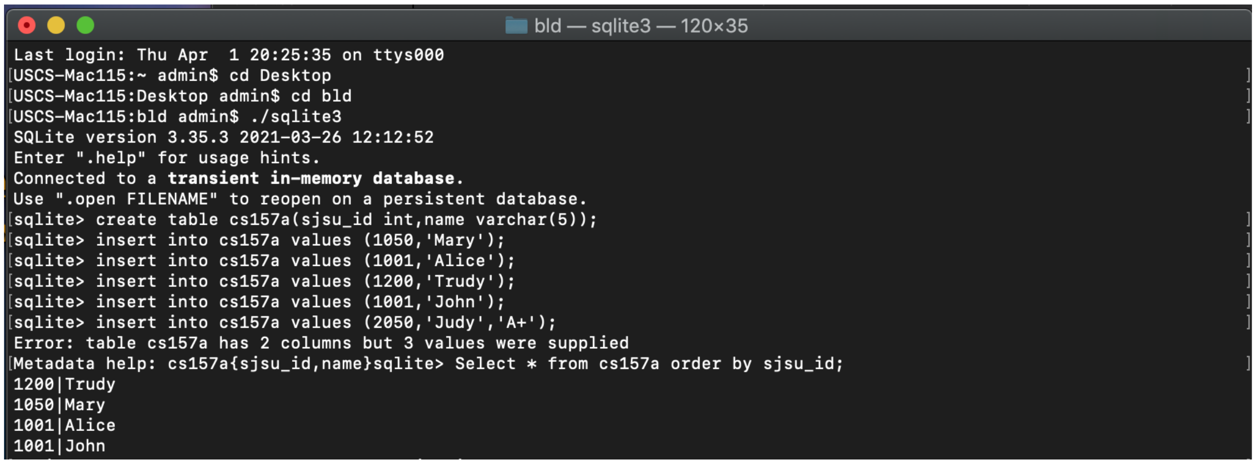
Test case Result for Problem 1:



bld — sqlite3 — 120×35

```
Last login: Thu Apr  1 20:25:35 on ttys000
USCS-Mac115:~ admin$ cd Desktop
USCS-Mac115:Desktop admin$ cd bld
USCS-Mac115:bld admin$ ./sqlite3
SQLite version 3.35.3 2021-03-26 12:12:52
Enter ".help" for usage hints.
Connected to a transient in-memory database.
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+');
Error: table cs157a has 2 columns but 3 values were supplied
Metadata help: cs157a{sjsu_id,name}sqlite> 
```

Test Case Result for Problem 2:



bld — sqlite3 — 120×35

```
Last login: Thu Apr  1 20:25:35 on ttys000
USCS-Mac115:~ admin$ cd Desktop
USCS-Mac115:Desktop admin$ cd bld
USCS-Mac115:bld admin$ ./sqlite3
SQLite version 3.35.3 2021-03-26 12:12:52
Enter ".help" for usage hints.
Connected to a transient in-memory database.
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+');
Error: table cs157a has 2 columns but 3 values were supplied
Metadata help: cs157a{sjsu_id,name}sqlite> Select * from cs157a order by sjsu_id;
1200|Trudy
1050|Mary
1001|Alice
1001|John
```