University of Windsor

School of Computer Science
Faculty of Science
COMP-3150: Database Management System (Fall 2022)

Lab#	Date	Title	Due Date	Grade Release Date
Lab5	Week 10	**Database Programming**	Two-Week Lab November 30, 2022, Wednesday Midnight EDT	Dec. 05, 2022

The objective of this lab is to enable you to do database programming, that is, how databases can be accessed from any programming language. For the sake of this lab, we will be focusing on C programming language.

You can write a C program that allows the users to perform the list of operations that you had conceptualized in Lab1 and operationalized through SQL commands in Lab4 through a C program.

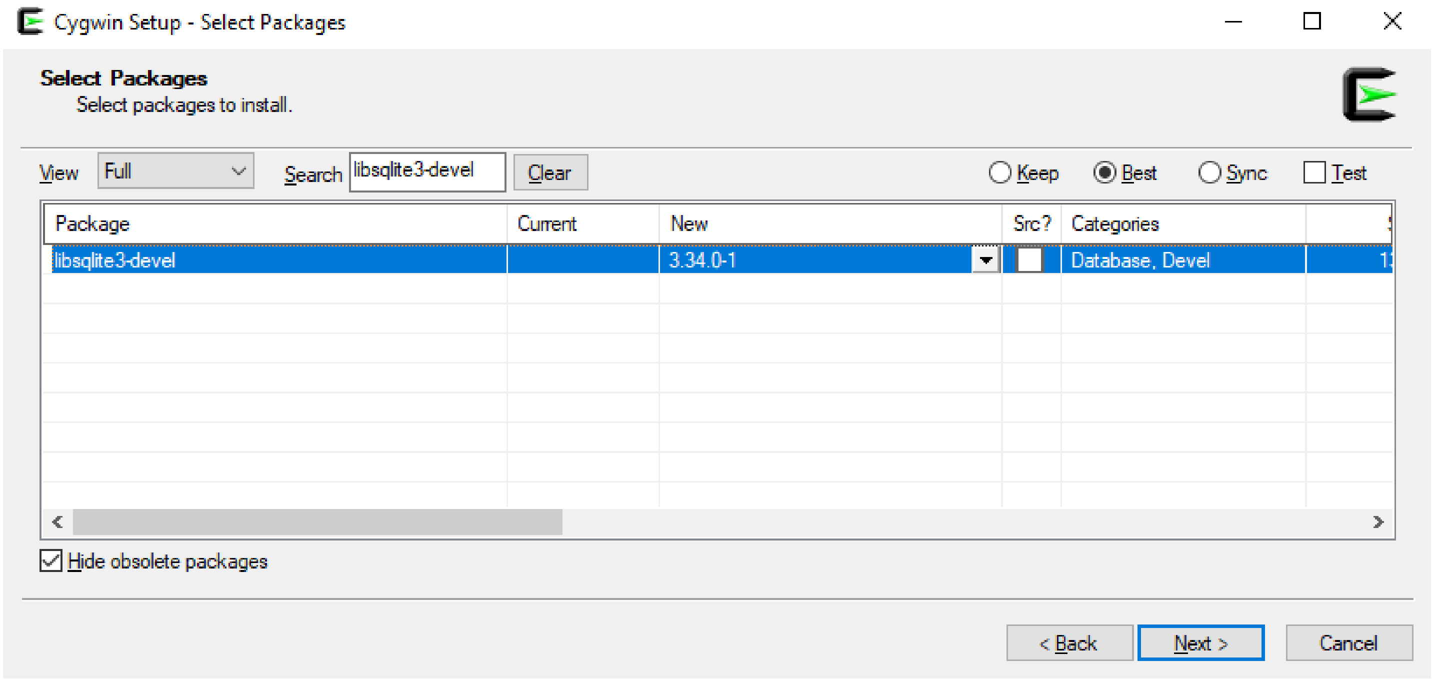
Step 1. Database Connector

Before you can write a C program that accesses the database, you will need the driver for accessing your database in SQLite. This is called the sqlite3 driver. You can install the right driver:

In Linux:

```
$ sudo apt-get install libsqlite3-dev
```

In Windows, you can use Cygwin setup:



Cygwin Setup - Select Packages

Select Packages
Select packages to install.

View: Full — Search: libsqlite3-devel — Keep / [✓ Best] / Sync / Test

Package	Current	New	Src?	Categories
libsqlite3-devel		3.34.0-1		Database, Devel

[✓] Hide obsolete packages

< Back — Next > — Cancel

Step 2. Integrated Development Environment (IDE)

You need to create a new C project in your IDE of choice, NetBeans[1], CodeBlocks[2], or any other environment. You can also use a text editor and build your C program using C compiler, cc. To be able to include SQL statements in your C code, you need to include the following library in your code:

```
#include <sqlite3.h>
```

[1] https://netbeans.apache.org/download/index.html
[2] https://www.codeblocks.org/downloads/

You will have to first connect to the database and ensure that the connection is established without any problems. This can be achieved by:

```
#include <stdio.h>
#include <sqlite3.h>

int main(int argc, char* argv[]) {

  sqlite3* db;
  int c;

  c = sqlite3_open_v2("/home/Moviesion.db", &db, SQLITE_OPEN_READWRITE, NULL);

  if(c != SQLITE_OK) {
    printf("Can't open database! %s\n", sqlite3_errmsg(db));
    return(0);
  } else {
    printf("Connect to database successfully.\n");
  }
  sqlite3_close(db);
}
```

In order to compile and build the program, either you use the IDE's build option, or C compiler `cc` with sqlite3 library `-l sqlite3`:

```
$ gcc Main.c -o Main -l sqlite3
```

You can find more help here:
https://zetcode.com/db/sqlitec/
https://www.tutorialspoint.com/sqlite/sqlite_c_cpp.htm

Step 3. Program to Database

Once you have created a connection to the database, you can access the tables through appropriate C commands and send your SQL queries. Depending on the SQL query, you will receive appropriate data or response back from the database that you can use in your code. Let's assume you want to create a new table:

```
//code to open database goes here

char* errMsg = 0;
//create SQL statement
sql = "CREATE TABLE Movie("  \
    “Id INTEGER PRIMARY KEY NOT NULL,” \
    “Title VARCHAR(255) NOT NULL,” \
    “...);”;

//execute SQL statement
int r = sqlite3_exec(db, sql, NULL, 0, &errMsg);

if( r != SQLITE_OK ){
   printf("SQL error: %s\n", errMsg);
   sqlite3_free(errMsg);
}
```

```
else {
   printf("Table created successfully\n");
}
sqlite3_close(db);
return 0;
```

The sql variable will need to be populated with the appropriate CREATE TABLE query. Similarly, an INSERT statement in C can be performed by using the same code snippet but a different SQL query as follows:

```
//code to open database goes here

//create SQL statement
sql = "INSERT INTO Movie(Id, Title) VALUES(1, ‘Psycho’);”;

//code to execute SQL statement
```

An UPDATE or DELETE operation in SQL can be done quite similarly by replacing sql variable with a proper SQL query. Please make sure to include c.close() and s.close() after you have done inserting your data and you are done with using a database connection.

Now, a SELECT statement that retrieves some data has some more details that we need to explore. First, we need to create a callback function which will be called by each row that matches the SELECT statement:

```
static int callback(void* data, int argc, char** argv, char** ColName){

  for(int i = 0; i<argc; i++){
    printf("%s = %s\n", ColName[i], argv[i]);
  }

  return 0;
}
```

Now, you have to send the statement to the database by executing it:

```
//create SQL statement
sql = "SELECT * from Movie";

//execute SQL statement
r = sqlite3_exec(db, sql, callback, 0, &errMsg);

if( r != SQLITE_OK ){
   printf("SQL error: %s\n", errMsg);
   sqlite3_free(errMsg);
}
else {
   printf("Movies are retrieved successfully\n");
}
sqlite3_close(db);
return 0;
```

Final Step. Deliverables

For this lab, you will need to fully implement all the queries that you had designed in Lab4 through a C program. When the program runs, it should show a prompt with all the possible actions that the user can execute. For instance,

```
$ Welcome to the moviesion application:
    0)  Exit
    1)  List of movies
    2)  List of directors
    3)  Add a new movie
    4)  ...
```

Once the user chooses an option, the program should execute the appropriate SQL command and retrieve the results for the user. If needed, for example in the case of entering a new movie, the program should ask the user for additional information needed to create the new movie. For example

```
please enter the title of the movie:
Inception

please enter the name of the director:
Christopher Nolan
```

Once the operation is completed the program should provide the user with the retrieved appropriate data or message and return back to the original program menu and wait for a new command by the user.

You should submit the following items in one single zip file Lab5_uwinid.zip:

(70%) Lab5_uwinid.zip

- (10%) DatabaseName.db → The database file whose tables have at least 5 rows. Instead of DatabaseName, use your own database name, e.g., mine is Movision, so, Movision.db.

- (50%) Main.c → The source code of the program which include the functions to accomplish each task of the application. <u>Remember, you might need more than one SQL statement for one task.</u> <u>Respect code convention when writing C codes and SQL statements</u> as we did above.

- (10%) Report.pdf → Lab report including name, student id, and the snapshots your program when run and the data that is returned. Make sure you include one screenshot for each of the options in your program's main menu. For instance, one screenshot when the user asked for the list of movies, one screenshot for when the user asked for all directors, one for when the user entered a new movie etc.

- (Optional) ReadMe.txt → Optional additional information that helps lab instructor or grader to evaluate.

(30%) Naming (PascalCase) and Formats

Instead of uwinid, use your own account name, e.g., mine is hfani@uwindsor.ca, so, Lab5_hfani.zip