# LAB 7 – TASK 12
# Database Programming in C

**John Dempsey**
COMP-232: Programming Languages
California State University, Channel Islands
October 9, 2024
Hard Due Date: October 16, 2024

## Task 12. Database

MySQL supports a C application programming interface (API) which allows you to write C programs to run SQL queries against a MySQL database. Your MySQL database is called **classicmodels** and is on comp232.com. The Entity-Relationship Diagram (ER Diagram) for the classicmodels database is seen below in Figure 1.

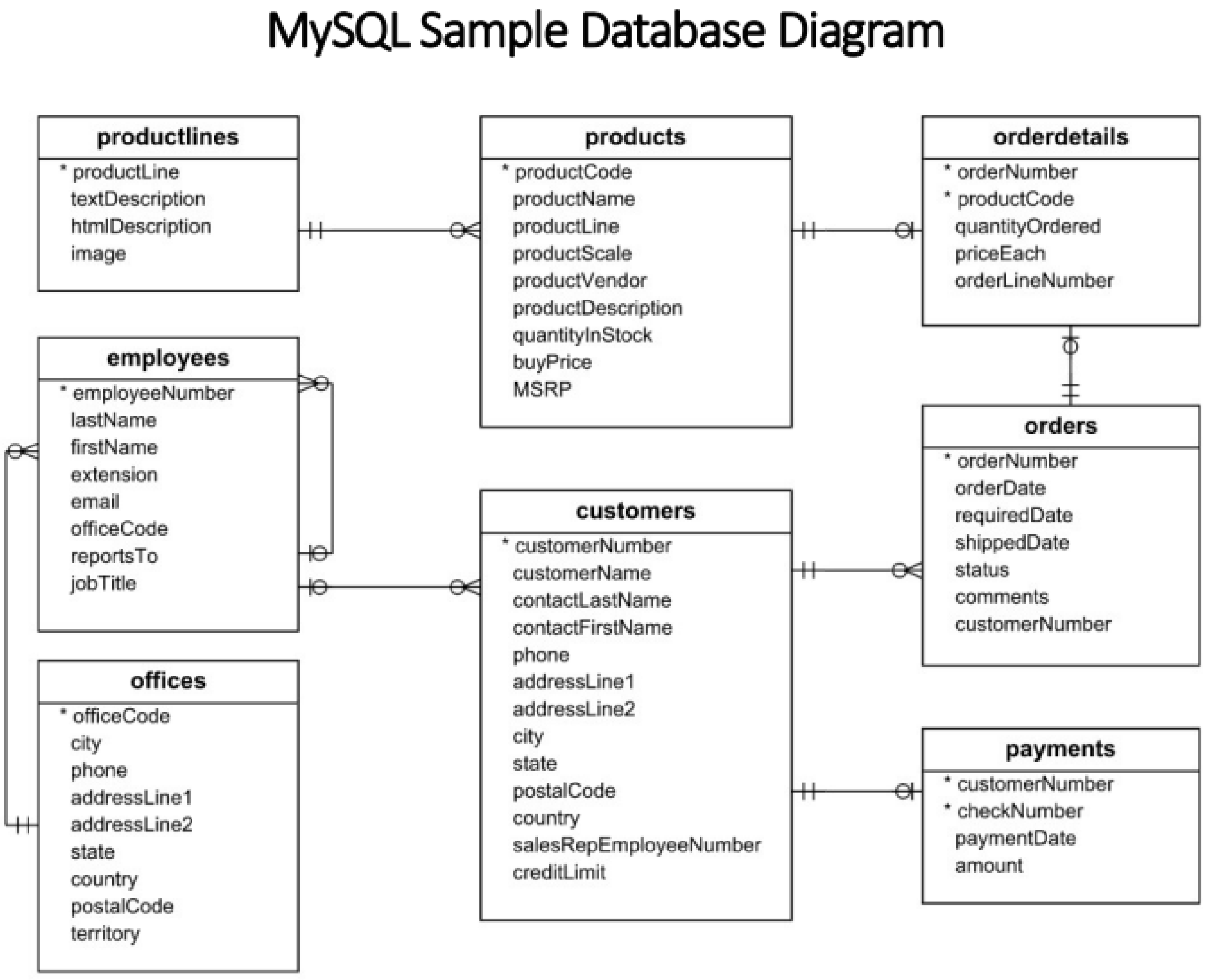

**Figure 1. ER Diagram for the classicmodels Database**

On your laptop, run:

```
% mkdir LAB7
% cd LAB7
% sftp john@comp232.com
% cd /home/LAB7
% mget *
% quit
% cat run
gcc client.c -o client `mysql_config --cflags --libs`
% mysql_config --cflags
-I/usr/include/mysql
% mysql_config --libs
-L/usr/lib/x86_64-linux-gnu -lmysqlclient -lpthread -ldl -lz -lssl -lcrypto -lresolv -lm -lrt
```

If you're a MacOS user, edit your ~/.profile file and add/update the following line:

```
PATH=$PATH:/opt/homebrew/opt/mysql-client/bin
```

To compile your client.c program, type:

```
% run
```

Running the above run command actual runs the following command:

```
% gcc client.c -o client -I/usr/include/mysql -L/usr/lib/x86_64-linux-gnu \
      -lmysqlclient -lpthread -ldl -lz -lssl -lcrypto -lresolv -lm -lrt
```

The client.c program below runs the query:

```
SELECT officeCode, city, state FROM offices;
```

**The client.c program follows:**

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <mysql/mysql.h>                 // MacOS users use: #include <mysql.h>

int main(int argc, char **argv)
{
      MYSQL             *con;
      int                count;
      int                number_of_rows;
      char               query[1000];
      MYSQL_RES         *result;
      MYSQL_ROW          row;

      printf("\nMySQL client version is: %s\n", mysql_get_client_info());

      con = mysql_init(NULL);

      if (con == NULL) {            // Allocates memory and returns a pointer to MYSQL structure.
        fprintf(stderr, "mysql_init() failed.\n");
        exit(1);
      }

      // mysql_real_connect establishes a connection from your client to the database server
      // running on comp232.com.  The parameters are con for connection, comp232.com
      // (or IP address), your user name, your password, the database id, followed by 0, NULL, 0.

      if (mysql_real_connect(con, "comp232.com", "class", "CSUCIcomp232$", "classicmodels",
                             0, NULL, 0) == NULL) {
        fprintf(stderr, "%s\n", mysql_error(con));
        mysql_close(con);
        exit(1);
      }
      printf("The connection is open.\n");       // The connection was successful.

      // Query 1: SELECT officeCode, city, state FROM offices;

      strcpy(query, "SELECT officeCode, city, state FROM offices");

      if (mysql_query(con, query) != 0)          // mysql_query runs the query.
```

```
    {
      printf("Query failed.\n");
      printf("%s\n", mysql_error(con));
      exit(1);
    }

    result = mysql_store_result(con);        // mysql_store_result returns buffered results from query.

    printf("Number of rows: %ld\n", (long) mysql_num_rows(result));

    count = 1;
    while((row = mysql_fetch_row(result)) != NULL) {
      printf("%2d. ", count);
      printf("officeCode = %s, ", row[0]);
      printf("city = %-15s, ",      row[1]);
      printf("state = %s\n",        row[2]);
      count = count + 1;
    }
    printf("\n\n");

    mysql_free_result(result);        // mysql_free_result frees memory used by buffered results.

    mysql_close(con);                 // mysql_close closes the connection to the server.

    printf("The connection is closed.\n");
}
```

```
john@oho:~/MYSQL$ client

MySQL client version is: 8.0.29

The connection is open.

Number of rows: 7
 1. officeCode = 1, city = San Francisco  , state = CA
 2. officeCode = 2, city = Boston         , state = MA
 3. officeCode = 3, city = NYC            , state = NY
 4. officeCode = 4, city = Paris          , state = (null)
 5. officeCode = 5, city = Tokyo          , state = Chiyoda-Ku
 6. officeCode = 6, city = Sydney         , state = (null)
 7. officeCode = 7, city = London         , state = (null)

The connection is closed.
```

## Your Task

Your task is to copy and update the client.c to implement and display the following results:

**Using the classicmodels database, make a copy of client.c to sf.c. Copy run to runsf.**

1. Display the total number of employees.
2. Display the firstname, lastname, jobtitle, and email for all employees. Order list by firstname.
3. Use query: **select firstname, lastname, jobtitle, email from employees order by firstname**
4. Output for the first few rows will look like:

```
Number of rows: 23
 1. Andy Fixter, Sales Rep, afixter@classicmodelcars.com
 2. Anthony Bow, Sales Manager (NA), abow@classicmodelcars.com
 3. Barry Jones, Sales Rep, bjones@classicmodelcars.com
 4. Diane Murphy, President, dmurphy@classicmodelcars.com
```

SOAR
SAVE OPEN SPACE & AGRICULTURAL RESOURCES

### Using the soar database, answer the following question:

### Did the Camarillo SOAR collect 10% or more signatures of registered voters needed to qualify their petition to be placed on the ballot?

To do this task, you'll need to:

1. Copy run to runsoar
2. Copy client.c to soar.c
3. Change the connection to comp232.com to access the **soar** database.
4. Run a query to count the number of voters in Camarillo who signed the petition, i.e., query is:
   **select * from voters_camarillo where signed_flag = 'Y'**
5. Run a query to count the total number of voters in Camarillo.
6. Divide the number of voters who signed the petition divided by total number of voters.
7. Output should look like if 10% or more of the voters signed the petition:

```
9999 out of 999999 voters, or 99.99% of all voters, signed the SOAR petition.
SOAR can be placed on the ballot.
```

**To receive credit for LAB 7 TASK 12, run the following two commands:**

```
% sf > task12.txt                // Creates task12.txt

% soar >> task12.txt             // Concatenates soar output to task12.txt file.
```

**Upload sf.c, soar.c, and task12.txt to comp232.com in your LAB7/TASK12 directory.**