CSC343 Worksheet 7 Solution

June 22, 2020

1. a)

```
void askUserForPrice() {
    EXEC SQL BEGIN DECLARE SECTION;
        int model;
        float speed;
        int ram;
        int hd;
        float price;
        char maker;
        float targetPrice;

        float minDiff;
        int modelSol;
        float speedSol;
        char makerSol;
    EXEC SQL END DECLARE SECTION;

    EXEC SQL DECLARE execCursor CURSOR FOR
        SELECT * FROM Product NATURAL JOIN PC

    EXEC SQL OPEN execCursor;

    printf("Enter target price:");
    scanf("%f", &targetPrice);

    while(1) {
        EXEC SQL FETCH FROM execCursor INTO :model,
            :speed, :ram, :hd, :price, :maker;

        if (NO_MORE_TUPLES) break;

        if (abs(price - targetPrice) >= minDiff) {
            continue;
        }

        minDiff = abs(price - targetPrice);
        modelSol = model;
        speedSol = speed;
        makerSol = maker;
    }
```

```
40
41          EXEC SQL CLOSE execCursor;
42
43          printf("maker=%c, model=%d, speed=%.2f\n", makerSol, modelSol
    , speedSol);
44
45      }
46
```

Notes:

- EXEC SQL
  - Allows to use SQL statements within a host-language program
- The DECLARE Section
  - is used to declare shared variables
  - **Syntax:**
    EXEC SQL BEGIN DECLARE SECTION;
    ... // Variable declarations in any language
    EXEC SQL END DECLARE SECTION;

    Example:

```
1           void getStudio() {
2               EXEC SQL BEGIN DECLARE SECTION;
3                   char studioName[50], studioAddr[256]; // <- c
    variables
4                   char SQLSTATE[6];
5               EXEC SQL END DECLARE SECTION;
6
7               EXEC SQL INSERT INTO Studio(name, address)
8                       VALUES (:studioName, :studioAddr);
9           }
10
```

- Cursors
  - Is the most versatile way to connect SQL queries
  - **Syntax:**
    EXEC SQL DECLARE < cursor name > CURSOR FOR < query >

    EXEC SQL OPEN < cursor name >;
    ...
    EXEC SQL CLOSE < cursor name >;

    Example:

```
        void getStudio() {
            EXEC SQL BEGIN DECLARE SECTION;
                char studioName[50], studioAddr[256]; // <- c variables
                char SQLSTATE[6];
            EXEC SQL END DECLARE SECTION;

            EXEC SQL INSERT INTO Studio(name, address)
                        VALUES (:studioName, :studioAddr);
        }

```

Example in Python:

```
        import sqlite3
        connection = sqlite3.connect("company.db")

        cursor = connection.cursor()

        staff_data = [ ("William", "Shakespeare", "m", "1961-10-25"),
                       ("Frank", "Schiller", "m", "1955-08-17"),
                       ("Jane", "Wall", "f", "1989-03-14") ]

        for p in staff_data:
            format_str = """INSERT INTO employee (staff_number, fname, lname, gender, birth_date)
            VALUES (NULL, "{first}", "{last}", "{gender}", "{birthdate}");"""

            sql_command = format_str.format(first=p[0], last=p[1], gender=p[2], birthdate = p[3])
            cursor.execute(sql_command)

```

- Fetch Statement
  - fetch data from the result table one row at a time
  - **Syntax:**
    EXEC SQL FETCH FROM < cursor name > INTO < list of variables >

    Example:

```
        void worthRanges() {
            int i, digits, counts[15];
            EXEC SQL BEGIN DECLARE SECTION;
                int worth;
                char SQLSTATE[6];
            EXEC SQL END DECLARE SECTION;
            EXEC SQL DECLARE execCursor CURSOR FOR
                SELECT netWorth FROM MovieExec;
```

```
9
10              EXEC SQL OPEN execCursor;
11              for (i=1; i < 15; i++) counts[i] = 0;
12              while(1) {
13                  EXEC SQL FETCH FROM execCursor INTO :worth; //
      fetches a row of value from movieExec and stores in worth
14                  if (NO_MORE_TUPLES) break;
15
16                  ...
17              }
18          }
19
```

b)

```
1   void findLaptops() {
2       EXEC SQL BEGIN DECLARE SECTION;
3           int model;
4           float speed;
5           int ram;
6           int hd;
7           int screen;
8           float price;
9
10          float minSpeed;
11          int minRam;
12          int minHd;
13          float minPrice;
14      EXEC SQL END DECLARE SECTION;
15
16      EXEC SQL DECLARE execCursor CURSOR FOR
17          SELECT model, speed, ram, hd, screen, price, maker
18          FROM Product NATURAL JOIN Laptop;
19
20      EXEC SQL OPEN execCursor;
21
22      printf("Enter minimum speed:");
23      scanf("%f", &minSpeed);
24
25      printf("Enter minimum ram:");
26      scanf("%f", &minRam);
27
28      printf("Enter minimum hard-drive space:");
29      scanf("%f", &minHd);
30
31      printf("Enter minimum price:");
32      scanf("%f", &minPrice);
33
34      while(1) {
35          EXEC SQL FETCH FROM execCursor INTO :model,
36              :speed, :ram, :hd, :screen, :price, :maker;
37
38          if (NO_MORE_TUPLES) break;
39
40          if (
41              speed >= minSpeed &&
```

```
                ram >= minRam &&
                hd >= minHd &&
                screen >= minScreen
            ) {
                printf("model=%d, speed=%.2f, ram=%d, hd=%d, screen=%
d, price=%.2f, maker=%c",
                    model, speed, ram, hd, screen, price, maker);
            }
        }

        EXEC SQL CLOSE execCursor;
    }

```

c)

```
    void findLaptops() {
        EXEC SQL BEGIN DECLARE SECTION;
            int model;
            float speed;
            int ram;
            int hd;
            int screen;
            float price;

            float minSpeed;
            int minRam;
            int minHd;
            float minPrice;
        EXEC SQL END DECLARE SECTION;

        EXEC SQL DECLARE execCursor CURSOR FOR
            SELECT model, speed, ram, hd, screen, price, maker
            FROM Product NATURAL JOIN Laptop;

        EXEC SQL OPEN execCursor;

        printf("Enter minimum speed:");
        scanf("%f", &minSpeed);

        printf("Enter minimum ram:");
        scanf("%f", &minRam);

        printf("Enter minimum hard-drive space:");
        scanf("%f", &minHd);

        printf("Enter minimum price:");
        scanf("%f", &minPrice);

        while(1) {
            EXEC SQL FETCH FROM execCursor INTO :model,
                :speed, :ram, :hd, :screen, :price, :maker;

            if (NO_MORE_TUPLES) break;

            if (
```

```
                speed >= minSpeed &&
                ram >= minRam &&
                hd >= minHd &&
                screen >= minScreen
            ) {
                printf("model=%d, speed=%.2f, ram=%d, hd=%d, screen=%
d, price=%.2f, maker=%c",
                    model, speed, ram, hd, screen, price, maker);
            }
        }

        EXEC SQL CLOSE execCursor;
    }

```

d)

```
    #include <stdbool.h>
    #include <string.h>
    ...
    void printSpecifications() {
        EXEC SQL BEGIN DECLARE SECTION;
            int model;
            bool color;
            char printType[50];
            float price;

            float speed;
            int ram;
            int hd;
            int screen;

            char maker;
            int productModel;
            char productType[50];

            char targetMaker;
        EXEC SQL END DECLARE SECTION;

        EXEC SQL DECLARE execCursor CURSOR FOR Product;

        printf("Enter manufacturer:");
        scanf("%f", &targetMaker);

        EXEC SQL OPEN execCursor;
            EXEC SQL FETCH FROM execCursor INTO :model,
                :maker, :productType;

            if (NO_MORE_TUPLES) break;

            if (strcmp(productType,'pc')) {

            } else if (strcmp(productType, 'laptop')) {

            } else if (strcmp(productType, 'printer')) {

```

```
40              }
41
42          EXEC SQL CLOSE execCursor;
43      }
44
```