CSC343 Worksheet 8

June 24, 2020

1. **Exercise 9.5.1:** Repeat Exercise 9.3.1, but write the code using C with CLI calls.

   a) **Notes:**

   - Using Call-Level Interface
     - Uses host language to connect to and access a database
     - Replaces embedded SQL
   - Standard SQL/CLI
     - Is database CLI for C
     - Included in file *sqlcli.h*
     - Creates deals with four kinds of records

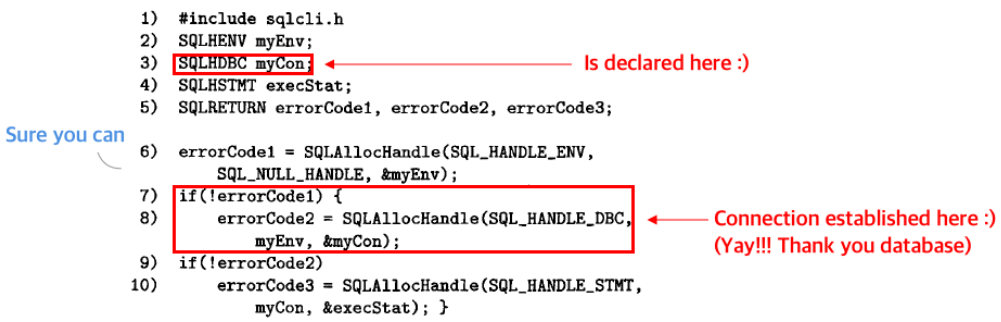
       1. Environment handle
          * Prepares one or more connections to database server
          * Is required
          * Is allocated using **SQLHENV**
          * Is established via function **SQLAllocHandle**

```
1)  #include sqlcli.h
2)  SQLHENV myEnv;
3)  SQLHDBC myCon;        <-- Is declared here :)
4)  SQLHSTMT execStat;
5)  SQLRETURN errorCode1, errorCode2, errorCode3;

6)  errorCode1 = SQLAllocHandle(SQL_HANDLE_ENV,   <-- Connection is prepared here :)
        SQL_NULL_HANDLE, &myEnv);                     (Hey DB, can I connect with you?)
7)  if(!errorCode1) {
8)      errorCode2 = SQLAllocHandle(SQL_HANDLE_DBC,
            myEnv, &myCon);
9)  if(!errorCode2)
10)     errorCode3 = SQLAllocHandle(SQL_HANDLE_STMT,
            myCon, &execStat); }
```

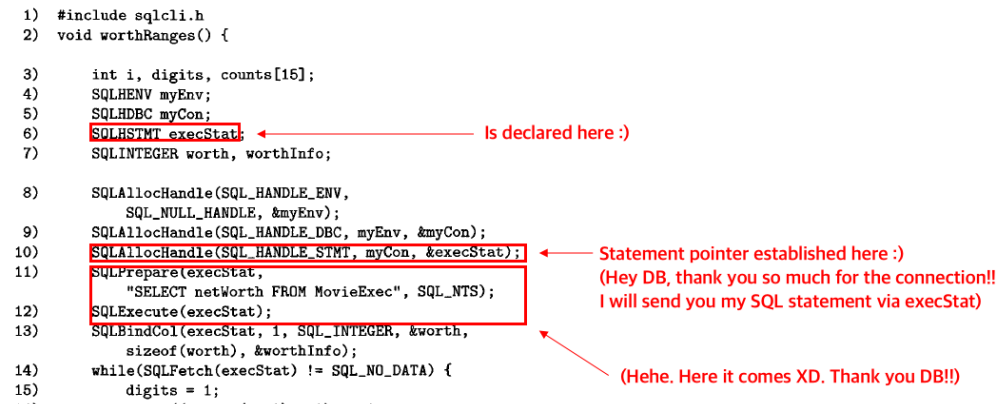
       2. Connection handle
          * Conenects application program to database
          * Is required
          * Is declared after **SQLHENV**

* Is allocated using **SQLHDBC**
* Is established via function **SQLAllocHandle**




```
1)  #include sqlcli.h
2)  SQLHENV myEnv;
3)  SQLHDBC myCon;
4)  SQLHSTMT execStat;
5)  SQLRETURN errorCode1, errorCode2, errorCode3;
6)  errorCode1 = SQLAllocHandle(SQL_HANDLE_ENV,
        SQL_NULL_HANDLE, &myEnv);
7)  if(!errorCode1) {
8)      errorCode2 = SQLAllocHandle(SQL_HANDLE_DBC,
            myEnv, &myCon);
9)  if(!errorCode2)
10)     errorCode3 = SQLAllocHandle(SQL_HANDLE_STMT,
            myCon, &execStat); }
```

3. Statements
   * Created by application program (the user)
   * Can be created as many as needed
   * Holds information about a single SQL statement, including cursor
   * Can represent different SQL statements at different times
   * Is required
   * Is declared after **SQLHDBC**
   * Is allocated using **SQLHSTMT**
   * Is sent using the function **SQLAllocHandle**




```
1)  #include sqlcli.h
2)  void worthRanges() {

3)      int i, digits, counts[15];
4)      SQLHENV myEnv;
5)      SQLHDBC myCon;
6)      SQLHSTMT execStat;
7)      SQLINTEGER worth, worthInfo;

8)      SQLAllocHandle(SQL_HANDLE_ENV,
            SQL_NULL_HANDLE, &myEnv);
9)      SQLAllocHandle(SQL_HANDLE_DBC, myEnv, &myCon);
10)     SQLAllocHandle(SQL_HANDLE_STMT, myCon, &execStat);
11)     SQLPrepare(execStat,
            "SELECT netWorth FROM MovieExec", SQL_NTS);
12)     SQLExecute(execStat);
13)     SQLBindCol(execStat, 1, SQL_INTEGER, &worth,
            sizeof(worth), &worthInfo);
14)     while(SQLFetch(execStat) != SQL_NO_DATA) {
15)         digits = 1;
16)         while((worth /= 10) > 0) digits++;
17)         if(digits <= 14) counts[digits]++;
        }
18)     for(i=0; i<15; i++)
19)         printf("digits = %d: number of execs = %d\n",
                i, counts[i]);
    }
```

4. Descriptions
   * Holds information about either tuples or parameters

```
1)  #include sqlcli.h
2)  void worthRanges() {

3)      int i, digits, counts[15];
4)      SQLHENV myEnv;
5)      SQLHDBC myCon;
6)      SQLHSTMT execStat;
7)      SQLINTEGER worth, worthInfo;

8)      SQLAllocHandle(SQL_HANDLE_ENV,
            SQL_NULL_HANDLE, &myEnv);
9)      SQLAllocHandle(SQL_HANDLE_DBC, myEnv, &myCon);
10)     SQLAllocHandle(SQL_HANDLE_STMT, myCon, &execStat);
11)     SQLPrepare(execStat,
            "SELECT netWorth FROM MovieExec", SQL_NTS);
12)     SQLExecute(execStat);
13)     SQLBindCol(execStat, 1, SQL_INTEGER, &worth,
            sizeof(worth), &worthInfo);
14)     while(SQLFetch(execStat) != SQL_NO_DATA) {
15)         digits = 1;
16)         while((worth /= 10) > 0) digits++;
17)         if(digits <= 14) counts[digits]++;
        }
18)     for(i=0; i<15; i++)
19)         printf("digits = %d: number of execs = %d\n",
                i, counts[i]);
    }
```

Is declared here :)

Statement pointer established here :)
(Hey DB, thank you so much for the connection!!
I will send you my SQL statement via execStat)

(Hehe. Here it comes XD. Thank you DB!!)

- Processing Statements
- Fetching Data From
- Passing Parameters to Queries

2. **Exercise 9.5.2:** Repeat Exercise 9.3.2, but write the code using C with CLI calls
3. **Exercise 9.6.1:** Repeat Exercise 9.3.1, but write the code using JAVA using JDBC.
4. **Exercise 9.6.2:** Repeat Exercise 9.3.2, but write the code using JAVA using JDBC.
5. **Exercise 9.7.1:** Repeat Exercise 9.3.1, but write the code using PHP.
6. **Exercise 9.7.2:** Repeat Exercise 9.3.2, but write the code using PHP.
7. **Exercise 9.7.3:** In Example 9.31 we exploited the feature of PHP that strings in double-quotes have variables expanded. How essential is this feature? Could we have done something analogous in JDBC? If so, how?