Quick-Start Guide on MySQL C-API

EECS 495 – Fall 2016 – Prof. Peter Scheuermann

Basic Functionality using C-API

How to start:

```
#include "stdafx.h"      // for windows users
#include <windows.h>     // for windows users
#include "mysql.h"       // you must copy this from MySQL folder to your
                         // project folder
#include <iostream>

using namespace std;

MYSQL *conn;    /* pointer to connection handler */

int main ( int argc, char *argv[] )
{
    conn = mysql_init ( NULL );

    // open connection
    mysql_real_connect (
            conn,           /* pointer to connection handler */
            "localhost",    /* host to connect to */
            "user_name",    /* user name */
            "password",     /* password */
            "test",         /* database to use */
            0,              /* port (default 3306) */
            NULL,           /* socket or /var/lib/mysql.sock */
            CLIENT_MULTI_RESULTS );            /* flags (none) */

    // close connection
    mysql_close ( conn );
    return 0;
}
```

Note:
CLIENT_MULTI_RESULTS flag is required in order to obtain results from a call to a stored procedure; without it, it will return an empty set, or null;

Note:
If at compilation it complaints that it cannot find "mysql.h" or other related header files:
You must copy the all the header files (.h) from the MySQL directory (mysql\include.h) to the project directory of Visual C++ (Windows). Under linux, search for these header files under the filesystem (particularly, look for "mysql.h")*

How to submit a plain SQL query and display the query results

```
// Submit query
MYSQL_RES *res_set;
MYSQL_ROW row;
mysql_query(conn,"SELECT * FROM Students;");
res_set = mysql_store_result(conn);
int numrows = (int)mysql_num_rows(res_set);


// Display results
for (int i = 0; i < numrows; i++)
    {
        row = mysql_fetch_row( res_set );
        if( row != NULL )
        {
            cout << "ID   : " << row[0] << endl;
            cout << "Name: " << row[1] << endl;
        }
    }
// free resources
mysql_free_result( res_set );
```

How to submit a query using stored procedures

Note:

Stored routines require the ***proc*** *table in the* `mysql` *database. This table is created during the MySQL 5.0 installation procedure. If you are upgrading to MySQL 5.0 from an earlier version, be sure to update your grant tables to make sure that the* `proc` *table exists*

Step1. Create the store procedure in MySQL. Login to MySQL using the command-line. Type the following (as example)

```
DELIMITER //
CREATE PROCEDURE listStudents()
BEGIN
        SELECT * FROM students;
END //
DELIMITER ;
```

Note

If you are using the mysql command-line utility, pay careful attention to this note.

The default MySQL statement delimiter is ; (as you have seen in all of the MySQL statement used thus far). However, the mysql command-line utility also uses ; as a delimiter. If the command-line utility were to interpret the ; characters inside of the stored procedure itself, those would not end up becoming part of the stored procedure, and that would make the SQL in the stored procedure syntactically invalid.

The solution is to temporarily change the command-line utility delimiter, as seen here:

```
DELIMITER //

CREATE PROCEDURE productpricing()
BEGIN
   SELECT Avg(prod_price) AS priceaverage
   FROM products;
END //

DELIMITER ;
```

Here, `DELIMITER //` tells the command-line utility to use `//` as the new end of statement delimiter, and you will notice that the `END` that closes the stored procedure is defined as `END //` instead of the expected `END;`. This way the `;` within the stored procedure body remains intact and is correctly passed to the database engine. And then, to restore things back to how they were initially, the statement closes with a `DELIMITER ;`

In C-API, you can call a stored procedure and display its results as follows:

```
      // Submit query via regular stored procedures
      MYSQL_RES *res_set;
      MYSQL_ROW row;
      mysql_query(conn,"CALL listStudents();");
      res_set = mysql_store_result(conn);
      int numrows = (int)mysql_num_rows(res_set);
      // Display results
      for (int i = 0; i < numrows; i++)
      {
        row2 = mysql_fetch_row( res_set );
        if( row != NULL )
        {
            cout << "ID  : " << row[0] << endl;
            cout << "Name: " << row[1] << endl;
        }
      }
      // free resources
      mysql_free_result( res_set );
```

MySQL Triggers

Support for triggers is included beginning with MySQL 5.0.2. A trigger is a named database object that is associated with a table, and that activates when a particular event occurs for the table. Some uses for triggers are to perform checks of values to be inserted into a table or to perform calculations on values involved in an update.

A trigger is defined to activate when an `INSERT`, `DELETE`, or `UPDATE` statement executes for the associated table. A trigger can be set to activate either before or after the triggering statement. For example, you can have a trigger activate before each row that is inserted into a table or after each row that is updated.

First, create a trigger from MySQL console (here is an example)

```
DELIMITER //

CREATE TRIGGER studentCount
   AFTER INSERT ON students
   FOR EACH ROW
  BEGIN
      UPDATE StudentMonitor
      SET count = count+1;
   END //

DELIMITER ;
```

assuming we already have a table “StudentMonitor” with a single attribute “count”, initialized to “0”. The purpose of this example-trigger is to count the number of times a new student is added to the list

On the MySQL C-API side, we test wether the trigger works by checking if the “count” attribute of the monitor table is increased when we insert a new student-entry to the “students” table:

```
// Triggers
mysql_query(conn,"SELECT * FROM StudentMonitor;");
res_set = mysql_store_result(conn);
row = mysql_fetch_row( res_set );
cout << "Number of inserts BEFORE a new insert: " << row[0] << endl;

mysql_free_result(res_set);

mysql_query(conn,
            "INSERT INTO students (id, name) value (100, 'auto');");

mysql_query(conn,"SELECT * FROM StudentMonitor;");
res_set = mysql_store_result(conn);
row = mysql_fetch_row( res_set );
cout << "Number of inserts AFTER a new insert: " << row[0] << endl;
// should have been increased by 1 if the trigger functions properly.

mysql_free_result(res_set);
```

Troubleshootings

Problem : I am getting the following linking error:

```
1>Linking...
1>mySqlAttempt.obj : error LNK2019: unresolved external symbol _mysql_close@4 referenced in
function _main
1>mySqlAttempt.obj : error LNK2019: unresolved external symbol _mysql_init@4 referenced in
function _main
```

Solution:

Step1. From Visual C++ 2008, press Alt-F7. Alternativelly, right click the project name in the Solution Explorer, click Properties, then under Configuration Properties, select Linker.

Step2. Click “Additional Dependencies” and write this under the text-box: “libmysql.lib”. Note that “libmysql.lib” you have to copy from the mysql installation folder to your project folder.

Step3. Rebuild. It should work

Problem : I cannot locate the mysql-related header files, I’ve searched under MySQL installation folder

Solution: You need to install mysql in custom mode, and make sure you install de developer packages. If you have already installed mysql, one simple solution is to download these developer packages manually (go to mysql website, and instead of downloading mysql installer, download the version without installer – which comes as a zip-folder. Unzip it to a suitable place, and you’ll be able to find these header files under that folder)