实验二：使用高级语言操作 MySQL 数据库

高宏 邹兆年

1. 实验目的

学会使用高级语言访问 MySQL 数据库，并进行查询。

2. 实验环境

MySQL 关系数据库管理系统、C++编译器。

本次实验主要利用 C 语言访问 MySQL 数据库，也可以使用 JAVA，PHP 等其他语言。不允许使用 ORM。

3. 实验内容

3.1 实验任务

在上次上机实验课建立的 COMPANY 数据库上，用 C 语言编写程序，完成如下查询，程序的命令行参数为：

company_query –q <Number> -p [Parameters]

其中，Number 代表待执行查询的序号，Parameters 为第 Number 号查询需要的参数列表。

待执行的 9 个查询为如下：

1：参加了项目编号为%PNO%的项目的员工号，其中%PNO%为 C 语言编写的程序的输入参数；

2：参加了项目名为%PNAME%的员工名字，其中%PNAME%为 C 语言编写的程序的输入参数；

3：在%DNAME%工作的所有工作人员的名字和地址，其中%DNAME%为 C 语言编写的程序的输入参数；

4：在%DNAME%工作且工资低于%SALARY%元的员工名字和地址，其中%DNAME%和%SALARY%为 C 语言编写的程序的输入参数；

5：没有参加项目编号为%PNO%的项目的员工姓名，其中%PNO%为 C 语言编写的程序的输入参数；

6：由%ENAME%领导的工作人员的姓名和所在部门的名字，，其中%ENAME%为 C 语言编写的程序的输入参数；

7：至少参加了项目编号为%PNO1%和%PNO2%的项目的员工号，其中%PNO1%和%PNO2%为 C 语言编写的程序的输入参数；

8：员工平均工资低于%SALARY%元的部门名称，其中%SALARY%为 C 语言编写的程序的输入参数；

9：至少参与了%N%个项目且工作总时间不超过%HOURS%小时的员工名字，其中%N%和%SALARY%为 C 语言编写的程序的输入参数；

3.2 建立 MySQL 工程方法

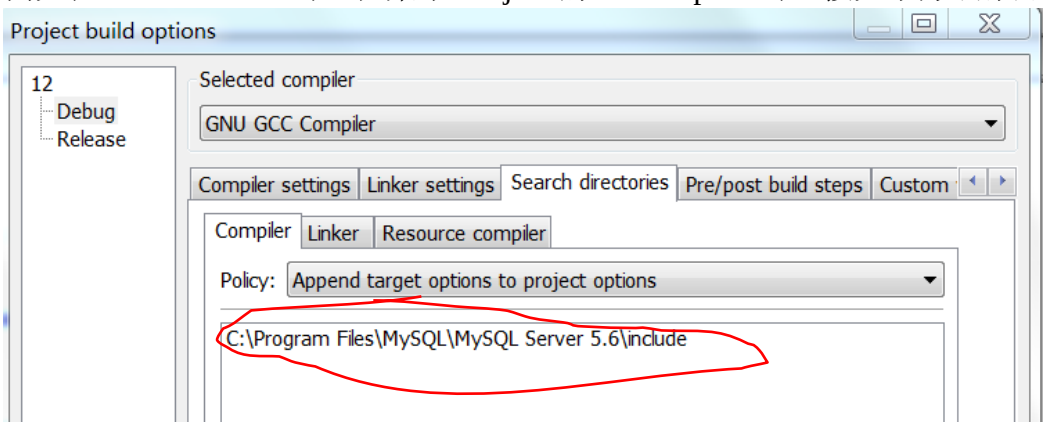
了解编程相关的资源文件。MySQL 安装路径下的 include 文件夹存放 C 语言编译相关的头文件（例如重要的 mysql.h），lib 文件夹存放相关的静态库（例如重要的 opt/libmysql.lib）

注：MySQL Server5.1.41 不用添加，MySQL Server5.6 需要自行添加以下路径。

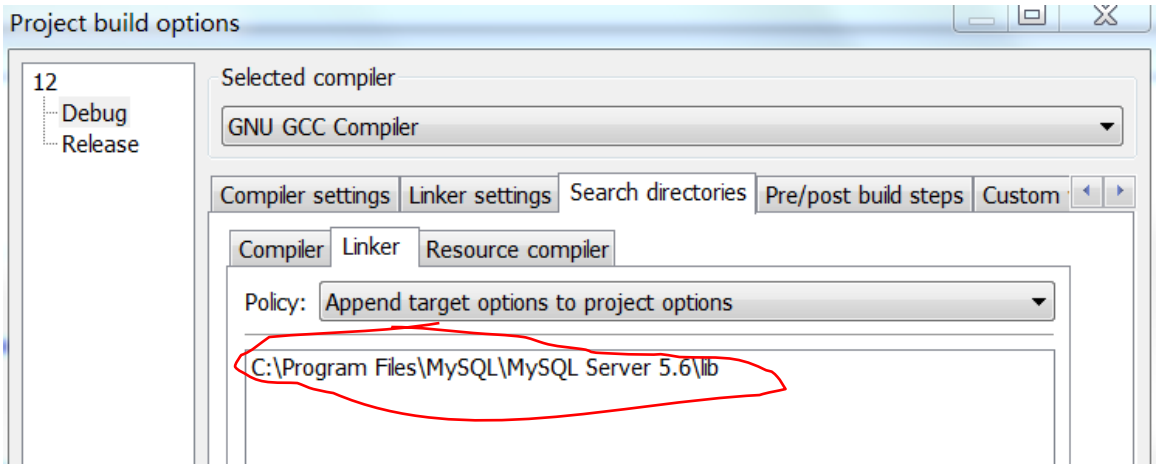
使用 C 语言编程环境（Code::Blocks 或 Visual C++）建立空白工程。在该工程的编译和链接配置中，首先添加引用路径"%MySQL 安装路径%/include/"

其中"%MySQL 安装路径%"为 MySQL 安装路径，在实验中心的计算机上为 C:/Program Files/MySQL/include/

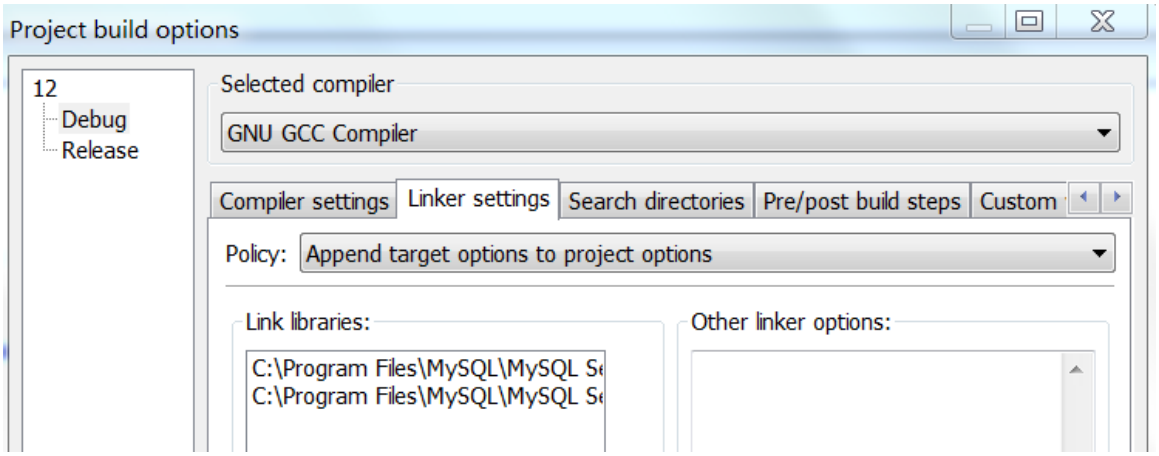
例如在 Code blocks 中，在菜单 Project 的 Build option 中，按如下方法添加:



然后，添加静态库路径 %MySQL 安装路径%/lib/



最后，添加需要的库文件%MySQL 安装路径%/lib/下的库文件 libmysql.lib 和 mysqlclient.lib



3.3 连接、断开 MySQL 服务器

在程序中必须加入#define __LCC__，以避免在 Windows 操作系统下进行编译时产生的错误。然后，在程序中加入#include <mysql.h>，让编译器能够找到所有 MySQL Client 的函数定义。

随后，使用 mysql_init 函数准备连接，mysql_init 函数的声明为

MYSQL *mysql_init(MYSQL *mysql);

其参数和返回值均为类型为 MYSQL 的结构体的指针。MYSQL 结构体是连接 MySQL 服务器的句柄，每一个句柄代表程序与 MySQL 服务器唯一的连接，所有 insert，select，delete 等操作都需要 MySQL 句柄作为参数以识别不同用户、数据库以及程序。若 mysql_init 操作成功，则返回刚刚初始化的句柄；否则，返回 NULL。

下面是一个简单的例子。

```
#ifndef __LCC__
#define __LCC__
#endif

#include <stdio.h>
#include <mysql.h>

int main(int argc, char **argv)
{
   MYSQL mysql_conn; /* Connection handle */

   if (mysql_init(&mysql_conn) != NULL)
{
      printf("Init succeeds!\n");
   }
```

```
else
{
      printf("Init fails!\n");
   }
   return 0;
}
```

编译并运行该程序，若句柄初始化成功，则打印”Init succeeds!”；否则，打印”Init fails!”。

在句柄被初始化后，就可以使用 mysql_real_connect 函数或者 mysql_connect 函数连接 MySQL 服务器了，这两个函数的声明如下：

```
MYSQL *mysql_real_connect(
   MYSQL *mysql, const char *host, const char *user,
   const char *passwd, const char *db, unsigned int port,
   const char *socket, unsigned long client_flag);
```

其中，第 1 个参数是 MySQL 句柄，第 2 至第 4 个参数分别为 host name、user name 及 password，第 5 个参数是待使用的数据库的名字，你也可把第 5 个参数设为 NULL，之后使用 mysql_select_db 函数来选择数据库，第 6 个参数是 MySQL 服务器的连接池，通常把它设为 MYSQL_PORT，第 7 个参数是 MySQL 服务器及 client 之间传输的渠道（socket 或 named pipe），但 MySQL 服务器会根据 host name 建立另一渠道，除非你有真的需要，否则你需在第 7 个参数填上 NULL。最后是 client_flag，包括压缩协议、查询协议、加密协议等。

与 MySQL 服务器的连接成功后，使用 mysql_close 函数断开连接，其函数声明为：

```
int mysql_close(MYSQL*);
```

下面是一个简单的例子：

```
#ifndef __LCC__
#define __LCC__
#endif

#include <stdio.h>
#include <mysql.h>

int main(int argc, char **argv) {
  MYSQL mysql_conn; /* Connection handle */

    if (mysql_init(&mysql_conn) != NULL) {
       if (mysql_real_connect(
            &mysql_conn, "localhost", "root",
```

```
            NULL, "company", MYSQL_PORT,
            NULL, 0) != NULL)
          printf("GOOD!\n");
       else
          printf("Connection fails.\n");
    } else {
       printf("Initialization fails.\n");
       return -1;
    }

    mysql_close(&mysql_conn);
    return 0;
}
```

至此，我们学会了如何连接、断开 MySQL 服务器。

3.4 查询 MySQL 数据库

在成功连接 MySQL 服务器后，便可以使用 mysql_query 函数查询数据库。mysql_query 函数的声明如下：

```
int mysql_query(MYSQL *mysql, const char *query);
```

第 1 个参数是 MySQL 连接的句柄，第 2 个参数是 SQL 语句。如果查询成功，mysql_query 返回 0，否则返回非 0。

如果你的查询语句不需要结果返回，例如 DELETE、UPDAE、INSERT 等，mysql_query 被执行后便完成整个操作了；如果你要执行 SELECT、SHOW、DESCRIBE 等，在存取结果前，必须使用 mysql_store_result 函数建立查询结果的句柄。mysql_store_result 的函数声明如下：

```
MYSQL_RES *mysql_store_result(MYSQL *mysql);
```

其输入参数为 MySQL 的连接句柄，其输出为 MYSQL_RES 结构体指针类型。MYSQL_RES 结构体表示的是 MySQL 数据库传递回来的查询结果，但我们并不会直接读取 MYSQL_RES 的数据，而是使用 MYSQL_ROW，也即是以行为单位读取数据。

例子如下所示：

```
#ifndef __LCC__
#define __LCC__
#endif

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

```
int main(int argc, char **argv) {
   MYSQL mysql_conn; /* Connection handle */
   MYSQL_RES *mysql_result;

   if (mysql_init(&mysql_conn) != NULL) {
      if (mysql_real_connect(
            &mysql_conn, "localhost", "root",
            NULL, "company", MYSQL_PORT,
            NULL, 0) != NULL) {
         mysql_query(
            &mysql_conn, "select * from table1");
         mysql_result = mysql_store_result(&mysql_conn);

         /* Free the result to release the heap memory*/
         mysql_free_result(mysql_result);
      } else {
         printf("Connection fails.\n");
      }
   } else {
      printf("Initialization fails.\n");
      return -1;
   }

   mysql_close(&mysql_conn);
   return 0;
}
```

请注意 mysql_result 是一个指针。因为 mysql_store_result 函数会自动分配内存存储查询结果，所以需要执行 mysql_free_result(MYSQL_RES*)来释放内存。

3.5 提取查询结构

在提取结果前必须使用上面介绍过的 mysql_store_result 函数为查询结果分配内存，然后使用 mysql_fetch_row 函数逐行提取数据。mysql_fetch_row 函数的声明如下：

MYSQL_ROW mysql_fetch_row(MYSQL_RES *result);

其输入参数为查询结果句柄，输出的类型为 MYSQL_ROW。MYSQL_ROW 是一个数组结构，数组中每一个元素依次为该元组各个属性上的值。

结果的元组数可以用 mysql_num_rows 函数返回。mysql_num_rows 函数的声明为

int mysql_num_rows(MYSQL_RES*);

其输入参数为查询结果句柄，输出结果为元组数量。

下面是一个简单的例子：

```
#ifndef __LCC__
#define __LCC__
#endif

#include <stdio.h>
#include <malloc.h>
#include <mysql.h>

int main(int argc, char **argv) {
   MYSQL mysql_conn; /* Connection handle */
   MYSQL_RES *mysql_result; /* Result handle */
   MYSQL_ROW mysql_row; /* Row data */
   int f1, f2, num_row, num_col;

   if (mysql_init(&mysql_conn) != NULL) {
      if (mysql_real_connect(
         &mysql_conn, "localhost", "root",
         NULL, "company", MYSQL_PORT, NULL, 0) != NULL){
         if (mysql_query(
               &mysql_conn,
               "select * from testtable limit 10") == 0) {
            mysql_result = mysql_store_result(&mysql_conn);
            num_row = mysql_num_rows(mysql_result);
            num_col = mysql_num_fields(mysql_result);

            for (f1 = 0; f1 < num_row; f1++) {
               mysql_row = mysql_fetch_row(mysql_result);

               for (f2 = 0; f2 < num_col; f2++)
                  printf(
                     "[Row %d, Col %d] ==> [%s]\n",
                     f1+1, f2+1, mysql_row[f2]);
            }
         } else
            printf("Query fails\n");
      } else {
            int i = mysql_errno(&mysql_conn);
            const *s = mysql_error(&mysql_conn);
            printf("Connection fails(ERROR %d): %s\n", i, s);
      }
```

```
    } else
        printf("Initialization fails\n");

    mysql_free_result(mysql_result);
    mysql_close(&mysql_conn);
    return 0;
}
```

请注意如果数据库很大，而又没有用 mysql_free_result 函数释放内存的话，很容易发生内存溢出的问题。

4. 参考资料

Abraham Silberschatz, Henry F.Korth. 《数据库系统概念（第六版）》