

北京邮电大学 计算机学院
《数据库系统原理》实验报告

姓名 王睿嘉

学号 2015211906

班级 2015211307

实验四 数据库接口

一、实验内容 and 环境描述

1. 实验目的

- 1) 了解通用数据库应用编程接口 ODBC 的基本原理和实现机制,熟悉主要 ODBC 接口的语法和使用方法;
- 2) 利用 C 语言 (或其他支持 ODBC 接口的高级程序设计语言) 编程实现简单的数据库应用程序,掌握基于 ODBC 的数据库访问基本原理和方法。

2. 实验内容

- 1) 以教科书第四章关于 SQL 语言的相关内容为基础,课后查阅、自学 ODBC 接口有关知识,包括其体系结构、工作原理、数据访问过程、主要 API 接口语法和使用方法等;
- 2) 以实验二建立的学生数据库为基础,编写 C 语言 (或其他支持 ODBC 接口的高级程序设计语言) 数据库应用程序,按照如下步骤访问数据库:
 - ① ODBC 初始化,为其分配环境句柄;
 - ② 建立应用程序与 ODBC 数据源的连接;
 - ③ 利用 SQLExecDirect 语句,实现应用程序对数据库的建立、查询、修改和删除等操作;
 - ④ 检索查询结果集;
 - ⑤ 结束数据库应用程序。

3. 实验要求

所编写的数据库访问应用程序中使用到以下主要的 ODBC API 函数:

- ① SQLAllocEnv: 初始化 ODBC 环境,返回环境句柄;
- ② SQLAllocConnect: 为连接句柄分配内存并返回句柄;
- ③ SQLConnect: 连接一个 SQL 数据资源;
- ④ SQLDriverConnect: 连接一个 SQL 数据资源,允许驱动器向用户询问信息;
- ⑤ SQLAllocStmt: 为语句句柄分配内存,并返回句柄;
- ⑥ SQLExecDirect: 把 SQL 语句送到数据库服务器,请求执行由其定义的数据库访问操作;
- ⑦ SQLFreeStmt: 释放与语句句柄相关的资源;
- ⑧ SQLDisconnect: 切断连接;
- ⑨ SQLFreeConnect: 释放与连接句柄相关的资源;
- ⑩ SQLFreeEnv: 释放与环境句柄相关的资源。

4. 实验环境

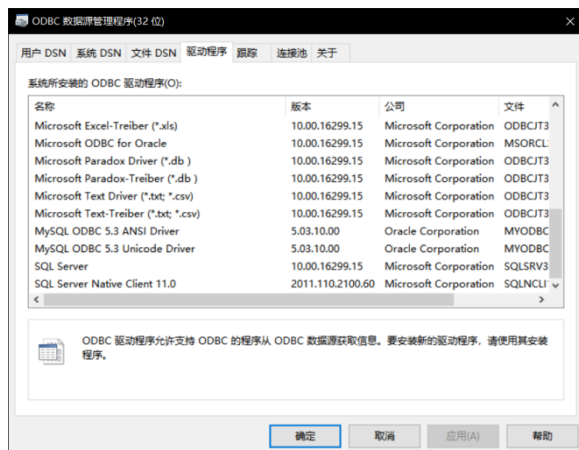
数据库系统: 5.7.21 MySQL Community Server (GPL)

二、实验结论和心得

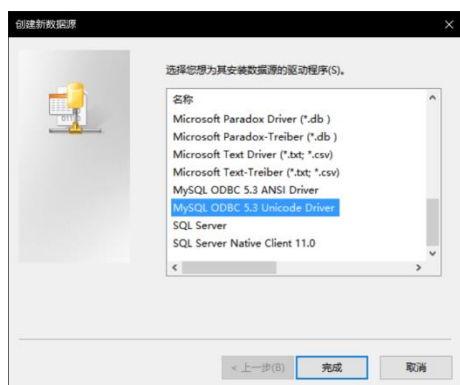
1. 实验结论

1.1 ODBC 数据源配置

下载 mysql-connector-c-6.1.11-win32.msi 并安装，在控制面板-管理工具-ODBC 数据源（32 位）-驱动程序中可以看到相应 mysql 驱动：



在用户 DSN 选项卡中，点击“添加”，从“创建新数据源”列表中选择刚安装的驱动：



填写数据源配置信息，点击“Test”进行测试，若出现“Connection successful”提示框则表示数据库连接成功：



1.2 ODBC 实现方法

初始化 ODBC，获取相应环境句柄

```
SQLHENV henv = SQL_NULL_HENV;    //初始化环境句柄
retcode = SQLAllocHandle(SQL_HANDLE_ENV, SQL_NULL_HANDLE, &henv);
//初始化ODBC环境，分配环境句柄
retcode = SQLSetEnvAttr(henv, SQL_ATTR_ODBC_VERSION, (SQLPOINTER)SQL_OV_ODBC3,
SQL_IS_INTEGER);
```

无论程序建立多少个 ODBC 连接，此过程只需执行一次。

与 ODBC 数据源建立连接

```
SQLHDBC hdbc = SQL_NULL_HDBC;    //初始化连接句柄
retcode = SQLAllocHandle(SQL_HANDLE_DBC, henv, &hdbc);
//为连接句柄分配内存并返回
retcode = SQLConnect(hdbc, szDSN, SQL_NTS, userID, SQL_NTS, passWORD, SQL_NTS);
//连接数据库
```

向 ODBC 数据源提交 SQL 语句，实现数据库操作

```
SQLHSTMT hstmt = SQL_NULL_HSTMT;    //初始化语句句柄
retcode = SQLAllocStmt(hdbc, &hstmt);    //为语句句柄分配内存，并返回
retcode = SQLExecDirect(hstmt, sqlOper, SQL_NTS);
//向 ODBC 数据源提交 SQL 语句，实现数据库操作
```

结束应用程序

应用程序完成数据库操作，退出运行之前，需要释放程序中使用的系统资源。这些资源包括：语句句柄、连接句柄和环境句柄等。

```
SQLFreeStmt(hstmt, SQL_DROP);    //释放语句句柄
SQLDisconnect(hdbc);            //切断连接
SQLFreeConnect(hdbc);           //释放与连接句柄相关的资源
SQLFreeEnv(henv);               //释放与环境句柄相关的资源
```

1.3 程序源码

```
2  #include <iostream>
3  #include <string>
4  #include <windows.h>
5  #include <sql.h>
6  #include <sqlext.h>
7  #include <sqltypes.h>
8  #include <odbcss.h>
9  using namespace std;
10
11 #pragma comment (lib,"odbc32.lib")
12 #pragma comment (lib,"libmysql.lib")
13 #define MAXBUFLEN 255
14
15 /*初始化 ODBC*/
16 SQLHENV henv = SQL_NULL_HENV;    //初始化环境句柄
17 SQLHDBC hdbc = SQL_NULL_HDBC;    //初始化连接句柄
```

```

18 SQLHSTMT hstmt = SQL_NULL_HSTMT;           //初始化语句句柄
19
20 /*结构*/
21 struct Course{
22     char cno[4];
23     char cname[15];
24     int credit;
25 };
26
27 struct Student{
28     char sno[7];
29     char sname[7];
30     char sex[3];
31     char bdate[20];
32     char dept[9];
33     char classno[5];
34 };
35
36 struct Sc{
37     char sno[7];
38     char cno[4];
39     int grade;
40 };
41
42 unsigned char sqlSelect[3][100] = { "select * from course", "select * from student", "select * from sc" };
43 unsigned char sqlOper[MAXBUFLen];
44
45 /*打印数据库信息*/
46 void priCourse(HDBC);
47 void priStudent(HDBC);
48 void priSc(HDBC);
49
50 int main(){
51     RETCODE retcode;           //定义返回代码
52     UCHAR szDSN[10] = "mysqlodbc"; //数据源
53     UCHAR userID[5] = "root";    //用户名
54     UCHAR passWORD[7] = "142857"; //密码
55     string s;
56
57     retcode = SQLAllocHandle(SQL_HANDLE_ENV, SQL_NULL_HANDLE, &henv);
58     //初始化ODBC环境，分配环境句柄
59     retcode = SQLSetEnvAttr(henv, SQL_ATTR_ODBC_VERSION, (SQLPOINTER)SQL_OV_ODBC3,
60     SQL_IS_INTEGER);
61     retcode = SQLAllocHandle(SQL_HANDLE_DBC, henv, &hdbc);

```

```

    //为连接句柄分配内存并返回
60  retcode = SQLConnect(hdbc, szDSN, SQL_NTS, userID, SQL_NTS, passWORD, SQL_NTS);
61  //连接数据库
62
63  if ((retcode != SQL_SUCCESS) && (retcode != SQL_SUCCESS_WITH_INFO))
64      cout << "Connect failed!" << endl;
65  else
66      while (true) {
67          cout << "mysql>";
68          getline(cin, s);
69          if (s != "exit") {
70              strcpy_s((char*)sqlOper, MAXBUFLEN, s.c_str());
71
72              retcode = SQLAllocStmt(hdbc, &hstmt);
73              retcode = SQLExecDirect(hstmt, sqlOper, SQL_NTS);
74  //向ODBC数据源提交SQL语句，实现数据库操作
75
76              if (retcode == SQL_SUCCESS || retcode == SQL_SUCCESS_WITH_INFO)
77                  cout << "Query OK(0.00 sec)" << endl;
78              else
79                  cout << "ERROR" << endl;
80              SQLFreeStmt(hstmt, SQL_DROP);
81
82              if (s.find("course") != string::npos)
83                  priCourse(hdbc);
84              else if (s.find("student") != string::npos)
85                  priStudent(hdbc);
86              else if (s.find("sc") != string::npos)
87                  priSc(hdbc);
88          }
89          else {
90              cout << "Bye~" << endl;
91              SQLDisconnect(hdbc);           //切断连接
92              SQLFreeConnect(hdbc);          //释放与连接句柄相关的资源
93              SQLFreeEnv(henv);              //释放与环境句柄相关的资源
94              break;
95          }
96      }
97  system("pause");
98  return 0;
99  }
100
101 void priCourse(HDBC hdbc){
102     HSTMT hstmt;

```

```

103 RETCODE retcode;
104
105 long lenOut1, lenOut2, lenOut3;
106 struct Course* ccell;
107 ccell = (struct Course*)malloc(sizeof(struct Course));
108
109 retcode = SQLAllocStmt(hdbc, &hstmt);
110
111 if (retcode == SQL_SUCCESS || retcode == SQL_SUCCESS_WITH_INFO){
112     retcode = SQLBindCol(hstmt, 1, SQL_C_CHAR, ccell->cno, 4, &lenOut1);
113 //将应用程序的数据缓冲绑定到结果集各列
114     retcode = SQLBindCol(hstmt, 2, SQL_C_CHAR, ccell->cname, 15, &lenOut2);
115     retcode = SQLBindCol(hstmt, 3, SQL_C_SLONG, &ccell->credit, sizeof(int), &lenOut3);
116     retcode = SQLExecDirect(hstmt, sqlSelect[0], SQL_NTS);
117 //将SQL语句送至数据库服务器，执行由其定义的访问
118
119     if (retcode == SQL_SUCCESS || retcode == SQL_SUCCESS_WITH_INFO){
120         cout << "Course is as following:" << endl;
121         cout << "*****" << endl;
122         cout << "cno      cname      credit" << endl;
123
124         retcode = SQLFetch(hstmt);
125         //相当于SQLFetchAdvances和SQLGetData两个函数的功能
126
127         while (retcode == SQL_SUCCESS || retcode == SQL_SUCCESS_WITH_INFO){
128             printf("%-10s%-20s%-16d\n", ccell->cno, ccell->cname, ccell->credit);
129             retcode = SQLFetch(hstmt);
130         }
131         cout << "*****" << endl;
132     }
133     SQLFreeStmt(hstmt, SQL_DROP);
134     //释放语句句柄
135 }
136
137 void priStudent(HDBC hdbc){
138     HSTMT hstmt;
139     RETCODE retcode;
140
141     long lenOut1, lenOut2, lenOut3, lenOut4, lenOut5, lenOut6;
142     struct Student* scell;
143     scell = (struct Student*)malloc(sizeof(struct Student));
144     retcode = SQLAllocStmt(hdbc, &hstmt);

```

```

145 if (retcode == SQL_SUCCESS || retcode == SQL_SUCCESS_WITH_INFO){
146     retcode = SQLBindCol(hstmt, 1, SQL_C_CHAR, scell->sno, 7, &lenOut1);
147     retcode = SQLBindCol(hstmt, 2, SQL_C_CHAR, scell->sname, 7, &lenOut2);
148     retcode = SQLBindCol(hstmt, 3, SQL_C_CHAR, &scell->sex, 3, &lenOut3);
149     retcode = SQLBindCol(hstmt, 4, SQL_C_CHAR, &scell->bdate, 20, &lenOut4);
150     retcode = SQLBindCol(hstmt, 5, SQL_C_CHAR, scell->dept, 9, &lenOut5);
151     retcode = SQLBindCol(hstmt, 6, SQL_C_CHAR, scell->classno, 5, &lenOut6);
152
153     retcode = SQLExecDirect(hstmt, sqlSelect[1], SQL_NTS);
154
155     if (retcode == SQL_SUCCESS || retcode == SQL_SUCCESS_WITH_INFO){
156         cout << "Student is as following:" << endl;
157         cout <<
            "*****" <<
            endl;
158         cout << "sno      sname      sex      bdate      dept
            classno" << endl;
159
160         retcode = SQLFetch(hstmt);
161
162         while (retcode == SQL_SUCCESS || retcode == SQL_SUCCESS_WITH_INFO){
163             printf("%-10s%-15s%-10s%-25s%-10s%-15s\n", scell->sno, scell->sname, scell->sex,
            scell->bdate, scell->dept, scell->classno);
164             retcode = SQLFetch(hstmt);
165         }
166         cout <<
            "*****" <<
            endl;
167     }
168 }
169 SQLFreeStmt(hstmt, SQL_DROP);
170 }
171
172 void priSc(HDBC hdbc) {
173     HSTMT hstmt;
174     RETCODE retcode;
175
176     long lenOut1, lenOut2, lenOut3;
177     struct Sc* cell;
178     cell = (struct Sc*)malloc(sizeof(struct Sc));
179
180     retcode = SQLAllocStmt(hdbc, &hstmt);
181
182     if (retcode == SQL_SUCCESS || retcode == SQL_SUCCESS_WITH_INFO) {

```



```

183     retcode = SQLBindCol(hstmt, 1, SQL_C_CHAR, cell->sno, 7, &lenOut1);
184     retcode = SQLBindCol(hstmt, 2, SQL_C_CHAR, cell->cno, 4, &lenOut2);
185     retcode = SQLBindCol(hstmt, 3, SQL_C_SLONG, &cell->grade, sizeof(int), &lenOut3);
186
187     retcode = SQLExecDirect(hstmt, sqlSelect[2], SQL_NTS);
188
189     if (retcode == SQL_SUCCESS || retcode == SQL_SUCCESS_WITH_INFO) {
190         cout << "Sc is as following:" << endl;
191         cout << "*****" << endl;
192         cout << "sno      cno      grade" << endl;
193
194         retcode = SQLFetch(hstmt);
195
196         while (retcode == SQL_SUCCESS || retcode == SQL_SUCCESS_WITH_INFO) {
197             printf("%-10s%-16s%d\n", cell->sno, cell->cno, cell->grade);
198             retcode = SQLFetch(hstmt);
199         }
200         cout << "*****" << endl;
201     }
202 }
203 SQLFreeStmt(hstmt, SQL_DROP);
204 }

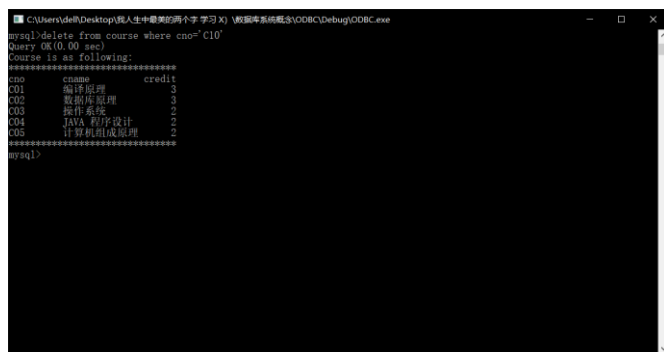
```

1.4 运行结果

查询操作：

插入、更新操作：

删除操作：



```
C:\Users\del\Desktop\人生中最美的两个字 学习 X\数据库系统概念\ODBC\Debug\ODBC.exe
mysql> delete from course where cno="C10"
Query OK (0.00 sec)
course is as following:
=====
cno      cname      credit
----      -
C01      编译原理      3
C02      数据库原理      3
C03      操作系统      3
C04      JAVA 程序设计      3
C05      计算机组成原理      3
=====
mysql>
```

应用程序退出：



```
C:\Users\del\Desktop\人生中最美的两个字 学习 X\数据库系统概念\ODBC\Debug\ODBC.exe
mysql> exit
Bye
请按任意键继续. . .
```

2. 实验心得

在本次实验中，遇到的主要问题有以下四点：

- ① ODBC 数据源命名。根据网上相关教程，命名 ODBC 数据源是任意的。起初，我直接将其命名为 ODBC，测试通过却始终出现无效 DSN 提示，可能是与关键词冲突导致。最终，更改名称，该问题得到解决；
- ② 数据源字长。起初使用 64 位 ODBC，无法连接成功。最终，重新安装 32 位驱动，该问题得到解决；
- ③ <mysql.h> 库的使用。该库操作数据源极为方便，但 64 位头文件与 32 位 IDE 不匹配，最终决定采用实验指导书所推荐的 <sql.h>；
- ④ 中文乱码。数据库查询结果中，中文字符均为乱码。将数据源字符集改为 GBK 后，该问题得到解决；

实践出真知，本次数据库接口实验是对课堂和书本所学知识的补充。通过自己动手、亲力亲为编写数据库应用程序，加深了对 ODBC 的理解和记忆，收获颇丰。