

《数据库系统原理》实验报告

实验名称 数据库接口实验

班 级 2015211307

学 号 2017526019

姓 名 刘禾子

实验四 数据库接口实验

■ 实验目的

1. 通过实验了解通用数据库应用编程接口 ODBC 的基本原理和实现机制，熟悉主要的 ODBC 接口的语法和使用方法；
2. 利用 C 语言(或其它支持 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 的数据库访问的基本原理和方法
3. 学习 java 语言，并采用 jdbc 接口方式对数据库进行访问

■ 实验平台及环境

Windows10、MySQL 5.7、Microsoft Visual Studio 2017

■ 实验内容

1. 以教科书第四章关于 SQL 语言相关内容为基础，课后查阅、自学 ODBC 接口有关内容，包括 ODBC 的体系结构、工作原理、数据访问过程、主要 API 接口的语法和使用方法等。
2. 以实验二建立的学生数据库为基础，编写 C 语言(或其它支持 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

Step1. ODBC 初始化，为 ODBC 分配环境句柄

Step2. 建立应用程序与 ODBC 数据源的连接

Step3. 利用 SQLExecDirect 语句，实现数据库应用程序对数据库的建立、查询、修改、删除等操作

Step4. 检索查询结果集

Step5. 结束数据库应用程序

■ 实验要求

1. 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：

SQLAllocEnv：初始化 ODBC 环境，返回环境句柄

SQLAllocConnect：为连接句柄分配内存并返回连接句柄

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，并返回语句句柄□

(6) SQLExecDirect

把 SQL 语句送到数据库服务器，请求执行由 SQL 语句定义的数据库访问

(7) SQLFetchAdvances

将游标移动到到查询结果集的下一行(或第一行)

(8) SQLGetData

按照游标指向的位置，从查询结果集的特定的一列取回数据

(9) SQLFreeStmt

释放与语句句柄相关的资源

(10) SQLDisconnect

切断连接


(11) SQLFreeConnect

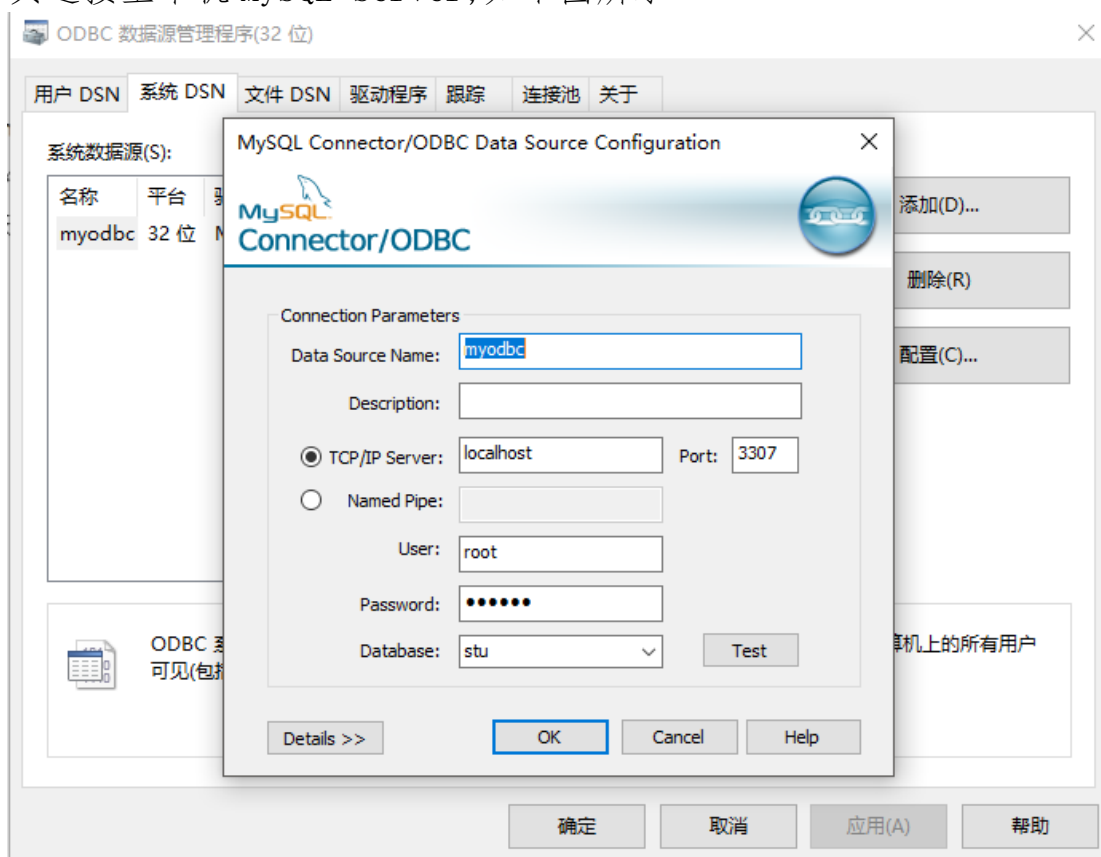
释放与连接句柄相关的资源

(12) SQLFreeEnv

释放与环境句柄相关的资源

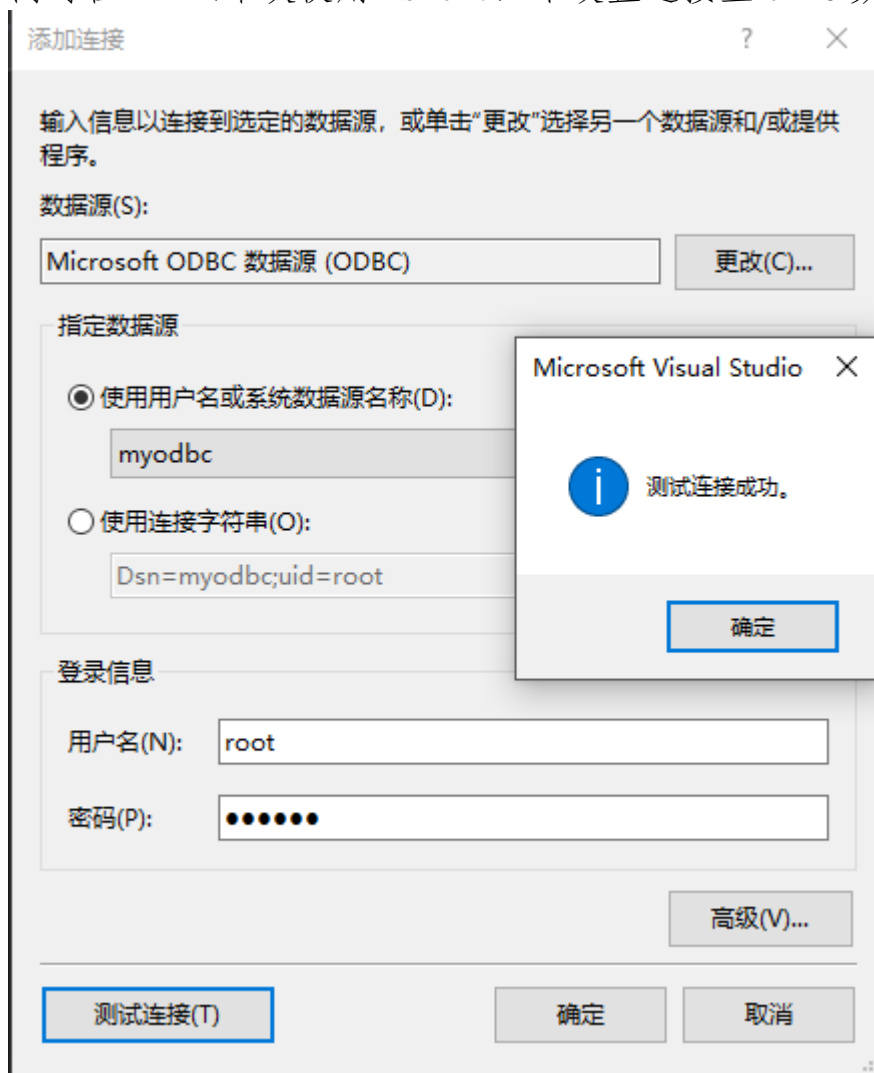
■ 实验步骤及结果分析

1. 根据机型下载  mysql-connector-odbc-5.3.10-winx64.msi，点击安装后进入控制面板->管理工具->ODBC 数据源管理工具（64 位）将其连接至本机 MySQL Server,如下图所示



测试其连接，显示 connection successful 则表示连接成功

2. 同时在 IDE（本次使用 VS2017）中设置连接至 ODBC 数据源



3. 源代码

```
#include<windows.h>
#include<stdio.h>
#include<sql.h>
#include<stdlib.h>
#include<sqlext.h>
#include<string.h>
#include<iostream>
#include<assert.h>
using namespace std;

struct student {
    char *sno = new char[10];
    char *sname = new char[30];
    char *sex = new char[30];
    char *bdate = new char[30];
    char *dept = new char[30];
}
```

```

    char *classno = new char[30];
};

struct course{
    char *cno = new char[5];
    char *cname = new char[20];
    char *lhour = new char[5];
    char *credit = new char[5];
    char *semester = new char[5];
};

struct sc{
    char *sno = new char[10];
    char *cno = new char[5];
    char *grade = new char[5];
};

SQLHENV henv;           //初始化环境句柄
SQLHDBC hdbc;           //初始化数据源
SQLHSTMT hstmt;         //初始化语句句柄
SQLRETURN retcode;      //初始化返回错误代码
char input[100];        //用户输入
bool flag;              //操作正确与否标志

void select_course() {
    retcode = SQLAllocHandle(SQL_HANDLE_STMT, hdbc, &hstmt);
    retcode = SQLSetStmtAttr(hstmt, SQL_ATTR_ROW_BIND_TYPE,
(SQLPOINTER)SQL_BIND_BY_COLUMN, SQL_IS_INTEGER);
    retcode = SQLExecDirect(hstmt, (SQLCHAR *)"select * from course;",
SQL_NTS);
    if (retcode == SQL_SUCCESS || retcode == SQL_SUCCESS_WITH_INFO)
    {
        cout << "Query OK(0.00sec)" << endl;
    }

    struct course course;
    cout << "===== " << endl;
    cout << "cno\tcname\tlhour\tcredit\tsemester" << endl;
    long columnlen;
    SQLBindCol(hstmt, 1, SQL_CHAR, course.cno, 5, &columnlen);
    SQLBindCol(hstmt, 2, SQL_CHAR, course.cname, 20, &columnlen);
    SQLBindCol(hstmt, 3, SQL_CHAR, course.lhour, 5, &columnlen);
    SQLBindCol(hstmt, 4, SQL_CHAR, course.credit, 5, &columnlen);
    SQLBindCol(hstmt, 5, SQL_CHAR, course.semester, 5, &columnlen);
    if (retcode<0)
    {
        cout << "没有执行语句" << endl;
    }
}

```

```

        retcode = SQLFetch(hstmt);
        while (retcode == SQL_ROW_SUCCESS || retcode ==
SQL_ROW_SUCCESS_WITH_INFO)
        {
            if (retcode == SQL_ROW_SUCCESS || retcode ==
SQL_ROW_SUCCESS_WITH_INFO) {
                printf("%s\t%s\t%s\t%s\t%s\n", course.cno, course.cname,
course.lhour, course.credit, course.semester);
                retcode = SQLFetch(hstmt);
            }
        }
        cout << "===== " << endl;
    }

void select_student() {
    retcode = SQLAllocHandle(SQL_HANDLE_STMT, hdbc, &hstmt);
    retcode = SQLSetStmtAttr(hstmt, SQL_ATTR_ROW_BIND_TYPE,
(SQLPOINTER)SQL_BIND_BY_COLUMN, SQL_IS_INTEGER);
    retcode = SQLExecDirect(hstmt, (SQLCHAR *)"select * from student;",
SQL_NTS);
    if (retcode == SQL_SUCCESS || retcode == SQL_SUCCESS_WITH_INFO)
    {
        cout << "Query OK(0.00sec)" << endl;
    }

    struct student student;
    cout <<
    "===== " <<
endl;
    cout << "sno\tlname\tsex\tbdate\t\t\tdept\tclassno" << endl;
    long columnlen;
    SQLBindCol(hstmt, 1, SQL_CHAR, student.sno, 10, &columnlen);
    SQLBindCol(hstmt, 2, SQL_CHAR, student.sname, 30, &columnlen);
    SQLBindCol(hstmt, 3, SQL_CHAR, student.sex, 30, &columnlen);
    SQLBindCol(hstmt, 4, SQL_CHAR, student.bdate, 30, &columnlen);
    SQLBindCol(hstmt, 5, SQL_CHAR, student.dept, 30, &columnlen);
    SQLBindCol(hstmt, 6, SQL_CHAR, student.classno, 30, &columnlen);
    if (retcode<0)
    {
        cout << "没有执行语句" << endl;
    }
    retcode = SQLFetch(hstmt);
    while (retcode == SQL_ROW_SUCCESS || retcode ==
SQL_ROW_SUCCESS_WITH_INFO)
    {

```

```

        if (retcode == SQL_ROW_SUCCESS || retcode ==
SQL_ROW_SUCCESS_WITH_INFO) {
            printf("%s\t%s\t%s\t%s\t%s\t%s\n", student.sno,
student.sname, student.sex, student.bdate, student.dept,
student.classno);
            retcode = SQLFetch(hstmt);
        }
    }
    cout <<
"===== " <<
endl;
}

void select_sc() {
    retcode = SQLAllocHandle(SQL_HANDLE_STMT, hdbc, &hstmt);
    retcode = SQLSetStmtAttr(hstmt, SQL_ATTR_ROW_BIND_TYPE,
(SQLPOINTER)SQL_BIND_BY_COLUMN, SQL_IS_INTEGER);
    retcode = SQLExecDirect(hstmt, (SQLCHAR *)"select * from sc;",
SQL_NTS);
    if (retcode == SQL_SUCCESS || retcode == SQL_SUCCESS_WITH_INFO)
    {
        cout << "Query OK(0.00sec)" << endl;
    }
    struct sc sc;
    cout << "===== " << endl;
    cout << "sno\tcno\tgrade" << endl;
    long columnlen;
    SQLBindCol(hstmt, 1, SQL_CHAR, sc.sno, 10, &columnlen);
    SQLBindCol(hstmt, 2, SQL_CHAR, sc.cno, 5, &columnlen);
    SQLBindCol(hstmt, 3, SQL_CHAR, sc.grade, 5, &columnlen);
    if (retcode<0)
    {
        cout << "没有执行语句" << endl;
    }
    retcode = SQLFetch(hstmt);
    while (retcode == SQL_ROW_SUCCESS || retcode ==
SQL_ROW_SUCCESS_WITH_INFO)
    {
        if (retcode == SQL_ROW_SUCCESS || retcode ==
SQL_ROW_SUCCESS_WITH_INFO) {
            printf("%s\t%s\t%s\n", sc.sno, sc.cno, sc.grade);
            retcode = SQLFetch(hstmt);
        }
    }
}

```

```

        cout << "=====" << endl;
    }

    void operation() {
        retcode = SQLAllocHandle(SQL_HANDLE_STMT, hdbc, &hstmt);
        retcode = SQLSetStmtAttr(hstmt, SQL_ATTR_ROW_BIND_TYPE,
(SQLPOINTER)SQL_BIND_BY_COLUMN, SQL_IS_INTEGER);
        retcode = SQLPrepare(hstmt, (SQLCHAR*)input, SQL_NTS);
        retcode = SQLExecute(hstmt);
        if (retcode == SQL_SUCCESS || retcode == SQL_SUCCESS_WITH_INFO)
            cout << "operation succeed!" << endl;
        else
            cout << "operation failed!" << endl;
    }

    void check_result(char c) {
        switch (c)
        {
            case 't':
                select_student();
                break;
            case 'o':
                select_course();
                break;
            case 'c':
                select_sc();
                break;
            default:
                break;
        }
    }

    int main() {
        retcode = SQLAllocHandle(SQL_HANDLE_ENV, NULL, &henv); //分配环境句柄
        //retcode<0分配失败
        assert(retcode == SQL_SUCCESS || retcode == SQL_SUCCESS_WITH_INFO);

        //设置将使用的ODBC版本
        retcode = SQLSetEnvAttr(henv, SQL_ATTR_ODBC_VERSION,
(void*)SQL_OV_ODBC3, SQL_IS_INTEGER);
        //分配数据源句柄
        retcode = SQLAllocHandle(SQL_HANDLE_DBC, henv, &hdbc);
        //建立连接

```



```

retcode = SQLConnect(hdbc, (SQLCHAR *)"myodbc", SQL_NTS, (SQLCHAR
*)"root", SQL_NTS, (SQLCHAR *)"962464", SQL_NTSL);
//判断连接是否成功
if (retcode == SQL_SUCCESS || retcode == SQL_SUCCESS_WITH_INFO)
{
    cout << "success connection" << endl;
}
else
{
    cout << "fail connection" << endl;
}
while (1) {
    cout << "mysql>";
    cin.getline(input, 100);
    switch (input[0])
    {
        case 's': //查询
            switch (input[15]) //from 之后第二个字母 sc(c) student(t)
course(o)
            {
                case 't': //查询student表
                    select_student();
                    break;
                case 'o': //查询course表
                    select_course();
                    break;
                case 'c': //查询sc表
                    select_sc();
                    break;
                default:
                    cout << "Error: table does not exist! " << endl;
                    break;
            }
            break;
        case 'i': //插入
            operation();
            if (retcode == 0 || retcode == 1)
                check_result(input[13]);
            break;
        case 'u': //修改
            operation();
            if (retcode == 0 || retcode == 1)
                check_result(input[8]);
            break;
    }
}

```

```

        case 'd': //删除
            operation();
            if(retcode == 0 || retcode == 1)
                check_result(input[13]);
            break;
        case 'e': //退出
            cout << "Bye~" << endl;
            goto out;
            break;
        default:
            break;
    }
}


out:SQLFreeHandle(SQL_HANDLE_STMT, hstmt);
SQLDisconnect(hdbc);
SQLFreeHandle(SQL_HANDLE_DBC, hdbc);
SQLFreeHandle(SQL_HANDLE_ENV, henv);
system("pause");
return 0;
}

```

4. 实验结果

1) 查询 student 表

查询所有学生信息，查询结果如下图

 选择E:\Software\Microsoft Visual Studio\Projects\odbc\Debug\odbc.exe

```

success connection
mysql>select * from student;
Query OK (0.00sec)
=====
sno      sname    sex      bdate                dept      classno
30201    吴磊      男       1980-01-02 00:00:00  电信      3022
30202    袁青春    男       1980-01-02 00:00:00  电信      3022
30203    唐雷      男       1980-01-02 00:00:00  电信      3022
30204    吴霏      男       1980-01-02 00:00:00  电信      3022
30206    连洪炽    男       1980-01-02 00:00:00  电信      3022
30207    王金柱    男       1980-01-02 00:00:00  电信      3022
30208    苏广学    男       1980-01-02 00:00:00  电信      3022
30209    唐元亮    男       1980-01-02 00:00:00  电信      3022
30210    葛艳杰    男       1980-01-02 00:00:00  电信      3022
30211    张永超    男       1980-01-02 00:00:00  电信      3022
30212    张伟      男       1980-01-02 00:00:00  电信      3022
30213    孙刚      男       1980-01-02 00:00:00  电信      3022
30214    车平跃    男       1980-01-02 00:00:00  电信      3022
30215    张鑫      男       1980-01-02 00:00:00  电信      3022

```

2) 插入 student 表

```
mysql>insert into student values('1111','张三','男','1996-11-26
00:00:00','计算机','3146');
operation succeed!
Query OK (0.00sec)
```

sno	sname	sex	bdate	dept	classno
1111	张三	男	1996-11-26 00:00:00	计算机	3146
30201	吴磊	男	1980-01-02 00:00:00	电信	3022
30202	袁青春	男	1980-01-02 00:00:00	电信	3022
30203	唐雷	男	1980-01-02 00:00:00	电信	3022
30204	吴霏	男	1980-01-02 00:00:00	电信	3022
30206	连洪炽	男	1980-01-02 00:00:00	电信	3022
30207	王金柱	男	1980-01-02 00:00:00	电信	3022
30208	苏广学	男	1980-01-02 00:00:00	电信	3022
30209	唐元亮	男	1980-01-02 00:00:00	电信	3022
30210	葛艳杰	男	1980-01-02 00:00:00	电信	3022
30211	张永超	男	1980-01-02 00:00:00	电信	3022
30212	张伟	男	1980-01-02 00:00:00	电信	3022
30213	孙刚	男	1980-01-02 00:00:00	电信	3022

3) 修改 student 表

```
mysql>update student set sname='李四' where sno='1111';
operation succeed!
Query OK (0.00sec)
```

sno	sname	sex	bdate	dept	classno
1111	李四	男	1996-11-26 00:00:00	计算机	3146
30201	吴磊	男	1980-01-02 00:00:00	电信	3022
30202	袁青春	男	1980-01-02 00:00:00	电信	3022
30203	唐雷	男	1980-01-02 00:00:00	电信	3022
30204	吴霏	男	1980-01-02 00:00:00	电信	3022
30206	连洪炽	男	1980-01-02 00:00:00	电信	3022
30207	王金柱	男	1980-01-02 00:00:00	电信	3022
30208	苏广学	男	1980-01-02 00:00:00	电信	3022
30209	唐元亮	男	1980-01-02 00:00:00	电信	3022

4) 删除 student 表

```
mysql>delete from student where sno='1111';
operation succeed!
Query OK (0.00sec)
```

sno	sname	sex	bdate	dept	classno
30201	吴磊	男	1980-01-02 00:00:00	电信	3022
30202	袁青春	男	1980-01-02 00:00:00	电信	3022
30203	唐雷	男	1980-01-02 00:00:00	电信	3022
30204	吴霏	男	1980-01-02 00:00:00	电信	3022
30206	连洪炽	男	1980-01-02 00:00:00	电信	3022
30207	王金柱	男	1980-01-02 00:00:00	电信	3022
30208	苏广学	男	1980-01-02 00:00:00	电信	3022
30209	唐元亮	男	1980-01-02 00:00:00	电信	3022
30210	葛艳杰	男	1980-01-02 00:00:00	电信	3022

■ 实验小结

通过此次实验，我基本了解了数据库应用编程接口 ODBC 的基本原理和实现机制，包括为 ODBC 分配环境句柄，建立连接，分配语句句柄，绑定结果集，释放句柄以及连接；前期配置 ODBC 数据源时出现问题，需要设置成 32 位数据源，64 位数据源在 VS 中找不到数据库，运行时还需把项目字符集从 Unicode 设置成未设置，否则运行出错；编码过程中意识到每次进行一个 SQL 操作都要重新分配语句句柄，否则显示不出查询结果。