**数据库实验报告**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **实 验 人：** | **伍建霖** | | **学 号：** | **20337251** | **日 期：** | **2022.11.17** |
| **院（系）：** | **计算机学院** | | | **专业（班级）：** | **网络空间安全** | |
| **联系方式：** | **QQ773542531** | | |  |  | |
| **实验题目：** | | **7.1 基于ODBC的数据库应用开发实验** | | | | |

* + - 1. **实验目的**

掌握基于ODBC驱动的数据库应用开发方法。

* + - 1. **实验内容和要求**

设置ODBC驱动数据源，基于ODBC驱动的数据库连接方法，实现数据库数据操纵等应用开发常见功能。

* + - 1. **实验重点和难点**

实验重点：基于ODBC驱动的数据库连接方法、数据库数据操纵功能等。

实验难点：不同的数据库应用开发工具具有不同的开发框架和模式。能够较为熟练地使用所选择的应用开发工具，是实现本实验的难点。

* + - 1. **实验工具**

MySQL、SQL Server、Navicat、Dev C++（或其他开发工具）

* + - 1. **实验准备**

1. 编写实验程序的基本目标

在本实验中，以Dev C++开发环境（或其他开发工具）、MySQL和SQLServer数据库为例，实现一个完整的示例程序。该程序实现把MySQL数据源中的university数据库中instructor表数据复制到SQLServer的university数据库中的T71\_instructor表中。

1. 实验程序的基本框架如下

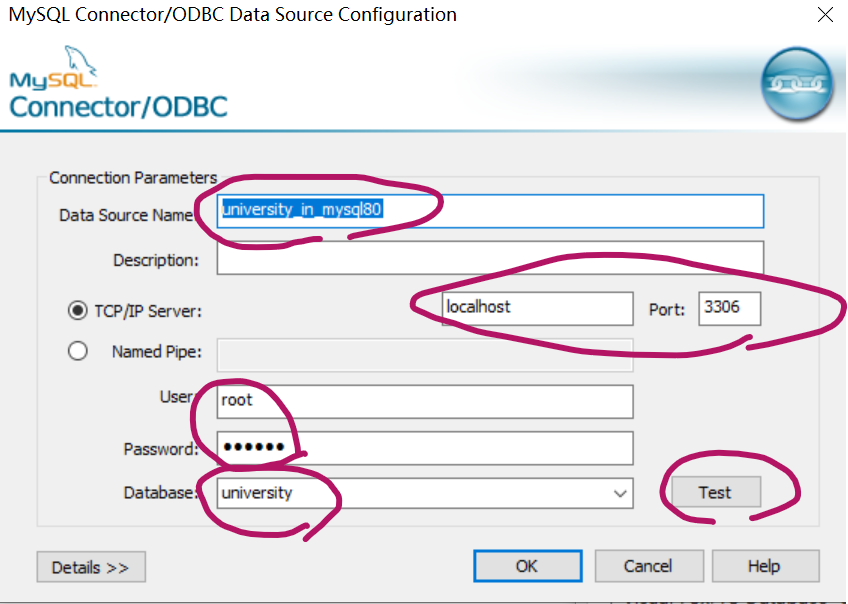
(1) 配置ODBC驱动数据源

基本方法：运用数据源管理工具来进行配置。打开Windows的“开始/设置/控制面板/管理工具/数据源(0DBC) ”， 出现“ODBC数据源管理器”界面， 点击“添加”按钮， 出现“创建数据源”界面， 选择希望安装数据源的驱动程序，出现“建立新的数据源”界面， 然后设置好数据源的名称、描述信息及服务器的名称等参数即可。

本实验要配置以下两个数据源：

1. 连接到MySQL数据库的数据源university\_in\_mysql80

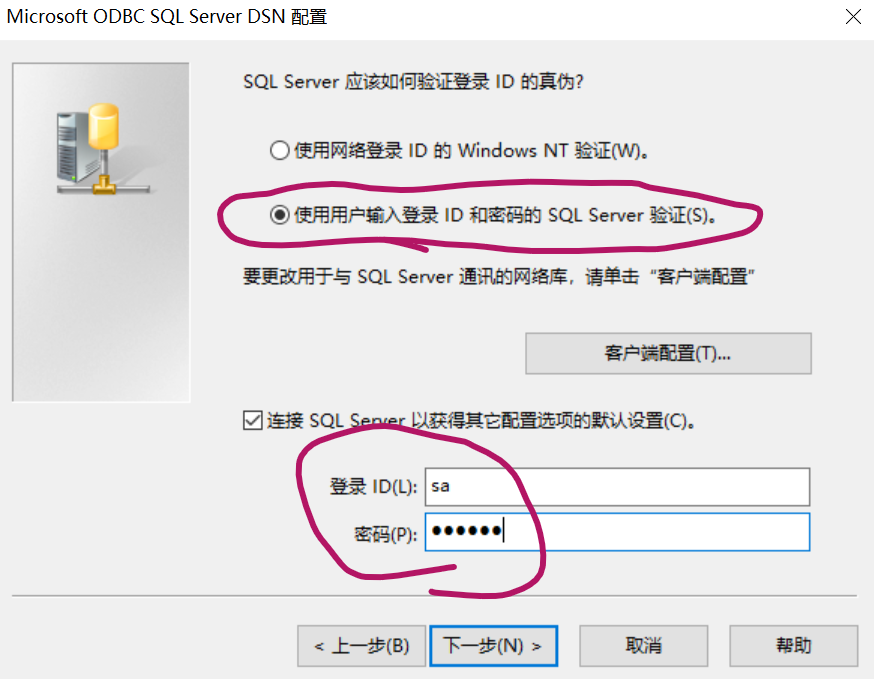
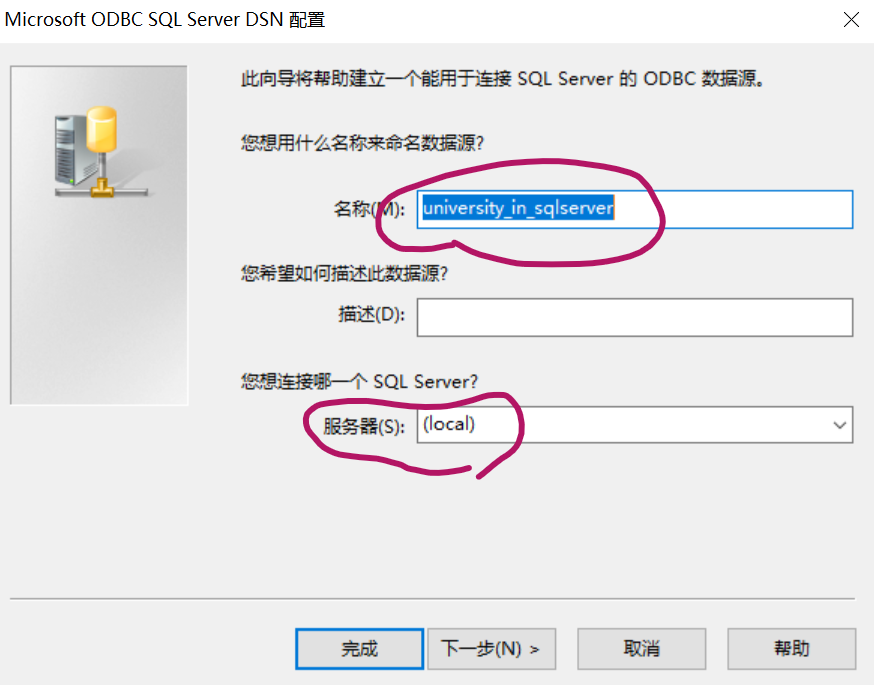
相关参数如下图所示：

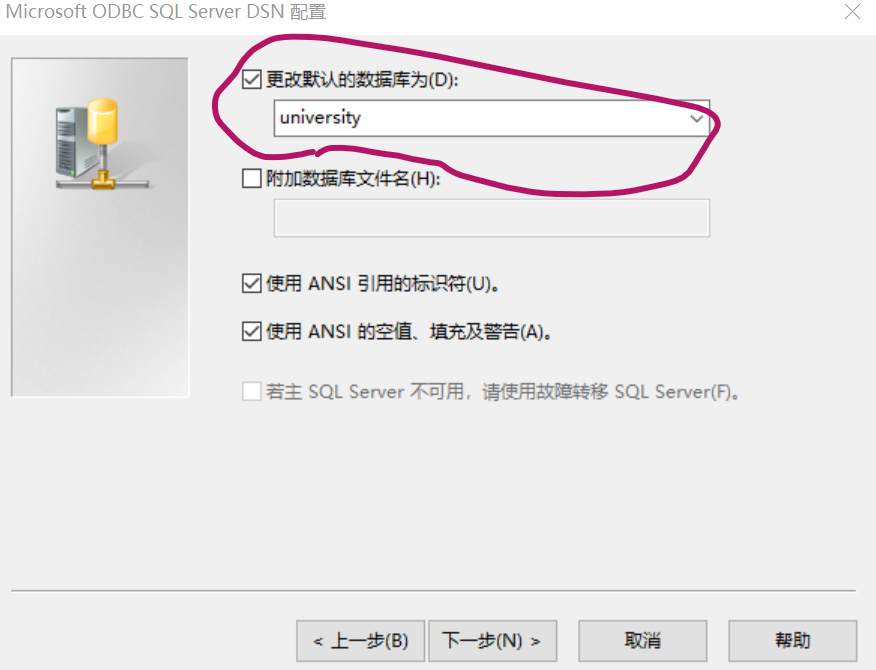


详细配置过程，可参阅页面“基于WINDOWS系统自带ODBC配置C/C++数据库程序开发环境”<https://www.freesion.com/article/6178440991/>

1. 连接到SQL Server数据库的数据源university\_in\_sqlserver

相关参数如下图所示：





详细配置过程，可参阅页面“C++连接数据库SQL Server ODBC配置”<https://blog.csdn.net/m0_53889370/article/details/118771465?spm=1001.2014.3001.5501>

对于修改SQL Server登陆认证方式，可参阅：<http://t.zoukankan.com/ling00218077-p-4240222.html>

(2) 基于ODBC驱动的数据库连接方法

/\*Step 1：定义句柄和变量\*/

/\*Step 2：初始化环境\*/

/\*Step 3：建立连接\*/

(3) 基于ODBC驱动的数据库数据操纵方法

/\*Step 4：初始化语句句柄\*/

/\*Step 5：两种方式执行语句\*/

/\*预编译带有参数的语句\*/

/\*直接执行SQL语句\*/

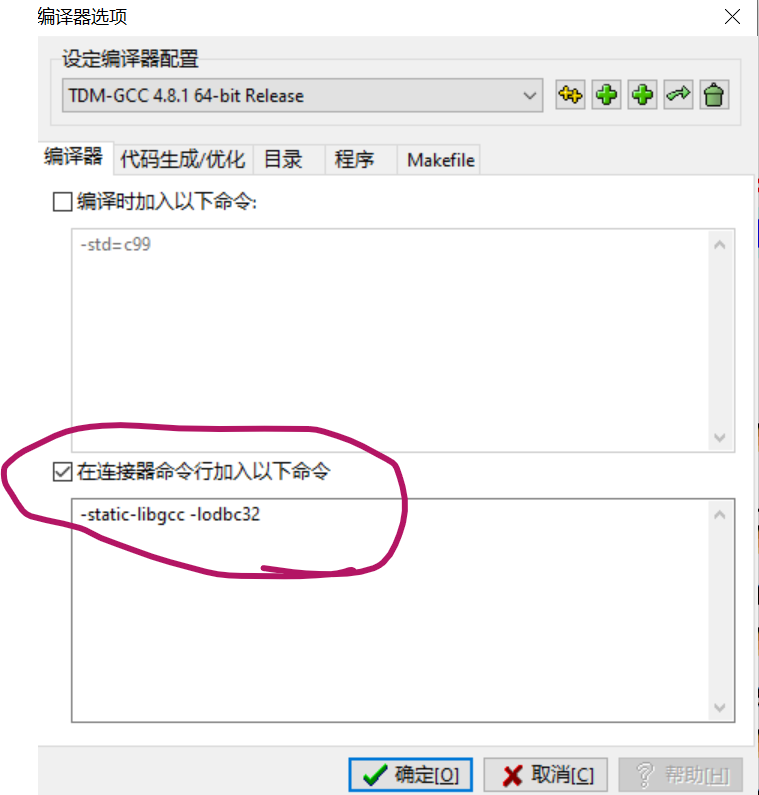
/\*Step 6：处理结果集并执行预编译后的语句\*/

(4) 中断基于ODBC驱动的数据库连接

/\*Step 7中止处理\*/

1. Dev c++ 的编译选项

要在Dev C++项目中使用ODBC,必须设置如下所示的编译选项（参阅<https://www.freesion.com/article/6178440991/#2devc_75>）：

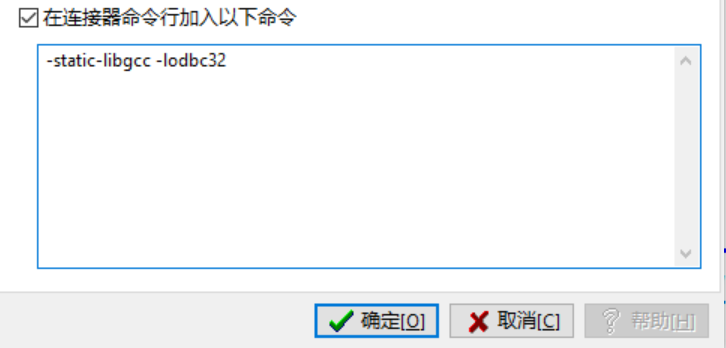


* + - 1. **实验过程**

1. 编写程序T71A, 显示MySQL数据源中的university数据库中instructor表中的所有记录。

**Step 1：创建项目**T71A，并将自动生成的 main.cpp 文件改名为 T71A.cpp

**Step 2：在“编译器选项”对话框中，勾选“在连接器命令行加入以下命令”，并且设置 -static-libgcc -lodbc32，即：**

****

**Step 3：在程序文件** T71A.cpp 中输入以下代码：

#include <windows.h>

#include <stdio.h>

#include <sql.h>

#include <sqlext.h>

int main() {

/\*Step 1：定义句柄和变量\*/

SQLHENV env; //环境句柄

SQLHDBC dbc; //连接句柄

SQLRETURN ret; //调用结果

SQLHSTMT stmt; //语句句柄

/\*Step 2：初始化环境\*/

SQLAllocEnv(&env);

//设置管理环境的属性

SQLSetEnvAttr(env, SQL\_ATTR\_ODBC\_VERSION, (void \*) SQL\_OV\_ODBC3, 0);

/\*Step 3：建立连接\*/

//分配连接句柄

ret = SQLAllocConnect(env, &dbc);

SQLCHAR \*server = (SQLCHAR \*)"university\_in\_mysql80";

SQLCHAR \*user = (SQLCHAR \*)"root";

SQLCHAR \*password = (SQLCHAR \*)"123456";

ret = SQLConnect(dbc, server, SQL\_NTS, user, SQL\_NTS, password, SQL\_NTS);

if(!SQL\_SUCCEEDED(ret)) //连接失败时返回错误值

return -1;

/\*Step 4：初始化语句句柄\*/

ret = SQLAllocStmt(dbc, &stmt);

/\*Step 5：两种方式执行语句\*/

SQLCHAR inst\_id[6] = {0};

SQLCHAR inst\_name[41] = {0};

SQLCHAR dept\_name[21] = {0};

SQLREAL inst\_salary;

SQLLEN lenOut1,lenOut2,lenOut3,lenOut4;

unsigned char query[] = "select id,name,dept\_name,salary from instructor";

/\*执行SQL语句\*/

ret = SQLExecDirect(stmt, (SQLCHAR \*) query, SQL\_NTS);

if (ret == SQL\_SUCCESS){

//将结果集中的属性列一一绑定至变量

SQLBindCol(stmt, 1, SQL\_C\_CHAR, inst\_id, sizeof(inst\_id), &lenOut1);

SQLBindCol(stmt, 2, SQL\_C\_CHAR, inst\_name, sizeof(inst\_name), &lenOut2);

SQLBindCol(stmt, 3, SQL\_C\_CHAR, dept\_name, sizeof(dept\_name), &lenOut3);

SQLBindCol(stmt, 4, SQL\_C\_FLOAT,&inst\_salary, 0, &lenOut4);

/\*Step 6：处理结果集并执行预编译后的语句\*/

while ((ret=SQLFetch(stmt))==SQL\_SUCCESS) {

printf("%s\t %s\t %s\t %g\n", inst\_id,inst\_name,dept\_name,inst\_salary);

}

}

else

printf("%d\n", ret);

/\*Step 7：中止处理\*/

SQLFreeStmt(stmt,SQL\_DROP);

SQLDisconnect(dbc);

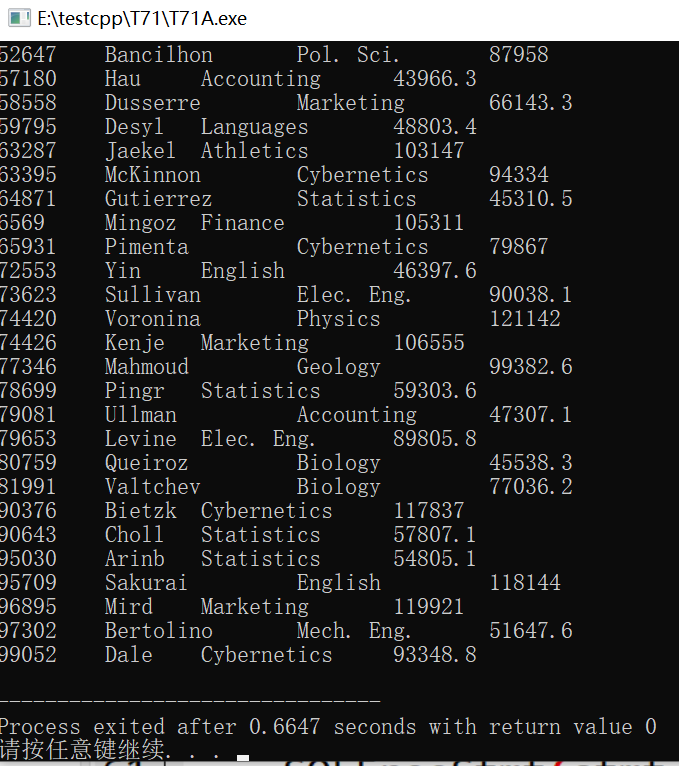
SQLFreeConnect(dbc);

SQLFreeEnv(env);

}

**Step 4：运行将显示**

验证截图如下：

****

1. 编写程序T71B, 根据程序T71A设计函数   
   int showTableByODBC(char \*dsn,char \*user\_id,char \*user\_password);   
   该函数显示指定数据源 dsn 中的university数据库中instructor表中的所有记录。

**Step 1：创建项目**T71B，并将自动生成的 main.cpp 文件改名为 T71B.cpp

**Step 2：在程序文件** T71B.cpp 中输入以下代码：

#include <windows.h>

#include <stdio.h>

#include <sql.h>

#include <sqlext.h>

int showTableByODBC(char \*dsn,char \*user\_id,char \*user\_password);

int main() {

showTableByODBC("university\_in\_mysql80","root","123456");

showTableByODBC("university\_in\_sqlserver","sa","123456");

return 0;

}

int showTableByODBC(char \*dsn,char \*user\_id,char \*user\_password)

{

/\*Step 1：定义句柄和变量\*/

SQLHENV env; //环境句柄

SQLHDBC dbc; //连接句柄

SQLRETURN ret; //调用结果

SQLHSTMT stmt; //语句句柄

/\*Step 2：初始化环境\*/

//SQLAllocHandle(SQL\_HANDLE\_ENV, SQL\_NULL\_HANDLE, &env);

SQLAllocEnv(&env);

//设置管理环境的属性

SQLSetEnvAttr(env, SQL\_ATTR\_ODBC\_VERSION, (void \*) SQL\_OV\_ODBC3, 0);

/\*Step 3：建立连接\*/

//分配连接句柄

//SQLAllocHandle(SQL\_HANDLE\_DBC, env, &dbc);

ret = SQLAllocConnect(env, &dbc);

SQLCHAR \*server = (SQLCHAR \*)dsn;

SQLCHAR \*user = (SQLCHAR \*)user\_id;

SQLCHAR \*password = (SQLCHAR \*)user\_password;

ret = SQLConnect(dbc, server, SQL\_NTS, user, SQL\_NTS, password, SQL\_NTS);

if(!SQL\_SUCCEEDED(ret)) //连接失败时返回错误值

return -1;

/\*Step 4：初始化语句句柄\*/

//SQLAllocHandle(SQL\_HANDLE\_STMT, dbc, &stmt);

ret = SQLAllocStmt(dbc, &stmt);

/\*Step 5：两种方式执行语句\*/

SQLCHAR inst\_id[6] = {0};

SQLCHAR inst\_name[41] = {0};

SQLCHAR dept\_name[21] = {0};

SQLREAL inst\_salary;

SQLLEN lenOut1,lenOut2,lenOut3,lenOut4;

// unsigned char query[] = "select dept\_name,sum(salary) from instructor group by dept\_name";

unsigned char query[] = "select id,name,dept\_name,salary from instructor";

/\*执行SQL语句\*/

ret = SQLExecDirect(stmt, (SQLCHAR \*) query, SQL\_NTS);

if (ret == SQL\_SUCCESS){

//将结果集中的属性列一一绑定至变量

SQLBindCol(stmt, 1, SQL\_C\_CHAR, inst\_id, sizeof(inst\_id), &lenOut1);

SQLBindCol(stmt, 2, SQL\_C\_CHAR, inst\_name, sizeof(inst\_name), &lenOut2);

SQLBindCol(stmt, 3, SQL\_C\_CHAR, dept\_name, sizeof(dept\_name), &lenOut3);

SQLBindCol(stmt, 4, SQL\_C\_FLOAT,&inst\_salary, 0, &lenOut4);

/\*Step 6：处理结果集并执行预编译后的语句\*/

while ((ret=SQLFetch(stmt))==SQL\_SUCCESS) {

printf("%s\t %s\t %s\t %g\n", inst\_id,inst\_name,dept\_name,inst\_salary);

}

}

else

printf("%d\n", ret);

/\*Step 7：中止处理\*/

SQLFreeStmt(stmt,SQL\_DROP);

SQLDisconnect(dbc);

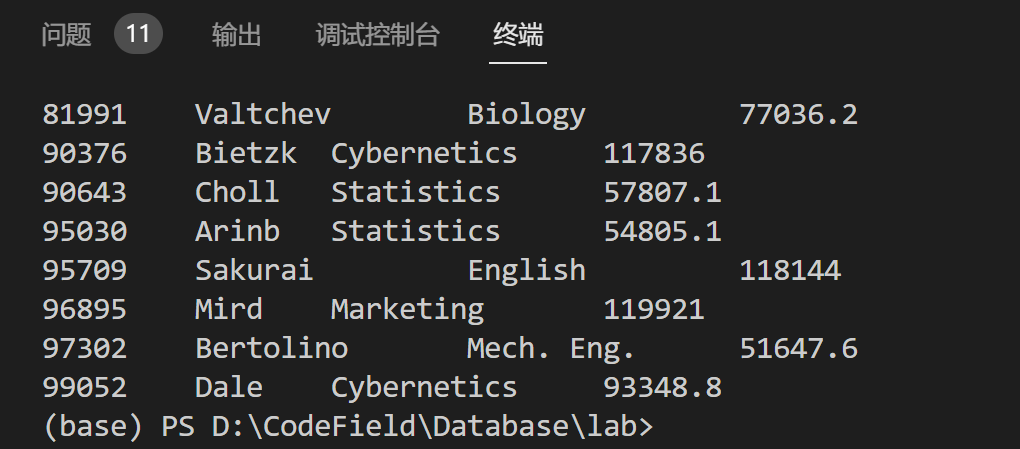
SQLFreeConnect(dbc);

SQLFreeEnv(env);

}

**Step 3：运行将显示**

验证截图如下：



1. 编写程序T71C,设计函数  
   int copyTableByODBC(char \*src\_dsn,char \*src\_user\_id,char \*src\_user\_password,   
    char \*target\_dsn, char \*target\_user\_id,char \*target\_user\_password);   
   该函数功能是：
   * 先在目标数据源target\_dsn创建T71\_instructor表（注：与instructor表结构一样）中，
   * 然后将源数据源src\_dsn中instructor表中的所有记录复制到目标数据源target\_dsn中的T71\_instructor表中。

**Step 1：创建项目**T71C，并将自动生成的 main.cpp 文件改名为 T71C.cpp

**Step 2：在程序文件** T71C.cpp 中输入以下代码：

#include <windows.h>

#include <stdio.h>

#include <sql.h>

#include <sqlext.h>

int copyTableByODBC(char \*src\_dsn,char \*src\_user\_id,char \*src\_user\_password,

char \*target\_dsn,char \*target\_user\_id,char \*target\_user\_password);

int main() {

copyTableByODBC("university\_in\_mysql80","root","123456","university\_in\_sqlserver","sa","123456");

copyTableByODBC("university\_in\_sqlserver","sa","123456","university\_in\_mysql80","root","123456");

return 0;

}

int copyTableByODBC(char \*src\_dsn,char \*src\_user\_id,char \*src\_user\_password,

char \*target\_dsn,char \*target\_user\_id,char \*target\_user\_password)

{

SQLRETURN ret; //调用结果

/\*Step 1：定义句柄和变量\*/

SQLHENV src\_env; //环境句柄

SQLHENV target\_env;

SQLHDBC src\_dbc; //连接句柄

SQLHDBC target\_dbc;

SQLHSTMT src\_stmt; //语句句柄

SQLHSTMT target\_stmt;

/\*Step 2：初始化环境\*/

SQLAllocEnv(&src\_env);

SQLAllocEnv(&target\_env);

//设置管理环境的属性

SQLSetEnvAttr(src\_env, SQL\_ATTR\_ODBC\_VERSION, (void \*) SQL\_OV\_ODBC3, 0);

SQLSetEnvAttr(target\_env, SQL\_ATTR\_ODBC\_VERSION, (void \*) SQL\_OV\_ODBC3, 0);

/\*Step 3：建立连接\*/

//分配连接句柄

ret = SQLAllocConnect(src\_env, &src\_dbc);

ret = SQLAllocConnect(target\_env, &target\_dbc);

ret = SQLConnect(src\_dbc, (SQLCHAR \*)src\_dsn, SQL\_NTS, (SQLCHAR \*)src\_user\_id, SQL\_NTS, (SQLCHAR \*)src\_user\_password, SQL\_NTS);

if(!SQL\_SUCCEEDED(ret)) //连接失败时返回错误值

return -1;

ret = SQLConnect(target\_dbc, (SQLCHAR \*)target\_dsn, SQL\_NTS, (SQLCHAR \*)target\_user\_id, SQL\_NTS, (SQLCHAR \*)target\_user\_password, SQL\_NTS);

if(!SQL\_SUCCEEDED(ret)) //连接失败时返回错误值

return -1;

/\*Step 4：初始化语句句柄\*/

ret = SQLAllocStmt(src\_dbc, &src\_stmt);

ret = SQLAllocStmt(target\_dbc, &target\_stmt);

/\*Step 5：两种方式执行语句\*/

/\*执行SQL语句\*/

/\*作为例子：创建目标表 T71\_instructor\*/

ret = SQLExecDirect(target\_stmt, (SQLCHAR \*) "create table T71\_instructor (ID varchar(5), name varchar(20) not null, dept\_name varchar(20), salary numeric(8,2) )", SQL\_NTS);

SQLCHAR inst\_id[6] = {0};

SQLCHAR inst\_name[41] = {0};

SQLCHAR dept\_name[21] = {0};

SQLREAL inst\_salary;

/\*方式一：预编译带有参数的语句\*/

//需要多次执行插入，因此预先声明插人语句

SQLLEN lenIn1 = SQL\_NTS,lenIn2 = SQL\_NTS,lenIn3 = SQL\_NTS,lenIn4=0;

ret=SQLPrepare(target\_stmt, (SQLCHAR \*)"INSERT INTO T71\_instructor(ID,name,dept\_name, salary) VALUES(?, ?, ?, ?)",SQL\_NTS);

if(ret==SQL\_SUCCESS)

{

//绑定参数

ret=SQLBindParameter(target\_stmt, 1,SQL\_PARAM\_INPUT, SQL\_C\_CHAR, SQL\_CHAR,

5,0,inst\_id, sizeof(inst\_id), &lenIn1);

ret=SQLBindParameter(target\_stmt, 2,SQL\_PARAM\_INPUT, SQL\_C\_CHAR, SQL\_CHAR,

20,0,inst\_name, sizeof(inst\_name), &lenIn2);

ret=SQLBindParameter(target\_stmt, 3,SQL\_PARAM\_INPUT, SQL\_C\_CHAR, SQL\_CHAR,

20,0,dept\_name, sizeof(dept\_name), &lenIn3);

ret=SQLBindParameter(target\_stmt, 4,SQL\_PARAM\_INPUT, SQL\_C\_FLOAT, SQL\_FLOAT,

8,2, &inst\_salary, 0, &lenIn4);

}

SQLLEN lenOut1,lenOut2,lenOut3,lenOut4;

unsigned char query[] = "select id,name,dept\_name,salary from instructor";

/\*执行SQL语句\*/

ret = SQLExecDirect(src\_stmt, (SQLCHAR \*) query, SQL\_NTS);

if (ret == SQL\_SUCCESS){

//将结果集中的属性列一一绑定至变量

SQLBindCol(src\_stmt, 1, SQL\_C\_CHAR, inst\_id, sizeof(inst\_id), &lenOut1);

SQLBindCol(src\_stmt, 2, SQL\_C\_CHAR, inst\_name, sizeof(inst\_name), &lenOut2);

SQLBindCol(src\_stmt, 3, SQL\_C\_CHAR, dept\_name, sizeof(dept\_name), &lenOut3);

SQLBindCol(src\_stmt, 4, SQL\_C\_FLOAT,&inst\_salary, 0, &lenOut4);

/\*Step 6：处理结果集并执行预编译后的语句\*/

while ((ret=SQLFetch(src\_stmt))==SQL\_SUCCESS) {

printf("%s\t %s\t %s\t %g\n", inst\_id,inst\_name,dept\_name,inst\_salary);

ret=SQLExecute(target\_stmt);

}

}

else

printf("%d\n", ret);

/\*Step 7：中止处理\*/

SQLFreeStmt(src\_stmt,SQL\_DROP);

SQLDisconnect(src\_dbc);

SQLFreeConnect(src\_dbc);

SQLFreeEnv(src\_env);

SQLFreeStmt(target\_stmt,SQL\_DROP);

SQLDisconnect(target\_dbc);

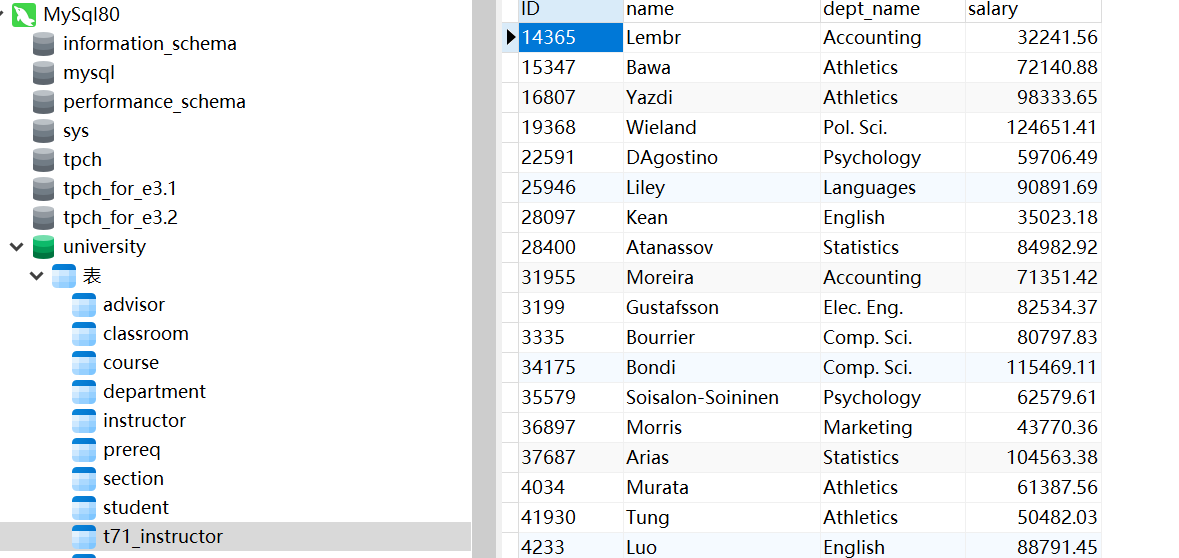
SQLFreeConnect(target\_dbc);

SQLFreeEnv(target\_env);

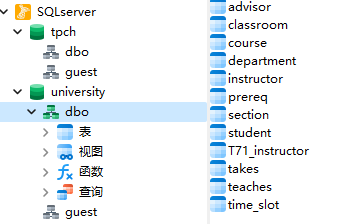
}

**Step 3：运行将显示**

验证截图1：在My SQL 数据库中的表T71\_instructor



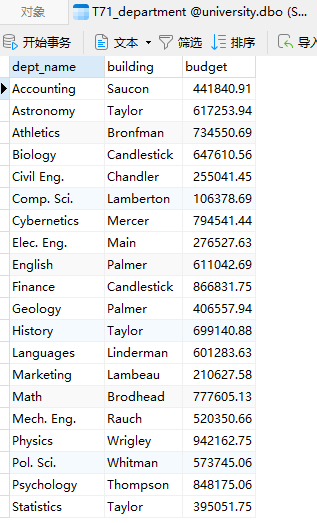
验证截图2：在SQL Server数据库中的表T71\_instructor

****

* + - 1. **与实验结果相关的文件**
* 程序T71A文件
* 程序T71B文件
* 程序T71C文件
* 程序T71More1文件
  + - 1. **思考题**

(1) 尝试编写程序 T71More1 (注：参考以上程序)， 实现将数据库university中的表department从数据源university\_in\_mysql80复制到数据源university\_in\_sqlserver中的T71\_department。

答：



#include <windows.h>

#include <stdio.h>

#include <sql.h>

#include <sqlext.h>

int copyTableByODBC(char \*src\_dsn,char \*src\_user\_id,char \*src\_user\_password,

            char \*target\_dsn,char \*target\_user\_id,char \*target\_user\_password);

int main() {

    copyTableByODBC("university\_in\_mysql80","root","dsbdsb","university\_in\_sqlserver","sa","dsbdsb");

    return 0;

}

int copyTableByODBC(char \*src\_dsn,char \*src\_user\_id,char \*src\_user\_password,

                    char \*target\_dsn,char \*target\_user\_id,char \*target\_user\_password)

{

    SQLRETURN ret; //调用结果

    /\*Step 1：定义句柄和变量\*/

    SQLHENV src\_env;  //环境句柄

    SQLHENV target\_env;

    SQLHDBC src\_dbc;  //连接句柄

    SQLHDBC target\_dbc;

    SQLHSTMT src\_stmt; //语句句柄

    SQLHSTMT target\_stmt;

    /\*Step 2：初始化环境\*/

    SQLAllocEnv(&src\_env);

    SQLAllocEnv(&target\_env);

    //设置管理环境的属性

    SQLSetEnvAttr(src\_env, SQL\_ATTR\_ODBC\_VERSION, (void \*) SQL\_OV\_ODBC3, 0);

    SQLSetEnvAttr(target\_env, SQL\_ATTR\_ODBC\_VERSION, (void \*) SQL\_OV\_ODBC3, 0);

    /\*Step 3：建立连接\*/

    //分配连接句柄

    ret = SQLAllocConnect(src\_env, &src\_dbc);

    ret = SQLAllocConnect(target\_env, &target\_dbc);

    ret = SQLConnect(src\_dbc, (SQLCHAR \*)src\_dsn, SQL\_NTS, (SQLCHAR \*)src\_user\_id, SQL\_NTS, (SQLCHAR \*)src\_user\_password, SQL\_NTS);

    if(!SQL\_SUCCEEDED(ret)) //连接失败时返回错误值

        return -1;

    ret = SQLConnect(target\_dbc, (SQLCHAR \*)target\_dsn, SQL\_NTS, (SQLCHAR \*)target\_user\_id, SQL\_NTS, (SQLCHAR \*)target\_user\_password, SQL\_NTS);

    if(!SQL\_SUCCEEDED(ret)) //连接失败时返回错误值

        return -1;

    /\*Step 4：初始化语句句柄\*/

    ret = SQLAllocStmt(src\_dbc, &src\_stmt);

    ret = SQLAllocStmt(target\_dbc, &target\_stmt);

    /\*Step 5：两种方式执行语句\*/

    /\*执行SQL语句\*/

    /\*作为例子：创建目标表 T71\_instructor\*/

    ret = SQLExecDirect(target\_stmt, (SQLCHAR \*) "create table T71\_department   (dept\_name  varchar(20) not null, building varchar(15), budget  numeric(12,2))", SQL\_NTS);

    SQLCHAR dept\_name[21] = {0};

    SQLCHAR dept\_building[16] = {0};

    SQLREAL dept\_budget;

    /\*方式一：预编译带有参数的语句\*/

    //需要多次执行插入，因此预先声明插人语句

    SQLLEN lenIn1 = SQL\_NTS,lenIn2 = SQL\_NTS,lenIn3=0;

    ret=SQLPrepare(target\_stmt, (SQLCHAR \*)"INSERT INTO T71\_department (dept\_name, building, budget) VALUES(?, ?, ?)",SQL\_NTS);

    if(ret==SQL\_SUCCESS)

    {

        //绑定参数

        ret=SQLBindParameter(target\_stmt, 1,SQL\_PARAM\_INPUT, SQL\_C\_CHAR, SQL\_CHAR,

                            20,0,dept\_name, sizeof(dept\_name), &lenIn1);

        ret=SQLBindParameter(target\_stmt, 2,SQL\_PARAM\_INPUT, SQL\_C\_CHAR, SQL\_CHAR,

                            15,0,dept\_building, sizeof(dept\_building), &lenIn2);

        ret=SQLBindParameter(target\_stmt, 3,SQL\_PARAM\_INPUT, SQL\_C\_FLOAT, SQL\_FLOAT,

                            12,2, &dept\_budget, 0, &lenIn3);

    }

    SQLLEN lenOut1,lenOut2,lenOut3;

    unsigned char query[] = "select dept\_name, building, budget from department";

    /\*执行SQL语句\*/

    ret = SQLExecDirect(src\_stmt, (SQLCHAR \*) query, SQL\_NTS);

    if (ret == SQL\_SUCCESS){

        //将结果集中的属性列一一绑定至变量

        SQLBindCol(src\_stmt, 1, SQL\_C\_CHAR, dept\_name, sizeof(dept\_name), &lenOut1);

        SQLBindCol(src\_stmt, 2, SQL\_C\_CHAR, dept\_building, sizeof(dept\_building), &lenOut2);

        SQLBindCol(src\_stmt, 3, SQL\_C\_FLOAT,&dept\_budget, 0, &lenOut3);

        /\*Step 6：处理结果集并执行预编译后的语句\*/

        while ((ret=SQLFetch(src\_stmt))==SQL\_SUCCESS) {

            printf("%s\t %s\t %g\n", dept\_name, dept\_building,dept\_budget);

            ret=SQLExecute(target\_stmt);

        }

    }

    else

        printf("%d\n", ret);

/\*Step 7：中止处理\*/

    SQLFreeStmt(src\_stmt,SQL\_DROP);

    SQLDisconnect(src\_dbc);

    SQLFreeConnect(src\_dbc);

    SQLFreeEnv(src\_env);

    SQLFreeStmt(target\_stmt,SQL\_DROP);

    SQLDisconnect(target\_dbc);

    SQLFreeConnect(target\_dbc);

    SQLFreeEnv(target\_env);

}

(2) 请调查目前比较流行的软件开发环境在基于ODBC驱动开发数据库应用方面各有什么优缺点?

答：软件开发环境具体是指什么。。。语言还是ide

VC++：功能强大但是不便使用

JAVA：使用简便，可跨平台，但性能不如VC++

* + - 1. **实验总结**