



数据库实验报告

实 验 伍建霖 学 20337251 日 期: 2022.11.17
人: 号:

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

一. 实验目的

掌握基于 ODBC 驱动的数据库应用开发方法。

二. 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。

三. 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。

四. 实验工具

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

五. 实验准备

(一) 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。

（二） 实验程序的基本框架如下

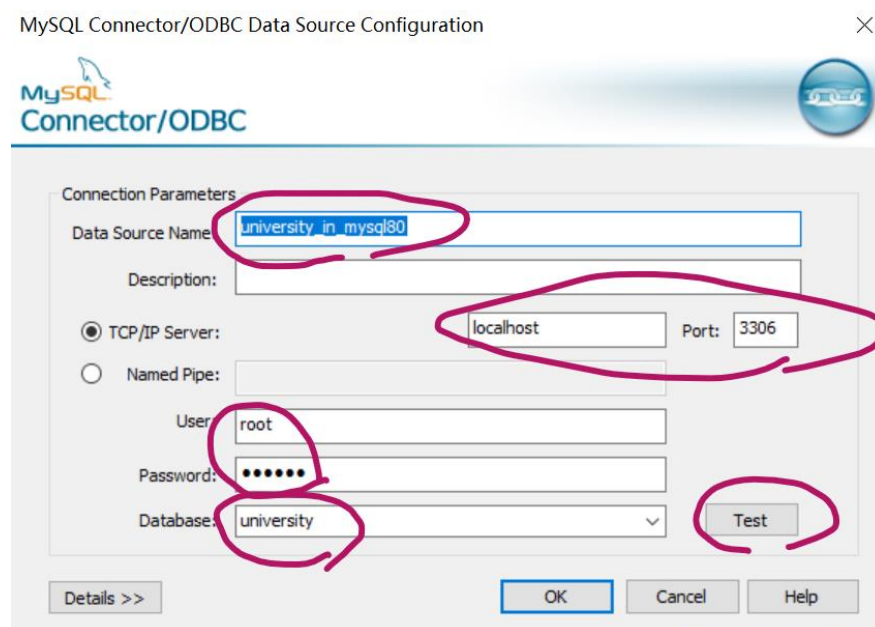
（1） 配置 ODBC 驱动数据源

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(ODBC)”，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：

a) 连接到 MySQL 数据库的数据源 university_in_mysql80

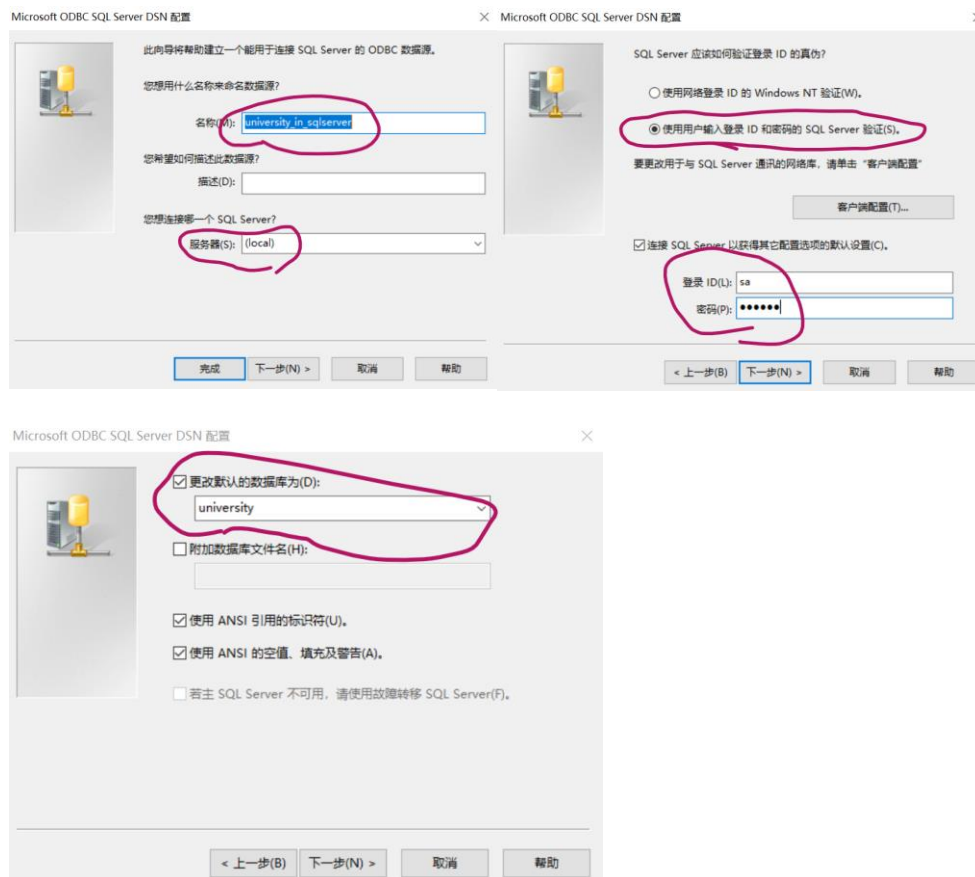
相关参数如下图所示：



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/>

b) 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*/
```

(三) 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) :

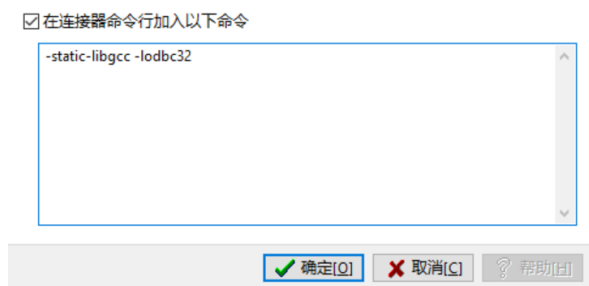


六. 实验过程

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 -lodbcc32, 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：

```
E:\testcpp\T71\T71A.exe
52647 Bancelhon Pol. Sci. 87958
57180 Hau Accounting 43966.3
58558 Dusserre Marketing 66143.3
59795 Desyl Languages 48803.4
63287 Jaekel Athletics 103147
63395 McKinnon Cybernetics 94334
64871 Gutierrez Statistics 45310.5
6569 Mingo Finance 105311
65931 Pimenta Cybernetics 79867
72553 Yin English 46397.6
73623 Sullivan Elec. Eng. 90038.1
74420 Voronina Physics 121142
74426 Kenje Marketing 106555
77346 Mahmoud Geology 99382.6
78699 Pingr Statistics 59303.6
79081 Ullman Accounting 47307.1
79653 Levine Elec. Eng. 89805.8
80759 Queiroz Biology 45538.3
81991 Valtchev Biology 77036.2
90376 Bietzk Cybernetics 117837
90643 Choll Statistics 57807.1
95030 Arinb Statistics 54805.1
95709 Sakurai English 118144
96895 Mird Marketing 119921
97302 Bertolino Mech. Eng. 51647.6
99052 Dale Cybernetics 93348.8

-----
Process exited after 0.6647 seconds with return value 0
请按任意键继续. . .
```

2. 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:

问题	11	输出	调试控制台	终端
81991	Valtchev	Biology	77036.2	
90376	Bietzk	Cybernetics	117836	
90643	Choll	Statistics	57807.1	
95030	Arinb	Statistics	54805.1	
95709	Sakurai	English	118144	
96895	Mird	Marketing	119921	
97302	Bertolino	Mech. Eng.	51647.6	
99052	Dale	Cybernetics	93348.8	
(base) PS D:\CodeField\Database\lab>				

3. 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_sqls
erver", "sa", "123456");
    copyTableByODBC("university_in_sqlserver", "sa", "123456", "university_in_mysq
l80", "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, 因此预先声明插入语句
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);
    }

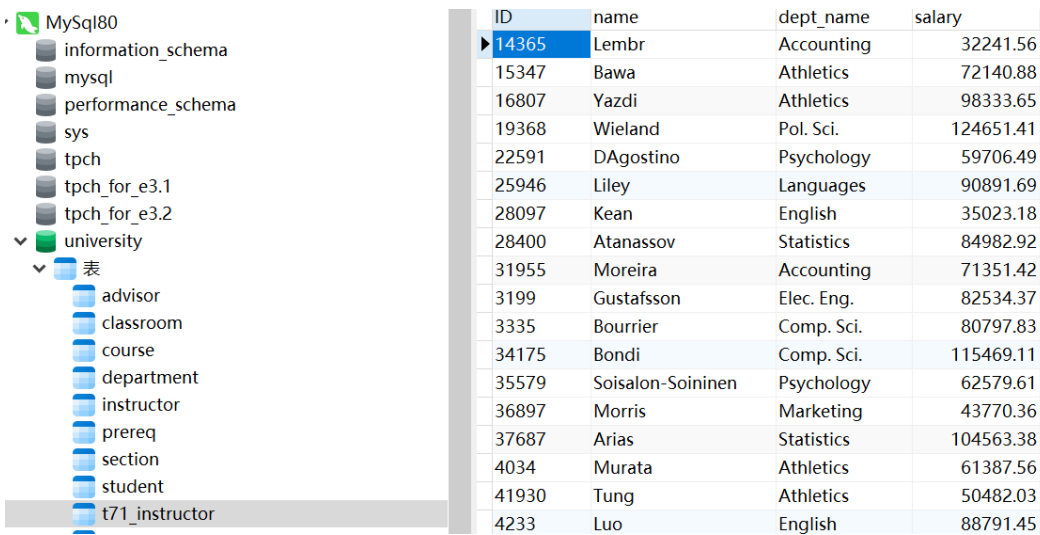
    SQLELEN lenOut1, lenOut2, lenOut3, lenOut4;
    unsigned char query[] = "select id,name,dept_name,salary from instructor";
    /*执行SQL语句*/
    ret = SQLExecDirect(src_stmt, (SQLCHAR *) query, SQL_NTS);
    if (ret == SQL_SUCCESS) {
        //将结果集中的属性列一一绑定至变量
        SQLBindCol(src_stmt, 1, SQL_C_CHAR, 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

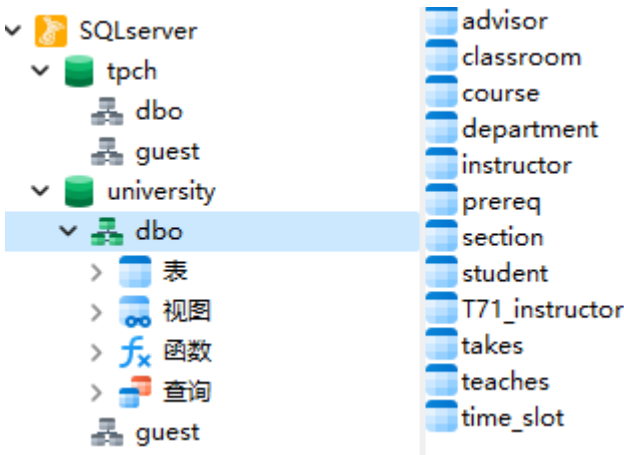
验证截图 1: 在 My SQL 数据库中的表 T71_instructor



The screenshot shows the MySQL 8.0 interface. On the left, the 'university' database is selected, and the 'T71_instructor' table is highlighted. On the right, the table's data is displayed in a grid view.

ID	name	dept_name	salary
14365	Lembr	Accounting	32241.56
15347	Bawa	Athletics	72140.88
16807	Yazdi	Athletics	98333.65
19368	Wieland	Pol. Sci.	124651.41
22591	D'Agostino	Psychology	59706.49
25946	Liley	Languages	90891.69
28097	Kean	English	35023.18
28400	Atanassov	Statistics	84982.92
31955	Moreira	Accounting	71351.42
3199	Gustafsson	Elec. Eng.	82534.37
3335	Bourrier	Comp. Sci.	80797.83
34175	Bondi	Comp. Sci.	115469.11
35579	Soisalon-Soininen	Psychology	62579.61
36897	Morris	Marketing	43770.36
37687	Arias	Statistics	104563.38
4034	Murata	Athletics	61387.56
41930	Tung	Athletics	50482.03
4233	Luo	English	88791.45

验证截图 2: 在 SQL Server 数据库中的表 T71_instructor



The screenshot shows the SQL Server Enterprise Manager interface. The 'university' database is selected, and the 'T71_instructor' table is highlighted in the list of tables.

Table Name
advisor
classroom
course
department
instructor
prereq
section
student
T71_instructor
takes
teaches
time_slot

七. 与实验结果相关的文件

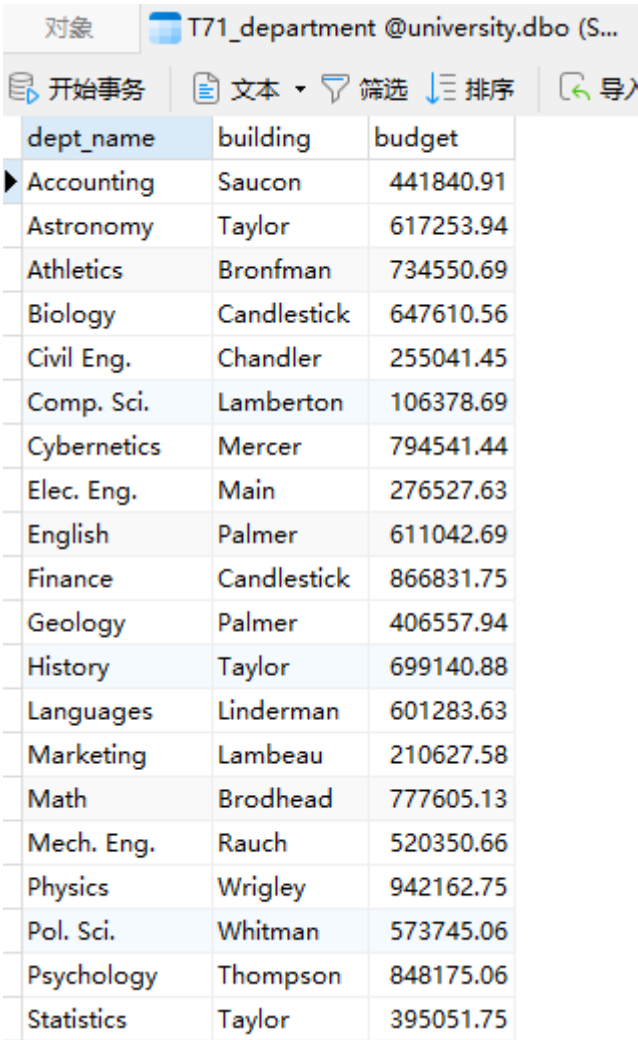
- 程序 T71A 文件
- 程序 T71B 文件
- 程序 T71C 文件

- 程序 T71More1 文件

八. 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：



对象	T71_department @university.dbo (S...	
开始事务	文本	筛选
排序	导	
dept_name	building	budget
Accounting	Saucon	441840.91
Astronomy	Taylor	617253.94
Athletics	Bronfman	734550.69
Biology	Candlestick	647610.56
Civil Eng.	Chandler	255041.45
Comp. Sci.	Lamberton	106378.69
Cybernetics	Mercer	794541.44
Elec. Eng.	Main	276527.63
English	Palmer	611042.69
Finance	Candlestick	866831.75
Geology	Palmer	406557.94
History	Taylor	699140.88
Languages	Linderman	601283.63
Marketing	Lambeau	210627.58
Math	Brodhead	777605.13
Mech. Eng.	Rauch	520350.66
Physics	Wrigley	942162.75
Pol. Sci.	Whitman	573745.06
Psychology	Thompson	848175.06
Statistics	Taylor	395051.75

```
#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, 因此预先声明插入语句
    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++

九. 实验总结