

数据库接口实验

【实验四】



目录

[一． 数据库接口实验 1](#_Toc513488328)

[1． 实验目的 1](#_Toc513488329)

[2． 实验环境 1](#_Toc513488330)

[3． 实验内容与要求 2](#_Toc513488331)

[4． 实验步骤及结果分析 3](#_Toc513488332)

[5． 实验小结 10](#_Toc513488333)

2018-5-7

[裴子祥 计科七班 学号2015211921]

[指导老师：杜军平]

# 数据库接口实验

1. 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

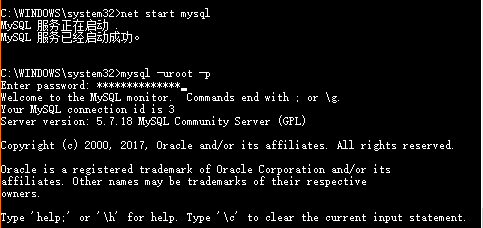
3．学习java语言，并采用jdbc接口方式对数据库进行访问

1. 实验环境

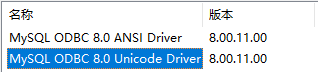
**Microsoft Windows 10 专业版 64位**



**数据库版本：5.7.18 MySQL Community Server (GPL)**



**ODBC: mysql-connector-odbc-8.0.11-winx64**



**编译器Dev-C++ 5.11, TDM-GCC 4.9.2 64-bit Release**



1. 实验内容与要求

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

(a) Step1. ODBC初始化，为ODBC分配环境句柄

(b) Step2. 建立应用程序与ODBC数据源的连接

(c) 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

(d) Step4. 检索查询结果集

(e) Step5. 结束数据库应用程序

3. 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连接句柄

(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

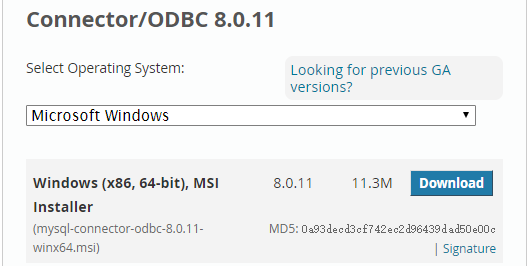
4. 释放与环境句柄相关的资源

1. 实验步骤及结果分析

**1、下载安装配置ODBC**

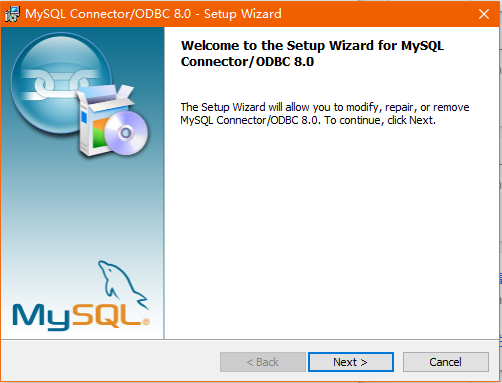
（1）下载ODBC

官网<https://dev.mysql.com/downloads/connector/odbc/> 下载最新ODBC版本



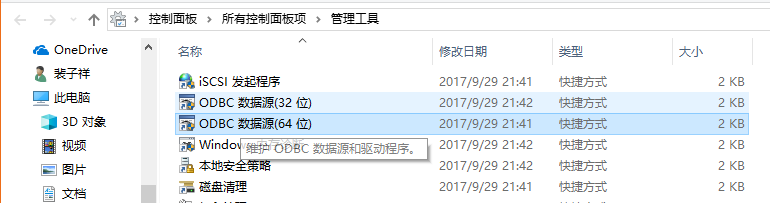
（2） 安装ODBC

根据提示依次安装



（3） 配置ODBC

● 进入控制面板->管理工具->ODBC数据源（64位）



● 添加ODBC到DNS

DSN：数据源名称

用户DSN：对当前用户可见，只能用于当前计算机上的数据源。

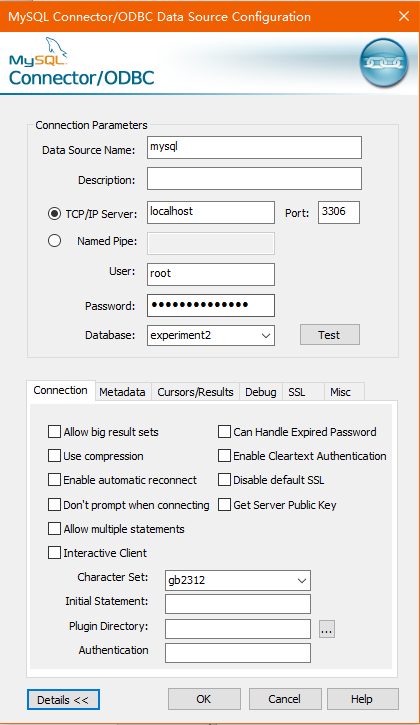
系统DSN：对当前计算机上所有的用户可见，包括NT服务器。

文件DSN：可有安装了相同驱动程序的用户共享的数据源。

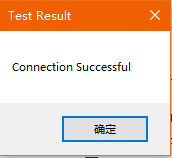
设置用户DSN，连接到本地mysql的数据库。

● 配置ODBC

编码格式“character set”设置为gb2312

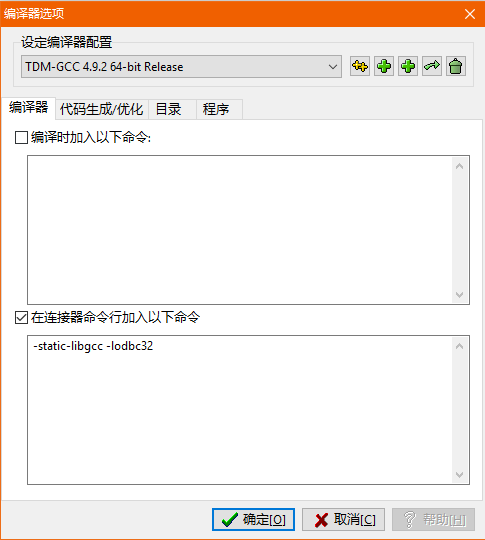


Test,successful



**2、配置DEVC++环境**

工具🡪编译选项



**3、编写代码并运行**

#include<stdio.h>

#include<stdlib.h>

#include"windows.h"

#include"sql.h"

#include"sqlext.h"

#include"sqltypes.h"

#include"odbcinst.h"

#pragma comment(lib, "odbc32.lib")

#pragma comment(lib, "libmysql.lib")

struct course\_cell

**{**

char cno**[**5**];**

char cname**[**20**];**

int lhour**;**

int credit**;**

char semester**[**5**];**

**};**

unsigned char sql\_select**[]** **=** "select \* from experiment2.course"**;** //SQL查询语句

unsigned char sql\_dolist**[**3**][**100**]** **=** **{**

"insert into experiment2.course values ('C06','数据库原理',48, 3,'')"**,**

"update experiment2.course set semester='#' where cno='C06'"**,**

"delete from experiment2.course where cno='C06'"

**};**

//插入、修改、删除语句

unsigned char do\_name**[**3**][**10**]** **=** **{** "insert"**,** "update"**,** "delete" **};**

void show\_course**(**HDBC hdbc**)**

//显示course的内容

**{**

HSTMT hstmt**;**

RETCODE retcode**;**

long long lenOut1**,** lenOut2**,** lenOut3**,** lenOut4**,** lenOut5**;**

struct course\_cell**\*** ccell**;**

ccell **=** **(**struct course\_cell**\*)**malloc**(sizeof(**struct course\_cell**));**

retcode **=** SQLAllocStmt**(**hdbc**,** **&**hstmt**);**//SQLALLocStmt；为语句句柄分配内存, 并返回语句句柄

**if** **(**retcode **==** SQL\_SUCCESS **||** retcode **==** SQL\_SUCCESS\_WITH\_INFO**)**

**{**

//SQLBindCol将应用程序的数据缓冲绑定到结果集的各列

SQLBindCol**(**hstmt**,** 1**,** SQL\_C\_CHAR**,** ccell**->**cno**,** 5**,** **&**lenOut1**);**//将数据缓冲绑定到结果集的列

SQLBindCol**(**hstmt**,** 2**,** SQL\_C\_CHAR**,** ccell**->**cname**,** 20**,** **&**lenOut2**);**

SQLBindCol**(**hstmt**,** 3**,** SQL\_C\_SLONG**,** **&**ccell**->**lhour**,** **sizeof(**int**),** **&**lenOut3**);**

SQLBindCol**(**hstmt**,** 4**,** SQL\_C\_SLONG**,** **&**ccell**->**credit**,** **sizeof(**int**),** **&**lenOut4**);**

SQLBindCol**(**hstmt**,** 5**,** SQL\_C\_CHAR**,** ccell**->**semester**,** 5**,** **&**lenOut5**);**

retcode **=** SQLExecDirect**(**hstmt**,** sql\_select**,** SQL\_NTS**);**

//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

**if** **(**retcode **==** SQL\_SUCCESS **||** retcode **==** SQL\_SUCCESS\_WITH\_INFO**)**

**{**

printf**(**"experiment2.course is as following:\n"**);**

printf **(**"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n"**);**

printf**(**"cno cname lhour credit semester\n"**);**

retcode **=** SQLFetch**(**hstmt**);**

/\*SQLFetch fetches the next rowset of data from the result set and returns data for all bound columns.

相当与SQLFetchAdvances和SQLGetData两个函数\*/

**while** **(**retcode **==** SQL\_SUCCESS **||** retcode **==** SQL\_SUCCESS\_WITH\_INFO**)**

**{**

printf**(**"%-10s%-20s%-15d%-16d%-5s\n"**,** ccell**->**cno**,** ccell**->**cname**,** ccell**->**lhour**,** ccell**->**credit**,** ccell**->**semester**);**

retcode **=** SQLFetch**(**hstmt**);**

**}**

printf **(**"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n \n"**);**

**}**

**}**

SQLFreeStmt**(**hstmt**,** SQL\_DROP**);**//释放语句句柄

**}**

int main**()**

**{**

HENV env**;** //定义环境句柄

HDBC hdbc**;** //定义链接句柄

HSTMT hstmt**;** //定义语句句柄

RETCODE retcode**;**

retcode **=** SQLAllocEnv**(&**env**);** //初始化ODBC环境，返回环境句柄

**if** **(**retcode **==** SQL\_SUCCESS **||** retcode **==** SQL\_SUCCESS\_WITH\_INFO**)**

**{**

retcode **=** SQLAllocConnect**(**env**,** **&**hdbc**);** //为连接句柄分配内存并返回链接句柄

**if** **(**retcode **==** SQL\_SUCCESS **||** retcode **==** SQL\_SUCCESS\_WITH\_INFO**)**

**{**

//printf("%d",env);

retcode **=** SQLConnect**(**hdbc**,** **(**SQLCHAR**\*)**"mysql"**,** SQL\_NTS**,** **(**SQLCHAR**\*)**"root"**,** SQL\_NTS**,** **(**SQLCHAR**\*)**"Pei13879619046"**,** SQL\_NTS**);** //连接一个SQL数据资料

**if** **(**retcode **==** SQL\_SUCCESS **||** retcode **==** SQL\_SUCCESS\_WITH\_INFO**)**

**{**

show\_course**(**hdbc**);** //显示初始的course内容

int i**;**

**for** **(**i **=** 0**;** i**<**3**;** i**++)** //依次执行插入、修改、删除操作

**{**

retcode **=** SQLAllocStmt**(**hdbc**,** **&**hstmt**);**

//为语句句柄分配内存，并返回语句句柄

retcode **=** SQLExecDirect**(**hstmt**,** **(**SQLCHAR**\*)**sql\_dolist**[**i**],** SQL\_NTS**);**

//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

printf**(**"%s "**,** do\_name**[**i**]);**

**if** **(**retcode **==** SQL\_SUCCESS **||** retcode **==** SQL\_SUCCESS\_WITH\_INFO**)**

printf**(**"success!\n"**);**

**else** printf**(**"fail!\n"**);**

SQLFreeStmt**(**hstmt**,** SQL\_DROP**);** //释放与 语句句柄相关的资源

show\_course**(**hdbc**);**//显示操作后的新course 内容

**}**

SQLDisconnect**(**hdbc**);** //切断连接

**}**

SQLFreeConnect**(**hdbc**);** //释放与连接句柄相关的资

**}**

SQLFreeEnv**(**env**);** //释放与环境句柄相关的资源

**}**

system**(**"pause"**);**

**return** 0**;**

**}**

**4、结果分析**

●对数据库操作指令

unsigned char sql\_select**[]** **=** "select \* from experiment2.course"**;** //SQL查询语句

unsigned char sql\_dolist**[**3**][**100**]** **=** **{**

"insert into experiment2.course values ('C06','数据库原理',48, 3,'')"**,**

"update experiment2.course set semester='#' where cno='C06'"**,**

"delete from experiment2.course where cno='C06'"

**};**

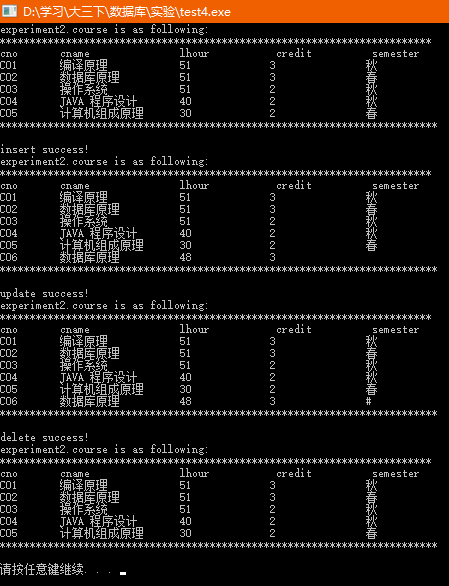
（1）"select \* from experiment2.course"

（2）"insert into test2.course values ('C06','数据库原理',48, 3,'春')";

（3）"update test2.course set semester='#' where cno='C06'";

（4）"delete from test2.course where cno='C06'";

●运行结果截图



添加一个实体('C06','数据库原理',48, 3,'春')后，course表里多了一行《数据库原理》信息，添加成功。

修改cno='C06'的semester后，这一项变为'#'，修改成功。

删除后，不再有cno='C06'的行，删除成功。

●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

SQLDriverConnect连接一个SQL数据资源，允许驱动器向用户询问信息

SQLALLocStmt为语句句柄分配内存, 并返回语句句柄

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

SQLFetchAdvances将游标移动到查询结果集的下一行(或第一行)

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

/\*SQLFetch fetches the next rowset of data from the result set and returns data for all bound columns.相当与SQLFetchAdvances和SQLGetData两个函数\*/

SQLFreeStmt释放与语句句柄相关的资源

SQLDisconnect 切断连接

SQLFreeConnect 释放与连接句柄相关的资源

SQLFreeEnv 释放与环境句柄相关的资源

1. 实验小结

**问题记录：**

●最开始设定字符集编解码方式为utf8，程序运行打印的是乱码，需要设置为gb2312编码标准。

●SQLBindCol(hstmt, 1, SQL\_C\_CHAR, ccell->cno, 5, &lenOut1);中第5个参数需要是long long \*才能进行转化，所以lenOut1需要定义为long long类型。

**心得总结：**

这次实验，我熟悉了通用数据库应用变成接口ODBC的使用方法，并对MySQL中的ODBC的使用和ODBC接口的语法和函数等有了进一步的了解。一个基于ODBC的应用程序对数据库的操作不依赖任何DBMS，不直接与DBMS打交道，所有数据库操作由对应DBMS的ODBC驱动程序完成。ODBC最大优点是能以统一方式处理所有数据库。ODBC 使用层次的方法来管理数据库，在数据库通信结构的每一层，对可能出现依赖数据库产品自身特性的地方，ODBC都引入一个公共接口以解决潜在的不一致性，从而很好地解决了基于数据库系统应用程序的相对独立性。

在devcpp中编写C语言，通过字符串编写SQL代码，并在数据库中进行相应操作。连接ODBC，我们通过编写的C语言代码可以访问数据库。这次实验让我系统地理解了ODBC的原理和使用，对于各个接口和函数有了更深的理解，之后在不断编码实践中更加熟练地运用。