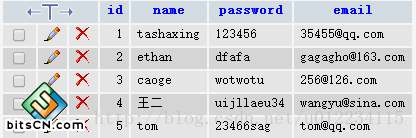
**一、环境配置**

1，装好mysql，新建一个C++控制台工程（从最简单的弄起，这个会了，可以往任何c++工程移植），在vs2010中设置，工程--属性--VC++目录--包含目录，将mysql server\include的绝对路径添加进去，例如C:\Program Files\MySQL\MySQL Server 5.6\include。将mysql server\lib文件夹下的libmysql.lib和libmysql.dll拷贝到工程目录下。

（也可以将include文件整个拷贝到工程目录下，然后在VC++目录里面设置相对路径）

如果安装的是wamp这种集成开发包，找不到include和lib也没关系，随便找个mysql免安装版根目录下的include文件夹和libmysql.lib以及libmysql.dll拷贝到工程目录，然后设置VC++目录即可。

新建一个数据库test，建立一张表user，如图



注意有些字段需要改字符编码为utf8或者gbk，防止中文乱码。

2，为工程添加附加依赖项wsock32.lib和libmysql.lib，一种方式是工程--属性--链接器--输入--附加依赖项，另一种是在程序开头用#pragma comment(lib,"xxx.lib")

3，为程序添加头文件"mysql.h"和WinSock.h

**二、示例代码**

#include <stdio.h>

#include <WinSock.h> //一定要包含这个，或者winsock2.h

#include "include/mysql.h" //引入mysql头文件(一种方式是在vc目录里面设置，一种是文件夹拷到工程目录，然后这样包含)

#include <Windows.h>

//包含附加依赖项，也可以在工程--属性里面设置

#pragma comment(lib,"wsock32.lib")

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

MYSQL mysql; //mysql连接

MYSQL\_FIELD \*fd; //字段列数组

char field[32][32]; //存字段名二维数组

MYSQL\_RES \*res; //这个结构代表返回行的一个查询结果集

MYSQL\_ROW column; //一个行数据的类型安全(type-safe)的表示，表示数据行的列

char query[150]; //查询语句

bool ConnectDatabase(); //函数声明

void FreeConnect();

bool QueryDatabase1(); //查询1

bool QueryDatabase2(); //查询2

bool InsertData();

bool ModifyData();

bool DeleteData();

int main(int argc,char \*\*argv)

{

ConnectDatabase();

QueryDatabase1();

InsertData();

QueryDatabase2();

ModifyData();

QueryDatabase2();

DeleteData();

QueryDatabase2();

FreeConnect();

system("pause");

return 0;

}

//连接数据库

bool ConnectDatabase()

{

//初始化mysql

mysql\_init(&mysql); //连接mysql，数据库

//返回false则连接失败，返回true则连接成功

if (!(mysql\_real\_connect(&mysql,"localhost", "root", "", "test",0,NULL,0))) //中间分别是主机，用户名，密码，数据库名，端口号（可以写默认0或者3306等），可以先写成参数再传进去

{

printf( "Error connecting to database:%s\n",mysql\_error(&mysql));

return false;

}

else

{

printf("Connected...\n");

return true;

}

}

//释放资源

void FreeConnect()

{

//释放资源

mysql\_free\_result(res);

mysql\_close(&mysql);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*数据库操作\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//其实所有的数据库操作都是先写个sql语句，然后用mysql\_query(&mysql,query)来完成，包括创建数据库或表，增删改查

//查询数据

bool QueryDatabase1()

{

sprintf(query, "select \* from user"); //执行查询语句，这里是查询所有，user是表名，不用加引号，用strcpy也可以

mysql\_query(&mysql,"set names gbk"); //设置编码格式（SET NAMES GBK也行），否则cmd下中文乱码

//返回0 查询成功，返回1查询失败

if(mysql\_query(&mysql, query)) //执行SQL语句

{

printf("Query failed (%s)\n",mysql\_error(&mysql));

return false;

}

else

{

printf("query success\n");

}

//获取结果集

if (!(res=mysql\_store\_result(&mysql))) //获得sql语句结束后返回的结果集

{

printf("Couldn't get result from %s\n", mysql\_error(&mysql));

return false;

}

//打印数据行数

printf("number of dataline returned: %d\n",mysql\_affected\_rows(&mysql));

//获取字段的信息

char \*str\_field[32]; //定义一个字符串数组存储字段信息

for(int i=0;i<4;i++) //在已知字段数量的情况下获取字段名

{

str\_field[i]=mysql\_fetch\_field(res)->name;

}

for(int i=0;i<4;i++) //打印字段

printf("%10s\t",str\_field[i]);

printf("\n");

//打印获取的数据

while (column = mysql\_fetch\_row(res)) //在已知字段数量情况下，获取并打印下一行

{

printf("%10s\t%10s\t%10s\t%10s\n", column[0], column[1], column[2],column[3]); //column是列数组

}

return true;

}

bool QueryDatabase2()

{

mysql\_query(&mysql,"set names gbk");

//返回0 查询成功，返回1查询失败

if(mysql\_query(&mysql, "select \* from user")) //执行SQL语句

{

printf("Query failed (%s)\n",mysql\_error(&mysql));

return false;

}

else

{

printf("query success\n");

}

res=mysql\_store\_result(&mysql);

//打印数据行数

printf("number of dataline returned: %d\n",mysql\_affected\_rows(&mysql));

for(int i=0;fd=mysql\_fetch\_field(res);i++) //获取字段名

strcpy(field[i],fd->name);

int j=mysql\_num\_fields(res); // 获取列数

for(int i=0;i<j;i++) //打印字段

printf("%10s\t",field[i]);

printf("\n");

while(column=mysql\_fetch\_row(res))

{

for(int i=0;i<j;i++)

printf("%10s\t",column[i]);

printf("\n");

}

return true;

}

//插入数据

bool InsertData()

{

sprintf(query, "insert into user values (NULL, 'Lilei', 'wyt2588zs','lilei23@sina.cn');"); //可以想办法实现手动在控制台手动输入指令

if(mysql\_query(&mysql, query)) //执行SQL语句

{

printf("Query failed (%s)\n",mysql\_error(&mysql));

return false;

}

else

{

printf("Insert success\n");

return true;

}

}

//修改数据

bool ModifyData()

{

sprintf(query, "update user set email='lilei325@163.com' where name='Lilei'");

if(mysql\_query(&mysql, query)) //执行SQL语句

{

printf("Query failed (%s)\n",mysql\_error(&mysql));

return false;

}

else

{

printf("Insert success\n");

return true;

}

}

//删除数据

bool DeleteData()

{

/\*sprintf(query, "delete from user where id=6");\*/

char query[100];

printf("please input the sql:\n");

gets(query); //这里手动输入sql语句

if(mysql\_query(&mysql, query)) //执行SQL语句

{

printf("Query failed (%s)\n",mysql\_error(&mysql));

return false;

}

else

{

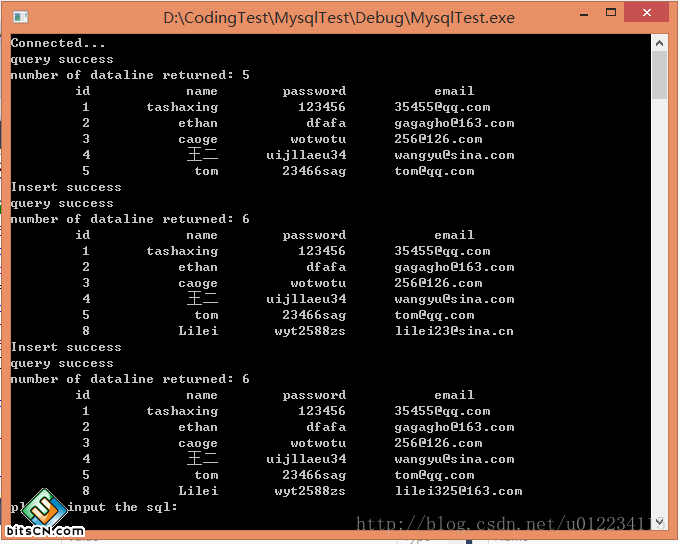
printf("Insert success\n");

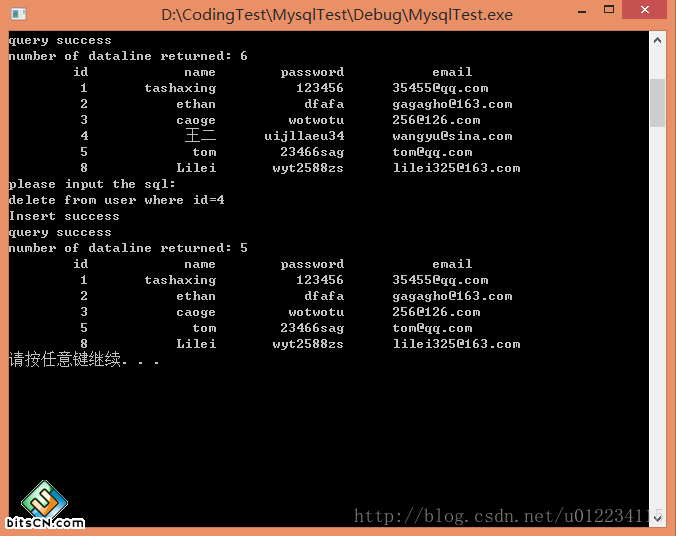
return true;

}

}

运行结果：





**三、mysql API接口汇总**

 mysql\_affected\_rows() 返回被最新的UPDATE, DELETE或INSERT查询影响的行数。

 mysql\_close() 关闭一个服务器连接。

 mysql\_connect() 连接一个MySQL服务器。该函数不推荐；使用mysql\_real\_connect()代替。

 mysql\_change\_user() 改变在一个打开的连接上的用户和数据库。

 mysql\_create\_db() 创建一个数据库。该函数不推荐；而使用SQL命令CREATE DATABASE。

 mysql\_data\_seek() 在一个查询结果集合中搜寻一任意行。

 mysql\_debug() 用给定字符串做一个DBUG\_PUSH。

 mysql\_drop\_db() 抛弃一个数据库。该函数不推荐；而使用SQL命令DROP DATABASE。

 mysql\_dump\_debug\_info() 让服务器将调试信息写入日志文件。

 mysql\_eof() 确定是否已经读到一个结果集合的最后一行。这功能被反对; mysql\_errno()或mysql\_error()可以相反被使用。

 mysql\_errno() 返回最近被调用的MySQL函数的出错编号。

 mysql\_error() 返回最近被调用的MySQL函数的出错消息。

 mysql\_escape\_string() 用在SQL语句中的字符串的转义特殊字符。

 mysql\_fetch\_field() 返回下一个表字段的类型。

 mysql\_fetch\_field\_direct () 返回一个表字段的类型，给出一个字段编号。

 mysql\_fetch\_fields() 返回一个所有字段结构的数组。

 mysql\_fetch\_lengths() 返回当前行中所有列的长度。

 mysql\_fetch\_row() 从结果集合中取得下一行。

 mysql\_field\_seek() 把列光标放在一个指定的列上。

 mysql\_field\_count() 返回最近查询的结果列的数量。

 mysql\_field\_tell() 返回用于最后一个mysql\_fetch\_field()的字段光标的位置。

 mysql\_free\_result() 释放一个结果集合使用的内存。

 mysql\_get\_client\_info() 返回客户版本信息。

 mysql\_get\_host\_info() 返回一个描述连接的字符串。

 mysql\_get\_proto\_info() 返回连接使用的协议版本。

 mysql\_get\_server\_info() 返回服务器版本号。

 mysql\_info() 返回关于最近执行得查询的信息。

 mysql\_init() 获得或初始化一个MYSQL结构。

 mysql\_insert\_id() 返回有前一个查询为一个AUTO\_INCREMENT列生成的ID。

 mysql\_kill() 杀死一个给定的线程。

 mysql\_list\_dbs() 返回匹配一个简单的正则表达式的数据库名。

 mysql\_list\_fields() 返回匹配一个简单的正则表达式的列名。

 mysql\_list\_processes() 返回当前服务器线程的一张表。

 mysql\_list\_tables() 返回匹配一个简单的正则表达式的表名。

 mysql\_num\_fields() 返回一个结果集合重的列的数量。

 mysql\_num\_rows() 返回一个结果集合中的行的数量。

 mysql\_options() 设置对mysql\_connect()的连接选项。

 mysql\_ping() 检查对服务器的连接是否正在工作，必要时重新连接。

 mysql\_query() 执行指定为一个空结尾的字符串的SQL查询。

 mysql\_real\_connect() 连接一个MySQL服务器。

 mysql\_real\_query() 执行指定为带计数的字符串的SQL查询。

 mysql\_reload() 告诉服务器重装授权表。

 mysql\_row\_seek() 搜索在结果集合中的行，使用从mysql\_row\_tell()返回的值。

 mysql\_row\_tell() 返回行光标位置。

 mysql\_select\_db() 连接一个数据库。

 mysql\_shutdown() 关掉数据库服务器。

 mysql\_stat() 返回作为字符串的服务器状态。

 mysql\_store\_result() 检索一个完整的结果集合给客户。

 mysql\_thread\_id() 返回当前线程的ID。

 mysql\_use\_result() 初始化一个一行一行地结果集合的检索。