

OTL 编程配置使用说明

一、参考资料

<http://blog.csdn.net/sherlockhua/article/details/4353531>

<http://www.xuebuyuan.com/1123237.html>

<http://blog.csdn.net/qianmianyuan/article/details/8891002>

http://blog.csdn.net/roger_77/article/details/633132

二、OTL 简介

OTL 是 Oracle, Odbc and DB2-CLI Template Library 的缩写, 是一个 C++ 编译中操控关系数据库的模板库, 它目前几乎支持所有的当前各种主流数据库, 例如 Oracle、MS SQL Server、Sybase、Informix、MySQL、DB2、Interbase / Firebird、PostgreSQL、SQLite、SAP/DB、TimesTen、MS ACCESS 等等。

OTL 中直接操作 Oracle 主要是通过 Oracle 提供的 OCI(Oracle Call Interface) 接口进行; 进行 DB2 数据库操作则是通过 CLI(Call Level Interface)接口来进行; 对于 MS 的数据库和其它一些数据库, OTL 则通过 ODBC(Open Database Connectivity)来操作, 例如 MySQL。

三、使用说明

3.1 宏定义

OTL 使用不同的数据库连接, 是由程序开始的宏定义来指定的。OTL 根据此宏定义来初始化数据库连接环境。

OTL 中用来区分连接方式的宏定义主要有下面这些:

OTL_ORA10G , OTL_ORA10G _R2, OTL_ODBC, OTL_ODBC_MYSQL ,
OTL_DB2_CLI ...

不同的宏对应的数据库 API, 具体说明如下:

宏定义名	说明
OTL_DB2_CLI	for DB2 Call Level Interface (CLI)
OTL_INFORMIX_CLI	for Informix Call Level Interface for Unix (when OTL_ODBC_UNIX is enabled).
OTL_IODBC_BSD	for ODBC on BSD Unix, when iODBC package is used
OTL_ODBC	for ODBC

OTL_ODBC_MYSQL	for <u>MyODBC/MySQL</u> . The difference between OTL_ODBC_MYSQL and OTL_ODBC is that transactional ODBC function calls are turned off for OTL_ODBC_MYSQL, since MySQL does not have transactions
OTL_ODBC_POSTGRES	for the PostgreSQL ODBC driver 3.5 (and higher) that are connected to PostgreSQL 7.4 / 8.0 (and higher) servers.
OTL_ODBC_UNIX	for ODBC bridges in Unix
OTL_ODBC_zOS	for ODBC on IBM zOS.
OTL_ODBC_XTG_IBASE6	for <u>Interbase 6.x</u> via <u>XTG Systems' ODBC driver</u> . The reason for introducing this #define is that the ODBC driver is the only Open Source ODBC driver for Interbase. Other drivers, like Easysoft's ODBC for Interbase, are commercial products, and it beats the purpose of using Interbase, as an Open Source.database server.
OTL_ORA7	for OCI7
OTL_ORA8	for OCI8
OTL_ORA8I	for OCI8i
OTL_ORA9I	for OCI9i. All code that compiles and works under #define OTL_ORA7, OTL_ORA8, and OTL_ORA8I, should work when OTL_ORA9I is used
OTL_ORA10G	for OCI10g. All code that compiles and works under #define OTL_ORA7, OTL_ORA8, OTL_ORA8I, OTL_ORA9I, should work with OTL_ORA10G.
OTL_ORA10G_R2	for OCI10g, Release 2 (Oracle 10.2). All code that compiles and works under #define OTL_ORA7, OTL_ORA8, OTL_ORA8I, OTL_ORA9I, and OTL_ORA10G should work with OTL_ORA10G_R2 .

3.2 链接库

我们在编译 OTL 的程序时，需要使用到相应的数据库 API，这就要程序在编译时联接 lib 库文件，不同的数据库对应的 lib 文件所在位置各不相同，下面是分别在 windows 与 Unix 下的数据库 API 所需要的头文件及 lib 文件所在的位置列表：

API	API header files for Windows	API libraries for Windows
OCI7	In <ORACLE_HOME>/oci/inc lude	<ORACLE_HOME>/oci/lib/<compiler_specific>/ociw32.lib
OCI8	In <ORACLE_HOME>/oci/inc lude	<ORACLE_HOME>/oci/lib/<compiler_specific>/oci.lib
OCI8i	In <ORACLE_HOME>/oci/inc lude	<ORACLE_HOME>/oci/lib/<compiler_specific>/oci.lib
OCI9i	In	<ORACLE_HOME>/oci/lib/<compiler_specific>/oci.lib

	<ORACLE_HOME>/oci/inc lude	
OCI10g	In	<ORACLE_HOME>/oci/lib/<compiler_specific>/oci.lib <ORACLE_HOME>/oci/inc lude
ODBC	Normally, in one of the C++ compiler system directories, no need to include explicitly.	Normally, in one of the C++ compiler system directories: odbc32.lib
DB2 CLI	In <DB2_HOME>/include	<DB2_HOME>/lib/db2api.lib <DB2_HOME>/lib/db2cli.lib

API	API header files for Unix	API libraries for Unix
OCI7	-I\$(ORACLE_HOME)/rdbms/demo -I\$(ORACLE_HOME)/rdbms/public	-L\$(ORACLE_HOME)/lib/ -lclntsh
OCI8	-I\$(ORACLE_HOME)/rdbms/demo -I\$(ORACLE_HOME)/rdbms/public	-L\$(ORACLE_HOME)/lib/ -lclntsh
OCI8i	-I\$(ORACLE_HOME)/rdbms/demo -I\$(ORACLE_HOME)/rdbms/public	-L\$(ORACLE_HOME)/lib/ -lclntsh
OCI9i	-I\$(ORACLE_HOME)/rdbms/demo -I\$(ORACLE_HOME)/rdbms/public	-L\$(ORACLE_HOME)/lib/ -lclntsh
OCI10g	-I\$(ORACLE_HOME)/rdbms/demo -I\$(ORACLE_HOME)/rdbms/public	-L\$(ORACLE_HOME)/lib/ -lclntsh
ODBC	ODBC bridge specific	ODBC bridge specific
DB2 CLI	-I/<DB2_HOME>/sqllib/include	-L/<DB2_HOME>/sqllib/lib -ldb2

四、 Oracle 的配置使用

4.1 VS2008 中具体配置

配置实例环境： 操作系统： win7 使用数据库： oracle10g

4.1.1 引入头文件

在 VS 中依次选择“工具→选项→项目和解决方案→VC++目录”，在右上方组合框中选择“包含文件”，把 oci 目录包含进来(如图 1)。例如我的是：
D:\oracle\product\10.2.0\db_1\oci\include。

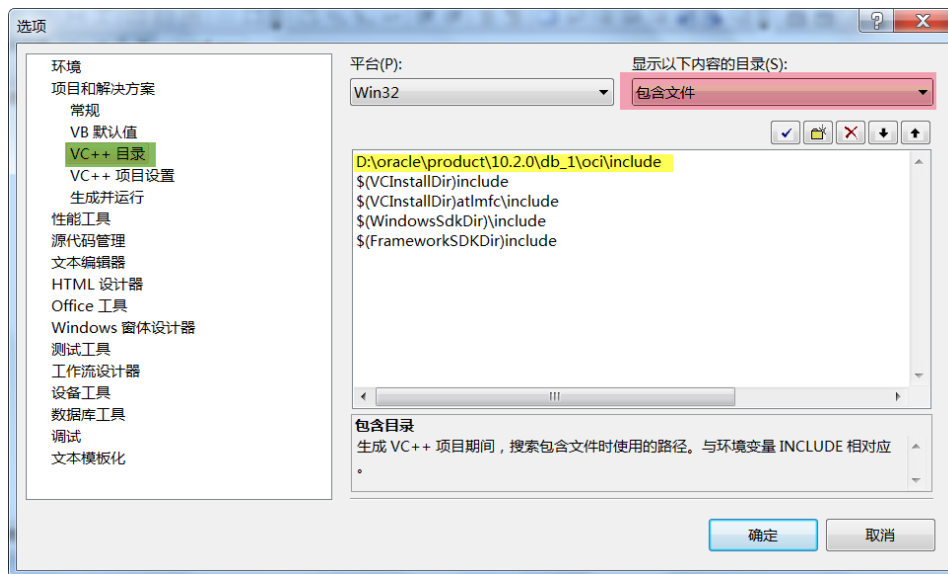


图 1 头文件引入

图 2 中给出了 oracle10g 中的 API 头文件。

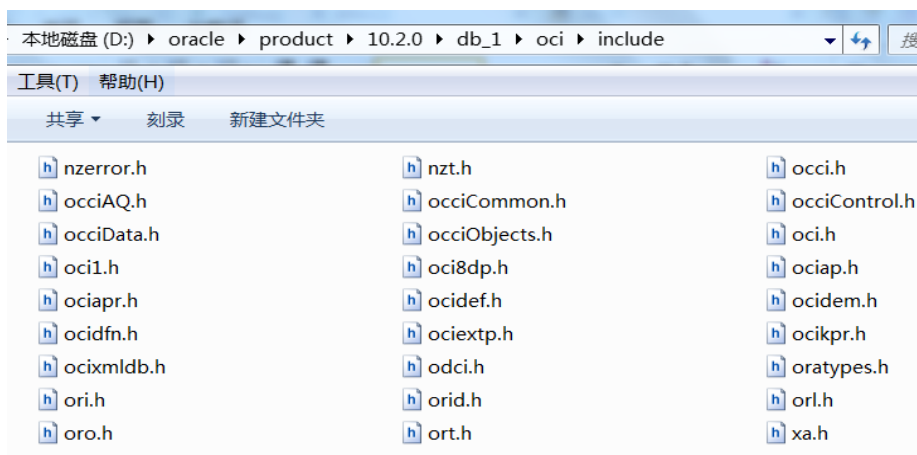


图 2 oracle10g 中的头文件

注：对于一个 OTL 程序而言，当未引入数据库 API 所需要的头文件时一般会出现“无法打开‘oci.h’的错误”，如下图 3：

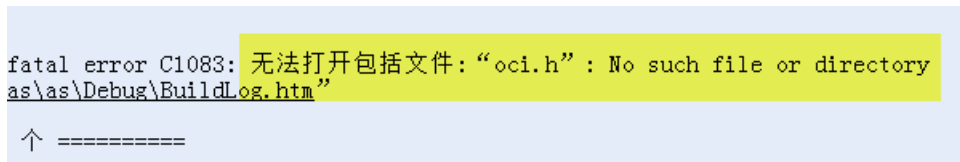


图 3 未引入头文件错误信息

4.1.2 引入库文件

由上面表格可以知道，oracle10g 所需要的库文件为：oci.lib。

在 VS 中依次选择“工具→选项→项目和解决方案→VC++目录”，在右上方组

合框中选择“库文件”，添加库文件目录(如图 4)。例如我的是：
D:\oracle\product\10.2.0\db_1\oci\lib\msvc。

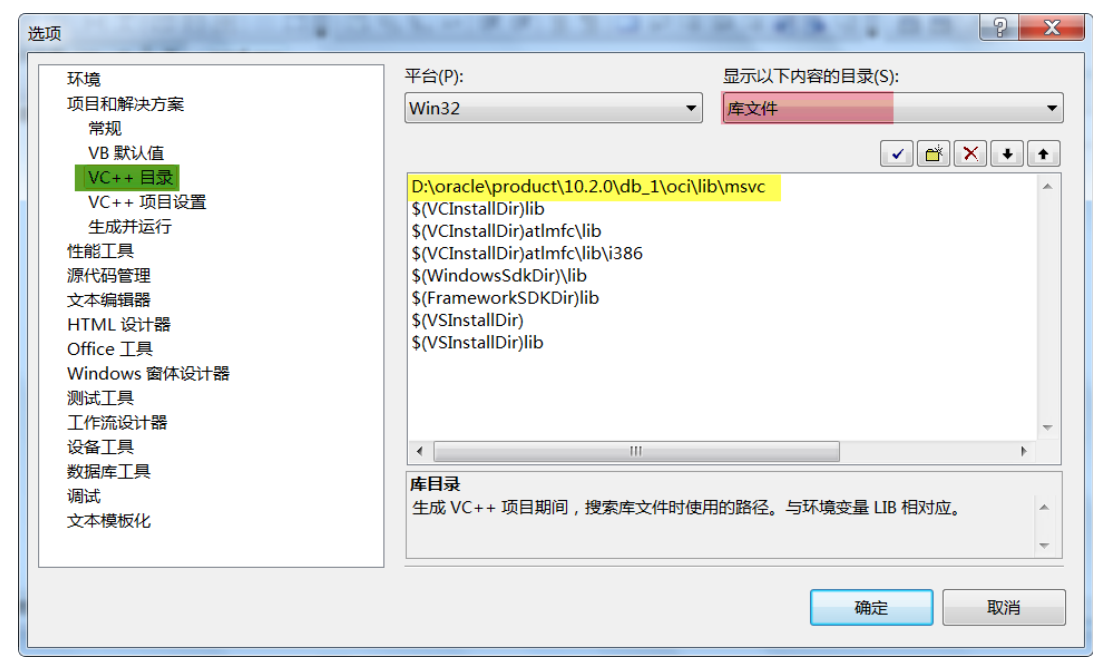


图 4 库文件引入

注意：这时并未配置完成，此时运行程序会出现“无法解析外部符号***”的错误，如图 5 所示

```
1>----- 已启动生成：项目：as, 配置：Debug Win32 -----
1>正在链接...
1>assd.obj : error LNK2019: 无法解析的外部符号 _OCIBindByName, 该符号在函数 "public:
int __thiscall otl_ref_cursor::first(void)" (?first@otl_ref_cursor@@QAEHXZ) 中被引用
1>assd.obj : error LNK2019: 无法解析的外部符号 _OCIAttrGet, 该符号在函数 "public:
unsigned int __thiscall otl_cur::rpc(void)" (?rpc@otl_cur@@QAEIXZ) 中被引用
1>assd.obj : error LNK2019: 无法解析的外部符号 _OCIStmtFetch, 该符号在函数 "public: int
__thiscall otl_cur::fetch(int, int &)" (?fetch@otl_cur@@QAEHHAH@Z) 中被引用
1>assd.obj : error LNK2019: 无法解析的外部符号 _OCIHandleFree, 该符号在函数 "public:
int __thiscall otl_conn::server_detach(void)" (?server_detach@otl_conn@@QAEHXZ) 中被引用
1>assd.obj : error LNK2019: 无法解析的外部符号 _OCIServerDetach, 该符号在函数 "public:
int __thiscall otl_conn::server_detach(void)" (?server_detach@otl_conn@@QAEHXZ) 中被引用
```

图 5 错误信息

4.1.3 添加依赖项

在工程左侧项目上，右键单击选择“项目属性→配置属性”在弹出的属性页对话框中“连接器→输入”，在附加依赖项中填入“oci.lib”，如下图 6 所示。这时完成配置完毕。

若此步不配置，也可以直接在代码中添加

```
#pragma comment(lib, "oci.lib")
```

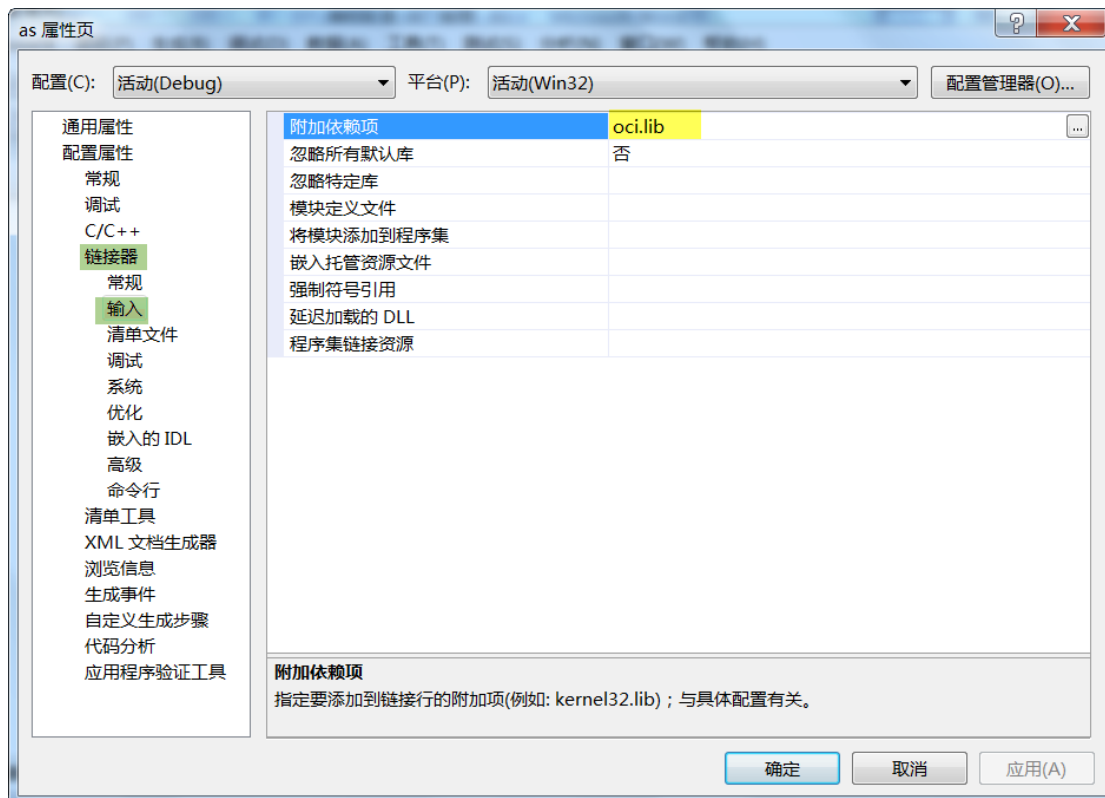


图 6 添加依赖性

4.2 VS2010/2012 中具体配置

在 VS2010/2012 中，设置思路同 VS2008 是一样的，只不过在“工具->选项->项目和解决方案->VC++ 目录”不再提供设置，如下图：

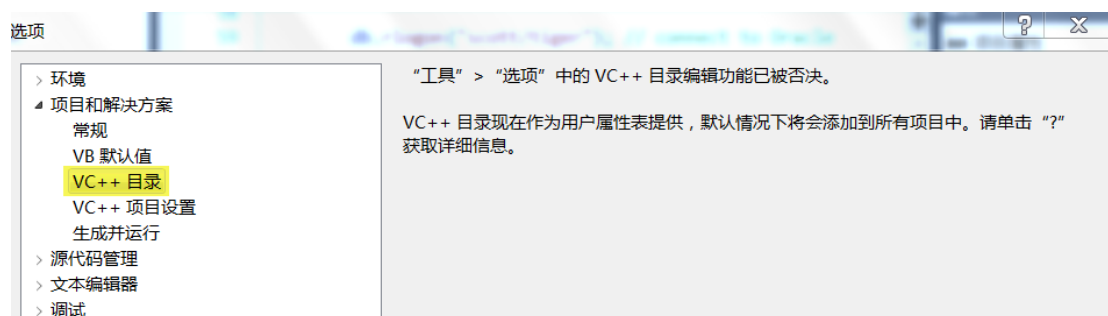


图 7 VS2010/2012 中 VC++目录

打开“属性管理器”视图，找到“[Microsoft.Cpp.Win32.user](#)”，双击打开，找到“[VC++ 目录](#)”

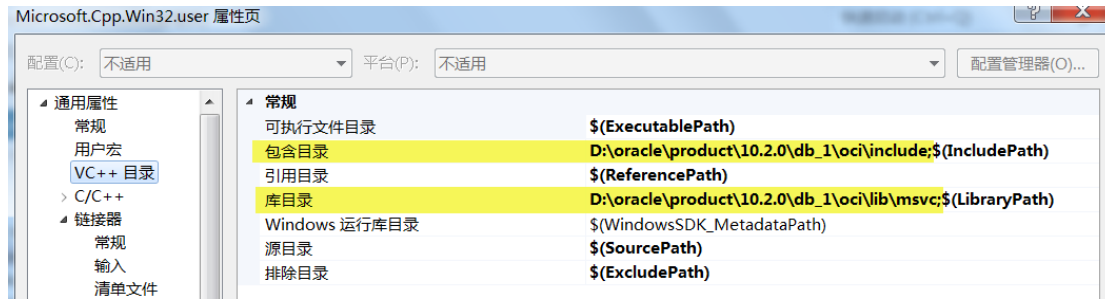


图 8 VS2010/2012 中头文件和库文件的引入

4.3 实例

```
#define OTL_ORA10G_R2          // Compile OTL 4/OCI10G
#pragma comment(lib, "oci.lib")

#include "otlv4.h"           // include the OTL 4 header file
#include <iostream>
using namespace std;

otl_connect db;              // connect object

void insert()                // insert rows into table
{
    otl_stream wr
        (50, // buffer size
         "insert into test_tab values(:f1<int>, :f2<char[7]>)", // SQL statement
         db // connect object
        );

    char tmp[7];

    for (int i=1; i<=20; ++i)
    {
        sprintf(tmp, "Name%d", i);
        wr << i << tmp;
    }
}

void select()
{
    otl_stream rd
        (50, // buffer size
         "select * from test_tab where f1>=:n1<int> and f1<=:n2<int>", // SELECT statement
         db // connect object
        );
}
```

```

    );

    rd << 8 << 16;

    int f1;
    char f2[7];

    while (!rd.eof())    // while not end-of-data
    {
        rd >> f1 >> f2;
        cout << "f1 = " << f1 << ", f2 = " << f2 << endl;
    }
}

int main()
{
    otl_connect::otl_initialize();    // initialize OTL environment

    try
    {
        db.rlogon("aptstest/test@apts206");

        otl_cursor::direct_exec
            (db,
             "drop table test_tab",
             otl_exception::disabled    // disable OTL exceptions
            ); // drop table

        otl_cursor::direct_exec
            (
             db,
             "create table test_tab(f1 int, f2 varchar(6))"
            ); // create table

        insert(); // insert records into table
        select(); // select records from table

    }
    catch (otl_exception& e)    // intercept OTL exceptions
    {
        cerr << "ERROR: code[" << e.code << "]\n";
        cerr << e.msg << endl;    // print out error message
        cerr << e.stm_text << endl; // print out SQL that caused the error
        cerr << e.var_info << endl; // print out the variable that caused the error
    }
}

```



```

    }
    db.logoff();                // disconnect from Oracle

    return 0;
}

```

运行结果：

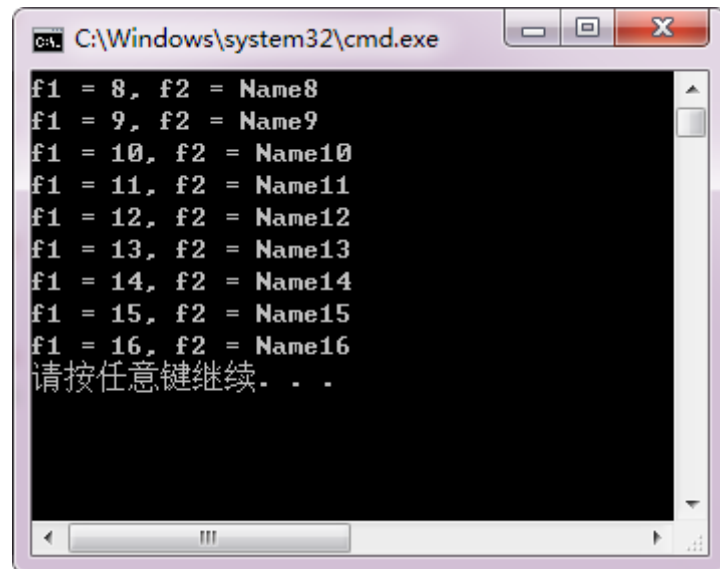


图 9 oracle 运行实例

五、 MySQL 的配置使用

5.1 MyODBC

因为 otl 只能通过 odbc 连接 mysql，所以首先需要安装 myodbc。

myodbc 官网下载地址：<http://dev.mysql.com/downloads/connector/odbc/>



图 10 MyODBC 下载界面

5.2 添加数据源

打开控制面板-》点击管理工具-》点击数据源，然后添加数据源

5.2.1 创建数据源

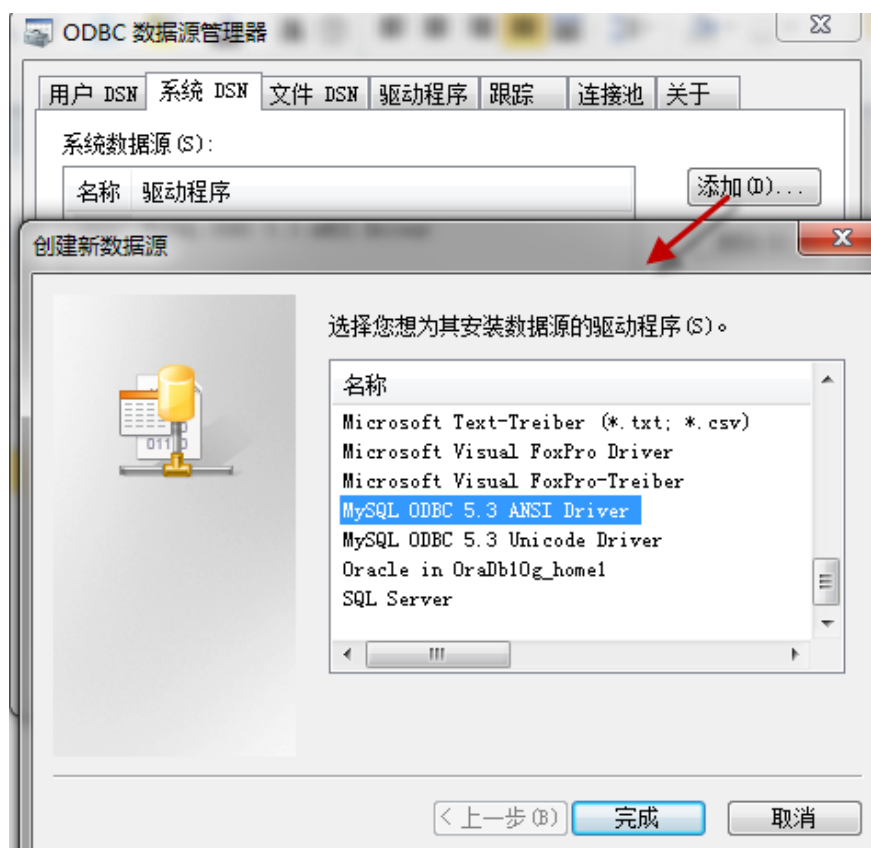


图 11 创建数据源

5.2.2 进行配置

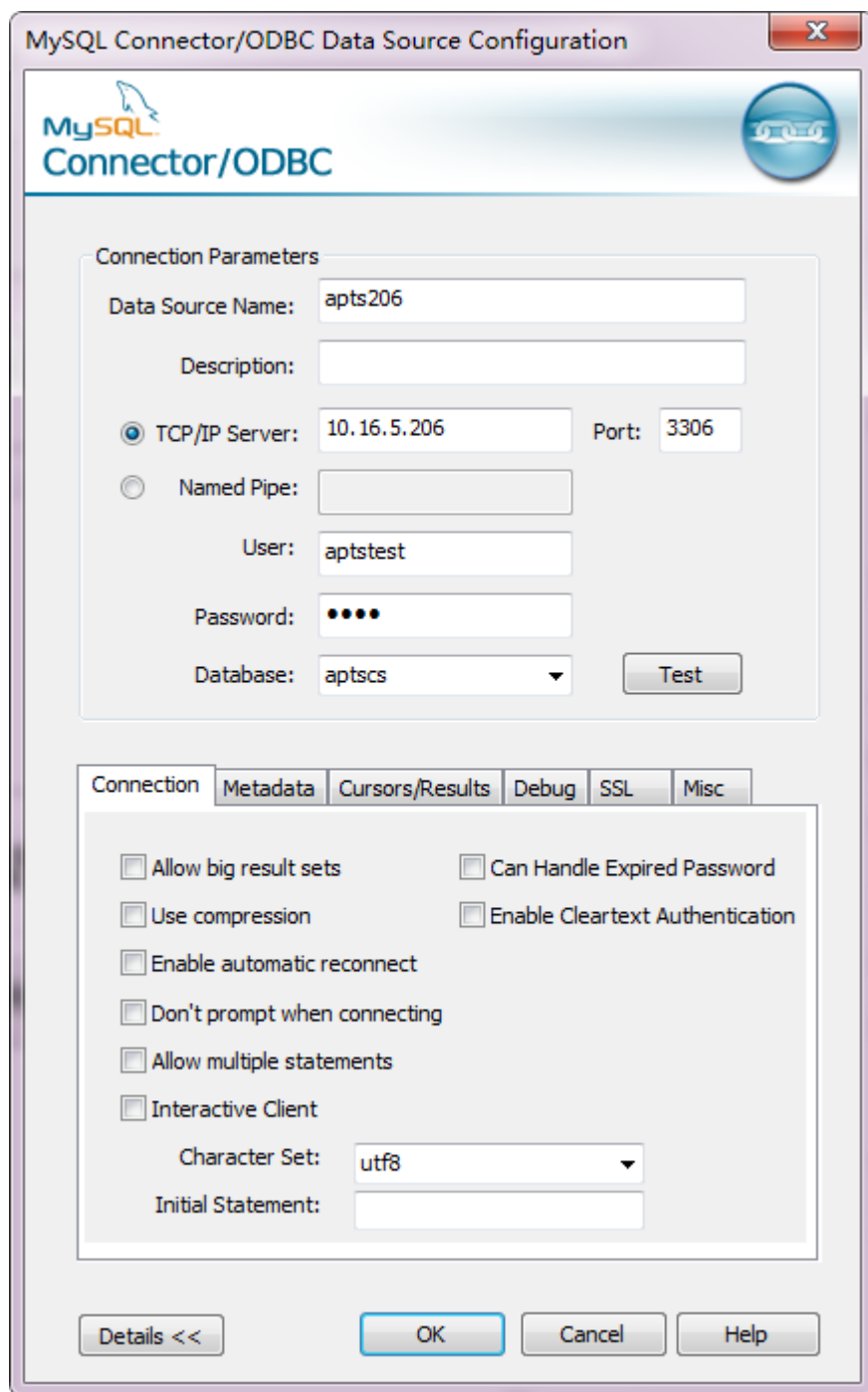


图 12 数据源配置

5.3 64 位系统的配置

在 64 位 Windows 系统中，默认“数据源(ODBC)”是 64 位的，包括“控制面板-》管理工具-》数据源 ”或在“运行”中直接运行“ODBCAD32”程序。

如果客户端是 32 位应用程序，仍然需要配置 32 位 ODBC 数据源，这时需要运行

“C:\Windows\SysWOW64\odbcad32.exe”来启动“ODBC 数据源管理器”，添加 32 位的 ODBC 数据源。

5.4 实例

```
#define OTL_ODBC          // Compile MySQL

#include "otlv4.h"        // include the OTL 4 header file
#include <iostream>
using namespace std;

otl_connect db;          // connect object

void insert()             // insert rows into table
{
    otl_stream wr
        (50, // buffer size
         "insert into test_tab values(:f1<int>, :f2<char[7]>)", // SQL statement
         db // connect object
        );

    char tmp[7];

    for (int i=1; i<=20; ++i)
    {
        sprintf(tmp, "Name%d", i);
        wr << i << tmp;
    }
}

void select()
{
    otl_stream rd
        (50, // buffer size
         "select * from test_tab where f1>=:n1<int> and f1<=:n2<int>", // SELECT statement
         db // connect object
        );

    rd << 8 << 16;

    int f1;
    char f2[7];

    while (!rd.eof()) // while not end-of-data
    {
        rd >> f1 >> f2;
        cout << "f1 = " << f1 << ", f2 = " << f2 << endl;
    }
}
```

```

    }
}

int main()
{
    otl_connect::otl_initialize();      // initialize OTL environment

    try
    {
        db.rlogon("aptstest/test@apts206");

        otl_cursor::direct_exec
            (db,
             "drop table test_tab",
             otl_exception::disabled      // disable OTL exceptions
            ); // drop table

        otl_cursor::direct_exec
            (
             db,
             "create table test_tab(f1 int, f2 varchar(6))"
            ); // create table

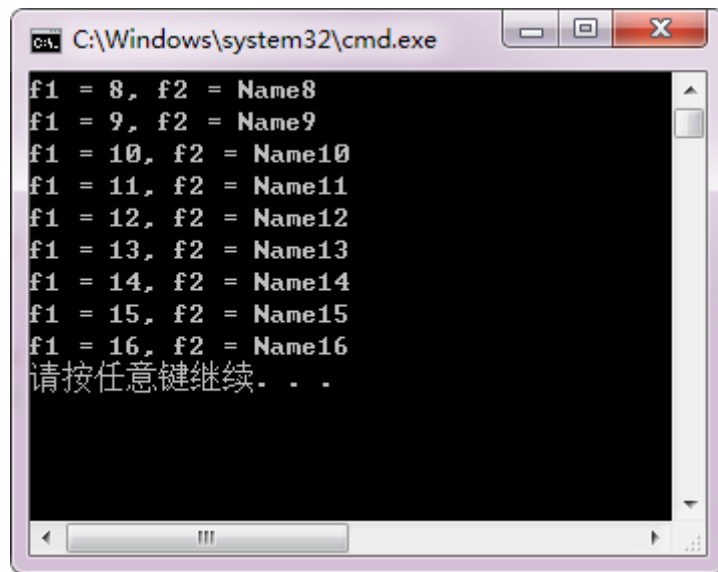
        insert(); // insert records into table
        select(); // select records from table

    }
    catch (otl_exception& e)      // intercept OTL exceptions
    {
        cerr << "ERROR: code[" << e.code << "]\n";
        cerr << e.msg << endl;      // print out error message
        cerr << e.stm_text << endl; // print out SQL that caused the error
        cerr << e.var_info << endl; // print out the variable that caused the error
    }
    db.logoff();                  // disconnect from Oracle

    return 0;
}

```

运行结果：



```
C:\Windows\system32\cmd.exe
f1 = 8, f2 = Name8
f1 = 9, f2 = Name9
f1 = 10, f2 = Name10
f1 = 11, f2 = Name11
f1 = 12, f2 = Name12
f1 = 13, f2 = Name13
f1 = 14, f2 = Name14
f1 = 15, f2 = Name15
f1 = 16, f2 = Name16
请按任意键继续. . .
```

图 13 mysql 运行实例