在跨平台的 C++程序中使用 OTL 来操作不同的数据库

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

本文主要介绍了怎样在 C++程序中使用 OTL 操作数据库。

一、OTL 介绍：

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
- 进行操作 DB2 数据库则是通过 CLI(Call Level Interface)接口来进行
- 至于 MS 的数据库和其它一些数据库，则 OTL 只提供了 ODBC(Open Database Connectivity)来操作的方式。

当然 Oracle 和 DB2 也可以由 OTL 间接使用 ODBC 的方式来进行操纵。

在 MS Windows and Unix 平台下，OTL 目前支持的数据库版本主要有：Oracle 7 (直接使用 OCI7)、Oracle 8 (直接使用 OCI8) 、Oracle 8i (直接使用 OCI8i)、Oracle 9i (直接使用 OCI9i) 、 Oracle 10g (直接使用 OCI10g) 、 DB2 (直接使用 DB2 CLI) 、ODBC 3.x 、ODBC 2.5。OTL 最新版本为 4.0，参见 http://otl.sourceforge.net/， 下载地址 http://otl.sourceforge.net/otlv4_h.zip。

- 优点：
  a. 跨平台
  b. 运行效率高，与 C 语言直接调用 API 相当
  c. 开发效率高，起码比 ADO.net 使用起来更简单，更简洁
  d. 部署容易，不需要 ADO 组件，不需要.net framework 等
- 缺点：
  a. 说明文档以及范例不足够丰富（暂时性的）

其实现在它提供有 377 个使用范例可参考，下载地址：ttp://otl.sourceforge.net/otl4_examples.zip。

二、OTL 的使用：

1. 宏定义

OTL 使用起来也很简单，使用不同的数据库连接，主要是根据需要在程序开始的宏定义来指定的。OTL 是首先根据这个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_ORA7, OTL_ORA8, OTL_ODBC, OTL_DB2_CLI, OTL_ODBC_MYSQL...

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 *MyODBC/MySQL*. 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_ POSTGRESQL	for the PostgreSQL ODBC driver 3.5 (and higher) that are connected to PostgerSQL 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 Interbase 6.x via XTG Systems' ODBC driver. 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 .

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/include	<*ORACLE_HOME*>/oci/lib/<compiler_specific>/ociw32.lib
OCI8	In <*ORACLE_HOME*>/oci/include	<*ORACLE_HOME*>/oci/lib/<compiler_specific>/oci.lib
OCI8i	In <*ORACLE_HOME*>/oci/include	<*ORACLE_HOME*>/oci/lib/<compiler_specific>/oci.lib
OCI9i	In <*ORACLE_HOME*>/oci/include	<*ORACLE_HOME*>/oci/lib/<compiler_specific>/oci.lib
OCI10g	In <*ORACLE_HOME*>/oci/include	<*ORACLE_HOME*>/oci/lib/<compiler_specific>/oci.lib
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

从上面可以看出，如果在 windows 下操纵 MS 的数据库，使用 MS VC++来编译 OTL 程序，就非常简单了，不用另外去找 ODBC32.lib，VC 的编译器中已经默认 link 到工程中了，具体请看如何编译 OTL：http://otl.sourceforge.net/otl3_compile.htm。

```
#define    OTL_ORA10G_R2     // Compile OTL 4/OCI10G
#include "otlv4.h"           // include the OTL 4 header file
#include <iostream>
using namespace std;
#pragma comment(lib, "oci.lib")

otl_connect db;          // connect object

void insert()        // insert rows into table
{
    otl_stream wr
        (50,   // buffer size
        "insert into test_tab values(:f1<float>,:f2<char[31]>)", // SQL
statement
        db     // connect object
        );

    char tmp[32];

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

```
void select()
{
    otl_stream rd
        (50,    // buffer size
        "select * from test_tab where f1>=:f<int> and f1<=:f*2",   //
SELECT statement
        db      // connect object
        );

    float f1;
    char f2[31];

    rd << 8;     // assigning :f = 8

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

    rd << 4; // assigning :f = 4
    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"); // connect to Oracle

        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 number, f2 varchar2(30))"
            );  // 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;
}
```