**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。

1. **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。

1. **OTL的使用:**
2. **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](http://www-4.ibm.com/software/data/db2/) Call Level Interface (CLI) |
| **OTL\_INFORMIX\_CLI** | for Informix Call Level Interface for Unix (when [OTL\_ODBC\_UNIX](http://otl.sourceforge.net/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](http://www.mysql.com/downloads/api-myodbc.html)*/[*MySQL*](http://www.mysql.com/). 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](http://www.borland.com/devsupport/interbase/opensource/) 6.x via [XTG Systems'](http://www.xtgsystems.com/)  [ODBC driver](http://www.xtgsystems.com/ib6odbc.php3). 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 . |

1. **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;

}