ORACLE 与 C 语言通用调用接口的实现①

黄　伟
(武汉数字工程研究所　武汉　430074)

摘　要

数据库调用接口是上层应用与数据库之间的桥梁,通过分析 ORACLE 为 C 语言提供的接口开发工具,提出了一种用 OCI 实现 C 语言与 ORACLE 的通用调用接口的方法。实践证明,这种调用模型具有简洁、方便、灵活的特性。

关键词:数据库调用接口　ORACLE　C 语言　OCI
中图分类号:TP392

Realization of Universal Call Interface between Oracle and C Language

Huang　Wei
(Wuhan Digital Engineering Institute, Wuhan　430074)

Abstract: Database call interface is a bridge between upper application and database. Through analyzing interface developing tools provided by ORACLE for C language, this paper introduces a method, with which a universal call interface between C language and ORACLE can be realized by OCI. Practice proves that this call model is succinct, convenient and flexible.

Key words: database call interface, ORACLE, C language, OCI
Class number: TP392

1　引言

对于任何一个数据库应用系统开发来说,数据库与上层应用系统数据的交互是系统设计和开发的一个重要环节,对于数据的安全性、准确性有较高的要求,因此,为了使开发人员脱离对数据库底层的直接调用,以及方便上层应用对数据库系统数据的交互,需要设计数据库的专门调用接口,从而方便系统开发。

ORACLE 是世界上最流行的大型数据库之一,在电子商务、企业信息管理等领域有着巨大的全球市场。很多数据库应用系统都是基于它开发的。由于上层开发工具层出不穷,ORACLE 也为许多开发语言(如 JAVA、C、COBOL、ADA 等语言)提供了专业的接口开发工具,使调用和操纵数据库的速度大大提高。本文在分析 ORCALE 的数据库调用接口工具 PRO * C/C++和调用接口 OCI 的基础上,着重讲述 ORACLE 与 C 语言通用接口的实现方法。

2　ORACLE 与 C 的接口开发工具

ORACLE 为 C 语言提供了两种数据库调用接口开发方式。

第一种是 PRO * C/C++,它能够使用预编译技术,用自带的编译器将用户编辑好的 PRO * C/C++源程序中嵌入的 SQL 语句转换成标准的运行时刻库(SQLLIB)函数调用,从而生成 C/C++源程序,再经过 C/C++编译器编译、连接后生成可执行程序。PRO * C/C++接口具有以下特点:(1)通过嵌入 SQL 语句实现对数据库调用,提供了四种 ORACLE 的处理方式。(2)使用宿主变量、指示符变量、游标变量等变量类型实现上层与数据库的数据交换。(3)使用预编译异常返回数据库操作过程中的异常。(4)可以通过静态调用或动态调

① 收到本文时间:2004 年 8 月 19 日

用来实现对数据库的操作。(5)程序设计简单,但需要两次编译才能生成执行程序。

另一种是OCI,它是ORACLE提供的一套头文件和库函数组成的数据库应用编程接口工具。实质是一组C函数库,程序员可以直接通过调用这些函数库,通过C/C++编译器可直接生成可执行程序。OCI接口对数据库的调用有如下特点:(1)提供的函数丰富多样,可灵活组织程序结构,但数据结构和参数晦涩难懂。(2)在存取数据时需要对数据进行预先定义和绑定。(3)对数据库的操作是通过把SQL写成字符串形式,并解析字符串来执行。(4)能充分发挥C语言的特点,可以直接像编译一般C语言程序一样编译程序。应用程序对硬件环境要求较低,使开发人员对程序设计和运行控制更加灵活。

用这两种接口工具开发各有所长,PRO*C/C++工具通过嵌入SQL,应用层和数据库的数据交换简单方便,容易理解。但是要通过预编译生成C源程序,才能最终编译成执行程序。OCI提供的函数丰富,但复杂,需要进一步封装才能供上层调用,直接调用OCI开发比较困难。但是,OCI对数据库的调用速度要明显优于PRO*C对数据库的调用。PRO*C适合静态的与数据库进行交互,由于采用嵌入SQL语句,缺少对数据以及变量描述性的交互行为,因此不利于把对数据库的数据交互写成通用的调用接口,它适合对特定数据库操作的特定行为进行函数调用。而用OCI开发应用程序,是通过ORACLE库调用来实现。一个含OCI调用的C程序就是一个用纯C语言的语句编写的程序。这样的程序不仅具有SQL非过程性能的优点,也具有第三代程序设计语言的过程性能,同时还具有PL/SQL的优点,使开发的应用具有更强的数据处理能力和更大的灵活性。鉴于以上分析,用OCI来编写ORACLE与C语言的通用调用接口是比较合适的选择。

3 用OCI调用ORACLE的过程

我们用一个常用的数据库调用过程来说明OCI调用在程序设计中的结构。例如我们通过一个条件查询来从数据库中提取一些数据。这是一个典型调用过程,涉及到输入条件数据、执行命令和取出结果数据的过程。对于OCI调用来说,它在整个程序中的结构如图1所示。

对于一个数据操作如查询来说,则可能没有第(7)、(8)、(9)个步骤,如果没有条件交互输入,则可能没有第(4)步。从图1可以看出,OCI调用数据库的步骤尤其的烦琐,如果让开发者去直接使用OCI来跟数据库进行交互,其难度很大。原因有4个:

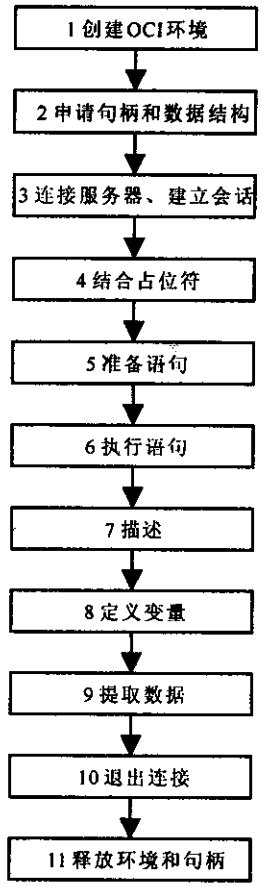

图1 OCI调节ORACLE的基本流程和结构

①OCI提供的函数众多,同一个功能可以用几个类似的函数共同实现,但必须搭配得当。

②每一个函数的参数多,晦涩难懂。

③为使ORACLE与外界数据类型的一致,OCI提供的数据类型宏定义很多,容易混淆,不易记忆。

④调用过程繁杂,不易使用。

OCI强大的功能决定了OCI提供的函数是数据库调用的底层函数,灵活多样的底层调用从而使上层应用更灵活,但同时这种灵活是以开发的难度和复杂度为代价的,因此,要想更好的开发数据库应用,就必须对OCI作进一步的封装,形成一个更简洁的通用的数据库调用接口为上层应用服务。

4 通用数据库调用接口的实现

一个通用的数据调用接口应该具备以下特点:

①调用的过程简洁、方便。

②调用的函数少、功能强、通用。

③函数参数少,参数的含义明确。

④为上层提供简单的接口,并屏蔽下层复杂的交互。

⑤有必要的出错处理,能够报告错误信息。

在目前,微软提供的 ADO 数据库引擎就是一个很好的例子,它提供的接口对各种开发环境(如 VC、VB、DELPHI、ASP 等)都是一致的,但能够对各种数据库(如 SQL SERVER、ORACLE 等)进行相同的数据操作,并且具有简单、安全、灵活的特性。

为了达到这个目标,深入分析 OCI 的特点可以看出:①OCI 在开始调用时都需要创建 OCI 环境和对数据申请句柄,OCI 句柄中,除环境句柄、定义句柄和结合句柄外,其他句柄可以用同一个函数分配。定义句柄和结合句柄是自动分配。结合句柄要在数据提取时分配;②OCI 在执行带条件输入查询语句前必须进行数据绑定操作,对 SQL 语句在执行前必须进行分析准备;③OCI 在提取数据前必须定义,然后进行分析准备和执行;④数据库退出时要释放句柄。

通过分析 OCI 的特点可以看出,如果对 OCI 函数做进一步的概括,OCI 对 ORACLE 数据库的调用和操作可以简化成如图 2 所示的过程模型。为实现这个模型,我们把众多 OCI 函数根据图 2 操作过程封装成了以下几个主要函数接口:

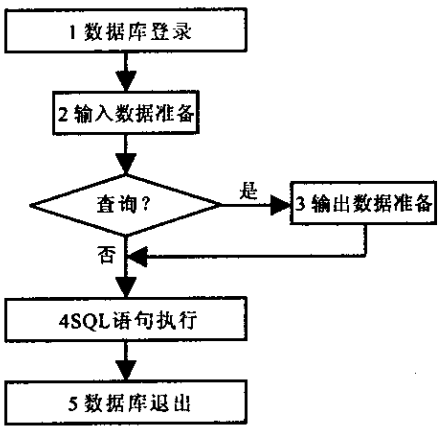

图 2　通用调用接口调用 ORACLE 的流程和结构

(1)登录函数

int Log _ On(cha * user,char * pwd,cha * dbname);负责数据库登陆,建立环境,分配环境句柄。

(2)输入数据准备函数

int Prepare _ In(int pos,void * variable,int len,int type);负责输入数据绑定。

(3)输出数据准备函数

int Prepare _ Out(int pos,void * variable,int len,int type);负责输出数据的定义。

(4)执行 SQL 语句函数

int Exec(char * sql, int in _ count);负责无输出数据的 SQL 语句的分析和执行。

(5)准备查询函数

int Prepare _ Fetch(char * sql);负责有输出数据的 SQL 语句的分析,以及输出数据与输出变量的结合。

(6)查询函数

int Fetch(int out _ count);负责有输出数据的 SQL 语句的执行。

(7)退出登录函数

int Log _ Off();断开数据库连接,清理句柄,释放空间。

这几个函数联合使用可以完成 SQL 数据定义、数据操纵、查询、事务控制等绝大部分的有关 ORACLE 的数据库调用,它屏蔽掉了 OCI 复杂的调用过程和交互,向上层提供了极其简单的操作接口,使 C 语言调用数据库的操作变得异常的简单。

5　实例

为更好说明这些函数的应用,现举例说明从一个工资表 emp 中取出工资大于 800 元的人的姓名和工资数的函数调用方法。

```
#include "oci _ type.h"
void main()
{
   char sql[200]= "select name,money
   from emp where money>:1";
   char name[30];
   float money=800;
   if(Log _ On("scott","tiger",NULL)<0)
   { /* 登录数据库 */
     printf("FAILED:Long _ On()\n");
     return ;
    }
     Prepare _ In(1,&money,sizeof(money),SQLT _
FLT);/* 准备输入 */
     Prepare _ Out(1,name,30,SQLT _ STR);
     Prepare _ Out(2,&money,sizeof(money),SQLT
_FLT); /* 准备输出 */
     Prepare _ Fetch(sql);/* 准备查询 */
     while(Fetch(2)>0) /* 查询 */
{
```

```
    printf("name=%s,money=%f\n",name,money);
/*输出数据*/
  }
  Log_Off(); /*注销连接*/
}
```

从上面的程序看，通用接口函数调用方法异常简单，开发人员完全可以用图 2 中的模型来进行数据库的操作，这使开发人员对数据库的控制变得轻松自如。

6 结论

OCI 是一个灵活的数据库调用接口工具，使用 OCI 对 ORACLE 调用的效率比其他方法调用 ORACLE 要高。由于 OCI 函数库太复杂，根据需要可以对 OCI 做进一步的封装，封装程度可大可小，封装库的偶合度可高可低。本文从工程实践中总结经验，提出了一种用 OCI 封装的对 ORACLE 调用的通用数据库调用接口模型。通过工程应用的实践证明这种模型具有极大的简洁性，同时控制方便灵活，有很高的适用性。

参 考 文 献

[1]Programmer's Guide to the ORACLE Call Interfaces, Part NO. 5411－70

[2]ORACLE sever SQL Language Reference Manual, Part NO. 778－70

[3]袁朋飞编著. ORACLE 数据库高级应用开发技术[M]. 北京：人民邮电出版社，2000

(上接第 107 页)

普通使用者身份，转入相应的系统功能页面

```
            break;
        defau lt:
            ……  // 确认该用户不存在，转入系统注册页面
            break;
        }
    }
}
```

前端控制器的命令控制：

```
public class Transaction
{     ……
      public static int ConfirmUser(string username, string userpwd)
      {
        int result=－1;
          DataSet ds1 = ConfirmMode.GetUserInfo(username);//调用模型的方法
          DataSet ds2 = ConfirmMode.GetUserInfo(username,userpwd);
        if (ds1.Tables[0].Rows.Count==0)
          result=－1;
else if (ds2.Tables[0].Rows.Count==0)
        result=0;
else
{
          if (Convert.ToInt32(ds.Tables[0].Rows[0]["usergrade"])==1)
            result=1;
          else if (Convert.ToInt32(ds.Tables[0].Rows[0]["usergrade"])==2)
            result=2;
          else if (Convert.ToInt32(ds.Tables[0].Rows[0]["usergrade"])==3)
            result=3;
    }
    return result;
  }
  ……
}
```

5 结束语

MVC 是软件应用框架构建中日益成熟并完善的主流模式，这里提出基于前端控制器的 MVC 模式优化策略和实现方法，在复杂软件系统的构建中必将具有广泛的实用价值。

参 考 文 献

[1] David West. Object Thinking[M]. Washington: Microsoft Press, 2004

[2] Andrew Filev, Tony Loton. VS. NET UML 建摸高级编程[M]. 北京：清华大学出版社，2003

[3] Tbai T, Lam H. Q, 王敏之译，. NET 框架精髓[M]. 北京：中国电力出版社，2001

[4] 王晓楠. MVC 的设计和实现[J]. 计算机系统应用，2004，30(3)，41～45