WWW环境下Oracle数据库访问技术

孙宝文 汪锦岭 (中央财经大学信息系 100081)

摘要：本文对Oracle对象服务器中的各个对象作了简要介绍，并结合实例说明了在WWW环境下使用Oracle对象服务器访问Oracle数据库的方法。

关键词：ASP OO4O Oracle对象服务器

1 概论

随着Internet技术的广泛使用，基于WWW环境的新一代数据库信息系统迅速发展，传统的C/S结构正逐步被B/S结构所取代。在WWW环境下开发应用系统的方法，比较普遍的是使用Microsoft的ASP技术，它的程序编制方便，运行效率高，安全保密性也较好。而Oracle数据库作为数据库技术的领先者，自Oracle 8起开始全面支持Internet/Intranet，在WWW环境下得到了广泛使用。要在WWW环境下充分发挥Oracle提供的各种特有功能，如Stored Function、Stored Procedure、Package、Multiple Cursor，以及Oracle的一些特殊数据类型如Long型等，使用ASP内置的ADO组件是无法做到的。为此，Oracle公司提供了一套独立的数据库接口——Oracle Objects for OLE（OO4O），其中的Oracle对象服务器（OOS）可供所有与OLE兼容的程序语言存取Oracle数据库，并能实现Oracle提供的各项特有功能。本文将介绍在WWW环境下使用Oracle对象服务器对Oracle数据库进行访问的方法。

图 1 一个医疗挂号中心门诊管理系统实例

OO4O是Oracle公司为了客户端存取数据库所发展的一个重要产品。要在WWW环境下使用Oracle对象服务器，必须将OO4O安装在Web Server所在的机器上，使它作为Oracle数据库的客户端，以形成一个Browser/Web Server/Database Server三层的网络结构。我们使用Oracle对象服务器技术，为一家医疗挂号中心开发了一个基于B/S结构的门诊管理系统，其网络结构简图如图1所示。

2 Oracle对象服务器中的对象简介

Oracle对象服务器共提供了OraClient、OraSession、OraConnection、OraDatabase、OraDynaset、OraSQLstmt、OraField、OraParameter及OraParameterArray等九个对象供程序人员使用。这九个对象分别有着不同的功能，它们之间的关系如图2所示。

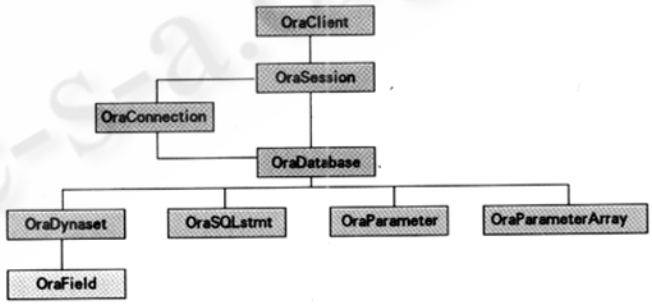

图 2 Oracle 对象服务器对象关系图

在图2中，每对上层对象和下层对象之间都是“一对多”的层次关系，一个上层对象可以对应多个下层对象。

2.1 OraClient对象

OraClient用来定义Oracle数据库的客户端（即Web服务器）的范围，它记录此客户端所有的OraSession对象。换言之，每个客户端只会存在一个OraClient对象。该对象不需要程序员来建立，系统会根据需要而自动建立。

2.2 OraConnection对象

OraConnection 对象表示对 OraDatabase 对象的连接，它也由系统根据需要自行建立。当用户要建立一个 OraDatabase 对象时，系统就会自动产生一个 OraConnection 对象。

2.3 OraSession 对象

OraSession 对象用于管理 OraDatabase 与 OraConnection 等对象，它是应用程序的最上层。应用程序必须在建立 OraSession 对象后，才能使用其他相关的对象。此对象通过 ASP 中的 CreateObject 函数来建立，其建立方法如下：

Set OraSession = CreateObject ("OracleInProServer.XoraSession")

其中 OracleInProServer.XoraSession 为 OO4O 在系统注册表中所注册的名称。

2.4 OraDatabase 对象

OraDatabase 对象表示对数据库服务器一个虚拟的登录（virtual login）。一般来说，当 OraDatabase 对象被建立时，通常需要提供用户帐号、密码和数据库别名等信息。其建立方法如下：

Set OraDatabase = OraSession.DbOpenDatabase (" 数据库别名 "," 用户帐号 / 密码 ",0)

2.5 OraDynaset 对象

OraDynaset 对象允许用户浏览或更新由 SQL SELECT 所返回的数据，它必须属于唯一的 OraDatabase 对象。该对象可由 OraDatabase 的 DbCreateDynaset 方法程序建立，格式如下：

Set OraDynaset = OraDatabase.DbCreateDynaset ("sql 语句 ",0)

2.6 OraSQLstmt 对象

OraSQLstmt 对象通常是用来运用 SQL 命令，或是调用 Stored Function、Stored Procedure。该对象可由 OraDatabase 的 CreateSQL 方法来建立，格式如下：

Set OraSQLstmt = OraDatabase.CreateSQL("sql语句",0)

2.7 OraField 对象

OraField 对象表示在 OraDynaset 对象中某一行（row）中的一个字段（column）或数据项目（data item），它间接地从 OraDynaset 的 OraFields 数据集合中取得其中相应字段的数据。

2.8 OraParameter 对象

OraParameter 对象是表示一个在 SQL 命令或 PL/SQL 程序块中所附加的变量。

2.9 OraParameterArray 对象

OraParameterArray 对象与 OraParameter 对象相似，不同之处在于它表示在 SQL 命令或 PL/SQL 程序块中所附加的数组（Array）类型的变量。

3 各种数据库操作的实现方法

Oracle 对象服务器的九个对象，每个对象都包含了各种丰富的属性和方法程序，为程序开发人员提供了一个功能强大的数据库开发接口。下面就以最常使用的几种数据库操作为例，介绍一下 Oracle 对象服务器在 WWW 环境下的使用方法。这里所有的 ASP 程序示例都是以 Oracle 8 Enterprise Edition V8.0.4 for Windows NT 为数据库系统，以 Microsoft IIS 4.0 为 Web 服务器实现的。

3.1 连接数据库

要访问一个 Oracle 数据库，第一步要做的就是建立与数据库的连接。假设有一个 Oracle 数据库，其别名为 ghzx，数据库中有一个 scott 用户，其密码为 tiger，则连接数据库的 ASP 程序如下：

```
<%  Set OraSession=CreateObject("OracleInProServer.XOraSession")
Set OraDatabase=OraSession.DbOpenDatabase("ghzx","scott/tiger",0) %>
```

3.2 查询数据

查询是 Web 应用程序中最重要的数据库操作。在 Oracle 对象服务器中，查询操作是通过 OraDatabase 对象的 DbCreatDynaset 方法程序实现的。我们将所要查询的 SQL 语句作为该方法程序的参数，并将执行结果保存到一个 OraDynaset 对象。OraDynaset 对象类似于一个数据表（table），它包含了该 SQL 语句所对应的查询结果。OraDynaset 对象的 RecordCount 属性表示查询结果中的记录条数，其 EOF 属性表示目前是否到达查询结果记录集合的尾部。OraDynaset 对象的 Fields 数据集合存放了查询结果中当前记录的情况，OraDynaset.Fields(i).value 表示当前记录第 i+1 个字段的值（i 从 0 开始编号），OraDynaset.Fields(i).name 表示当前记录第 i+1 个字段的名称。如果已知字段名称，我们也可以通过字段名称而不是字段的顺序号来访问各字段的数据。例如已知第 1 个的字段名称是 Column_1，则 OraDynaset.Fields(0).value 等价于 OraDynaset.Fields("Column_1").value。此外，OraDynaset 对象还提供了 DbMoveFirst、DbMoveLast、DbMovePrevious、DbMoveNext 等方法程序，分别表示将查询结果中的记录指针移到第一条、最后一条、以及当前记录的上一条、下一条，使程序开发人员可以很方便地对数据查询结果进行各种灵活处理。

假设在 ghzx 数据库中存在一个名为 department 的数据表，记录了该挂号中心内部各个部门的基本情况，其中

有部门编号（depno）、部门名称（depname）、员工人数（empcount）三个字段。现要求查出该表的所有记录，并将结果以表格形式显示于Web页面中。相应的ASP程序如下：

```
<%   sql="select * from department"
  Set OraDynaset=OraDatabase.DbCreateDynaset(sql,0)
  '列出表格的栏目名称
  response.write "<TABLE><TR>"
  for i=0 to OraDynaset.Fields.count-1
   response.write "<TD>" & OraDynaset.Fields(i).name
& "</TD>"
  next
  response.write "</TR>"
  '列出表格的具体数据
  Do Until OraDynaset.EOF
    response.write "<TR>"
    for i=0 to OraDynaset.Fields.count-1
       response.write "<TD>"&OraDynaset.Fields(i).
value&"</TD>"
    next
    response.write "</TR>"
    OraDynaset.DbMoveNext '记录指针移向下一条
  Loop
  Response.write "</TABLE>"   %>
```

3.3 增加记录

利用Oracle对象服务器向Oracle数据表中增加记录的方法有两种，第一种是利用SQL语言的Insert语句，第二种是利用OraDynaset对象的DbAddNew方法程序。下面分别加以介绍。

第一种方法是将要增加的记录写成SQL语言的一个Insert语句，然后利用OraDatabase对象的DbExecuteSQL方法程序来运行这个SQL语句。假设要向department表中增加一条部门编号为“002”、部门名称为“技术部”、职工人数为10的记录，相应的ASP程序如下：

```
<%   sql="insert into department values('002','技术部
',10)"
  OraDatabase.DbExecuteSQL(sql)   %>
```

此方法同样适用于对记录的修改和删除操作，只需要将相应的SQL语句改为Update或Delete语句即可，以下不再赘述。

第二种方法首先生成一个与该数据表相对应的OraDynaset对象，然后用该对象的DbAddNew方法程序向数据表中增加数据。DbAddNew方法程序用于向OraDynaset对象的查询结果缓冲区中增加一条新的空白记录，DbUpdate方法程序则用于将对OraDynaset查询结果缓冲区所作的改动回写至原数据表。假设仍要向department表中增加一条上述记录，其对应的ASP程序如下：

```
<%   sql="select * from department where 0=1"
  Set OraDynaset=OraDatabase.DbCreateDynaset(sql,0)
  OraDynaset.DbAddNew
  OraDynaset.Fields("depno").value="002"
  OraDynaset.Fields("depname").value="技术部"
  OraDynaset.Fields("empcount").value=10
  OraDynaset.DbUpdate   %>
```

以上程序第1行select语句中的where子句实际上可以换成其他任何查询条件，因为我们只需使OraDynaset对象与要增加记录的数据表对应起来即可。这里使用"where 0=1"是为了使所产生的OraDynaset对象不含任何查询结果，以节约Oracle客户端的Cache空间。

3.4 修改记录

这里介绍利用OraDynaset对象的DbEdit方法程序来修改记录的方法。DbEdit方法程序使OraDynaset对象进入修改模式，此后程序人员就可以对OraDynaset对象中当前记录各字段的值进行修改。修改完毕后，再调用DbUpdate方法程序将修改结果回写到原数据表中去。假设要将编号为“002”部门的员工人数改为15，相应的ASP程序如下：

```
<%   sql="select * from department where depno='002'"
  Set OraDynaset=OraDatabase.DbCreateDynaset(sql,0)
  If OraDynaset.RecordCount<>0 then '如果存在这一
记录
    OraDynaset.DbEdit
    OraDynaset.Fields("empcount").value=15
    OraDynaset.DbUpdate
End if   %>
```

3.5 删除记录

这里介绍利用OraDynaset对象的DbDelete方法程序来删除记录的方法。DbDelete方法程序用于将OraDynaset对象的查询结果集合中的当前记录删除，并直接删除原数据表中的相应记录。假设要将department表中depno为“002”的记录删除，则相应的ASP程序如下：

```
<%   sql="select * from department where depno='002'"
  Set OraDynaset=OraDatabase.DbCreateDynaset(sql,0)
  If OraDynaset.RecordCount<>0 then '如果存在这一
记录
     OraDynaset.DbDelete
End if   %>
```

3.6 运行 Stored Function 和 Stored Procedure

Stored Function 和 Stored Procedure 是 Oracle 数据库的一个重要特色。所谓 Stored Function 是指一个存储在 Oracle 数据库中的一段 PL/SQL 程序，它可以输入参数并返回一个计算值。而 Stored Procedure 和 Stored Function 差不多，其最大区别在于前者没有返回值。下面我们就以 Stored Function 为例，介绍一下它们在 Oracle 对象服务器中的调用方法。

要调用一个 Stored Function，首先要用 OraDatabase 对象中Parameters数据集合的Add方法程序来设置好所有的参数（包括入口参数和返回值），然后再用OraDatabase 的 DbExecuteSQL 方法程序执行相应的 PL/SQL 语句。OraDatabase.Parameters.Add 方法程序的格式如下：

OraDatabase.Parameters.Add 参数名称，参数值，输入/输出类型

其中“输入/输出类型”为 1 表示入口参数，为 2 表示出口参数（返回值），为 3 表示该参数既可作为入口参数，又可作为出口参数。

使用该方法程序加入一个参数以后，还必须设置该参数的ServerType属性，以确定该参数的数据类型。几种常用的数据类型与相应的 ServerType 属性值的对应关系如表 1 所示。

假设数据库中有一个名为 Dep_count 的 Stored Function，入口参数为一个整型数据 x，返回值为 department 表中员工人数字段值大于 x 部门的个数。令 x=5，要求将函数执行的结果显示于 Web 页中。相应的 ASP 程序如下：

```
<%  OraDatabase.Parameters.Add "x",5,1  '入口参数
OraDatabase.Parameters("x").ServerType=3  'INTEGER 型
 OraDatabase.Parameters.Add "y",0,2  '返回值
OraDatabase.Parameters("y").ServerType=3  'INTEGER 型
 '调用 Stored Function
 OraDatabase.DbExecuteSQL("Begin: y:=Dep_count(:x); end;")
 Response.write "人数>5 的部门个数为" & OraDatabase.Parameters("y").value
 '删除参数
 OraDatabase.Parameters.Remove "x"
 OraDatabase.Parameters.Remove "y"  %>
```

表 1 数据类型与 ServerType 值对应关系表

数据类型	ServerType 属性值
VARCHAR2	1
NUMBER	2
SIGNED INTEGER	3
FLOAT	4
DATE	12

4 结束语

Oracle对象服务器的九个对象提供了多种丰富的属性和方法程序，使程序开发人员可以很方便地对Oracle数据库进行各种类型的访问。经测试表明，在同等情况下，其运行效率远远优于 Microsoft 的 ADO 组件。当然，Oracle 对象服务器还存在着一定的不足之处，比较明显的是 Oracle 自第 8 版起，开始由传统的关系型数据库向对象－关系型数据库转变，支持表级和字段级的对象功能，而 Oracle 8所附带的OO4O 2.2却并没有支持这一新特色，给程序开发带来了不便。相信随着新版本的推出，Oracle 公司必将能解决这一问题。■

参考文献

1 林金霖，ASP 实务经典，中国铁道出版社，1999 年 12 月。

2 [美] 佩吉，Oracle8/8i 开发使用手册，机械工业出版社，2000 年