OLE DB /ADO 的概念及对象模型

崔炜　张璐　程治刚　(武汉水利电力大学研 9703 班　430072)

摘要:本文从应用需求的角度分析了 UDA 技术出现的背景,详细介绍了 OLE DB/ADO 的概述及对象模型,并在最后给出了一个 ADO 编程实例。

关键词:数据访问　ODBC　UDA　OLE DB　ADO

1. UDA 出现的背景

传统上数据库应用程序只能使用一具体针对某数据库的 API,往往被编写成存取一指定的数据库;而在实际应用中,所需数据又往往分布在不同的数据库中,这就给应用程序的开发及系统集成带来了很大的局限性。

为此,Microsoft 基于 X/Open 和 SQL Access Group (SAG) Call Level Interface (CLI) 推出了 ODBC (Open Database Connectivity,开放数据库互连)技术。ODBC 的结构框架如图 1 所示。一个 ODBC 应用程序在逻辑上包含五个层次:应用层、ODBC 接口层、驱动管理层、驱动层和数据源层。ODBC 使用分层结构,通过提供一标准的基于 SQL 的 API,以及将具体针对数据库的代码放入 ODBC 驱动器中,从而在应用层提供了一致的数据库访问接口。

但是随着近几年网络技术和数据库技术的飞速发展,现在的应用系统对数据集成的要求也越来越高。这些数据有可能分布在不同的地方,并且使用不同的格式,不仅包括传统关系数据库中的数据,还包括操作系统中的文件、顺序索引文件、桌面数据库、电子邮件、目录服务、多媒体数据、空间数据等等。因此,对于这些非基于 SQL 的数据源,ODBC 已不能提供一个一致的数据访问接口。

针对这一问题,一种解决方案是通过扩展数据库引擎和编程接口,使数据支持新的数据类型,包括文本、空间数据、视频数据和音频数据等,从而可以把所有这些数据都转移到数据库系统中,然后按照操作数据库的方法对这些数据进行访问。这种方案虽然能够按统一的方式对数据进行各种操作,但这种间接访问方式同时也带来数据更新不及时、空间资源冗余和访问效率低等许多问题。另外,将所有数据转移到数据库系统中就将是一个极其费时且费用昂贵的过程。

而 Microsoft 推出的 UDA(Universal Data Access,一致数据访问技术)则较好地解决了这些问题。UDA 的层次结构如图 2 所示。

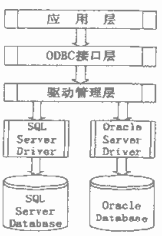

图 1　ODBC 的结构框架图

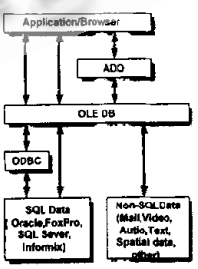

图 2　UDA 的层次结构图

UDA 具有以下优点:

(1)它使得应用通过一致的接口来访问各种各样的数据,而不管数据驻留在何处,也不需要进行数据转移或复制、转换,在实现分布式的同时也带来了高效率;

(2)UDA 技术在统一数据访问接口的同时,它的多

层结构使数据使用方有了更多的选择机会，而它强大的扩展能力也给数据提供方留下了更多的扩展余地；

(3)UDA 结构中的组件数量较少，从而降低了出错的可能性，提高了系统的稳定性；

(4)UDA 从一推出就得到了广泛的工业支持，包括开发工具销售商、数据存取组件开发商、DBMS 销售商。到目前为止，可获得大多数流行的商业关系数据库的 OLE DB 服务程序(provider)，如 Oracle、Microsoft SQL Server、IBM DB2、Sybase、Informix 等，另外还有 VSAM、AS-400 File、IMS/DB 的 OLE DB 服务程序。这一点，可以说是应用程序开发人员最关心的问题。

因此，可以说 UDA 技术是继 ODBC 之后的数据访问技术的一次飞跃。

从图 2 中可以看出，UDA 包括两层软件接口：OLE DB 和 ADO(ActiveX Data Object)，分别对应于不同层次的应用开发。下面将分别介绍 OLE DB/ADO 的概念及对象模型。

2. OLE DB 的概念及对象模型

OLE DB 是 UDA 的核心，在系统级建立了数据访问的一组标准 COM 接口。OLE DB 标准的具体实现是一组符合 COM 标准、基于对象的 C++ API。这组接口封装了各种数据系统的访问操作，为数据使用方和数据提供方建立了标准。OLE DB 还提供了一组标准的服务组件，用于提供查询、缓存、数据更新、事务处理等操作。因此，数据提供方只需实现一些简单的数据操作，在使用方就可以获得全部的数据控制能力。OLE DB 的对象模型如图 3 所示。

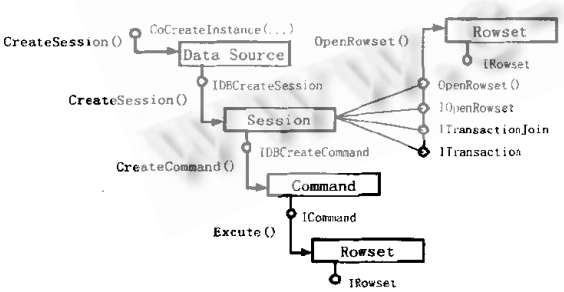

图 3　OLE DB 的对象模型

从图 3 中可以看出，OLE DB 的对象模型非常简单，主要包括四个 COM 对象：

①数据源(Data Source)对象—它对应于一个数据提供者，负责管理用户权限、建立与数据源的连接等初始操作；

②会话(Session)对象—在数据源连接的基础上建立会话对象，会话对象提供了事务控制机制；

③命令(Command)对象—数据使用者使用命令对象执行各种数据操作，如查询命令、修改命令等；

④行集(Rowset)对象—提供了数据的抽象表示，它可以是命令执行的结果，也可以直接由会话对象产生，它是应用程序主要的操作对象。

3. OLE DB 与 ODBC 的关系

虽然 OLE DB 和 ODBC 标准都是为了提供一致的数据访问接口，但 ODBC 标准的对象是基于 SQL 的数据源，而 OLE DB 标准的对象则是范围更为广泛的任何数据存储。因此，符合 ODBC 标准的数据源实际上是符合 OLE DB 标准的数据存储的一个子集。符合 ODBC 标准的数据源要符合 OLE DB 标准，还必须提供相应的 OLE DB 服务程序(OLE DB Provider)。如数据源提供相应的 OLE DB Provider，则应用程序可使用 ADO 直接调用 OLE DB Provider)。如数据源没有相应的 OLE DB Provider，Microsoft 已经为所有的 ODBC 数据源提供了一个统一的 OLE DB Provider，即 OLE DB Provider for ODBC，应用程序可使用 ADO 调用 OLE DB Provider for ODBC，再由 OLE DB Provider for ODBC 调用相应的 ODBC 驱动程序。

4. ADO 的概念及对象模型

ADO(ActiveX Data Object)是一组基于 OLE DB 数据的高级自动化(Automation)应用层接口。尽管 OLE DB 是一个功能强大的数据访问接口，但是对于大多数应用程序开发人员来说，他们通常使用不支持函数指针和其他 C++ 调用机制的高级编程语言，因此他们感兴趣的并不是 OLE DB 提供的底层数据访问控制功能，如内存管理、手工集合组件等。另外，由于 OLE DB API 是 C++ API，只能提供 C++ 语言调用接口，不能直接用于其他高级编程语言。所以，UDA 在 OLE DB 之上又提供 ADO 对象模型。ADO 具有以下特性：

(1)易于使用。利用 ADO 对象模型完成一个简单的数据访问任务，只需编写很少的代码。

(2)编程语言无关性。ADO 可以用于大多数流行的编程语言，包括脚本编程语言，如 Visual Basic、Java、C++、VBCcript、Jscript 等。

(3)数据源无关性。ADO 能访问任何 OLE DB 数据

源,并具有自适应性。

(4)OLE DB 功能完备性。ADO 允许 C++ 程序员直接访问底层 OLE DB 接口。

(5)可扩展性。ADO 使用数据源属性集动态说明特定数据提供者的属性。另外,ADO 允许访问被说明成列值的 COM 对象(如行集和流),从而提供了数据类型的可扩展性。

ADO 的对象模型包括七个对象,如图 4 所示。

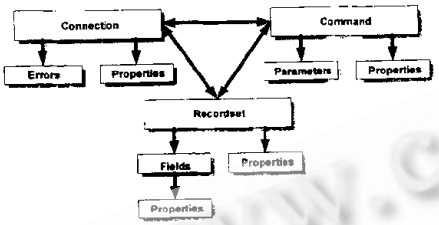

图 4　ADO 的对象模型

在 ADO 模型中,主体对象有三个:Connection、Command 和 Recordset,可被独立创建和释放。而 Parameter 对象虽然可以独立于 Command 对象创建,但不能独立于一个 Command 对象使用。Field、Error 和 Property 对象则不能独立创建,而只能依存于它们的父对象而存在。

ADO 的 Connection 对象封装了 OLE DB 的 Data Source 和 Session 对象,定义了和数据源间的简单会话。Connection 对象提供以下基本操作:定义连接属性,指定局部事务的范围,提供错误的精确定位等。

ADO 的 Command 对象封装了 OLE DB 的 Command 对象,说明要执行的数据定义或数据操作语句。如使用关系数据库,则这些语句为 SQL 语句。Command 对象通过 Parameter 对象集指定操作参数、定义要执行语句的行为。

ADO 的 Recordset 对象封装了 OLE DB 的 Rowset 对象。不管是查询结果还是由其他方式产生,Recordset 对象都提供了实际的数据接口。Recordset 对象提供了以下控制:锁定机制,游标类型的选择,一次存取的记录数等等。Recordset 对象使用一个 Field 对象集,其中包含了记录集各列的元数据(metadata),如名称、类型、长度、精度以及实际数据值。开发人员使用 Recordset 对象来操作记录、改变数据(假设底层数据源可更新)。

ADO 模型的每一个主体对象都包含了一个 property 对象集,通过 Property 对象动态说明特定数据提供者的性能。由于各个数据提供者的功能不一致,因此对 ADO 模型来说,允许开发人员动态设置与特定数据源有关的参数是非常重要的。这一点同时也避免了 ADO 模型由于使用与特定环境有关的属性而造成的错误。

5. ADO 编程实例

下面介绍一个简单的例子来说明在 Java 中如何使用 ADO 对象。

```
import com.ms.ado.*;
class test
{
  public static void main(String args[])
  {
    Connection cn=new Connection();
    Recordset rs=new Recordset();
    cn.open("dsn=pubs","sa","");
    rs=cn.execute("SELECT Name FROM Customers");
    while(!rs.getEOF())
    {
      for(int i=0;i<
```

6. 小结

做为一致的数据访问接口,由于 OLE DB/ADO 对象模型的简单性、可扩展性、数据访问的一致性以及易用性,UDA 的应用范围将越来越广泛。

参考文献

[1] OLE DB/ADO: Making Universal Data Access a Reality, Microsoft Corporation, 1998

[2] Aaron Skonnard, Say UDA for All Your Data Access Needs, April 1998

[3] 罗会涛,OLE DB 的概念与编程,计算机世界,1998.6.15

[4] 潘爱民,一致的数据访问技术 ADO/OLE DB(一),微电脑世界周刊,1999.4

(来稿时间:1999 年 6 月)