Solaris 2.5 与 NT 4.0 关系数据库信息互用的方法

中国科学院软件研究所 陈燕 刘明迪

本文给出 Solaris 2.5 与 NT 4.0 关系数据库通过 CORBA 实现互操作的一个解决方案。该方案允许用户采用面向对象技术，以统一的接口透明访问两个异构平台的关系数据库。

引言

所谓关系数据库互操作，是指在分布式数据库网络环境中，在网络任意节点上的用户，可以使用网络其他异构节点上的关系数据库中的资源。关系数据库互操作问题的研究，国外业界主要是沿着以下两个方面：一是发展关系数据库的有关标准；二是承认各厂商数据库产品的千差万别，建立关系数据库互操作支撑件。发展有关标准无疑可以使异种关系数据库互操作的实现基础更好，起点更高；但标准的建立难度大，时间长，无法解决当务之急。因此，在建立标准尚需时日和应符合标准的产品尚不完善的情况下，人们常把解决关系数据库互操作问题放在构造关系数据库互操作支撑件上。国外产业界对此做了不少的工作，已研究并推出一些产品性的基于不同实现策略的关系数据库互操作支撑件。其中代表性的产品有 Sybase 公司的 OpenClient/OpenServer 和 OmniSQL Gateway、Oracle 公司的 SQL* Connect 等。

本文旨在符合 CORBA 2.0规范的前提下，提出一个 Solaris 2.5 下 Oracle 7.0 关系数据库与 NT 4.0 下 SQL Server 6.0 关系数据库实现互操作的解决方案。

CORBA 概述

1.CORBA 系统的体系结构

CORBA（Common Object Request Broker Architecture）是对象管理集团 OMG 于 1991 年提出的中间件技术规范。它借助于客户/服务器计算模式和面向对象技术，在具有不同操作系统、语言、网络协议和硬件结构的异构系统间提供应用层的互操作性，从而有效地实现分布对象之间的互操作性。OMG 在它的 OMA 中定义 CORBA 的四个主要部分：ORB、对象服务、通用设施和应用对象。其中，ORB 是 CORBA 规范的核心，它定义 CORBA 的对象总线；对象服务为分布对象定义提供的系统级基本功能；通用设施定义直接由应用对象使用的功能；应用对象则指所有以 CORBA 为运行环境的应用。此外，CORBA 规范还包括：接口定义语言 IDL、静态调用接口 IDL Stub、ORB 接口、动态调用接口 DII、静态框架接口、动态框架接口 DSI、对象适配器 OA、接口池 IR 和对象实现池、以及 ORB 间互操作协议 IIOP。其中最重要部分是 IDL，所有 CORBA 对象和服务都严格通过 IDL 接口来定义。

2. 对象请求代理 ORB

ORB 是 CORBA 的核心，是对象间建立客户 / 服务

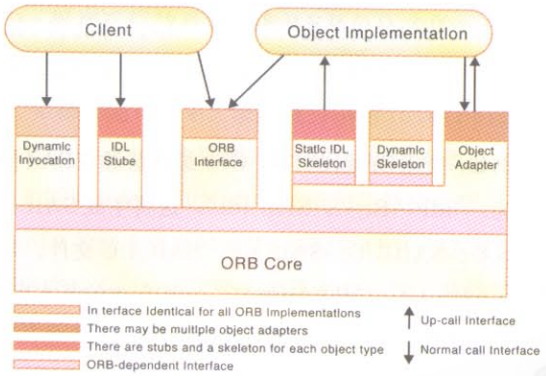

图 1　CORBA ORB 的体系结构

器关联的通信机制。通过 ORB，客户对象透明地调用位于本地或远地服务对象上的操作。由 ORB 解释和转交调用请求，并负责发现能够实现该请求的对象，传递参数，调用相应操作，将结果返回给客户对象。客户方不必关心服务对象的位置、编程语言、操作系统、以及任何非对象界面的系统信息。CORBA ORB的体系结构如图1所示。ORB 作为一个完整的对象，同时驻留在客户方和服务器方，为两者分别提供专门的操作。

3. 接口定义语言 IDL

CORBA对象使用接口定义语言IDL定义标准的对象接口。IDL是强类型的、语言中立的描述语言。提供给对象实现及客户调用的IDL接口描述，是与操作系统、编程语言无关的。CORBA 定义从IDL到编程语言的映射，通过映射，用户可以使用自己熟悉的语言编写CORBA对象以及实现客户程序。IDL也是面向对象的，它允许表示接口的抽象（封装）、多态消息（功能调用）和接口的继承。IDL 实现语言层次上的互操作。

4. 对象服务和通用设施

对象服务是一组基于IDL接口定义的系统级服务。目前 OMG 所公布的对象服务包括：命名服务、事件服务、生命周期服务、持久对象服务、并发控制服务、外部服务、关系服务、事务服务、时间服务、安全服务、交易器服务、许可服务、属性服务、查询服务等。通用设施定义是更高级的对象服务和接口。通用设施分为水平方向和垂直方向两种，水平方向的通用设施面向功能，垂直方向的通用设施面向应用领域。

通过 CORBA 实现关系数据库互操作的解决方案

本文给出Solaris 2.5下Oracle 7.0关系数据库与NT 4.0下SQL Server 6.0关系数据库通过CORBA（符合CORBA 2.0的产品Orbix 2）实现互操作的一个解决方案（本方案简称为InterDB）。该方案允许用户采用面向对象技术，以统一接口透明访问两个异构平台的关系数据库。所谓透明，是指屏蔽两个关系数据库在语义、语法和地理位置上的差异。本方案也提供了相应的安全措施。

1. 运行环境

InterDB涉及了一个分布式数据库网络系统环境，最小运行环境为：SUN 工作站一台、操作系统为Solaris 2.5、关系数据库为Oracle 7.0、Orbix 2 for Solaris 2.5；服务器一台、操作系统为 NT 4.0、关系数据库为SQL Server 6.0、Orbix 2 for NT 4.0。

2. 系统结构

图2描述了InterDB 的系统结构，InterDB由请求关系数据库互操作服务的客户应用程序、系统管理器、元数据字典、两种关系数据库系统及其转换器构成。它将各个部件封装成对象，依次插在ORB这根“软总线”上。

InterDB 中 Oracle 7.0 与 SQL Server 6.0 两种关系数据库系统均配有转换器，转换器将各数据库系统封装成对象。该对象接口定义数据库对象可提供服务的形式，该接口是由IDL语言书写的，而对象服务的具体实现则由转换器完成。转换器将来自系统管理器分解后的请求转换成符合该数据库规定的SQL语言，交给该数据库管理系统执行，再将返回结果转换成用户定义的表示形式，交给系统管理器以便综合处理后将结果返回给用户。

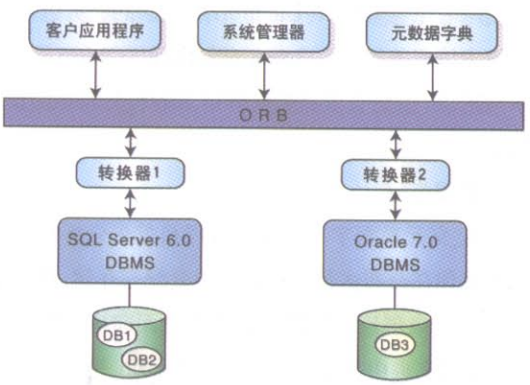

图 2　InterDB 的系统结构

元数据字典由系统管理器负责创建、维护和管理。元

数据字典主要包括允许全局用户使用的表、列、视图信息，以及这些信息的来源和使用权限等。元数据字典是动态更新的。为不同的客户应用程序系统管理器提供统一操作多个关系数据库的接口。客户应用程序向系统管理器发出数据库访问请求，系统管理器查询元数据字典，然后将请求分解为若干子请求，生成一定的执行计划，分别发送到相应的转换器具体执行，并综合处理各子请求的返回结果，将最终结果返回给客户应用程序。本方案中提供三个对象服务的接口定义。一是系统管理器向客户提供的统一操作多个关系数据库接口；另两个是系统管理器分别与两个转换器之间的接口。客户应用程序请求系统管理器提供服务，系统管理器再请求相应转换器提供服务。

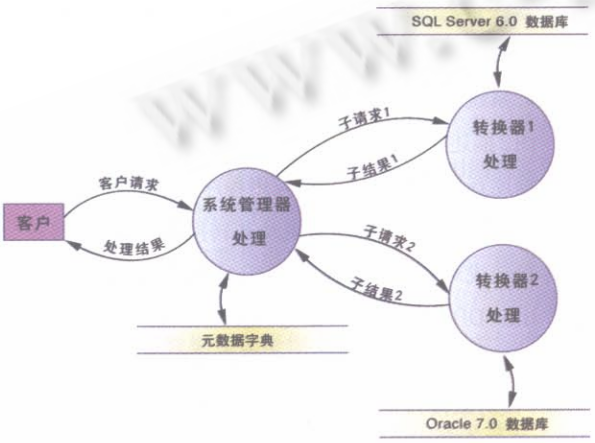

图 3 InterDB 的 0 层数据流图

这三个服务在被使用前，必须在Orbix的实现池中注册。Orbix Daemon维护着服务名与服务实现映射的实现池。每个服务有与之对应的服务进程。Orbix Daemon根据请求信息以及实现池中的注册信息确定并启动服务进程。请求者依据 Orbix Daemon 所返回的服务进程的引用，请求该服务操作。Orbix 对象代理（Object Proxy）是本地对象，由它负责完成具体请求的转交。其工作原理见图 3。

3. 实现步骤

(1)定义元数据字典。系统管理员首先分析和设计数据信息如何在分布式数据库系统中分配与存储，然后通过系统管理器来创建、维护和管理这个面向应用的数据字典。在数据字典中主要包括全局的表、列、视图以及它们的来源和使用权限等。

(2)定义接口。使用 CORBA IDL 语言，分别定义三个服务接口：客户与系统管理器接口（称为 Saif）；系统管理器与转换器 1 的接口（称为 SQLif）；系统管理器与转换器 2 的接口（称为 ORACLEif）。

(3)实现系统管理器和转换器。为实现系统管理器和转换器，需要根据接口定义的操作编写相应C++实现类。CORBA 支持两种实现类机制：BOAImpl 方法和 TIE 方法。但客户方并不需要关心服务器方实现机制以及实现类定义。

例如：使用Orbix的IDL C++编译器编译SAif接口命令为："idl -B -S SAif.idl"。这将生成 SAif.hh、SAifS.C、SAifC.C、SAif_i.h、SAif_i.C 文件。其中SAif_i.h和SAif_i.C是SAif接口C++实现类的基本框架，以此为基础编写类中各方法的实现代码。

(4)编写服务主函数和注册服务。服务接口及接口实现类定义后，要为服务编写主函数。主函数的目的是实例化实现类的对象，并等待客户请求。下面给出编写 SAif 服务主函数示例：

首先创建 SAif 实现类的对象，然后调用 CORBA::Orbix.impl_is_ready()。该函数的作用是通知 Orbix Daemon该服务已完成初始化工作，等待接收来自客户的调用请求。

服务主函数的代码经编译链接，生成服务的可执行文件（服务器方）。服务在被使用前，必须在 CORBA 实现池中注册，利用实现池建立服务名与服务实现间映射。Orbix 服务注册命令为："putit serverName commandLine"。ServerName 为服务名；commandLine 表示服务对应的可执行文件全路径。

(5)编写客户应用程序。在 Orbix 程序员手册中给出了编写客户对象的方法。首先，应该得到服务对象的引用；其次，访问服务对象在IDL接口中定义的属性和操作。

服务对象引用是指服务对象在客户对象中的标识。当服务对象引用进入客户地址空间后，Orbix建立对象代理，作为远程实现对象的代表。Orbix将代理对象上的函数调用转交给相应实现对象中的函数。

客户应用程序通过系统管理器服务名对系统管理器提供的服务进行定位，从而获取对该服务对象的引用。服务定位具体通过调用静态成员函数 _bind()来完成，它返回对代理对象的引用，然后通过对象引用，调用服务对象定义的各种操作，进行多数据库访问。

下面按照 SAif 接口，给出客户应用程序的示例：

①调用静态成员函数_bind()，获取对代理对象的引用。

②使用指定用户帐号和口令，调用 InterPrepare()，

准备一条将执行的多数据库操作语句，如检索满足一定条件的数据或更新某数据，调用后填充DBPROC中的SQLStatement域。

③调用InterExecute()，执行已准备的多数据库操作语句，根据运行结果填充DBPROC中的nCols、nRecordCount、nCurrentRecord、set等域。

④根据运行结果，决定是否调用InterFetch()、InterMoreResults()来处理并显示返回的结果集。

⑤调用InterEnd()，结束该语句并释放资源。

(6)多数据库语言的选择。1992年11月，ANSI和ISO共同研究公布了数据库语言SQL的新标准，替换了原来的SQL89标准，称为SQL92或称为SQL2标准。

SQL92标准规定的内容分为三级：初级、中级和完全级。目前个别主流的数据库产品达到了初级的要求，还没有哪个SQL产品实现SQL92的所有性能，而且没有哪两个SQL产品具有完全相同的SQL。数据库厂商们甚至还引入一些新的特性，加强其SQL的特殊性，使各种产品差异更大。

由于Oracle 7.0和SQL Server 6.0两种关系数据库产品所支持的功能并不完全相同，例如：Oracle 7.0支持的日期/时间类型为DATE，表示时刻值，当不明确给定时间时取当天午夜为缺省值，格式为'14-JUN-1998'；而SQL Server 6.0支持的日期/时间类型为DATETIME，也表示时刻值，但格式为'06/14/1998'或'03/15/1999 12:00 AM'。又如：SQL Server 6.0为访问可编程数据库提供了可调用API应用程序接口；而Oracle 7.0除提供可调用API外还提供嵌入式SQL接口。再如：SQL Server 6.0客户/服务器结构是建立在远程事务处理能力基础上，对分布式事务处理提供支持，在SQL Server API中有专用的两阶段提交调用集来同步两个SQL Server拷贝的提交，但它缺乏足够的透明性；而Oracle 7.0则采用强有力的、透明的两阶段提交机制，来保证分布事务更新的完整性。它的两阶段提交还可用来操作其他数据库和文件系统，甚至包括那些本身不支持两阶段提交的老系统。由此可见，虽然存在数据库语言标准，但各厂家产品并不完全遵循标准，所提供的数据库语言并不完全相同，都各具有自己的特点。在本方案中，多数据库系统的数据库语言选择是很重要的问题。原因是：第一，如果多数据库的SQL语言是标准的，那么就可以保证客户端程序的开放性，使得用户的程序容易移植；第二，如果多数据库系统的数据库语言的功能定义强了，则各局部数据库实现不了，反之，如果定义弱了，则不能完全体现各局部数据库的功能，也不能满足用户的需要。因此我们建议在多数据库管理系统中语言选择要根据用户实际应用背景和局部数据库的能力，正确采用SQL92中的初级、中级或完全级。所以我们在本方案中不作具体规定。

4. 服务安全

InterDB利用CORBA安全服务提高关系数据库互操作的安全性。对于在实现池中注册的每一服务，Orbix维护着两个访问控制表，即启动控制表和调用控制表。启动控制表定义通过Orbix Daemon启动服务的用户或用户组；调用控制表定义使用服务的用户或用户组。Orbix实现身份验证级的安全性。服务器方的系统管理员可根据需要利用Orbix的chmodit命令为不同服务建立访问控制表。通过访问控制表，可较有效地限制网络用户对各数据库的访问，防止非法用户对数据资源的恶意窃取和破坏，起到安全隔离作用。此外，Orbix支持服务实现代码中的安全检查，即利用Orbix提供的get_principal()操作，获取客户应用程序的信息，从而对该访问者加以限制，而设置访问控制表更为灵活方便。

结束语

异构平台关系数据库的互操作问题在实际应用中经常遇到，如MIS系统、大型的集成软件工程环境等。解决这个问题的方法很多，但用面向对象技术解决这个问题，还是新的尝试。我们认为这个方案是可行的，它能为应用系统提供统一的数据管理、检索、存储服务，有助于各种应用系统的实现。此外，CORBA 2.0规范对数据库应用并没有特殊、专门的支持，数据库互操作对于CORBA来说，也仅是普通的应用。本方案给出了通过CORBA实现关系数据库互操作的一个途径。对多数据库系统所遇到的一些问题，除采用上述方法外，本方案也不涉及更多的解决方案，还需要业界共同去探索。■