基于CORBA的分布式通用数据库查询对象模型的研究

梁寿愚 华南理工大学控制科学与工程学(510640)

Abstract

This paper has presented a CORBA-based model for distributed general query database,this model combines technology of distributed object,technology of heterogeneous databases query and technology of data type discriminator which resolve the problem of data transfer in heterogeneous distributed system,provide the same interface component for general query database which can transplant heterogeneous platforms and query any database.Used this model implementation I have successfully developed the information exchange soft for heterogeneous distributed system in South Electric Manager Center.

Keywords:object for distributed general query database,CORBA,technology of distribute object,technology of ODBC API function query database,data type discriminator

摘 要

本文提出基于CORBA的分布式通用数据库查询对象模型,该模型结合目前最先进的分布式对象技术、异构数据库访问技术以及为了解决异构分布式计算机系统数据传输问题而提出的数据类型鉴别器技术,为异构分布式数据库系统提供可跨平台移植的、跨数据库查询的具有统一接口的分布式通用数据库查询构件。利用分布式通用数据库查询对象模型概念的实现成功开发出某电力部门异构分布式计算机信息交换系统软件。

关键词:分布式通用数据库查询对象,CORBA,分布式对象技术,ODBC API数据库访问技术,数据类型鉴别器

0 引言

多数大型企业现有的底层支持环境是由多种不同类型的计算机软硬件组成，同时在地理上往往又分布在不同的场地,需要通过网络互连访问,所以构成一个异构的分布式计算机环境。在这种环境中为了能对数据信息进行集成和一体化的管理，需要在底层支撑环境与上层应用系统之间建立一个方便、高效的信息集成环境。然而,异构的分布式计算机环境中的数据库属于多数据库范畴,逻辑结构上存在着异构性、分布性和自治性，单纯依靠现有的网络技术或数据库技术都不能很好地解决企业内部信息资源共享的问题。如何能屏蔽企业环境中各种层次上的异构性，如何实现在地域上分布的“数据库孤岛”间进行数据交流和信息共享,成为分布式数据库系统发展的技术难点。

一种解决方案是放弃原有的数据库系统，重新设计全新的分布式数据库系统，这种方案实现起来非常困难。因为,首先是工作量巨大;其次是导致数据库太专用而又重复；再者企业的发展会不时地要求增加一些新的数据资源，从而使这种分布式的数据库维护相当复杂。另一种解决方案是将原有的数据库系统集成为一个支持异构的分布式计算机环境、具有统一的集成模型、用户能够透明地访问异构的分布式系统中的多种数据库资源。

本文采用第二种解决方案，设计基于CORBA的分布式通用数据库查询对象模型，分布式通用数据库查询对象并不是一个已有的名词，它是我在某电力部门异构分布式计算机系统集成接口软件这个工程项目的开发过程中提出来的新概念，这个概念结合了目前最先进的分布式对象技术、异构数据库访问技术以及为了解决异构分布式计算机系统数据传输问题而提出的数据类型鉴别器技术，为异构分布式数据库系统提供可跨平台移植的、跨数据库查询的具有统一接口的分布式通用数据库查询构件。

1 全新概念:分布式通用数据库查询对象的提出

分布式对象就是可被远程访问的对象，它不仅可像普通对象一样使用,也可从网络的其他地方访问到。分布式对象可以在不同的应用和用户间共享信息,让位于不同地方的人协作完成一项任务，或是在网络范围内分配计算，其形式就是把应用程序中需要的功能归结为一个个对象并分布在网络上，每一个对象可使用本系统或者其他系统的对象提供服务。CORBA为我们提供了分布式对象技术结构。如果把这种对象用

于实现数据库的查询，那么它就是一个具体的分布式数据库查询对象。如果这种分布式数据库查询对象是在网络中的某一台服务器针对此运行系统的特定环境(例如:特定的操作系统、特定的数据库环境)开发的特定对象,它的实现不难实现,编写与所需数据库、数据表相关联具有特定数据类型的 IDL，采用 CORBA 技术在分布式计算机系统中可以很方便的做到。然而这种分布式数据库查询对象有明显的缺点，就是不能在异构环境移植或是跨数据库查询，所以这不是我要实现的目标。

分布式通用数据库查询对象的提出要求解决这个问题，它的目标是能够在分布式计算机系统中实现跨数据库查询，不是对特定数据库数据表的特定字段进行查询,而是能够自动根据 SQL 查询语句返回相应的数据集,并且要求这个对象代码是可以跨平台移植的。简单的来说就是一劳永逸的做法，只要能实现这个对象,它就能够在网络中的任何平台上安装,并且可以查询任何数据库,返回客户需要的任何形式的数据集,而以后要做的工作或许只是不断重用这个组件。

可能许多人认为这是不可能的，然而我已经在电力调度中心的项目中把它实现。我提出的分布式通用查询对象这个全新的概念结合三大技术:CORBA 的分布式对象技术、ODBC API 数据库访问技术、数据类型鉴别器技术。

2 分布式通用数据库查询对象结合三大技术

2.1 CORBA 的分布式对象技术

CORBA(Common Object Request Broker Architecture,公共对象请求代理体系结构)是 OMG 组织在 1991 年提出的公用对象请求代理程序结构的技术规范,并迅速成为分布式对象计算领域的主流技术。结合计算机工业中两个重要的趋势：面向对象软件开发和客户机/服务器计算,CORBA 规范软件系统采用面向对象的软件实现方法开发应用系统，实现对象内部细节的完整封装,保留对象方法的对外接口定义,从而提供软总线机制,使得在任何环境下、采用任何语言开发的软件只要符合接口规范的定义，均能够集成到分布式系统中;引入中间件(Middle ware)作为事务代理,完成客户机(Client)向服务对象方(Server)提出的业务请求,从而实现客户与服务对象的完全分开,客户不需要了解服务对象的实现过程以及具体位置。CORBA 在基于网络的分布式应用环境下实现应用软件的集成,使得面向对象的软件在分布、异构环境下实现可重用、可移植和互操作。

分布式通用数据库查询对象要求跨平台可移植、面向对象的分布式计算应用程序;CORBA 提供不依赖于计算平台、编程语言、网络协议的编程接口和模型，使得它非常适合分布式通用数据库查询对象应用程序的开发和系统集成。

分布式通用数据库查询对象对应的伺服程序的代码实现对数据库的通用查询,POA 把这个伺服程序链接到分布式通用数据库查询对象上，服务程序通过 ORB 监听客户端的请求,当客户程序所提出对分布式通用数据库查询对象的调用请求时，伺服程序执行请求,并且通过 POA 和 ORB 向客户程序返回 out 和返回值,这个返回值在这里通常是一个数据集。可见,分布式通用数据库查询对象应用 CORBA 分布式对象技术实现客户端对服务器端的对象调用。

2.2 ODBC API 数据库访问技术

开放数据互连 (ODBC,Open DataBase Connectivity) 是 Microsoft Windows 开放服务体系结构(WOSA,Windows Open Service Architecture)的一部分，是一个数据库访问的标准接口。 ODBC 是 80 年代发展起来的数据访问技术,它提供了对关系数据库访问的统一接口，实现了对异构数据源的一致访问。ODBC 技术的推出为异构数据访问提供了解决方案,引起了数据访问方面的巨大变革,成为最通用的数据访问技术。

ODBC 已经成为一种标准,几乎所有的关系数据库都提供 ODBC 驱动程序，这使 ODBC 的应用非常广泛，基本上可用于所有的关系数据库；另一方面，ODBC 可以在 Windows、UNIX、Linux 等多种操作系统上使用,因此是一种跨平台的应用程序的有用工具。ODBC 体系结构：

ODBC 对各种关系数据库的开放性以及在多种操作系统可跨平台的移植性，符合我们提出的分布式通用数据库查询对象所必须的基本条件;并且 ODBC 是一种低层的访问技术,因此 ODBC API 可以使应用程序从底层设置和控制数据库，完成一些高层数据库技术无法完成的功能；高层数据库技术由于封装了许多在大部分时间和情况都不使用的代码导致影响查询效率,而直接使用 ODBC API 函数可以由程序员自己控制代码的复杂程度，是提高访问查询速度的有效手段;这些都是我选择 ODBC 作为分布式通用数据库查询对象访问数据库技术的重要原因，最终测试的结果也证明了我的观点是正确的。

2.3 数据类型鉴别器技术

数据类型鉴别器是我在某电力部门异构分布式计算机系统集成接口软件这个项目为了解决异构分布式计算机系统数据传输问题而提出的新概念，在这里将对这个新概念进行全面的论述。

2.3.1 为什么会出现数据类型鉴别的问题

CORBA 可以通过 IDL 影射成多种编程语言,但因此引出了一个重要问题。OMG 接口描述语言 IDL 作为接口定义的通用标准,被各种不同的中间件所采纳,包括 DCOM、J2EE 和 CORBA。使用接口描述语言 IDL 编写的对象接口,使得与语言无关的独立性成为可能。

IDL 的大量篇幅涉及到数据类型的定义,只有当数据的类型用 IDL 定义时,这些数据才能在客户程序与服务器程序之间交换。你不能在客户程序和服务器程序之间随意地交换数据,因为它会破坏 CORBA 的语言独立性。也就是说 CORBA 为了维持语言的独立性,不得不定义其中几种较为通用的数据类型,以便可以影射成不同的语言,它们被各种语言所认识,但是许多语言的特定数据类型在 IDL 中并没有定义(例如:日期),特别是在数据库查询中,更会有许多各种各样的数据类型。如果遇到这些数据类型需要传输,有必要进行数据类型的转换。

2.3.2 数据类型鉴别器概念的提出

IDL 的数据类型有基本类型和用户定义类型之分,基本 IDL 类型包括:短整型(short)、长整型(long)、无符号短整型(unsigned short)、无符号长整型(unsigned long)、浮点型(float)、双精度型(double)、字符型(char)、字符串型(string)、布尔型(bool)、八位字节(octet)、any 类型;用户定义类型允许你定义复杂类型:枚举(enum)、结构(struct)、联合(unin)、数组、序列(sequence)。

考虑到我们在客户端向服务器端提出传输某些数据的要求,这些被返回的数据或许在 IDL 里被定义的异常复杂,一般用户定义的类型,又有序列又有结构,序列里有结构,结构里有序列都有可能,无非是想很方便的取回需要的数据,并且要求特定的形式返回(例如一个表要以矩阵形式返回,我们下面的例子会提到)。然而当赋值时,特别是从数据库取出值时,就会遇到很多麻烦,因为这时只能对 CORBA 中 IDL 定义的基本数据类型赋值,而这些并不一定符合赋值的要求(例如没有相应的定义),当然可以使用很笨的方法,对每个要传输的数据类型都相应的进行手工转换。实际上是很难做到的,因为当程序员在编写服务器端程序时仍然无法知道客户端需要哪种数据和确切的数据类型,所以只能编写通用的数据类型;与此同时,客户端也想知道传回来的是什么数据类型(我们在下面例子都会提到)。所有这些都只能在 IDL 里编写数据类型鉴别器才能解决。

2.3.3 数据类型鉴别器的设计与实现

IDL 中编写数据类型鉴别器的思路是:通过在 IDL 定义一个枚举类型来代替用户自定义的数据类型,再通过联合来选择对这些数据类型相应的 IDL 基本数据类型的定义。

下面通过一个比较完整的例子来说明这个问题:

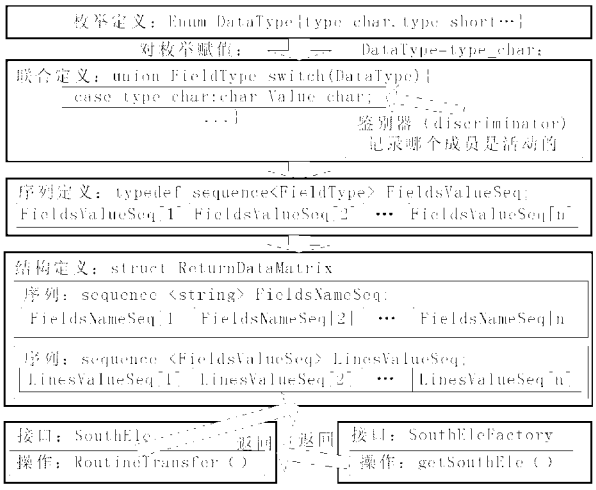

图 1 CORBA 分布式异构系统数据类型鉴别器设计的 IDL 定义原理图

为了简化论述,这个 IDL 定义原理图里没有出错处理。首先对这个 IDL 的接口和操作进行解释:这里用到一个对象工厂 SouthEleFactory 接口通过 getSouthEle 操作创建返回一个 SouthEle 对象,而 SouthEle 对象里有一个操作 ReturnDataMatrix RoutineTransfer(in string ConnStr, in string SQLStr),当输入一条连接语句和一条 SQL 语句时它会返回一个数据类型 ReturnDataMatrix,从前面的结构定义可以看出,ReturnDataMatrix 拥有域名序列、域值序列的行序列也就是一个矩阵,可见它相当于一个数据集,也是操作的对象。

从这个 IDL 可以看出,由于客户端请求是不确定的,有时请求这个表和字域,有时请求另外的表和字域。我们不能在服务器端程序中预先定义传输的数据类型,客户端程序也不可能在调用前先了解它需要的数据的类型,于是定义了一个数据类型鉴别器。

在前面提到的 IDL 的数据类型当中,枚举和联合与鉴别器的关系最大。枚举类型的定义和 C++相似,这个 IDL 引入一个 DataType 的类型,假设用来代替用户希望定义的数据类型。注意:IDL 只能保证枚举的序数值从左到右递增,没有任何限定并且甚至不是邻接的,当然人们不用去理会它。我们使用一个联合 FieldType 对它进行选择分配数据类型的空间,选择定义基本的 IDL 数据类型。联合的语义与 C++相似,每次只允许联合中的一个成员是活动的。但是 IDL 增加了一个鉴别器(discriminator),用来表示当前哪个成

员是活动的。

这里的 IDL 的枚举类型影射到 C++里仍然是枚举类型，可以很方便的在程序中使用；而 IDL 的联合不能被影射成 C++联合，因为 C++不允许联合中包含带有特殊构造函数的类成员，C++的联合也不带有判别功能，IDL 的联合于是被影射成 C++的类。类中有一个用于存取每个联合成员的成员函数和一个用于修改每个联合成员的成员函数；此外还有用于控制鉴别器和解决初始化和赋值问题的成员函数。

IDL 的原理图已经有了一个很好的思想，但不是写出 IDL 就万事大吉，因为工作只是开始起步，其实这个 IDL 也是反复编写实现后经过多次修改的结果。不管是编写服务器端的实现还是客户端的应用程序都是很复杂的。

以下框图是我的一个解决方案：

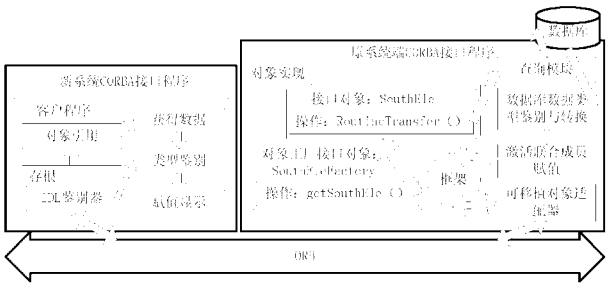

图 2 CORBA 分布式异构系统数据类型鉴别器解决方案

原系统端 CORBA 接口程序的对象实现已经在 IDL 的定义中进行了解释。这个服务器端的实现代码首先自动对从数据库中提取的数据进行辨别后进行了相应的转换，假设我们使用的数据库是 SQL Server，编程语言是 C++，也就是把 SQL Server 定义的数据类型转换为 C++的数据类型；例如把数据库的时间类型 Datetime 转换为 C++的 string 类型，这样转换只是为了更容易赋值于 IDL 的基本类型(例如：string 类型)。

接着必须定义一个 FieldType 联合变量；前面已提到，IDL 的联合被影射为 C++的类，所以确切来说这个“变量”实际上是用这个类定义的对象，这个类里已经有了 IDL 里自定义数据类型的成员函数，调用相应的成员函数可以给一个联合成员赋值或者说激活一个联合成员；作为一种副作用，通过给成员赋值将设置鉴别器的数值。

由此可见，在服务程序的实现中，数据库的数据类型被转换为 IDL 的基本类型，并在对联合变量的赋值时同时设置了鉴别器的数值，为客户端对数据类型的鉴别做好了准备。

客户端程序把数据通过 ORB 从服务器端取过来后，把结构 ReturnDataMatrix 分解成序列，再把序列最终分解为 FieldType 联合变量，然后通过其成员函数 _d()读取鉴别器的数值判断数据类型；根据此判断，再调用相应的自定义数据类型的成员函数，便可以使用其数值进行赋值或显示操作了。注意：此时在客户端中可以很清楚的了解到服务器端（也是 IDL 编写者）定义的数据类型。

2.3.4 数据类型鉴别器技术总结

CORBA 在数据传输时所涉及的数据类型之间的转换非常繁琐，特别是从数据库中动态查询所获得的数据集，尚未发现有成功转换的例子；我提出的数据类型鉴别器的概念较好地解决了这个问题，它能够使数据从服务器端传输到客户端能自动地被鉴别出是什么数据类型，即使是异构平台之间的数据传输也会根据相应的平台数据类型存储方式进行转换。

3 结束语

基于本文提出的分布式通用数据库查询对象模型，分布式数据库系统设计者可以任意选择数据库进行独立的设计、存放、管理和维护数据信息，另一方面又可以通过全局查询语言(SQL 语句)访问共享的数据信息，为解决企业异构分布式计算环境中的“数据库孤岛”之间进行数据交流和信息共享问题奠定了基础。笔者编写异构分布式计算机系统的跨平台跨数据库的通用查询软件成功，并开发出某电力部门异构分布式计算机信息交换系统软件，这个模型的提出起着至关重要的作用。

参考文献

1 Jason Garbis,Dirk Slama,Perry Russell.CORBA 企业解决方案[M].李师贤,郑红,吴涛等译.北京：机械工业出版社，2001

2 OMG.The Common Object Request Broker: Architecture and Specification (CORBA) Version2.3.1[M].1999

3 OMG.The Common Object Request Broker: Language Mapping(CORBA) Version 2.3.1[M].1999

4 Michi Huenning,Steve Vinoski.基于 C++ CORBA 高级编程[M].许金梧，许科，吕志民，等译.北京：清华大学出版社，2000

5 Borland/Inprise 公司.C++Builer 5 开发人员指南[M].梁志刚，汪浩，康向东，刘存根，等译. 北京：机械工业出版社，2001

6 Borland/Inprise 公司. VisiBroker for Java 开发人员指南[M].李文军，周晓聪，李师贤，等译. 北京：机械工业出版社，2001

[收稿日期：2002.6.6]

欢迎订阅 2003 年 工业控制计算机 杂志 月刊 邮发代号：28-60 全年定价：60.00 元