数据类型鉴别器在 CORBA 分布式异构系统中的应用

朱 斌[1], 梁寿愚[1], 朱海云[2]
(1. 华南理工大学, 广东广州 510640; 2. 广东省电信科学技术研究院, 广东广州 510630)

摘 要: 分布式异构系统中的数据类型转换, 一直是计算机、自动化领域内人们关心的重要问题. CORBA 应用于电力系统, 处理异构系统的数据类型转换, 在国内尚无成功的经验. 本文提出将 CORBA 应用在电力系统并提出了数据类型鉴别器的概念, 较好地解决了某电力控制中心的分布式异构系统中数据库的通用查询问题, 实验结果证明是可行的.

关键词: 分布式异构系统; 数据类型鉴别器; 数据库的通用查询; CORBA
中图分类号: TP311.52 文献标识码: A 文章编号: 0372-2112 (2002) 11-1617-03

Use of Data Type Discriminator in Realizing General Query Database for CORBA Distributed Heterogeneous System

ZHU Bin[1], LIANG Shou-yu[1], ZHU Hai-yun[2]
(1. *South China University of Technology, Guangzhou, Guangdong 510640, China;*
2. *Guangdong Telecommunication Academy of Science and Technology, Guangzhou, Guangdong 510630, China*)

Abstract: Data type conversion in distributed heterogeneous system is all along an important problem related to the computer and automatic domain. In electric power system, there wasn't any successful application that use CORBA in such a conversion in our country. This paper presents that CORBA can be used in electric power system, and further, it presents a new concept of data type discriminator. These two ideas above are feasible with the solution of the problem of a general query about the database in a distributed heterogeneous system of an electric control centre.

Key words: distributed system; data type discriminator; general query about database; CORBA

1 引言

在计算机的发展过程中, 计算机的结构以及微处理器的指令没有统一的标准, 形成了所谓的异构系统. 异构系统之间具有不同的物理结构, 基于不同的操作系统, 这种不兼容性使他们之间难以协同工作, 要把它们组成分布式计算机系统, 成为业界的一大难题. 也曾经有许多解决方法, 例如它们之间使用 TCP/IP 进行通讯以协调工作, TCP/IP 主要是基于传输的, 多用于通讯业的标准; 而基于接口对象的 CORBA 则更适合在异构系统之间的协调工作[1]. 同样, 数据库的发展也出现这个问题, 各个数据库开发商使用自己的标准来定义数据类型, 也没有形成统一的标准, 因而对不同数据库的查询比较困难. 近来比较多使用 XML 接口方案, XML 对数据形式的描述具有很大的优势, 有利于数据的协作; 但这些数据之间的协作使用还需要研究. 在本文中, 首先简要介绍 CORBA 及 IDL; 其次论述 CORBA 通过 IDL 影射对各种编程语言的支持, 而引起的最大问题——数据类型转换; 然后提出使用数据类型鉴别器的概念以解决这个问题; 最后通过一个实例讨论如何利用数据鉴别器实现对数据库的通用查询.

2 CORBA 简介

CORBA(Common Object Request Broker Architecture) 公共对象请求代理体系结构[2] 是 OMG(Object Management Group) 对象管理组织在 1991 年提出的公用对象请求代理程序结构的技术规范. CORBA 规范充分利用了现今软件技术发展的最新成果, 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; 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; 提供软总线机制; 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; 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. 对象接口定义实际上是指 IDL(Inteface Definition Language) 接口定义语言, IDL 使用统一的接口定义把客户端和服务器端彼此联结起来; 同时通过 IDL 影射出不同的编程语言. 可见, IDL 在 CORBA 中起着重要的作用.

收稿日期: 2001-05-08; 修回日期: 2002-06-27

3 CORBA 支持多种编程语言引起的问题

CORBA 可以通过 IDL 影射成多种编程语言,但还存在一些问题. OMG 接口描述语言 IDL[3] 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,具有与语言无关的独立性. IDL 在客户机和服务器程序之间建立统一的约定,描述在应用程序中需要使用到的类型和对象接口. 因为这些描述与实现的语言无关,所以不必考虑客户程序的编程语言与服务器程序的编程语言是否一致. 事实上,在分布式异构系统的客户端和服务器端之间需要传递哪些数据类型,服务器端集成哪些对象接口,接口中需要哪些操作,都可以利用最简单的文本编辑器把它写出来. 有趣的是,如果拿到一份 IDL 文本,不管服务器端如何实现接口,都可以很快地编写出客户端的程序. 一旦 IDL 接口定义完成,就可以使用 IDL 编译器把它映射成根(Stub)代码和框架(Skeleton)代码. 根(Stub)代码属于客户端,框架(Skeleton)代码属于服务器端. 被影射的可以是 C++ ,Java等代码,利用这些代码可以方便的扩展成客户端应用程序和服务器端应用程序. 同时,IDL 与语言无关的特性也引起了一些问题. 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,但不能随意交换数据,因为会破坏 CORBA 的语言独立性. 即 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,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. 如何利用 IDL 设计数据类型鉴别器,并对所传输的数据库的数据类型进行相应的转换,是本文所讨论的主要问题.

4 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[4],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). 当客户端向服务器端提出传输某些数据的要求时,这些被返回的数据或许在 IDL 里被定义得异常复杂. 一般用户定义的类型,又有序列又有结构,序列里可能有结构,结构里可能有序列,目的是为了方便的取回需要的数据,并且要求特定的形式返回. 然而当赋值时,特别是从数据库取出数值时,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,而这些不一定符合赋值的要求,当然可以对每个要传输的数据类型相应的进行手工转换. 实际上是很难做到的,因为编写服务器端程序时,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;同时,客户端也想知道传回来的是哪种数据类型(我们在下面的例子里会提到). 这些只能在 IDL 里编写数据类型鉴别器解决.

5 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:通过在 IDL 定义一个枚举类型,代替用户自定义的数据类型;通过联合定义,选择对这些数据类型相应的 IDL 基本数据类型.

下面通过一个比较完整的例子来说明这个问题:

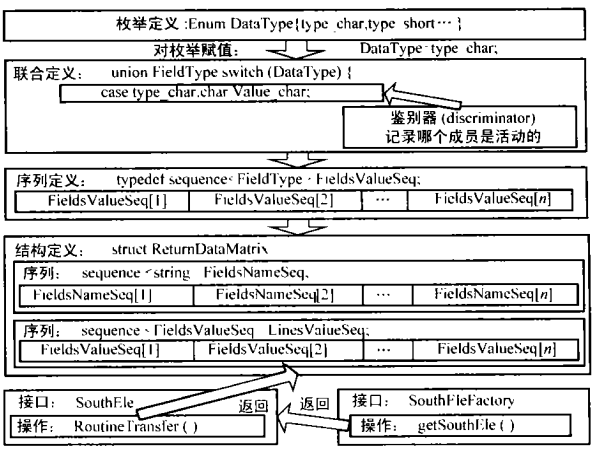

图 1 CORBA 分布式异构系统数据类型
鉴别器设计的 IDL 定义原理图

为了简化论述,这个 IDL 定义原理图里没有出错处理. 它是某电力调度中心分布式异构计算机系统接口软件 IDL 的一部分. 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,这是一个矩阵,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[4],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[5],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 的原理图设计是成功的, 但只是开始起步, 其实这个 IDL 也是经过反复编写、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.

以下框图是我们的一个解决方案:

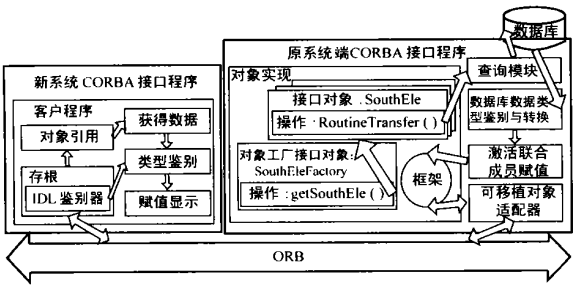

图2　CORBA 分布式异构系统数据类型鉴别器解决方案

这个服务器端的实现代码, 自动对从数据库中提取的数据, 进行辨别后再作相应的转换, 假设使用的数据库是 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定义的数据类型.

6　实验结果

采用这个 CORBA 分布式实现异构系统数据类型鉴别器解决方案, 我们已经在某电力调度中心分布式异构计算机系统接口软件中实验成功, 并且了分布式异构系统中跨平台跨数据库的远程通用查询, 就像在本地查询一样方便. 实验结果见图 3(a) 和图 3(b) , 可见实际应用时, 常用的单用户、字域数为 5000 时, 数据库的查询传输时间不超过 0. 3 秒; 20 个用户、字域数为 5000 时的数据库查询传输时间为 1 秒, 这种速度可以满足一般工程实际应用的要求.

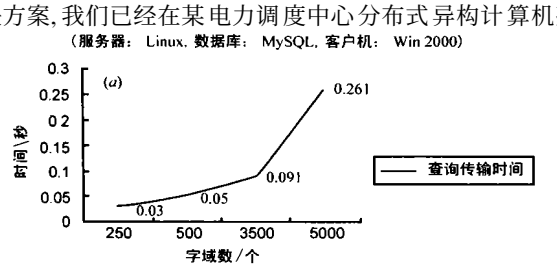

图3(*a*)　单用户查询传输时间图

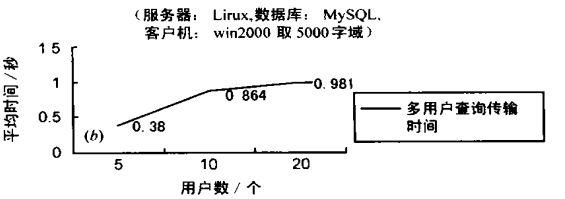

图3(*b*)　多用户查询传输时间图

7　结论与展望

CORBA 标准中语言影射的独立性, 使我们充分利用原系统的代码, 而且具有即插即用的扩展性, 它的稳定和可靠的性能较好. 但目前 CORBA 在国内应用较少, 主要是因为数据传输时所涉及的数据类型之间的转换非常繁琐, 难以实际应用于数据库的查询; 特别是电力系统行业, 长期以来国内急需解决异构系统中跨平台跨数据库的远程通用查询问题, 但至今没有使用 CORBA 的成功经验. 我们提出数据类型鉴别器的概念较好地解决了这个问题, 通过这个问题的解决, 可以使 CORBA 在我国分布式异构系统中(特别是在电力行业中) 的应用更加广泛.

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]　Version2. 3 1, The Common Object Request Broker: Architecture and Specification (CORBA) [S] .

[3]　Version 2 3 1, The Common Object Request Broker: Language Mapping(CORBA) [S] .

[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.

作者简介:

朱　斌　男, 1940 年 3 月出生于云南个旧. 教授, 研究生导师, 中国电子学会高级会员, 1963 年毕业于华南理工大学计算机专业, 毕业后至今长期在华南理工大学计算机系从事教学和科学研究工作, 参加和主持过国家自然科学基金、国家九五攻关、广东省多项科研, 主要研究方向是计算机应用、智能控制, 曾获国家教委、省科委科技进步奖.