对象关系数据库管理系统体系结构的研究

李 明 叶晓俊

(清华大学计算机科学与技术系 北京100084)(清华大学软件学院 北京100084)

An Approach for ORDBMS Architecture

LI Ming[1] YE Xiao-Jun[2]

(Department of Computer Science[1],College of Software[2],Tsinghua University,Beijing 100084)

Abstract In last years,with the Object-Oriented Programming popular and the field of database application extended,many DBMS manufacturers integrate the object model into traditional Relational Database Management Systems (RDBMSs)one after another,in order to meet the needs for complex engineering applications. This paper firstly gives a total remark on the reason for the prevalence of the Object-Relational Database Systems (ORDBMs)and on the benefit of ORDBMSs comparing with RDBMSs and Object Oriented Database Management Systems(OODBMSs),then introduces an approach for ORDBMS architecture which integrates the Java class library into ORDBMS metadata repository.

Keywords Object-relational model,ORDBMS architecture,Database metadata

1 ORDBMS的提出

数据管理技术最早源于文件系统,为了解决数据从程序的分离、数据的持久化和共享、数据的冗余等问题,产生了数据库管理系统(DBMS)。传统的DBMS经历了层次、网状、关系三个阶段。关系数据库解决了数据存储路径的隐藏,数据之间的联系不必在设计数据库时就给出,而是在查询(或事务处理)时通过关系运算动态生成,这样的数据模式具有简单、易用,易扩展等特点,因此关系数据库成了最为流行的数据库。但是由于数据库应用范围的不断扩展,关系数据库在工程应用(CAD/CAM/CAE)、多媒体管理、地理信息管理等方面明显表现出以下不足[1~3]:

(1)无法直接描述现实世界中对象之间的复杂关系:如继承、包含、组合等关系;无法直接表达现实世界的许多信息;另外,为满足数据库建模的要求必须对现实世界的实体进行必要的信息分解。

(2)语义信息都是在事务处理时(应用程序、存储过程、触发器中)决定,这与现实世界中某些信息(比如记录的标识、对象之间的各种关系)是确定的不相符。

(3)没有对象的概念,不能建立对象的行为模型(即没有实现数据和方法在数据库中的绑定),缺乏数据抽象和数据隐藏、无法表示复杂的数据类型(对象)。

随着OOP的成熟,人们开始研究面向对象的数据库管理系统(OODBMS),但是进展比预期的要缓慢得多,究其原因主要是OODBMS在以下几个方面的不足[4~6]:

(1)面向对象数据库系统在程序接口、对象实现、对查询的支持等方面存在许多差异,给OODBMS的应用程序开发、移植等都造成不良的影响。

(2)在一些方面无法与RDBMS匹敌:由于底层实现仍然采用面向对象的编程方法,导致了对于视图的支持不足,模式演化艰难,与应用程序语言结合很紧密,另外,在鲁棒性、可伸缩性、容错性、性能等方面也不如RDBMS。

(3)应用开发工具如报表、数据转换和管理工具不多,对客户机/服务器环境的支持不够,因此,真正建立在OODBMS上的应用很少。

(4)与原来的关系数据库差异较大,无法做到对它的兼容。

(5)无法将对象模型和关系模型集成:对于像CAD/CAM一类的工程应用,OODBMS虽然解决了大对象存储和面向对象建模的需求,但如何进行OODBMS应用系统与基于关系的数据库应用(MIS/ERP/CRM等)的集成仍然是个问题。

在这样的情况下,各大数据库厂商纷纷推出相应的对象关系数据库管理系统(ORDBMS)[7,8]:既具备关系RDBMS的功能,同时又支持面向对象的特征:抽象数据类型(ADT),对象之间的继承(概括)关系、包含(聚集)关系,对象的封装,对象(包括方法和成员变量)在数据库中的可持久性,对象的消息驱动特性,对象的多态性等。由于ORDBMS是关系数据库技术和对象数据库技术的结合,它具有另外一些特性,面向对象的视图、触发器、权限设置等特性。总的来说,ORDBMS具有下列优点:

(1)具有RDBMS的优点:支持透明的存储路径,通过标准的SQL可完成对类型、对象视图、对象引用、表、存储过程等数据库对象的管理,提供对事务、恢复、数据完整性的支持。

(2)具有OODBMS的优点:容易表达对象间的各种复杂的关系、通过对象的封装在数据库中实现方法与数据的关联,对对象的标识、对象的多态性和覆盖性等都提供了支持。

(3)相对于OODBMS有比较高的性能,可以利用RDBMS成熟的技术及其研究成果;相对于RDBMS,它能更好地满足应用的需求。

2 ORDBMS的体系结构

2.1 概述

到目前为止,ORDBMS体系结构的实现主要有以下几种方法[4,5,9]:

(1)在RDBMS上增加一层包装层软件:用于将有关

OODBMS 的特点转化成 RDBMS 所能管理的形式，以模拟 ORDBMS 的功能。这种方法实现的 ORDBMS 的功能很大程度上受制于底层的 RDBMS，且在性能上可能会差一点。

(2)修改原有 RDBMS 的体系结构：扩充原来关系系统中不适合于 ORDBMS 的地方(比如放宽要求符合1NF 的要求，对象的标识在底层实现)，增加面向对象的特性。

(3)重新构建 ORDBMS，这种模式的工作量太大。

(4)与某一种面向对象的编程语言相结合，由该语言提供 ORDBMS 的标准类库定义，共同提供 ORDBMS 的功能。

从现有的 ORDBMS 产品来看，一般采用第二种解决方案[7,10]：继续采用 SQL 语言(SQL-3)作为数据库语言，并与原来的 RDBMS 实现完全兼容。但是这些商用的对象关系数据库仍然存在对 SQL 标准的支持度不够：从现有的几大主流 ORDBMS 来看，都没有做到对 SQL-3标准的基本支持，特别是在 SQL-3拓展的特性部分的接口，没有两个主流 ORDBMS 产品能够相互兼容。例如在 SQL-3中，数据库中过程代码的运用主要包括存储过程，触发器，以及用户定义的规则(UDR：User Defined Routines)，每个 DBMS 中都是用它自己的 SQL 方言实现：Oracle 用 PL/SQL，Sybase 和 SQL Server 用 Transact-SQL，而 DB2支持一种很接近 SQL-3的过程方言。而且这些 DBMS 编程方言中使用的灵活程度和表达式的复杂程度同高级程序设计语言相比还是相距甚远。

与此同时，Java 语言以其平台无关性而风靡世界，用 Java 来编写独立于数据库的过程代码是一个非常理想的选择，因此嵌入式 SQL 编程 SQL/J 标准也被相应地提出，并被纳入 SQL-3标准中。SQL/J 标准分为三部分[11]：

· Part 0：在 Java 程序中嵌入 SQL 语句，现已成为 SQL-99规范的第10部分 SQL/OLB。

· Part 1：用 Java 语言实现数据库中的 SQL 过程代码(Routines)。

· Part 2：用 Java 编程语言来创建数据库中 SQL Types。

虽然当前的各大数据库厂商对 SQL/J 的支持参差不齐，但是 SQL/J 已经得到各大数据库厂商的支持，同时也得到广大用户的青睐。然而 Java、SQL/J 与 DBMS 还没有达到完美结合，这主要表现在以下几个方面：

(1)编程语言与 DBMS 的 SQL 语法及数据类型不一样，这样就导致编程人员不但要学习编程语言的知识，而且要学习 SQL(甚至是学习某个 ORDBMS 的 SQL)，还要能对这两种语言中的数据类型进行转化。这显然提高了编程的难度。

(2)在引入对象的概念以后，编程语言定义的类与数据库中的类也会存在冗余、对象之间的相互对应及其管理还没有自动解决。

2.2 ORDBMS 体系结构的提出

Java 数据类型在各个平台上都是一致的，而且比较接近于一般的高级语言数据类型，另外，Java 的类库涵盖绝大多数高级语言的基本类库功能，因此，利用 Java 中的数据类型和类的定义作为数据库系统提供的默认数据和类定义，这样就可以做到数据库的类型不会因为平台、机器的异构而产生不一致性。

对于非 Java 语言，利用 CORBA IDL 将数据库中的类库和编程语言中的类库的接口信息都转化为由 CORBA IDL 定义的中间语言，这样就可以支持不同编程语言之间不同格式的类型定义和类的定义，也能很容易地实现编程工具中的类到数据库中的类的所有记录的转化，而且可以很容易地实现开发工具对数据库中对象的可视化编程，导航式编程帮助等。

利用 Java 语言平台无关性、CORBA 中的 IDL 和现有数据库中 SQL 语言，可构建图1所示的 ORDBMS 体系结构，显然既能充分地利用高级程序设计语言在编程实现上灵活度和表达式的复杂程度，又能集成 Java 语言和 ORDBMS 中对象的优点。

2.3 ORDBMS 体系结构的特点

与其它的对象关系数据库体系结构相比，该体系结构中具有以下几个特点：

1)ORDBMS 中元数据的管理　ORDBMS 的元数据(ORDBMS 默认类型、类库信息)包括 Java 类库、为 ORDBMS 添加的一些基础类库、用户自定义的类库等，其中，由 Java 实现的都可以用 Java 对象集源码经过类库元数据生成器直接生成，在对象管理器与数据库存储系统的支持下，实现对象数据的持久存储；由其他编程语言实现的必须经过编程接口和类持久器完成元数据的存储。由于 CORBA IDL 只能对各种编程语言对象接口进行转换，其它编程语言对象方法在数据库端的运行在目前还没有解决。现在能做的就是将这些语言的对象成员变量储存于数据库中，如果要调用成员方法，必须下载到客户端的程序运行地址空间中运行。

对象管理器在接收到经过优化器、对象模式转换器和事务管理器处理过的编程语言请求后，根据数据库的元数据信息，找到对应的存储在存储管理模块中的表，并根据请求在数据库服务端创建相应的 Java 对象，将存储于关系表中对应的数据取出存于对象中，最后再根据请求，控制对象成员函数运行管理器，调度对象成员函数运行。

这样做的好处是，随着语言的发展，类库的不断扩展，数据库类库也可以很容易地做到数据字典的更新，以满足新的需求。对于数据库系统的维护也提供了良好的保证。

2)编程语言与数据库的交互　编程语言与数据库的交互接口包括两个方面：

图1中左边表示数据库到编程语言转换的实现方法：通过对象转换模块将数据库对象接口转换为用 CORBA IDL 描述的中介，再在类构造器/类远程调用控制器的管理下，完成与编程语言的交互。

· 类构造器负责将序列化的类成员变量在编程语言中构造出来，类成员函数可以下载 Java 成员函数的中间码在客户端的 Java 虚拟机上运行。如果客户端编程语言是 Java，则可以在客户端程序运行的同一地址空间中创建对象，并运行该对象的对应的成员方法，如果客户端编程语言不是 Java，但能支持 CORBA，则必须通过 CORBA 调用。

· 类远程调用控制器负责将编程语言中对数据库对象的调用请求转化成远程调用，可以采用 Java RMI 技术(只针对客户端是 Java 语言)或是 CORBA 技术。

右边表示编程语言到数据库的转换方法：与传统的通过 SQL 进行交互的方式不同，在这种方式下直接经过编程语言/CORBA IDL 对象模式转换器转换，并在类持久器的协助下，直接生成相应的数据库元数据。

类持久器负责将经过 IDL 转化后的编程语言中的对象持久化；类查询器根据编程语言中的查询条件向数据库发出请求。类持久器和类查询器必须能够支持嵌套使用。类持久器和类查询器的实现方案有两种：

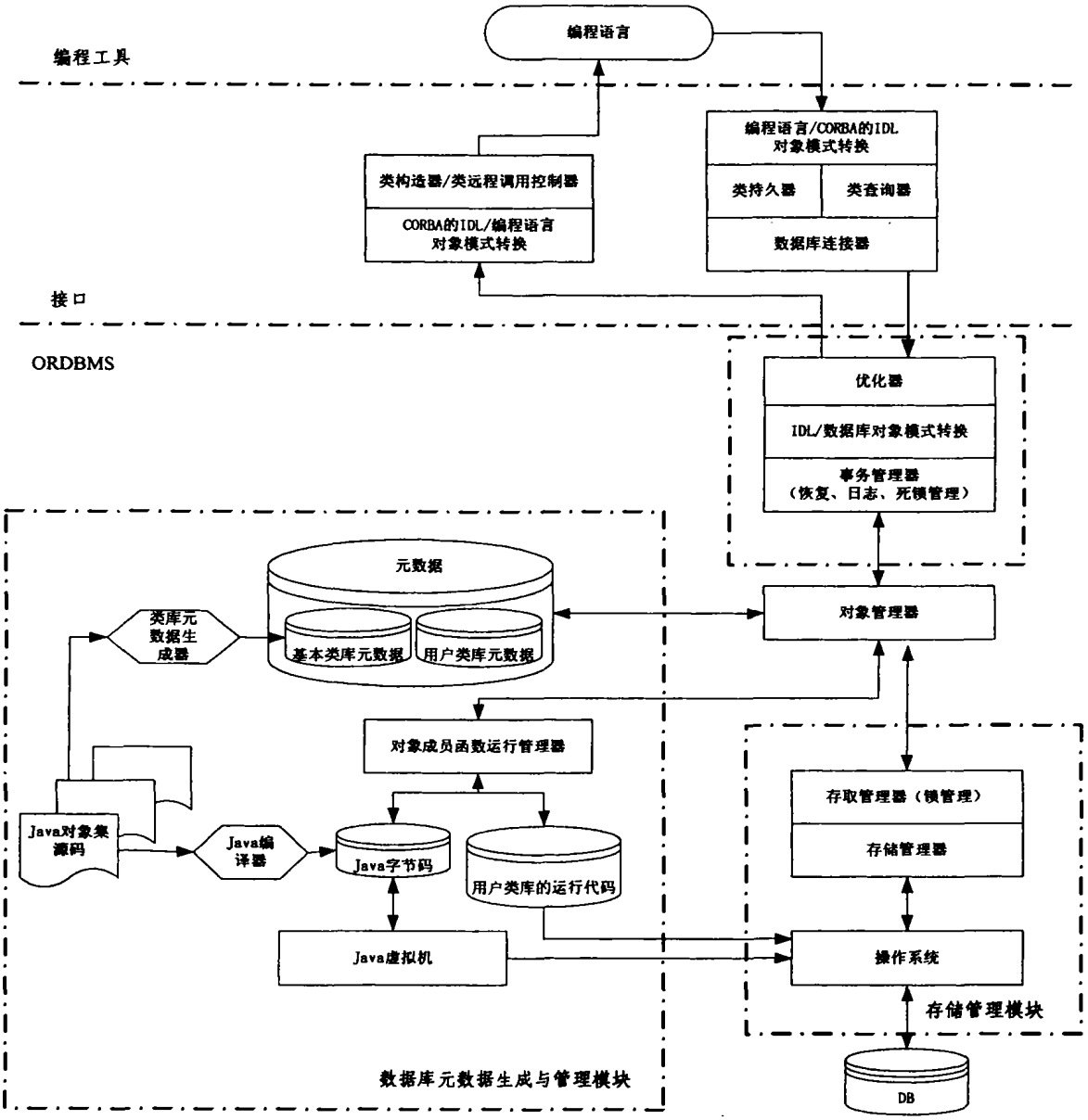

图1 ORDBMS体系结构图

·直接将请求发往ORDBMS处理，ORDBMS中的对象管理器再根据请求，控制对象成员函数运行管理器，调度对象成员函数运行。

·通过脚本语言作为中介：将请求通过类持久器和类查询器转化成对应的SQL，然后经过语法分析，再进行相关处理。这种方案的优点是能够取得很好的数据库可移植性，缺点是性能不好，语法的灵活程度受SQL的限制。

类持久器、查询器、数据库连接器可以放在数据库驱动程序中实现，并向各种编程语言提供接口，各种编程语言要对这些接口进行封装。

3）*对象数据的关系化存储* 在ORDBMS底层，所有的数据组织最终都要通过关系表来存储[4~7]。关系表与对象类之间的对应关系对编程人员是透明的。这种对应关系通过对象管理器实现，其基本方法有：

·每个类都对应一张表，对象之间的继承和包含关系通过OID引用来模拟。这样做的优点：管理简单，表与类之间的对应关系比较清楚。

·将一个类和它引用的所有类都存入一张表中：根据类的定义建立继承树放入类的数据字典中，继承树中的每个节点代表一个属性或是另一个类，而且还包含该节点代表的属性或对象占用的空间，这样就能很方便地知道类的包含类的属性在表中的相对位置。这样做系统维护较难：因为每一个类不再对应一张表，修改类的定义时，必须搜索类继承树，并修改所有子类对应表中的关于该类的信息。

3 系统实现的难点

3.1 系统性能的考虑

由于该体系结构结合了Java、CORBA，其性能需要进一步改进。在实现上可以采取一些措施：应用Java本地化技术将Java代码直接编译成在不同的平台上运行的二进制代码，采用Java的即时编译（JIT）来加速某些关键代码段，适当地修改CORBA的实现协议（例如：可以将透明的路径查找改为利用数据库连接查找；不支持许可证服务、交易服务）

（下转第81页）

出了相应的处理方法。系统的实现采用的是业界成熟的 J2EE 技术和 XML 技术，具有良好的跨平台性，并提供了提高性能的办法。用户可以根据自己的具体情况，参考本文中的模型上实现自己的系统。

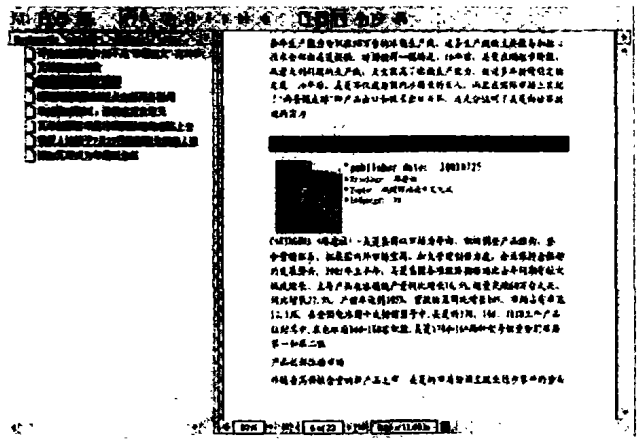

图9 PDF 输出

参 考 文 献

1 XML 规范. http://www.w3.org/xml
2 NewsML 主站点:http://www.newsml.org/
3 NewsML v1.0 规范:http://www.iptc.org/site/NewsML/specification/NewsMLv1.0.pdf
4 Apache Tomcat:http://jakarta.apache.org/tomcat
5 Sun JAXP:http://java.sun.com/xml
6 FOP 主站点:http://xml.apache.org/fop/
7 FO 规范:http://www.w3.org/TR/xsl
8 XSLT 规范:http://www.w3.org/TR/xslt
9 WML 规范:www.wapforum.org/what/technical/SPEC-WML-19990616.pdf
10 CHTML: http://www.w3.org/TR/1998/NOTE-compactHTML-19980209/

(上接第75页)

等，以提高系统性能。

3.2 编程语言和数据库对象元数据冗余的处理

根据体系结构中的元数据管理将编程语言中的类加入数据库中，如果数据库本身提供的类与编程语言提供的类功能相近时，会导致数据库中的冗余不断增加。解决的方法是在两者之间建立对应关系。这种方法需要人为地建立对象及其成员方法之间的对应关系，显然这样做的工作量很大。

3.3 对象方法的运行管理

对象方法可以运行在客户端，也可运行在服务器端。如果全部放入 ORDBMS 中运行，数据库服务器将成为运行瓶颈。如何确定方法的运行是否放在服务器上要根据下列各种因素：方法所需参数的容量，方法需要从数据库中获取的数据量的大小，网络带宽的大小，服务器/客户机的性能差距和当时的负荷情况，方法的代码量等。针对不同的情况，ORDBMS 要提供对不同的两种运行方式的支持和判断。

对象方法运行于服务器端时安全性控制是个关键，目前解决的方法有[5,9]：

(1)DBMS 进程与函数运行进程分开，函数运行进程由用户 ID 标识。这种方法的缺点是引进了远程过程调用及其相关的开销，降低了性能。

(2)单独开辟一个进程，用来检查函数代码中的所有的跳转语句，可以采用在每个跳转语句的前后插入一些检查代码，或者对于每个跳转命令，进行系统调用两种方法。由于这种方法将程序运行于 DBMS 的查询处理进程中，开销比较小。

(3)采用解释型语言如 Java 来编写函数，将边界检查的工作交给语言解释器来完成。

(4)在体系结构上采取硬件措施，缺点是不能保证所有的机器都有这样的安全机制。

展望 本文在回顾了传统数据库模型的不足和 ORDBMS 的发展后，针对 ORDBMS 的实现提出了一个新的结合 Java 的体系结构。其主要目的是：

(1)实现编程语言对象和 ORDBMS 元数据之间的透明转化，简化了应用开发的难度，提高了编程语言和 ORDBMS 中对象的集成度。

(2)系统具有很好的可扩充性：从数据库元数据的生成上看，采用类库元数据生成器，方便了数据库元数据的扩充；从接口转换上看充分利用 CORBA/J2EE 技术，便于系统的应用集成；从数据库对象的实现语言上看采用 Java 具有很好的跨平台性。

基于上述的 ORDBMS 体系结构，还可以扩展出以下的一些特性：

(1)易于与中间件集成。由于结合了 Java 和 CORBA 中的 IDL，加上 JAVA 本身对于 CORBA 的支持，就可以很容易地集成到中间件平台中。

(2)可以拓展成分布式数据库。因为数据库类库具有对 CORBA 的支持，只要解决分布的对象之间的同步更新(即一致性)和事务处理问题，就可成为一个具有良好的垂直划分分布式特性的数据库。

(3)这种体系结构的 ORDBMS 也可扩展到其他编程语言，只要它对数据库交互接口和数据库元数据导入工具支持，就可以实现与 DBMS 结合。

参 考 文 献

1 宛延悟. 工程数据库系统. 清华大学出版社，1999
2 Peter M,et al. Object-Oriented DataBases:A Semantic Data Model Appoarch. Prentice Hall,1992
3 冯玉林，黄涛，倪彬. 对象技术导论. 科学技术出版社，1998
4 Stonebraker M,Brown P. Object-Relational DBMSs:tracking the next great wave. Morgan Kaufmann Pubishers,Inc.,1999
5 鞠时光. 对象关系型数据库管理系统的开发技术. 科学出版社，2001
6 Fortier P J. SQL-3:Implementing the Object-Relational Database. McGraw-Hill,1999
7 Saracco C M. Universal Database Management:A Guide to Object/Relational Techonology. Morgan Kaufmann Pubishers,Inc.,1999
8 Date C J, Darwen H. Foundation for Future Database Systems. Addison-Wesley,2000
9 Brown P. Object Relational Database Development:A Plumber's Guide. Prentice Hall PTR; 2001
10 Oracle 用户手册. Application Developer's Guide - Object-Relational Features. Oracle Corporation,2001
11 http://www.sqlstandards.org/SC32/WG3/Progression_Documents/CD/cd1r1-olb-2001-06.txt,SQL/J 标准，2001