面向对象 多数据库 数据库

计算机科学1996Vol. 23№5

33-37

面向对象的多数据库技术

张 斌 石祥斌 郑怀远
(东北大学计算机系 沈阳 110006)

摘 要 This paper attempts to expound object-oriented techniques employed by object-oriented multidatabase systems, mainly discusses serveral key techniques in multidatabase systems, namely, distributed object-based architectures, object-oriented common data model, object-oriented view, schema translation and schema integration of schema integrated process.

关键词 Multidatabase systems, Object-oriented multidatabase systems, Object-oriented techniques, Schema translation, Schema integration, Object-oriented view.

现代的计算环境大多数都包含分布、异构和自治的软件和硬件系统,迫切需要对这些软件和硬件所提供的服务进行协调的技术,多数据库系统的目标是协调使用由分布、异构和自治的数据库系统提供的信息和服务。一个多数据库系统(MDBS)[1]是一个已存在的、具有自治性和异构性的数据库系统的邦联,它所面临的最大挑战之一是解决成员系统的异构性问题,这种异构性主要表现在三个层次中:平台级、数据模型级和数据模式级。目前主要存在两种类型的多数据库系统:全局模式多数据库系统和多数据库语言系统,而前者主要包括带全局模式的多数据库系统和联邦数据数据库系统两种类型。

随着数据应用领域的不断复杂化,传统多数据库系统已不能满足要求,近来,许多研究者已经在应用面向对象技术到多数据库系统方面做出了努力[4~8],且已建立了一些实用的或实验的多数据库系统[4,10~12]。这些研究对 MDBS 设计和实现的各个领域产生了如下影响:(1)引入了分布式基于对象的体系结构、改变了传统的 MDBS 的体系结构;(2)使用面向对象数据模型作为公共数据模型,产生了新的模式结构、模式转换和模式集成方法;(3)面向对象事务管理技术影响了传统的多数据库事务管理。下面将详细论述其中几项关键技术。

1 分布式基于对象体系结构

文[8][9]提出了一个称为基于对象体系结构的分布式系统体系结构,它将一个分布式异构系统模型化为一个分布式相互作用的对象的集合,这些对象表示分布式系统资源,即:各种成员系统的资源被模型化为对象,而成员系统提供的服务被模型化为对象的方法,方法组成对象的接口。这样,每个成员系统定义一个服务接口并为这些服务提供实现支持,客户通过以公共语言表示的请求与异构系统交互;分布式对象管理器负责转换客户请求到可用的服务、传递请求到适当的系统、提供以公共语言表示的应答给客户。

将成员系统对象化满足了异构性和自治性需求,首先,由于发送给分布的成员系统的消息仅取决于其接口,而与其方法的实现无关,成员系统间的各种差异被封装在对象的实现中;再者,支持了异构性;由于成员系统的使用是独立的和透明的,因而满足了自治性。

对象管理组(OMG)[7]提出了一个如图1所示的对象管理体系结构(OMA),其中,对象之间的通讯借助于对象请求代理(ORB)进行,ORB 提供对象透明地提出请求和接受应答的机制。

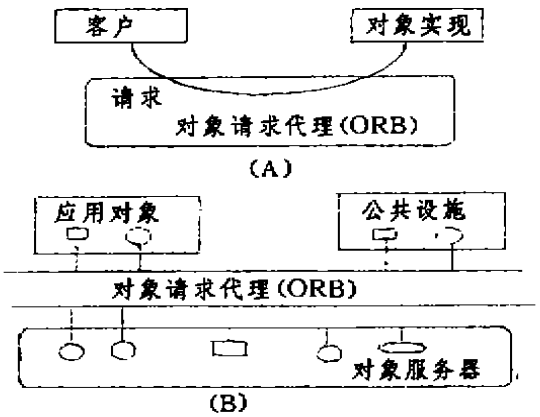

图1 (A)通过对象请求代理发送的一个请求
(B)对象管理体系结构(OMA)

在多数据库系统中,成员系统是数据库系统,因此面向对象多数据库系统的体系结构是一般的分布式基于对象体系结构的特例,对象表示的分布式系

统资源是存储在数据库中的信息；而对象的方法表示的服务是检索和修改这些信息的有效方法，应该支持如持久性、查询、事务、并发共享等数据库功能。

2 面向对象公共数据模型

为解决模型异构性，集成多数据模型的系统必须提供一个模型中的概念与另一个模型中的概念的映射。最常用的方法是提供一个公共数据模型(CDM)，每个成员模型被映射到CDM上。一般认为，能够刻画集成任务的所有步骤的数据模型应该是语义丰富的，应该提供一个机制，不仅能够表示成员数据库能够表示的语义，而且还能够表示与集成过程相关的附加语义；理想地，应该有能力在可能的将来扩展中表示被加入到系统中的新的成员系统的语义。从这一角度来看，采用面向对象公共数据模型(OOCDM)是合适的。

使用OO模型作为CDM有下列的优点：(1)有丰富的语义，支持复杂类型和抽象机制，提供传统数据模型所不能表示基本构造符之间的关系；(2)允许通过方法的概念来捕捉对象的行为；(3)使得通过行为映射来集成非传统数据库成为可能；(4)由于数据的实际存储和检索是由成员系统支持的，因此不会因为在CDM上支持对象而产生较大的性能损失；(5)元类机制增加模型的灵活性，因为它允许模型本身的任意提炼(例如：增加新的关系)。

但是一般的OO模型缺少某些必要的解决多数据库系统特殊问题的概念和机制，需要对作为OOCDM的OO模型进行扩充，扩充应包括如下几方面：

1)语义扩充：一个用于CDM的OO模型应该支持某些附加的语义关系，通过基本模型的元类机制以捕捉成员模式中的及其间的语义。所增加的语义关系一般有两种：已存在语义关系的特化和与其它数据模型(如关系模型)所支持的语义关系的对应语义关系，后一种情况的一个典型例子是Part-Of关系，基本OO模型仅能够表示相关对象的集合(聚集)，允许一个对象可以有其它对象作为其实例变量，但是它不能表示对象间的依赖关系，因为一个对象不能够占有其实例变量的值，而只能简单地保持对它们的引用。为此可以定义组合对象(Composite Object)是一个具有互斥的成员对象层次的对象[6]，增加依赖的表示。这些成员对象是依赖对象，其存在依赖于对应的构成对象的存在，因为它们是其构成对象的一部分。

2)状态和行为：大多数作为CDM的OO模型都不区分对象的状态和行为，而是使用相同的结构(一般称作函数)来表示实例变量和方法，一个实例变量由Get和Set函数来表示，Set函数将一个值赋予变量，而Get函数返回变量的值。这种方法将使模型具有更少的结构，从而使可能的结构冲突的数量达到最少。更重要的是，通过允许一个对象的状态在全局模式中被提炼，可极大地增加集成的灵活性。如：考虑类"Employee"的一个对象，在不同成员模式中，Employee. Salary以美元或英镑为单位；在全局模式中，Employee. Salary以马克为单位。这样，若Employee. Salary被表示为一个函数，就可以在全局模式中定义一个适当的函数来实现必要的转换。

3)向上继承：在基本OO模型中，仅提供了子类构造符，实现了从超类到子类的继承定义，而没有提供超类构造符。由于在多数据库系统中，模式集成是从已有的成员模式集成产生一个全局模式，因此既需要从超类出发定义子类，又需要从子类出发定义超类，这样就需要某种机制来定义从子类到超类的继承，称其为概括或向上继承。

3 OOMDBS中的模式转换

模式转换主要解决模型异构问题。模式转换一般是一对一的，例如：当目标模式(模式B)以OO模型表示，而源模式(模式A)以关系模型表示时，关系映射到类；元组映射到对象；模式A中两个关系间的包含关系用以决定模式B中对应类之间的语义关系(如：子类/超类关系)。模式转换产生了使模式B中的模式结构相应地映射到模式A中的模式结构，以后的命令转换将使用这些映射把包含模式B中的模式结构的命令转换成包含模式A中的模式结构的命令。

在OOMDBS中，主要有两种映射方法：结构映射和操作映射。前者定义不同模式间数据元素之间的对应，一旦结构映射被定义，所有集成应用都将使用这一定义；而操作映射定义不同模式间操作的映射，即在源模式中抽取一些原始操作，以此定义目标模式中的一些称为抽象操作的基本操作，以后目标模式中的所有操作均使用这些抽象操作来实现。

这两种方法的基本区别如图2所示。图2(A)表明，为了使用结构映射，集成视图被定义为数据元素的集合；而结构映射被定义为集成视图中元素与局部系统中元素的对应的集合。一旦结构映射被定义，集成系统支持由集成应用提出的操作到局部系统的原始操作的自动转换。图2(B)表明使用操作映射的相应情况，集成视图被定义为抽象对象上的抽象操作的集合；而操作映射被定义为这些抽象操作在局部系统的原始操作上的实现。抽象操作应该是尽可能小的基本抽象操作的集合，而复杂的抽象操作由集成系统在这个基本抽象操作的基础上自动提供。一旦操作映射被定义，集成系统转换由集成应用提出的操作为基本抽象操作，然后触发基本抽象操作的实现。操作映射的实现以OO技术为基础，它需要使用OOCDM才能定义集成视图为操作的集合，因为只有OO模型提供方法和封装机制。

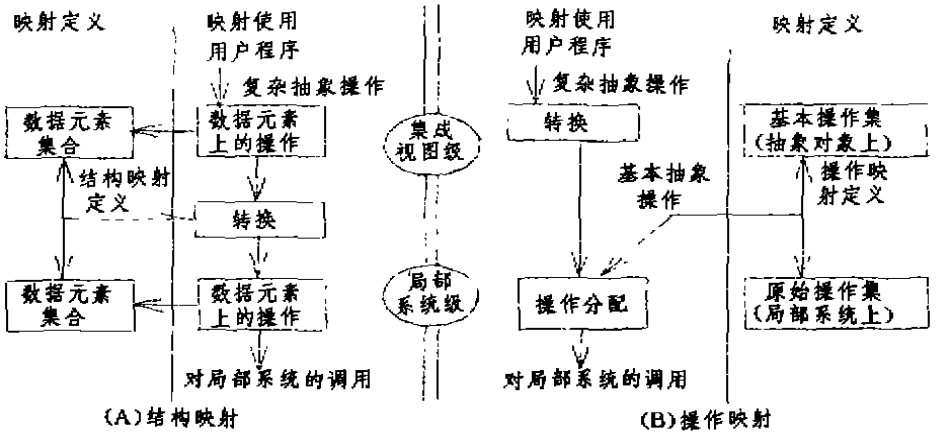

图2 结构映射和操作映射比较

4 面向对象的视图集成机制

大多数 OOMDBS 系统的模式集成都采用 OO 视图模型来定义全局模式(集成模式)。视图是在一个或多个已存在的数据库上定义虚拟数据库的一种途径,视图不被存储,而是在对其进行查询时被重新计算出来。视图的定义依赖于数据模型和用于描述视图的语言。在关系数据模型中,视图的定义是基于查询的,在 OO 数据模型中,有许多不同的方法来定义 OO 视图。一般来说,OO 视图被定义为一个虚类的集合。这些虚类由已存在的对象构造出的虚对象来说明,定义了一个模式,而说明它们的对象构成了基于该模式的(虚)数据库。一旦虚类被定义,它们将同任何其它类一样被使用,在虚类定义中被使用到的类称为该虚类的基类。

OO 模型虽然在构造新对象方面是语义丰富的,但不支持分组已存在对象的机制,这需要在视图模型中解决。加入到 OO 系统中的最主要的视图功能有:①从成员数据库向视图输入类,这种虚类直接对应于存在的类;②定义一个称为派生类的新类,这种虚类不直接对应于存在的类。

(1)类的输入:一个视图能够通过类的输入语句从其它数据库中得到数据。一旦某个类被输入,其定义和实例在视图中就变成可见的了。通过显式的隐藏语句或通过输入语句的可见子句来实现一部分输入数据是否被隐藏,类输入机制在一个类的输入是否导致其所有超类的输入方面是不同的。有些类输入机制在一条输入语句中仅输入类,而有些类输入机制则一条输入语句中输入整个类层次。有些类输入机制区别行为方法的输入和对象的输入,这样,局部方法能够在被输入的对象上执行,而被输入的方法能够在局部对象上执行。

(2)类的派生:一个派生类的建立需要指定其三个成员:①派生类的初始成员(外延);②派生类的结构和行为(内涵);③派生类与其它派生类之间的关系。

不同的 OO 视图机制在指定派生类的成员方面有不同的方式,一般是直接定义其中的一个成员,而由系统来派生出其余的两个成员。主要有下列几种生成派生类的方法:

·对应于 OO 模型支持的类之间的语义关系,提供各种类构造符,应用到已存在的基类上,生成与基类有对应关系的新的派生类[1,2]。主要构造符有:Generalization 或 Superclass、Specialization 或 Subclass 等。这种方法是显式定义派生类的第三个成员,而隐含地由系统生成其余两个成员。被定义为子类的派生类继承其超类的所有函数,在子类中可以提炼超类的函数及定义新的函数。子类的外延通常被定义为其超类外延的交集。被定义为超类的派生类继承其子类的公共函数(称为向上继承),在超类中可以提炼子类的函数,超类的外延通常被定义为其子类外延的并集。

·将派生类定义为已存在的基类上的查询结果。派生类的内函(结构和行为)和它在类继承层次中的位置可以由系统自动产生。这一方法包括通过虚对象的说明来定义类的机制。定义一个派生类中的函数能够使用基类中定义的函数。文[3]提出了一个完整的方法来产生派生类在类继承层次中的位置,该方法是通过基类上的查询谓词来定义派生类是基类的子类,而包括基类的派生类被隐含定义为基类的超类。

·派生类的结构和行为被显式定义。

·上面方法的结合，特别是使用包含有类构造符的查询语言。

表1 冲突分类

冲突类型			说 明	解决策略
对象标识冲突			含义相同的对象具有不同的 OID 值	提供一个称为 Same_OID(OID1,OID2) 的函数定义机制，以指定对象间的等价关系，使等价的对象被当做相同的对象处理。
模式冲突	命名冲突	同义词冲突	含义相同的元素具有不同的名字，有类、属性和方法命名同义词冲突	对发生同义词冲突的元素进行换名
		同形异义词冲突	含义不同的元素具有相同的名字，有类、属性和方法同形异义词冲突	对发生同形异义词冲突的元素进行换名
	模式结构冲突	类结构冲突	来自不同成员模式的类在类结构定义上发生的冲突	分别改变 C1和 C2的结构，即在 C1和 C2中补充丢失的属性和去掉多余的属性。
		属性冲突 — 属性结构冲突	两个相关类中属性结构定义发生冲突	两个类中补充丢失属性或在两个类中去掉丢失属性
		属性冲突 — 属性类型冲突	含义相同的类中含义相同的属性具有不同的属性类型	O_O 多数据库系统中属性类型是非常丰富的，其冲突情况很复杂，其中包含有聚合层次的冲突和类型构造符间的冲突，冲突的解决应根据属性类型的不同情况，采取不同的策略
		属性冲突 — 属性长度冲突	含义相同的类中相同含义的属性有不同的数据长度	将发生冲突的属性对中的属性长度统一定义为最大者。
		类与属性冲突	当等价的特征在一个成员模式中表示为类的属性(集)，而在另一个成员模式中表示为一个独立的类时，就发生了类与属性冲突。	首先将用属性(集)表示的等价特征聚合为一个新类，在原类与新类之间形成一个引用关系，将类与属性冲突转化为类冲突，然后在将新类和发生类与属性冲突的类之间的冲突通过类冲突来加以解决。
	行为冲突	方法调用参数个数冲突	含义相同的方法具有不同的调用参数个数	重新定义一个全局统一的方法
		方法调用参数类型冲突	含义相同方法的某些调用参数类型具有不同的数据类型	重新定义一个全局统一的方法
		方法返回参数类型冲突	相同的方法具有不同的返回参数类型	重新定义一个全局统一的方法
		方法体定义冲突	含义相同的几个方法其实现代码的定义不是等价的	重新定义这两个方法的方法体
	对象实例约束冲突		来自不同成员模式的对应类在对象实例约束上发生的冲突	在集成模式中将两个对象实例约束统一，对对象建立方法进行提炼。
语义冲突	继承关系冲突		成员模式中的某些对应类之间有不一致的继承关系	将发生冲突的继承语义在全局模式中统一
	数据语义冲突	量纲冲突	相关对象实例中等价的数据元素有不同量纲表示的数据值	在全局数据库中将发生量纲冲突的数据项定义为适当的量纲。
		精度冲突	相关对象实例中等价的数据元素有不同精度表示的数据值	在全局数据库中将发生精度冲突的数据项定义为最高精度
数据冲突	不兼容对象实例冲突		含义相同的数据项在不同的数据库中有含义不同的数据值	一种方法是将数据的最近修改作为冲突的数据项的值
	数据表示冲突		含义相同数据项在不同数据库中有含义相同但表示符号不同的数据值	定义相关的转换函数，将表示同数据项的多种表达，在全局数据库中进行统一。

5 OOMDBS中的模式集成

模式集成是集成若干个已存在模式到一个统一的集成模式中的过程[2]。文[2]标识了模式集成的四个主要步骤：预集成、模式分析、模式一致化、模式合并与重构。(1)预集成阶段主要要确定集成规则、选择要集成的模式、集成优先次序、集成策略等。模式转换被认为是预集成的一部分。(2)模式分析阶段分析并比较要集成的模式。确定各种可能的冲突，标识模式间的语义。模式对象的比较是基于语义而不是语法的。(3)模式一致化阶段主要是解决在模式分析阶段确立的各种冲突，使成员模式达到一致以便以后的集成。(4)模式合并与重构阶段是要建立最后的集成模式，模式合并与重构的原则是：①完备与正确性。集成模式必须完全正确地包含出现在所有被集成模式中的所有概念；②最小性。如果同一概念出现在不同的成员模式中，则这一概念在集成模式中仅能出现一次；③易理解性。集成模式对于设计者和用户来说应该是易于理解的。

OO模式集成与传统模式集成相比将更加复杂，其中主要问题有：

·冲突问题：是模式集成的关键问题之一。为了保证集成模式的最小性，出现在不同的成员模式中的同一概念应该被合并。但是，在成员系统中存在着成员模式的结构和语义的不一致，将产生各种各样的冲突。由于OO模型能表示复杂结构和丰富的语义，其中的冲突问题将更加复杂和难于解决。表1列出了各种可能的冲突分类及相应的解决策略。

·模式间关系：为实现集成，不仅需要标识公共概念集合(解决冲突问题并实现模式合并)，而且还需要标识在不同模式中不同概念的集合(这些概念通过某些语义性质而彼此相关，称为模式间性质—Inter-Schema Properties)，它们保持一个模式中的对象集合与另一个模式中的对象集合间的语义关系。为了保证集成模式的完备与正确性，这些关系必须在集成模式中被表示出来。目前还没有一个系统方式来表示OO集成模式中模式间的关系[1]，最常见的是直接对应于OO模型所支持的语义关系的模式间关系，主要有：聚合、特化、概括和任意(Arbitrary)关系。

·虚对象的对象标识问题：在OO系统中，每个对象在其建立时被赋予一个唯一的对象标识符。因此，一个虚对象被建立时也必须被赋予一个唯一的对象标识符，并且一个虚对象每次建立都应被赋予相同的对象标识符；更复杂的是，若一个虚对象是一个组合对象(见第2节)时，其对象标识符应该随实际对象的修改而改变。通常的解决办法是将虚对象的对象标识符定义为它所依赖的实际对象的对象标识符的函数。

结论 OO模型为异构环境提供了一个自然模型，将成员系统的资源和服务模型化为对象和方法，解决了成员系统实现的异构性并保证了它们的自治性。在多数据库系统中使用OO技术能够表示异构成员系统的实体间的各种关系，解决它们之间的各种冲突。OO技术为解决多数据库系统的各种问题提供了一个很有前途的方向。

参考文献

[1]Evaggelia, P. 等, Object Orientation in Multidatabase Systems, ACM Computing Surveys, 27(2)1995

[2] Batini C. 等, Comparision of Methodologies for Database Schems Integration, ACM Compu. Surv., 18(4)1986

[3]Abiteboul S. 等, Objects and Views, In Proc. of the ACM SIGMOD ACM Press, New York, 1991

[4]Kaul M. 等, Viewsystem: Integrating Heterogeneous Information Bases by Object-oriented Views. In IEEE Intl. Conf. on Data Engineering 1991

[5]Sheth, A. P. 等, On Automatic Reasoning for Schema Integration, Int. J. Intell. Cooperative Inf. Syst., 2(1) 1993

[6]Gagliardi R. 等, An Operational Approach to the Integration of Distributed Heterogenous Environments, In Proc. of the PARBASE-90 Conf. 1990

[7]Soley R. M., Using Object Technology to Integrate Distributed Application, In Enterprise Integration Modeling. Proc. of the First International Conference, MIT Press, 1992

[8]Manola F. 等, An approach to Interoperable Object Models. In Proc. of the Intl. Workshop on Distributed Object Management 1992

[9]Nicol J. R. 等, Object Orientation in Heterogeneous Distributed Computing Systems, IEEE Computer 26 (6)1993

[10]Ahmed R. 等, Pegasus: A System for Seamless Integration of Heterogeneous Sources. In COMPCON'91

[11]Bertino E. 等, Integration in Heterogeneous Database Applications Through an Object-oriented Interface, Inf. Syst. 14(5)1989

[12]Bukhres O. A. 等, Object-oriented Multidatabases: Systems and Research Overview. In Proc. of the Intl. Conf. on Information and Knowledge Management 1992