基于类网络的OODB事务处理模型*)

李伟华　胡守仁　（国防科技大学计算机系）

摘　要

In the database field, transaction processing has long been recognized commonly as not only an advanced,but also a difficult topic. The transaction models for existing object-oriented databases(OODBs) are almost all extensions to, or borrowed from traditional transaction models. They captures only a few characteristics of OODBs. Our Contributions in this paper are three aspects. First, we take a new look at the relationship between database and its transaction processing. A database is viewed as an object's structure, and its transaction processing system as the method of this object.Second, we present a transaction model for an OODB——GOODbase, which is being developed by us on SUN workstation. The idea of this model can be applied generally to other OODBs. Third, we distinguish two kinds of concurrency in our model, namely, entra- and intra-object parallelisms. Some implementation issues are also considerated here.

1、引　言

当前，OODB已成为数据库领域的热门话题，并从实验上和理论上获得了高度重视[1,2,4,8,10,14,16-20]。但是，关于这种系统的定义还存在着不少争议，有三点可以概括OODB初期研究阶段的状况：(1)缺少形式化的定义，(2)没有共同的数据模型，(3)大量的实验性活动。这些问题同样存在于OODB事务处理的研究之中[1,4,14]。

事务处理机制是数据库系统中广泛采用的控制程序并发执行的手段，通常，事务被看作数据库并发和恢复的基本单位。OODB比关系数据库（RDB）具有更丰富的语义和更强的数据模拟能力，自然会包含许多可能影响我们对事务认识的新特性。

1.1　面向对象数据模型概要

GOODbase[16,18,19]使用的数据模型可以用若干概念来表征，这些概念主要借自于Smalltalk-80[5]，ORION[2,4,8]和〔1〕。

所有概念上的实体都模拟成对象，每个对象都有一个唯一性标识符，对象的状态存储在它的属性（或称实例变量）中，属性的值本身也是一个对象，对象的行为封装在它的方法中，所以，对象可以用来描述数据的结构和行为。由IS-PART-OF关系所联系的一组对象统称为一个组合对象。对象之间的通信是通过发送消息来实现的，处理一个消息就是执行相应的方法。

固定的模式和确定的范围。

4.CASE库是CASE的核心。目前技术发展的趋势是所有的软件工具只要能与CASE库相连，就成其为CASE工具的范畴。CASE库的发展不单纯是数据或实体集成的执行者，而是整个开发环境中的驱动和控制力量[5]。（参考文献共19篇略）

*)863高技术及国家自然科学基金资助项目

为概念上和管理上的简单起见，相似的对象群集在一个类中，属于一个类的所有对象都是该类的实例；对象是其所属类的实例集合中的一个成员，类描述了其实例的属性以及可以施加到这些实例之上的方法。类层次是由类构成的一个层次结构，其中的节点表示类，节点之间的边表示IS-A关系，高层的类称为超类，低层的类叫作子类。超类的属性和方法可被其所有子类继承，子类可以重新定义继承的属性和方法。一个类有两个以上的直接超类时就称为多继承，这时，类层次就泛化为类网络。

1.2 OODB与RDB的不同之处

OODB比RDB具有更强的数据模拟能力，我们认为，OODB是RDB的外延，RDB是OODB的内涵，用一个OODB来模拟一个RDB是很容易的，反之则很困难。OODB与RDB的差别主要体现在：OODB是一个面向对象程序设计系统与一个数据库系统的集成系统，而RDB是一个纯数据库系统[1,2,8,17]；OODB包括了强类型、抽象数据类型、过程和继承等概念，而RDB仅有关系和元组作为类型；OODB支持类型的扩展，程序员可以随意建立新类型，不只限于关系，而RDB却有一堵无形的墙，使得程序员不能使用关系以下级别的类型。OODB可以在数据库而不是程序设计语言一级进行类型检查，操作数据的消息可以直接送给数据库中的数据，而不是先将数据移到程序设计语言的存储空间，这就使得程序更为有效，从概念上来看更简单。OODB也可以看成是程序设计环境中的一部分，它与该环境中的其它部分动态集成，OODB比RDB更适合于存储程序、对象和类型。在一个系统中，OODB可能包含对多种事务模型的性质的支持，这对RDB来说却是一件相当困难的事情，但是，在OODB的研究者中，很少有人注意到了这一点，本文其余部分将主要讨论这一问题。

2. 相关工作的评述

许多研究者已经强调了本文涉及的事务模型问题，这些工作大致可分为四类：传统的原子事务模型[3,6]，协作事务模型[9,12]，嵌套事务模型[7,11]，以及基于抽象数据类型的事务模型[13,15]。对于每一类事务模型，我们先略述其代表性的工作，随后说明这一领域需要进一步工作的原因。

传统的原子事务模型是一个平板模型，它可以用四个特征来刻画：即原子性、一致性、分离执行和永久性[3]。该模型奠定了事务处理的基础，但对许多新型应用领域而言，它的限制过于严格。

在嵌套事务模型[11]中，事务分解成若干子事务，子事务能使事务中的故障局部化，可开发事务内部的并行性。子事务又可以进一步分解成其它的子事务，这样，事务将以层次方式扩展。相对于父事务和同胞事务而言，子事务的执行具有原子性，可以独立地夭折而不会引起整个事务的夭折。但是，如果父事务夭折，它的所有子事务都会夭折。只有当所有的子事务都终止时，父事务才能提交。子事务可以访问当前由其父事务所访问的任何对象，此外，数据库的任何对象也能被子事务访问。

协作事务模型[9,12]是在CAD/CAM、CASE和设计应用的背景下提出的，由检入/检出数据访问模型支持。在这种模型中，事务也分解成子事务，每个子事务都具有自己的语义和类型。该模型支持三种不同类型的子事务：项目事务分割成协作事务；协作事务分割成一组承包者事务；承包者事务要么具有与协作事务相似的结构而成为一个委托者协作事务，并能作为一个局部项目事务进一步分解，要么具有原子事务的结构。协作事务具有类似嵌套事务的层次结构，但是不支持对象访问的继承关系。

上面三种模型都没有利用抽象数据类型的语义，这样就不能开发对象内部的并行性。同样，事务分解层次也没有紧密对应于

OODB的类网络结构，事务层次的组织不够自然。

在基于抽象数据类型的事务模型[13 15]中，冲突关系一般定义在类型的操作上——这里的类型与OODB中的类相当，操作与类的方法相当，如果OP_1与OP_2不能并发执行的话，则说操作对（OP_1，OP_2）在类型T的冲突关系R中。一般地说，如果OP_1OP_2的结果不同于OP_2OP_1的结果，就一定存在操作冲突。控制机制是非集中式的，因为只需考虑已在对象x上执行的操作的信息，就可决定另一个操作能否进行。这种模型很适合于开发对象内部的并行性，但它没能利用类网络或者说任务划分层次的优点，不适于需要开发宏观并行性的设计应用。

上述模型都是为特定的应用领域开发的，其目的不在于像OODB所针对的混合环境。尽管可对它们加以改造而用于OODB事务处理，但就单个模型而言都不能满足OODB应用领域的大部要求。

3. 基于类网络的事务模型

3.1 我们关于OODB事务处理的观点

在OODB范型中，是否存在某些东西会改变我们对事务处理的看法，这在OODB的研究者中有较大的分歧，一时难以达成共识。但我们认为，不同的数据操作特征需要颇为不同的事务控制机制，这种不同性大约有一半表现在模型级，另一半则表现在实现级。OODB包含了诸如数据抽象和类网络等新特性，它们提供了增加事务处理并行性级别的其它语义。实际上，当系统根据用户定义的类型和操作获得更多应用一级的语义后，比起把所有操作仅仅看作读写操作而言，我们显然可以利用这些信息来达到更大的并发性。

在OODB中，既然所有概念上的实体都被模拟成对象，我们为何不把整个数据库系统看作一个对象呢？！这是一个虚拟对象，

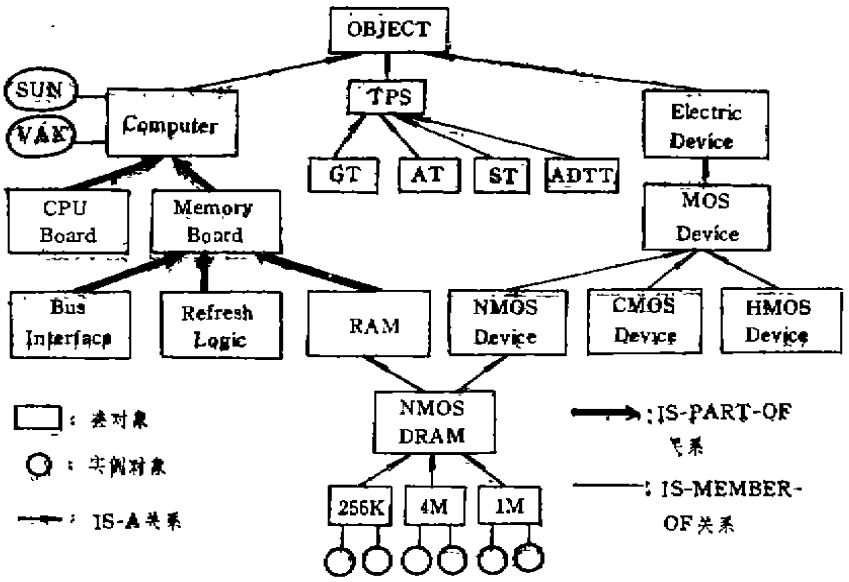


图1 一个可能的OODB的类网络

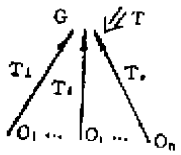
数据库作为该对象的结构，而运行于其上的程序（可能改变数据库的状态）作为该对象的方法，这些程序被看作是根事务。换句话说，数据库系统是单个的抽象数据类型，数据库是它的状态而事务是它的操作。事务处理系统（TPS）也可以建成一个真正的对象，它是一个类对象。假定OBJECT是整个系统的根对象，图1则给出了一个可能的

OODB的类网络。

TPS处理根事务，其子类GT、AT、ST和ADTT分别对应不同的事务模型，这样，我们就有了一个集成式事务处理系统。我们可以把各种具体的实现作为它们进一步的子类，即是说，在同一个系统中，某一事务模型可以有几个实现方案。但是，我们不允许事务类层次中的节点具有可以永久存放在数据库中的静态实例对象，它们可以有动态实例对象，以作为构造动态事务处理层次（DTPH）的构件。DTPH在对应的应用程序终止后消失。

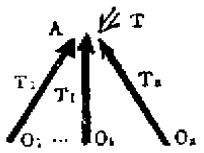
3.2 模型的静态特征

正如我们在图1中所看到的，在OODB中存在三种形式的抽象，即泛化（IS-A关系）、汇聚（IS-PART-OF关系）、相联（IS-MEMBER-OF关系）。OODB的组织就是由这三种关系来表征的，我们的事务模型的静态特征主要也是基于这三种关系。图1中以TPS为根的事务类层次中的GT、S-T、AT节点分别对应上述抽象，它们都称为T节点，图2给出了上述三种事务管理形式的图表示。



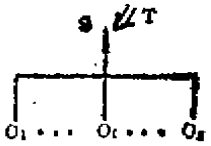
G是O_i的泛化对象，j=1，…，n，T定义为：

Case O_j of
O_1: T_1 on O_1
⋮
O_i: T_i on O_i
⋮
O_n: T_n on O_n



A是O_j的聚合对象，j=1，…，n，T定义为：

T_1 on O_1 and
⋮
T_i on O_i and
⋮
T_n on O_n.



S是同一个类的所有实例对象的集合，T定义为：

for each O_j in S do
T on O_j
j=1，…，n

图2 三种事务管理形式的图表示

其中T_i是由GT或AT根据某些类描述信息而创建的子事务。创建一个动态事务对象时要赋一个事务标识符、检验锁相容性、在所访问的对象上置锁、维持一个修改表等等。对每一个事务对象都要进行这样的工作。

我们还在模型中开发了对象内部的并行性，这一想法主要是受〔13，15〕的启迪。因为对象具有唯一性标识符，除非两个操作之间具有操作冲突关系，那么，不同的事务可以在同一个对象的不同（甚至相同）的属性（包括对象名）上激活不同（或者相同）的操作，这一工作由ADTT来完成。有关详细思想参见〔13，15〕。

综上所述，我们知道，该模型不仅能开发事务和对象之间的粗粒度并行性，也能开发事务和对象内部的细粒度并行性，对于绝大多数应用环境而言，它能够满足OODB的事务处理的要求。

3.3 模型的动态特征

前面提到，对于特定的应用程序（或设计过程），TPS层次将创建相应的动态事务处理层次，这种DTPH是系统中某一子类网络的映象。为了直观起见，我们将用一个例子来说明DTPH的创建过程。

OODB的开发包括设计类网络和创建数据库中的实例对象，由于篇幅限制，这里只说明创建实例对象时DTPH的生成。

假如我们要设计一台计算机，比方说SUN（图1），这是一个设计项目的例子，可能有若干人共同工作，每人负责一特定的任务，也可能有一组人对同一子任务进行工作。

当设计进行时，组内成员自然可能互相通信，或者是非形式化的，或者是通过数据库中的某个对象。我们从例中看到，每个设计者都负责一组与自己任务有关的对数据库的改变，这些改变可能影响其它设计者的工作。

设计过程通常以层次方式分解，如计算机的设计过程分解成CPU插件板设计和存储器插件板设计，后者又进一步分解为总线接口设计、刷新逻辑设计和RAM设计。每一个子任务都可能以一种或多种方式被约束，如RAM阵列必须放置在为它分配的空间内。叶节点（如256K、1M和4M）表示与单个设计者相联系的事务，内部节点（如Bus Interface、Refresh Logic和RAM）称作事务组，它们协作完成一个共同的任务（如Memory Board），树的根就是管理计算机设计的事务。

设计一台SUN计算机的DTPH的生成示于图3，可以用下面几步加以解释。我们将图1中的对象的名字加上SUN-作前缀来命名与相应子任务对应的事务。

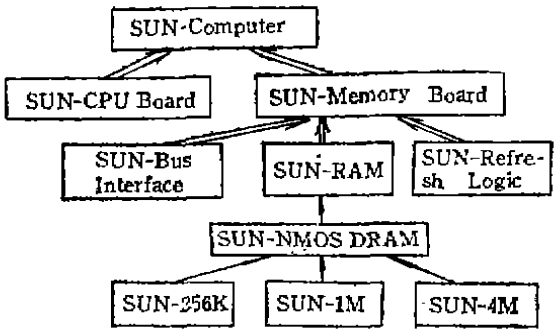


图3 动态事务处理层次的构造

（1） 项目负责人发送一个带有特定参数（比如说Computer等）的消息给TPS，然后TPS生成一个名为SUN-Computer的根事务对象。

（2） 由于Computer是一个汇聚对象，SUN-Computer就发送一个带有相应参数的消息给AT，AT就生成两个子事务对象SUN-CPU Board和SUN-Memory Board，作为CPU插件板和存储器插件板的设计任务。

（3） 用同样的方式继续上面的过程。

…

（4） SUN-NMOS DRAM发送一个带参数（如1M）的消息给GT（因为NMOS DRAM是一个泛化对象），然后GT就产生一个名为SUN-1M的事务对象。（也许还需要256K、4M的芯片）

（5） 最后，SUN-1M发送一个消息给ST，生成一个事务去选择（或设计）给定数目的存储器芯片。

当整个设计过程成功地结束后，DTPH就被删除，一个新的实例对象SUN就加到Computer类的实例集合中。如果设计过程没有成功地结束，DTPH也被删除，由子事务创造的实例对象可以成为、也可以不成为永久性实例对象，这取决于某些特殊的选择。

4. 实现考虑

基于类网络的事务模型比其它的事务模型更加复杂，这样，在实现时应该考虑某些特殊问题。

4.1 事务的描述

我们认为在描述事务时应该考虑下面的信息：

（1） 为事务赋一个唯一性标识符——这是为了区分不同的事务。

（2） 所涉及的对象类型——这是事务中操作所作用的基本数据类型，有可能影响事务的进行一步分解。

（3） 不同事务之间的关系——这是为了允许事务之间可能发生的相互影响和通信，并允许事务之间可能的并行性。

（4） 操作访问的逻辑——说明操作唤醒的语义，包括前件和后件。

（5） 操作的顺序——这是为了给出访问路径的信息。

（6） 在同一对象上不同操作之间的关系——这是为了允许对象内部的并行性。

上述信息并非都得在同一级别进行描

（转封底）

(接第59页)

述，对一个事务而言，也不一定非得描述上述所有的信息。最后一个信息就可以在抽象数据类型中加以描述。

4.2 事务对象集合

TPS 层次中的每一个节点都有一个事务对象集合（TOS），TOS也是一个系统类（即SET）的实例对象。TOS用于管理由对应的T节点产生的所有事务对象，所有的根事务对象都放在TPS节点的TOS中。因为不同的T节点产生的事务对象可能会有不同的并发控制和恢复机制，引入TOS可以简化对事务对象的描述和管理，并有利于事务的并发控制和恢复。

4.3 封锁模式

我们的模型允许有比其它模型更多的并发性，为了开发这些潜在的并发性，就不应该在对象上设立任何不必要的限制。换句话说，我们提供了更多的封锁模式，而不仅仅局限于严格的读／写封锁模式，以达到尽可能大的并行性。我们为基于类网络的事务模型选用了预约封锁的思想，详细情况将另文介绍。

5. 结束语

在本文中，我们从一个新的度角来看待OODB与其事务处理之间的关系，数据库看作是一个虚拟对象的结构，动态事务处理层次看作是该对象的方法，据此提出了基于OODB类网络的事务模，其思想适用于一般的OODB的事务处理，该模型的动态特征和静态特征较好地刻画了OODB事务处理的特点。在模型中，可以区分粗粒度和细粒度并行性，它们分别对应对象之间和对象内部的并行性。我们还强调了此模型的几个实现问题。

本文着重在技术及实现方面，因而不象传统的事务处理那样具有很好的理论认识感，虽然这种情形几乎存在于OODB的所有研究方面，但是，这是OODB事务研究的一个新思想。我们还不能说本文的模型是完备的，却认为这是该领域中有益的表现。今后，对基于类网络的OODB事务模型的工作将集中在雅致的形式化和详细的实现技术方面，这是我们在开发GOODbase时所不能回避的。（参考文献共20篇略）

计算机科学

第6期(双月刊)
1991年12月23日出版
国内统一刊号：CN51—1239
代号：78—68
定价：2.15元　国外定价：5美元

编辑者：中国科学技术情报研究所重庆分所
顾　问：《计算机科学》审编委员会
出版者：中国科学技术情报研究所重庆分所
重庆市中区胜利路132号
邮政编码：630013
印刷者：中国科学技术情报研究所重庆分所印刷厂
总发行处：四川省重庆市邮政局
订购处：全国各地邮政局