第4卷 第5期
1993年10月

软 件 学 报
JOURNAL OF SOFTWARE

Vol. 4, No. 5
Oct. 1993

复杂对象及其并发控制

徐庆云　　王能斌　　陈　钢

（华东石油地质局研究中心，南京 210031）　（东南大学，南京 210018）

COMPLEX OBJECTS AND THEIR CONCURRENCY CONTROL

Xu Qingyun

(*Research Center of East China Bureau of Petroleum Geology, Nanjing* 210031)

Wang Nengbin and Chen Gang

(*Southeast University, Nanjing* 210018)

Abstract　A complex object is an abstraction and description of a complex entity of the real world. Many applications such as CIMS, CAD and OA need to define and manipulate complex objects. In this paper, a definition of the model of complex objects is given, and the concurrency control mechanism of complex objects in WHYMX object-oriented database system is described.

摘要　复杂对象是现实世界复杂实体的抽象与描述，CIMS、CAD和OA等许多应用领域需要定义和操纵复杂对象. 本文给出了复杂对象模型的定义，并论述了面向对象数据库系统WHYMX的复杂对象并发控制机制.

§0. 引　言

面向对象数据库系统WHYMX是针对CIMS需要而研究开发的新一代DBMS.

在WHYMX数据模型中，现实世界中的任何实体都被抽象为对象，每一对象都有唯一的对象标识符. 一个对象拥有自己的属性与操作. 属性是对象定义的结构部分，操作是对象定义的行为部分. '相似'的对象可以聚集成一个类. 属于同一类的所有对象具有相同的属性、操作、约束和直接超类，这些信息统称为类的特性. 类所拥有的对象也叫类的实例. WHYMX支持分类、聚合、概括、多继承性和封装等数据抽象，它的数据模式为具有代数格性质的类格结构. 类与类之间或者具有聚合联系（即低层类是高层类的组成部分）或者具有概括联系（即高层类是低层类的抽象，低层类是高层类的例化）. 在概括联系中，高层类是低

本文1991年1月9日收到，1991年6月18日定稿. 本文的工作得到国家自然科学基金和国家高技术计划的支持.
作者**徐庆云**，工程师，1990年硕士毕业于东南大学，主要研究领域为数据库，计算机图形学. **王能斌**，教授，主要研究领域为数据库及信息系统. **陈钢**，讲师，主要研究领域为数据库管理系统.

层类的超类，低层类是高层类的子类. 超类的特性被其子类继承，这种继承具有传递性，使得一个类可以递归地继承其超类链上所有超类的特性. 子类也可以不用继承的特性而另行定义.

类格结构以及类间的继承性大大减少了数据冗余，但也带来了事务管理的复杂化. 例如，对类更新时，不但本类及其实例需要更新，还要对其所有子类及其实例做相应的修改，这就导致了存取冲突的可能. 访问类或实例时，不只是对所要访问的类或实例加锁，还要对其超类链上的所有超类加意向锁；访问复杂对象时，不仅对所访问的复杂对象加锁，还要对其超类链上的全部超类以及组成此复杂对象所有有关的类加意向锁，以避免事务间冲突造成数据库的不一致性，保证事务运行的原子性和数据操作的语义完整性.

由于复杂对象的概念大大提高了对现实世界描述的能力，所以在 WHYMX 中，我们把复杂对象作为物理聚集(clustering)、对象管理、恢复和并发控制的单位. 传统的封锁法不足以解决复杂对象的并发控制问题，需要扩展. 本文首先定义了复杂对象模型，描述了它的语义，然后给出了它的并发控制策略和封锁协议.

§1. 复杂对象模型

1.1 对象之间的引用关系

假设有两个对象 O 和 O′，如果对象 O′的一个属性的值是 O 的对象标识符，则称对象 O′引用了对象 O. 如果 O 是 O′的一部分(a part of)，则称 O 与 O′间存在聚合关系. 这种 O′对 O 的引用称之为聚合引用. 反之，如果 O′仅仅引用了 O 的标识符，而 O 不是 O′的一个组成部分，这种引用称之为简单引用. 例如一架飞机引用公司的标识符作为它的生产厂家的属性值，显然，这个公司不是飞机的一部分. 一个对象可以是一个乃至多个对象的组成部分. 如果 O 只是 O′的一部分，这种聚合引用关系称为排它聚合引用；如果 O 是 O′以及其它一些对象的一部分，这种聚合引用关系叫做共享聚合引用. 聚合引用的语义还可根据对象是否存在依赖于引用它的对象进一步扩展. 一个从 O′到 O 的依赖聚合引用是指 O 存在依赖于 O′，而一个从 O′到 O 的独立聚合引用是指 O 不存在依赖于 O′.

从上述讨论可知，两个对象之间可能存在着简单引用、依赖排它聚合引用、独立排它聚合引用、依赖共享聚合引用和独立共享聚合引用等五种引用关系.

1.2 复杂对象

在面向对象系统中，具有相同特性的一组对象组成一个类. 类的定义如下：

```
CLASS {ATTRIBUTE /* 属性集 */
        a1: domain—class1
        a2: domain—class2
            ……
        an: domain—classn
      OPERATION      /* 操作集 */
      CONSTRAINT     /* 约束集 */
      SUPERCHAIN     /* 直接超类集 */}
```

其中 $a_i(i=1,2,\cdots,n)$是类的属性，$domain-class_i$ 是属性 a_i 的定义域. 属性域(相当于程序设计语言中的数据类型)或是基本类(例如整型类，字符串类)，或是用户定义类. 域为基本类

的属性称之为基本属性,它的属性值为一基本对象(即基本类的实例);域为用户定义类的属性称之复杂属性,它的属性值为一对象标识符.复杂属性分为简单引用属性和聚合引用属性.我们称含有聚合引用属性的类为复杂类,称该类的实例为复杂对象.下面借用 BNF 范式符号给出复杂对象的定义:

〈对象〉::=〈基本对象〉|〈简单对象〉|〈复杂对象〉

〈基本对象〉::=〈基本属性〉*

〈简单对象〉::={〈基本属性〉}〈简单引用属性〉*

〈复杂对象〉::={〈基本属性〉}{〈简单引用属性〉}〈聚合引用属性〉*

〈基本属性〉::=〈整型类实例〉|〈实型类实例〉|〈字符串类实例〉|〈布尔类实例〉

在上述定义中,{}表示出现零或多次,*表示出现一次或一次以上,|表示'或'关系.在上式中,〈简单引用属性〉〈聚合引用属性〉的含义见本节开始的说明.

每一复杂对象通过聚合引用属性与一些组成对象相联系,而组成对象本身又可含有聚合引用属性,从而形成一个递归嵌套的复杂对象层次结构.复杂对象层次上的对象所属类又组成了一个复杂类层次,其中组成对象所属类称之为组成类.复杂类层次上的所有类需要在类定义约束条件中说明其对象是否可被排它抑或共享聚合引用,独立抑或依赖聚合引用.这一工作可以在创建复杂对象层次或复杂类层次时进行,也可在定义之后再做修改.复杂对象层次可以按从根到叶的顺序生成,也可按从叶到根的顺序生成,这样可以利用已有的对象"装配"成为一个复杂对象.一复杂对象可以被别的复杂对象聚合引用,成为后者的组成部分,构成一个更为高级的复杂对象,称之为复杂对象组合.一个复杂类属性的聚合引用性质可以被该类的所有子类所继承.例如,人与哺乳动物均是复杂类,并且人是哺乳动物的子类,前者从后者那里继承了消化系统等聚合引用属性.

§2. 复杂对象的并发控制机制

事务在访问复杂对象时,常常还要访问被聚合引用的部分或全部组成对象,因而我们把复杂对象层次作为封锁单位,而不是个别地对组成对象加锁.

为了解决事务间的冲突、增加系统并发度,需要引入几种新的意向封锁方式:

ISC/IXC 锁:如果在复杂对象上加 S/X 锁,则要对该复杂对象层次上具有排它独立或排它依赖聚合引用联系的组成类加 ISC/IXC 锁.

SIXC/SIXCS 锁:若要分别查询和更新一组成类的不同对象,则对该类加 SIXC 锁(若该类被排它独立或排它依赖聚合引用)或 SIXCS 锁(若该类被共享独立或共享依赖聚合引用).

ISCS/IXCS 锁:若在复杂对象上加 S/X 锁,则要对该复杂对象层次上具有共享独立或共享依赖聚合引用关系的组成类加 ISCS/IXCS 锁.

复杂对象的封锁协议定义如下:

1. 复杂对象查询

(1)沿复杂类的所有超类链,对每一超类加 IS 意向锁.

(2)对复杂类加 IS 意向锁.

(3)对复杂对象加 S 锁.

(4)对在类定义约束条件中说明是排它独立或排它依赖聚合引用的组成类加 ISC 意向

锁,对说明是共享独立共享或依赖聚合引用的组成类加 ISCS 意向锁.

2. 复杂对象更新

(1)沿复杂类的所有超类链,对每一超类加 IX 意向锁.

(2)对复杂类加 IX 意向锁.

(3)对复杂对象加 X 锁.

(4)对在类定义约束条件中注明是排它独立或排它依赖聚合引用的组成类加 IXC 意向锁,对注明是共享独立或共享依赖聚合引用的组成类加 IXCS 意向锁.

3. 组成对象查询

(1)对组成类的所有超类链上的超类加 IS 意向锁,对该组成对象所属的组成类到复杂类这一路径上的各组成类加 ISC 意向锁(若类定义约束条件中说明是排它独立或排它依赖聚合引用)或 ISCS 意向锁(若类定义约束条件中说明是共享独立或共享依赖聚合引用),对复杂类加 IS 意向锁.

(2)对组成对象加 S 锁.

4. 组成对象更新

(1)对组成类的所有超类链上的超类加 IX 意向锁,对该组成对象所属的组成类到复杂类这一路径上的各组成类加 IXC 意向锁(若类定义约束条件中说明是排它独立或排它依赖聚合引用)或 IXCS 意向锁(若类定义约束条件中说明是共享独立或共享依赖聚合引用),对复杂类加 IS 意向锁.

(2)对组成对象加 X 锁.

5. 分别查询和更新一组成类的不同对象

(1)对该组成类的所有超类链上的超类加 SIX 意向锁,对该组成类到复杂类这一路径上的各组成类加 SIXC 意向锁(若类定义约束条件中说明是排它独立或排它依赖聚合引用)或 SIXCS 意向锁(若类定义约束条件中说明是共享独立或共享依赖聚合引用),对复杂类加 SIX 意向锁.

(2)对要查询的组成对象加 S 锁,对要更新的组成对象加 X 锁.

由于类格结构是类的有根连接无循环有向图,一个类可有多个超类,多继承性又导致了更新传播效应,所以在访问复杂对象时需要对它的所有超类链加意向锁,以避免一些事务沿不同的路径访问类格时产生冲突.

由于复杂对象与组成对象之间有着整体与部分的关系,无论两者存在何种聚合引用关系,在对前者加锁的同时,还要对后者所属的组成类加意向锁,这样访问复杂对象的事务可以访问它的组成对象而无需另外加锁,还可防止其它事务同时访问该复杂对象的组成对象时产生冲突,避免该复杂对象的组成对象出现不一致性. 所以把复杂对象层次作为加锁的单位是合适的. 在访问组成对象时,除了对其超类加意向锁外,还要对其复杂类层次路径上的类加意向锁,这是为了防止不同事务沿该组成对象的超类链方向和复杂类层次方向同时访问类格时出现冲突.

§ 3. 封锁方式转换

事务有时在数据单位上先施加排它性较弱的锁,然后在需要时再升级为排它性较强的

锁，这主要是为了增加系统并发度和减少封锁表冗余项.

各种封锁方式的排它性有强有弱，对它们排序，可以得到各种封锁方式排它性的偏序关系，据此可以导出图1所示的封锁转换矩阵.

请 求 方 式

原有方式	IS	IX	S	SIX	X	ISC	IXC	SIXC	ISCS	IXCS	SIXCS
IS	IS	IX	S	SIX	X	S	SIXC	SIXC	ISCS	SIXCS	SIXCS
IX	IX	IX	SIX	SIX	X	X	X	X	IXCS	IXCS	SIXCS
S	S	SIX	S	SIX	X	S	SIXC	SIXC	S	SIXCS	SIXCS
SIX	SIX	SIX	SIX	SIX	X	SIXC	SIXC	SIXC	SIXCS	SIXCS	SIXCS
X	X	X	X	X	X	X	X	X	X	X	X
ISC	S	X	S	SIXC	X	ISC	IXC	SIXC	ISCS	IXCS	SIXCS
IXC	SIXC	X	SIXC	SIXC	X	IXC	IXC	SIXC	IXC	IXCS	SIXCS
SIXC	SIXC	X	SIXC	SIXC	X	SIXC	SIXC	SIXC	SIXCS	SIXCS	SIXCS
ISIC	ISCS	IXCS	S	SIXCS	X	ISCS	IXC	SIXCS	ISCS	IXCS	SIXCS
IXCS	SIXCS	IXCS	SIXCS	SIXCS	X	IXCS	IXCS	SIXCS	IXCS	IXCS	SIXCS
SIXCS	SIXCS	SIXCS	SIXCS	SIXCS	X	SIXCS	SIXCS	SIXCS	SIXCS	SIXCS	SIXCS

图1　封锁转换矩阵

§4. 死锁解除方法

在多个事务并发执行的情况下，如果采取封锁措施保证数据的一致性，就不可避免地会出现死锁. WHYMX 采用了超时法来解决死锁问题. 超时法的基本思想是：当一事务对某一数据单位申请同一方式锁均遭拒绝时，便假定已发生死锁，于是由系统撤销该事务.

设 reqtimes 是事务 T 申请锁已花的时间(以申请次数来表示)，lockcount 是 T 已获得的锁的数目，WAITTIME 为申请封锁等待的时限，当(reqtimes/(lockcount+1))>WAITTIME 时认为已超时，事务 T 被撤销. 一般而言，事务已封锁的对象越多，其开销越大、代价越高，而系统处理死锁时选择开销小、代价小的事务撤销.

结束语：复杂对象的并发控制是实现面向对象 DBMS 中一个比较麻烦的问题. 本文给出了复杂对象模型的定义，并提出了并发控制策略和封锁协议，有效地解决了复杂对象的并发控制问题. WHYMX 是一个面向对象的 DBMS 实验系统，并发控制是其中的一部分，作者对参加此项研究工作的其他同志谨致谢意.

参考文献

1 Jorge F. Garza, Won Kim, Transaction Management in an Object—Oriented Database System, ACM—SIGMOD, 1988.

2 Xu Qingyun and Wang Nengbin, Concurrency Control of an Object—Oriented Database System for CAD/CIMS, 2nd International Conference on CAD&CG, 1991.

3 Won Kim et al., Composite Object Revisited, Technical Report, MCC, 1989.

4 徐庆云，王能斌等，面向对象数据库的并发控制，全国第九届数据库学术会议论文集.

5 徐庆云，王能斌，面向对象数据库的事务管理，计算机研究与发展，No. 8, 1992.

6 徐庆云，王能斌，WHYMX 的嵌套事务管理，计算机应用，No. 5, 1991.