维普资讯 http://www.cqvip.com

OBMS IDKE 模型 数据库 面向对象

计算机科学1992 Vol.19 No.2

编者按语

自八十年代以来，非传统应用领域对数据库需求的增长和数据库研究的深入导致了面向对象数据库(简称OODB)的产生。OODB在近几年发展中所显示的强大生命力使其有可能成为下一代数据库系统的主流。

下面刊登的三篇论文，阐述了一个OODBMS——OBMS/IDKE的数据模型、无缝的面向对象编程语言、动态模式管理以及对象存贮/存取管理中的关键技术问题。从此可以看到，OBMS/IDKE吸收了现有OODB系统和OO模型的特点，从对象语义出发对OO模型的语义关联进行了考察，形成了自己的对象模型语义体系，以及形成了从对象的物理存贮、缓冲区管理、概念模型到语言界面一整套具有特色的对象库系统。

OBMS/IDKE:模型及其无缝的面向对象编程语言*)

33-43 TP311.13

唐元昌 王珊

(中国人民大学数据与知识工程研究所，北京100872)

摘 要

With the development of database application, it is urgent to breake through Relation Model and RDBMS's limitation in structure modelling and semantics expressing. In the paper, we have briefly summarized varions approachs to the aim, and introduced our object base management system OBMS/IDKE, especially its object-oriented data model PUC which is based on complete semantics inheritance, and its seamless object-oriented programming language ODPL.

一、关系模型以来的发展方向(后关系模型)

在CAD/CAM、OIS、VLSI、CASE等计算机应用领域，传统的关系数据库缺乏灵活、丰富的建模能力，已不能满足要求。为了解决这一问题，人们提出了许多新的并发展了一些原有的数据模型。这些尝试沿着如下几

*) 国家自然科学基金资助项目

on in Incremental Semantic Evaluation, 2th Proceeding of ICYCS

〔6〕施丽华，拓广属性文法及在增量语义分析中的应用，硕士论文，华东师大，91.1

〔7〕面向Pascal的句法制导的程序设计环境的设计与实现，技术报告，华东师大计算机科学系，90.9

维普资讯 http://www.cqvip.com

个方向进行：

1.对传统的关系模型(1NF)进行扩充 引入少数构造器或新思想，使关系模型能够表达复杂的数据类型，增强其结构建模能力。我们称这样的模型为复杂数据模型。按照其扩充的侧重点，复杂数据模型可分为二种，一种是偏重于结构。首先出现的有嵌套关系模型(NF^2)[10,14]，其它的还有V-relational模型[1]，Format模型[9]等等。基于这种扩充的系统有Unidata公司的商品化系统Unidata[23,24]，它能表达"表中表"，并且表中的一个域可以是一个函数（它称为虚域）。另一种是偏重于语义的扩充，像伯克利大学的POSTGRES[18,22]，它支持关系之间的继承，也支持在关系上定义函数和运算符，但关系的结构仍然是一张平面表，"表中表"只能通过关系上定义的函数来模拟。

总的来说，在复杂数据模型和基于它们的数据库系统里，客观世界中的实体用tuple或其中的key来表示，不支持太多的语义关联，不区分类和型，因而它们的数据库语言也不支持类型检查。和关系数据模型和系统一样，这种模型和系统的主要缺点是不能保证实体的确定性；实体的引用只能通过码或数据冗余来达到。其主要优点是建立在这类模型上的系统实现起来相对简单些。

2.提出和发展一些相比关系模型来说全新的数据构造和数据处理原语 能表达复杂的结构和丰富的语义。这样的模型中比较有代表性的有函数数据模型FDM[20]，IFO模型[3]、语义数据模型SDM[8]、RM/T模型[6]以及E-R模型等，我们统称为语义数据模型。它们的特点是引入了丰富的语义关联（如ISA，ISP等），能更自然、更恰当地表达客观世界中实体间的联系。加上比较丰富的结构构造器（如tuple，set，list等），因此它们又具有很强的结构表达能力。例如，FDM通过多值属性隐式地提供set功能，允许用抽象类来构造新的抽象类；支持类之间的ISA关联。ISA关联可以用来表示Subset和Union关系。在FDM中用ISA关联表达Union时还可以通过约束条件保证ISA的源（子类）之间的相交或不相交。

在[4，5，6]中，人们还对各种不同的ISA关联进行了讨论。例如，IFO模型将ISA区分为二种，一是特化（Specialization），二是概括（Generalization），它们对型的内部结构和属性的继承有着不同的影响。

也许是由于它们比较复杂，在程序设计语言和技术方面没有相应的支持，计算机硬件也没有发展到一定的水平，因此直到八十年代末期，它们都没有在数据库系统实现方面有什么大的突破，而至多被当作数据库设计中概念建模的一种工具（如E-R模型）。

3.将语义数据模型和O-O程序设计方法结合起来 提出面向对象模型，开发出了众多研究性或商品化的对象库系统。这类模型和系统具有如下一些特点：支持复杂对象；支持对象标识和对象的封装性；支持类或型及其层次结构，能表达丰富的语义；支持复载、过载和滞后联编；具有计算完备性、可扩充性、持久性；拥有传统DBMS所具有的功能[2]。

目前比较著名的OODB系统如OZ+[15,17]，Iris[7,13]，ORION[11,17]，Gemstone[17,19,21]以及Vbase（ONTOS）[16,25]等，都基本上具备上述大部份特征。但相比语义数据模型，它们所支持的语义关联还不够丰富，比如不区分特化和概括这二种ISA关联，而统称为子类-超类关联，并且在这种关联中，它们还存在着与我们在后面将要讲述的对象的语义不一致的地方。它们专注于永久对象，没有严格区分类和型。

O-O技术在数据库领域的应用仍在继续发展，人们正不断往O-ODB及O-O数据模型中加入一些新的特点，并结合人工智能等技术加以实施。如让对象能自动完成完整性约束检查，自发地触发一系列事件，使对象表现出更大的能动性。

O-O模型和系统吸收了语义数据模型和

维普资讯 http://www.cqvip.com

O-O程序设计语言及方法学的优点，但系统实现难度还是比较大，效率也比较低。

在语义数据模型和O-O数据库系统与模型之间还有一些中间性质的模型。逻辑数据模型LDM就是这样的例子，它很接近于实际系统中底层的数据表示，有些O-ODB系统就使用和它极为相似的形式存贮对象。

二、OBMS/IDKE系统的目标

OBMS/IDKE系统的研制是沿着上述第三个方向进行的。它是一个通用的OBMS核心，以语义严格的O-O模型PUC为基础，具有很好的开放性和可扩充性，在其中用户可以定义新的类和型，并且系统定义和用户定义的类和型在使用上没有区别；支持多继承和版本，提供动态建模/改模功能，复杂集合条件的查询；支持一种由C++扩充而来的无缝而灵活的编程语言ODPL(Object-Oriented Database Programming Language)——它集高级语言C++计算、控制、处理挥发性对象的能力和数据库语言处理永久性对象的能力于一体。通过引入对象库类和集合等概念，在很大程度上取消了二种语言之间的"阻抗不匹配"现象。

在实现上，与上述目标相适应，OBMS/IDKE系统采取了一种基于散列的对象分割存储和多层次的对象标志策略，在缓冲区管理上灵活运用了对象的思想，将缓冲区当作一个大的对象来看待和处理，大大提高了缓冲区管理的模块独立性和界面清晰度。从而形成了从对象的物理存贮、缓冲区管理到概念模型、语言界面具有一整套特色的对象库管理系统。以OBMS/IDKE为基础，可以根据不同应用领域的情况，开发出不同的应用开发环境，更好地满足这些特殊领域的特殊需要。

图1表示了OBMS/IDKE系统体系结构以及它和不同应用领域之间的关系。该系统在SUN 3/280 UNIX环境下开发。

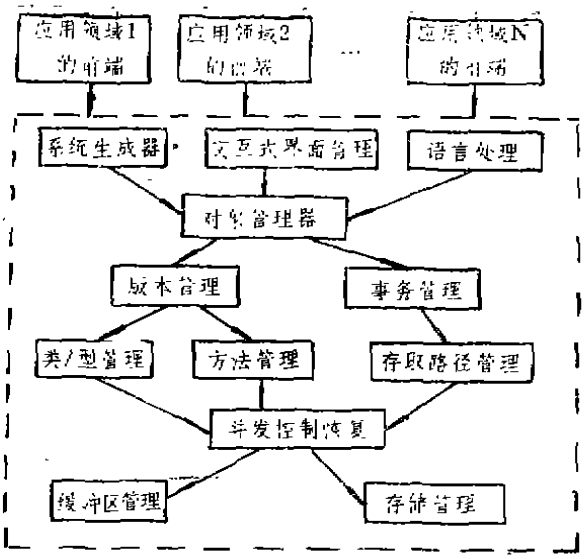

图1 虚线框内为OBMS/IDKE系统体系结构

三、OBMS/IDKE系统的数据模型PUC

当前的O-O模型对结构和结构之间的语义关联研究较多，但对"对象"的语义研究还较少，而且对类之间语义关联的研究也没有结合对象的语义来进行。

PUC模型和一般O-O模型有许多不同之处，但最根本的区别还是对"对象（object)"的理解，即是否给"对象"赋与了语义，对象的各部分是紧密的还是松散的。因此，为了说明PUC模型，我们首先从对"对象"的认识入手。

3.1 对"对象"的认识

在理论研究中提出来的O-O模型和O-ODB系统所使用的O-O模型总的来说建立在这样的认识上：对象是客观世界中的一个"能动"的实体，由数据信息和动作（包括约束，界面，触发器等）二部分组成。这些动作是它对外施加主动行为的手段和途径，是其"能动"之所在。对"对象"的认识也就仅此而已。在此基础上提出了类的概念，以及类之间的各种关联，如组合关联(ISP)、继承关联（或称子类-超类关联，ISA）、可相交与不可相交关联等。

在PUC模型里，一个对象也是客观世界

维普资讯 http://www.cqvip.com

里的一个实体，一个类也是代表具有相同性质的一些对象构成的集合。

一个对象必定对应到一个唯一的类，此类能精确地、完全地反映和刻画这个对象各个方面的信息和特征，我们称此类为该对象的精确类。虽然抽象地看，一个对象可以从不同层次看作是另外一些类的对象，但这些类却都不能精确、完全地描述此对象的各个方面，我们称这些类为此对象的非精确类。

图2表示出了由类PERSON、TEACHER、STUDENT及TA、WORKER构成的一个对象库模式。John目前是一个参与教学辅导工作的高年级学生，显然John可以看成是上述前四个类中任何一个里的一个对象，但只有类TA可以精确、完备地刻画出John的身份，因此TA是John的精确类，其它三个类都只是描述了John在某一方面的信息，因此都只能是John的非精确类。

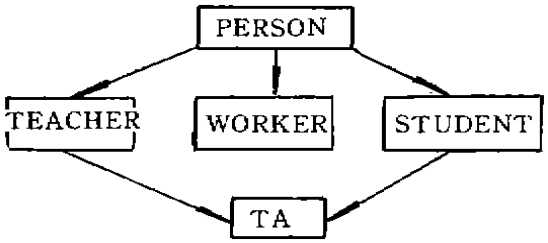

图2（→表示特化）

如果我们引入一个虚类SUPER-CLASS，将它作为所有类的超类，那么一个对象的精确类和非精确类必定按ISA关联构成一个格（Lattice），SUPER-CLASS是这个格的最大上界，这个对象的精确类是这个格的最小下界。我们把这个格称为此对象的抽象格，这个抽象格中的每个结点都叫做该对象的抽象类。SUPER-CLASS是所有对象的抽象类，任何对象在SUPER-CLASS上的抽象是此对象的OID。

一个对象的抽象格穷尽地描述了该对象能够从不同抽象层次被看待的方式，不同的方式和抽象层次对应此格上唯一的一个类。对一个对象，我们虽然可以从其抽象格中不同抽象类的角度看到不同的"映象对象"（该对象在这些抽象类上的抽象），但它们并不是独立的，它们其中的任何一个都只是该对象的一部分。一个"映象对象"在相应的类上能够如何被处理、如何动作不仅受这些类的数据信息框架和动作、约束所制约，而且也与这个对象本身所具有的性质有关，即还取决于这个对象的精确类。这就是说，通过继承，不仅超类影响子类的框架，子类也可以反过来影响子类中的对象在其超类上的"映象对象"的信息项和动作；子类中的对象虽然可以看作是超类中的对象，但是否能和超类中的某些对象（这些对象的精确类为此超类）一样表现还取决于子类是否有特殊限制。

举个例子来说，设类SPIER（"特工"）是类PERSON的一个子类。对一般人来说，其经历、背景信息在PERSON这一级是可见的，但如果某人S1是一名特工，他虽然从PERSON这一级来看是一个人，但其履历、工作性质等方面的信息则不允许在PERSON这一级被看到，这是因为类SPIER具有一些特殊的约束和限制，作为SPIER类中的一个对象就应满足这些约束和限制，即使它被抽象成PERSON类中的一个对象也是如此。

因此，一个对象在其抽象格中不同抽象类上的"映象对象"之间的联系是紧密的、有机的、逻辑上不可分的，相互间有着约束关系，它们共同构成一个语义实体。

PUC模型中对对象的这种认识，一方面更符合客观世界中的实际情况，另一方面可以使PUC模型本身及OBMS/IDKE系统语义严格，并且可以降低系统实现的复杂性，提高系统的性能。

3.2 PUC模型的特点

下面我们先从继承、共享和数据隐蔽三个方面谈谈PUC模型的特点。

3.2.1 **对象的继承** 正如上所述，一个对象是一个逻辑上不可分的整体，它的各个"映象对象"不能作为完全独立的对象来看待

维普资讯 http://www.cqvip.com

和处理。但这点要和一个对象中通过聚集（aggregation）关联得到的构成部分（成分对象）区别开来。当一个对象引用另一个对象时，被引用的对象可以是独立的。

在PUC模型里，从结构上看，继承使得子类可以拥有超类所拥有的框架（数据信息项和动作）；从语义上看，继承使得一个对象在其所有抽象类中的"映象对象"不可分割，互相依赖；从数量上看，一个子类和其任何一个超类之间存在这样一种关系：设OBJ（A）代表子类A中的对象集合，OBJ（B）代表超类B中的对象集合，∏（OBJ（A））表示OBJ（A）中的每个对象在B上的"映象对象"构成的对象集合，则∏B(OBJ（A））≤OBJ（B）—EXTOBJ（B），其中EXTOBJ（B）是精确类为B的对象构成的集合；且当A是B的唯一子类时，OBJ(A)和OBJ（B）—EXTOBJ（B）之间的对应关系是一一对应关系，唯一的对应函数是：对任意的OA∈OBJ(A)，∏B(OA)←→OA；当B有多个子类时，OBJ（A）和OBJ（B）—EXTOBJ（B）之间存在这样的对应关系：对OBJ（A）中的任一对象OA，有且仅有OBJ（B）—EXTOBJ（B）中的一个对象∏B（OA）与之对应。

3.2.2 **对象的共享** 面向对象数据库系统中存在大量共享。借助于共享，O-O模型才能表达和处理复杂的大对象和体现丰富的语义。我们撇开对象（类）之间结构、语义动作的共享不谈（O-O程序设计语言和对象库都支持这类共享），对象库中对象之间数据信息的共享有如下几种形式：一是一个对象通过聚集关联引用其它对象；二是一个对象和这个对象在其所有抽象类上的"映象对象"共享一份存贮；三是一个对象的不同版本在对象的部分数据信息上的共享。这三类共享是PUC模型所支持的。此外，在有些O-O DB系统中还存在第四类共享，即二个对象在它们的某个公共抽象类上的共享，这一特点是它们从层次、网状数据库沿用过来的。我们称这种形式的共享为继承共享。

继承共享是不必要也不合理的。它有二种形式，一是二个子类中的对象在它们某个公共超类上的共享，如图3及图4所示；另一种是一子类中的对象和某个超类中的一个对象在此超类上发生共享，如图5所示。

继承共享只是我们在对对象的语义缺乏深入了解的情况下对aggregation（ISP）关联的一种极易出现的误解。

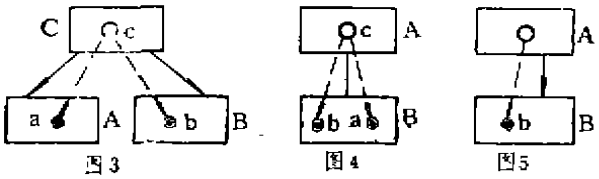

图3　图4　图5

→表示ISA关联　⋯→表示对象之间的继承关系　●表示其精确类为其所在类的对象　○表示该类的某个子类中的对象在此类上的"映象对象"

3.2.3 **一个例子** 利用前面介绍的PUC模型的特点，可以很清楚地将一个O-O对象库应用模式中的非ISA关联区分出来。例如，有三个类分别为STUDENT—表示学生信息，COURSE—表示课程信息和SCG—表示学生选课成绩。对一个初级用户来说，很容易建立起图6所示的对象库模式。

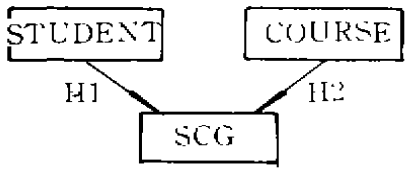

图6

ISA关联H1和H2表现的都是1:m数量关系，体现的是一种继承共享。如果SCG中有多个同一学生的选课对象，则在STUDENT中将会出现表示同一学生实体的多个对象，这就出现了数据的大量冗余和语义混乱，并会引起修改的不一致性。

图6中的这种非ISA关联之所以不能用通常的O-O模型从理论上区分出来的原因是，在一般的O-O模型中，一个对象对应到

维普资讯 http://www.cqvip.com

它所能属于的那些类上的各部分之间的联系是松散的，继承仅仅只是结构的继承，只要结构相包容，ISA关联即可成立，而无其它语义约束。基于结构继承的O-O模型的系统无法保证对象的语义，并且为了这种不合理的继承共享，不得不增加实现的复杂性，牺牲系统的运行效率，减弱OODB系统所应具备的一些优良特征（相比层次、网状数据库系统），如动态建模/改模等。

3.2.4 **数据隐蔽** 在一般O-O模型和OODB系统中，数据隐蔽通常是指在一个类中被定义为private的属性（包括数据信息项和动作）不能为外界窥视，用户只能透过一个类的公共界面看到属于该类的对象中的一些信息。我们称之为浅层数据隐蔽。PUC模型也支持这种数据隐蔽。

和一般的O-O模型不一样，PUC模型给出了深层数据隐蔽的概念。

例如，图7所示的数据库模式中，A是B的超类，A的公共界面是F1、F2和F3，而B是以private方式继承A的F1、F2，B的公共界面是f1，f2和F3，其中f1，f2是B自己提供的，而F3则是从类A以public方式继承过来的公共界面。若b是精确类为B的一个对象，其在类A上的抽象为bA；a是精确类为A的一个对象。那么如果我们从类A的角度看A中的所有对象，其结果将是图8所显示的。其中～是不可视标识。

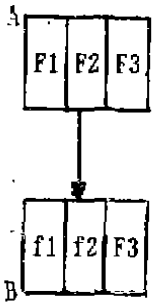

图7

VIEW 界面 / 对象	F1	F2	F3
a	a.F1	a.F2	a.F3
bA	～	～	bA.F3

图8

这样的数据隐蔽是合理的。在图7所示的情况下，对B中的一个对象，模式的设计者不希望该对象的有关信息能够通过类A的界面F1，F2被看到。对一个对象来说，只有其精确类才能完全、精确地刻画它的所有性质，越靠近精确类的抽象类越能更完备、更精确地描述该对象的性质。对象的信息可视性也是对象的性质之一。因此，既然子类B不允许属于它的任一对象b的有关信息通过A的界面F1和F2被看到，那么在超类A这一级，也不能通过A的界面F1和F2看到映象对象bA中的相应信息。我们称这一种信息数据叫做"深层数据隐蔽"。基于伪OID概念，深层数据隐蔽可以很容易地实现。

在一个类上的浅层数据隐蔽是由该类的定义决定的，对可属于该类的任何对象都有效。而在类A上的深层数据隐蔽则是由类A的定义及其某个子类B的定义共同决定的，它只对精确类为B的对象有效。A的不同子类B和A一起可构成不同的深层数据隐蔽。

3.3 PUC模型总结

PUC模型提供了比较丰富的结构构造方法和义语关联，具有很强的建模和表达能力；同时，这些语义关联又很精炼，因而基于PUC模型的OBMS/IDKE系统实现起来效率更高。

3.3.1 **对象标识符** 对象标识符分为二种，一是对象的内部标识符，它是系统统一分配的，供系统或用户编程使用，在全系统中唯一地标识一个永久对象，有关细节可参见本期有关OBMS/IDKE存储/存取一文；二是对象的外部标识符，它是用户给对象指定的名称，用户可以直接使用。

为了刻画对象被抽象的方式，对象的内部标识又有两种机制：对象标识符(OID)及对象伪标识符（POID或OID'）。

基于PUC中所使用的标识一个对象的方法，我们可以支持三种不同级别的对象相等和等同（详见本期OBMS/IDKE存储/存取一文）。

维普资讯 http://www.cqvip.com

3.3.2 **类（$class）、型（type）、集合（set）与快照（$snapshot）** PUC模型区分类和型。型（Type）是指值的结构，描述值的内涵（intention）。完全是一个type的有int，char，long，double，float，text这些简单type以及用户在对象库类中定义的任何struct，ODPL语言C++部分定义的任何挥发性class和struct。PUC模型中纯粹的type只能用来定义变量或属性的值的结构，而它们的值则需要在赋值之后才具备。

类这一概念是和永久数据的保存相联系的。一个类就是一个可区别的、由存贮在对象库中的对象构成的集合。为了与C++语言中的"class"区分开来，PUC模型引入了对象库类（$class）这一概念，其中的$表示该类为对象库类。对象库类是一种超级type，它不仅描述值的内涵，而且描述了其外延（extention）。对象库类集type定义和变量说明于一体，一方面，一个对象库类的名字可以当作一个type名来使用，在程序中定义变量和属性，另一方面，它也可以像变量一样接收值，存放许多对象。正是由于前一个特点，ODPL语言支持类型检查，是一种强类型的语言，由于后一个特点，ODPL又支持永久对象。

PUC的type之间可以定义一些语义联系。但只有对象库类之间才能定义后面要讲的所有语义关联。

快照是一种特殊的对象库类，用来存放查询的值结果。但不能在它和其它type之间定义任何有实质意义的语义关联。

PUC显式地支持set。它和一个type名一起就可以定义一个集合变量或属性。set中的元素都是同质的。但是，用户可以通过POID来间接地实现异质集合。set允许嵌套，因此使用PUC模型可以直接表示上下文相关数据。

3.3.3 **对象的封装性** PUC模型支持浅层数据隐蔽和深层数据隐蔽。已在3.2.4节中进行过讨论，这里不再重复。

3.3.4 **ISA关联** PUC模型所支持的继承从语义上看是完全语义继承；从形式上来看是全继承、多继承。如果出现名冲突，则以其超类层次深度优先的遍历顺序确定有效名。

PUC模型将ISA关联区分为二种，一种是特化（简称S关联），另一种是概括（简称G关联）。

我们用domain（X）表示类X中可能拥有的合法对象的集合。如果从类B到类A存在S关联（如图9），则domain（A）＞domain（B），即类A不仅包含来源于B的对象，而且还包括精确类为A的对象；如果从类B到类B1，B2，…，Bn存在G关联（如图10），则有

$$\text{domain}(B)=\bigcup_{i=1}^{n}\text{domain}(Bi)$$

这就是说，类B中的对象完全来源于B1，…，Bn，没有精确类为B的对象，类B依赖于类B1，…，Bn。我们称类B为依赖类，它依赖于它的子类。往依赖类中加入一个精确地属于它的对象是非法的。

依赖类可用于表示纯粹抽象的概念，可以减少对象库中数据信息的冗余和模式结构的冗余，方便从客观世界到概念模式，从概念模式到对象库模式的转化。

G关联只能在对象库类之间定义；而定义type之间的S关联时，对象库类不能作为非对象库类type的Subtype。

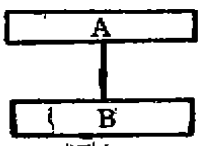

图9

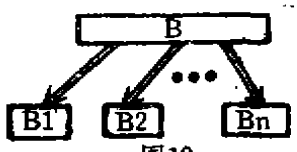

图10

3.3.5 **ISP关联（或称为聚集关联）** ISP关联是通过PUC的结构构造器体现出来的。如图11，它表示类A由类型分别为int、类B、嵌套定义结构C，int集合、类D的五个域组合而成。

维普资讯 http://www.cqvip.com

通过ISP关联引用某个类A中的一个对象时，PUC模型支持依赖和非依赖二种形式的ISP关联。所谓依赖的ISP关联是指，当引用对象被删除后，被它引用的对象也要一起删除。被引用对象的存在依赖于引用对象的存在（如图11中A的类型为D的域f5）。所谓非依赖的ISP关联是指被引用对象的存在不依赖于引用对象的存在（如图11中A的类型为B的域f2）。实现时对ISP关联进行了进一步划分（见本期OBMS/IDKE模式管理一文）。

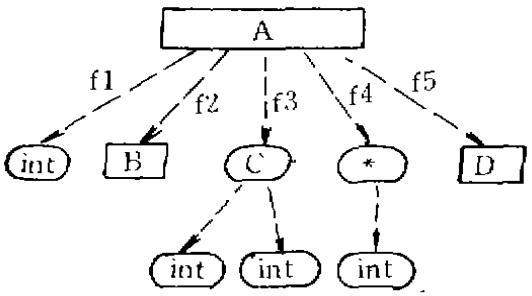

图11

定义对象库类时，被引用型只能是简单type或对象库类；而定义非对象库类的type时，可以引用简单type或任何已定义过的type（包括对象库类）。

3.3.6 **类之间的可相交关联与不可相交关联** 在PUC中，一个对象库模式的二个类之间的相交或不相交已由其类层次（按ISA关联构成）所决定。具体地说，如果二个类具有某个公共子类，则它们是可相交的，并且它们的相交部分就是它们的公共子类中的对象；否则，这二个类是不相交的，如若不然，就会出现一个对象的所有抽象类并不构成格的情形，与对象的语义相矛盾。

3.3.7 **数量关联** PUC模型显式支持类之间的1:m数量关系。一旦用户说明了二个类之间的这种数量关联之后，系统将自动维护二个类中对象之间的这种关系。

数量关联只能定义在二个数据库类之间，并且要求它们没有子类-超类关系。如果类A到类B之间存在一个1:m关联，我们在图示中用一条从A到B的……→边表示。

四、ODPL语言的无缝性及其实现

4.1 对象库类、挥发性类、永久对象变量和挥发对象变量

对象库类存放在对象库中，用来存放永久对象。已在对象库中的对象库类不需定义就可以被任何使用此对象库的程序所引用——定义变量，定义属性，对其中的对象进行操纵等（如果不违反授权约束的话）。挥发性类不存放在对象库中，也不能用来存放对象，其生命期就是定义此挥发性类的程序的运行过程。如下面的ODPL程序1和程序2，程序1里定义的类PERSON，STUDENT均为对象库类，而类B为挥发性类。

程序1

```
main ( ) {
$ class PERSON
{ int sec #; int age; char name[20]…};
$ class STUDENT : public $PERSON
{ int S#; $PARTY interest { }; …};
$class TEACHER : public $PERSON {…;
    Connect with $STUDENT M1; …}
$class TA:public $STUDENT, $ TEACH-
     ER{…};
 $class PARTY { char name[20];
text purpose; $ STUDENT chairman;
int no-of-mem; …};
class B {int fB1; $PERSON f2{ }}…}
```

程序2

```
main ( )
{$PERSON v1, v2{ }, $v3{ }, $v4; …}
```

对象库类和挥发性类都可以用来定义变量，这些变量同样可以分为永久对象变量和挥发对象变量。永久对象变量以字符'$'打头，它所代表的对象当事务提交时都将被永久化，存放到对象库中。永久对象变量只能用对象库类定义。挥发性变量所代表的对象一旦程序运行结束就将丢失。它能用任何类定义。

永久对象和挥发对象存在于两个不同的

维普资讯 http://www.cqvip.com

对象空间(永久对象空间和挥发对象空间)，一个对象空间中的对象可以通过恰当的赋值语句在另一个对象空间中复制镜面映射对象。

和对象库类不同，永久对象变量的永久性只体现在其值是永久的，其名的生命期和挥发性对象变量一样。

在程序中，两类对象变量的使用语法基本一样，如在整个对象的赋值(赋值在语义上有所区别)，修改某个域值等方面，虽然二种变量的实现方式完全不同。

永久对象可以用一个永久对象变量、对象的外部标识符或永久对象指针等来表示。例如：

```
$STUDENT*$stdp, $std(参数)AS @John;
/*创建一外部标识为@John的永久对象*/char objname[10]; strcpy(objname, '@John'); 若 $stdp= &$std 则$stdp->age=20, (*$stdp).age=20, @John.age=20, $std.age=20, obj(objname).age=20都是等价的。
```

ODPL中引入了集合，如程序2中 $v3 被定义成一个永久对象集合变量，v2被定义为一个挥发对象集合变量。集合和对象库类虽然有许多本质上的区别，但都是多值实体。因此ODPL为它们提供了许多相似的操作，限于篇幅，这里不详述。这些都有利于消除"阻抗不匹配"现象。

4.2 对象操纵举例

这里仅结合4.1节程序1中定义的对象库类举几个ODPL中对象操纵的例子(对C++的扩充部分也以'$'开头)。

4.2.1 对象查询 当我们查找的是整个对象时，查询结果放在一个特定的变量中，当查找的是结构值时，查询结果既可以放在一个快照中，也可以直接输出到指定的标准输出上。例如，

```
struct qry.{$STUDENT $std{}}std_set;
$PARTY$pt; $STUDENT $st; 查询 $std_set=$st IN ALL $STUDENT<
    {@John, @Erich, @Smith}
where $st.age<25 && EXIST$pt IN
$st.interest ($pt.chairman~$st && $pt.no_of_mem>20
```

是找出外部标识符为@John, @Erich, @Smith的三名学生中年龄小于25且是某大团体(人数多于20)主席的学生，并将它们的伪OID存入std_set.$std中。

4.2.2 对象的创建和命名 创建一个对象有多种方法，可以通过给永久对象变量赋一个挥发性对象变量，往一个永久对象集合中加入一个挥发对象，利用相应对象库类的构造函数，或直接加入。创建一个对象时可以给它赋一名字(对象的外部标识符)。例如

```
$STUDENT st1 (参数), $st_set{},
$std (参数);
$$std=st1 AS @John;
$ADD st1 TO $st_set AS @Erich;
$ADD OBJECT $TA (参数) AS @Smith;
```

等。

此外，我们还可以为一个对象在其不同的抽象类上定义不同的外部标识符，如对前面定义的精确类为$TA的对象@Smith，定义它在$TEACHER上的外部标识为@Blark；$NAME @Smith ON $TEACHER AS @Blank。

4.2.3 对象删除 对象删除有二方面的含义，一是彻底删除，和传统RDB中元组的删除一样；二是对象退化，指一个对象的抽象格退缩成其一个子格。对象退化时其OID不变，但要确保对象的语义。如

```
$For $std IN {@John, @Erich}
{@delete $std From $STUDENT}
```

是将@John, @Erich从学生中除名。

4.2.4 对象修改 对象修改也有二方面的含义，一是修改对象的某个属性上的值，例如@Smith.sal=200；二是修改对象的精确类，我们称这种修改为对象变迁。对象变迁时，新旧精确类之间一定要有子类-超类关系。例如定义变量$TEACHER $t，语句

维普资讯 http://www.cqvip.com

$FOR $t IN $TEACHER WHERE $t.age>65

{ $MOVE $t TO $PERSON }

是使年龄大于65的所有教师全部退休。

4.2.5 **涉及到关联的对象操纵** 与ISA，ISP有关的对象操纵已在上面的例子中有所体现。下面说明一下与数量关系有关的对象操纵。

例如，前面我们定义了从对象库类$TEACHER到对象库类$STUDENT的一个1:m数量关系M1，表示一个老师可能是某班的班主任，因而与该班的这些学生相联系。语句

$FOR $std IN {@John, @Erich}

{$CONNECT $std WITH @Smith ON $TEACHER BY M1}就是将学生@John, @Erich与教师@Smith（其精确类为$TA）相联系；语句

$DISCONNECT @John [WITH @Smith] ON $TEACHER BY M1则是将对象@John从与$TEACHER中所有对象的数量关联M1中解脱出来或仅从与对象@Smith在$TEACHER上的数量关联M1中解脱出来（如果[]中的项出现的话）。

对与一个对象相联系的对象也可进行操纵，如将与教师@Smith相联系的所有学生的奖学金增加100元：

$FOR $std IN $STUDENT CONNECTED WITH @Smith BY M1

{ $std.scholarship+=100 };

又如，为学生@John所属的教师加薪100元：

$FOR $t IN $TEACHER OWNS @John BY M1

{ $t.sal+=100 }。

4.3 ODPL语言的实现

对一个ODPL源程序的处理过程可粗略地描述如下：

1.通过ODPL预编译器，将ODPL源程序中对C++的扩充部份区别出来并加以处理，转化成C++源程序。具体来说，是将ODPL源程序中直接或间接引用到的所有对象库类的定义（C++代码）取出，作为输出的C++代码的一部份；对在ODPL源程序中定义的对象库类进行转换，产生相应类定义的C++代码，并形成一系列OBMS/IDKE系统函数调用，以便在此程序运行时进行类定义，将有关对象库类定义的C++代码、ODPL代码、函数的二进制代码等写入对象库；利用对象库的信息对对象查询进行优化；用一系列OBMS/IDKE函数调用替换对永久对象（变量），集合等的操纵。

2.利用C++预编译器编译上面得到的C++代码。

3.用C编译器编译上面得到的C代码，得到.EXE代码。

运行上面得到的.EXE代码时，被运行程序首先自动检查本程序中引用的对象库类是否在本程序的源代码最后一次被编译以来又发生了变化。如果确认发生了重大变化，则返回要求重编译的信息并退出运行。

五、结束语

本文总结了近年来数据库系统和模型的发展情况，概述了一个实际的OODBMS系统OBMS/IDKE的研制背景和目标，从模型和编程语言两方面介绍了其突出特点，对O-O技术在数据库领域的应用从理论和实践上作了深入和有益的探讨。

维普资讯 http://www.cqvip.com

参考文献

1. S.Abiteboul and N.Bidoit: Nonfirst Normal form relations:an algebra allowing data restructuring. J. Comput. Syst. Sci., 33:361-393,1986.
2. Malcolm Atkinson, Francois Bancilhon, David DeWitt etc.: The Object-Oriented Database System Manifesto. The first International Conf. on Deductive and Object-Oriented Database, 1989, Japan.
3. S.Abiteboul and R. Hull: IFO: A formal semantic database model. ACM Trans. on Database Systems, 12(4):525-565,Dec. 1987.
4. C. Batini, M. Lenzerini, and S.B. Navathe: A comparative analysis of methodologies for database schema integration. ACM Computing Surveys,18(4):323-364, December 1986.
5. E.F.Codd: Extending the database relational model to capture more meaning.ACM Trans. on Database Systems,4(4):397-134,December 1979.
6. U. Dayal and H.Y. Hwang: View definition and generalization for database integration in a multidatabase system. IEEE Trans. on Software Engineering, SE-10(6):628-644,1984.
7. D.H.Fishman, D.Beech, H.P.Cate etc.:Iris:An object-oriented Database Management System. ACM Transactions on Office Info. Sys., vol.5,no.1,pp.48-69, Jan. 1987.
8. M. Hammer and D.Mcleod: Database description with SDM:a semantic database model. ACM Trans. on Database Systems,6(3):351-386, 1981.
9. R.Hull and C.K.Yap: The format model: A theory of database organization. Journal of the ACM, 31(3):518-537, 1984.
10. H. Jaeschke and H.J. Schek: Remarks on the algebra of non first normal form relations. In Proc. ACM SIGACT-SIGMOD Symp. on Principle of Database Systems, 1982.
11. W.Kim,J.Banejee,H.T.Chou etc.: Composite Object Support in an Object-Oriented Database System. Proc. of the ACM Conf. on Object-Oriented Programming Systems, Languages and Applications, pp.118-125, 1987.
12. L.Kerschberg and J.E.S.Pacheco: A Functional Data Base Model.Technical Report, *Pontificia Universidade Catolica do Rio de Janeiro*, February 1976.
13. P.Lyngbaek, N.Derrett, D.H.Fishman etc.:Design and Implementation of the Iris Object Manager.Proc. of a Workshop on Persistent Object Systems:Their Design, Implementation and Use. Scotland, August 1987.
14. A. Makinouchi: A consideration on normal form of not-necessarily-normalized relations in the relational data model. Proc. of Intl. Conf. On Very Large Data Bases, 447-453, 1977.
15. O.M.Nierstrasz: An Object-Oriented System. Office Automation, Edited by Dionysios Tsichritzis, pp.167-190, Srpinger-Verlag, 1985.
16. ONTOS Object Database Documentation Release 1.5. Ontologic Inc., 1990.
17. Object-Oriented Concepts,Databases,and Applications. ACM press,1989.
18. The Postgres Papers, edited by Michael Stonebraker and Lawrence A.Rowe, 25 June 1987.
19. D.J.Penney and J.Stein: Class Modification in the GemStone Object-Oriented DBMS.Proc. of the ACM Conf. on Object-Oriented Programming Systems,Languages, and Applications, Sept. 1986.
20. D.Shipman: The functional model and the data language DAPLEX. ACM Trans. on Database Systems, 6(1):140-173,1981.
21. Programming in OPAL. Servio Logic Corporation, 1988.
22. The Postgres Reference Manual. Edited by Sharon Wensel,25 March 1988, Electronics Research Laboratory, College of Engineering, U.C.Berkeley.
23. Unidata Inc. : Unidata RDBMS User's Guide. UNIDATA Inc. 1987.
24. Unidata Inc. : UniBasic Reference . UNIDATA Inc. 1987.
25. Vbase Technical Overview. Ontologic Inc., Vbase release 0.8,1987.
26. Wang Shan and Liu Yi: An Object-Oriented Model for Geographic Information Systems. Conference Proceedings of Information Technology:Advancement, Productivity and International Cooperation,the 3th Pan Pacific Computer Conference, Bejjing, August 16-19, 1989. pp552-558.
27. 王珊：面向对象的数据库系统，1990.1.10，计算机世界。
28. 武志文：OBMS/IDKE系统的动态模式定义和模式更新，中国人民大学硕士学位论文,1991。
29. 任永杰：OBMS/IDKE存储系统设计，中国人民大学硕士学位论文，1991.
30. 唐元昌：OBMS/IDKE的模型及其对象操纵，中国人民大学硕士学位论文，1991.