对象管理系统

冯克清、王铁男　　王成耀

（北京信息工程学院）　（北京科技大学）

摘　要

本文简述了对象管理系统OMS的概念及其产生背景，论述了引入OMS的必要性和迫切性。文章重点分析了PCTE中的OMS，包括：1、OMS的系统结构和功能组成。2、OMS的基本要素。主要分析阐述了OMS对象库的结构及其基本元素：对象、连结、关系和属性；数据类型化及类型继承性；数据定义和操纵以及用户如何在对象库中"航行"。3、简要综述了EMERAUDE OMS的功能特点。

在集成化的软件工程环境中，工具集成要求工具之间共享数据且交换信息，同时需要管理不同工具所产生的数据对象之间的关系。在传统意义上，文件和进程间通讯就用于这种目的。实践表明，如何有效地组织环境信息库是实现软件工程环境的技术关键。在过去十年里，工具通过一个共享的数据库通信的思想已经形成。在这种方式下，每个工具可以插入或修改数据，也可以访问其它工具产生的数据。由于有了这种思想和集成环境本身的需求，许多软件研究部门开始考虑满足这种要求的共享数据库的结构及内容，试图定义一种统一的机制以方便存贮和访问，可以认为这正是对象管理系统(Object Management System，简称OMS)产生的背景。

随着"面向对象"概念的产生，如何对对象实施有效的管理受到了广泛重视，一些发达国家在这方面都相继推出了自己的产品与研究成果。然而，在软件工程环境中通过数据库进行工具通信的思想可向前追溯十年。Stoneman报告为Ada语言描述了程序设计的支撑环境，一个商业性的计算机辅助软件工程环境——IDE的图形软件1985年就开始了共享数据库的开发，这一思想引起了广泛重视。在欧洲，首当其冲的要属欧共体ESPRIT计划发起的PCTE (Portable Common Tool Environment) 标准的定义，一个实现它的工业产品EMERAUDE已投入市场，这里都包含了OMS的内容。IBM的一个项目AD/Cycle 也包括了一个支持数据通信的数据库。这种思想使得不同厂商通过数据库集成他们各自的工具，以支持软件开发的整个过程。此外，其它方法也在发展，其共同点是每种方法都有一个基于对象管理的公共界面，这种界面是由程序语言建造的，它使得工具制造者可插入、修改或访问对象。

一、为什么要引入OMS

1、为什么要引入OMS

在集成化的软件工程环境中，需要涉及大量的信息，这些信息不仅类型多，而且具有复杂的联系，同时，软件生命周期各个阶段之间要不断通过相互的软件工具来交换信息，一个工具可能需要另一个工具提供的信息才能有效工作。为了统一管理集成化环境所涉及到的各种信息，加强工具之间的联系，实现软件工具的高度集成，需要定义一种统一的内部信息结构。为了便于管理这种内部信息结构，环境中通常需要建立数据共

享的环境信息库，由该信息库提供统一的工具接口，这样，便可避免在各工具之间建立直接的复杂联系，而且通过信息库，使工具获得必要的输入信息，并随时将操作结果放回信息库中，同时为软件文档的版本控制、查询、追踪和配置管理提供了支持。由此可见，建立统一的内部信息结构是构造集成化软件工程环境的重要途径。

人们曾提出了构造计算机软件的两大课题，一是如何克服复杂性障碍，二是如何将现实世界模型在计算机中自然地表现出来。事实上，这两个问题相互之间有着密切的联系。为克服复杂性，就必须有一种强有力的表达形式。实践证明，现实世界的信息愈复杂，就愈有必要以自然形式将现实世界的结构模型化。仅对记录、字符进行操纵的传统数据库技术无法有效地支持软件开发过程，利用字符流、字节或块进行操作的文件系统则更难以胜任，因此，必须有一种能对多种信息类型进行操纵和管理的信息库系统才能满足集成化软件工程环境的复杂性要求。

七十年代初提出的关系数据模型是面向终端用户的，它将数据的逻辑表示与具体的物理实现分开，设计了标准的非过程的查询语言，已发展为目前应用最为广泛的数据库系统。但关系数据库有许多局限性，不能满足一些复杂应用的需求。

语义数据模型是另一种在许多方面优于关系模型的数据模型，它也是七十年代提出来的，比关系模型的提出稍晚。语义模型具有表示数据结构方面的强大机制，它开始被提出时主要是作为方案设计的工具，一个方案先用语义模型设计，然后转换成通用的数据模型并加以实现。当时强调的是为典型数据库应用中数据间的关系提供准确的模型。

无论是关系模型还是语义模型，在某些特定应用领域，均不能自然、有效地将现实世界模型化。因此，需要建立一种能对各种类型的复杂信息进行有效管理的信息管理系统，由此，产生了所谓的对象管理系统。

2. 什么是OMS

软件工程环境必须管理相当数量的实体，这些实体统称为对象（Object）。对象的概念已被广泛用于软件工程环境领域。对象可以是传统意义上的文件、程序、一个为给定目标机设计的产品、文档、程序库，甚至可以是更为抽象的工程项目。OMS的目标就是以一种统一的方式去管理所有这些对象，即提供一个方便访问的存贮机制，并且管理对象的各种特性及相互关系，特别是，为了支持软件产品的不同版本，以控制对给定对象的更新，OMS必须管理不同版本之间的各种依赖关系。

二、PCTE中的OMS

1. PCTE与OMS

PCTE是欧共体ESPRIT计划的一个项目，旨在为软件工程环境的开发建立基础。在PCTE中，OMS是负责对象管理的一个子系统，OMS数据库（即对象库）是为所有项目、用户及PCTE工具所使用的环境信息的中心存贮区，由构成OMS的一组操作来管理。

OMS的功能包括两个方面：模式描述与数据库操纵，类似于一般DBMS中的DDL和DML。OMS的数据定义语言（DDL）是基于E-R模型的超集，数据操纵由系统提供的数据操纵原语实现，这些原语可由C程序调用。

数据模型用来描述现实世界的实体，OMS的数据模型是E-R模型家族的成员，用来定义组成数据库结构的各种类型的对象、关系和属性，专门用于软件工程环境。

类型是程序设计的基本概念，也是构成程序的基础。在传统的DBMS中，模式是实体型的集合，用来描述数据库的结构和语义特性。在OMS中，模式作为联系工具与相关数据结构的手段。构成总体OMS模式的类型定义被组织成多个模式定义集（Schema Definition Set，简称SDS）。每个SDS是一

个OMS对象，它包含了总体OMS模式的部分视图的描述。对模式的更新总是通过对SDS的更新来实现，在SDS与实际模式之间的映射由OMS模式管理设施自动维护。

然而，项目、用户和工具总是访问它们关心的那部分信息，这就需要建立工作模式（Working Schema）。同可执行程序由一个或多个编译单元组成类似，工作模式由一个或多个SDS组成，通过SDS，在项目级和用户级，OMS实现了模式定义的分散管理，从而并不需要存在一个权威的"模式管理者"。由于工作模式只包含所需的类型定义，它是总体模式的部分视图，象传统的"外模式"或视图一样，因而可以说，工作模式是一个过滤器，它对用户遮盖了实际的数据库结构。工作模式的重要性在于：

·支持项目和PCTE设施中工具及工具集的移植。

·支持现有数据库结构下新工具的集成。

·支持数据库结构的项目（或用户）级定义，如专门的方法学和工作方式。

另外，要保证数据库的完整性，首先要保证对数据的操纵符合工作模式中描述的语义特性，其次要控制在并发活动语境中的一致性。

OMS是传统文件管理系统的进化，它取代了传统的文件系统结构（如UNIX的层次结构），以适应不同环境的需要。需要指出，OMS不同于传统的文件管理系统与数据库管理系统，其主要区别在于OMS适用于描述各种对象类型，它不仅允许应用程序员借助DDL定义新的对象类型，而且可以编制程序以操纵这些对象类型的实例。尽管如此，OMS的概念可被看作是CODASYL网状模型的扩充，支持不同的数据定义级（内模式、概念模式与外模式），借助路径名的描述对对象实施操纵。

2. OMS的基本要素

（1）对象、连结、关系、属性 OMS的对象库是一个信息项的网状模型，其基本元素是对象（Object）、连结（link）、关系（relationship）与属性（attribute），这种数据抽象是对系统的简化描述，它强调了表达方面某些共有的特性，而把不重要的信息予以忽略。抽象在程序设计语言和设计方法学的发展过程中一直起着主导作用，可以说，程序设计语言的发展过程就是抽象级不断提高的过程，整个软件工程的进展也是与抽象程度的提高紧密相关的。

1）对象　对象是可明确标识的实体，如源文件、工具、程序库、项目等。对象具有下列特性：

·相关属性集

·相关关系集

·内容

2）连结　连结是源对象与目的对象之间的单向联系，用来建立从一个对象到另一个对象的简单参照。连结是构成路径名的基本元素。

3）关系　关系表示两个对象间的相互依赖，它由一对连结组成，其中一个连结的源对象是另一个的目的对象，关系（A，B）可看成是两个单向连结（A→B）和（B→A）的组合。

4）属性　属性可应用于对象、连结和关系，以表示对象、连结或关系的内在特性。

（2）类型化　客观世界的事物是多种多样、千差万别的，为了有效地表示这些事物，需要将它们加以区分、加以组织，以抽象出共同的东西。按照事物所固有的特性、行为或用途，在某个领域内将其分类，这就产生了类型。再将各种类型加以严格的形式化定义就产生了类型系统。所以，类型化即是以共同的特性和统一的行为将一个可计算的世界分解成多个子集，在同一子集中的所有成分具有相同的特性和行为，这些子集是可以区分的，因而形成了不同的类型。

把一个系统加以类型化，可以减少表示上的麻烦。类型化实质上是给系统以某种限

制，这有助于正确性的加强，可以防止基本逻辑单元之间交互作用的不一致性，这对程序设计具有极其重要的意义。

由于新类型的定义可以使用其它类型的某些特征，这就形成了类型的继承性。继承性是类型的合成机制，其前提是相同的对象具有相同的特性，将具有相同特性的对象组合在一起，在面向对象的语言中称为“类”(class)。由于同一类对象具有相同的操作，从而也具有相同的行为。OMS是对对象实施管理的，因而有必要引入类的概念。当然，类的实现方法会因设计者的不同而有所差异。通常将具有相同特性和功能的事物归入一类，如在面向对象的语言中，常常将数据结构及所能实施的操作均相同的对象看作一类。一个类可以继承其父类的操作和特性，而该类本身的操作和特性又可为其子类继承。利用类及其继承机制，可以生成新的对象，这无疑有利于资源共享和软件重用，便于维护和管理。

为了支持对象库的类型结构，对象、连结、关系和属性都应具有相应的类型。对象、连结、关系和属性的建立来源于一个事先存在的类型定义。

1)对象类型　在OMS中，每个对象属于一个特定的对象类型(object type)，如源文件、程序库、工具等。对象类型并不是相互分离的，它们可以共享公共特性。例如，所有对象都有定义其访问权限的属性，所有对象均享有其公共祖先类型“对象”所具有的最小公共特性集。按照继承机制，所有的对象类型构成了类型层次，一个对象类型可定义成另一个的子类型，一个事先定义的对象类型作为根，在层次上的子类型继承了其父类型的所有特性。例如，设T_1是对象类型T_2的子类型，则T_1具有T_2的所有特性外加一些新的属性类型和连结类型。

一个对象类型由下列要素组成：

- 名字
- 父类型
- 一组属性类型
- 自身为源的一组连结类型
- 自身为目的的一组连结类型。

基本的对象类型有文件、管道、消息队列、块设备、字符设备等，任何一个新的对象类型均是从这些基本的预定义类型派生出来的。

对象的内容与其类型有关。对于六个预定义的基本对象类型，均有其专门种类的内容，如“文件”的内容是一个文件，“管道”的内容是一个命名管道，等等。基本对象类型的子类型与其父类型具有相同种类的内容，而其它对象类型则没有内容。对象的内容由PCTE提供的操作来管理，如对文件的内容可施行打开、关闭、读、写等操作，这些操作不属于OMS的范畴。

2)连结类型　连结类型是其所有实例(即连结)的定义，它由下列几部分加以描述：

- 名字
- 基数。基数用来定义可始于一个源对象的相应类型的连结个数，其数值可以取“1”或“n”。若连结类型的基数是“1”，则从一个指定对象只能发出这种类型的一个连结；若基数为“n”，则可有该类型的多个连结始于一给定对象，在这种情况下，必须使用一组排序的关键字属性，以区分具有同一源对象的同类型的不同连结。
- 范畴。范畴可分为以下三种：

i) 组成连结(Composition Link)。对于一个组成连结，其目的对象是源对象的组成部分。当一个对象建立时，其组成连结便被同时建立。数据库中每个对象至少是另一个对象的组成部分，因此，当一个对象不再是任一对象的组成成分时，该对象便被删除。

ii) 参照连结(Reference Link)。参照连结可指向任何对象，只要参照连结存在，其目的对象就不能删除。参照连结由程序建立，不能属于组成连结类型。

iii) 隐含连结 (Implicit Link)。隐含连结可说明为组成一关系类型的两个连结类型之一。隐含连结的建立与删除是当其反向连结建立或删除时由系统自动完成的

• 稳定度。当建立一个连结时，必须说明其目的对象的稳定度。稳定度可分为下列二级：

i) 只要连结存在，其目的对象不可删除，但可修改。

ii)只要连结存在，其目的对象不可删除也不可修改。

• 一组源对象类型和一组目的对象类型

• 一组属性类型

• 关键字（除基数为"1"之外）。对于基数为"n"的连结类型，必须定义一个能标识目的对象的关键字，以区分不同的实例。关键字由目的对象的若干属性类型组成。

3)关系类型 一个关系类型是由两个连结类型定义的，其中一个连结类型的源类型是另一个的目的类型。关系类型的功能不是命名一对连结，而是在两个连结类型间强制一种完整性约束。属于一个关系类型的两个连结类型不能都具有"组成范畴"，不能都是隐含的。与基数相关，连结类型可以是一对一、一对多或多对多的。

4)属性类型 属性类型的定义由属性名、值类型和初始值组成，可用于对象类型及连结类型，其中，值类型可以是整型、布尔型、日期型或字符串型。初始值必须与相应的值类型一致，它可以显式设定，亦可以取 OMS 提供的预定义的缺省值。每个对象类型都隐含地具有一组公共系统属性类型，主要涉及对象的物理标识、存取控制等。

(3) **数据定义与操纵** 在PCTE 的OMS 的规范中，提出了一个数据定义语言(DDL)，根据这个语言，一组对象、连结、关系和属性的类型定义有组织地放在了SDS中。每个SDS由一组相关的类型定义所组成，例如，一组专用于特定用户的定义，一组专用于给定工具的定义，一组对工作在给定项目上的所有用户所必需的定义。这些定义被系统用于：

• 当对象或连结建立时，初始化各种属性。

• 根据OMS数据空间的语义特性，保证操作的正确性。

• 当翻译请求 (Request) 时，帮助推导某些信息。

每个SDS 是完备的、一致的和自包含的 OMS类型定义空间的子集。SDS 是 OMS 的对象，因此可按照OMS 的命名规则访问。此外，每个SDS有一个唯一的系统标识符。一个给定的类型定义可属于多个SDS，使得用户或工具之间可共享公共信息。

SDS是构成进程工作模式的基本元素，用户可通过用户工作模式的环境来操纵对象库。工作模式是多个SDS 的有机组成，因而代表了对象及关系特性的约束集。由于不同的用户可在不同的工作模式下访问同一对象，因而每个用户具有不同的对象视图，一个进程可继承其父进程的工作模式，它可能由 shell建立，也可能由建立模式原语动态地建立。

对象、关系、连结及属性的建立与存取由数据操纵操作实现，OMS并不提供所有的数据操纵操作，如对象的内容更新、数据的物理分配等操作是由其它PCTE 子系统提供的。

(4) **在对象库中航行** 在软件工程环境中存在着众多的对象，它们之间的相互作用是通过消息传递完成的。那么 OMS 是如何完成消息传递的呢？

OMS 管理对象的主要手段是通过对象之间的连结"航行"对象库，即从一个特定的参照对象开始，借助于一组连结序列（称为路径名），最终到达所要的目的对象。

在分布式的对象库中，事先定义的参照对象可有以下几种：

• 公共根参照对象：公共根起源于网络的主机上。

• 局部系统参照对象：通过主机的初始化进程建立。

• 静态语境参照对象：在开发程序运行的静态语境中，它是在一个进程启动时隐含建立的，不能显式修改。

• 用户自己的参照对象：这个对象是在装入时按照口令文件建立的，它同每个用户相联系，其类型也不是事先定义的。

3. EMERAUDE OMS的功能特点

按照PCTE标准，一个实现产品EMERAUDE所提供的OMS具有下列功能及特点：

(1) 对象库是基于网状模型的，对象是网上的节点，关系是对象间的一对连结。作为一个集成化的软件工程环境，需要涉及大量的对象，且对象之间具有复杂的联系，网状模型正适合于复杂关系的表达。

(2) 对象库是强类型化的，库中的各类对象、关系及属性的类型定义保存在SDS中。类型化是数据的抽象，如果没有类型，就没有继承机制，这样，对象中数据和操作就可能出现大量重复。继承性符合软件重用的目标。

(3) 用户能够修改对象库，可建立和删除对象，但要严格符合SDS中的类型定义。

(4) 用户能够在对象库中航行，从一个指定的对象开始，通过设定的路径名完成。路径名在语法上是UNIX的超集。

(5) EMERAUDE OMS具有透明的分布式结构。

对象管理系统体现了一种基于抽象数据类型和面向对象程序设计的新思想，它是以数据库技术、分布式技术为基础的信息管理技术，因而成为集成化软件工程环境的一个重要组成部分，起着核心的作用。面向对象的思想较传统的程序设计思想更接近于人的思维，面向对象的数据库较传统的数据库具有更强的表达能力和操纵能力，特别适合于管理"演化"系统，而这正是整个软件开发过程所需要的。人们正逐步采用这种新思想进行软件工程环境的构造，OMS的标准和实现方法正成为人们研究的重要课题。

参考文献

1. PCTE 1.5/C
2. The Emeraude Environment, written and produced by Syntagma Systems Literature, Truro, UK. printed by Cornwall Litho, Redruth, UK.
3. 面向对象计算的现状和展望，闵华译自日刊《情报处理》1988，作者：米泽明宪，《计算机科学》，1990.1
4. 软件工程环境的集成模型及其应用，田捷、薛明，《计算机研究与发展》1990.1

（接第55页）

某一事件的确定程度，或一个系统的总体情况。但在实际环境中，证据所表现的作用是多种因素的综合，数值法把所有支持的和反对的证据组合在一起，用一个数字表示其效果，当一个推理者希望区分不同因素的作用时，这个组合后的数字就不适用了。数值法所表示的不确定信息是隐含的，其语义是不清楚的，而推理要求直接显式的知识，这些都是数值法的不足。近年来发展起来的非数值的方法，如认证论[10]，非单调逻辑[11]等等，企图在这些不足之处能加以弥补，限于篇幅，不在本文介绍了。不管怎样，对不确定性信息的表示和处理，仍然是一个有待发展的课题，所以，数值的方法和非数值的方法，将并行地发展下去，而应用则是根据域的知识特点和解题的需要，选择合适的表示和处理方法。正因为如此，最古老的概率法，最近有人认为它是最值得拥护的方法[12]。（参考文献略）