RSOODM—一种引入联系的面向对象的数据模型

董继润　李庆忠　（山东大学计算机系，济南）

摘　要

In this paper, we describe the semantics and concepts of RSOODM in a formal way. RSOODM not only includes the traditional object-oriented concepts, but also introduces two kinds of relationship types. One is called Association Relationship type, another is called Pipe Relationship type. The Association Relationship type adds an intermedium type between different object types, and enhances the semantic relationships and constraints between them. The Pipe Relationship type adds an interface between different object instances instead of message passing. Data can flow into or out of objects. It enhances relationships of data between different objects, and enhances the concepts of derived attributes.

一、引　言

最近几年，面向对象的数据模型日益得到人们的普遍重视，主要原因是面向对象的数据模型为人们提供了表达信息的高度抽象方法，和人们对信息的看法极为接近。面向对象的模型主要用于复杂的应用中，例如计算机辅助设计(CAD)/制造（CAM），办公信息系统(OIS)，人工智能（AI）及计算机辅助软件工程（CASE)中。目前已经出现了许多面向对象的数据模型，这些模型都是基于某些特定环境提出的，例如Gemstone[5,6]采用对象/消息范畴，对象由消息激活。Vbase[7]采用抽象数据类型范畴，运用对象的方法（method）执行所需要的操作。ORION[1]也采用了对象，类，消息及复杂对象的概念。在IRIS[2]中，属性，方法，约束都是用数学函数来定义的。所有这些模型都是简单地将数据和操作封装在对象中，只能通过传递消息建立两个不同对象的接口，并且不支持不同类对象之间的关联联系，从而缺乏不同对象之间的语义关系。

在RSOODM中，不但有传统的对象，而且还有两种联系类型：第一种是关联联系，它描述多个不同类型对象间的关联约束；第二种是通道联系，为两个对象（可以同型，也可不同型）间建立起内部通道，充当两个对象的接口，两个对象的数据可通过通道联系流出和流入对象。本文为这种数据模型提供了一系列形式与非形式的描述。第二节描述本模型的基本概念，第三节描述关联联系类型，第四节论述了通道联系类型。

二、RSOODM的基本概念

这一节主要讨论本模型有关对象的形式化定义。

定义2.1　一个面向对象的数据模型Θ，由对象类型集合$\{O\}$和联系类型集合$\{R\}$组成：

$$\Theta=\{O\}\cup\{R\}$$

定义2.2　用户定义的对象用来建立一个机构的概念实体模型，并分类成为对象类型。一个对象类型是以下形式：

$$O=\{A_0,\ M_0,\ C_0\}$$

其中A_0表示一个非空的属性集合；M_0表示操作集合；C_0表示完整性约束。A_0，M_0，C_0被称为O的特性，并封装在O的内部。

```
Object Type student
    Attributes
        ssn          : string[10];
        name         : string[20];
        birth_date   : Date;
        age          : compute_age;
        advisor      : professor;
        semester     : 0.. 4;
        course_taken : set of ref course;
    Operations
        compute_age:
            age: =today - birth_date;
    Constraints
        max_semester:
            if semester>=4 then
                write("You must complete");
end;
```

图2.1 对象类型student的表示。

定义2.3 属性A_0由下列组成：

A_0::=｛〈基本属性〉|〈机器内部属性〉|〈元组属性〉|〈引用属性〉|〈集合属性〉|〈导出属性〉

定义2.4 基本属性表示一系列基本属性类型：$\rho \cup \xi$。这里ρ是基本属性集合，ρ={integer, real, charater, boolean}；ξ是结构化的基本属性类型，ξ={string, array, file, subrange}。它们都称作基本对象类型。

在图2.1中，对象类型student的ssn, name, 等都是具有基本对象类型的属性。

定义2.5 机器内部属性类型指模型指定的内部定义属性类型，如图2.1中的birth-date:date，date就是机器内部属性类型。

定义2.6 元组属性类型指以以下方式构造的类型：

〈属性名〉:tuple

〈n_i〉:〈t_i〉

｛〈n_j〉:〈t_j〉

end tuple

n_i，n_j是属性名t_i，$t_j \in \rho \cup \xi$。

例如有一个元组类型属性Items。

```
Items:tuple
  item-name:string[10];
  quantity:integer;
  unit-price:real;
  and tuple;
```

定义2.7 引用属性包括下列形式：

$O_i \cdot a : f\ \Theta$

$\Theta = \{O_j\} \cup \{r_k\}$；$O_i$和$O_j$是对象类型，$r_k$是联系类型，$O_i \cdot a$是$O_i$的属性$a$。

f::=ref |exclusive ref| dependant ref |own ref

• ref表示没有任何约束的引用，即一属性的域是另一对象类型。

• exclusive ref表示被引用的对象不能被两个或多个对象通过own ref属性来引用。

• dependant ref表示当引用对象被删除后，被引用对象也随之被删除。

• own ref相当于exclusive dependant ref，意思是这样的对象，它被一个对象引用后，就不能被另外对象引用了，并且引用对象被删除后，被引用对象也随之被删除，与ORION[1]中的复杂对象(Composite object)的定义类似。

例如对象EMP表示职工，某中有一个属性为children，表示子女：children:set-of own ref person。一个子女只能被其父亲引用，因而只能被一个对象引用，当引用它的对象（父亲）被删除后，父亲所引用的所有子女对象也随之被删除。

定义2.8 集合属性类型是某个对象类

型的集合：$O_i \cdot a$:set-of f O_j。其中O_i，O_j是对象类型，$O_i \cdot a$是O_i的属性a，f是引用语义。

定义2.9 导出属性类型。属性的域是一个用户定义的操作，它的值由用户定义的操作决定；实际上导出属性有两类：一类可由对象内部数据导出，如student 中的age由操作Compute-age导出；另一类还依赖于其它对象，不但由本对象中数据而且需要其它对象的数据一同导出，这类导出属性将在通道联系中详细描述。

定义2.10 操作类型M_0：

M_0::=〈内部操作〉|〈用户定义的操作〉

〈内部操作〉::={插入，修改，删除，检索}

另外用户自己还可以定义操作，例如对象student中compute-age是用户自己定义的操作。

定义2.11 数据约束类型。

〈数据约束类型〉::=〈内部约束〉|〈用户定义约束〉

〈内部约束〉::=〈数据操作〉is〈约束类型〉

〈数据操作〉::=插入|删除|修改

〈约束类型〉::=级联|限制|置空。（见图2.2）。

	级联	限制	置空
插入	×	如果被新对象引用的对象一个也不存在则拒绝操作。	如果被新对象引用的对象不存在，那么引用对象的值置空。
	删除所有被引用对象	如果被删除的对象被别的对象所引用，拒绝操作。	该引用对象置空。
	×	如果被修改的被引用对象不存在，则拒绝操作。	如果被修改的被引用对象不存在，则将值置空。

图2.2 约束模式定义

定义2.12 继承。是一个层次图，在这个层次图中，如果x在y的上方，我们说x是y的超类型(suppertype)，y是x的子类型(subtype)，如果y是x的子类型，这意味着y支持在x中定义的所有特性，并且y可有另外定义的特性表达形式见图2.3。

```
Object Type person
        Attributes
          ssn: string[10];
          name: string[20];
          birth__date: Date;
        Operation
          [略]
      end;
Object Type student Subtype of person
        Attributes
            dept: ref department;
            course__taken: set—of ref course;
      end;
```

图2.3 子类型定义

三、关联联系

传统的面向对象数据模型对于对象的本身研究不够，没有足够重视对象间的相互关系，对象单纯靠其属性去引用其它对象，反映不了它们之间的语义联系。

例如有一个对象类型student表示学生，另一个对象类型course表示课程。如果要描述某学生选修了某门课程并且具有一个成绩，单用一个学生对象和课程对象是表示不出来的，这时要引入一个关联联系类型Select Course（见图3·1）。

在图3·1中，对于没选课的学生对象的属性course-taken的值为空，没有被学生选修的课程对象的属性takenby的值为空。

对于每个选课的学生对象S，课程对象

```
Object Type student
    Attributes
      ssn          : string[10];
      name         : string[20];
      birth__date  : Date;
      course__taken: set-of ref SelectCourse;
  end;
  Object Type course
    Attributes
      cn       : string[2];
      name     : string[20];
      taken__by: set—of ref SelectCourse;
  end;
  Association Type SelectCourse
    Attributes
      st  : ref student;
      crs : ref course;
      score: integer;
  end;
```

图3.1 关联联系类型实例

C和选课关联联系实例sc，sc是$S.course\text{-}taken$的一个成员，并且sc也是$C.takenby$的一个成员，当且仅当$sc.st$是S，$sc.crs$是C。

实际上这是一个相互引用的语义关系。

定义3.1 关联联系类型R是这样的一种类型：$R=(A_r, M_r, C_r)$，其中A_r是属性集合，M_r是操作集合，C_r是约束集合。

定义3.2 在A_r中有K个关联属性，$R.a_i:\text{ref}\Theta_i(i=1\cdots K)$，其中$R$是关联联系类型，使得在对象类型$\Theta_i$中有一属性$a_{ij}$，$\Theta_i.a_{ij}$:set-of ref R；对于每一个对象类型Θ的任意一个对象O_{in}，$O_{in}.a_{ij}$的值或空或为：对于每个O_{in}，有一个关联类型R的实例r，使得r是$O_{in}.a_{ij}$的成员，当且仅当$r.a_i$是O_{in}（$i=1, \cdots, K$）。我们称O_{in}为参与对象。

```
Pipe Type TIME__DEP MULTI white
      Transmits
       / * expected completion time coming into projects * /
      exp__compl: time to black;
    end;
Pipe Type TIME__NEED MULTI black
      Transmits
       / * expected completion time going out of project * /
       exp__compl: time to black;
    end;
    / * Object to simulate many-many relationship for
      project dependences * /
Object Type MANY
      Pipes
       from__obj: black TIME__NEED;
       to__obj: white TIME__DEP;
      operations
     to__object. exp__compl: =from__obj. exp__compl;
    end;
Object Type PROJ
      Pipes
        depends__on: black TIME__DEP;
        needed__by: white TIME__NEED;
      Attributes
        pno: integer;
        local__work: time;
        expected__time: derv__time;
      Operations
      derv__time;
        expected__time: =local__work+
          iterator lasted: time
           init 0
           for each dep in depends__on do
             latest: =later__of (lastest, dep. exp__compl);
       pass__time;
           needed__by. exp__compl: =expected__time;
  end;
```

图4.1 通道联系类型实例

在图3.1中，Select Course是否可定义成为对象类型呢？答案是肯定的，但为什么要设置关联联系类型呢？理由是：

(1)对象属性的值可以空，但关联的属性值不能空。

(2)关联属性对参与对象的引用不能是集合，而对象的属性可以是集合。

(3)当一个参与对象被删除后，相应的关联类型实例紧接着被删除。当一个关联类型实例被删除后，参与对象只是被修改，这种语义不能用对象的exclusive，dependant ref语义表示。

由此看来，引入关联联系类型加强了不同对象类型之间的语义联系。

四、通道联系类型

传统的面向对象数据模型只是将数据及操作封装在对象里面，实现对象间通讯是利用消息进行的。这样就不能将这两个对象间的语义联系有效地描述出。在我们的模型中，数据和操作称为对象的内部特性，我们还要定义对象的外部接口—通道联系，使得一系列数据经通道联系从对象中流入流出。

引入通道联系后，增强了导出属性的特征，因为对象中的某个属性不但受本对象内部数据的约束，而且还要受到其它对象中某属性值的约束，这时，引入通道联系后，导出属性的值可以通过通道联系导出。

通道联系和关联联系不同，关联联系描述不同对象类型之间的语义关系，而通道联系描述两个对象间的通信联系。

通道联系是数据流入流出对象的外部接口，因而是有方向的。我们定义了两个口：白口(White)和黑口(Black)。数据可以从黑口流向白口，也可以从白口流向黑口。如果定义了一个通道联系类型，一个对象类型便可定义拥有这个通道联系类型的黑口或白口。两个对象若能发生联系，当且仅当一个对象拥有指定通道联系类型的白口，另一个对象拥有黑口。

通道联系可以是一对一，一对多的。多对多的联系则需转换成两个一对多的联系。

定义4·1 通道联系类型P是这样的类型：$P=(t_A, r)$。其中t_a表示传送属性，形如：$O_i \cdot a : O_j$ To 〈接口〉。它将域为O_j的对象O_i的属性a的值传送到黑口或白口，r表示联系类型，即通道联系表达的对象和对象间的关系，包括一对一，一对多联系、形式为〔MULTI〕〈接口〉。MULTI表示接口联系是一对多的。

例如，在软件工程中一项工作，其工作对象类型为PROJ，其中有一个属性local-work表示这项工作单独完成所需时间。另一个属性为expected-time表示所期望的完成时间，这一预期完成时间依赖于其它工作的完成时间，因此改变某项工程的完成时间，可能要影响到整个工程的完成时间，于是，expected-time的值必须从其它对象中导出，见图4·1。

在图4·1中，derv-time是导出属性类型，并且expected-time的值通过通道联系由derv-time导出。

五、结束语

本文用一种形式化的方法描述了一个新的面向对象的数据模型，此模型的提出，为建立一个面向对象的系统提供了理论依据，使面向对象的系统不仅能支持对象类型，而且支持两种联系类型。（参考文献略）

计算机科学

第5期(双月刊)
1991年10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