面向对象 数据库 管理信息系统 概念模型 (11)

计算机科学2000Vol. 27№. 3

40-42,34

面向对象与MIS概念模型设计*)

Object-oriented and MIS Conceptual Modelling

郭胜辉 孙玉芳 TP311.13 TP399

(中国科学院软件研究所 北京 100080)

Abstract The paper considers the MIS conceptual modelling using the object-oriented techniques, and presents the behaviour and structure of objects. The paper also illustrates the concepts, tools and techniques for the design and specification of object behaviour and structure by the insurance MIS instances

Keywords Management information system, Concept modelling, Object, Object behaviour

1 引言

为了降低软件开发和维护的代价,计算机科学家试图通过研究程序设计语言、软件工程和数据库管理原理来提供一些工具和技术解决日益增长的软件复杂性[1]。这些技术中有一些是基于抽象原则和支持逐步求精的软件开发方法。

在数据库设计中,评价设计的一个主要标准就是可理解性[2]:能否使用户、数据库设计者和数据模型的实现者理解数据库的结构?概念数据模型起着用户与数据库专家之间的桥梁作用,因此与工程有关的三方都能理解数据模型尤其重要,总而言之,工程最终的成功依赖于数据模型的准确性。

在从用户的要求到数据模型构造之间转换时可能出现一些错误,然而,错误也可能出现在数据库专家对数据模型的理解上,这种问题的出现是因为构模者可能不对其设计的数据模型的具体实现负责。因而,数据模型的实现者必须能读懂和理解概念数据模型。本文的重点是如何定义和构造一个可理解的、易实现的数据模型,介绍如何利用对象构模技术对MIS进行概念模型设计。

2 如何定义对象类

数据库应用既有结构特性也有行为特性。结构指状态和静态性质(即实体和它们之间的关系),行为指状态的转变和活动性质(即操作及它们之间的关系)[3]。一个数据库应用的完整的设计和细化必须包括结构和行为两方面。例如,对于保险信息管理系统的投保单录入过程的部分语义模型通过图1的关系 policy-reservation (reservation #, agency, policy-type, assured, arrival-date, depature-date)表示其结构特性。对于修改 policy-reservation 的活动的语义有诸如插入投保单(insert-policy-reservation)、删除投保单(delete-policy-reservation)和修改投保单(update-policy-reservation),它们都需要准确定义。而录入一组投保单(make-group-reservation)和登录(convert-reservation-to-registration)等活动可用于设计更复杂的操作。数据库设计要求有结构构模和行为构模的概念、工具和技术。

利用面向对象的方法,分析员可通过定义一系列具有属性和行为的对象来构造模型。行为是指对象的内部操作和消息,即对象数据的管理和向别的对象请

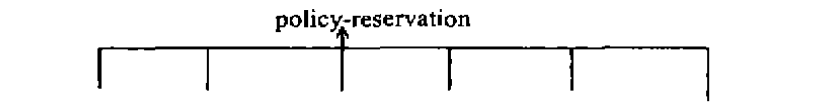

图1 投保单的部分语义结构模型

*)本文得到国家863-306-ZD-07-4项目的资助。郭胜辉 硕士,主要研究领域为数据库和网络管理信息系统。孙玉芳 研究员,博士导师,软件所副所长,主要研究领域为操作系统,中文信息处理,大型数据库和网络应用系统。

求服务。对象分成具有共同性质的对象类，组织成层次结构，子类继承超类的包括数据定义和行为的特性。对象之间的相互作用通过消息发送进行控制。关于对象的定义和图形表示的方法在文[4,5]中有详细的论述。在此仅补充两点：一是引入联合的概念；二是给出在分析和设计中经常用到的聚集(aggregation)、归纳(generalization)和联合概念的谓词描述。

所谓联合是把成员对象之间的关系看作一种更高级的集合对象的抽象形式，亦即"是一个成员"的关系，例如，经理的集合是 agency 成员的联合。又如，在保险应用中，我们常常要登录一批保单，而所谓一批保单就是一些保单的联合。

聚集、归纳和联合可用下述谓词演算和集合论概念简单明了地表达(当 P 是一个谓词，令集合 Ps 记作 $\{x|P(x)\}$)：

聚集：令 A 是含有成份 $P_i(i=1,2,\cdots,n)$的聚集，则

$$A(x)\Leftrightarrow\exists\ y_1,y_2,\cdots,y_n.P_1(x,y_1)\&P_2(x,y_2)\&\cdots\&P_n(x,y_n)$$

归纳：令 G 是类 C_i 的归纳，则

$C_i(x)\Rightarrow G(x)$，$x\in C_{i_1}\Rightarrow x\in G_i$ 或 $C_{i_1}\subset G_i$；

联合：令 S 是集合类型，且 M 是成员类型，则

$S(x)\Leftrightarrow\forall\ y(y\in x\Rightarrow M(y))$，$x\in S_i\Leftrightarrow x\in P(M)$或 $x\in S_i\Leftrightarrow x\in P(M)$

这里 P(x)是 x 上的幂集。

3 对象行为的定义

就数据库应用的行为特性的设计和细化来说，对象方法提供了关于对象的基本数据库操作，组成面向应用的操作的三种控制抽象形式(顺序、选择、重复)和两种过程抽象(概念模型的动作抽象和事务模型的事务抽象)。

每个数据库操作是关于对象类的一个实例的修改操作或选取操作。修改操作有插入(INSERT)、删除(DELETE)和修改(UPDATE)。修改操作确保每个对象所扮演的角色的语义行为(例如，只有当聚集类实例的每个成份存在时该实例才能被插入、删除一个归纳类实例导致依赖于它的特类实例被删除)。

控制抽象用于使一些操作相互联系并构成更高级的复合操作。三种形式的控制抽象(顺序、选择、重复)是三种形式的数据抽象的模拟。聚集相应于一些操作的顺序或平行关系，一个聚集的操作由它的每一个成份的顺序或平行操作组成。例如，对于某张投保单在某日的关于 policy-reservation 的动作 insert-policy-reservation 由操作序列(创建投保单号(reservation #)，得到被保人(assured)，投保一个险种(policy-type)等，加上插入这张投保单)组成(图2)。在此，reservation # 由系统自动生成，agency(代理人号)和 policy-type(险种)对象独立于投保过程而存在，assured(被保人)、arrive-date(投保日期)、depature(登录日期)要在投保过程中得到。类似地，我们可设计删除投保单(delete-policy-reservation)的动作(图3)。在本文的图里，符号"→"表示一个操作的性质，来自请求这个操作的对象(即消息发送对象)，指向由这个操作请求的对象(即接收消息的对象)；双箭号⇒用于区分动作或事务与它的操作。

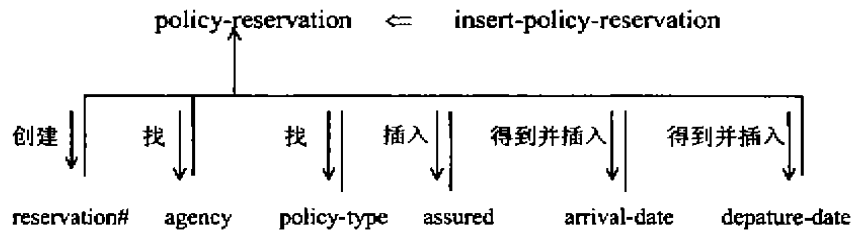

图2 插入投保单动作

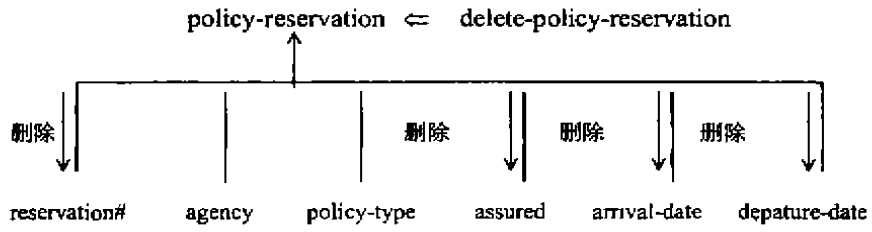

图3 删除投保单动作

一个关于一般类的操作由对每一个特类的一个选择操作组成(case 用于不重叠的特类;if-then 控制结构的顺序或平行操作用于重叠的特类)。故归纳相应于选择(例如:if-then 或 case 控制结构);联合相应于重复(例如 do-while 或 foreach 控制结构)。一个关于集合的操作由应用于集合中每个成员的顺序或平行操作构成。

概念级的行为构模包括每一个对象动作的定义、设计和细化。一个动作由一个数据库的修改操作和提供必需的相关操作来定义。在请求数据库操作之前,某些前提条件需满足且其他相关的对象的动作可能是必须的;动作请求后,事后条件必须被检查且数据库操作被执行。动作提供了修改一个对象的唯一手段。一个动作的局部作用范围包括与其直接相关的对象和所有通过聚集、归纳和联合的相关对象;一个动作的整个作用范围由请求动作的层次推断出来。

对象的行为特性和结构特性组成了定义对象语义的一个抽象。对象分析的结果得到对象的层次结构和动作的层次结构。

事务级的行为构模包括事务的定义、设计和细化。事务是一个面向应用的操作,能修改一个或多个对象。事务设计成完成特定用户的要求。例如,事务 cancel-reservation 包括检查"投保单"存在;请求动作 delete-policy-reservation 且确信"投保单"被删除。为了表示事务和其子动作之间的关系,引入事务对象,并与其作用范围内每个对象联系。图4中的事务是其作用域里的对象的一个聚集。图4的事务 *convert-reservation-to-registration* 用于从得到的投保单 policy-reservation 创建一张正式保单 policy-registration。前提条件是检查 policy-reservation 存在"投保单"信息,且确保该投保单可登录。事务然后创建、找到、请求登录信息,且请求动作 insert-policy-registration 和 delete-policy-reservation。事后条件检查动作成功与否。事务设计是抽象的,即不考虑事务所涉及的动作细节。

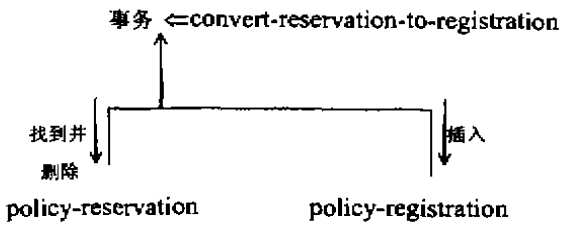

图4 投保单登录事务

4 行为细化:谓词细化

动作和事务有类似的组成部分:

N(V),I,O,L:P{A}Q,D。

其中:N 为操作名,V 为参量表。I,O,L 定义作操作域(动作的局部作用范围和事务的作用范围):I 为通过动作或选取操作存取的对象;O 为通过数据库操作 D 修改的对象;L 是局部于这个操作的对象。操作结果由前提条件 P,活动 A,事后条件 Q,可能的数据库操作 D 细化。P,Q 为谓词表,A 由0次或多次对象的操作组成。

谓词行为细化,有如下形式:

N(V),I,O,L:P{}Q,D。

动作或事务的结果完全由 P{}Q 定义。活动 A 定义了结果怎样得到,因此这里排除作为细化的不合适的细节。操作的结果利用 P,Q 中的谓词细化,这些谓词可能是简单的、抽象的、半形式的,易于细化和修改。

由图2的动作图可导出图5的动作 insert-policy-reservation 的细化,类似地,可得到图6和图7,现在解释如何对谓词进行细化,例如:对图6中的 P 细化有:policy-reservation-exist(n):某个记录 h 存在于 policy-reservation(h. reservation=n)

动作和事务的程序设计语法为:

动作:(动作名)((参量表))

I:((对象表))

O:((对象表))

if (P) then (请求活动 A){else 终止请求}end if

if (Q) then (数据库操作 D){else 取消请求并终止动作或事务}end if

例如,对图2的动作可编制图8的程序。

```
动作:insert-policy-reservation(g,p)
  I:(n:reservation#,g:agency,p:policy-type,A:assured,a:
    arrival-date,d:depature-date)
  O:(policy-reservation)
  P:policy-exists(p)?,legal-assured(A)?
  Q:policy-reserved(n,g,p,a,d)?
  D:insert-policy-reservation(n,g,p,A,a,d)
```

图5 插入投保单动作细化

```
动作:delete-policy-reservation(n)
  I:(n.reservation#,p:policy-type)
  O:(policy-reservation)
  P:policy-reservation-exists(n)?
  Q:policy-available(n,g,p,a,d)?
  D:delete-policy-reservation(n,g,p,A,a,d)
```

图6 删除投保单动作细化

```
事务:convert-reservation-to-registration(n1)
作用域:(n1:reservation#,n2:policy-registration)
  P:reservation-verifies(n1)?,registration-available(n1)?
  Q: policy-reservation-canceled (n1)?, policy-registration-
     exsts(n2)?
```

图7 登录保单事务细化

(下转第34页)

计算机科学2000Vol.27№.3

Java芯片系统存储管理的设计和实现*)

Java Chip System: the Design and Implemetation of Memory Management

秦莹 陈虎 戴葵 杨晓东

(国防科技大学计算机学院 长沙410073)

Abstract Java chip system is a platform of Java applications, which consists chip and its operation system. Java chip implements JVM(Java virtual machine) instruction set, while os manages hardware such as memory, I/O etc, and classloader, thread manager are also included. Memory management is one part of chip system, it not only supports the execution of Java instruction, but supplies the basic support to the Java chip system. In this paper, we introduce the design and implementation of memory management in Java chip system.

Keywords Java chip system, Memory management

Java是一种面向对象、通用、支持并发的程序设计语言，自1995年推出以来，其应用领域已经从主要面向Internet转向大型分布式软件和嵌入式系统。Java语言源程序(*.java)并不基于任何一个平台，而是基于一台抽象的机器——Java虚拟机(Java virtual machine—JVM)。JVM有自己的一套指令集并且使用独立的存储区域。编译后的代码称为类文件(*.class)。有相应的Java虚拟机规范、定义虚拟机指令集、类文件的格式、Java虚拟机的实现等标准，确保了类文件的可移植性。

现有的JVM多在宿主机操作系统上用软件实现，解释执行Java虚拟机指令。与软件实现JVM不同，Java芯片系统用芯片直接实现Java虚拟机指令集中的大部分指令，并在系统软件的配合下支持Java应用程序运行。它主要面向嵌入式系统，在当今网络计算机上也有较好的应用前景。

存储管理是Java芯片系统一个重要组成部分，其主要任务是分配和回收。分配是指为Java程序运行分配存储空间，包括分配运行前可确定的静态的程序空间和分配程序运行时动态申请的存储空间。回收是指将程序不再使用的存储空间变为可以重新分配的空间。由于Java语言没有提供显式释放在程序运行时动态申请的内存手段，需要由回收程序确定哪些程序空间不再使用，垃圾的自动回收也就成为存储管理的重要任务。

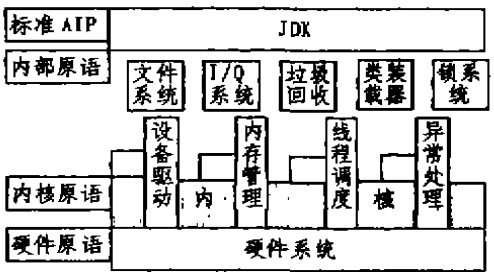

图1 Java芯片系统结构

1 Java芯片系统的特点

Java芯片系统由硬件和软件两部分组成：硬件是一块Java芯片，实现Java虚拟机指令集中大部分指令；系统软件，由管理硬件(如内存、I/O)的内存管理、I/O控制和支持类文件运行的异常处理、线程控制等内核程序组成，还包括类装载器，垃圾回收等高层程序。软硬件共同支持Java应用程序的运行。Java芯片系统的基本结构如图1所示。

*)本课题得到国家自然科学基金资助。秦莹 硕士生，主要研究方向：计算机体系结构、Java存储管理实现技术；陈虎 博士生，主要研究方向：计算机体系结构、Java实现技术等；戴葵 博士，副教授，主要研究方向：计算机体系结构、神经网络等；杨晓东 教授，博导，主要研究方向：并行计算机体系结构、故障检测等。