基于XML的多维概念模型的研究*)

陆昌辉　邓　苏　张维明

(国防科学技术大学管理科学与工程系　长沙410073)

摘　要　对于数据仓库概念模型的研究,目前缺乏统一的标准,不利于模型的交流与共享。基于XML的多维概念模型利用XML这一标准交换语言,不仅很好地解决了这一问题,而且也为元数据的集成与共享打下了基础。针对多维模型的特点,定义了一个特定的DTD,它能够完整地描述多维概念模型的各种语义特征,并针对基于UML的多维概念建模方法,定义了基于XML的多维概念模型与基于UML类图的多维概念模型的映射方法,为其应用奠定了实践基础。

关键词　数据仓库,多维建模,概念模型,XML,UML

The Research of Multidimensional Conceptual Model Based on XML

LU Chang-Hui　DENG Su　ZHANG Wei-Ming

(Department of Management Science and Technology, National University of Defense Technology, Changsha 410073)

Abstract　There isn't a unified standard among the conceptual models of data warehouse, and it brings much difficult to the exchange and share of these models. The multidimensional conceptual model based on XML makes use of a standard exchange language——XML, and it not only resolves this problem very well, but also builds solid base for the integration and share of metadata. Considering the features of multidimensional model, defines a special DTD, which can perfectly describe various semantic characteristics of multidimensional conceptual model. According to the multidimensional conceptual modeling which is based on UML, this paper describes the mapping arithmetic between the multidimensional conceptual model based on XML and that based on UML class diagram, and it prepares the application base for the wide use of this technique.

Keywords　Data warehouse, Multidimensional modeling, Conceptual model, XML, UML

近几年来,数据仓库技术被广泛地应用到决策支持系统中。从目前情况来看,许多企业和机构已经建立了相对完善的OLTP系统(联机事务处理系统)。随着时间的推移,这些系统积累了大量的历史数据,其中蕴含了许多重要的信息。因此如何在现有系统的基础上应用数据仓库技术,已显得十分迫切,其中多维建模技术是其关键技术之一[1]。正如文[2]所述,借助多维建模技术,各种OLAP工具能够方便地对数据进行复杂分析和可视化。多维建模的最大优点是考虑问题的方式与数据分析师的方式非常相似,而且查询的性能也非常高。

Abelló在其博士论文中,对多维建模技术进行了很好的归纳总结,指出了目前的研究主要集中在逻辑建模阶段[3]。Abelló的两大突出贡献是:1)提出了一种新的分类方法,按概念模型、逻辑模型、物理模型以及形式化表示来对已有的多维模型进行分类;2)提出了一种基于UML的概念建模方法[3]。该概念建模方法以一种通用的建模语言UML为基础,虽然较ME/R[4],StarER[5]等方法来说有一定的优越性,但目前还只是研究了怎样对UML进行扩展(定义新的构造类、标记值以及约束)来满足多维建模的要求,还并不是很成熟。这样,对于多维概念建模的研究,就出现了多种方法并存,而且它们各成一派,没有一种统一的描述机制,不利于模型的交流与共享。为了解决这一问题,本文引入了XML这一标准交换语言,把它作为多维概念模型的描述基础,这样不仅有助于数据与元数据的集成和共享,而且,这一思想也符合CWM描述的规范,具有一定的通用性[6]。本文在Abelló博士的研究基础上,对多维概念模型的DTD,以及用XML描述的多维模型与用UML类图描述的多维模型之间的相互转换方法进行了研究。

1　多维概念模型的定义

多维概念模型主要有星型模式、雪片模式、星族模式等形式[7],在这里,我们定义的多维概念模型是基于雪片模式的,根据文[7]的描述,很容易从雪片模式得到星型模式和星族模式。

定义1(维级)　可用一个三元组来表示,标记为$DL=(Lname,Al,E)$。其中$Lname$为维级的名称,Al为其属性集,E是该维级对象允许的操作集合,比如创建和删除等。

定义2(维)　由维级组成,它可用一个四元组来表示,标记为$DC=(Dname,DL,ag,E)$。其中$Dname$为维度名称,DL是组成该维的维级集合,ag是一个定义在DL上的有向的、非循环的弱连通图,其中顶点代表各个维级,边表示各维级之间的关系。E是该维对象允许的操作集合,如添加和删除等。维的属性由构成该维的维级的属性组合而成。

定义3(事实)　设$DC_1,DC_2,\cdots,DC_n$是n个维,事实(FC)是定义在这n个维上的组合,它可以表示成一个如下的五元组:$FC=(Fname,DC,Af,AP,E)$。其中$Fname$为事实的名称,DC是维集合,Af为事实属性集,AP为聚合模式,它可以表示成一个这样的三元组,$AP=(ai,dj,agt)$,$ai\in$度量属性,$dj\in DC$,$agt\in\{\Sigma,\Phi,C\}$(Σ表示各种聚合操作,Φ表示除求和之外的求平均值、最大值、最小值等聚合操作,C表示只能进行计数的聚合操作)。E是该事实对象允许的操作集合。

定义4(雪片模式)　在基于UML的多维概念建模方法

*)Supported by the National Natural Science Foundation of China under Grant No. 60172012(国家自然科学基金)。陆昌辉　博士研究生,主要研究领域为数据仓库技术,数据挖掘技术。邓　苏　博士,教授,主要研究领域为系统仿真技术和数据仓库技术。张维明　博士,教授,主要研究领域为信息与决策技术,软件工程技术。

中，雪片模式是由 UML 类组成的有根图，它满足下列条件：

·从根类到与其直接相连且路径长度为1的所有类的连接属于关联类型。

·设类 $class_i$ 到根类的路径长度 $n \geqslant 1$，类 $class_j$ 到根类的路径长度 $m=n+1$，则类 $class_i$ 与类 $class_j$ 之间的边属于聚合类型连接。

·设类 $class_i$ 通过关联类型连接与根类相连，则所有与类 $class_i$ 通过聚合连接相连且路径长度 $n \geqslant 1$ 的所有类及其边就组成一个子图，子图之间没有边相连。

·在一个子图中，如果从根类到类 $class_i$ 与类 $class_j$ 的路径长度相等，则类 $class_i$ 与类 $class_j$ 之间没有边相连。

其中根类为事实，每个到根类的子图为该事实的一个维，在子图中，从根类到叶子类的路径 $class_1, class_2, \cdots, class_n$ 为一个维层次，该路径中的类 $class_1, class_2, \cdots, class_n$ 为所对应维的维级。

2 模型的 DTD 描述

由于 XML 的简单性(规范简单)、开放性(标准开放)、可扩充性(可以自定义数据类型和数据结构)、灵活性(几乎可以描述任何的数据模型)等，它迅速成为了一种数据/元数据的标准交换格式[6]。为了用 XML 文档很好地描述用 UML 类图构建的多维模型，需要定义一个特殊的 DTD，下面列出了其核心部分：

```
〈!ELEMENT snowflake_schema (class*)〉
〈!ATTLIST snowflake_schema name CDATA #rerquired〉
〈! ELEMENT class (cls_name, AttributeList, MethodList,
    PointsTo*)〉
〈!ATTLIST class stereotype CDATA #rerquired〉
〈!ELEMENT cls_name (#PCDATA)〉
〈!ELEMENT AttributeList (Attribute*)〉
〈! ELEMENT Attribute (att_name, att_type, modifer, AID?,
    calc_att?, link_path?, type?, aggfunc*)〉
〈!ELEMENT MethodList (Method*)〉
〈!ELEMENT Method (meth_name, InputParams*, OutputRes)〉
〈!ELEMENT att_name (#PCDATA)〉
〈!ELEMENT att_type (#PCDATA)〉
〈!ELEMENT modifer (?|1)〉
〈!ELEMENT AID (?|1)〉
〈!ELEMENT calc_att (#PCDATA)〉
〈!ELEMENT link_path (#PCDATA)〉
〈!ELEMENT type (measure|descriptive)〉
〈!ELEMENT aggfunc ((dimension_name,functype)|NULL)〉
〈!ELEMENT dimension_name(#PCDATA)〉
〈!ELEMENT functype (sum|count|min|max|avg)〉
〈!ELEMENT meth_name(#PCDATA)〉
〈!ELEMENT InputParams(#PCDATA)〉
〈!ELEMENT OutputRes(#PCDATA)〉
〈!ELEMENT PointsTo (class_name, link_type, source_card, tar-
    get_card)*〉
〈!ELEMENT class_name (#PCDATA)〉
〈!ELEMENT link_type (association|aggregation)〉
〈!ELEMENT source_card (0..*|1..*|0..1|1)〉
〈!ELEMENT target_card (0..*|1..*|0..1|1)〉
```

雪片模式是由类组成的集合(事实、维级、维均通过类来描述)，类包含 *cls_name*、*AttributeList*、*MethodList*、*PointsTo* 四个元素和 *stereotype* 属性，*cls_name* 唯一标识这个类，*AttributeList* 是该类的所有属性集，*MethodList* 是该类的所有方法集，*PointsTo* 是该类的所有连接集，*stereotype* 用来表示该类所属的构造类类型。

$Method = (meth\_name, InputParams*, OutputRes)$。其中 *meth_name* 表示方法的名称，*InputParams* 表示方法的输入参数，*OutputRes* 表示方法的输出结果。

$$Attribute = (att\_name, att\_type, modifier, AID?, calc\_att?, link\_path?, type?, aggfunc*) \mid att\_type \in DataTypes \wedge modifier \in \{?,1\} \wedge type \in \{measure, descriptive\} \wedge aggfunc \in \{SUM, COUNT, MIN, MAX, AVG, NULL\}$$

Attribute 描述了类的每个属性的名称，数据类型，修饰符，属性标识符(AID)，规则，路径，类型以及聚合模式信息。其中，*att_name* 是唯一标识该属性的名称，*att_type* 是该属性的数据类型，*modifier* 标识该属性是否允许存在空值，*AID* 唯一地标识了该属性的数据源和位置。*calc_att* 是用来计算该属性的规则，如果有一规则与该属性相连，则该属性标记为 *calculated attribute*。所有非计算属性，其 *calc_att* 元素均标记为 NULL，表明该属性并不是通过计算得出的。计算属性的 AID 元素均定义为 NULL，表明计算属性并不存在于现有的任何数据源中。*Type* 元素用来区分度量属性和描述性属性，对于度量属性，必须指明其聚合模式信息 *aggfunc*，包括沿哪些维进行聚合(*dimension_name*)，以及聚合函数的类型(*functype*)，而对于描述性属性，*aggfunc* 则设为 NULL。

对于计算属性定义规则的表达式可以用如下的巴科斯范式来描述，其中 f 和 g 分别为一元函数和二元函数。

```
〈EXPR〉→〈EXPR〉+〈TERM〉|〈EXPR〉−〈TERM〉|〈TERM〉
〈TERM〉→〈TERM〉×〈FACTOR〉|〈TERM〉÷〈FACTOR〉|〈FAC-
TOR〉
〈FACTOR〉→(〈EXPR〉)|f(〈EXPR〉)|g(〈EXPR〉,〈EXPR〉)|a
a∈R∪{AttributeList[att_name_i]}
```

link_path 用来对来自其他类的属性进行描述，如果其对应的属性属于类本身，则该元素设置为 NULL。

$$PointsTo = \{ (class\_name, link\_type, source\_card, target\_card) \mid source\_card \in \{0..*, 1..*, 0..1, 1\} \wedge Target\_card \in \{0..*, 1..*, 0..1, 1\} \}$$

PointsTo 描述了 UML 雪片模式中类之间的关联，*class_name* 是连接的类名，$link\_type \in \{association, aggregation\}$，*source_card* 和 *target_card* 分别是描述该连接源和目的的基数。

Pedersen 在文[8]中提出了一个关于数据仓库模型的评价标准，根据这个标准对定义的 DTD 进行分析：

(1)显示地表达维的层次：在该 DTD 中，用类来表示维级(通过属性 Stereotype 的内容来确定是否为维级)，通过这些维级类之间的关联(*PointsTo* 元素)从而就表示了维的层次。

(2)平等地对待维和聚合属性：把它们统一定义成类的属性，用 *Attribute* 元素来描述。

(3)支持维上的多重层次、支持非到上的层次、支持非严格的层次、支持事实与维之间的“多对多的关系”、支持跳跃式的层次：所有这些维上的、事实与维之间的各种关联均通过类之间的关联(*PointsTo* 元素)来表示。

(4)支持对度量进行正确的聚合或抽象：通过类的 *Attribute* 元素的 *Type* 子元素和 *aggfunc* 子元素就能进行正确的描述。

(5)支持度量的抽象或聚合的不同层次的粒度：在(4)的基础上，结合 *PointsTo* 元素就能进行表示。

(6)处理时间的变化：时间维和其他维在描述上并没有什么区别，只不过对事实进行分析的角度不同而已，对于它的各种变化，该 DTD 当然完全支持。

总之，对于数据仓库中的多维模型，该 DTD 完全能够描述它；反之，符合该 DTD 定义的 XML 文档并不一定有一个相应的多维模型与之对应(如，该 DTD 并不能限制维与维之间进行关联，而在多维模型中，维与维之间是不能有关联的)。

3 模型与 XML 文档之间的映射

从前面的 DTD 定义与 UML 类图的集合描述[9]中不难看出，由于它们的元素与属性之间存在一种对应关系，因此，符合该 DTD 定义的 XML 文档一定能够转换为相应的 UML 类图，相应地，UML 类图也一定能够用符合该 DTD 定义的 XML 文档来表示。必须注意的是，这里的 UML 类图并不全部符合多维概念模型的描述，对应地，符合该 DTD 定义的 XML 文档并不全部都存在多维概念模型与之对应，多维概念模型的 XML 表示只不过是其一个真子集。

3.1 模型到 XML 文档的转换

这个用 UML 类图描述的多维概念模型到 XML 文档的映射算法分为两部分，第一部分就是 translateUMLClass，该算法的作用就是从模型中的根类开始，把根类以及从根类可达的类映射成相应的 XML 文档。第二部分就是 translateUML 算法，它的作用就是找出该模型的事实类，也就是 UML 类图中的根类，然后从它开始进行转换。

translateUMLClass 算法的描述如下：

```
(1)translateUMLClass(aClass As Class, XMLFile As file):
(2)把一个 UML 类转换为对应的 XML 文档片断;
(3)Set theAssociations = aClass.getassociations //下面是递归调用，把所有从该类引出的连接指向的目的类的相关信息记入 XMLFile
(4) for AssNum% = 1 to theAssociations.Count
(5)     Set aAssociation = theAssociations.GetAt (AssNum%)
(6)   If aAssociation.Role1.SupplierName = aClass.Name then
(7)         translateUMLClass (aAssociation.Role1.class, XMLFile)
(8) Next AssNum%
(9)End
```

TranslateUML 算法的描述如下：

```
(1)translateUML (theClassDiagram As ClassDiagram):
(2)生成新的 XML 文档 XMLFile，在 XMLFile 中记入模式的开始标记符〈SNOWFLAKE〉，并把模式的名称和其它信息也记入 XMLFile 文档中
(3) Set theClasses = theClassDiagram.GetAllClasses()
(4) For ClsNum% = 1 to theClasses.Count
(5)     Set aClass = theClasses.GetAt(ClsNum%)
(6)     If aClass.Stereotype = "Fact" then //找到模式的事实类并对其进行转换
(7)       translateUMLClass(aClass,XMLFile)
(8) Next ClsNum%
(9)把模式的结束标记符〈/SNOWFLAKE〉记入 XMLFile
(10)End
```

3.2 XML 文档到模型的转换

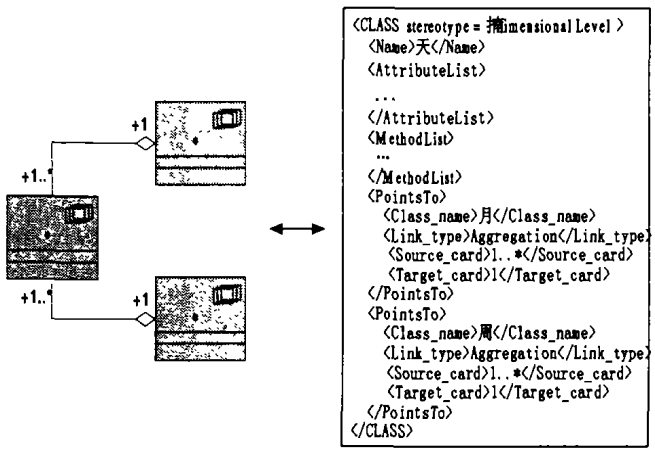
XML 文档到 UML 类图的转换算法可分为两步，第一步就是生成与 XML 文档对应的 UML 类图中的所有类，第二步就是建立类之间的连接，其具体如下：

```
(1)generateUML(XMLFile As File)
(2)把文件指针 FilePtr 置为文件的开头，置初始结果 Result 为 FALSE，再按顺序对 XMLFile 进行扫描
(3)If FilePtr 指向模式的开始标记符〈SNOWFLAKE〉，Then
(4)     创建一个新 UML 类图，其名称为 snowflake.name，将 Result 置为 TRUE，FilePtr 指向下一行
(5)Else
(6)     FilePtr 指向下一行，转(3)
(7)While not FilePtr = EOF(XMLFile)do //生成 UML 类图中的所有类
(8)     If FilePtr 指向类的开始标记符〈CLASS〉，Then 在创建的 UML 类图中添加一个类，并根据 XML 文件中的相应内容对该类的名称、stereotype、attribute、method 等进行设置，将 FilePtr 指针指向下一行
(9)Wend
(10)If Result = FALSE Then 显示错误信息，算法结束
(11)将 FilePtr 指针指向文件的开头 //重新对文件进行扫描，建立类之间的连接
(12)While not FilePtr = EOF(XMLFile)do
(13)     If FilePtr 指向类的开始标记符〈CLASS〉 Then
(14)          把该类赋给 TheSourceClass，并将 FilePtr 指针指向下一行
(15)     While not FilePtr 指向类的结束标记〈/CLASS〉 do
(16)          If FilePtr 指向连接的开始标记〈PointsTo〉 Then
(17)               theTargetClass = PointsTo.Class，在 theSourceClass 与 theTargetClass 之间建立连接，并根据文件的相应内容对该连接的类型、source_card、target_card 进行赋值，并将 FilePtr 指针指向下一行
(18)     Wend
(19)Wend
(20)End
```

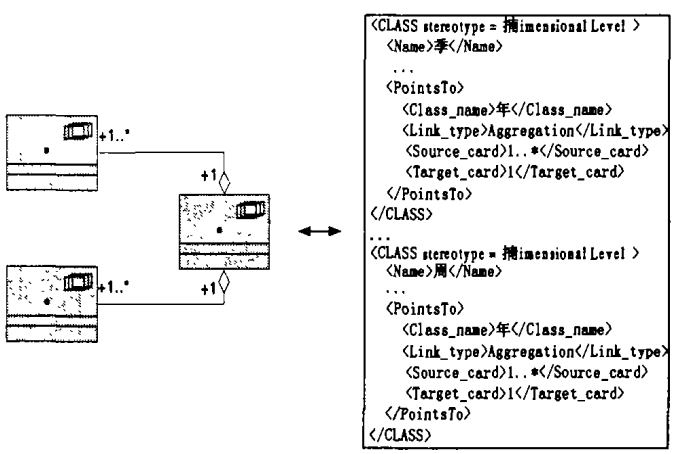
3.3 映射结果分析

下面根据多维模型中的一些特殊情况来对这个映射进行分析。

维上的多重层次：根据定义4，这是指从子图的顶点出发，有多条路径到叶结点。这包含两重含义，一种就是顶点和叶结点均相同，但路径不同；另一种含义就是顶点相同，叶结点不同，其路径自然也就不同。造成这两种情况的原因无非就是维级间的连接有相同的源端或目的端，其映射结果分别如图1(a)、图1(b)所示。



(a) 维级间的连接有相同的源端



(b) 维级间的连接有相同的目的端

图1

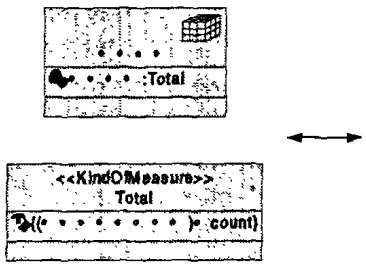
对于维级之间、事实与维之间的"多对多的关系"[8]，在 UML 类图中通过连接的类型以及连接的基数就能进行完整的表示。相应地，映射成 XML 文档时，只需指明引出该连接的类的 *PointsTo* 元素的 *class_name*，*link_type*，*source_card*，*target_card* 等内容就行了。图2显示的是在医疗诊断系统中，病人(维)与诊断(事实)之间的多对多关系，对于维级之

间的“多对多的关系”描述，其方法与此类似。


```
</CLASS stereotype=" Fact" >
  <Name>诊断</Name>
  ...
  </PointsTo>
    <Class_name>病人</Class_name>
    <Link_type>Association</Link_type>
    <Source_card>1..*</Source_Card>
    <Target_card>1..*</Target_card>
  </PointsTo>
</CLASS>
```

图2 事实与维之间的“多对多的关系”

在基于 UML 的多维概念建模方法中，为了描述度量能够沿着哪些维进行何种聚合操作，它引入了一种度量类型的构造类，在该构造类中指明了这种类型的度量能够进行聚合的维，以及聚合操作的类型（求和、计数、求最小值、求最大值、求平均值等）。对于这种情况，只需要对事实类的 *AttributeList* 的 *type*、*aggfunc* 等内容进行指定就可以了。图3显示的是在教学管理事实中，学生人数这个度量属性可以沿课程维、教员维、学期维进行计数聚合。




```
</CLASS stereotype=" DWCube" >
  <Name>教学管理</Name>
  ...
  </AttributeList>
    <Attribute>
      <att_name>学生人数</att_name>
      ...
      <type>measure</type>
      <aggfunc>
        <dimension_name>课程、教员、学期</dimension_name>
        <functype>Count</functype>
      </aggfunc>
    </Attribute>
  </AttributeList>
  ...
</CLASS>
```

图3 度量种类的描述

4 与相关工作的比较

本文主要有两大贡献：一是针对多维概念建模的要求，提出了一种基于 XML 的多维概念模型的描述方法，并给出了具体的 DTD 描述，为多维概念模型的交流与共享提供了一个公共的平台；二是在 Abelló 提出的基于 UML 的概念建模方法的基础上，给出了 UML 类图到 XML 文档的映射方法，进一步深化了该方法的研究（为其元数据的集成与管理，以及到逻辑模型的映射打下了坚实的基础）。下面主要从多维概念模型的研究以及 UML 类图到 XML 的映射这两个方面与相关的工作进行详细的比较。

对于多维概念模型的研究，国外主要有 StarER 模型[5]、ME/R 模型[4]，以及基于 UML 的多维概念模型[3]，在国内，主要有哈尔滨工业大学的李建中、高宏在偏序和映射的基础上提出的一种多维模型[10]，复旦大学的陈明等提出的 DWER 模型[11]，以及上海交通大学的李琪、白英彩对数据仓库中维的建模进行了研究，并提出了一种基于关系数据库的 SQL (D)数据模型[12]。StarER 模型、ME/R 模型、DWER 模型是在 ER 模型的基础上，根据多维建模的要求对其进行不同的扩展，它们虽然对应用领域的静态结构建模非常有用，但并没有摆脱 ER 模型对动态部分（预测查询行为）和功能部分（在维上增加度量或者层次级别间的功能关系）的建模能力相对来说还非常弱这一弊病。而且，这些模型各自扩展的方法不同，不同模型之间（也就是 StarER 模型、ME/R 模型、DWER 模型之间）难以进行交流。李建中、高宏等提出的多维模型虽然能描述数据仓库中的一些复杂层次关系和语义，但是没有相应的建模工具来支持，不利于该方法的推广使用，也不利于模型之间的交流与共享。基于 UML 的多维概念模型在动态建模部分和功能建模部分的能力有所增强，由于它是在 UML 的基础上进行了扩展，无法与其它模型进行交流。文[12]提出的虽然支持不平衡、异构的维层次结构，但它是建立在关系数据库的基础上的，在面向对象数据库、多维数据库中并不适用。本文提出的基于 XML 的多维概念模型就很好地解决了这一问题，它采用 XML 这一标准交换语言为基础，而且能描述多维模型的各种语义情况，其他的多维模型（如 StarER 模型、ME/R 模型、DWER 模型、基于 UML 的多维模型等）都能转换成用 XML 表示的多维模型，从而为各种多维概念模型之间的交流提供了一个统一的平台，为以后的元数据的集成与管理奠定了坚实的基础。另外，本文提出的多维模型不仅可以应用于关系数据库，在多维数据库、面向对象数据库中也可以广泛使用。文[13]对基于 XML 的数据立方数据模型的面向对象的实现进行了研究，它利用 XML 来描述半结构化数据，为基于 Web 数据仓库的应用提供了一种表示和实现方法。但是它并没有为多维概念模型提供一个统一的描述基础，也没有研究其他多维概念模型与它之间的映射方法。

关于 UML 类图到 XML 文档映射的研究目前还相当少，文[9]为了给基于 Web 数据的逻辑集成提供指导，首次在 OLAP 系统中考虑了从概念层集成基于 Web 的数据源，也是首次描述了从 XML 数据中自动构造 UML 类图的算法，并且仍然保存重要语义信息的文章。由于在生成的 UML 类图中，每一个类都包含有一组相关属性，而在文[9]中，UML 类图是直接根据 XML DTD 来构建的，它不可能决定属性的数据类型。在本文提出的映射方法中，是根据符合特定的 XML DTD 的 XML 文档来进行转换的，在该 XML 文档的 Attribute 元素中就包含了属性的数据类型信息，从而很好地解决了这个问题。本文提出的映射方法并不是 XML DTD 与 UML 类图之间的转换，而是符合该 DTD 定义的 XML 文档实例与 UML 类图之间的转换，这与用户在进行多维概念建

模时得到的一个个模型实例是相符的。

总结 为了解决数据仓库概念模型的交流与共享，本文提出了一种基于 XML 的多维概念模型，它充分利用了 XML 的可扩展性和统一标准性，不仅能很好地解决这一问题，而且也为元数据的集成与共享打下了基础。

针对多维模型的特点，本文定义了一个特定的 DTD，该 DTD 能够完整地描述多维概念模型的各种语义特征。对于多维概念建模，目前比较新的方法是基于 UML 的多维概念建模方法，它的一个显著优点就是建立在一种广泛接受的面向对象建模语言的基础上，从而使得开发人员在进行具体应用时，不必重新学习一些新的模型及其相关概念，便于广泛应用。而且，这种方法与 OMG 组织制定的 CWM 标准也是相符的，为今后的工作奠定了一个基础[6]。根据这一实际情况，本文设计了基于 XML 的多维概念模型与基于 UML 类图的多维概念模型之间的映射方法。把 UML 类图映射为相应的 XML 文档，这给模型的交流与共享，以及元数据的集成与共享带来了极大的方便。把 XML 文档逆向映射为 UML 类图，能够对 XML 数据进行快速的图示浏览，从而为多个多维模型的集成提供了一定的指导。

在具体的实现中，我们选择了在 Rational Rose 2002中嵌入一套自定义的组件，该组件不仅广泛支持基于 UML 的多维概念建模方法[14]，而且也能非常方便地进行 XML 文档与 UML 类图之间的转换。

参 考 文 献

1 Kimball R. The data warehouse toolkit. New York, John Wiley & Sons, 1996

2 Trujillo J C, Palomar M, Gomez J, Song I-Y. Designing data warehouses with oo conceptual Models. IEEE Computer, 2001, 34(12):66～75

3 Abelló A. YAM²: a multidimensional conceptual model: [PhD Thesis]. Computer Science in the Facultat d'Informàtica de Barcelona (FIB)2002

4 Blaschka M. FIESTA: A framework for schema evolution in multidimensional databases: [PhD thesis]. Technische Universitat Murchen, Germany, 2000

5 Tryfona N, Busborg F, Christiansen J G B. starER: A conceptual model for data warehouse design. In: Proc. of the ACM 2nd Intl. Workshop on Data warehousing and OLAP (DOLAP'99), Kansas City, Missouri, USA ,1999. 3～8

6 OMG. Common warehouse metamodel, February 2001. Version 1.0.

7 Moody D L, Kortink M A R. From enterprise models to dimensional models: a methodology for data warehouse and data mart design. In: Manfred A. Jeusfeld, eds. Proc. s of the Second intl. Worskshop on Design and Management of Data Warehouses 2000, Stockholm, Sweden

8 Pederson T B, Jensen C S. Multidimensional data modeling for complex data. In:Proc. of the 15th Intl. Conf. on Data Engineering (ICDE'99), Sydney Australia: IEEE Computer Society, 1999. 336～345

9 Jensen M R, MØller T H, Pedersen T B. Converting XML data to UML diagrams for conceptual data integration. Data & Knowledge Engineering, 2003, 44(3):323～346

10 李建中，高宏. 一种数据仓库的多维数据模型. 软件学报，2000，11(7):908～917

11 陈明，吴国文，施伯乐. 数据仓库概念模型的设计. 小型微型计算机系统，2002,23(12):1453～1458

12 李琪，白英彩. 数据仓库中维的建模和查询. 计算机研究与发展，2002，39(5):612～618

13 Wang Xiao-ling, DONG Yi-sheng. XML-based data cube [C]. RIDE02, San Jose, USA

(上接第157页)

网络质量无关，只与事务本身的实时特性相关，所以随着 DP 增加，基本上是线性变化。图4是由于 SSRTD 采用了接纳控制和放行管理，有效地控制了系统的负载，减少了事务运行时内外存的 I/O 交换时间。

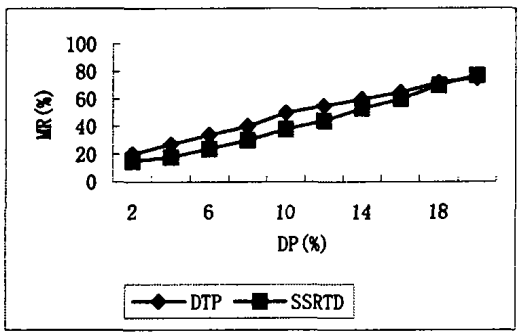


图3 断接概率对 MR 的影响

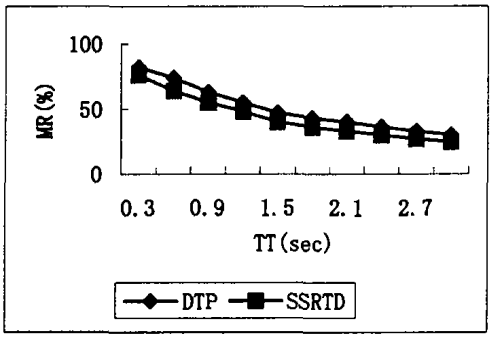


图4 事务负荷对 MR 的影响

结束语 传统分布式实时事务管理机制不能较好地支持移动分布环境中实时事务的执行。因此，必须为移动分布式实时事务建立新的事务模型及其事务管理机制。本文分析了移动计算环境的软实时事务调度管理，研究了一种根据移动主机的连接情况、事务的扩展截止期和结果相似性等因素决定发送执行结果的实时事务调度策略 SSRTD。全文工作概括如下：

(1)给出了一个移动实时数据库系统模型；

(2)给出了“扩展截止期”和“结果相似性”的概念；

(3)给出了根据“扩展截止期”和“结果相似性”计算事务执行结果的发送时间方法；

(4)给出了 SSRTD 协议的相关算法；

(5)对 SSRTD 协议进行了性能测试和分析，证明其具有较好的性能。

移动实时事务是一个新研究领域，有许多方面需要研究。我们将进行的下一步工作重点是研究移动实时数据库恢复处理机制。

参 考 文 献

1 Chrysanthis P K. Transaction processing in mobile computing environment. In: Proc. of 6th IEEE Workshop on Advances in Parallel and Distributed Systems, New Jersey, 1993

2 Lam K Y, Kuo T W, Tsang W H. Concurrency control in mobile distributed real-time database systems. Information systems, 2000, 25(4): 261～286

3 廖国琼，刘云生. 移动实时嵌套事务的并发控制. 计算机学报，2003，26(10): 1326～1331

4 刘云生，廖国琼. 移动实时嵌套事务提交. 软件学报，2003，14(1): 139～145

5 Saad-Bouzefrane S, Sadeg B, Amanton L. In:Proc. of 4th IEEE Intl. Symposium on Object-Oriented Real-Time Distributed Computing, Magdeburg, Germany, 2001

6 刘云生. 现代数据库技术(第一版). 北京:国防工业出版社，2001