计算机科学 1998 Vol.25№.2

面向对象的多数据库系统中语义冲突分类框架*)

The Framework for Classifying Semantic Confilicts in Object-Oriented Multidatabase Systems

王国仁 于 戈 张 斌 郑怀远

(东北大学计算机系 沈阳 110006)

摘 要 This paper presents a three-layer framework for classifying semantic conflicts in object-oriented multidatabase systems, namely schema structural layer semantic conflicts, object-instance layer ones, and behavior-method layer ones. The corresponding resolution to these conflicts are also suggested in this paper.

关键词 Object-oriented multidatabase systems, Schema integration, Semantic conflicts

一、前言

多数据库系统中模式集成是关键性问题之一。模式集成一般包括[1,2]:(1)视图集成,产生一个全局统一的概念模式;(2)数据库集成,将若干个已存在的数据库模式集成为一个全局模式,它是被集成库模式的一个虚拟视图。本文研究的是与数据库集成相关的语义冲突问题。人们对模式集成过程中的语义冲突问题进行了深入的研究,Batini[1]将语义冲突分为两大类,即命名冲突和结构冲突。命名冲突又分为同义词和同形异义词两种;结构冲突又分为:类型冲突、依赖冲突、关键字冲突、约束冲突等。Shova[2]根据上述分类框架,针对二元集成方法和ER模型提出了各种冲突的解决策略。这种分类框架的一个主要缺点是并未考虑数据冲突问题,因为在数据库集成过程中除了模式可能发生冲突以外,很可能在数据级发生冲突(这一点与视图集成不同)。Spaccapietra[4]将冲突分为:语义冲突、描述冲突、数据模型冲突和结构冲突等,这种分类框架在语义冲突分类方面与文[1][2]基本类似,也未考虑在数据级发生的各种冲突问题。因此,文[3][5]将语义冲突分为两大类,即模式冲突和数据冲突,Kim[5]针对关系模型、Kamel[3]针对扩展ER模型提出了大致相同的冲突分类框架及相应的冲突解决策略。

本文针对面向对象的多数据库系统的模型特征,提出了一个三级语义分类框架,即从模式结构级、对象实例级和行为结构级来进行语义冲突分类。为了系统而清晰地对语义冲突进行分类,必须有一个统一的分类基础,也就是必须将用各种模型表示的模式首先映射到一个统一的中间数据模型表示的模式中来,然后再分析各类冲突并予以解决。要找到一个中间数据模型以使该数据模型能够表示各局部数据模型中的概念是比较容易的[6],在本文的分类框架中,中间映射模式是以面向对象的数据模型为基础的。

二、模式结构级语义冲突及其解决策略

模式结构级语义冲突主要是指与参加集成的各个局部数据库模式相应的中间映射模式之间可能发生的各类冲突。主要包括三种冲突类型:(1)类级冲突;(2)属性级冲突;和(3)类与属性级冲突。

2.1 类级冲突

类级冲突是指来自两个不同模式中的类在定义上发生冲突,可分为三种:(1)类命名冲突;(2)类结构冲突;(3)类关键字冲突。

①类命名冲突:包括类名同义词(相同含义的类具有不同的名字)和类名同形异义词(不同含义的类具有相同的名字)。当两个类C1、C2具有相同的含义(类、属性及对象的含义涉及到类、属性和对象间的等价理论,超出了本文的讨论范围)但有不同的名字时就发生类名同义词冲突。类名同义词冲突有两种可能的解决方法:(1)对发生类名同义词冲突的类进行换名;(2)建立类名同义词表以记载类名同义词冲突。例如在模式S1中使用类Corporation来表示公司,而在模式S2中用类Company来表示公司,这时在〈Corporation, Company〉之间就发生了类名同义词冲突。解决冲突的换名子句是:rename Corporation in S1 to Company;或 rename Company in S2

*)国家863计划资助项目

to Corporation。

当模式 S1 中的类 C 与模式 S2 中的类 C 有相同的名字但具有不同的含义时就发生了类名同形异义词冲突，其解决有两种可能的方法：(1)对发生了类名同形异义词冲突的类进行换名；(2)建立类名同形异义词表以记载类名同形异义词冲突。例如在一个模式 S1 中用 Employee 来表示对象类月薪雇员，而在模式 S2 中用 Employee 来表示对象类周薪雇员，这时在〈S1. Employee，S2. Employee〉就发生了类名同形异义词冲突，可将月薪雇员类更名为 Employee_Salary，而将周薪雇员类更名为 Employee_Wage，换名子句是：rename Employee in S1 to Employee_Salary；或 rename Employee in S2 to Employee_Wage。

②类结构冲突。可以分为包含冲突和相交冲突两种。包含冲突是指在含义相同的两个相同类 C1 和 C2 中一个类的属性集是另一个类的属性子集时就发生了包含冲突。包含冲突解决策略如下：

(1)若 attrSet(C1)＝attrSet(C2)，则合并这两个类为一个新类。Merge(C1,C2) into C3，且 attrSet(C3)＝attrSet(C1)＝attrSet(C2)。

(2)若 attrSet(C1)⊂attrSet(C2)，则在 C1 和 C2 之间构造一个超类/子类层次。Add subclass C2 to C1。

冲突解决后 C1′和 C2′(相应与原类 C1 和 C2，以后类同)的属性结构为：

attrSet(C2′)＝attrSet(C2)－attrSet(C1)

attrSet(C1′)＝attrSet(C1)

例如设 attrSet (RA) = {name, sno, gpa, salary}, attrSet(Student)＝{nmame, sno, gpa}，这时在〈RA,Student〉之间就发生了包含冲突。该包含冲突的冲突解决语句为：

Add subClass RA to Student；且

attrSet(Student′)＝attrSet (Student)

attrSet(RA′)＝attrSet(RA)－attrSet(Student)

相交冲突是指在含义相同的两个类 C1,C2 中，当它们的属性集的交不为空时所发生的冲突，其解决办法是通过概括语义将发生相交冲突的类概括为一个新的超类，即 generalize(C1,C2) into C3；其中，attrSet (C3) = attrSet (C1) ∩ attrSet (C2), attrSet (C1′)＝attrSet(C1)－attrSet(C3), attrSet(C2′)＝attrSet(C2)－attrSet(C3)。

例如，设 attrSet (Teacher) = {ss #, name, salary}, attrSet (Student) = {ss #, name, sno, gpa}，则〈Teacher, St_dent〉之间就发生了相交冲突。相交冲突 Teacher CT4 Student 的冲突解决语句为：generalize(Teacher, Student) into Person；其中 attrSet (Person)＝{ss #, name}, attrSet (Teacher′)＝{salary}, attrSet(Student′)＝{sno,gpa}。

③类关键字冲突。当两个含义相同的类具有不同的关键字约束时就发生了类关键字冲突，它包括候选关键字冲突和主关键字冲突两种。如果两个含义相同的类 C1,C2 具有不同的候选关键字时在类 C1 和 C2 之间就发生了候选关键字冲突，其解决方法是把两个类的候选关键字的交集分别作为两个类的候选关键字即可。如果两个含义相同的类 C1 和 C2 具有不同的主关键字时在 C1 和 C2 之间就发生了主关键字冲突，其解决方法是从两个类的公共候选关键字中挑选出一个关键字分别作为类 C1 和 C2 的主关键字。

2.2 属性级冲突

属性级冲突是指两个类在属性级发生的语义冲突情况，也可以分为三种类型：(1)属性命名冲突；(2)属性结构冲突；和(3)属性约束冲突。

①属性命名冲突。可分为两种：属性名同义词冲突和属性名同形异义词冲突。在含义相同的两个类 C1 和 C2 中含义相同的两个属性 A1 和 A2 具有不同的名字时，在 C1. A1 和 C2. A2 之间就发生了属性名同义词冲突，其解决的可能方法有两种：(1)对发生属性名同义词冲突的属性进行换名；即：rename A1 of C1 to A2；或 rename A2 of C2 to A1；(2)建立属性名同义词字典，记载所发生的属性名同义词冲突。在两个含义相同的类 C1 和 C2 中含义不同的两个属性具有相同的名字 A 时，在 C1. A 和 C2. A 之间就发生了属性名同形异义词冲突。它也有两种解决方法：1)对发生属性名同形异义词冲突的属性进行换名；即 rename A of C1 to A1；或/和 rename A of C2 to A2；(2)建立属性名同形异义词字典。

②属性结构冲突。是指在两个相同的类中属性结构定义发生冲突，当在含义相同的两个类 C1 和 C2 中，含义相同的特征 P 在两个类中具有不同的结构时，在 C1. P 和 C2. P 之间就发生了属性结构冲突。其解决的可能方法有两种：(1)分别在两个类中补充丢失属性，即在 C1 补充丢失属性(attrSet(C1. P)∪attrSet(C2. P)－attrSet(C1. P))，在 C2 中补充丢失属性(attrSet(C1. P)∪attrSet(C2. P)－attrSet(C2. P))；(2)分别在两个类中去掉丢失属性，即在 C1 中去掉在 C2 中丢失的属性 attrSet(C1. P)－attrSet(C2. P)，在 C2 中去掉在 C1 中丢失的属性 attrSet(C2. P)－attrSet(C1. P)。例如在模式 S1 的类 Person 中用 name 来表示人的姓名特征，而在模式 S2 的类 People 中用 FirstName 和 LastName 来表

示人的姓名特征，这时在 Person 和 People 之间就发生了属性结构冲突。

③属性约束冲突。是指在两个相关的类中相同含义的属性具有不同的属性描述，主要包括：属性类型冲突和属性长度冲突。当在两个相关的类 C1 和 C2 中含义相同的两个属性 A1 和 A2 具有不同的数据类型时，在 C1. A1 和 C2. A2 之间就发生了属性类型冲突。其解决方法是在全局概念模式中将发生属性类型冲突的属性统一到某种属性类型。例如在模式 S1 的类 Student 中的属性 Grade 为整型，即用百分制来表示学生的成绩，而在模式 S2 的类 RA 中的属性 Grade 字符类型，即用 5 分制(ABCDE)来表示 RA 的成绩，则这两种成绩表示方式就发生了属性类型冲突。可将 Student 中的 Grade 属性在全局概念模式中定义为字符类型，并在原属性和目标属性(即百分制和 5 分制)之间定义一个转换函数，将百分制转换为 5 分制。这种冲突解决机制也将用于解决对象实例级的一些冲突类型，进一步的讨论可参见第 3 节。

当在两个相关的类 C1 和 C2 中含义相同的两个属性 A1 和 A2 具有不同的数据长度时，在 C1. A1 和 C2. A2 之间就发生了属性长度冲突。它的解决方法是在全局概念模式中将发生属性长度冲突的属性对的属性长度统一定义为最大者即可。

2.3 类与属性级冲突

当在两个相关的类 C1 和 C2 中等价的两个特征 P1 和 P2 在一个类中是用属性(集)来表示，而在另一个类中是用类来表示的时就发生了类与属性级冲突。它的解决策略如下：(1)首先将用属性(集)表示的等价特征聚合为一个新类，在原类与新类之间形成一个引用关系，这一步是解决类与属性冲突的基础；(2)在将新类和发生类与属性冲突的类之间的冲突通过类级冲突来加以解决。实际上通过第一步的聚合后将类与属性冲突转化为类级冲突，根据前面讨论的类级冲突解决策略即可解决类与属性冲突。在模式 S1 的类 Person 中用两个原子属性 City 和 Street 来表示一个人的家庭住址，而在模式 S2 中的类 People 中用一个单独的类 Address 来表示人的家庭住址，Address 类也有两个属性 City 和 Street。冲突解决过程如下：

① Aggegrate City, Street from Person into HomeAddress with homeAddress。这条语句的作用是将 Person 类中的属性 City, Street 聚合为一个新类 HomeAddress，在聚合前 Person 类的结构为：

```
type Person is.....
  STRUCT [.....,City:string, Street:string;].....
end type Person;
```

经过聚合后类 Person 和 HomeAddress 的结构为：

```
type Person is.....
  STRUCT [...,homeAddress:HomeAddress;]...
end type Person;
type HomeAddress is ...
  STRUCT [City:string, Street:string;]
end type HomeAddress;
```

②合并新型 HomeAddress 和 S2 中的类 Address

Merge(HomeAddress, Address) into HomeAddress

三、对象实例级冲突分类及其解决策略

在讨论对象实例级冲突分类及其解决策略之前先介绍语义值模式的概念。

3.1 语义值模式

1. 语义值。在数据库系统中，任何简单值都是一个数据类型的一个实例，即任何简单值都具有一个数据类型，一个简单值的语义是由它的类型所确定的。例如 3 的数据类型是美元，则它表示 3 美元，它与类型为日元的简单值是不能直接比较的。实际上有的语义仅靠一个数据类型来表示是不够的，即语义可能是多维的。因此一个语义值应该定义为一个具有语义环境的简单值，语义环境为一个集合，集合中的每个元素是从语义特征值到语义特征的一个赋值表达式。语义值={语义特征 i=语义特征值 i, $1 \leqslant i \leqslant n$}

2. 语义值模式。是指用来描述对象实例中属性值的语义值结构的，例如上述例子中语义值模式为 Length integer (LengthUnit:integer, LengthScale: integer)。

3. 语义值说明。该语句用来定义一个语义特征值是如何产生的。原则上讲，语义特征值可分为两种类型：一利是存储型，一种是导出型。存储型是指将语义特征值存储在数据存放的地方；导出型则只存储简单数据，而数据的语义特征值是根据具体的数据环境来导出的。一般地，导出型语义值可通过以下几种方式得到：(1)规则；(2)谓词；(3)查表；(4)函数表达式。

4. 转换函数。在面向对象的多数据库系统中，在成员数据库中的数据集成到全局数据库中之前必须经过语义值转换，因为两者的语义环境可能不同。下面是一个转换函数的例子：

cvtVal (40 (LengthUnit='feet', ScaleFactor=1), (LengthUnit ='inches', ScaleFactor = 10)) = 48 (LengthUnit='inches', ScaleFactor=10)

下面开始讨论对象实例级冲突分类及其解决策略。在面向对象的多数据库集成系统中，除了在模式

结构级可能发生各种冲突以外，在对象实例级也可能发生各种各样的语义冲突问题。对象实例级语义冲突主要包括两种类型：(1)不兼容对象实例冲突；(2)对象实例表示冲突。

3.2 不兼容对象实例冲突

不兼容对象实例冲突是一个非常普遍但又非常难以解决的语义冲突问题，当含义相同的数据项在不同的数据库中存在不一致的数据值时就发生了不兼容对象实例冲突。设C1是成员数据库DB1中的一个类，I1是C1中的一个数据项，C2是成员数据库DB2中的一个类，I2是C2中的一个数据项，且01和02分别是C1和C2中的对象实例，如果01与02的含义相同且I1与I2的含义相同但I1与I2又具有不同的值时，在对象01.I1和02.I2之间就发生了不兼容对象实例冲突。解决不兼容对象实例冲突的一种简单方法是将对象实例的最近修改作为对象实例冲突部分的值，这种方法虽然解决了不兼容对象实例冲突，但不能保证数据的正确性。例如在一个数据库中对象实例"JOHN"的生日为"JUNE 30, 1968"，而在另一个数据库中同一对象实例"JOHN"的生日为"MAY 2,1966"，这时就发生了不兼容对象实例冲突。这种冲突一般是由对数据库进行维护时造成的不兼容对象数据。

3.3 对象实例表示冲突

对象实例表示冲突是指用不兼容的符号、量纲和精度来表示相关对象实例中等价的数据元素，主要包括表达冲突、量纲冲突和精度冲突。表达冲突是指在两个相关的类C1和C2中含义相同的属性A1和A2具有不同的数据表达。例如在一个成员数据库的类Corporation中用"Texas"来表示某个公司的地址，而在另一成员数据库的类Company中用"TX"来表示同一公司的地址。这类冲突使用语义值的概念来解决是比较容易的，只要将表示同一公司地址的多种表达，如"Texas"、"TX"、"Tx"等，在全局数据库中进行统一即可，这一点通过转换函数很容易做到。量纲冲突是指在两个相关的类C1和C2中含义相同的属性A1和A2具有不同的量纲表示。量纲冲突也是通过语义值来加以解决的，解决的过程如下：分别定义发生量纲冲突的局部数据库的语义值模式和语义值说明，然后再定义全局数据库中相应的语义值模式和语义值说明，最后定义相关的转换函数，使发生量纲冲突的属性值在全局数据库中得到统一。精度冲突是指在两个相关的类C1和C2中相同含义的属性具有不同的精度，它的解决比较容易，在全局数据库中将发生精度冲突的数据项定义为最高精度即可。

四、行为方法级冲突分类及其解决策略

4.1 方法的提炼

在一个超类与子类层次中，子类中的对象实例可以继承超类中的结构信息与行为特征，在面向对象的数据模型中应允许在定义子类的框架中对继承来的方法进行提炼加工，以适合于操作子类中的对象实例。下面通过一个例子来说明方法的提炼过程。设类cylinder是类pipe的一个超类，在这两个类中都需要一个体积计算公式volume。当cylinder定义了一个volume方法以后，子类pipe中的对象实例就可以继承其超类中的方法volume来计算其体积，但这时的计算公式却不正确。使用方法提炼这一概念可以解决这一问题，具体的提炼过程如下：

```
type cylinder is ENCAPSULATION volume, height, radius
  STRUCT [radius:float;height:float;]
  METHODS DECLARE volume:→float;...
  IMPLEMENTATION
    DEFINE volume is return self. radius * *2.0 *3.14
          *self. height;...
end type cylinder;
type pipe is SUPERTYPES cylinder ENCAPSULATION
    innerRadius
  STRUCT [innerRadius:float;]
    METHODS
      DECLARE hollowBodyVolume:→float;
      REFINE volume;→float;...
    IMPLEMENTATION
        DEFINE hollowBodyVolume is return self. innerRadius * *2.0*3.14* self.height;
      DEFINE volume is return supervolume-self. hollowBodyVolume;...
end type pipe;
```

在上面的例子中可以看出，需要提炼的继承方法在子类定义中通过REFINE子句来指明。在IMPLEMENTATION子句中要给出被提炼方法的新编码，super.volume表示使用提炼前的版本。

下面讨论行为方法级冲突问题。行为冲突是面向对象的多数据库系统所特有的冲突类型，是指在几个相同含义的类中意义相同的行为方法发生冲突。行为冲突可以分为以下两种类型：方法签名冲突和方法体定义冲突。

4.2 方法签名冲突

当在等价的几个类中含义相同的方法具有不同的方法签名时就发生了方法签名冲突。它包括：(1)方法命名冲突；(2)方法调用参数冲突；(3)方法返回参数冲突。

①方法名冲突。当在等价的几个类中含义相同的方法定义为不同的名字，或含义不同的方法定义为相同的名字时就发生方法名冲突。它也可分为两类：方法名同义词冲突和方法名同形异义词冲突。在两个相关的类C1和C2中若两个含义相同的方法M1和M2具有不同的名字时发生了方法名同义词冲

突。方法名同义词冲突有两种可能的解决方法：(1)对发生方法名同义词冲突的方法进行换名。即 rename M1 of Cl to M2；或 rename M2 of C2 to C1；(2)建立方法名同义词字典以记载发生冲突的方法名同义词。

在两个相关的类 C1和 C2中若两个含义不同的方法具有相同的名字时就发生了方法名同形异义词冲突。它也有两种可能的解决方法：(1)对发生方法名同形异义词冲突的方法进行换名，即 rename M of C1 to M1；或/和 rename M of C2 to M1；(2)建立方法名同形异义词字典以记载发生冲突的方法名同形异义词。

②方法调用参数冲突。是指含义相同的方法具有不同的调用参数定义，具体的冲突表现有调用参数个数不同和参数的数据类型不同。在两个相关的类 C1和 C2中，若两个含义相同的方法 M1和 M2具有不同的调用参数个数时就发生了方法调用参数个数冲突。在全局概念模式中重新定义一个全局统一的方法。

在两个相关的类 C1和 C2中，若两个含义相同的方法 M1和 M2具有不同的调用参数类型时就发生了方法调用参数类型冲突。在全局概念模式中重新定义一个全局统一的方法。

③方法返回参数类型冲突。是指含义相同的方法具有不同的返回参数定义，即返回参数的类型可能不同。在两个相关的类 C1和 C2中，若两个含义相同的方法 M1和 M2具有不同的调用参数类型时就发生了方法返回参数类型冲突。在全局概念模式中重新定义一个全局统一的方法。

4.3 方法体定义冲突

在两个相关的类 C1和 C2中，若两个含义相同的方法 M1和 M2具有不同的体定义时就发生了方法体定义冲突。它的解决策略如下：(1)如类 C1是类 C2的一个超类，则通过方法提炼的概念对 C2中的方法 M2加以提炼；反之亦然；(2)否则只能在全局概念模式 中重新定义这两个方法。

参考文献

[1] Batini,C. and Lenzerini, M., A Comparative Analysis of Methodology for Database Schema Integration, ACM Computing Surveys, 18(4) 1986

[2] Shoval, P. and Zohn, S., Binary-Relationship Integration Methodology, IEEE Trans. on Knowledge Eng., 6,1991

[3] Kamel, M., Identifying, Classifying, and Resolving Semantic Conflicts in Distributed Heterogeneous Databases: A Case Study, J. of Database Management, 6(1)1995

[4] Spaccapietra, S. and Parent, C., Conflicts and Correspondence Assertions in Interoperable Database, SIGMOD RECORD, 20(4)1991

[5] Kim, W. and Seo, J., Classifying Schemantic and Data Heterogeneity in Multidatabase Systems, Computer, 24(12)1991

[6] Reddy, M.P. et al., A Methodology for Integration of Heterogeneous Databases, IEEE Trans. on Knowledge and Data Eng., 6(6) 1994

(上接第94页)

更是涉及到系统环境的很多方面，所以对其进一步优化，提高其运行效率是一项需要持续坚持的细致工作；3)继续跟踪 ODP 和 CORBA 的发展，使 OSEware 最终能和其它的同类系统进行互操作。

参考文献

[1] 唐雪飞，开放系统中的互操作性研究，电子科技大学博士论文，1996

[2] ISO/IEC JTC 1/SC 21/N 7047, Working Document on Topic 9.1-ODP Trader, 1992

[3] OMG,X/Open, CORBA 2.0 Pre-publication draft, 1995/6/2

[4] OMG, X/Open, The Common Object Request Broker: Architecture and Specification, Revision 2.0, Updated July 1994

[5] W. Rosenberg and Denney, Understanding DCE, Open System Foundation, 1992

[6] J. Shirley, Guide to Writing DCE Application, Open System Foundation, 1992

[7] Dimitri Konstantas, Design Issues of a Strongly Distributed Object Based System, Proc. of 2nd Intl. Workshop for Object-Orientation in Operating Systems(I-WOOOS'91), IEEE, Paloalto,1991

[8] Jack C. Wileden et al., Specification Level Interoperability, CACM, 34(5)1991

[9] James M. Purtilo and Joanne A. Atlee, Module Reuse by Interface Adaptation, Software Practice & Experience, 21(6)1991

[10] E. N. Elnozahy et al., Experience Using DCE and CORBA to Build Tools for Creating Highly-Available Distributed Systems, Intl. IFIP Workshop on Open Distributed Processing, Feb. 1995