第 36 卷 第 3 期 计 算 机 工 程 2010 年 3 月
Vol.36 No.3 Computer Engineering February 2010
·软件技术与数据库· 文章编号：1000—3428(2010)03—0058—03 文献标识码：A 中图分类号：TP311.52

PIM 到关系 PSM 的转换方法研究

何 曼，刘湘伟，郝成民
(合肥电子工程学院软件工程中心，合肥 230037)

摘 要：针对模型驱动体系结构中的模型转换问题，提出一种从 PIM 关系类图到 SQL 关系 PSM 的二步转换法，通过依赖、泛化、关联 3 类关系的初步转换，使其脱离关系线，得到仅与属性类型相关的 PIM 类图，并对不同类型的属性制定相应的转换规则，利用一个具体示例展示了转换过程及其结果。实验结果表明，与简单的关联关系转换相比，该方法更全面，能够完整实现模型转换。
关键词：模型驱动体系结构；模型转换；关系；对象约束语言

Research on Transformation Method from PIM to Relation PSM

HE Man, LIU Xiang-wei, HAO Cheng-min
(Center of Software Engineering, Electronic Engineering Institute, Hefei 230037)

【Abstract】 Aiming at the model transformation problem in Model Driving Architecture(MDA), a two step transform method is proposed, which transforms PIM relation diagram to SQL PSM. It makes the depend, inherit, associate away from the relation line to PIM class diagram by the first transformation. The PIM class diagram is only about with the type of attribute. The transformation rules to different attributes are introduced. A practical example is given to show the transformation process and result. Experimental results show that, compared with the simple association transformation, this method is more comprehensive and more material, the model transformation is realized more perfectly.
【Key words】 Model Driving Architecture(MDA); model transformation; relation; Object Constraint Language(OCL)

1 概述

模型驱动体系结构(Model Driving Architecture, MDA)是 OMG 最新提出的软件体系结构方法学，是种基于模型的软件开发框架。它将模型不仅作为设计语言，还作为编程语言，大大提高了模型的重用价值，从而提高软件的开发效率以及平台的可移植程度和互操作性。在 MDA 中，软件开发过程是由对软件系统的建模行为驱动的，它将模型分为 4 大类[1]：计算无关模型(CIM)，平台无关模型(PIM)，平台相关模型(PSM)和代码实现模型(ISM)，通过模型的一步步转换来完成整个系统的开发，其中最关键的步骤是从 PIM 模型到 PSM 模型的转换，虽然已经提出了许多转换方法，但目前还没有形成一个统一的方案。

在模型转换过程中，PIM 模型中 UML 类的类名、属性、操作等概念与目标代码中的相应概念一一对应，比较容易定义转换规则；而类中的关系，如依赖、泛化、关联等概念却与代码中的表达形式不一致，为其定义一套全面且准确的变换规则成为用 MDA 方法开发软件的难点。文献[2-3]给出利用元模型之间的映射，将 PIM 模型转换成 SQL 代码和 Java 代码的模型转换方法。但它们都只研究了关联 PIM 的转换。

本文在此基础上对 UML 类图中所有的关系类图到 SQL 表的转换进行研究，提出一种简单易行的二步转换法，对类图中的各种关系进行初步转换，使各种关系以不同的属性类型表现出来，对不同类型的属性定义相应的转换规则，并将转换规则用 OCL(对象约束语言)形式化表达，以便能被转换工具识别。

2 基本概念

2.1 MDA模型转换

在 MDA 中，模型转换是系统能否实现的关键技术，它描述了由一种语言所描述的模型是如何转换到由另一种语言所描述的模型的。它由一组规则集合所形成的转换规则库、源模型、目标模型和转换工具 4 个组成部分构成。其中，转换规则最关键，其定义一般都需要包括下列信息[2]：源语言引用，目标语言引用，可选转换参数，一组命名的源语言模型元素(称为 *S*)，一组命名的目标语言模型元素(称为 *T*)，双向标记，源语言条件，目标语言条件。换言之，模型转换就是指通过转换规则，将源模型转换为目标模型，最后输出这个目标模型的过程。

2.2 对象约束语言

对象约束语言(OCL)是种能用于构造软件模型的建模语言，它被 OMG 定义为 UML 标准的附加语言，用于面向对象的分析与设计。OCL 能制定各种约束和查询，功能强大，除了可用于编写导航表达式、布尔表达式和其他查询语句以外，它还可以用于构建约束、监护条件、动作、前置条件与后置条件、断言和其他 UML 表达式。它在 MDA 中的作用主要表现在 3 个方面：(1)精化 PIM 模型；(2)形式化描述转换规则；(3)定义建模语言。

2.3 UML关系类图

关系是事物间的联系。在面向对象的建模中，最重要的 3 种关系是依赖、泛化和关联。在 UML 类图中，把关系描述成一条线，并用不同的线区别关系的种类。

(1)依赖

依赖是种使用关系，它说明一个事物规格说明的变化可能影响到使用它的另一个事物，但反之未必。在大多数情况

作者简介：何 曼(1984－)，女，硕士，主研方向：软件测试，软件工程；刘湘伟，教授、博士生导师；郝成民，讲师
收稿日期：2009-10-10 **E-mail**：heman@2008.sina.com

下，在类的语境中用依赖指明一个类把另一个类作为它的操作参数或属性类型。

(2)泛化

泛化是一般事物和该事物的较为特殊的种类之间的关系。泛化意味着子类继承父类的特性，特别是父类的属性和操作。

(3)关联

关联是种结构关系，它就像整个系统中的“胶粘剂”，把系统中的各个类通过它们之间相关联的信息连接起来。

在本文中，仅考虑 3 种关系中对属性的影响，因为将类图转换成关系 ER 图时只需要对属性进行转换，不需要考虑操作。

3 PIM中关系类图的初步转换

将 UML 类图转换成 SQL 中的表是比较简单的转换，只需把 Class 转换成 Table，把属性转换成列就行。但类图往往不是单一存在的，它们之间都通过各种各样的关系紧密联系在一起。这就给转换带来了一定的难度，直接将这些关系表现在 SQL 表中，转换规则会比较复杂，转换起来也比较麻烦。所以本文提出先将 PIM 中的类图进行初步转换，使类图间的关系脱离关系线，表现在不同类型的属性中。

3.1 依赖关系转换

在 UML 建模中，一个类把另一个类作为它的属性类型有 2 种情况：(1)被依赖的类为简单数据类型，即只有属性没有操作的类；(2)被依赖类为普通类。在第(1)种情况下，将关系转换成 SQL 中的 ER 表，最好的方法是把数据类型内联到依赖类的表中。所以，在进行初步转换时，仅针对被依赖类是数据类型的依赖关系。其转换规则是：对于 PIM 中属性类型为 DataClass 类型的依赖属性，用 DataClass 类中的所有属性来代替。其 OCL 的形式化描述为

```
Transformation DependencyToAttributes(UML,UML) {
source
  att:UML::Attribute;
      dt:UML::DataType; --UML 图中的属性和数据类为源对象
target
      attributes:UML::Attributes;--属性为目标对象
source condition
att.oclIsTypeOf(dt:OclType); --变换的初始条件是属性类型为 dt 数据类
target condition  --none
unidirectional;
mapping
dt.attributes() <-> att.class.attributes();
--dt 数据类的属性转换成 att 类中的属性
}
```

3.2 泛化关系转换

泛化关系意味着继承，子类继承父类的所有属性。在转换时将父类的属性内联到子类中。转换规则和依赖关系的初步转换规则相似。只是变换的源对象为父类的属性，目标对象为子类的属性，变换初始条件为子类与父类之间存在继承关系，变换内容是把源对象的属性转换成目标对象的属性。

3.3 关联关系转换

关联关系中存在 2 个关联端，关联端的多重性不一样，转换规则不一样。

(1)对多重性为 0 或 1 的无向关联端，对面的类中存在一个与关联端同名的公有属性，且属性类型为关联端旁边的类。形式化描述如下：

```
Transformation SingleAssociationToAttribute(UML,UML) {
source
      ae:UML::AssociationEnd;--UML 图中的关联端为源对象
target
      att:UML:: Attribute;--UML 类属性为目标对象
source condition
      ae.upper<=1;--变换的初始条件是关联端多重性为 0 或 1
target condition
      att.visibility=VisibilityKind::public and
      att.type.isTypeOf(class)；--结果条件是产生 UML 类的公有属性
unidirectional;
mapping
ae.name <-> att.name;--关联端名称的小写转换为属性名称
ae.type <->att.type;
}
```

(2)对多重性大于 1 的无向关联端，对面的类中存在一个与关联端同名的公有属性，且属性类型是 Set。其变换规则的形式化描述与上一规则类似，只是属性的类型由 Class 变为 Set。由于篇幅关系这里就不再详述。

(3)对于有向关联，上述 2 个规则只用于箭头反方向的类。

3.4 转换规则的组合

当关系不是单独只在 2 个类之间存在，而是将多个类联系在一起时，不同类与关系的转换就存在一个优先级问题，需要对规则进行优先级排列。相依赖的 2 个类，被依赖的那个类要先进行转换，因为依赖它的类要转换它的属性，所以被依赖类若有其他的关系时，要先进行转换。子类父类的优先顺序亦然，父类高于子类。相关联的 2 个类转换顺序不受影响。因此，被依赖类和父类中的关系转换级别最高。

以上是对 PIM 模型的精化，属于 PIM 到 PIM 的转换。接下来便要进行 PIM 到 SQL PSM 的转换。

4 PIM到SQL关系PSM的转换

通过上一节的转换，PIM 类图中属性的类型主要有 4 种情况：属性类型为普通的 UML 数据类型，如 Int, string 等，且对一个实体只有一个属性值；属性为普通的 UML 数据类型，但对于一个实体可以有多个属性值，即多值属性；属性类型为另一个 Class；属性类型为 Set。不同情况下对应的转换规则不同。

4.1 单值属性的转换

属性类型为 UML 中的普通数据类型，则只需要将属性直接变换成 SQL 中的列即可。

```
Transformation AttrToCol-1(UML,SQL) {
source
      att:UML::Attribute;--UML 图中的属性为源对象
target
      column:SQL::Column;--ER 表中的列为目标对象
source condition
att.type.isTypeOf(UMLDataType); --变换的初始条件是属性类型为 UML 元模型中的基本数据类型
target condition
      column.nullable = true;
unidirectional;
mapping
 att.name <->column.name;
```

```
att.type <->column.type;
}
```

4.2 多值属性的转换

属性为多值属性，即一个实体可能含有一个或多个值。将此类属性转换成 SQL 表时，需要创建一个新表来包含这些多值属性。转换规则为：(1)PIM 中的一个多值属性转换成一个同名的 SQL 表；(2)SQL 表中有一个属性名加“ID”的主键列；(3)SQL 表中有一个与属性名称相同、类型相同的单值列；(4)SQL 表中有一个属性所在类的类名加“ID”的外键列。其形式化描述为

```
Transformation AttrToCol-2(UML,SQL) {
params
      tidname:String =“ID”;
      tidType:SQLDataType=INTEGER;
source
        att:UML::Attribute;   --UML 图中的属性为源对象
target
        table:SQL::Table;
        primary:SQL::Key;
        tid:SQL::Column;
        lib:SQL::Column;
        col:SQL::Column;
foreign:SQL::ForeignKey;--SQL 表、列、主键、外键为目标对象
source condition
        att.MultiValued=ture;      --变换的初始条件是属性类型为多值属性
target condition
        table.primary=primary                    and
        tid.table=table                  and
        tid.key=primary                  and
tid.nullable = false                  and
lib.table=table                  and
lib.key =foreign                  and
col.table=table;
unidirectional;
mapping
att.name+tidname <-> tid.name;
 att.type <-> primary.Key;--属性名加“ID”变换为主键
att.name <-> table.name;--表名为属性名
att.name <-> col.name;
att.type <-> col.type;--表中有一个同名同类型的属性
att.class.name+tidname <-> foreign.name;
att.class.type <-> foreign.referencedKey;
--属性所在类的类名加“ID”变换为外键，指向属性原来的类
}
```

4.3 Class属性的转换

属性类型为 Class，该属性对应的列包含外键，外键指向表示 Class 的表，外键名称即是属性名加“ID”后缀，也是 Class 的主键名。

```
Transformation AttrToCol-3(UML,SQL) {
params
      tidname:String =“ID”;
source
        att:UML::Attribute; --UML 图中的属性为源对象
target
        foreign:SQL::ForeignKey;--SQL 中的外键为目标对象
source condition
        att.type.oclIsTypeOf(Class);      --变换的初始条件是属性类型为 UML 元模型中的类
target condition
        column.nullable = true;
unidirectional;
mapping
att.name+tidname <-> foreign.name;
att.type <-> foreign.referencedKey;
}
```

4.4 Set属性的转换

属性为 Set 类型时，表明该属性可能包含有多个 Set 中的对象，也可以理解为属性为多值属性。不同的是，Set 类型是一组类的集合，它的取值是一个或多个类，且这个类表是已经转换生成的，所以，无需再为 Set 属性创建一个新表，只需在表中添加一个外键。如果在初步转换前，Set 属性所联系的两个类之间是有向关联，则外键通过所在类中的 Class 属性已经添加，转换时无需进行任何操作。

转换规则为：如果 Set 列集的实例类 SLL 中有一个与 Set 属性所在类 CLL 同名的 Class 属性，则将此 Set 属性转换成 SLL 表中的列时，无需做任何操作；否则在 SLL 的 SQL 表中添加一个名为 cllID 的外键。

由于在 SQL 表中添加外键的形式化表述上一节中有所描述，这里不再重复，只将转换规则的前提判断条件用 OCL 语言表示如下：

```
source condition
      att.type.isTypeOf(Set);
        att.type.elementType=cl;
   Bag(cl.attributes().name)::excludes(att.name:
cl.attributes().name)=true;
```

--变换的初始条件是属性 att 的类型为 cl 类的 Set 列集，且 cl 中没有名为 att 的属性

在整个转换过程中，除了本文所述的转换规则外，当然还包含有其他一些基本的转换，如：UML 类到类，属性到属性，UML 类到 SQL 表，UML 数据类型到 SQL 数据类型的转换；以及主键列、外键列的生成等[2]。这些转换比较显而易见，不再赘述。

5 转换示例

下面以一个简单的类图为例，展示上述方法的转换结果。建立一个简单的实体关系类图，如图 1 所示。

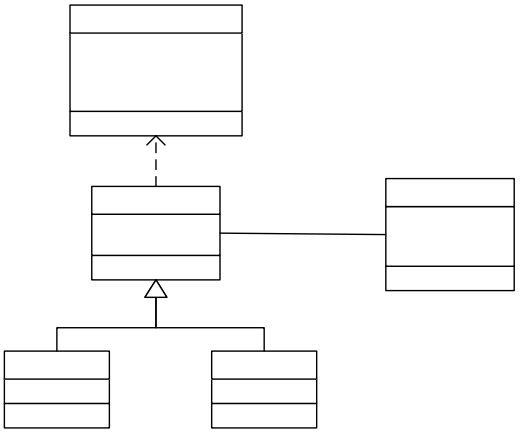

图 1　PIM 关系类图

从图 1 可以看出，一个学校对应有多个学生，一个学生可能有一个联系方式，也可能有 2 个或 3 个，学校的地址属

（下转第 63 页）