对象关系模型的数学分析和应用①

Mathematical Analysis and Application of Object Relational Model

佟 辉 (东信北邮信息技术有限公司 北京 100083)
廖建新 (北京邮电大学网络与交换技术国家重点实验室 北京 100876)
(东信北邮信息技术有限公司 北京 100083)
阮友森 (大唐软件技术股份有限公司 北京 100083)
王 纯 (北京邮电大学网络与交换技术国家重点实验室 北京 100876)
(东信北邮信息技术有限公司 北京 100083)

摘要: 目前大多数数据库都是关系数据库,在关系数据库中数据是以表的方式存储的,由于有比较好的数学理论基础支持,这种关系数据库被大家广泛接受与使用。人们在进行面向对象程序设计时大量使用数据库,随之而来的也出现了一些问题,如"阻抗失配"。本文介绍了一种实现经典的关系与对象的互相转化的方法,并对这个对象属性和关系的转换模型进行了数学分析。提出了一种通过这个模型实现数据词典的自动生成和数据库柔性管理系统的设计方法。

关键词: 对象关系模型 面向对象数据库 ORM (Object Relational Mapping) MDA (Model - Driven Architecture)

1 概述

面向对象的软件设计方法和关系数据库理论是现代软件开发设计中不可缺少的两个部分。目前关于面向对象开发技术与数据库技术的映射模型也是一个研究的热点。本文将分析对象模型和关系模型两者之间存在的关系,并用数学的方法加以描述、分析;同时应用对象关系模型提出一种柔性管理系统的实现方法。

本文组织结构如下:第 2 节介绍了经典的关系模型和面向对象模型各自的特点;第 3 节提出了一种对象模型的数学表示方法,同时提出了一种对象关系模型的数学表示方法;第 4 节提出了一种对象关系模型的应用:通过定义对象属性树动态生成数据词典和柔性管理系统;第 5 节总结全文。

2 经典的关系模型与面向对象模型

2.1 关系模型

1970 年 6 月,IBM 公司的高级研究员 Edgar Frank Codd 的一篇论文[1]开创了关系数据库的新时代。由于关系数据库是建立在严格的数学理论基础之上的,所以得到了相当广泛的发展,30 多年来,它的应用领域越来越广,影响的范围越来越大。

在关系模型中,用二维表结构来表示实体及实体之间联系。包括关系名、属性名和关键字,元组等概念。其中关系名就是表名,属性名就是表的字段名,关键字和数据库中的关键字是一个概念,元组就是记录。关系中属性个数称为元数,元组个数为基数。分别对应表列的个数和表中记录的个数。这些概念都是以集合代数理论为基础的。

2.2 面向对象模型

面向对象模型是为了描述客观世界而建立的一种模型,它的特点是通过可构造的手段将客观世界表示出来,可以用有限的构造手段与有限的步骤建立一个客观世界的模型。这种模型是在面向对象的设计方法中总结出来的,其中的好多原始思想都来自于 Simula

① 基金项目:国家杰出青年科学基金(No. 60525110);国家 973 计划项目(No. 2007CB307100,2007CB307103);新世纪优秀人才支持计划(No. NCET-04-0111);电子信息产业发展基金重点项目(下一代网络核心业务平台)资助项目。

和 Smalltalk 语言程序设计。

对象是客观世界中概念化的基本实体，是通过对客观实体的抽象得到的。对象由属性和方法组成，可以把它看成模型和驱动的组合体（其中模型对应属性，驱动对应方法）。我们这里只讨论模型（属性）部分。为了描述简单，本文约定下面的对象专指对象模型（属性）部分。

对一组有相同属性和相同方法的对象在面向对象模型中称之为类。对象是类的具体化，是类的实例。如果一个类 A 的所有属性和方法都在另一个类 B 中存在，则可以称类 A 是类 B 的子类，类 B 是类 A 的父类。

对象之间的关系可以分成三种：

（1）继承关系。一般与特殊的关系。具有传递性和单向性。

（2）组成关系。整体与部分的关系。表现为嵌套性、引用性、递归性等。

（3）通信关系。基于消息的关系。

面向对象模型已经被广泛地应用于程序设计中，应用封装性实现对类的属性和方法的抽象，同时设定其访问范围（共有，私有）；应用继承性实现子类和父类的划分，提高类的使用效率；应用多态性实现类方法的重载和动态绑定，优化程序结构；应用组成关系时现一个类对另一个类的引用，减少代码的冗余；应用通信关系构造消息实现对象与对象之间的相互作用等等。

2.3 两种模型的比较

关系模型有利于进行分析管理，因为其简单，且数据冗余小；对象模型由于直接描述客观实体，因为其结构和客观世界更接近，但管理、分析起来复杂，数据有冗余。

能不能建立一种逻辑关系，使其二者联系起来，且能够自由转换，来适应各种不同的数据存储使用要求呢？下面我们通过一个例子来引入对象关系模型。

3 对象的关系模型

3.1 一个例子

（1）表示成关系有下面的三个表：

分别是学生表 S，课程表 C，学生选课表 SC。

（2）表示成对象关系

这里把学生和课程这两个对象各自的属性也都看成了对象，这样在这个对象关系图 1 中每一个节点都是一个对象，他们之间的连接线表示对象之间存在的关系，其中 Rs 表示学生关系，Rc 表示课程关系，Rsc 表示学生选课关系。

表 1　学生表 S

学生编号 SNO	学生姓名 SN
s1	张三
s2	李四

表 2　课程表 C

课程编号 CNO	课程名称 CN
c1	数据结构
c2	编译原理

表 3　学生选课表 SC

学生编号 SNO	课程编号 CNO
s1	c1
s1	c2
s2	c2

从图 1 中可以看出对象可以分成两种：

元对象：这种对象只作为其它对象的属性存在，其本身没有任何子属性。我们称其为元对象，符号 Ot。

图 1 中学生编号，学生姓名，课程编号，课程名称都是元对象。

简单对象：由一个以上的元关系对象组成的对象，我们简称其为简单对象，符号 Os。

图 1 中学生，课程都是简单对象。

另外还有一种对象，他本身是简单对象的继承（或组成），同时还包含其它的元对象和简单对象，我们称为复杂对象，几个复杂对象组成的对象也是复杂对象。符号 Oc。

如果学生继承自人这个对象的话，那么人相对于学生来说就是简单对象，而学生就是复杂对象了。

3.2 多对象与对象之间的关系描述

图 1 中我们还可以看出学生和课程之间存在联接关系。由于一个学生可以选择一门或多门课程当然也可以不选，同时一门课程也可以被零个、一个或多个学生选择，所以学生和课程之间是多对多的关系。

此外对象之间还存在着多对一和一对一的关系。主键列;多对多的对象关系可以用三个表来表示,分别

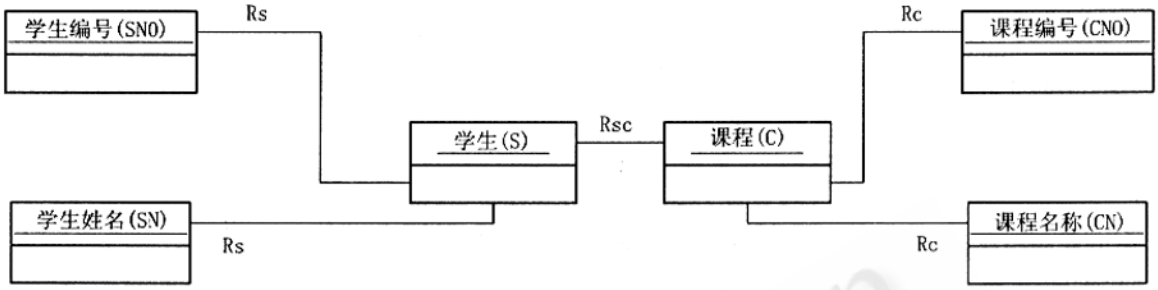

图 1　学生选修课程

设对象 A 和对象 B 存在联接关系,用 Oa 表示对象 A,用 Ob 表示对象 B,A(Oa) = Ra,A(Ob) = Rb,Oa 和 Ob 的主键设为 Rpka 和 Rpkb,定义 PK(On)为取对象 On 的主键操作,则可得下面的表达式。

PK(Oa) = Rpka,PK(Ob) = Rpkb

则 Oa 和 Ob 的关系可以表示成 R(Oa,Ob) = PK(Oa) × PK(Ob) = Rpka × Rpkb。

也就是说,对象之间的联接也可以用关系模型来表示。我们用 × 来表示对象的联接,则有:A(Oa × Ob) = A(Oa) * (PK(Oa) × PK(Ob)) * A(Ob),其中 * 为自然连接符。

下面我们讨论一下联接化简方案[3]。

(1) 一对一联接。由于这种连接的特点,我们可以直接把它们用一个关系来描述,如下:

A(Oa[1×1]Ob) = A(Oa) × A(Ob) = Ra × Rb

(2) 一对多联接。在这种联接中,如果 Ob 是被多联接的对象,Oa 是相对于 Ob 的单一联接对象可以表示成 Oa 的对象关系中引入 Ob 对象关系的主键域组成一个关系再同 Ob 的对象关系自然联接。

A(Oa[1 × n]Ob) = (A(Oa) × PK(Ob)) * A(Ob) = (Ra × Rpkb) * Rb

(3) 多对多联接。这种关系只能引入一个新的关系来表示其中的联接,再分别与两个表示对象的关系自然联接。

A(Oa[n × n]Ob) = A(Oa) * (PK(Oa) × PK(Ob)) * A(Ob) = Ra * (Rpka × Rpkb) * Rb

也就是说,一对一的对象关系可以用一个表来表示;一对多的对象关系可以用两个表来表示,其中每个表表示一个对象,其中的一个表中增加另一个对象的是描述每个对象一个表,描述两个对象的关系一个表(可以根据需要在描述两个对象的关系的表中增加描述两个对象关系的属性的列)。

在图 1 所表示的学生选课的例子中由于学生和课程是多对多的关系,所以可以用三个表来表示,如表 1,表 2,表 3。

4　对象关系模型的应用

根据上面的模型,我们在关系数据库的基础上,通过引入描述这些对象的属性和各个对象之间关系的对象关系属性树[4],就可以在关系数据库的基础上建立面向对象的数据库系统。

通过对象的关系属性树,数据库系统本身就可以了解到数据库中的关系表和对象本身的逻辑关系,从而可以提供基于对象属性的数据库维护服务(如查询、修改、删除等),这些对象的方法是自动生成的,而且随着对象关系属性树的变化而变化。

我们在进行项目开发时,其数据词典的定义也可以由 IDE (Integration Development Environment 集成开发环境)或数据库管理系统根据对象属性树自动生成,即提供数据词典生成引擎服务。而我们要做的只是根据需求定义对象关系属性树。

还有就是对这些数据本身的维护界面也可以由数据库系统根据定义的对象关系属性树自动生成,从而极大的简化开发人员的工作量。

对于图 1 中描述的系统,通过对象关系属性树,数据库根据统一的命名规则可以自动建立三张表,如 t_student、t_course、t_student_course;对于对象来说,其对应的表的字段由对象的属性来决定,如果该对象和

其他对象存在一对多的关系时还要加入其它对象的主键属性字段；对于由对象多对多关系形成的表的字段由其相关对象的主键属性字段组成，如果还有描述这个多对多关系本身的属性，那么也要将其作为这个表的字段加入这个表。例子中由对象关系树可知 t_student 表由学生编号(c_sno)，学生姓名(c_sn)两个字段组成。

这些表与字段的信息也同时被数据库系统保存在对象关系属性树中。如图 2 所示。

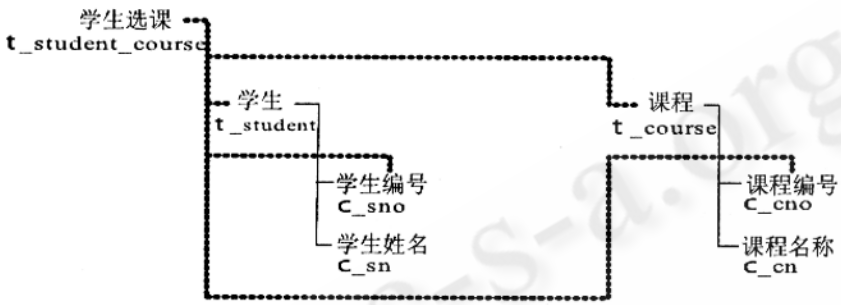

图 2　学生选修课程 对象关系属性树 1

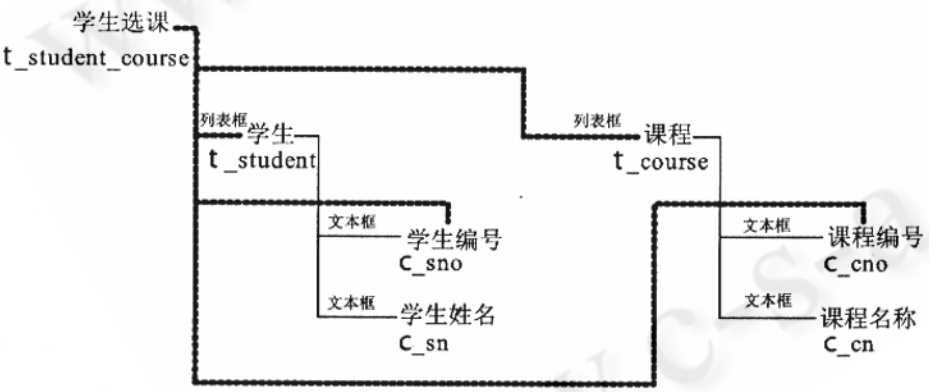

图 3　学生选修课程 对象关系属性树 2

其中实线表示该树权的根是实际定义的对象，虚线表示该树权的根是对象之间的关系。图 2 中学生选课表示学生和课程两个对象之间的关系，学生编号和课程编号是表 t_student_course 中的属性列。

根据图 2 的信息，数据库应用管理系统引擎可以提供对各个对象的添删改服务和对象之间关系的配置服务，比如对于学生这个对象来说，如果要添加一个学生，则数据库应用管理系统引擎通过对象属性树得知需要将数据插入 t_student 表，同时知道需要用户提供学生编号(c_sno)和学生姓名(c_sn)的值。

如果我们在对象关系属性树中同时定义了对象属性以及对象之间关系在管理系统中的呈现方式，那么我们便可以给用户提供由数据库应用管理系统引擎自动生成的管理界面了。

根据图 3 的信息，数据库应用管理系统引擎便可以提供对这个应用系统数据的 GUI(Graphical User Interface 图形用户接口)管理维护服务。如对于例子中的这个应用系统来说其管理系统可以由三大部分组成，学生信息管理，课程信息管理和学生选课管理。而这种 GUI 管理界面又可以归结为两种，一种是对象的管理，如学生和课程，对于这种类型的管理一般都是添加、修改、删除，界面中需要管理的字段信息可以从该系统的对象关系属性树中获取；另一种是对象之间的多对多关系管理，这种管理 GUI 中所需呈现的信息同样可以从该系统的对象关系属性树中获得。此外，对于对象的管理还需按照该对象的各个实体之间是线性关系还是递归关系来划分，如果是线性关系则在呈现时可用列表框，如果是递归关系，那么在呈现时就要用树控件了。在自动生成数据词典时也和这个有关，若是递归关系，则需要在表定义中加入代表主从关系的 parentid 字段。

由上面的阐述可以看出，只要定义好针对任意一个应用系统的对象关系属性树，则不需要开发任何代码，就可以应用本系统生成该应用系统的管理维护系统。

5　总结

随着数据库系统发展的日趋成熟，人们越来越多地考虑怎样更好的用数据库来维护系统的数据，面向对象技术也已经发展得相当成熟了，而从对象到关系

(下转第 73 页)

(上接第77页)

数据库存在一个阻抗失配的问题。同时随着应用系统规模的越来越大,系统应用逻辑的更新的越来越频繁,留给开发人员的时间越来越少,对系统本身数据的维护系统的开发也占用了开发人员的大量时间精力。但这部分开发工作大多都是重复性劳动,人们越来越需要这样的一个工具,就是开发人员通过设计系统的UML(Unified Modeling Language 统一建模语言)图就可以完成所有的开发工作,而不需要编写代码。

本文希望能对 MDA(Model – Driven Architecture 模型驱动架构),ORM(Object Relational Mapping 对象关系映射)和面向对象数据库等相关领域的研究人员提供一个模型参考。

参考文献

1 E. F. Codd. A Relational Model of Data for Large Shared Data Banks. Communications of the ACM, 1983.1(26.1):64 – 69.

2 萨师煊、王珊,数据库系统概论,第三版,高等教育出版社,2000.

3 ScottWAmbler. Mapping Objects To Relational Databases.
http://www. agiledata. org/essays/mappingObjects. html#MappingInheritance.

4 王青、张为民、蔡建平,一种面向对象的可重用库管理系统的模型,软件学报,1997.8(5):391 – 396.