UML 类图在关系数据库中的实现

张　日希
（广州市职工大学，广东 广州 510030）

摘　要：UML 是目前面向对象程序设计中的一种标准的建模技术。在关系数据库系统的设计过程中，先利用 UML 建立商业模型，然后将其映射成表。主要讨论如何将 UML 类图中的类映射成表的策略。
关键词：UML；类；表；关系；建模；映射
中图分类号：TP312　　文献标识码：A　　文章编号：1001-3695(2001)12-0131-03

The Realization of UML Class Diagram in Relational Database

ZHANG Xi
(*Guangzhou Staff & Worker's University, Guangzhou Guangdong* 510030, *China*)

Abstract: UML is a standard modeling technology in object-oriented program. In the process of relational database design. We may build commercial models under UML first and map it into tables. This paper mainly discusses the artifice of mapping the classes into tables.
Key words: UML; Class; Table; Relation; Modeling; Mapping

1　概述

在关系数据库设计中，用来创建数据库逻辑模型的标准方法是使用实体关系模型(ER 模型)。ER 模型的中心思想是：可以仅通过实体和它们之间的关系合理地体现一个组织的数据模型。但这样做似乎对描述一个组织的信息过于简单化，并且词汇量也远远不足。所以，迫切需要使用更加灵活、健壮的模型来代替 ER 模型。

标准建模语言 UML 是由世界著名的面向对象技术专家发起的，在综合了著名的 Booch 方法、OMT 方法和 OOSE 方法的基础上而形成的一种建模技术。它通过用例图、类图、对象图、交互图等模型来描述复杂系统的全貌及其相关部件之间的联系。UML 可以完成 ER 模型所能做的所有建模工作，而且可以描述 ER 模型所不能表示的关系。

在 UML 中，类图主要用于描述系统中各种类及其对象之间的静态结构。在数据库领域中，类与表相对应。本文主要讨论将 UML 类图中的类及其对象映射成关系型数据库中的表的策略。

2　UML 类图中的类映射成表的策略

(1)将类图中的属性类型映射成表的域

域的使用提高了设计的一致性，且优化了应用的移植性。简单的域是非常容易实现的，在映射时仅需替换相对应的数据类型和数据尺寸。但要注意：有时可能要求在域的约束中加入 SQL 的 Check 串(如限定域的取值范围等)。

(2)将类的属性映射成表的字段

一般地，可将类的属性直接映射成表的一个字段，但要注意以下两种特殊情况：

①并不是类中的所有属性均是永久的。如发票中的"合计"属性可能是用于计算而不需保存在数据库中，此时，该类属性(称为派生属性)映射成 0 个字段。

②一般地，类中的属性是单值的，但如果在类中存在多值属性，则该属性映射成多个字段。

(3)将类直接或间接地映射成表

在关系数据库中，一个关键问题是表主键的惟一性策略的选取。适当的方案能优化继承、组合等类之间的关系的实现。为此，可以在处理数据库关系时嵌入对象标识符(OID)的概念，即采用对象标识符 OID(对象惟一的标识符)作为相应的数据库中所有表的主键，从而简化了关系数据库的主键方案，使数据库发生更新时，不会出现完整性问题，并且还可以避免在数据库操作时的诸多限制。但要注意的是，OID 不应包含有商业内涵(这一点可能和传统的关系理论相悖)，因为任何具有商业意义的字段不在控制范围之内，从而设计者面临值和规划改变的危险。

在 UML 类图中，类之间的关系依据其紧密程度的不同分为：继承、关联、聚集和组合。下面分别讨论在将类映射成表的过程中这些关系的实现技术。

收稿日期：2001-01-07

①继承的实现

策略一: 将整个类层次映射为单个数据库表。

类层次中的所有类映射为单个的数据库表, 表中保存所有类(基类、子类)的属性。例如, 图 1 的类层次模型的实现如图 2 所示。

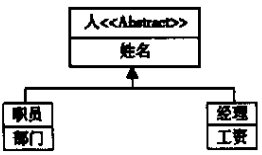

图 1

图 2

该策略实现简单, 支持多态, 报表操作实现简单。但增加了类层次中的耦合, 类层次中任何类的属性的增加会导致表的变更; 某个子类属性的修改错误会影响到整个层次结构, 而不仅仅是该子类。并且该策略还浪费了大量的数据库空间。

策略二: 每个具体子类映射成单个数据库表。

数据库表包括自身的属性和继承的属性, 每个具体的子类包含各自的 OID。抽象基类不参与映射, 它的所有属性都复制到子类对应的表中。例如, 图 1 的类层次可映射成两个表, 其中"职员"表包含"职员 OID"(主键)、"姓名"和"部门"字段; "经理"表包含"经理 OID(主键)"、"姓名"和"工资"字段。

该策略由于在表中包含了具体子类的所有信息, 所以报表操作实现简单。但类的修改会导致相对应的表及其子类所对应表的更改; 角色的更改会造成 ID 的重新赋值(因为不同子类的 ID 可能重复)。并且该策略还难以在支持多重角色时保持数据的完整性。

策略三: 每个类均映射为数据库表。

为每一个类创建数据库表, 表中包含特定于该类的属性和 OID。例如, 图 1 的类层次可映射成三个表, 其中"人"表包含"人 OID"(主键)和"姓名"字段; "职员"表包含"人 OID"(主键及外键)和"部门"字段; "经理"表包含"人 OID"(主键及外键)和"工资"字段。值得注意的是: "人 OID"作为所有表的主键。

该策略与面向对象的概念相一致, 对多态的支持最好, 并且对于对象可能充当的角色仅需要在相应的表中保存记录, 易于修改基类和增加新的类。但由于数据库中存在大量的表, 所以访问数据的时间较长, 对报表的支持较差(除非定义视图)。

②关联的实现

在 UML 类图中, 通过关联表示类的实例之间存在的某种关系。在关系数据库中, 可通过外键实现类的关联。外键允许表中的某一行与其它表中的行相关联。

为了下面论述方便, 现假定 M 表示强制(Mandatory), O 表示可选(Optional)。

情况一: 1 对 1 的关联

如果关联是 O—M, 则可将外键放置在可选的一端, 该外键不能为空值; 其它 1 对 1 的情况外键可放置在任意一边, 具体情况依赖于性能等因素。但要注意的是: 对于 1 对 1 的情况, 不要在两个表中均放置对方的主键。这样, 增加了冗余, 并且不会提高性能。对于关联的强制性, 一般在商业规则的对应层实现, 而不在物理层中实现。

情况二: 1 对多的关联

将外键放置在"多"的一方。如果"1"方是可选的, 则外键可有空值, 以表明"多"方的记录可以独立于"1"方存在; 如果 1 方是强制性的, 则外键一定要非空。

情况三: 多对多的关联

实现多对多关系, 通常需要建立一个相交表, 并把它与关系两端的表建立联系。在传统实现中, 关联表的属性包含关系中两个表的主键, 并且关联表的主键往往是它们的组合。该实现方法意味着两个表的联系是静态的, 不能跟踪一方退出某个项目然后又在某个时候加入的情况。另一种实现方法是将关联表视为普通表, 使用自身的主键 OID, 然后加入实现关系所必须的外键。例如, "项目"与"经纪人"类之间的多对多关联可用图 3 的方式实现。

图 3

使用该方法, 可保证在物理层中所有的表具有相同的形式, 从而简化了实现, 提高了运行效率。但要注意的是: 一些数据库在连接具有复合外键的表时, 性能较差; 并且, 存在着向关联表中增添字段的可能。

③聚集的实现

在 UML 类图中, 聚集描述了部分与整体之间的关系。例如"委员会"与"成员"两个类之间就存在着聚集关系。该关系在关系数据库中的实现如图 4 所示。

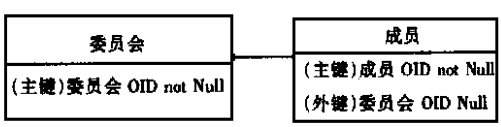

图 4

④组合的实现

在 UML 类图中, 组合由聚集演变而成, 它表示一个部分对象仅属于一个整体, 并且部分对象通常与整体对象共存亡。例如, "多边形"与"点"两个类之间就存在组合关系。

组合的映射和聚集类似, 但要注意的是: 子表中的外键必须为强制非空的。

3 引用完整性及关系约束检查的策略

在 UML 中, 类之间的关系反映了具体的商业规则; 因此将类映射到关系数据库时, 必须保证类之间关

系的正确定义,并确保在数据库中实施对数据的约束。以下就 1 对多的情形为例加以说明(1 对 1 关系可以视为特殊的 1 对多关系;多对多关系则可以分解为两个 1 对多关系)。

(1)父表操作的约束

将类的关系映射到关系数据库后,父表在操作上的约束规定如表 1 所示。

表 1

类关系	关系类型	Insert	Update	Delete
关联	数据无耦合关系则一般不映射			
	O-O	无限制	无限制,对子表中的外键可能需要附加的处理	无限制,一般将子女的外键置空
	M-O	无限制	修改所有子女(如果存在的话)相匹配的键值	删除所有子女或对所有的子女进行重新分配
聚集	O-M	插入新的子女或合适的子女已存在	至少修改一个子女的键值或合适的子女已存在	无限制,一般将子女的外键置空
组合	M-M	对插入进行封装,插入父记录的同时至少能生成一个子女	修改所有子女相匹配的键值	删除所有子女或对所有的子女进行重新分配

(2)子表操作的约束

将类的关系映射到关系数据库后,子表在操作上的约束规定如表 2 所示。

表 2

类关系	关系类型	Insert	Update	Delete
关联	数据无耦合关系则一般不映射			
	O-O	无限制	无限制	无限制
	M-O	父亲存在或者创建一个父亲	具有新值的父亲存在或创建父亲	无限制
聚集	O-M	无限制	兄弟存在	兄弟存在
组合	M-M	父亲存在或者创建一个父亲	具有新值的父亲存在(或创建父亲)并且兄弟存在	兄弟存在

施加子表的约束主要是为了防止碎片的产生。在一些情况下(如在 O-M, M-M 约束中),一个子女(子表中的记录)只有在当其兄弟存在时才能被删除或修改,即最后一个存在的子女是不能被删除或修改的。此时,可以对父记录进行即时的更新,或者禁止该操作。而子表约束可以通过在数据库中加入触发器来实现;一个更合理、可行的方法是将对子表方的限制放在业务层中实现。

应用程序在执行数据约束时有很重要的作用。在四类约束 M-M, M-O, O-M, O-O 中,键值的修改可能会改变表之间的关系,而且可能违反一些约束。违反约束的操作是不允许的。而前面的规则仅为具体实现提供了可能性,具体的应用必须根据实际的要求和商业规则进行适当的选择。但在设计和开发时,必须考虑所分析的约束。

目前,面向对象已成为软件开发的主流技术。在一个良好的项目设计中,我们可以先使用面向对象的 UML 技术建立商业模型,接着引入映射层,对将类设计映射至关系数据库的逻辑进行封装。通过物理层封装对数据库的访问细节,并为开发者提供简单而又完备的接口,从而使开发者可以运用面向对象的技术解决关系数据库领域中的问题。

参考文献:

[1] 刘超,张莉. 可视化面向对象建模技术[M]. 北京:北京航空航天大学出版社,1999.

[2] Paul Dorsey. Oracle 8 UML 对象建模设计[M]. 北京:机械工业出版社,2000.

作者简介:

张日希(1963-),男,讲师,主要研究方向为软件工程、数据库技术。

(上接第 122 页)的数据库接口的复杂性;增强了分布式客户机/服务器应用程序的可维护性,增加了数据库访问的安全机制;支持与多种关系数据库的连接。这就充分利用了企业原有的各种数据库系统,保留了原来的数据,而且克服了该厂客户端多、维护起来麻烦的缺点。

5 结束语

本文对企业 Intranet 的异构数据库的集成技术进行了研究。在进行理论研究的同时,也参加了开发企业 Intranet 管理信息系统的实践,设计了数据库原型网关 BDEGATE,实现了在企业环境下异构数据库系统的集成,保留了客户原有的数据库资源;通过具体工作以对企业的安全和数据集成要求有了更进一步的体会。并由软件实现了对系统用户的认证和访问权限的控制,做到了依照用户访问权限的不同区别访问系统资源,满足了客户对于系统安全性的要求。

参考文献:

[1] Venkatar Ramesh, Suda Ram. Integrity Constraint Integration in Heterogeneous Database. An Enhanced Methodology for Schema Integration[J]. Information System, 1997, 22(8): 423-446.

[2] Chen Wei. Design and Implementation of a Multi Database Query Interface[D]. M.S. Thesis BUAA.

[3] 姚领众,宋翰涛,等. 基于并行的异构数据库联合使用系统[J]. 计算机系统应用,1997,(12).

[4] 章勇,沈延森,等. Web 与数据库集成及其安全性技术[J]. 计算机应用研究,1999,16(3):43-44.

[5] 张心耕,张京生,等. 异构数据库互连的设计方法[J]. 计算机工程与应用,1998,(8).

作者简介:

吴大强(1977-),男,硕士研究生,研究方向为计算机网络和应用;孙亚民(1946-),男,教授,博士生导师,研究方向为计算机网络和通讯技术。