基于软件体系结构的对象持久层设计方案 *

秦奕青[1,2]，杨炳儒[2]，李 健[3]
(1. 北京机械工业学院 计算机与自动化系，北京 100085；2. 北京科技大学 信息工程学院，北京 100083；3. 中国科学院 光电科技集团国科东方光电技术公司，北京 100080)

摘 要：针对 Scott Ambler 提出的健壮持久层设计方案存在的问题，提出了一种基于软件体系结构的对象持久层设计方案。将体系结构作为对象持久层的整体视图，反映了对象持久层的整体功能和结构，并利用层模式实现了对象持久层的可再用性、可维护性、可修改性和可移植性等质量属性。作为设计方案的应用，给出了一种对象持久层的类设计模型。

关键词：对象持久层；持久化应用系统；软件体系结构；层模式；类设计
中图分类号：TP311.52 **文献标志码**：A **文章编号**：1001-3695(2008)01-0154-03

Software architecture-based design of persistence layer

QIN Yi-qing[1,2], YANG Bing-ru[2], LI Jian[3]
(1. *Dept. of Computer & Automation, Beijing Institute of Mechanical Industry, Beijing* 100085, *China*; 2. *College of Information Engineering, University of Science & Technology Beijing, Beijing* 100083, *China*; 3. *GK East Optoelectronic Technologies INC, Chinese Academy of Sciences, Beijing* 100080, *China*)

Abstract: By analyzing the drawbacks of an existing design of a robust persistence layer, presented a new one first based on the software-architecture. An architectural view of a persistence layer was a representation of the whole system from the perspective of the structure and function. As a typical architectural style, layered pattern was also utilized to achieve the qualities such as reusability, maintainability, modifiability and portability. In addition, presented a class design model of the persistence layer as an application of the design.

Key words: persistence layer; PAS(persistent application system); software architecture; layered pattern; class design

0 引言

持久性(persistence)是对象的生存特性。具有持久性的对象被称为持久对象。使对象成为持久对象的过程被称为持久化。要求持久化服务的企业级应用被称为持久化应用系统 PAS[1]。

一般 PAS 遵循三层体系结构。最上层是用户接口层，实现用户与 PAS 之间的交互；然后是业务逻辑层，实现 PAS 的业务逻辑；后台由相应的关系数据库支持。基于该体系结构，可进一步将中间的业务逻辑层进行细分，将数据存储逻辑从业务逻辑中分离出来，单独封装成独立的对象持久层，形成多层体系结构。由于对象持久层屏蔽了数据存储逻辑的全部细节，使得 PAS 对关系数据库的访问和操作完全透明，PAS 开发人员只需关注业务逻辑的实现，而不必考虑数据存储逻辑的实现。与其他持久化方法[2]相比，对象持久层对数据存储逻辑的功能级封装实现了数据存储逻辑和业务逻辑的真正解耦合，从根本上提高了软件的开发效率，增强了软件的可维护性和可扩展性。

关于对象持久层的设计方法，Scott Ambler 提出了一种健壮持久层设计方案[3]。但该方案存在两个问题：(a)基于类的设计只注重细节，没有从整体上把握对象持久层的结构特征和功能特性，不利于实现更为复杂的对象持久层的设计要求。(b)没有考虑对可再用性、可维护性、可移植性、可修改性等质量属性的支持，而这些属性对于对象持久层来说很重要，因为不同的 PAS 对数据存储逻辑会有不同的要求，所以特定的 PAS 需要在对象持久层的公共特性之外加入属于自己的内容，而上述属性可以为此提供很好的支持。

1 基于软件体系结构的软件设计方法

软件体系结构的概念及其相关思想源于 Christopher Alexander 的建筑学思想[4]。Shaw 和 Garlan 将软件体系结构抽象地定义为：对组成系统的元素、元素的相互作用、指导元素组合及设计的模式和原理，以及这些模式上的约束等的描述[5]。由软件体系结构的概念可以看出，软件体系结构不是软件系统本身，它以一种有别于模块等概念的方式使得大家能够从一个系统的整体角度来理解、描述和构造系统。

自从 20 世纪 90 年代软件体系结构及其相关思想兴起以来，复杂软件系统的设计和开发表现出从模块和类等细粒度软件构造的设计向上延伸至系统体系结构等粗粒度软件构造的设计的嵌套层次[6]。软件设计方法也从六七十年代兴起的结构化方法[7]，到 80 年代中后期兴起的面向对象方法，发展至现在的基于软件体系结构的软件设计方法。相对于面向模块和

收稿日期：2006-11-08；**修回日期**：2007-01-26 **基金项目**：国家科技成果重点推广项目(2003EC000001)

作者简介：秦奕青(1969-)，女，北京人，博士研究生，研究方向为软件技术、数据挖掘(qyq_email@sina.com.cn)；杨炳儒(1943-)，男，天津人，教授，博导，研究方向为知识发现与智能系统、柔性建模与集成技术等；李健(1965-)，男，湖南长沙人，硕士，研究方向为对象建模技术.

类等细粒度软件构造的设计方法而言,基于软件体系结构的软件设计方法首先考虑的是体系结构这种粗粒度软件构造的设计,然后在其基础之上再进行更细粒度的模块和类设计。这样有助于从一系列相关的目标出发对系统进行整体描述,特别是在大型复杂系统的设计中会表现出特别的优势。

体系结构模式(风格)和质量属性为基于软件体系结构的软件设计提供了基础。一个给定系统的体系结构具有某种特定的模式,系统通过这种模式来获得相应的质量属性[8]。例如,实现分布计算的 C/S 模式可以支持系统的简单性,用于通信的代理者模式则可以支持系统的可修改性、互操作性和可再用性等质量属性要求,而应用最广的层模式则可以支持系统的可再用性、可维护性、可移植性和可修改性等质量属性。

层模式的设计思想早在 Dijkstra 的经典文献 THE[9] 中就被提出,其核心是:如果正在设计的系统混合了低层与高层的问题,并且高层操作要依赖于低层操作,那么可以将其分成适当的层次,依次放置。层模式的主要结构特征就是下一层的服务只被上一层使用,上一层依靠下一层提供的服务来完成相应的功能。在某一层中所有用到的组件放在同一抽象层,每个独立层都要防止其上层越过它直接访问其下层。层模式在软件设计中被广泛应用,如文献[10,11]分别基于层模式为移动计算环境和分布式系统提供了体系结构设计方案。

层模式作为应用最广泛的体系结构模式,被大多数应用系统开发人员所熟悉和使用。在多层应用系统框架下,依据层模式理论建立对象持久层的体系结构是一个合理的选择。

2　基于软件体系结构的对象持久层设计方案

本文的对象持久层设计要达到两个目标,即给出对象持久层的整体视图和实现系统的可再用性、可维护性、可移植性和可修改性。由上述对比讨论中可以看出,基于软件体系结构的软件设计方法和基于层模式的设计可以实现这两个目标。首先,按照基于软件体系结构的软件设计方法,利用层模式来设计对象持久层的体系结构模型。该体系结构模型通过层次化分解定义了系统中各实体之间的关系,所以它作为全局视图实现了对对象持久层整体的表示和控制。其次,层模式本身支持可再用性、可维护性、可移植性和可修改性等质量属性,所以基于层模式建立的对象持久层可以自动支持这些要求。

2.1　对象持久层的体系结构层次模型

对象持久层位于业务逻辑层与底层关系数据库之间,它不仅要实现内部对象之间的交互,而且既要为上层业务对象提供服务,又要利用下层的关系数据库服务。这样,依据层模式理论,按照将面向对象应用与数据库应用相分离、面向对象数据库特性与关系数据库特性相分离的原则,将对象持久层的体系结构构建成两层模式,即上层是对象层,下层是数据访问层(图 1)。

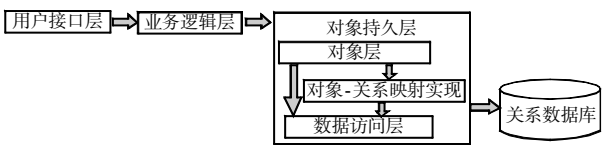

图 1　PAS 与对象持久层体系结构

上层的对象层封装了对象持久层中具有面向对象特征的功能,直接为上层业务对象提供服务,负责对 PAS 用户的持久化要求进行处理,将其转换为数据访问层可接受的形式。下层的数据访问层封装了持久层中典型的数据库功能,负责与关系数据库交互,实现关系数据库的连接以及对象数据到关系数据库的持久存储。

对象持久层的一个重要工作就是实现对象数据与关系数据之间的转换。这种转换依据制订的对象关系映射原则,通过建立对象层与数据访问层之间的联系,将可持久化对象的数据从对象层映射到数据访问层,将对象、对象属性和对象关联等转换为关系数据库中的表、记录、属性等;反之亦然。对象—关系映射功能模块是对象层与数据访问层之间众多关联中最重要、最主要的一个,所以在体系结构中特别表示出来,但它并不构成一个单独的层,为了方便,称之为对象—关系转换子层。

2.2　对象持久层的体系结构组件模型

对象持久层的体系结构层次模型只是反映了对象持久层的结构特征,而其功能特性还要通过组件集成来实现。在对象持久层的层次模型基础上,笔者为每层设计了组件。这些组件能为上层提供接口,实现被请求的服务。组件模型如图 2 所示。

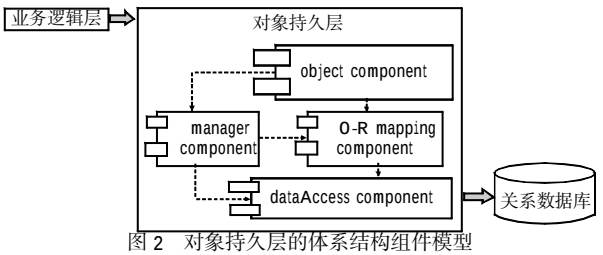

图 2　对象持久层的体系结构组件模型

Object 组件分布在对象层,为上层业务对象提供接口服务,将对象持久化的要求传送到数据访问层,并实现可持久对象的管理。O-R mapping 组件分布在对象—关系转换子层,为 object 组件提供对象结构到关系结构的转换接口,使得对象属性及其关系能够按照规则映射到数据库语义结构上。DataAccess 组件分布在数据访问层,为 object 组件和 O-R mapping 组件提供到底层关系数据库的连接和操作接口。此外,为了减少组件对象之间的依赖关系,更好地支持系统要求的质量属性,笔者特别设计了 manager 组件,用来负责整个持久层的管理和协调工作,实现对象层与数据访问层之间的通信,其他组件对象通过 manager 对象实现交互。至此,笔者利用基于软件体系结构的软件设计方法,设计完成了对象持久层的体系结构模型。该模型作为对象持久层的整体视图,给出了对象持久层的整体结构和功能框架。而且,该模型通过层模式获得了可再用性、可维护性、可修改性和可移植性等质量属性。

3　基于对象持久层体系结构的类设计模型

本文设计完成的对象持久层体系结构模型作为总体的结构和功能框架,可以支持对象持久层的模块和类等细粒度软件构造的设计。在此,笔者给出一种基于该体系结构的对象持久层的类设计模型作为具体的应用说明。

从组件的构造设计入手,参考对象持久层的体系结构模型,进一步应用层次化分解方法,将模型中的组件设计成由更低层的组件复合而成、不同的低层组件能够合作完成一个高层组件的功能。这样可以进一步提高软件的可再用性。在对象持久层系统开发中,由低层组件复合而成的组件本身不对应任何实现体,而构成复合组件的低层组件是无须再分的最小基本单元,有其对应的实现体,即高级面向对象程序设计语言支持的类结构。因此,object 组件由实现对象层功能的类构成;O-R mapping 组件由实现对象—关系映射的类构成。DataAccess 组件由支持对关系数据库的访问操作的类构成;Manager 组件由相应的持久化管理者类构成(图 3)。

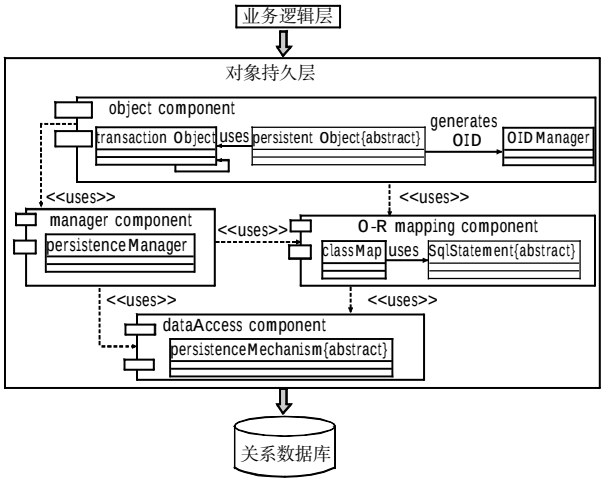

图 3　对象持久层的类设计图

组件中的各个类相互协作完成相应的功能。Object 组件中的抽象类 persistentObject 封装所有持久化对象需要的持久化操作接口,所有有持久化要求的业务类均将它作为根类继承;persistentObject 类使用 OIDManager 类为持久对象生成对象标志;transactionObject 类为业务对象提供事务支持。O-R mapping 组件中的 classMap 类(层次)实现类到关系表的映射,而 SqlStatement 类(层次)依据 classMap 中封装的信息构造 SQL 语句;这两个类(层次)实现 object 组件中对象到关系数据库之间的映射逻辑。DataAccess 组件中的 persistenceMechanism 类(层次)封装 ODBC 或 JDBC 接口,实现对关系数据库的连接和访问。Manager 组件中的 persistenceManager 类负责整个持久层的管理和协调工作,实现对象层与数据访问层之间的通信,其他对象通过 persistenceManager 对象实现交互。

4　应用

该对象持久层设计方案已经成功应用在北京一家服装服饰有限公司的 ERP 开发中,它为该系统的持久化服务提供了有力的支持。如果应用系统的开发人员需要实现雇员信息的存储或查询,那么应用该设计方案,通过下述语句即可实现。

```
//存储一个雇员的信息
Employee johnD = new
Employee("John Doe", 5000);
persistMngr.makePersistent(johnD);
//查询工资 5 000 元以上的雇员信息
Collection allEmps =
persistMngr.getExtent(Class.forName(
"Employee"));
Query q = new Query(
Class.forName("Employee",allEmps,"salary > 5000"));
Collection emps = q.execute();
```

由此看出,该方案只要求应用系统开发人员说明业务需要的持久化服务是什么,而不必说明系统如何做,使得业务逻辑与数据存储逻辑分开,从而提高了开发人员的工作效率,增强了软件的可再用性、可维护性、可修改性和可移植性。

5　结束语

本文依据基于软件体系结构的软件设计方法,提出了一种基于软件体系结构的对象持久层设计方案。该方案将体系结构作为对象持久层的整体视图,反映了对象持久层的整体结构和功能,并利用层模式实现了对象持久层的可再用性、可维护性、可移植性和可修改性等质量属性。本文将对象持久层体系结构构建为两层模式,即对象层和数据访问层,并为每一层均设计了组件。本文还根据对象持久层体系结构模型给出了对象持久层的类设计模型。这种基于软件体系结构的对象持久层设计方案,有利于 PAS 开发人员进一步地设计和开发工作。今后的工作要增加游标、对象代理等功能,不断对设计方案进行完善。

参考文献:

[1] 秦奕青,李健,等. 正交持久性准则及其应用研究[J]. 计算机工程与设计,2006,27(2):272-274.

[2] 秦奕青. 常用对象持久化方法研究[J]. 北京机械工业学院学报,2003,18(1):20-24.

[3] AMBLER S W. The design of a robust persistence layer for relational databases[EB/OL]. (2005-06-21). [2005-10-10]. http://www.Ambysoft.com/downloads/persistenceLayer.pdf.

[4] ALEXANDRER C. The timeless way of building[M]. Oxford: Oxford University Press,1979.

[5] SHAW M, GARLAN D. Software architecture: perspectives on an emerging discipline[M]. Upper Saadle River, NJ: Prentice Hall, 1996.

[6] KRUCHTEN P. Software design in a post modern era[J]. IEEE Software,2005,22(2):16-18.

[7] PARNAS D L. On the criteria to be used in decomposing systems into modules[J]. Communications of ACM,1972,15(12):1053-1058.

[8] NIEMELÄ E, KALAOJA J, LAGO P. Toward an architectural knowledge base for wireless service engineering[J]. IEEE Transactions on Software Engineering,2005,31(5):361-379.

[9] DIJKSTRA E W. The structure of the "THE" multiprogramming system[J]. Communications of the ACM,1968,11(5):341-346.

[10] MALEK S, MIKIC-RAKIC M, MEDVIDOVIC N. A style-aware architectural middleware for resource-constrained, distributed systems[J]. IEEE Transactions on Software Engineering,2005,31(3):256-272.

[11] SALCEDA J, DIAZ I, TOURINO J, *et al*. A middleware architecture for distributed systems management[J]. Journal of Parallel and Distributed Computing,2004(64):759-766.