第18卷 第4期 茂名学院学报 Vol.18 No.4
2008年8月 JOURNAL OF MAOMING UNIVERSITY Aug.2008

一种基于ORM理论的DAL设计与实现

孟亚辉[1],张党进[2]

(1.茂名学院 理学院,广东 茂名 525000;2.电信科学技术第十研究所,陕西 西安 710061)

摘要:DAL是以ORM为理论基础,在业务逻辑层和企业信息系统层之间进行数据类型相互转换和数据操作的一个单独模块。通过分析和研究ORM理论,并结合项目实际需求,提出了一种DAL设计方案。该方案采用类反射机制,实现了数据对象与表、存储过程的相互映射过程,并在开发过程中解决了一些具体问题,目前在应用系统中得到了很好的应用。

关键词:数据访问层;对象关系映射;类反射

中图分类号:TP311 **文献标识码**:A **文章编号**:1671-6590(2008)04-0053-04

在Web Services时代,企业级架构得到广泛应用,多层体系结构成为大型应用系统设计的主流。其一般分为三层,即:视图层、业务逻辑层和企业信息系统层。在视图层中,要求把页面和代码分开、数据与显示分开;在业务逻辑层中,要求不同业务模块之间相互独立,通过连接器有机组合起来,其中把各个独立模块中访问数据库操作分离出来,形成一个单独模块或者划分成一个层,即数据访问层(DAL,Database Access Layer)。在多层体系结构系统中,数据访问层是处在业务逻辑层和企业信息系统层之间的是一个数据通道。从数据类型上说,数据访问层把这两层不同存放数据格式进行相互转换,实现业务数据对象与数据库通信。

在J2EE和.NET[1]这两大主流Web Services架构中,有许多成熟的DAL解决方案,例如J2EE中有Hibernate[2]、Entity Bean、JDO等,在.NET中,有NHibernate、LINQ[3]等,都是以ORM为理论基础。通过映射机制,实现一个实例化对象与数据库表的一行记录对应,从而把对数据对象的创建、读取、更新和删除操作(CRUD,Create、Read、Update、Delete)转化为对数据库的插入、查询、修改和删除操作(ISUD,Insert、Select、Update、Delete),实现了把业务逻辑层抽象的持久性数据模式映射到数据库中的物理数据模式。

1 ORM理论基础

DAL要求把业务逻辑层的数据对象结构与数据库表二维结构进行相互转换,这种异构体转换正是ORM理论研究内容,所以DAL设计者选择以ORM为理论基础。ORM分为两个部分,一部分是对象映射,对象映射包括实体映射和关系映射;另一部分是数据对象管理,主要是对在缓存中数据对象状态进行管理。

在业务逻辑层中,数据是以对象形式存放在内存中,即以数据对象为最小单元;而在数据库中,数据是永久性存放在二维库表文件中的。这两种数据类型各有特点,在数据对象设计中,要求根据业务定义数据对象属性,类型以及数据对象之间的组合、继承等关系;在数据库表设计中,要求根据业务需求,定义表字段、类型以及表之间的父子关系等。其相同点是,一个类有对象名,表也有表名;对象有属性,其中包括名称和类型,表有字段,其中包括字段名称和类型。这些相同点使得一个实例化对象和一张表的一行记录通过对象的OID和表的Identity关联起来成为ORM理论实现的基石。其中表名和类名、表字段名称和类属性名称、表字段类型和类属性类型的对应关系(即实体映射)以及表之间的关联关系对应到对象之间的组合关系的映射描述(即关系映射),这是ORM理论所在。其实ORM就是一个“桥梁”。通过映射机制,解决

收稿日期:2008-04-07;**修回日期**:2008-05-18

作者简介:孟亚辉(1978—),男,陕西杨陵人,硕士(在读),助理工程师,从事计算机系统研究与软件开发。

了两种异构体的相互转换问题,从而使得在实现过程中,数据与代码进行有效分离。

为了提高数据检索率,ORM 提出了建立缓存机制,管理数据对象在缓存中的三种状态(即:临时状态、持久化状态和游离状态)之间相互转化并保持与数据库同步等操作。

2 DAL 体系结构设计

2.1 DAL 设计原则

DAL 从三层结构中分离出来,封装了一切访问数据库相关操作,给业务逻辑层提供数据对象持久化操作,减轻了业务逻辑层处理数据而访问数据库操作的负荷。DAL 设计时,首先要求其与业务无关性,这样才能适应不同业务对数据库访问操作的请求;其次,DAL 所访问的数据库类型对业务逻辑层来说,是完全透明的,这样系统实施时有更多的数据库选择,更大程度地满足用户的不同需求。

ORM 核心是针对数据对象和数据库表之间的映射,但是对于存储过程来说,相对比较复杂,另外存储过程类型比较多,所以在设计 DAL 设计时,要求完全能够适应不同类型的存储过程调用。

在 DAL 实现过程中,要求数据跟具体代码分离,即 CRUD 接口实现类参数化。

2.2 组件结构图

图 1 描述了 12 个组件模块之间的关系,其中映射信息和数据库配置信息加载到系统缓存中,提供映射信息和数据库配置信息高速查询。[4]关于功能模块的说明见表 1。

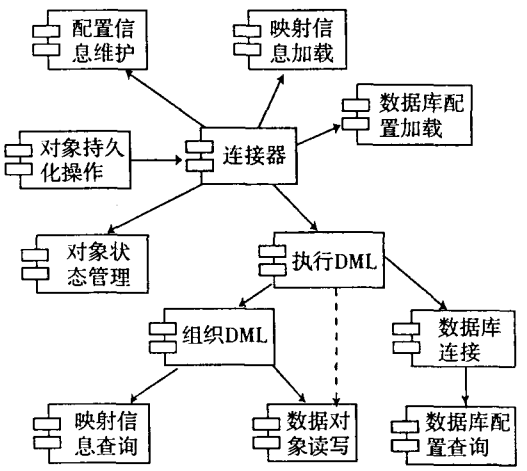

图 1 组件及其关系图

表 1 功能描述

模块名称	功能
配置信息维护	根据数据库表结构生成对应数据对象类文件和映射信息文件
数据库配置装载	装载数据库配置信息,包括数据库类型,连接等配置信息
数据库配置查询	根据对象所对应的表所在库名,查找数据库配置信息
映射信息装载	读取所有数据库表映射信息文件,放在缓存中
映射信息查询	通过对象名称查找数据库表名、字段及其关联等信息
数据对象读写	采用类反射机制,对数据对象中数据进行读写
组织 DML	根据操作类型和具体的数据对象,动态组织 DML
执行 DML	向数据库连接池申请一个连接,执行 DML
数据库连接	根据数据库配置信息,打开和关闭数据库连接等操作。
对象持久化操作	提供数据对象的 CRUD 操作
对象状态管理	对数据库对象三种状态进行有效管理
连接器	1~10 组件进行 Run Time 的连接管理

2.3 DAL 流程

图 1 中,连接器[5]把 10 个组件有机组合起来,按照其流程处理,分为两个分支。其一是根据操作请求,首先到缓存中检索数据对象,处置完之后返回,否则,通过映射信息到数据库中查找。处理流程如图 2 所示。

组织 DML 时,首先根据操作类型,组织 DML 首部(例如 SQL 语句中的 select、update 等),然后从数据对象中读取数据名称,通过映射信息,获取 DML 中间部分,最后从数据对象中读取数值,定义 DBParameter 并加载到 DBCommand 中。图 2 第 17 节点中,对执行 DML 结果进行处理。如果操作成功,数据库返回成功处理库表行数或者是结果集,把这些信息装载到对应数据对象返回;否则,把处理异常信息装载到数据对象返回。

3 DAL 实现

3.1 绑定数据对象与表的映射信息

如图 2 所示,根据操作类型和数据对象组织 DML 时,要求从对象名称映射到库名及其表名称,对象属性信息映射到表字段信息(包括字段名称、长度、类型及其是否可为空等信息),即数据对象要求与其对应表信息进行动态绑定,图 3 中基类 ValueObject 绑定了数据对象与表的映射信息。其中 mapMsg 是数据对象

与表映射信息集合类。映射信息装载到内存,数据对象实例化时,mapMsg就关联到该数据对象与表的映射信息(其实mapMsg是所有对象与表映射信息集合类中的一个对象引用)。

3.2 访问不同数据库类型

由于DAL所访问的数据库类型对业务逻辑层是完全透明的,所以DAL在实现过程中要考虑不同数据库在连接方式和基本数据类型上的差异。数据库配置中描述了数据库类型,故通过数据对象的GetDBProvider方法可以获取数据库具体类型信息,并组织调用对应数据库连接。

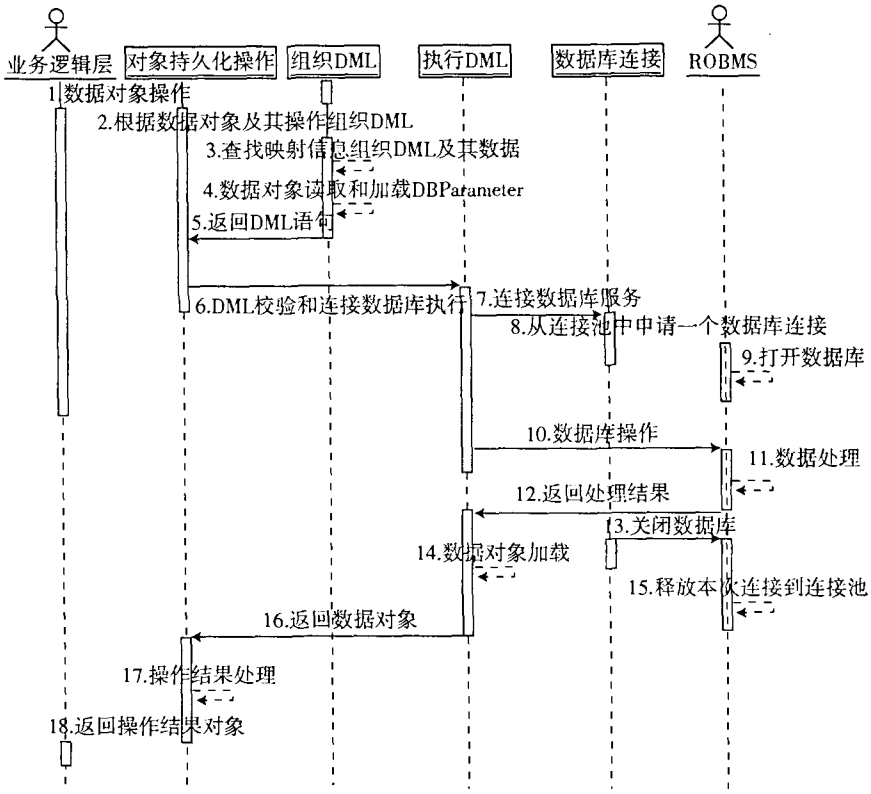

图2　DAL时序图

不同数据库类型的基本数据类型有所不同,所以在组织DML过程中,通过数据对象的attributeName来获取对应表字段的类型时,返回所要访问库对应的数据类型。

3.3 存储过程处理

ORM核心是针对数据对象和数据库表之间的映射;也在于表结构相对比较简单。从函数角度来看,表输入和输出完全相同,即这种映射是双向的,可以通过单一的映射过程实现库表操作,但是对存储过程而言,输入参数和输出参数却不一定相同,所以其输入和输出两个映射过程不能重叠。在实现存储过程时,定义了两个数据对象和两份映射信息。参数输入与一般表操作映射过程相同;而在装载返回的数据时,根据结果集中参数信息来"反"映射,把数据装载到数据对象中,这一点与表映射有所不同。

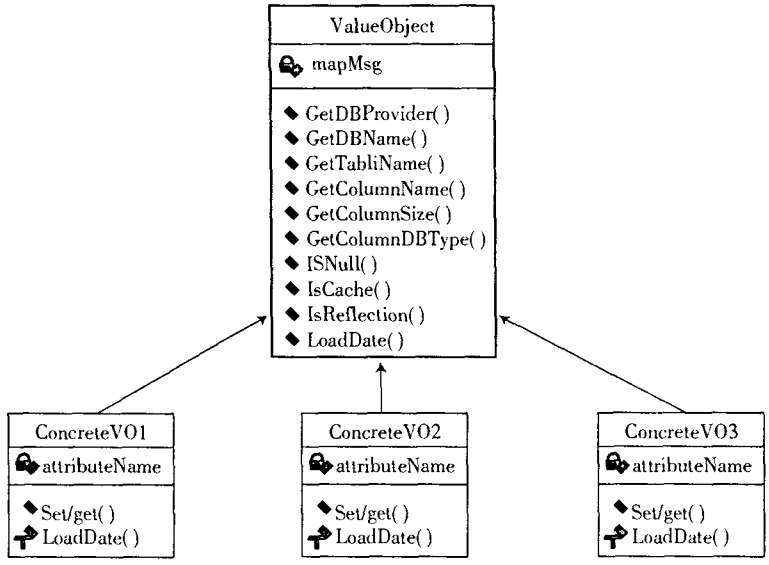

图3　数据对象类图

一般,实现结果集映射到数据对象有两种方法。一种是通过遍历数据对象属性名映射表字段名称来获取结果值,这种方式比较适合表结果集处理,因为在组织查询的DML时,遍历数据对象名称来映射表字段名称,组成所要查询的字段名称顺序,这个顺序与结果集读取完全一致,所以比较适合表查询;另一种方式就是遍历结果集中每一个字段,通过字段名称逆向映射对象的属性名称,把数据加载到数据对象中,这种方式适应于存储过程返回结果集处理,因为存储过程返回集合的映射信息与输入参数的映射信息具有不对称性。也就是说,存储过程返回结果集是根据业务需求,组织返回不同的列。

另外,存储过程参数中可以包含输出参数,返回结果集有多个或者没有,这样在存储过程的映射信息

中必须进行描述,因为执行存储过程操作与其具体的输出参数、返回结果集及其返回类型有关。

3.4 唯一性映射

每个数据对象实例通过自身的OID标识唯一性,每一行数据库记录通过主键标识唯一性。那么在实现数据对象实例与表记录关联时,必须在两种异构体中实现唯一性映射。

为了减少映射复杂度,一般采取数据对象的GetHashCode与表的主键关联。对于表而言,提倡使用代理主键,因为代理主键没有业务逻辑含义,从而不会因业务变化而被修改。

3.5 批量查询操作

如图2所示,在组织DML时从数据对象中读取数据,和在执行之后把返回结果集装载到数据对象中,都需要涉及到数据对象读写。在DAL实现中,采用类反射机制,根据属性名称从对象实例中读写数据。

通过把对象参数化,实现了动态组织DML和装载执行DML返回结果集到数据对象中。但是相对于静态读写数据对象数据方式而言,性能比较低。实验表明,两者在批量数据装载时,消耗时间为3:1,原因在于前者需要遍历实例化对象的所有属性,找到要操作的属性信息,调用其SetValue和GetValue方法同静态语句执行性能相同。所以对批量数据对象操作,不宜采用类反射机制来实现数据对象读写。

在DAL设计中,对每一个数据对象操作有两种数据读写方式。一是调用数据对象读写模块,动态语句方式加载数据;另一种方式,调用LoadData静态语句方法方式加载数据。在数据对象读写时,根据IsReflection判断使用那种方式来加载数据(见图3)。其次,在批量对象操作中,数据对象不能进入缓存。通过IsCache判断,批量操作的数据对象能否被写到缓存中。如果批量操作对象放到缓存中,其一容易导致缓存溢出;其次减少了其他操作数据对象在缓存中的命中率,即批量操作时,关闭其缓存写操作。当然在对其他相关操作影响不大的情况下,也可以选择定时清理缓存来提高执行效率。

4 结论

在DAL设计和实现中,有许多种理论作为指导和成熟解决方案,但它们各有优缺点及适用条件。本文以ORM为理论指导,在DAL设计时,以与业务逻辑无关性、数据库类型完全透明为原则,采用类反射机制,把数据与代码进行分离,从而实现了对象映射和对象管理两个核心部分。

随着数据库技术日益成熟,许多数据库设计模式的应用实践,很多数据库设计者趋向于用存储过程来替代复杂的业务逻辑层处理业务,这要求DAL设计要适应各种存储过程情况,而在一些成熟DAL解决方案中,只是简单地进行处理。本文在分析存储过程与表结构的差异基础上,把ORM"本地化",设计了一种适应项目实际需求的存储过程解决方案,并在项目开发和维护中使之趋于完善。

[参考文献]

[1] 甑镭.NET与设计模式[M].北京:电子工业出版社,2005.
[2] KING G. Hibernate In Action CHRISTIAN BAUER[M].[S.l.]:Manning Publications Co.,2005.
[3] EICHERT S,WOOLEY J.LINQ In Action FABRICE MARGUERIE[M].[S.l.]:Manning Publications Co.,2008.
[4] Bass L.软件架构实践[M]. 北京:清华大学出版社,2003.
[5] Erl T. SOA概念、技术与设计[M].王满红,陈荣华,译.北京:机械工业出版社, 2007.

Design and Realization of DAL on the basis of ORM Theory

MENG Ya - hui[1], ZHANG Dang - jin[2]

(1. College of Science, Maoming University, Maoming 525000, China;
2. The Tenth Research Institute of Telecommunications and Science Technology, Xi'an 710061, China)

Abstract: On the basis of the ORM theory, DAL can realize the data type conversion and operation between the business logical layer and the enterprise message layer. One design of DAL is produced through analyzing and studying the ORM theory, according to real requirements in specific project. This design not only adopts class reflection, thus realizing the mapping among data object, table, and stored procedure, but also gives successful solutions to some specific problems. At present, DAL is functioning well.

Key words: database access layer; object relational mapping; class reflection