文章编号:1674-0076(2008)03-0055-05

轻量级ORM持久层的研究与实现

肖光星,钟海云,官振兴,张正球
(福建师范大学 软件学院,福建 福州 350007)

摘 要:基于Hibernate、iBATIS、JPOX JDO等开源持久层的分析与研究,结合Hibernate的易用性和iBATIS开发的高自由度等特点,提出了一个三模块关系型数据库持久层的轻量级架构与实现方案,该架构能够有效缓和数据库资源管理与对象加载所产生的性能瓶颈.

关键词:持久层;ORM;关系数据库;对象关系映射
中图分类号:TP311.13 **文献标识码**:A

Research and implementation of a lightweight ORM persistence layer

XIAO Guang-Xing, ZHONG Hai-Yun, GUAN Zhen-xing, ZHANG Zheng-qiu
(Faculty of software, Fujian Normal University, Fuzhou 350027, China)

Abstract: Based on the analysis and research of the open source persistence layers such as Hibernate、iBATIS、JPOX JDO, with the user-friendliness of Hibernate and the unlimited exploitation space of iBATIS, this paper designs a three-component-based lightweight persistence layer for relation database. The persistence layer helps to solve the problem of performance bottle-neck in database resource management and object construction.

Key words: persistence layer; ORM; relation database; object relation mapping

0 引 言

持久层是为了在与关系数据库交互过程中能够继承面向对象程序设计思想而提出的数据库访问层的解决方案.其作用是在提供与数据库交互的同时方便用户将对象从内存持久化到关系数据库或XML文档等存储载体上.持久层的设计可以组件化,进而进行功能定制[1].

当前已有一些开源持久层,如Hibernate、iBATIS,JPOX JDO等,这些都较适用于中小规模软件开发的轻量级持久层,且能使开发人员可在与关系数据库交互过程中延续面向对象的思想,但在一些性能表现、开发空间自由度等方面上还存在着许多差异.文献[2]对Hibernate,JDBC,iBATIS,JPOX JDO,Castor各开源持久层做了包括性能、易用性、开发所需代码量等方面的系统比较.Hibernate是JDBC的轻量级的对象封装,它是一个独立的对象持久层框架[3],基于Hibernate的开发很容易带来局限性,而iBATIS在一定程度上克服了这种的局限性,但它又不具有Hibernate操作的简易性.

通过对Hibernate和iBATIS的研究与分析,发现可将Hibernate的易用性与iBATIS的高开发空间自由度结合在一起.因此本文结合Hibernate的易用性和iBATIS的高开发自由度等特点,提出了一个结构清晰的三模块关系数据库持久层的轻量级架构,并且也能有效缓和数据库资源管理与对象加载所带来的性能瓶颈.

1 轻量级关系型数据库持久层体系构架

基于对一些开源持久层的分析与研究,结合文献[4]对于鲁棒的持久层的功能描述,归纳出满足中小型企业软件开发的关系型数据库持久层应包括以下基本功能:(1)对象持久化操作;(2)基于关联的多对象持久化操作;(3)支持面向对象的事务处理;(4)支持代理;(5)必要时能返回记录形式的结果集;(6)支持不同数据

收稿日期:2008-03-18
基金项目:福建省教育厅A类项目(0311036)
作者简介:肖光星(1986-),男,本科生.

库版本和厂商;(7)允许调用存储过程;(8)支持游标.

本文依此把关系型数据库的持久层分成三模块:映射信息提供模块、数据库资源管理模块、用户接口模块.三模块的整体关系如图 1.

映射信息提供模块为用户接口模块和数据库资源管理模块提供支持对象关系映射所需的数据映射信息和数据库资源配置信息.数据库资源管理模块负责数据库连接、SQL 语句句柄等数据库资源的优化管理.用户接口模块是逻辑层开发人员直接可见的持久层 API,用户(即逻辑层开发人员)通过调用该模块中的各编程接口对普通 JAVA 数据对象进行增、删、查、改等操作.

以下将对三模块的内部结构与实现方案分别进行详细介绍,具体的性能优化方案将在"持久层性能优化"中介绍.

1)映射信息提供模块

映射信息模块主要是为其余两个模块提供用户的配置信息.配置信息使用具有良好扩展性的 XML 文件作为外存载体.为了不频繁访问读写速度较慢的外存设备,在持久层第一次被实例化过程中,将所有配置信息加载并用 HashMap 组织缓存这些信息于内存中.

映射信息模块的上下文关系如图 2 所示.

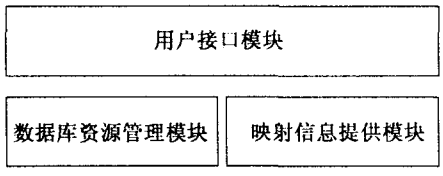

图 1 三模块体系持久层架构

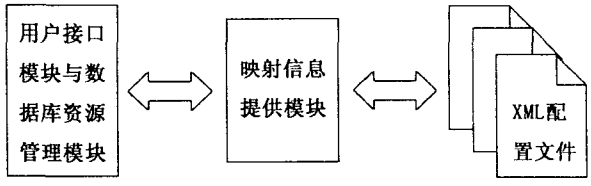

图 2 映射信息模块的上下文关系

配置信息主要包括数据库资源配置信息、对象关系映射信息以及为扩展开发空间而定义的 SQL 语句.这些信息的部署允许灵活变动.本文提出的持久层方案将这些配置信息部署在 4 个配置文件中,分别为持久层基本信息配置文件(主要定义其它配置文件路径)、数据库资源池配置文件、对象关系映射信息配置文件和 SQL 语句定义文件.例如,持久层基本信息配置文件的 DTD 定义如下:

```
<! ELEMENT database - driver ( # PCDATA) >
<! ELEMENT database - url ( # PCDATA) >
<! ELEMENT database - user ( # PCDATA) >
<! ELEMENT database - password ( # PCDATA) >
<! ELEMENT resourcePooling _ cfg _ file _ path ( # PCDATA) >
<! ELEMENT sql-statements _ file _ path ( # PCDATA) >
<! ELEMENT mapping _ cfg _ file _ path ( # PCDATA) >
```

数据库资源池配置文件用于配置具体的资源池,根据用户选择不同的资源池而相应的配置文件也不同.关系映射信息配置文件主要是配置 JAVA 普通类与数据库表的对应情况,与大多数开源持久层的设计是一样的.SQL 语句的定义在用户接口模块中有说明.

2)数据库资源管理模块

数据库资源是持久层最频繁使用的资源之一,持久化的每一个动作都涉及数据库连接、PreparedStatement 等数据库资源的使用.使用资源池模式管理这些频繁使用的数据库资源,可以避免频繁使用的资源重复创建所带来的性能消耗.

数据库资源管理模块的内部结构如图 3.

此模块采用分层结构设计,从下到上依次为数据库驱动层、资源池层和接口层.资源池层处于中间层,它包括一个资源池工厂和多个可独立的资源池模块.该层中使用工厂模式[5]集成多个资源池模块,可以提供不同需求环境下的理想选择.另外资源池配置的不同以及连接的目标数据库的不同,所需的数据库驱动也是不一样的,所以独立出数据库驱动层,允许用户根据需求配置使用不同的数据库驱动.接口层是对用户接口模块提供服务的接口,主要提供连接建立、Prepared Statement 的获取以及 SQL 语句执行等服务.

虽然很多 JDBC 驱动本身就支持连接池与语句池,但为了使得整个持久层易于迁移而不依赖于特定的驱动,本文选择使用独立于数据库驱动的连接池与语句池(即资源池层中的资源池模块).当前开源社区已经有许多性能良好的开源连接池,例如 c3p0,proxool,dbcp 等,而且有些连接池带有语句池功能,可以封装成可独立的资源池模块.直接使用这些开源资源池可以在缩短开发周期的同时得到最高的整体效益.

3)用户接口模块

用户接口模块提供普通 JAVA 对象的持久化操作,并支持事务、存储过程的执行以及类似 iBatis 的 SqlMap 的 XML 参数化 SQL 语句的定义、执行和对象持久化操作.

用户接口模块内部结构抽象如图 4.图中对象的持久化操作是指在应用程序软件中直接持久化和反持久化普通 JAVA 数据对象.对象持久化操作依赖于对象数据与关系数据映射模块和 SQL 语句(PreparedStatement 对象或 Sql String 实例)生成器模块.每次对象的持久化操作都需根据具体的操作翻译成相应的 PreparedStatement 或 Sql String 通过数据库资源管理模块接口与数据库交互.

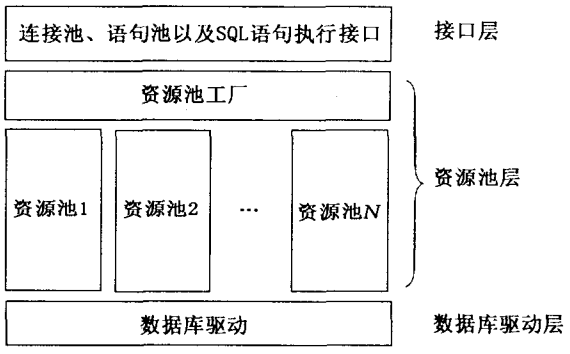

图 3　数据库资源管理模块内部结构

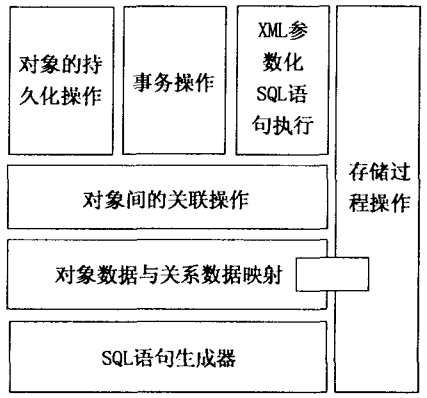

图 4　用户接口模块内部结构

持久层提供的事务操作是面向对象的事务操作.由于 JDBC API 的特性,面向对象的事务操作模块可以直接利用对象的持久化操作模块完成.因为使用同一个非自动提交事务的连接(即该数据库连接的 autoCommit 属性为 false)提交事务的逐个过程可以在内存中累积操作过程,待用户下发 Commit 提交命令后才把积累在这一连接上的操作过程全部当作一个事务来处理.

XML 参数化 SQL 语句是指定义在 XML 文件中的有参 SQL 语句.持久层对 XML 参数化 SQL 语句执行的支持可以大大增加逻辑层开发的自由度.XML 文件中的有参 SQL 语句示例如下:

< SQL id = "sqlStringID" parameterClass = "int" resultClass = "package.subPackage.Tuser" > SELECT * FROM t _ user WHERE id = # id # < /SQL >

它可以选择地定义该 SQL 语句的参数类型(如示例的 parameterClass = "int")、参数名称(如示例的 # id #)及 SQL 语句执行后的返回值类型(如示例的 resultClass = "package.subPackage.Tuser"),但 SQL 语句字符串(如示例的 SELECT * FROM t _ user WHERE id = # id #)是必不可少的.在 SQL 语句生成器中,所有 XML 参数化 SQL 语句都将被翻译成 PreparedStatement 对象,这些 PreparedStatement 对象在设置完参数后(如果语句中有定义参数),由数据库资源管理模块将其送到数据库被执行.返回的 ResultSet 将使用对象数据与关系数据映射子模块进行封装,每条记录生成一个由 XML 参数化 SQL 语句定义时指明的 resultClass 对象,最后以 resultClass 对象或其对象链表的形式返回.而用户要调用这些定义在 XML 文件里的参数化 SQL 语句时只要通过对应 SQL 语句的 ID 来索引.

对于以上所述的 3 个用户接口子模块:对象持久化操作子模块、事务操作子模块、XML 参数化 SQL 语句执行子模块,它们在操作被定义为有相互关联的对象时都会被对象间的关联操作子模块截获,由对象间的关联操作子模块进行允许延时加载的级联操作.

对象数据与关系数据映射这一子模块主要利用反射技术和映射信息,将面向对象的数据操作翻译成面向关系型数据库的数据操作,并为加载得到的类赋予唯一 ID 标识.但这一子模块所生成的操作信息(如要操作的表及字段等)并不能被关系型数据库直接执行,它将通过 SQL 语句生成器生成相应的 SQL 语句或调用

已经准备好的 Prepared Statement 进行对应的持久化操作.

存储过程操作子模块提供对数据库存储过程操作的支持.这样可以使得一些复杂事务操作流程可以在数据库中事先编制好,并且存储过程的一些执行结果可以调用对象数据与关系数据子模块进行对象封装,以对象形式的结果集返回.

在用户接口模块中,所有结果以面向对象形式返回的查询方法,都需重载,使得相同的查询结果也可以以记录的形式返回.这样的重载方法实现时只需跳过对象封装这一步骤,直接将 ResultSet 返回即可.

2 持久层性能优化

以上的三模块体系关系型数据库持久层架构是较抽象的体系结构,在这一体系结构上可以做进一步的性能优化以求得持久层的稳定性能.上面的设计实现中已经提到了使用资源池缓和数据库资源大批量重复创建所带来的性能消耗,下面将介绍具体的持久层性能优化.

1)使用连接池优化数据库连接

在一台 Mobile AMD Sempron Processor 3000 + 1.79GHz CPU,768M RAM 的机器上,使用 JDBC3.0 API 和 mysql-connector-java-5.1.5 驱动,不使用连接池的情况无间隔地连续 1 000 次连接 mySql5.1.22 community 数据库时的平均每次连接的耗时是 260 ms;使用连接池,不计连接池的初始化时间,同样无间隔地连续 1 000 次连接相同的数据库时的平均每次连接耗时仅为 4 ms.很显然,在数据库连接管理上,持久层应该配置使用连接池.

在图 2 中反应这一优化的设计是资源池的使用.在持久层中使用资源池统一管理数据库连接,以降低上层应用频繁连接数据库时创建连接的平均开销.

2)使用语句池优化 SQL 语句的执行

一条 SQL 语句传入数据库管理系统(DBMS)后,DBMS 要对这条 SQL 语句做一系列的预处理,例如语法检查、语义分析、执行计划、性能优化等,这些预处理都存在性能消耗.在持久层中应用 JDBC3.0 的新特性 Prepared Statement Pooling,将预处理后的 SQL 语句对象(Prepared Statement Object)缓存,除第一次执行之外,其它时间的执行就可以省去这些 SQL 语句预处理过程,从而提升持久层性能.本文的实践证明了合理地使用语句池,可以使得 SQL 语句执行的性能得到显著提升.

语句池集成在数据库资源管理模块中,即图 2 所示的数据库资源管理模块的中间层,资源池层.资源池层统一管理 Prepared Statement,当接口层得到 SQL 语句执行调用时,首先在语句池中搜索该条 SQL 语句是否已经被准备,如果存在,就直接返回相应 Prepared Statement 实例,如果不存在,则创建相应的 Prepared Statemnt 并加入语句池.

3)使用缓存机制优化对象加载过程

在用户接口模块的对象数据与关系数据映射子模块中,一些数据操作需要将关系型数据封装成普通 JAVA 数据对象,这样的操作通常被称为对象加载.对象加载是利用反射技术实现的对象的动态加载.而反射是一种大开销的操作.反射是对象关系映射的性能瓶颈之一.因此,持久层的对象加载过程应该充分利用已加载过的数据对象,避免重复加载.用户接口模块的对象数据与关系数据映射子模块的中应该设计缓存来存储那些已被加载的数据对象,这样,当下一次再次需要这些数据对象时,只要这些数据不成为脏数据,就不必从数据库中读出数据集然后再反射形成对象了,只要从缓存中取出数据对象当作查询结果返回便可.这样就省去了数据库查询和对象反射封装所带来的开销.

缓存的实现也可以使用开源软件,例如 ehcache、oxcache 等,这些缓存框架都支持硬盘缓存,都有丰富的开发文档与测试用例,使用也很容易.

3 持久层性能分析比较

在 Mobile AMD Sempron Processor 3000 + 1.79GHz CPU,768M RAM 机器 + Windows XP + JDK1.5 + Eclipse3.3.1.1 + MySQL5.1 的运行环境下,对本文设计的持久层与 Hibernate3.2,iBATIS2.3 做了简单的性能测试比较(person 类是只有 int 类型的 id 和 String 类型的 name 2 个字段的简单类,与其对应的数据库表也是只有 int 类型的 id 和 varchar(20)类型的 name 2 个字段的表),测试结果如表 1.

表 1　测试比较结果　　ms

访问层	在数据库中创建 100 个 Person 类	从数据库中读取 100 个 Person 类	更新数据库中的 100 个 Person 类	删除数据库中的 100 个 Person 类
Hibernate3.2	3 469	250	3 467	3 133
iBATIS2.3	3 687	271	3 038	3 265
本文设计的持久层	3 598	254	3 089	2 879

从结果数据中可看出该持久层的性能已达到甚至在某些地方优于 Hibernate 和 iBATIS 等优秀开源持久层的水平.同时该持久层集成了 Hibernate 的易用性与 iBATIS 的高开发空间自由度的优点，即可直接使用面向对象方法操作关系型数据，又可以在开发中自由编写复杂 SQL 语句，缓和用单纯面向对象方法操作关系型数据所带来的束缚感.

4　结束语

本文通过对 Hibernate、iBATIS、JPOX JDO 等开源持久层的研究与分析，结合 Hibernate 的易用性和 iBatis 开发的高自由度等特点，提出了一个结构清晰的三模块关系型数据库持久层的轻量级架构.它在中小型分层软件开发中既能得到完全的面向对象思想，又给开发人员足够的自由空间.本文设计实现的持久层在实践中已经证明是可行的方案，但这个持久层不支持分布式事务处理，因此这样的持久层只适用于非分布式的中小型系统当中，以后将在对分布式事务的支持上做进一步的研究.

参考文献：

[1] 曾　一，徐　珂，李　颖.基于组件架构的对象持久层框架设计[J].计算机科学，2005，32(12)：120－124.

[2] Jim White.Sizing up open source Java persistence [EB/OL].(2007-2-15)[2007-2-20].http://www.devx.com/ Java/Article/33768.

[3] 林寒超，张南平.Hibernate 技术的研究[J].计算机技术与发展，2006，16(11)：112 － 116.

[4] Scott W.Ambler.The design of a robust persistence layer for relational databases [EB/OL].[2007-2-12].http://www.ambysoft.com/downloads/persistenceLayer.pdf.

[5] 易　燕，周　骋，戴祝英.运用设计模式实现数据持久层框架[J].计算机工程与设计，2005，26(12)：3365－3371.

(上接第 11 页)

3　结束语

本文定理 1 从局部凸空间到局部 G－凸空间推广了文献[1]的定理 1，并给出定理 1 的两个应用，即定理 2 和定理 3.同时定理 1、定理 2、定理 3 也从局部凸空间的可允许集到局部 G－凸空间推广了文献[2]的定理 1、定理 2、定理 3，并且定理的条件完全不一样.

参考文献：

[1] Fu jingyi.Generalized quasi-vector variational inequalities in ordered vector spaces [J].Applied Functional Analysis，1997，3：21－25.

[2] 江慎铭，曹寒问，邹　群.可允许集上的广义向量拟均衡问题[J].数学的实践与认识，2006，36(8)：33－335.

[3] Watson P J.Coincidences and fixed points in locally G-convex spaces [J].Bull.Austral.Math.Soc，1999，(59)：297－304.

[4] 江慎铭，傅俊义.(伪)单调集值映射拟均衡问题[J].数学的实践与认识，2006，36(5)：229－234.

[5] Lee G M，Lee B S，Sen S.Chang.On vertor quasivarational inequalities[J].Math Anal Appl，1996，203：626－638.