

文章编号:1001-9081(2005)10-2325-03

## 基于 J2EE 的客运信息管理系统数据持久层的 Hibernate 解决方案

李 敏<sup>1</sup>, 黄 强<sup>1</sup>, 李 昊<sup>2</sup>, 尹治本<sup>1</sup>

(1. 西南交通大学 信息科学与技术学院, 四川 成都 610031;

2. 西南交通大学 交通运输学院, 四川 成都 610031)

(lmtutu@163.com)

**摘 要:**以成都市 ITS 建设为背景,主要讨论了其中客运信息管理系统数据持久层的设计,通过对目前主流的持久层解决方法的比较提出了基于 J2EE 的 Hibernate 解决方案,并对该方案的优势进行了说明。

**关键词:**客运信息管理系统;ORM;Hibernate;Middlegen;Hibernate extension;持久化

**中图分类号:** TP311 **文献标识码:** A

### Hibernate persistence solution for passenger transport information management system based on J2EE

LI Min<sup>1</sup>, HUANG Qiang<sup>1</sup>, LI Hao<sup>2</sup>, YIN Zhi-ben<sup>1</sup>

(1. School of Information Science and Technology, Southwest Jiaotong University, Chengdu Sichuan 610031, China;

2. Department of Traffic and Transportation, Southwest Jiaotong University, Chengdu Sichuan 610031, China)

**Abstract:** Based on the construction of ITS in Chengdu, the design of data persistence layer of the passenger transport information management system was discussed in this paper. By comparing the main solutions for data persistence layer currently, the Hibernate solution on J2EE was put forward, and the advantages of the Hibernate solution were described.

**Key words:** passenger transport information management system; ORM; Hibernate; Middlegen; Hibernate extension; persistence

## 0 引言

在基于 J2EE 的客运信息管理系统中,数据持久层负责存储从应用到数据库的数据,也负责数据的检索、更新和删除。目前,我们使用的绝大多数数据库是关系型数据库,而设计和开发往往又是面向对象的方式,为了提高数据访问的效率,持久层的解决方案是整个系统的关键。对持久层的实现存在多种方案,例如 JDBC、EJB、JDO、ODBMS 以及 ORM 等。在这些方案中用 JDBC 运行效率最高,但 DAO 对象和 SQL 语言耦合太过紧密;EJB 功能强大,但使用复杂且不够灵活;JDO 没有一个开源的代码而且不是一个轻量级的封装,没有统一的标准而且产品存在分裂的问题;ODBMS 是一种有发展前途的技术,但目前都还处在不成熟阶段;ORM 是一种较为理想的解决方案。ORM 工具在 Java 对象与数据库表之间建立映射关系,具有自我存储到关系数据库的能力,对对象的改变能够直接存储到数据库而不用数据库存取的代码,这样就能形成相对独立的对象持久层,从而降低 J2EE 应用与数据库耦合度并简化程序开发。目前存在许多 ORM 工具,比如:TopLink、CocoBase、Hibernate、Castor、Torque 等<sup>[5]</sup>。在这些工具中 Hibernate 框架完全是为了满足开发人员的需要产生的,重要的是 Hibernate 能够创建 DBA 容易接受的 SQL 语句,因此本文提出的客运信息管理系统采用 Hibernate 进行系统持

久层的设计。

## 1 Hibernate 框架结构

在应用 Hibernate 框架时,首先编写 O/R 映射描述文件,完成对象、关系数据库之间的映射。持久对象可以根据映射文件生成。然后编写业务逻辑类。这些 JavaBean 实现了具体的业务逻辑,也封装了对 Hibernate 的访问。基于 Hibernate 的应用系统总体框架如图 1。

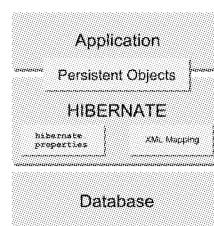


图 1 系统总体框架

如图 1 所示,整个系统主要有三层:应用层(Application)、基于 Hibernate 的数据持久层(Hibernate)、数据库层(Database)<sup>[5]</sup>。在应用层主要是对对象进行操作;在数据库层主要针对的是关系型数据表。而在对象范例和关系范例存在着“阻抗不匹配”,使对象与关系表间的直接数据操作存在一定的困难。在基于 Hibernate 的应用系统中“阻抗不匹配”问题就由中间的 Hibernate 层的 O/R Mapping 来解决,Hibernate 在对象范例和关系范例中建立映射关系(O/R Mapping),将应用层中对对象的操作直接作用于关系数据库中的表,使程序员不用再去关心数据库的操作问题。Hibernate 提供了好几种

收稿日期:2005-04-26;修订日期:2005-07-06 基金项目:交通部交通信息化示范工程项目(200552)

作者简介:李敏(1981-),女,四川绵阳人,硕士研究生,主要研究方向:数据库、软件工程、计算机网络与信息系统;黄强(1981-),男,四川成都人,硕士研究生,主要研究方向:软件工程、虚拟机、人工智能、神经网络;李昊(1977-),男,博士研究生,主要研究方向:智能交通、交通控制;尹治本(1964-),男,云南人,教授,主要研究方向:软件工程、网络信息系统、决策支持系统、算法设计。

不同的运行方式,其中比较极端的有两种:轻型体系和全面解决体系。在轻型体系中,应用程序自己提供 JDBC 连接,并且自行管理事务,这种方式使用了 Hibernate API 的一个最小子集;在全面解决体系中,对应用程序来说,所有的底层 JDBC/JTA API 都被抽象了,由 Hibernate 管理所有的细节<sup>[1]</sup>。

## 2 第三方工具简介

对客运信息管理系统这样一个大的管理系统而言,所涉及的数据库表数量较大,而且随着时间的推移对数据库表的改动也较大。在这种情况下,如果所有的对象与表间的映射文件与和表相对应的 Java 对象的建立都用手工来完成将是一个很繁琐的过程,而且开发的效率也不高,在系统升级的过程中所要做的工作也是一个很耗时的过程,所以我们在对客运信息系统持久层的设计中用到了对 Hibernate 提供支持的第三方工具 Middlegen 和 Hibernate Extension。Middlegen 用来从现有的数据库 schema 自动生成对应的映射文件,并且在 Middlegen 所提供的界面中可修改数据库表之间的关系、数据类型等属性,所生成的映射文件基本可以直接使用,只是在有特殊需要的时候要稍微做一些修改;Hibernate Extension 可以为映射文件自动生成 POJO(Plain Old Java Object),这样根据 Middlegen 提供的映射文件就能生成对应的可运行的 Java 类的定义。通过这两个工具的支持使得从关系型数据库表到 Java 对象的转换过程得到极大的简化<sup>[1]</sup>。

## 3 基于 Hibernate 的客运信息管理系统

下面以客运信息管理系统为例来说明如何采用 Hibernate 进行数据持久层的设计。

考虑到上面所提到的客运信息管理系统所要提供的三个方面的服务,我们得出系统的功能结构如图 2 所示。

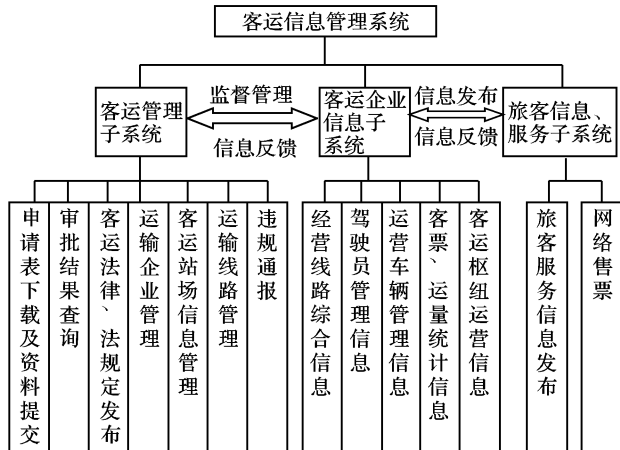


图2 客运信息管理系统开发功能图

由图2所示的客信息管理系统由客运管理、客运企业信息、旅客信息服务三个子系统组成,在三个子系统所对应的功能模块中,运营车辆管理信息是一个基本的功能模块。下面我们就以该功能模块的车辆信息的录入为例来说明本系统对数据持久层的设计。该例中包含了三个实体对象 Driver(司机基本信息)、Car(车辆基本信息)、Statistic(营运信息),在这三个实体中 Driver 与 Car 是一个多对多的联系,两个实体通过 Statistic 实体建立联系(该例中只涉及实体的部份属性)。

我们设计数据持久层从已有的数据库表 schema 开始,按下面步骤进行:

1) 由 Middlegen 导出数据库表结构,生成与表对应的映射文件,映射文件是一种用以描述持久域和对象间的关系,以

及子类和对对象的代理等内容的 XML 文件。在这一步中首先需要配置与所使用的后台数据库相对应的配置文件,修改其中相应的项:

```
<property name = "database.url" value = "jdbc: JDBCConnect://localhost/travel"/>
<property name = "database.userid" value = "user"/>
<property name = "database.password" value = "mypass"/>
```

Hibernate 所支持的所有数据库配置文件均在 MiddleGen 目录下的 \config\database 子目录中。

接下来要对 build.xml 文件进行修改,修改的内容包括对数据库配置文件、应用程序名、输出路径等,以便在 Ant 中使用该文件。

```
...
<!DOCTYPE project [ <! ENTITY database SYSTEM "file:./config/database/mssql.xml" > ] >
<property name = "name" value = "TravlerManage"/>
<property name = "build.gen - src.dir" value = "C:\TravellerManage"/>
<hibernate destination = "${build.gen - src.dir}"
package = "org.hibernate.travel"
genXDocletTags = "true" genIntegratedCompositeKeys = "false"
javaTypeMapper = "middlegen.plugins.hibernate.HibernateJavaTypeMapper"
/>
...
```

在这两个文件配置好后运行 Ant, Middlegen 显示数据库中三个表间的关系如图 3 所示。

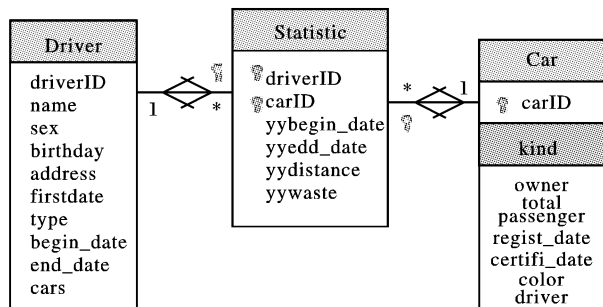


图3 实体关系图

在 Middlegen 的界面中对数据库表间的关系及表中列的属性进行调整,运行就能在输出路径中找到相应的数据库表映射文件。这样就省去了对所需的数据库表——设置烦琐的过程。

这样由此生成与表 Car、Driver、Statistic 对应的三个映射文件:Car.hbm.xml、Driver.hbm.xml 和 Statistic.hbm.xml。以 Driver.hbm.xml 为例其映射文件主要内容为:

```
<hibernate-mapping>
<class name = "Cartest.StationInfo" table = "StationInfo">
<!-- 定义表中列的映射 -->
<id name = "driverId" type = "java.lang.String" column = "driverID">
<meta attribute = "field-description">
@hibernate.id generator-class = "assigned"
type = "java.lang.String"
column = "driverID" </meta>
<generator class = "assigned" /> </id>
<property name = "name" type = "java.lang.String"
column = "name" not-null = "true" length = "30">
<meta attribute = "field-description">
@hibernate.property column = "name" length = "30"
not-null = "true"
</meta>
</property>
...
<!-- 定义与 Car 的多对多的关系 -->
<set name = "cars" table = "Statistics" lazy = "false" inverse = "false" cascade = "all" sort = "unsorted">
```

```
<key column = "driverID" /key>
<many - to - many class = "Car" column = "carID" outer -
join = "auto" />
</set>
```

```
...
</hibernate - mapping>
```

2) 由 Hibernate Extension 根据 1) 中生成的映射文件产生对应的 POJO, 这由 Hibernate Extension 包中的 hbm2java. bat 工具来实现。为了使用该工具, 首先我们需要配置一些参数, 打开 tools\bin\setenv. bat 文件, 修改其中的 JDBC\_DRIVER 和 HIBERNATE\_HOME 环境变量, 使其指向我们的实际 JDBC Driver 文件和 Hibernate 所在目录, 如:

```
set JDBC_DRIVER = c: \mysql\mysql. jarset HIBERNATE_HOME =
c: \hibernate(这要根据自己的实际情况而定, 选择实际的目录)
```

由此生成对应的 POJO。

3) 建 Hibernate 配置文件 hibernate. cfg. xml。该文件用于配置数据库连接属性以及一些 Hibernate 的数据库操作属性。

```
<?xml version = "1.0" encoding = "utf - 8"?>
<!DOCTYPE hibernate - configuration PUBLIC " - //Hibernate/
Hibernate Configuration DTD//EN"
"http://hibernate. sourceforge. net/hibernate - configuration - 2.
0. dtd">
<hibernate - configuration>
<session - factory>
<! 定义数据库驱动类型、数据库 url、用户名及密码!>
<property name = "hibernate. connection. driver _ class"> com.
jnetdirect. jsq. JSQDriver </property>
<property name = "hibernate. connection. url"> jdbc:
JSQConnect://localhost/travel </property>
<property name = "hibernate. connection. username"> user </
property>
<property name = "hibernate. connection. password"> mypass </
property>
<property name = "dialect"> net. sf. hibernate. dialect.
SQLServerDialect </property>
```

4) 车辆信息录入 DAO

```
public static void createCarInfo ( CarInfo Car ) throws
```

```
HibernateException{ Session session = sf. openSession();
Session. save( Car );
}
```

采用 Hibernate 进行数据持久层的设计具有以下几个方面优点: 1) Hibernate 是开源的, 不用专门购买, 而且有丰富的文档支持; 2) Hibernate 对持久层进行了封装, 不用程序员去写 SQL 代码; 3) Hibernate 有多种第三方工具的支持, 简化了系统数据持久层的设计; 4) 利用 Hibernate 进行数据持久层的设计由于能够完整地多表间的关系进行映射, 所以能够极大地提高数据访问的效率, 改善系统性能。

#### 参考文献:

- [1] HIBERNATE. 符合 Java 习惯的关系数据库持久化[EB/OL]. <http://www.hibernate.org/152.html>, 2005.
- [2] Hibernate In Action[EB/OL]. <http://www.JavaFan.NET>, 2005.
- [3] ALAMER D. Internationalized data in Hibernate[EB/OL]. tts symposium February 22, 2005.
- [4] 宋汉增, 沈琳. 利用 Hibernate 对象持久化服务简化数据库访问[J]. 计算机应用, 2003; 23(12): 135 - 137.
- [5] 沈锐. 基于 J2EE 物流系统持久层的 Hibernate 解决方案[J]. 电脑知识与技术, 2005; 3: 13 - 15.
- [6] 田珂, 谢世波, 方马. J2EE 数据持久层的解决方案[J]. 计算机工程, 2003; 29(22): 93 - 95.
- [7] 黄烟波, 等. 基于 Struts 和 Hibernate 的 J2EE 架构[J]. 计算机时代, 2004, 10: 29 - 30.
- [8] 李军怀, 周明全, 耿国华, 等. XML 在异构数据集成中的应用研究[J]. 计算机应用, 2002, 22(9): 10 - 12.
- [9] 朱庆伟, 吴宇红. 一种对象/关系映射框架的分析和应用[J]. 电子科技, 2004, 172(1): 54 - 57.
- [10] 飞思科技产品研发中心. Java Web 服务应用开发详解[M]. 北京: 电子工业出版社, 2002.

(上接第 2319 页)

倾向性的演化情况。

## 5 测试环境和方案

在完成民意模型的 HLA 改造后, 建立相应的硬件测试环境, 观察在该环境下 StarLink 支持民意联盟规模的能力。

测试环境配置如下: 1) 21 台微机(1 台为服务器, 其余 20 台为客户机, 分别编号为 1 ~ 21, 且以 100M 以太网交换机组网); 2) 服务器配置为: Pentium 4 1.4G, 256MB 内存, 40GB 硬盘; 3) 客户端配置为: Pentium 4 1.4G, 256MB 内存, 40GB 硬盘, 操作系统为 Windows Professional 2000 Service Pack 2 SDK 2.0。

实验步骤: 1) 在服务器上启动 StarLink, 并在其上加入第一个个体盟员, 观察其对资源的开销; 2) 同时在 1 号机上分别启动观察器和环境盟员; 3) 在 2 号客户机上添加个体盟员, 分阶段记录其对资源的开销; 4) 当 2 号客户机不能再添加个体盟员后, 依次在 4 ~ 21 号客户机上同时添加个体盟员直至不能添加为止。

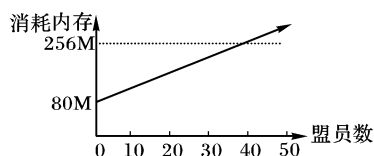


图5 盟员运行数目与内存消耗示意图

运行结果如图 5 所示。从图 5 可以看出, 内存消耗与运

行的盟员数目存在线性关系。当在单台计算机上运行大约 40 个盟员时, 程序消耗的内存超过了实际内存, 此时需要大量地利用虚拟内存, 因此程序运行速度明显下降, 最终导致整个仿真系统的推进速度迅速变慢。

## 6 结语

通过该测试表明, 随着联盟成员数目的增加, 网络开销呈增长趋势。尽管 CPU 和网络仍然能够具有足够的能力, RTI 支持能力仍然逐渐呈饱和趋势, 该实验表明, RTI 的规模支持能力与其实现方式存在着极大的关联性。在该仿真模型下, StarLink 能够有效支持的联盟成员数目至少为 600 个。

#### 参考文献:

- [1] 罗批, 司光亚, 胡晓峰, 等. SWARM 及其平台下建特定民意模型的探讨[J]. 系统仿真学报, 2004, 16(1): 5 - 7.
- [2] 姚益平. 高性能分布式交互仿真运行支撑平台关键技术研究[D]. 长沙: 国防科技大学研究生院, 2004.
- [3] MORSE KL. An Adaptive, Distributed Algorithm for Interest Management[D]. Irvine: University of California. 2000.
- [4] TAN G, XU L. An Agent-Based DDM for High Level Architecture[EB/OL]. <http://www.sisostds.org>, 2000.
- [5] 曲庆军, 等. HLA/RTI 中 DDM 的弱服务器模式实现方案[J]. 计算机仿真, 2003, 20(1): 90 - 93.