文章编号:1000-582X(2007)08-0034-06

一种匹配目的端模式的异构数据交换

潘大四[1], 杨梦宁[2]
(1.浙江警官职业学院 信息技术与管理系,浙江 杭州 310018;2.重庆大学 软件学院,重庆 400030)

摘　要:XML 已成为异构数据库之间进行数据交换的工业标准,但由于 XML 与数据库以及数据库之间模式的差异,须解决模式转换的问题。针对以往异构数据库之间数据交换灵活性差、实现困难等问题,利用对象/关系数据映射和对象与 XML 数据绑定技术,提出一种以目的端持久类作为源端生成 XML 数据交换文档的绑定类的方法,得到完全符合目的端模式的 XML 文档。该方法易于解析和存储,有效解决数据交换过程中模式不匹配的问题。

关键词:数据交换;XML;对象/关系映射;数据绑定
中图分类号:TP391　　**文献标志码**:A

许多组织、企业内部根据自身的情况与需求,采用不同的技术和体系结构建立了分布式的工作环境,业务数据由于种种原因通常都被保存在不同数据库中,不同系统之间数据交换的要求日益增多,因此不可避免要进行异构数据交换。所谓异构数据交换是指异构数据处理环境下(包括计算机体系结构的异构、操作系统的异构、数据库管理系统的异构和应用软件的异构等)不同数据源之间的互操作。

目前,常见的异构数据交换技术可以分为以下几种[1-2],如表 1 所示。

表 1　常见的异构数据交换技术

数据交换方法	交换机理	优点	缺点
中间数据库法	建立一个中间数据库,并根据关系和字段的定义在每个数据库和中间数据库之间建立一个中间件,使二者之间可以进行数据交换。	所需转换的模块少,可扩展性强	实现难度大,转换质量不高,转换时间长
电子数据交换(EDI)法	通过统一的格式来交换企业信息,企业之间通过专用的数据网络来交换数据	通用性好	格式不灵活,成本昂贵,安全控制困难,连接可靠性低
模型映射法	模型映射的原理是利用 XML 文档中数据模型的结构,显性或隐性地将其映射成数据库的结构,反之亦然	结构简单	效率低、灵活性差
数据仓库法	中心数据库负责提取各个场地自治系统的数据,并对各种数据有高度控制权	数据集中	交互性与实时性较差

XML 文档作为数据库之间传递数据的中介应当易于解析,即源端生成的 XML 文档结构应符合目的端数据库的结构,反之亦然。笔者在结合对象/关系数据映射(ORM)[3]和 XML 数据绑定技术[4]基础上,给出一种以目的端持久类作为源端生成 XML 数据交换文档的绑定类的数据交换方案,该方案与传统数据交换方案相比,源端生成 XML 文档与目的端解析 XML 文档的绑定类完全相同,易于解析,同时源端绑定类即为目的端数据库的持久类,满足目的端数据库结构匹配的要求。运用对象/关系映射技术,使数据交换与具体数据库类型无关。运用 XML 数据绑定技术,使读写 XML 文档数据时无需关注 XML 文档的具体结构。

1　相关研究

1.1　数据库的数据存取

存取数据库中的数据必须要连接数据库,实现数

收稿日期:2007-04-10
作者简介:潘大四(1970-),男,工程师,硕士研究生,主要研究方向为图形/图像处理、网络数据库,(E-mail)pandasi@ajjy.com.cn。

据库访问有 3 种方法:

1)基于 JDBC 直接连接的数据库访问;

2)基于连接池的数据库访问;

3)基于 ORM(Object/Relational mapping,对象关系映射)工具的数据库访问。

通过 JDBC 或者通过 SQL 字符串进行数据访问和查询,这些检索通常返回结果集,应用程序通过存储程序来匹配返回列的数据类型,从效率上来讲这种方案是非常脆弱的,因为它依赖于字符串和数据库中表及列的代码匹配,改变数据表名会导致要查找并修改所有实例的代码,将花费大量的开发时间去更新和维护持久层。最好的实现方式是不依赖于 JDBC、SQL 字符串和结果集,在 Java 对象和数据库实体之间建立一种自然映射。对象/关系映射(ORM)能够将对象映射到关系数据库中的记录行,将数据库结构封装,像使用普通对象一样调用数据库的相关接口,实现数据库的相关操作,不再需要复杂的持久层,仅需要少量的手工代码来维护映射,从而脱离具体的数据库,更适合于灵活的数据库移植与更新,数据库模式发生变化不影响程序,可以有效地提高开发的效率。

目前发展比较成熟的 ORM 产品有 Hibernate, Xdoclet, HyperJAXB 等,其中 Hibernate[5-6] 的轻量级 ORM 模型逐步确立了在 Java ORM 架构中的领导地位,甚至取代复杂而又繁琐的 EJB 模型。Hibernate 是一个开源的对象关系映射工具,它减轻了对 JDBC 的直接操作,使得数据检索和更新、事务处理、数据库连接池更加简单,完全着眼于关系数据库的 OR 映射,使它对于最终用户几乎是透明的。Hibernate 通过工具集可以匹配数据库结构自动生成 Java 代码和 ORM 文档。

1.2　XML 文档的读写

从数据库中读取数据信息后,应该描述这些数据信息并将其存储在 XML 文档中,反之应解析 XML 文档,提取数据存入数据库中。异构数据交换中大多数问题都会集中到解析 XML 和 XML 结构等方面。在这类技术领域,W3C 提出了 DOM 和 SAX 规范用来解析数据,SAX(the Simple API for XML)适合读 XML 而不适合写,在将对象写入 XML 文件时需要自己编写大量代码解决。DOM(Document Object Model),需要将整个文档都读入内存构造层次树解决,内存占用量大,对于较大的文档来说,效率较低,甚至可能造成系统崩溃。

数据绑定提供了一种简单而直接的方法,在使用之前将数据转换成对象或者将对象转换成数据的过程称为数据绑定。有了数据绑定,应用程序可以在很大程度上忽略 XML 文档的实际结构,而直接使用那些文档的数据内容,读写 XML 文档代码清晰,容易理解,并且方便开发和维护。除了简化编程之外,对于在内存中处理文档,通常所需要的内存比文档模型方法(如 DOM 或 JDOM)要少。由于不需要遍历文档结构以获取数据,因此用数据绑定方法访问程序内的数据要比用文档模型方法快。另外在输入时,一些特殊类型的数据(譬如数字和日期)可以被转换成内部表示,而不是保留为文本形式;这使应用程序可以更有效地使用数据值。

目前发展比较成熟的数据绑定产品有 Castor XML[7], JAXB RI, Xgen, Castor XML 数据绑定不需要定义 XML 文档格式,只要数据用类 JavaBean 的对象定义,CastorXML 就能自动生成表示这些数据的文档格式,而且可从文档重构原始数据。

2　以目的端持久化类构造 XML 文档的数据交换

数据交换指将本地数据变换成能被另一个系统识别的格式。由于不同数据库模式通常是由不同设计人员设计的,因此几乎所有数据库之间模式都是不匹配的,如命名冲突,即源模型中的标识符与目的模型中的不同,这时就需要重新命名;格式冲突,同一种数据类型可能有不同的表示方法和语义差异,这时需要定义两种模型之间的变换函数;结构冲突,如果两种数据库系统之间的数据定义模型不同,如分别为关系模型和层次模型,那么需要重新定义实体属性和联系,以防止属性或联系信息的丢失。

另外由于数据库表结构是平面的、结构化的,而 XML 文档是层次的、半结构化的,因此要求生成的 XML 文档结构必须适合目的端数据库的结构。

2.1　体系结构

在现实工作环境中,实际情况是数据库系统中已经存在并且存放着大量数据,需要解决的问题是如何对这些已知的不同模式的数据库之间进行数据交换。

具体解决思路为,首先通过对象/关系数据映射机制将数据交换双方(目的端与源端)的数据库表映射为包装类(持久化类),数据记录持久化使得数据交换脱离具体数据库类型,解决数据库之间格式冲突问题。然后将目的端的包装类传递到源端,源端映射的包装类从源端数据库中获取数据生成对象,由此对象将目的端传递过来的类实例化,此时便可通过 Java 函数变换解决不同数据库之间的命名冲突、格式冲突。最后在源端将此实例化对象通过数据绑定生成 XML 数据文档后传递到目的端,由于生成与解析 XML 数据文档使用相同 Java 包装类,即源端数据形成 XML 数据文档时,数据绑定时所使用类就是目的端持久化类,因而当 XML 数据文档传回到目的端后易于解析、存储。

数据库A中的数据转换到数据库B中的步骤如下(从RDBMS－A中转换数据到RDBMS－B中时,去除图1中标记a处虚线框模块,反之,若从RDBMS－B中转换数据到RDBMS－A中,去除图1中标记b处虚线框模块):

1)从数据库RDBMS－A导出需进行数据交换的表结构,由Hibernate持久化服务生成对应的ORM文件和Java Bean类。如图1中①处。

2)从数据库RDBMS－B导出相对应需要存入数据的表结构,由Hibernate持久化服务生成对应的ORM文件和Java Bean类。如图1中②处。

3)连接数据库,获取数据由Hibernate持久化服务构造JAVA OBJECTS(A),将步骤2中生成的Java Bean类传递到数据库A这端,从JAVA OBJECTS(A)中取得数据构造生成JAVA OBJECT(B),如图1中③处。然后利用Castor绑定JAVA OBJECTS(B),编组生成满足RDBMS－B模式的XML数据文档。

4)通过网络传递XML数据文档。如图1中④处。

5)根据2中生成的Java Bean类与输入的XML文档进行数据绑定,解组后用Hibernate持久化对象服务存入数据到RDBMS－B中。如图1中⑤处。

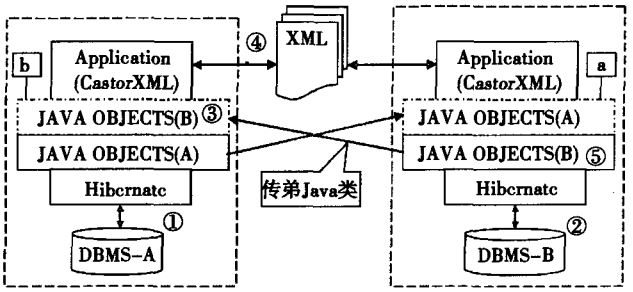

图1 数据交换框架

2.2 持久层设计

持久层设计即构建Hibernate基础代码(ORM映射文件与Java代码),通常有以下途径:

1)手工编写。

2)直接从数据库中导出表结构,并生成对应的ORM文件和Java代码。这是实际开发中最常用的方式,也是笔者所推荐的方式。通过直接从目标数据库中导出数据结构,最小化了手工编码和调整的可能性,从而最大程度上保证了ORM文件和Java代码与实际数据库结构相一致。由数据库产生代码通过MiddleGen for Hibernate和Hibernate_Extension工具包,可以根据现有的数据库,导出数据表结构,生成ORM和POJO(Plain Ordinary Java Object),即无格式普通Java对象。

具体步骤为:

1)配置目标数据库参数,本文数据交换双方使用的数据库分别为MySql与MS SQLServer,对应为MiddleGen目录下的mysql.xml、mssql.xml文件,按照实际运行环境修改文件中相应的数据库URL及数据库用户名和密码。

2)修改Build.xml以及MiddleGen根目录下的build.xml文件,此文件是Middlegen－Hibernate的Ant构建配置。Middlegen－Hibernate将根据build.xml文件中的具体参数生成数据库表映射文件。可配置的项目包括:目标数据库配置文件地址,Application name,输出目录,对应代码的Package name。

3)数据库中的表结构导入到MiddleGen的操作界面后,选定数据库表视图中的表元素,即可调整各个数据库表的属性,MiddleGen即可生成这些数据表所对应的Hibernate映射文件,每个映射文件对应了数据库中的一个表。

4)仅有映射文件还不够,还需要根据这些文件生成对应的POJO。Hibernate Extension工具集中hbm2java工具根据映射文件生成POJO,最终完成数据表对应的ORM文件和java代码。

2.3 数据绑定

在源端生成目的端所需的XML数据文档时,多数系统采用为源端方提供一个XML schema文件,即提供一个映射文件,供源端按照此映射文件形成XML数据文档,这样不仅生成映射文档复杂,转换时还需读取映射文档的结构,程序开发效率不高。CastorXML几乎能将任何"bean－like"的Java对象在XML之间自动进行转换。转换框架也可使用一组类描述符(ClassDescriptor)和字段描述符(FieldDescriptor)来描述如何来进行编组(Marshalling)和解组(Unmarshalling)。

笔者提出一种新的方案,首先Hibernate持久化对象服务自动生成目的端Java类(图2①处),接着将Java类传递到源端进行XML数据绑定(图2中②),构造XML数据文档,目的端数据绑定的Java类与源端数据绑定的Java类完全相同,从而确保编组XML数据文档与解组XML文档结构相同,使得源端形成的XML数据文档的结构完全符合目的端需求,提高解析性能。DOM和SAX都是从结构的角度处理XML文档,而Castor是以对象的方式进行处理的,编组(Marshalling)是在内存中为对象生成XML表示的过程。与Java对象序列化一样,这种表示需要包含所有依赖的对象:主对象引用的对象、这些对象引用的对象等等。解组(Unmarshalling)与编组相反,它是内存中根据XML表示构建一个对象的过程。

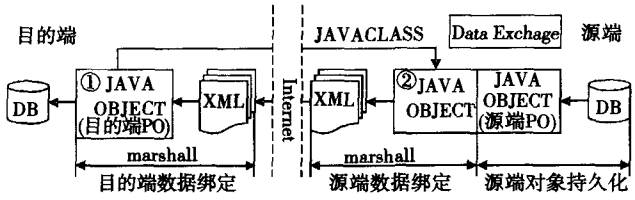

图 2　以目的端持久化类绑定源端 XML 文档

marshaller 和 unmarshaller 负责对象和 XML 文档间的转换，这种转换过程在服务器端完成。运行时，客户端的请求发送到服务器端后，对内存中的对象进行相应操作，然后将处理完毕的对象 marshall 成为 XML 文档。接着以 XML 文档格式发送回客户端，最后在客户端执行 unmarshall 操作将 XML 文档还原为内存中的对象。

2.4　数据交换系统实现

结合 Hibernate 与 Castor XML 技术，描述上述匹配目的端（数据请求端）模式的数据交换系统的实现过程。

1）持久层实现。Hibernate 分别从两个对应数据库中导出对应数据表，及对应的 ORM 文件和 Java 代码。

ClassFromSrcPO、ClassSrcChild 为源端生成的 Java class，分别代表一对多关系数据表对应的类，ORM 文件包括 ClassFromSrcPO. hbm. xml，ClassFromSrcChild. hbm. xml。映射文件中包括主键属性，外键属性，一般属性及其成员函数声明，另外包含关联关系类的声明。

ClassFromDesPO、ClassDesChild 为目的端生成的 Java class，分别代表对应于源端的一对多关系数据表对应的类 ORM 文件 ClassFromSrcPO. hbm. xml、ClassDesChild. hbm. xml，映射文件中包括主键属性，一般属性及其函数定义，另外包含关联关系类的声明。

生成 ApplicationContext. xml 中描述数据库 JDBC 驱动，数据 URL，数据库用户名，数据库用户密码，sessionFactory，数据库对应的 Dialect，映射资源等。

2）业务逻辑层实现。从源端数据库取出数据，形成目的端所需 xml 数据文档，以下是源端服务器中形成 XML 数据文档的实现过程，由 4 个模块组成：

```
protected void ConfigSetup( );
public List DataSelect ( String strhql, Session session);
public void TransferData( List listSrc);
protected void ConfigDown( );
```

ConfigSetup 函数中，创建配置对象，通知对象要映射的 class，取得会话工厂对象，使用已经配置的 JDBC 信息取得会话对象。

DataSelec 函数中取得源数据端数据，返回 List 对象。

ConfigDown 函数关闭对话，完成清理工作。

TransferData 函数：将 DataSelect 函数取得的数据赋值给由目的端 Java 类生成的对象，编组成 XML 数据文档，简化描述如下：

```
Public void TransferData( List listSrc) {
//…..
for ( ListIterator iter = ListSrc. listIterator( ) ; iter. hasNext( ) ; ) {
ClassFromSrcPO srcBean = (ClassFromSrcPO) iter. next( );
ClassFromDesPO Desbean = new ClassFromDesPO ( );
//(1)从源端读取数据赋值，若类型不匹配，转换后赋值
DesBean. setAttri( srcBean. getAttri( ) );
// 若有一对多关联关系，继续循环赋值
Iterator iterChild = srcBean. getChildClass( ). iterator( );
Set setChild = DesBean. getChildClass( );
while ( iterChild. hasNext( ) ) {
ClassFromSrcChid srcChild = ( ClassFromSrcChild). iterChild. next( );
SetChild. add ( new ClassFromDesChild ( srcChild. getAttr1( ),…….. );
}
}
File file = new File( "test. xml" );
Writer writer = new FileWriter( file );
Marshaller marshaller = new Marshaller( writer );
//(2)数据绑定到 XML 文档，为方便编组和解组，不使用映射文档
marshaller. marshal( Desbean );
}
```

代码注释 1）处，Java 对象从源端数据库中提取数据，若数据格式与目的端数据类型不同，或者同一种数据类型可能有不同的表示方法和语义差异，这时需要定义两种模型之间的变换函数，进行类型转换。注释（2）处，CastorXML 默认情况下，不需要定义输出 XML 数据文档格式，用类 JavaBean 的对象表示，Castor 就能自动生成表示这些数据的文档格式。源端生成的 XML 文档输出到目的端，由于目的端解组使用的类 JavaBean 与源端编组类 JavaBean 相同，而这些 JavaBean 类本身就是由目的端 Hibernate 持久化服务生成的，即：

源端数据绑定类 = 目的端数据绑定类 = 目的端持久化类

源端生成的XML文档传回目的端,解组后可以直接以访问对象方式将数据保存到目的数据库中,完成数据交换。

3 实验结果

通过实验来验证上述方案比直接使用JDBC、DOM技术在数据交换性能方面的提高。实验运行环境:Windows Server 2000,CPU Intel 双核3.0 G,内存2 G,2种不同数据库:SQL Server 2000和MySQL Server4.1,Hibernate3,castor-1.1。在不考虑带宽条件下,主要影响数据交换效率的4个重要因素:

1)数据库存取时间;

2)解组与编组XML文档时间;

3)内存使用;

4)启动时间。

最初几次实验发现速度并没有明显改观,经调试发现原因在于未对Hibernate进行优化条件下,以Hibernate存取数据库所用时间比通过JDBC访问要多,从处理过程上讲,Hibernate通过JDBC访问记录前必须要多经过一层对象到sql语句转换,JDBC是访问数据库最底层的接口和代码(指实现接口的驱动程序),Hibernate是对JDBC的更高一级的封装,就执行效率来说没有SQL/Procedure快,但Hibernate可以通过空间换取时间,因为它支持连接池、并发操作,这是优于JDBC的更高级的管理功能,通过以下几个方面调整,使得以Hibernate方式与JDBC方式存取数据库时间相等。

1)修改hibernate.properties,设置

hibernate.jdbc.fetch_size=50

hibernate.jdbc.batch_size=100

2)one-to-many的关系里将lazy设成true

3)HQL优化

4)映射文件优化(ID生成策略,二级缓存,延迟加载,关联优化。

在访问数据库效率相当的条件下,问题集中在解组与编组XML效率上,实验以一个完整格式的XML文档(test.xml,211KB)来考查两种方式下解组、编组(图3)以及内存使用情况(图4)。

程序开始执行到整个操作返回为止(将数据分解成对象,然后将对象编组回文档),大多数的时间花费在类装入以及为获得数据绑定框架代码而由JVM进行的本机代码生成(如图5)。可以看到这一启动时间通常比实际处理时间(即使是处理相当大的文档)要大好几倍,该启动时间所占总交换时间比例较大,是提高效率更关键的因素。综合比较数据库存取时间、解组与编组时间、内存使用及启动时间,可见以本方案进行数据交换可减少数据交换所需总时间。

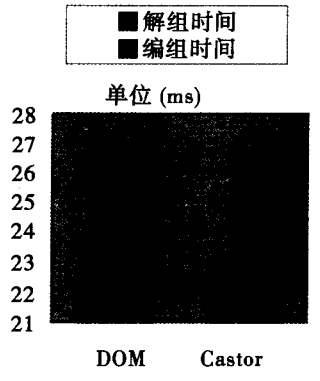

图3 解组与编组时间比较

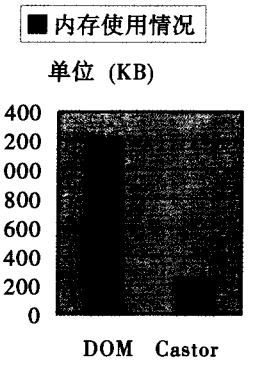

图4 内存使用情况比较

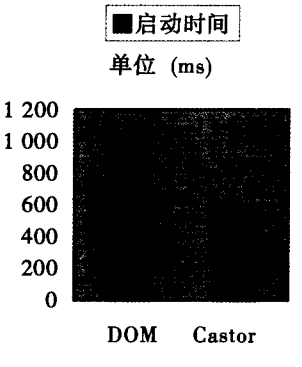

图5 启动时间比较

4 结 论

笔者针对以往异构数据库之间数据交换灵活性差、实现困难等问题,提出一种以目的端持久类作为源端生成XML数据交换文档的绑定类的数据交换方案,该方案利用Hibernate的对象持久化技术,简化对数据库数据的操作过程,专心地实现业务逻辑而不用分心于繁琐的数据库方面的逻辑,同时使得数据交换脱离具体数据库类型;利用XML数据绑定技术,可以通过操作Java对象读写XML数据而不必考虑文档结构;结合两种技术,用目的端生成的JavaBean持久类绑定源端XML数据文档,从而确保生成的XML数据

文档结构符合源端数据库结构,易于解析。实验表明文中方案有效降低了异构数据交换难度,提高了数据交换效率。

参考文献:

[1] 丁月华,杨敏,文贵华,等. 基于 XML 的异构数据源集成与交换的实现[J]. 计算机应用与软件,2006, 23(10):134-135,143.

[2] 李长河,赵洁,张亚玲,等. 一种安全异构数据交换技术的研究与实现[J]. 计算机工程,2007 ,33(2):88-89,93.

[3] 蔡雪焘. Hibernate 开发及整合应用大全[M]. 北京清华大学出版社, 2006:6-9.

[4] MCLANGHILIN B. Java and XML Data binding [M]. [s. l.]:Reilly& Associates, 2006:11-50.

[5] 夏昕,曹晓刚,唐勇. 深入浅出 Hibernate[M]. 北京:电子工业出版社, 2005:53-77.

[6] 何国辉,卿银波. 基于 XML 的数据交换系统设计[J]. 计算机工程与设计,2007,28(3):583-587.

[7] SCOTT L BAIN. XML Data Binding with Castor[DB/OL]、[2007-03-02.]http://www.netobjectives.com/xml.2003.

Data Exchange Mechanism Matching Target-End Schema Among Heterogeneous Data

PAN Da-si[1], *YANG Meng-ning*[2]

(1. Information Technology and Management Department, Zhejiang Police Vocational Academy, Hangzhou 310018, China; 2. College of Software Engineering, Chongqing University, Chongqing 400030, China)

Abstract: XML has become an industry standard for exchanging data between heterogeneous databases. Due to the different schema applied in XML and different databases, the problem of schema transformation has to be resolved. The autrors propose a novel data exchange mechanism binding source-end XML document by using target-end persistent class to deal with the difficulty of implementation and poor flexibility, based on the technology of object/relational mapping and XML data binding. The built XML document is consistent with source-end schema and is easy to be parsed and stored, hence this mechanism can resolve the problem of schema mismatching effectively.

Key words: data exchange; XML; ORM; data binding

(编辑　张小强)