基于 J2EE 的 SDAI 实现方法*

王 欣
(中国民航飞行学院 计算机与信息工程系, 四川 广汉 618307)

摘 要:提出了基于 J2EE 平台,采用多层结构的 SDAI 实现 EAST(EJB Access to STEP data)系统,从而实现产品数据的共享。EAST 系统以关系数据库作为 SDAI 的数据存储系统,基于 EJB 组件实现了可以多用户并发访问的 SDAI 对象层。还阐述了用 EJB 实现 SDAI 会话,访问远程 repository 以及 SDAI 数据字典和迟联编的实现机制。

关键词:STEP;SDAI;J2EE;产品数据共享
中图分类号:TP311.132.3 **文献标识码**:A **文章编号**:1672-6693(2005)04-0024-05

SDAI Implementation Method Based on J2EE

WANG Xin
(Dept. of Computer and Information Engineering, China Civil Aviation Flight College, Guanghan Sichuan 618307, China)

Abstract: This paper suggests EAST architecture implementation system, a based on J2EE-terrace and using multi-stratification SDAI to serve as the software infrastructure for product data sharing. EAST, taking relational database, as its data storage system has a standard access interface object layer implemented by EJB component supporting concurrect access, and a data dictionary to facilitate late binding. Implementation technologies of SDAI session, access to remote repository, data dictionary and late binding are also described.

Key words: STEP; SDAI; J2EE; product data sharing

现代企业越来越多地采用虚拟企业、计算机集成制造和并行工程等先进制造技术。计算机辅助工具 CAx 间信息交换和共享是其中的关键技术之一。而实现产品信息的无缝交换和共享,必须建立统一的产品数据模型,这是多系统之间信息共享的基础;其次是要提供标准的数据访问机制。STEP(Standard for the Exchange of Product Model Data)是一个关于产品数据表示和交换的国际标准。它以一种不依赖于具体系统的中性机制描述产品整个生命周期中的产品数据。

为了屏蔽各种实现方式中不同数据存储系统之间的差异,STEP 中定义了标准化的数据访问接口 SDAI(Standard Data Access Interface)[1],以支持对 EXPRESS 语言建模的数据的存取。本文将 SDAI 和 J2EE(Java 2 Platform Enterprise Edition)[2] 两种标准结合,提出了 SDAI 实现 EAST(EJB Access to STEP data)系统,以实现虚拟企业中的产品数据共享。

1 SDAI 的功能

SDAI 是 STEP Part22 中定义的标准的数据访问接口。各种应用系统可以通过这一接口动态、随机地访问用 EXPRESS 建模的产品模型的数据实例。SDAI 允许访问不同的数据存储系统,屏蔽了不同物理存储方式之间的区别,为不同的数据存储系统提供了一个标准化的数据访问机制。

SDAI 可以访问的数据存储系统包括文件、内存工作区或数据库等。中性文件实现的数据交换仅限于文件级的粗粒度方式,且实时性差。数据库则可以实现有效的数据共享,提供灵活多样的数据访问粒度,实现数据的实时共享,而且数据库不仅共享产品数据,还共享表达数据语义的产品数据模型。

虽然对象模型能较好地反映 EXPRESS 模型的

* 收稿日期:2005-05-25
作者简介:王欣(1973-),男,四川绵阳人,讲师,博士,研究方向为计算机辅助设计与仿真。

语义,但面向对象数据库技术还不成熟,未在企业中得到大规模应用。

与之相比,关系数据库技术成熟,在企业中应用广泛,并提供标准的数据访问接口,如JDBC、ODBC、SQL等,易于实现多数据库产品数据的共享。因此,本文在实现SDAI时,选择关系数据库作为SDAI数据存储系统,在不同的关系数据库之上提供SDAI接口。

分布式对象计算、软件组件、中间件技术以及建立在这些技术之上的多层计算模式为SDAI的实现提供了新的方法。组件提供定义良好的接口,将复杂的业务逻辑从客户端和服务器端分离出来,集中在中间层实现。组件应用于多层结构中,极大地提升了软件系统的可扩展性,更适合虚拟企业的复杂计算环境。

SDAI标准由EXPRESS语言描述,但必须由编程语言来实现语言联编。SDAI语言联编分为早联编和迟联编。迟联编比早联编具有更好的通用性和开放性,与具体的应用模式无关,可以用一致、抽象的方式访问任意EXPRESS模式的数据实例。本文实现了SDAI迟联编。

本文提出建立在J2EE平台基础之上的采用多层计算模式的扩展的SDAI实现EAST系统。SDAI的功能由运行在中间层应用服务器中的EJB组件来实现。

2 基于J2EE的SDAI实现

2.1 EAST系统总体结构

J2EE是一个分布式环境中开发、部署和运行应用程序的计算平台和应用开发标准。在J2EE体系中,Enterprise JavaBeans[3]规范提供了一个标准的服务器端分布式组件模型。J2EE提供了各种系统级服务,使得开发人员可以专心于业务逻辑的实现。Enterprise beans包含两种类型:session beans和entity beans。EJB组件部署在中间层的应用服务器中。应用服务器提供了EJB的运行环境—EJB container。

EAST系统采用可扩展的多层计算模式,具有以下特点。

(1) SDAI集中在中间层,用enterprise beans组件实现,客户端计算机无需安装SDAI库。

(2) SDAI实现采用迟联编方法,使得应用系统通过一个统一的接口就可以访问任何应用EXPRESS模式描述的数据,具有很好的通用性。

(3) SDAI集中在中间层,整个系统易于维护,易于根据虚拟企业的需求进行扩展。

应用程序通过中间层的SDAI可以访问虚拟企业内各个关系数据库中的STEP数据,实现了数据的有效共享。

EAST实现了数据字典。数据字典存储在关系数据库中,供应用程序查询,实现了数据共享的关键——数据模型的共享,并且是SDAI迟联编的基础。EAST系统体系结构如图1所示。

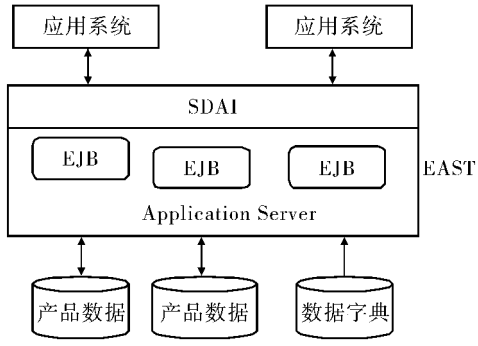

图1 EAST系统体系结构

2.2 SDAI会话的实现

SDAI会话是所有SDAI操作的运行环境。SDAI规范规定,应用程序对数据的所有操作都必须处于会话的环境之中。SCHEMA session_schema中定义了描述SDAI会话的数据结构,其中最重要的实体是sdai_session。和描述产品信息的数据不同,会话状态的描述不需要持久化,因此sdai_session很适合用session beans来实现。

Session beans表示应用程序和服务器之间建立的会话,是非持久的,生命期只持续一个会话的时间。一个session bean包含会话过程中的会话状态。当多个应用程序并发访问,即多个session bean同时运行时,每个session bean都能保持自己的会话状态,不会互相干扰,保证了并发访问的正常进行。

与实体sdai_session相关的会话操作包括打开和关闭repository、事务处理和事件记录等操作。SDAI会话为应用程序提供了一个运行环境。应用程序通过open session操作发出创建并初始化环境的请求,然后应用程序可以访问数据字典,查询将要访问数据的数据结构。产品数据存储在repository表示的存储系统中,打开repository后就可以通过一系列操作访问其中的数据。最后通过close session操作结束当前会话。

在本系统中,当应用程序通过EJB的机制新创建一个实现sdai_session的session bean的实例时,就表示应用程序和服务器之间建立了一个会话,这

个会话在基于 EJB 的 SDAI 实现中就是一个 SDAI 会话。以此作为访问入口点,应用程序就可以进行后续的各种 SDAI 操作。

因为 SDAI 会话是用 session bean 实现的,每当应用程序发出开始一个 SDAI 会话的请求时,EJB container 都会负责创建一个 session bean 的实例,代表一个 SDAI 会话。每个应用程序的 SDAI 会话状态都会得到很好地维护,多个应用程序的 SDAI 会话之间不会互相干扰。因此,基于 EJB 的 SDAI 实现可以保证多个并发应用程序的同时访问,是一个多用户的 SDAI 实现。

OpenRepository 操作表示打开一个 repository,访问其中的数据。一个 repository 就表示一个物理存储系统。Part 22 中不包含对远程 repository 访问的支持,但是允许一个实现系统以其特定的方式访问远程 repository。在 EAST 系统中,用一个关系数据库实现一个 repository。因此 OpenRepository 操作就是和数据库建立连接。在虚拟企业环境中,运行 EJB 的应用服务器和数据库分布在网络中不同的主机上,实现 SDAI 的 EJB 需要访问远程 repository。利用 Java 数据库访问接口 JDBC 实现对远程 repository 的访问。

2.3　SDAI 对数据库中 STEP 数据的访问

在 Part 22 中,实体 EntityInstance 抽象地描述所有 EXPRESS entity 的实例,它的子类 ApplicationInstance 是表示应用模式实体实例的抽象数据类型。存储在 repository 中的 ApplicationInstance 的实例就是产品数据。为了给应用程序提供一个导航并定位到确定产品数据的手段,SDAI 定义了组织实体实例的存储结构。一个 repository 中包含若干 model,一个 model 是若干个实体实例的集合,是实体实例的最基本的组织单位。这使得无论产品数据的物理存储机制如何变化,对用户呈现的都是由 model 和 repository 构成的逻辑组织结构,并且给用户提供了进行导航并定位到指定的产品数据的手段。

ApplicationInstance 实例表示的产品数据需要持久化存储到 repository 中。因此,使用 EJB 中的 entity beans 实现 ApplicationInstance 实体。

Entity beans 是存储系统中的数据在内存中的映射,对应关系数据库中的记录或面向对象数据库中的对象等。Entity beans 将底层存储系统中的数据和对这些数据的获取、更新、存储等操作通过面向对象技术封装在组件里,为底层的数据提供了一个位于内存中的面向对象的视图,是实际数据之上的对象层。

作为内存中的对象,entity beans 需要持久化到存储系统中,例如存储到面向对象数据库或通过对象——关系映射存储到关系数据库。

当 entity beans 用关系数据库作为持久化存储系统时,涉及到两种映射:(1)对象模型向关系模式的映射;(2)运行时把关系数据库中查询得到的数据库记录转换成 beans 对象,或把 beans 对象转换成数据库记录。

Entity beans 通过特有的方法——ejbLoad 和 ejbStore 来实现内存中对象实例的属性值和数据库中持久数据之间的同步。EJB container 负责根据当前事务的情况自动调用 ejbLoad 和 ejbStore 方法。

与此相似,用关系数据库作为 SDAI 数据存储系统时,也涉及两种映射。用面向对象语言实现的 SDAI 是面向对象的导航方式的数据访问接口,关系数据库提供 SQL 作为面向记录的数据访问接口。SDAI 就是关系数据库之上的对象层,应用程序通过它可以操作 EXPRESS 建模的 STEP 对象,导航式地访问 STEP 数据和将 STEP 数据持久化保存。此时,两种映射为:(1) EXPRESS 模式向关系模式的映射;(2)运行时把关系数据库中查询得到的数据库记录转换成 STEP 数据对象,或把 STEP 数据对象转换成数据库记录。

以关系数据库作为 SDAI 存储系统时,为了在关系数据库中存储 STEP 数据对象,需要在 SDAI 系统运行之前,完成 EXPRESS 数据模式向关系模式的映射[4]。这就是数据字典实现的功能。

2.3.1　数据字典的实现

迟联编机制的核心是数据字典,即以一种通用的、独立于模式的格式存放对模式信息的描述数据。STEP Part 22 中专门定义了一个数据字典模式——SCHEMA SDAI _ dictionary _ schema。SDAI _ dictionary _ schema 是应用模式的元模式。EXPRESS 应用数据模式就作为该模式的实例数据保存在数据字典中。

数据字典作为应用数据的元数据存储在关系数据库中。数据字典的使用包括建立和动态查询过程。

(1) 在 SDAI 实现系统运行之前,SDAI _ dictionary _ schema 模式映射为一系列关系表。使用 SQL 语句,将应用数据模式的定义信息作为数据字典内容加入到数据字典中。这是数据字典的建立过程。

(2) 在 SDAI 实现系统启动时数据字典被装载到内存中。Part 22 中定义的游标(Iterator)操作使

得在数据字典中可以根据实体的名称查找实体的定义和根据属性的名称查找属性的定义。

本文使用 Java 的聚集类 Vector 实现数据字典在内存中的组织。Vector 类也提供了 Iterator 接口,恰好对应 SDAI 的游标操作。为了实现数据字典的层次逻辑结构,各个 Vector 对象也组织成层次结构。

2.3.2 SDAI 迟联编的实现 在 SDAI 的若干类操作中,最核心的就是对 EXPRESS 实体实例的操作,包括:(1)创建实例;(2)获取实例的属性值;(3)设置实例的属性值。

在迟联编中,实体 ApplicationInstance 是表示应用模式实体实例的抽象数据类型。通过 model 操作创建的应用模式实体的实例都是 ApplicationInstance 类型的实例。这些对象实例中只有属性,不包含方法,可以看作是数据对象。

迟联编的关键在于提供一个统一的操作集,并且操作都通过类和属性的名称进行参数传递。例如以下例子创建一个实例,并设置和读取属性值。

```
ApplicationInstance ap1 =
model. CreateEntityInstance(" Customer ");
ap1. putAttribute(" id ", 1000);
ap1. putAttribute(" name ", " someone ");
name = ap1. getAttribute(" name ");
```

可以看出,使用迟联编接口时,在编译时不能确定要创建哪个实体的实例和要访问的属性,只能在运行时根据参数动态地确定。因此要实现 SDAI 迟联编,必须实现:(1)运行时根据实体的名称动态地确定要创建的实例;(2)根据属性的名称动态地确定要访问的属性。

本文提出使用 Java 的 Reflection 机制实现对实体实例的动态访问。在 Java 程序中,每当编译一个类时,自动创建一个与之联系的 Class 对象,其中包含对应的类的元数据。

可以通过如下方法,实现在运行时根据类的名称动态地创建该类的实例。

```
Class c = Class. forName(" Customer ");
Object cust = c. newInstance( );
```

Class 对象的 getField(String name)方法根据属性名称返回表示该属性的 Field 类型的对象。Field 类中包含一系列 get 和 set 方法用于读写属性值。

设置上述 Customer 类的实例 cust 的属性" id "为 1000,可以用如下方法实现。

```
Field f = c. getField(" id ");
f. setInt (cust,1000);
```

这样就实现了在运行时根据属性的名称动态地访问属性值。

除了上述的动态访问之外,当以关系数据库作为持久存储系统,而通过 SDAI 创建、保存、访问和更新 EXPRESS 实体实例时,必然涉及到 STEP 数据对象在运行时的合成和分解。SDAI 实现系统必须实现:(1)从数据库读取对象时,将数据库记录的各字段的值组成对象;(2)保存对象时,把对象分解,将属性值保存到数据库记录的各字段中。

为了简化实现,避免复杂的内存操作,采用委托(delegation)机制实现了 STEP 数据对象的表示以及对象的合成和分解工作。委托是面向对象中常用的技术,指的是一个对象将方法的执行委托给另一个对象。在该方法中,应用模式的每一个实体,直接映射为一个 Java 类。这种类是数据对象类,其中只有属性项和操作数据库的方法,没有其它方法。操作数据库的方法负责数据对象的合成和分解,即将对象属性值写入到数据库表中和从表中读取数据填充到属性项中,例如。

```
Class Customer {
    int id;
    String name;
    void storeRow(Connection con) { . . } //将属性值写入到数据库中
    void loadRow(Connection con) { . . } //读数据库,刷新属性值
}
```

在迟联编接口的基础上,在实现 ApplicationInstance 的 entity bean 的内部保留一个对象引用 objref,指向真正实现应用模式实体的 Java 对象。

实现 ApplicationInstance 的 entity bean 的 ejbLoad 和 ejbStore 方法所负责的 entity bean 和数据库之间的同步,即对象的合成和分解,实际上也是通过 objref 委托给被引用对象的数据库操作方法去实现。基于委托的迟联编技术如图 2 所示。

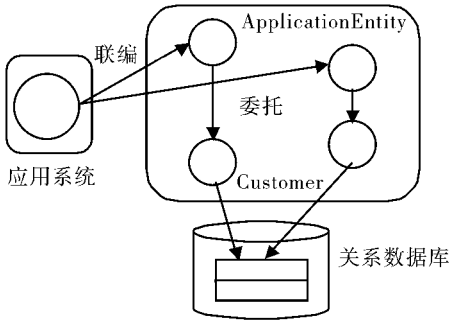

图2 基于委托机制的迟联编

通过 EAST 的迟联编接口,对于任意应用模式的实体,应用程序创建的都是 ApplicationEntity 类型的对象。实现 ApplicationInstance 实体的 entity bean 相当于是那些实现应用模式实体的 Java 对象的 wrapper。它将实现应用模式实体的各种具体的 Java 类向用户隐藏起来,用户看到的只是一个实现了迟联编方式的 SDAI 接口。

从上面 Java 类 Customer 的例子可以看出,与应用模式实体对应的 Java 类中只包含从实体转换过来的属性和数据库操作方法,形式十分规则。因此很容易构造一个模式转换器,自动将一个 EXPRESS 应用模式转换为一组 Java 类。而且在抽象的迟联编接口之下,在 EAST 系统中添加新的应用数据模式,这种扩充对于应用程序是完全透明的,应用程序只需要查询数据字典就可以访问更多的数据模型的数据。这种方式类似于组件的即插即用,应用程序在不受任何影响的情况下就可以访问更多种应用协议的数据。

3　结论

以 SDAI 作为统一的 STEP 数据访问接口,以关系数据库 SDAI 数据存储系统,可以实现有效的产品数据共享。本文在 J2EE 尤其是 Enterprise JavaBeans 组件模型的基础之上,采用多层计算模式,实现了一个可扩展的 SDAI 实现系统 EAST。

参考文献:

[1] ISO/FDIS 10303-22. Product Data Representation and Exchange-Part 22, Implementation Methods: Standard Data Access Interface[S]. ISO TC184/SC4, 1998.

[2] Sun Microsystems, Inc. Java 2 Platform Enterprise Edition Specification, v1.3[EB/OL]. http://java.sun.com, 2005-02-01.

[3] Sun Microsystems, Inc. Enterprise JavaBeans Specification, Version 2.0[EB/OL]. http://java.sun.com, 2005-02-01.

[4] LI Xiaolin, YI Hong, WU Xiying. EXPRESS Schema Implementation in Relational Database[J]. Journal of Computer Integrated Manufacturing System, 1999(1): 64-68.

(责任编辑　游中胜)