# 基于JMS消息的远程异构数据库存储

# Remote Heterogeneous Database's Storage Based on JMS Message

摘　要：JMS具有Java语言特有的平台无关性，满足了异构平台的交互行为；XML正逐步成为数据表示和信息交换的工业标准，并广泛应用于异构数据库之间的数据交换。JMS和XML的集成应用给现代企业的分布式应用提供了一个信息交互平台。本文讨论了这样一种应用模式，并辅以一个具体的实例详述了其应用。

关键词：JMS　XML　异构数据库

沈良忠　黄德才（杭州浙江工业大学 信息工程学院　310012）

## 1 JMS概述

Java消息服务(Java Message Service)简称JMS，是一种接口规范，最初是为了使Java应用程序可以访问已有的面向消息的中间件MOM(Message Oriented MiddleWare)而设计的，现在已经成为J2EE开发平台的一部分。JMS规范给出了标准的基于消息的中间件的主要访问方法，消息系统的应用为分布式系统的通信提供了便利，因为消息的产生者和接收者以异步的方式进行通信，不同于基于RPC或者RMI的方式，彼此不需要知道对方的任何信息。消息的产生者可以直接把消息发送到目标对象，消息的使用者就可以从目标对象中提取消息。

一个JMS消息就是一个字节包，其中封装了作为消息的载体并加上一些元数据以标志该消息，主要包括消息头，消息属性和消息体。消息头包含客户端应用和JMS提供者使用以标志和路由消息的各种选项；

消息属性是一些属性名/值对，用于消息的选择机制；消息体中就是消息的具体内容了，其消息类型又具体分为MapMessage、StreamMessage,ObjectMessage,BytesMessage和TextMessage，其中TextMessage现在被广泛地用于支持XML格式的数据。为了在XML和JMS API之间架起桥梁，一些厂商如BEA系统公司提供了自定义的JMS扩展，提供了XMLMessage类，以便直接支持XML消息机制。Sun公司目前也正在开发用于XML消息的Java API（JAXM）。

## 2 JMS编程模型

JMS支持两种消息模式：点到点和发布/预定。点到点模式采用传统的队列模式，保证发送的消息只能被一个客户端所接收，而且只能被接收一次，消息由消息队列进行维护；而发布/预定模式最大的特点就是发送的消息可以被多个客户端所接收，只要客户端预订了该消息主题，则当主题发现有新的消息到来的时候，就会向所有登记的客户端广播该消息，消息由消息主题进行维护。

尽管JMS编程模型支持两种类型的消息传递，但是采用统一的编程模式。如图1所示：

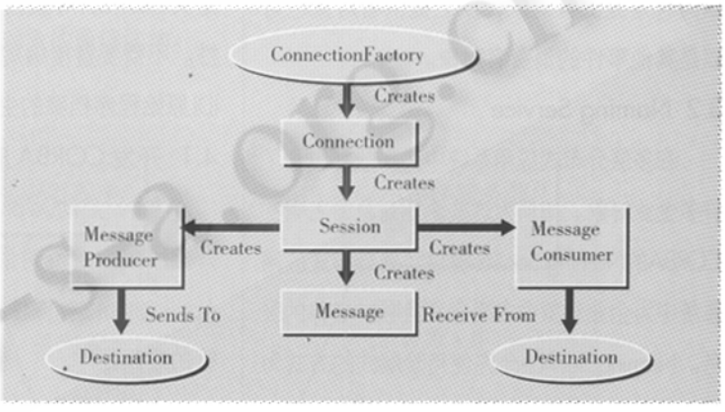


图 1

（1）ConnectionFactory被称为连接对象工厂，它和Destination（目标对象，具体包括队列对象和主题对象）一起由消息服务提供者进行管理并注册进服务器的JNDI目录中，JMS客户端可以通过使用JNDI服务查找该对象。

（2）Connection对象由ConnectionFactory对象进行创建，其封装了客户端到目标对象连接的底层细节。

（3）Session对象由Connection对象创建，用于客户端到目标对象的通信，消息活动的执行，并且封装了对消息的事务处理。

（4）MessageProducer和MessageConsumer对象均由Session对象所创建，表示消息的发送者和接收者；消息服务提供者以Persis5ent模式处理消息，把消息放入一个数据源（如服务器硬盘），作为客户端发送操作的扩展，这样确保消息在其他系统失败的情况下依然可用。

像JMS API这样的消息系统允许双方交换基于JMS API的消息载荷，前提是双方在会话的时候均能提供兼容的JMS API服务。因此，双方是否能遵从相同的格式或协议就成了解决问题的关键，而这个就是XML大显身手的地方。

## 3 XML与数据库的转换

XML被广泛的应用于异构数据库之间的数据表示和转换。由于XML文档在本质上讲是层次性的，而现在的主流数据库均是关系型的，因此两者之间需要一种映射机制。模板驱动和模型驱动是目前常见的两种方法。其中模型驱动又可以分为基于表映射和基于对象映射。

由于模板驱动到目前为止仅支持从一个关系数据库提取数据生成XML文档的情况，而模型驱动中基于对象的映射虽然应用广泛，但是比较复杂，而且对此类应用没有特别的优势存在，所以采用基于表映射。基于表映射把XML文档看作是具有特定结构的序列表，直接与数据库中的一张表格相对应，虽然在应用上有所限制，但是操作非常简单，如图2。

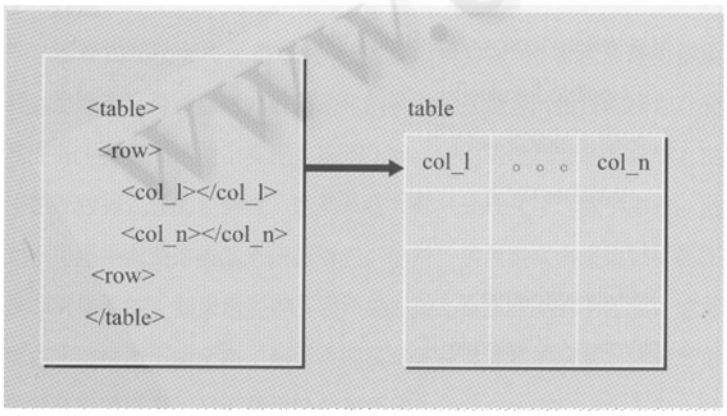


图 2

从数据库选取数据，然后生成XML文档，可以采用以下的步骤：

（1）打开从一个表上选取的数据集（数据集也可以通过多表之间的关系查询得来，但是为了操作的方便，采用一表对应于一个XML文档）；

（2）以数据集相关的表名创建文档元素；

（3）对于数据集中的每一条记录，创建一个行元素，默认为<row> … </row>；

（4）对于记录中的每一字段，创建一个列元素，并且把该记录对应于该字段的值作为该列元素的值，默认为<fieldname>fieldvalue</fieldname>；

（5）关闭数据集；

生成的XML文档将作为消息载体被传输，一旦客户端接收消息后，就解析出XML文档，然后读取XML文档生成标准SQL语句执行数据库操作。生成标准SQL语句的过程可以通过下面的步骤来完成：

① 取得文档根元素的名称，即用于存储操作的表名；

② 在每个行元素开始，首先创建SQL语句模板，例如Insert into tablename（）values（），然后读取列元素名称和元素值，分别将其加入SQL语句模板中；

③ 在每个行元素结束时，执行上一步中创建的SQL语句,实现对数据库的存储操作。

## 4 JMS和XML集成应用实例分析

本应用实例是一个VOD（视频点播系统），客户端使用Windows操作平台、MySQL数据库；远程数据中心使用Linux操作平台、Oracle数据库。数据中心负责提供影片的数据信息给每个客户，这样就形成了一个分布式的网络体系结构，如图3。

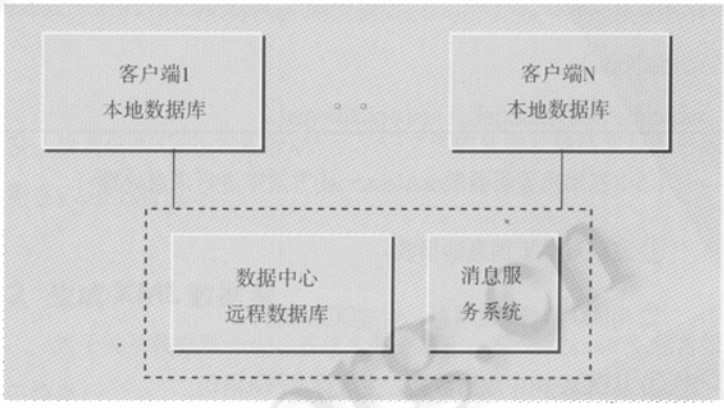


图 3

由图3可知，消息系统的发布/预定机制比较适合此类应用。将新添加的影片信息保存为XML文档，该文档将被作为消息载体发布到消息服务系统中，客户端设置的监听器发现新的消息到来时，就从消息服务系统接收该消息并解析出XML文档，然后对该XML文档进行读取，生成标准SQL语言并执行，该过程完成数据插入到本地MySQL数据库的操作。

### 4.1 JMS客户端操作

（1）通过在JNDI目录服务中查找连接对象工厂和主题对象，创建到消息服务提供者的连接。

```
String FactoryName="TopicConnectionFactory";
String TopicName="Movie";
Context jndiContext=new InitialContext();
TopicConnectionFactory topicConnectionFactory=
(TopicConnectionFactory)jndiContext.lookup(FactoryName);
Topic topic=(Topic)jndiContext.lookup(TopicName);
TopicConnection topicConnection=topicConnection Factory.
createTopicConnection();
```

（2）创建会话对象，同时设定事务处理和消息确认参数。消息的发送者和接收者都要在会话层，完成消息的发送和接收。

```
TopicSession topicSession=topicConnection.create
TopicSession(false,Session.AUTO_ACKNOWLEDGE);
```

（3）创建消息发布者进行XML文档的发布。

```
TopicPublisher publisher=topicSession.createPublis her(topic);
TextMessage message=topicSession.createTextMessage();
MESSAGE.SETtEXT(xmldOCUMENT);
publisher.publish(message);
```

（4）创建消息预订者并且为之注册消息监听器,以便异步接收消息。

```
TopicListener topicListener=new TopicListener();
TopicSubscriber subscriber=topicSession.createSub
scriber(topic);
subscriber.setMessageListener(topicListener);
```

（5）在消息监听器的onMessage()方法中进行消息处理：

① 完成对XML 消息的接收；

② 用JAXP解析XML消息，生成DOM树；

③ 从DOM树中取出数据，生成标准SQL语句；

④ 执行该SQL语句，完成对MySQL数据库的数据存储操作。

```
public void onMessage(Message in Miessage) {
TextMessage message=(TextMessage)in Message;
 try {
     String text=message.getText();
     StringReader reader=new StringReader(text);
     try {
     DocumentBuilderFactory dbf=DocumentBuilder
     Factory.newInstance();
     DocumentBuilder db=dbf.newDocumentBuilder()
     InputSource source=new InputSource(reader);Document doc=db.parse
(source);
     //得到文档元素
     Element table=doc.getDocumentElement();
     String strTable=table.getNodeName();
     //声明行元素和列元素
     NodeList nlist_row,nlist_col;
     Node node_row,node_col;
     nlist_row=table.getElementsByTagName("row")
     //针对行元素的外循环
     for(int row=0;row<nlist_row.getLength();row++)
      { node_row=nlist_row.item(row);
     if(!node_row.getNodeName().equals("text")) {
     nlist_col=nlist_row.item(row).
     getChildNodes();
     String strSQL="insert into"+strTable+"(";
     String strValues="values(";
     String strName,strValue;
     intj=0;
     //针对列元素的内循环
     for(int i=0;i<nlist_col.getLength();i++) {
     node_col=nlist_col.item(i);
      strName=node_col.getNodeName();
      if(!strName.equals("#text")) {
       strValue=node_col.getFirstChild().
          getNodeValue();
      if(j>0) {  strSQL+=",";
         strValues+=","; }
      j++;
      strSQL+=strName;
      strValues+=""+strValur+"";
     } }
     strSQL+=")"+strValues+")";
     ....//得到标准的SQL语句，并且执行数据库操作
      } } } catch (JMSException e) {
       e.printStackTracel);
```

## 5 结束语

JMS和XML的集成应用提供了一种解决方案，两者的结合应用不仅可以构建安全、可靠、灵活的应用系统，而且可以整合原有的应用系统，具有高度的扩展性，相信JMS和XML的集成应用将会越来越广泛。


参考文献

1 陈华君，J2EE构建企业级应用解决方案， 人民邮电出版社。

2 http://www.javaworld.com/javaworld/jw-02-2000/jw-02-jmsxml.html。