基于 XQuery 处理器的异构数据集成中间件①

王 波，张永祥

(重庆大学 计算机学院，重庆 400044)

摘 要：利用中间件方式进行异构数据集成是异构数据集成研究的热点。针对目前中间件方式处理关系数据库数据与 XML 数据集成在查询易用性以及效率的不足，基于各数据源的数据格式以及存储方式，把异构数据源分成关系数据库数据源，XML 数据源和非结构化数据源三类，对每一类用 XML 模式处理，构建虚拟视图，使待集成的异构数据源构建成一个逻辑数据库。通过在中间件中引入 XQuery 处理器，使其处理 XML 数据和关系数据库数据并行，并结合 SQL 语言以及 JDBC 设计逻辑数据库操作语言及接口，为用户操作异构数据源提供统一的方法。在此基础之上设计并实现了异构数据集成中间件,通过模拟实验表明该中间件可以支持数据查询与更新操作，简单易用，扩展性较好。

关键词：数据集成；中间件；全局视图；SQL

Heterogeneous Data Integration Middleware Based on XQuery Engine

WANG Bo, ZHANG Yong Xiang

(Computer School, Chongqing University, Chongqing 400044, China)

Abstract: The research of heterogeneous data integration based on middleware is a hot research in this area. Considering the querying efficiency and the ease of use, classified the heterogeneous data source three types based on the data's format and saved method, using xml schema to rebuild each data source, construct a logic database. Take the XQuery processor as a part of middleware to process the xml data specially, then designed uniform interface to operate the data source refer to SQL and JDBC for the user. Based on this idea, designed and implemented a heterogeneous data integration middleware, the experiment showed that the middleware can deal with searching and updating data, with simple using and good expansibility.

Key words: data integration; middleware; global view; SQL

异构数据问题一直是数据库领域的经典问题，国内外学者在这方面做了大量研究，典型的集成方式有以下三种，联邦数据库集成，数据仓库方式，基于中间件方式[1]。联邦数据库集成方式查询的效率较高，其缺点是扩展性不够好，对于新添加数据源，需做工作量较大；数据仓库方式要求较高，需要特别的硬件平台支持，结构设计较为复杂，通用性不强；中间件方式相对来说灵活，也出现了多种数据集成中间件，其中针对关系数据库的较多。典型的中间件如斯坦福大学的 TSIMMIS, Nimble[2]分析这些中间件，发现在应用接口上比较复杂，用户需要学习 MSL 语言以及 XML-QL 语言来进行数据查询。类似文献[3]中的中间件接口易用但其查询处理的策略上可以进一步的改进。

在综合分析考虑异构数据集成的实现，易用，查询处理效率以及可扩展的特点上，选用中间件的方式，参考 SQL 语言，XQuery[4]语言以及 JDBC 接口，设计并实现了基于 XML 模式的异构数据集成中间件 HDAM(Heterogeneous Data Source Access Middleware)。

① 收稿时间:2011-02-23;收到修改稿时间:2011-03-24

1　HDAM整体设计

数据集成的主要目的是给用户提供访问数据源的统一接口，而用户不必考虑具体数据源的内部特性，以及数据之间的异构性[5]。数据集成中间件的用户主要是了解一些专业知识的用户，比如会使用 SQL 的程序员，应用的场景主要是某个业务需要访问多种数据源，比如多个关系数据库，原生 XML 数据库中的数据，本地文件系统中的 XML 格式数据，Excel 文件中的数据等。基于以上几个方面的分析，本文设计的中间件主要提供给懂得 SQL 的程序员应用，能够集成关系数据库的数据，XML 格式的数据，Excel 以及文本文件中的数据。

1.1 整体设计结构图

中间件 HDAM 的整体设计结构图如下：

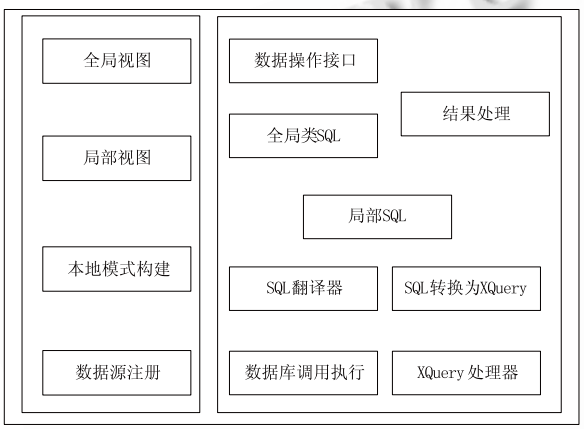

图 1　HDAM 整体结构

如图 1 所示，HDAM 分成左右两个部分。左边由四个模块组成，数据源注册模块，提供数据源注册接口，负责数据源的加载；本地模式构建模块，从注册的数据源中获取注册数据源的元数据，从而构建本地模式；局部视图模块，根据本地模式以及内部定义的局部视图 Schema 构建局部视图；全局视图模块，由局部视图以及自定义的全局视图 Schema 构建整个注册数据源的全局视图。左边模块实现异构数据源的视图集成，为数据操作提供基础。右边部分分为数据操作接口模块，提供全局数据操作查询及数据更新接口；全局类 SQL，结合标准 SQL 语法定义数据操作语言的语法规则；局部 SQL，针对某个数据源本地模式的标准 SQL 语句；SQL 翻译器，结合具体关系数据库的特性，把局部 SQL 语句翻译成数据库可以执行的语句；数据库调用执行，传递操作语句，并对返回结果的初步处理；SQL 转为换 XQuery，根据 SQL 以及 XQuery 的语法规则，把 SQL 语句翻译成 XQuery 语句；XQuery 执行处理器，执行 XQuery 语句，并处理结果；结果集成模块，对各数据源返回的结果结合全局查询语句进行最终的处理。

1.2 数据集成流程

图 2 是从用户视角看整个集成流程，及用户应用该中间件需要做的工作。首先，注册需要集成的数据源到中间件中。其次，构建全局视图，此处构建的全局视图类似新建的关系数据库，关系数据库中以表为基本的存储单位，此处的全局视图是基于局部视图，每一个局部视图类似关系表，以 XML 模式展现。再者，根据业务逻辑的需要，结合 SQL 语法及全局视图编写类 SQL 语句实现集成数据的操作。最后，调用执行接口执行编写好的类 SQL 语句并对返回结果处理。

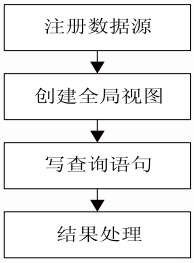

图 2　集成流程

2　数据源注册

在 HDAM 中，可以集成的数据源有关系数据库数据，XML 格式的数据以及 Excel 中的数据，根据这些数据的特点分成三种类型加以描述及处理。关系数据库中的数据，存储在关系数据库服务器上，对应的数据操作最终都是由服务器来执行，把这类数据源称为第一类数据源。而对于 XML 格式存储的数据，可以分成两种，其一是存储在原生 XML 数据库服务器（如 Exist）中的数据,对这种数据的操作在服务器上执行，其二是本地磁盘中或者远程 web 服务器上存储的 XML 格式的数据文件，对这种数据的操作，必须由中间件实现其操作功能，这两种数据源称之为第二类数据源。对于本地磁盘中存储的 Excel 数据以及文本文件数据如果不采取适当的格式转换，就没有办法对其数据进行操作，在 HDAM 中把这类数据转换成 XML 格式的数据，而后利用 HDAM 的 XQuery 处理器进行

相应的操作，在中间件中把这类数据源称为第三类数据源。

2.1 数据源分类注册

考虑到三类数据源各自特点，对每一类数据源设计注册接口，以便处理。第一类数据源利用 JDBC 方式注册到 HDAM 中，需要用户提供数据库所在服务器地址，端口号，数据库名，需要集成的表，用于连接的用户名密码基本信息。第二类数据源，原生 XML 数据库服务器中的数据同第一类数据源提供的基本信息类似，此处不是利用 JDBC 而是 XQJ 与服务器连接，对于本地文件只需提供路径以及文件名。第三类数据源需要用户提供路径及文件名，Excel 数据还需提供具体的每张表单的名称，序号，列的数量，利用 POI 软件包加载到 HDAM 中。

2.2 本地模式提取

本地模式定义：用 XML Schema 来表示一个数据源的元数据信息。比如关系数据库中用二维表来存储数据，则相应的表的基本信息（列的信息，主键信息，外键信息）称为元数据，用 XML Schema 语法来组织这些元数据，称这种组织好的 XML 模式为本地模式。第一类数据源通过 JDBC 中的 DatabaseMetaData 接口可以获取注册数据源的元数据信息，再经过一定的转换规则使其以 XML 模式重新构建成本地模式[6]。对于第二类数据源，如果注册的数据文件存在模式文件，则直接获取之，如果不存在则应用模式构建算法建立与之匹配的模式文件。第三类数据源必须构建模式提取算法获取本地模式。

3 全局视图构建

通过获取注册数据源的本地模式，使得所有数据在逻辑上都用 XML Schema 描述，从而初步消除不同类型数据源之间的异构性。为了进一步的消除其异构性，比如命名以及属性等的异构性，则应该有本地模式转换为局部视图以屏蔽掉一些存储细节，给用户提供比较直观的视图，以便用户更好的对数据进行操作。

3.1 本地模式到局部视图

局部视图定义：在本地模式的基础上，通过 HDAM 内部命名规则，对每一个注册数据源分配唯一的名字，提取本地模式的字段名。第一类数据源的局部视图以“db_index”为前缀，其中 index 表示注册到中间件中的关系数据库的编号，由 HDAM 根据数据源注册的顺序自动生成。其次对于每一个注册的表，在此基础之上利用“#”标志，则对于注册的不同关系数据库，以表为最小的基本单位，构建局部视图。依据同样的规则第二类数据源以“x_index”为前缀，第三类数据源以“o_index”为前缀构建局部视图。从而第一类数据源以表位基本单位，构建局部视图；第二类数据源以 XML 文件为基本单位构建局部视图；第三类数据源，如果是 Excel 文件，以表单为基本单位，若是文本文件，以文件为基本单位构建局部视图。

3.2 基于局部视图构建全局视图

在构建好局部视图的基础之上，以局部视图为基本单位构建全局视图，同时建立起局部视图之间的关系，在逻辑上构成一个完整的全局视图[6]。此时构建好的全局视图类似于构建好的一个关系数据库，呈现给用户。

4 数据操作

类 SQL 语言定义：以标准的 SQL 语言为基础，在此之上每一个字段前面添加 HDAM 的内部命名标志，目前不支持 SQL 语言中的简写，别名以及嵌套查询。在全局视图基础之上用户可以利用类 SQL 语言，根据业务需求编写类 SQL 查询语句来查询注册的数据。全局类 SQL 语句执行流程如图 3。

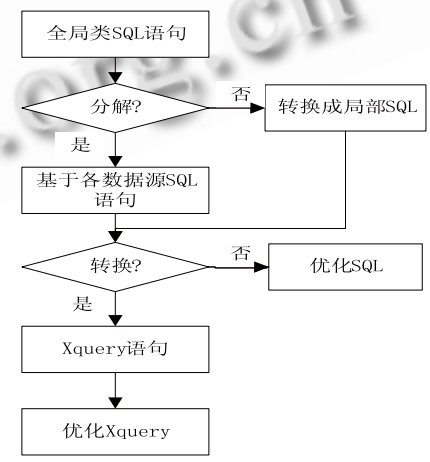

图 3 全局类 SQL 语句执行流程

4.1 基于全局视图的类 SQL 查询语言

典型的 SQL 查询语句格式：

SELECT table1.col1, table1.col2, table2.col1…… FROM table1, table2, table3 inner join table4 on

table3.col1 = table4.col1, ……WHERE table1.col4 > pram1 and table2.col4 like ‘%zh’ or table4.col4 < pram2 ……GROUP BY table5.col5

类 SQL 语言的格式基于 SQL 标准，结合全局视图对标准查询语句的每一个字段添加数据源标志前缀。数据源前缀有以下三种：“db_index#”,“x_index#”, “o_index#”其中 index（index=1,2,3……）为注册每一个数据源时 HDAM 自动生成。根据这种规则，假设 table3,table4 为第一类数据源, table1, table2 为第二类数据源， table5, talbe6 为第三类数据源。以上查询语句用类 SQL 语句表示为：

SELECT x_1#table1.col1, x_1#table1.col2, x_1# table2.col1……FROM x_1#table1, x_1#table2, db_1#table3 inner join db_1#table4 on db_1#table3.col1 = db_1#table4.col1, ……WHERE x_1#table1.col4 > pram1 and x_1#table2.col4 like ‘%zh’ or db_1# table4. col4 < pram2 ……GROUP BY o_1#table5.col5

4.2 全局语句的分解

类 SQL 语句基于全局视图，能够表示某一业务需求，但须经过一定的方法加以处理使其变得可以被具体的数据源服务器调用执行。针对类 SQL 语言的特点设计了全局语句分解算法，使其分解成基于某个具体数据源的 SQL 语句。分解算法主要步骤如下：

分解全局查询语句

输入：全局查询语句

输出：针对各个局部数据源的 SQL 语句

Step1: 验证类 SQL 语句的合法性，查询语句必须包含 SELECT 以及 FROM 关键词，字段是否合法，是否含有嵌套语句，是否存在简写以及别名，本转换算法目前不支持嵌套，简写及别名；如果输入的语句合法，转下一步，否则返回提示信息；

Step2: 根据全局查询语言的关键词，select, from, where, group by, order by 把全局查询语句分解成五个子句；

Step3: 对每一子句，根据数据源标志前缀“db_ index#”,“x_index#”及“o_index#”把每一子句分解成不同的部分；

Step4: 根据全局查询语句中出现的数据源前缀重新组合查询子句，使其变成针对具体数据源的查询语句；

Step5: 去除数据源前缀，返回标准的 SQL 语句。

4.3 SQL 语句转换为 XQuery 语句

经过以上分解，全局类 SQL 语句转换成基于局部数据源的标准 SQL 语句，注册数据源若是第二类或者第三类，数据源所在的服务器是不能够执行的。这两类数据都能够以 XML 格式表示，要查询 XML 数据，存在两种方式，其一利用 DOM 或者 SAX 接口调用相应的方法解析 XML 数据过滤不需要的数据；其二是利用 XQuery 语言，应用该语言查询 XML 数据类似于 SQL 语言查询关系数据库中的数据[7]。方法二查询效率较高，目前也存在比较成熟的 XQuery 处理器，HDAM 采用这种方法查询 XML 数据。

采用 XQuery 语言查询第二，三类数据，则相应的 SQL 语句必须转换为 XQuery 语句。比较 SQL 查询语句与 XQuery 的 FLOWR 语句，两种查询语句之间存在很大的相似之处。典型的 FLOWR 语句由 For, Let, Where, Order by, Return 四个部分组成，与 SQL 查询语句的 FROM, WHERE, ORDER BY, SELECT 对应。HDAM 中利用这一特点，把查询语句转换为 FLOWR 语句，转换算法如下:

转换 SQL 为 XQuery

输入：SQL 查询语句

输出：XQuery 的 FLOWR 语句

Step1: 验证 SQL 语句的合法性，一条查询语句必须包含 Select 以及 From 子句，检测是否含有嵌套语句是否存在简写以及别名，本转换算法目前不支持嵌套，简写及别名；如果输入的语句合法，转下一步，否则返回提示信息；

Step2: 根据查询语句的关键词 select, from, where, group by, order by 把查询语句分解成五个子句；

Step3: 对每一子句分别处理，select 子句转化为 return 子句，from 子句转换为 for 子句，where 子句转换为 where 子句，order by 转换为 order 子句；

Step4: 根据上一步转换的子句，按照 FLOWR 的语法规则构造 FLOWR 查询语句。

4.4 基于各数据源语句的执行

经过上面全局语句的分解及转换之后，全局语句变成基于各个数据源的查询语句。对于第一类数据源, 基于注册数据源的查询语句是标准 SQL 语句，而具体的关系数据库服务器在实现上存在各自的一些特点，在执行之前应该按照各个数据库服务器的特点进行修改，以便某个关系数据库服务器能够执行。HDAM 中利用一个开源的数据库查询语言翻译器，把标准的 SQL 语句翻译成具体数据库能够执行的 SQL 语句。查询结果应该转换成 XML 格式，以方便结果的合成。第二类，三类数据源如果是本地数据文件则调用 HDAM 内置的 XQuery 处理器（Saxon）进行查询，查询结果为 XML 格式。根据全局类 SQL 语句的顺序重

新组合各个数据源查询的结果，返回给用户。

如果注册数据源只有第一类数据源，数据动态操作支持数据的插入，删除，更新；更新操作语句与数据查询语句一样使用类 SQL 语言。若注册数据源中有第二类或者第三类数据源，数据的更新意义不明显，数据量较大的情况下处理效率比较低，目前 HDAM 不支持第二类，三类数据源的动态操作。

5 HDAM实现及验证

在实现上分两部分组成，第一部分构建全局数据源视图，第二部分在全局视图的基础上结合 SQL 语句实现查询与数据更新操作。HDAM 开发中，主要涉及到以下软件包 Jdom,poi,CowNewSQL,saxon9he。

假设某市教育局想获得该市几所高校学生选修课的情况，高校甲学生选修课的数据存储在关系数据库 mysql 中，高校乙的信息以 XML 文档存储，另一高校用 Excel 表格存储学生选课信息。利用 HDAM 查询三所高校的数据操作流程如下：

(1) 注册数据源

```
GlobalSource gs = new GlobalSource();
String dbUrl1 = jdbc:mysql://172.20.52.167:3306/test";
String dbUser1 = "root";String password1 = "root";
String tableNames1[] = {"ChosCourseInf", TInf"};
……
gs.regXAccess(xUrls, null, null);
gs.regOAccess(oUrl1, null, null, extInf);
```

(2) 创建全局视图

```
gs.createGView()
<GView>
<Lview Alias="db_1#ChoseCourseInf">
  <Filed>stuNo</Filed>
  ……
  <Key>stuNo</Key>
  <FKey
ref="teacherinf.teacherId">teacherId</FKey>
</Lview>
<Lview Alias="db_1#TeacherInf">
  <Filed>teacherId</Filed>
  ……
  <Key>teacherId</Key>
</Lview>
……
</GView>
```

(3) 根据需求及全局视图编写类 SQL 语句，假设查询选修心理学的学生姓名，任课老师

```
String sqlStr = "select db_1#ChoseCourseInf.stuName, db_1#ChoseCourseInf.courseName, db_1#TeacherInf.name … … where db_1#ChoseCourseInf.courseName = '心理学' and x_1#chosecourse.cName = '心理学' and " o_1#course.kechengming = 'xinlixue'" ;
```

(4) 查询执行及结果处理

```
XmlSet xmls = gs.executeQuery(sqlStr);
```

查询结果片段如下：

```
<stuname>苏望阳</stuname>
<coursename>心理学</coursename>
<name>赵强盛</name>
……
```

6 结论及总结

在比较分析目前存在的异构数据集成方案中，本文选择以中间件的方式作为数据集成的方案，针对开发者设计了异构数据集成中间件 HDAM。 该中间件以数据源的现有存在方式为基础，对数据源进行分类处理，中间件内置 XQuery 处理器，对不同的类别的数据进行并行处理，以提高中间件的查询效率。操作语言选用类 SQL 语言，接口易于调用，扩展性较好，最后通过实验验证了中间件的可行性以及正确性。该中间件的功能还有待进一步的完善，比如进一步的支持嵌套查询等。

参考文献

1 杨雪梅,董逸生,王永利,钱江波.异构数据源集成中的模式映射技术.计算机科学,2006,33:87−90.

2 袁晓洁,于仕涛,李志梁.基于 Mediation 的异构数据集成系统 HDIS 设计与实现.计算机工程与应用,2006,1:161−165.

3 李烨,冯志勇.基于 XQuery 的数据集成研究.微处理机,2008,3(6):120−123.

4 王银辉.XQuery 权威指南.北京:电子工业出版社,2009.

5 甄玉钢,刘璐莹,康建初.基于 XML 的异构数据库集成系统构架与开发.计算机工程,2006,32(2):85−87.

6 刘威,杨丹.基于虚拟视图的异构数据库集成平台的研究.计算机技术与发展,2009,19(6):91−94.