基于虚拟视图的异构数据库集成平台的研究

刘 威,杨 丹
(重庆大学 软件学院,重庆 400044)

摘 要:随着信息化建设的推进,企业各部门都根据自身需求建立了信息管理系统。为解决企业内部异构数据库数据共享的问题,提出了一个基于虚拟视图的异构数据库集成平台的系统架构方案,详细描述了各组件的功能,设计了元数据的数据对象模型,以及它们与XML文件的映射规则,研究和讨论了虚拟视图的定义和SQL语句解析等关键问题。平台采用标准SQL语句作为查询语言,降低了用户的使用难度。设计了一个基于XML的缓存机制,有效地实现了查询优化,提高了系统的响应效率。

关键词:异构数据库集成;XML;虚拟视图;SQL;缓存机制
中图分类号:TP311 **文献标识码**:A **文章编号**:1673-629X(2009)06-0091-04

Research of Heterogeneous Database Integration Platform Based on Virtual View

LIU Wei, YANG Dan
(School of Software Engineering, Chongqing University, Chongqing 400044, China)

Abstract: With the promotion of information development, enterprises have developed their own information management system based on their requirements. In order to share data among the heterogeneous databases, gives a system architecture of heterogeneous database platform based on virtual view. Each component is described in detail. The designs of mapping between data object model and XML is presented. The definition of virtual view and SQL statement parser are discussed mainly. The platform adopts the standard SQL as the query language to reduce the usage difficulty. The cache mechanism based on XML improves the response efficiency, implements the optimization of query.

Key words: heterogeneous database integration; XML; virtual view; SQL; cache mechanism

0 引 言

在信息化建设过程中,企业先后构建起多个信息系统。由于开发的时间、需求,以及开发商的不同,以至于在企业内部,每个部门都建立了自己独立的信息系统,每个系统都使用独立的数据库,导致各部门之间信息不能分享,从而形成了"信息孤岛"。如何从整体上有效地组织和管理这些数据,消除"信息孤岛"现象,就是数据集成需要解决的问题。

目前最典型的数据集成方案主要有三种:联邦数据库、中间件和数据仓库。它们在不同的着重点和应用上解决了数据共享的问题。从体系结构上,这三种方案可以分为两类[1]:一类是虚拟视图法,另一类是物化方法。

收稿日期:2008-09-09;**修回日期**:2008-12-25
作者简介:刘 威(1985-),男,硕士研究生,主要研究方向为软件工程、数据库等;杨 丹,博士生导师,主要研究方向为软件工程、运筹学与系统工程等。

文中提出了一种基于虚拟视图法的异构数据库集成方案,设计了一个以XML为中间媒介的缓存机制,使用标准SQL作为查询语言,屏蔽底层数据库和信息系统的异构性,向用户提供统一的数据视图和查询接口。

1 异构数据库集成平台的系统架构

使用虚拟视图法的好处在于,数据仍保存在各数据库中,集成系统仅提供一个虚拟的集成视图和对该集成视图查询的处理机制[2]。该方法不需要保存底层数据,底层数据库具有高度的自治性,而且查询结果保证是最新的。所实现的系统分为用户接口层、虚拟视图层和数据适配层,其结构如图1所示。

系统的主要工作流程:用户输入SQL查询某个虚拟视图,SQL解析器先查询缓存区,如果有缓存,则直接返回结果;否则,根据虚拟视图和全局数据字典,将该SQL查询分解成各底层数据库对应的子查询,然后

交由数据适配器。数据适配器将子查询分发到各相应数据库,并将查询结果返回给结果集合并器。结果集合并器根据连接条件合并结果集,返回给用户。

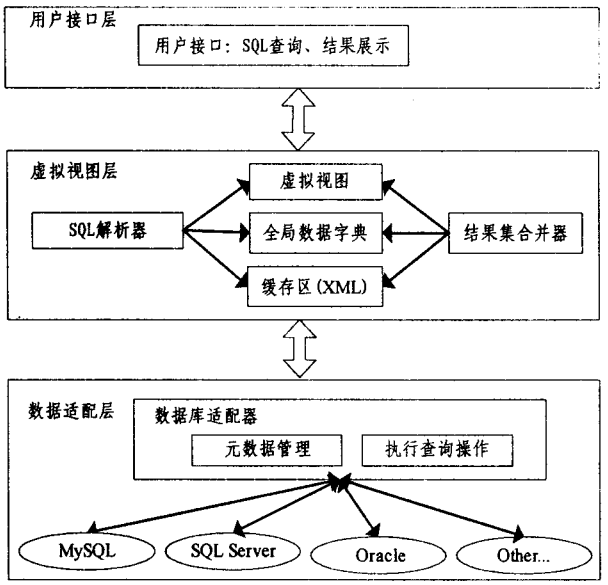

图 1 系统结构

系统主要组件及功能介绍如下:

(1) 数据适配器:负责数据库元数据管理和执行查询操作。数据库的元数据是指数据库的结构信息,包括数据库连接配置、表属性、列属性的描述。当有新的数据库添加进来时,数据适配器就负责提取其元数据。在本系统中,利用统一 DTD 描述的 XML 文件来存储这些元数据信息,以屏蔽各数据库的差异性。数据适配器的另一个功能就是接收虚拟视图层分解后的子查询语句,执行具体的查询操作,并返回结果集。

(2) 虚拟视图:虚拟视图是提供给用户的查询视图。在本系统中,虚拟视图由用户手动定义和维护。用户可以将来自不同数据库的表添加到同一个虚拟视图中,并通过 SQL 语句实现跨多数据库的查询。

(3) 全局数据字典:提供对所有元数据对应的数据对象模型(Data Object Model)的查询接口,以在创建虚拟视图和分解 SQL 语句时作为参考。

(4) 缓存区:包括对数据库元数据、SQL 解析以及结果集的缓存,以 XML 为数据载体。

(5) SQL 解析器:该模块实现对 SQL 语句的语法解析。它利用虚拟视图和全局数据字典验证 SQL 语句的合法性,并将查询语句拆分成针对各具体数据库的子查询 SQL,同时记录分解后各子查询之间的组合条件。

(6) 结果集合并器:接收数据适配器执行子查询后的结果集,并根据记录的子查询之间的组合条件,对结果集进行过滤合并。

2 系统实现

2.1 数据对象模型

文中实现的系统只针对关系型数据库,暂不考虑支持半结构化数据等其他数据源。综合分析关系型数据库的特点,设计了如图 2 所示的元数据对应的数据对象模型。

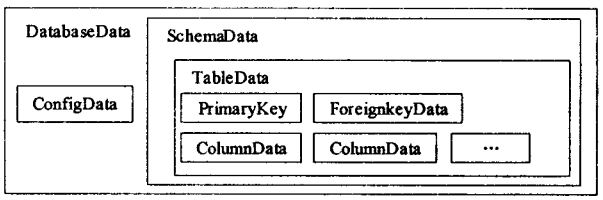

图 2 数据对象模型 - 元数据

一个 DatabaseData 对象对应于一个数据库,它包含一个 ConfigData 对象和一个或多个 SchemaData 对象。ConfigData 就是数据库的连接配置信息,包括配置名、驱动名、数据库 URL、用户名、密码、创建时间,对应的 XML 文件的 DTD 描述如下:

```
<! ELEMENT config (database * )>
<! ELEMENT database (name, driver, url, user, pwd, createtime)
>
<! ELEMENT name (#PCDATA)>
<! ELEMENT driver (#PCDATA)>
<! ELEMENT url (#PCDATA)>
<! ELEMENT user (#PCDATA)>
<! ELEMENT pwd (#PCDATA)>
<! ELEMENT createtime (#PCDATA)>
```

一个 SchemaData 对象包含一个或多个表(TableData),而一个 TableData 对象包含一或多列(ColumnData)以及主键(PrimaryKey)、外键(ForeignkeyData)等约束信息。DatabaseData 对应的 XML 文件的 DTD 描述如下:

```
<! ELEMENT database (schema * )>
<! ATTLIST database name CDATA "">
<! ELEMENT schema (table * , view * )>
<! ATTLIST schema id CDATA "">
<! ELEMENT table (tablename, primarykey, foreignkey, column
+)>
<! ELEMENT view (tablename, primarykey, foreignkey, column
+)>
<! ELEMENT tablename (#PCDATA)>
<! ELEMENT primarykey (#PCDATA)>
<! ELEMENT foreignkey (#PCDATA)>
<! ELEMENT column (columnname, type, allownull, attribute,
defaultvalue)>
<! ELEMENT columnname (#PCDATA)>
<! ELEMENT type (#PCDATA)>
<! ELEMENT allownull (#PCDATA)>
```

<!ELEMENT attribute (#PCDATA)>
<!ELEMENT defaultvalue (#PCDATA)>

2.2 数据适配器

数据适配器定义了接口 DbAdapter.java，它向外界提供两个方法，一个是 public DatabaseData getDatabaseData(ConfigData data)，另一个是 public ResultData query(SQLCommand sqlCmd)。前者负责元数据的提取并返回数据对象 DatabaseData，后者接收包含 SQL 查询语句以及数据库连接信息的 SQLCommand 对象并返回结果集 ResultData。接口实现主要使用 JDBC 等相关技术。java.sql.DatabaseMetaData 接口提供了获取数据库元数据的方法，如表属性、约束等。另一个接口 ResultSetMetaData 则提供了用来获取列的相关属性的方法。这些接口都是由相应的驱动包实现。

2.3 虚拟视图与全局数据字典

虚拟视图在相关文献中也称为全局视图。全局视图是相对于底层数据库的局部视图而言。从局部视图映射到全局视图，一般采用三种方法：GAV(Global-As-View)、LAV(Local-As-View)、GLAV[3]。这三种方法可应用范围很广，从关系型数据库到 XML 等结构化数据。鉴于文中实现的系统只考虑关系型数据库，因此采用了由用户自定义虚拟视图的方式，定义好的虚拟视图以 XML 形式存储，同时也定义了其对应的数据对象模型 VirtualViewData。虚拟视图的 XML DTD 描述如下：

<!ELEMENT root (virtualview*)>
<!ATTLIST virtualview name CDATA "DefaultView">
<!ELEMENT virtualview (table*)>
<!ELEMENT table (tablename,dbname,schema)>
<!ELEMENT tablename (#PCDATA)>
<!ELEMENT dbname (#PCDATA)>
<!ELEMENT schema (#PCDATA)>

从上面的 DTD 中可以看出，虚拟视图只保存表名、数据库名和模式名，不再存储其他具体数据。一个虚拟视图里面的表可以来自多个不同的数据库，而用户无需知道哪个表来自哪个数据库，只需要把这个虚拟视图当成某个实际数据库，进行查询操作即可。虚拟视图就像一个索引，每个索引项的具体内容都保存在底层局部视图中。

全局数据字典(Dictionary.java)主要是以 Hashtable 的形式存储 DatabaseData，VirtualViewData 等数据对象，并提供查询接口。比如方法 TableData lookup(String vvName, String table)，参数 vvName 是虚拟视图的名字，table 是虚拟视图 vvName 中的表，这个方法将返回包含表 table 具体属性的数据对象 TableData。

2.4 查询处理

查询处理是数据集成平台的关键问题。现在大部分基于 XML 的数据集成平台都采用 XQuery 作为查询语言[4,5]。XQuery 是一种用于查询 XML 的语言。使用 XQuery 对只熟悉 SQL 的用户来说，增加了使用难度，因此，本系统采用标准 SQL 作为查询语言。

用户查询时，先选定某一个虚拟视图，然后输入 SQL 语句。SQL 解析器负责解析该 SQL 语句，并拆分成多个子查询，其主要工作流程为：

(1) 验证 SQL 语法的正确性。

(2) 将 SQL 语句拆分成三部分：SELECT_LIST(SELECT 和 FROM 之间的语句)，FROM_LIST(FROM 和 WHERE 之间的语句)，WHERE_LIST(WHERE 后面的语句)。

(3) 解析 FROM_LIST 部分，得到待查询的表。根据全局数据字典，验证每个表是否存在。不存在，则抛出异常 SQLParseException。如果只是单表查询，则生成 SQLCommand 对象交由数据适配器执行；如果是多表查询，则检查多表是否位于同一个数据库，是，则生成 SQLCommand 对象交由数据适配器执行；否，继续下一步。

(4) 解析 SELECT_LIST 部分，得到待查询的所有列名。根据全局数据字典，将列与 FROM_LIST 中解析出的表一一对应起来。如果某列不属于 FROM_LIST 中的任何一个表，则抛出 SQLParseException。

(5) 解析 WHERE_LIST 部分。通过连接谓词 AND 或 OR 拆分成相对单独的条件。根据全局数据字典，验证条件里面涉及的表及列名是否合法。再依次解析每段条件，判断该条件是否可以分解到子查询中。为了获得较高的查询效率，应尽可能将查询条件分解到子查询中[6]。

(6) 根据以上解析结果，合并生成一个或多个 SQLCommand 对象和一个或多个保存着连接条件的 SQLCondition 对象。

下面以文献[7]里面使用的 CAPS 数据库为例子，来示例查询处理过程。CAPS 数据库共有以下四个表：

Head(CUSTOMERS) = {cid, cname, city, discnt}
Head(PRODUCTS) = {pid, pname, city, quantity, price}
Head{AGENTS} = {aid, aname, city, percent}
Head{ORDERS} = {ordno, month, cid, aid, pid, qty, dollars}

假定 CUSTOMERS、PRODUCTS、AGENTS 三个表都位于 MySQL 中，ORDERS 表位于 SQL Server 2000 中。虚拟视图创建如下：

<?xml version="1.0" encoding="UTF-8"?>

```
<! DOCTYPE root SYSTEM "virtualview. dtd">
<root>
  <virtualview name="caps">
    <table>
      <tablename>customers</tablename>
      <dbname>caps_mysql</dbname>
      <schema/>
    </table>
    <table>
      <tablename>products</tablename>
      <dbname>caps_mysql</dbname>
      <schema/>
    </table>
    <table>
      <tablename>agents</tablename>
    <dbname>caps_mysql</dbname>
      <schema/>
    </table>
    <table>
      <tablename>orders</tablename>
    <dbname>caps_sqlserver</dbname>
    <schema>dbo</schema>
    </table>
  </virtualview>
</root>
```

针对名为"caps"的虚拟视图,输入查询语句 select c. cname, ordno, dollars from customers as c, orders as o where c. cid = o. cid and o. month = 'jan' and c. discnt <= 10。经 SQL 解析器解析,得到两条子查询语句,以及一个不能分解到子查询的条件:

子查询一:select customers. discnt, customers. cid, customers. cname from customers where customers. discnt <= 10;

子查询二:select orders. month, orders. cid, orders. dollars, orders. ordno from orders where orders. month = 'jan';

连接条件:customers. cid = orders. cid。

子查询和连接条件将被分别包装成 SQLCommad 和 SQLCondition 对象,前者交由数据适配器执行,返回结果集 ResultData;后者作为结果集合并器合并 ResultData 对象的连接条件。

2.5 缓存设计

数据库集成平台中,元数据的提取、SQL 语句解析以及结果集合并都是非常耗时的操作。设计一个合理的缓存机制将极大地提高查询效率[8]。在本系统中,设计了两块缓存区域。一个是元数据,在数据库第一次添加进来时,数据适配器将其元数据提取出来,在返回数据对象的同时,也将元数据写入 XML 文件。下次启动时,系统将首先搜寻缓存区,而不需要重新提取。元数据缓存默认由系统定时更新,用户也可以自行设置更新周期或手动更新。另一个是 SQL 解析与结果集。SQL 语句经过解析后,其分解出的子查询和连接条件将被缓存到 XML 文件中,同时最后的结果集也会被缓存。下次再分解同样的 SQL 语句时,将先查询缓存。用户也可以分别设置 SQL 解析与结果集缓存的有效时间。如果结果集的缓存时间超过有效期,则使用 SQL 解析缓存;如果 SQL 解析缓存也超过了有效期,则重新分解,否则只需重新执行查询操作即可。

3 结束语

设计了一个基于虚拟视图的异构数据库集成平台的体系结构,并在分析关系型数据库特点的基础上,采取了由用户自定义虚拟视图的方式。为了降低用户的使用难度,采用标准 SQL 作为查询语言,并详细讨论了 SQL 语句的分解算法。同时,为了节省查询时间,设计了一个基于 XML 的缓存机制,实现了查询优化。本系统下一步将重点考虑支持其他非关系型数据库的数据源。

参考文献:

[1] Florescu D, Levy A, Mendelzon A. Database techniques for the World-Wide Web: a survey[J]. ACM SIGMOD Record, 1998,27(3):59-74.

[2] 杨先娣,彭智勇,刘君强,等.信息集成研究综述[J].计算机科学,2006,33(7):55-80.

[3] 杨雪梅,董逸生,王永利,等.异构数据源集成中的模式映射技术[J].计算机科学,2006,33(7):87-91.

[4] 王亮明,李 东.基于 XML 的异构多源数据查询[J].计算机应用与软件,2007,24(12):107-109.

[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[J]. 计算机工程,2006,32(2):85-87.

[6] 刘志都,李自豪.异构空间数据系统查询分解算法的研究[J].计算机应用研究,2007,24(12):97-102.

[7] O'Neil P, O'Neil E. 数据库原理、编程与性能[M].第 2 版.北京:机械工业出版社,2002:18-19.

[8] 赵 洁,张 鹏,齐德昱.多数据库中间件中分布异构数据缓冲区系统的实现[J].计算机应用研究,2008,25(1):215-219.