异构数据源集成系统查询分解和优化的实现*

王宁[1] 王能斌[2]

[1](电力自动化研究院电网研究所 南京 210003)

[2](东南大学计算机科学与工程系 南京 210096)

E-mail: bbxyiqih@public1.ptt.js.cn

摘要 通用异构数据源集成系统需要集成包括WWW在内的各种数据源,有些数据源既无规则的模式结构,又无强有力的查询功能,给全局查询的分解和优化造成一定的困难.异构数据源集成系统Versatile一方面利用局部动态字典的模板操作构造集成系统全局动态字典,作为查询分解和优化的依据.一方面采用基于缓存和数据源能力的查询分解和优化策略,以便充分利用数据源的查询能力,简化包装器的设计,并取得较高的查询效率.

关键词 异构数据源,数据集成,动态字典,查询分解,查询优化,查询能力,缓存.

中图法分类号 TP311

随着计算机网络的普及和WWW的出现,人们的注意力逐渐由多库集成[1,2]转向多数据源集成.异构数据源集成系统除了集成具有规则结构(well structured)的数据之外,还需集成来自WWW等数据源的半结构化(semistructured)数据[3].这些数据源不仅数据模型不同,且查询能力各异,给查询的分解和优化带来了新的问题.首先,半结构化数据与传统数据库中数据的最大区别在于,它们不遵循某个固定的模式[4].在数据库系统中,模式一方面用于定义数据库的结构,是用户构造有意义的查询的依据,另一方面也是查询处理器进行查询优化的依据.如果缺少模式,这些工作将变得十分困难.其次,异构数据源集成系统不同于多库系统,由于集成范围加大,各种数据源的查询能力千差万别[5].异构数据源集成系统主要采用包装器[5]屏蔽各种数据源的差异.包装器的设计方案有两种:一种是利用包装器实现数据源未提供的查询能力.另一种是仅向各数据源发送它们均能完成的查询(文献[6]中称其为"least common denominator"),其余查询由全局查询处理器完成.以上两种方案或导致非常复杂的包装器,或导致全局查询处理器负担过重.再次,在多库系统中,全局结点并不真正存储数据,基于全局模式的查询被分解成子查询,发送到相应的局部源去执行.这种方法对多库系统是适用的,而异构数据源集成系统需要集成声音、图像等多媒体信息及WWW上的HTML文件,这些数据的检索速度慢,每次都从局部数据源检索,效率太低.近年来,有人提出利用实视图[7,8]的方法进行数据集成,采用这种方法可提高查询速度,但实视图维护的代价又太大.

Versatile是一个通用异构数据源集成系统的原型[9].它引入模板和动态字典的概念统一描述各种异构数据源的模式,不通过扫描数据库,而是利用局部动态字典的模板操作构造集成系统全局动态字典,为查询的分解和优化奠定了基础.在查询处理的过程中,Versatile利用缓存来存放用户对全局视图的查询结果,利用元数据仓库保存DBA(database administrator)提供的查询关系向量和各种数据源的查询能力向量.查询关系向量给出查询结果之间等同或包含关系的语义信息,查询处理器据此判断查询结果是否包含在缓存中,对于无法由缓存取得结果的查询,则根据数据源的能力,将查询分解到各源执行.Versatile采用基于缓存和数据源能力的查询分解和优化策略,不仅充分利用了数据源的查询能力,简化了包装器的设计,而且取得了较高的查询效率.

---

* 本文研究得到国家自然科学基金资助.作者**王宁**,女,1967年生,博士,工程师,主要研究领域为数据库和分布对象技术.**王能斌**,1929年生,教授,博士生导师,主要研究领域为数据库及信息系统.

本文通讯联系人:王宁,南京 210003,电力自动化研究院电网研究所

本文1998-04-20收到原稿,1999-01-20收到修改稿

本文第1节简单介绍Versatile系统的对象集成模型和代数.第2节给出全局动态字典的构造方法.第3节介绍基于缓存和数据源能力的查询分解和优化策略.最后是结束语.

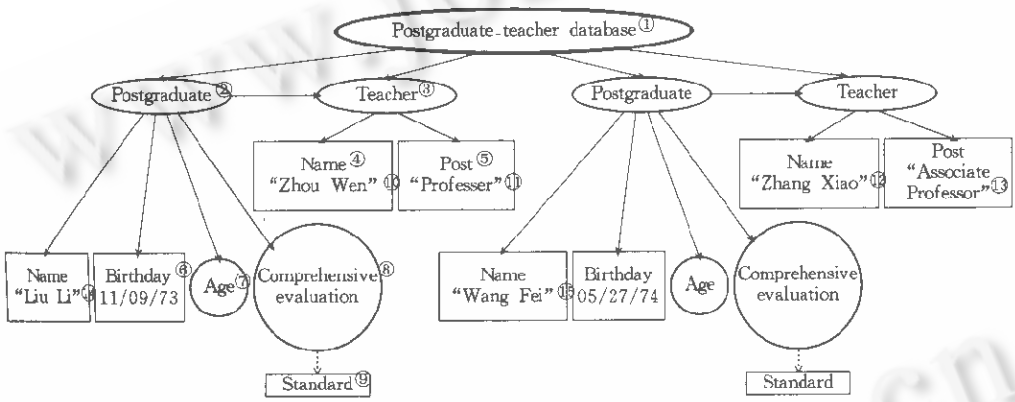
1 Versatile概述

Versatile是一个基于CORBA[10]的可扩展的异构数据源集成系统原型.在IONA公司的Orbix产品上,Versatile目前正对微软公司的SQL Server、面向对象数据库系统Versant、文件系统、超文本数据(即WWW中的数据)进行包装和集成.由于采用具有较强描述能力的OIM(object integration model)对象模型作为集成系统的公共数据模型,该系统不仅能集成上述数据源的数据,而且能集成随时插入的新数据源中的数据.

1.1 OIM对象模型

OIM对象模型[11]是Versatile系统的公共数据模型.一个OIM对象O是一个带根连通有向图,表示成$O(r,V,E)$.有向图中的每个结点表示对象,边表示对象与其成员之间的关系.根结点r是一个聚集对象,它是引用类型的,V是该聚集对象及其所有成员对象的集合,E是对象与其成员之间关系的集合.OIM对象模型将元数据附在数据上,便于集成来自各种异构数据源的异构数据,特别是自描述数据.

例1:一个反映研究生和教师情况的OIM对象O_1如图1所示.



①研究生_教师库,②研究生,③教师,④姓名,⑤职称,⑥生日,⑦年龄,⑧综合测评,⑨标准,⑩周文,⑪教授,⑫张晓,⑬副教授,⑭刘立,⑮王飞.

Fig.1 Structure of OIM object O_1 for postgraduate_teacher database
图1 研究生_教师库的OIM对象O_1的结构

为简单起见,图1中每个结点表示一个对象.椭圆形结点和矩形结点表示常规对象,其中,椭圆形结点是复杂对象,矩形结点是原子对象,加粗的椭圆形结点表示根对象.圆形结点表示方法对象.实有向边表示复杂对象与其成员之间的引用关系,虚有向边表示方法对象与其参数之间的关系.

O_1以研究生_教师库为根,它有4个成员:两个研究生对象和两个教师对象.姓名和职称对象是教师对象的成员.研究生对象的成员除姓名、生日、教师这3个常规对象之外,还包含年龄和综合测评两个方法对象.其中,年龄对象表示的方法不带参数,而综合测评对象表示的方法以标准对象作为其参数.

1.2 OIM对象代数

OIM对象模型从便于异构数据源的集成出发,提供一系列OIM对象操作的定义,它们分别是对象并、对象差、对象选择、对象投影、对象粘贴及对象切削操作,这些操作的总和称为OIM对象代数.OIM对象代数是Versatile系统全局查询语言OIQL(object integration and query language)的数学基础.为方便以后叙述,这里简单介绍OIM对象代数的几个基本操作,关于OIM对象代数的详细情况,请见文献[11].

(1) 对象并.其结果的亲子对象集是两个给定OIM对象O_1,O_2亲子对象集的并,表示成$O_1 \oplus O_2$.

(2) 对象差.其结果的亲子对象集是两个给定OIM对象O_1,O_2亲子对象集的差,表示成$O_1 \ominus O_2$.

(3) 对象选择.按照一定的条件f,在给定OIM对象O_1中选取根的若干亲子对象,表示成$\sigma[f](O_1)$.

(4) 对象投影. 在给定 OIM 对象 O_1 的所有亲子对象中,沿指定路径集$\{p_1,\ldots,p_k\}$选取从路径终点出发的子对象,表示成 $\Pi[p_1,\ldots,p_k](O_1)$.

(5) 对象粘贴. 规定被粘贴对象 O_1 为基本对象,按照一定的条件 f,在基本对象的某点 p_1 粘贴其他对象 O_2 的某点 p_2 的子对象,表示成 $O_1\otimes[p_1,p_2,f]O_2$.

(6) 对象切削. 在给定 OIM 对象 O_1 的所有亲子对象中,沿指定路径集$\{p_1,\ldots,p_k\}$去除从路径终点出发的子对象,表示成$\overline{\Pi}[p_1,\ldots,p_k](O_1)$.

2 动态字典的建立

在异构数据源集成系统中,通过包装为各种数据源提供标准界面,以实现系统的可扩展性,这一点已成共识. Versatile 系统也是利用包装层将各种数据源的异构数据转换成 OIM 对象,以适应互操作的需要. 为了帮助用户构造有意义的查询以及协助查询处理器进行查询分解和优化,各局部结点的包装层和全局结点都要有自己的数据目录,在 Versatile 系统中称为动态字典. 所谓动态,一方面是指有些数据源没有显式模式,局部包装层的模式信息并非用户预先指定,而是通过扫描数据库得到,如 Lore 中的数据导则(DataGuide)[4];另一方面是指全局结点的模式信息由局部结点的相应信息构造而成.

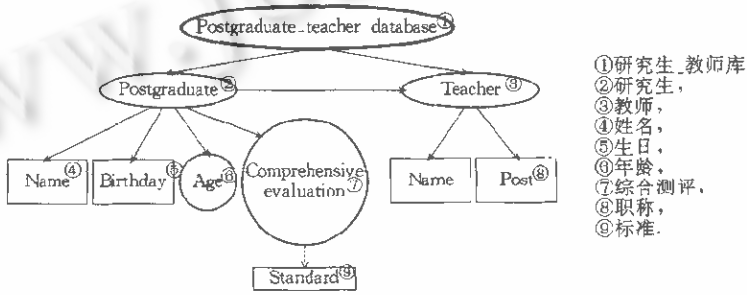


Fig. 2 Exact template of OIM object O_1 for postgraduate_teacher database
图2 研究生_教师库的OIM对象 O_1 的一致模板

Versatile 采用模板[12]$M(O)$表示 OIM 对象 O 的结构,模板本身也是 OIM 对象,在其所有从根出发的路径的集合中不存在同类路径*. 模板与 Lore 中的数据导则不同,它不仅能描述对象的结构特征,还能描述对象的行为特征. 一致模板 $EM(O)$是一种特定的模板,它能精确地反映 OIM 对象 O 的结构. 图 1 所示的 OIM 对象 O_1 的一致模板如图 2 所示. 然而,对于没有固定模式结构的 OIM 对象来说,一致模板或者难以确定,或者确定时所需的计算量太大(一般通过扫描数据库取得),因而一般用模板取代. 模板中有的对象,数据库中不一定存在. 但是,模板不会"屏蔽"数据库中的对象,也就是说,数据库中有的对象,模板中必有沿同类路径的同类对象. 因此,虽然模板可能与数据库中对象的准确结构不完全一致,但作为元数据使用仍然是合理的.

Versatile 系统的动态字典是模板的集合,有全局动态字典和局部动态字典之分,分别对应于 Versatile 的全局视图和局部视图. 在 Versatile 系统中,局部视图由局部数据源的 DBA 定义,它向系统提供可共享的 OIM 对象,某个数据源提供的可共享 OIM 对象的模板集合构成该数据源的局部动态字典. 全局视图是由局部视图通过 OIM 对象代数操作构造而成的,是可供终端用户访问的对象,全局 OIM 对象模板的集合构成 Versatile 的全局动态字典.

Versatile 是一个可扩展的异构数据源集成系统,参与集成的数据源是多种多样的. 有的具有显式的模式结构,如数据库系统,其模式结构可直接转化为模板;有的没有显式的模式结构,如文件系统,Versatile 系统提供模

* 两条路径属同类路径是指,它们经过的结点数相等,且除第 1 个结点外,其对应结点是同类结点. 同类结点一般是指两个结点具有相同的对象名和对象类型. 如果两个方法结点是同类结点,它们除了具有上述条件之外,参数之间还需存在一一对应的关系,且对应的参数结点均是同类结点.

式描述语言描述其结构,它的模板可从模式描述文件取得;还有的数据源包含自描述数据,如 WWW 上的 HTML 文件,其模板可用类似于 Lore 中的数据导则的获取方法,通过扫描数据库取得[4].

在异构数据源集成系统中,除各个数据源数据模式的描述外,全局结点集成数据的模式描述同样重要. Versatile 系统并非利用实视图进行数据集成,若像 Lore 那样,通过扫描数据库的方法取得全局数据的模式显然是不切实际的.

Versatile 系统的全局模板由局部模板通过模板操作构造而成. 设有 OIM 对象 O_1,O_2 及其模板 $M(O_1)$,$M(O_2)$,$P'_1=\{p'_1,p'_2,\ldots,p'_n\}$,$p'_1,p'_2,\ldots,p'_n,p_1,p_2$ 均是从根出发的路径. 模板操作共有 5 种,它们分别是:(1) $M(O_1)$和 $M(O_2)$的模板并,记作 $M(O_1)\oplus_m M(O_2)$;(2) $M(O_1)$的同类模板,记作$=M(O_1)$;(3) $M(O_1)$在 P'_1 上的模板投影,记作 $\Pi_m[p'_1,p'_2,\ldots,p'_n](M(O_1))$;(4) $M(O_1)$在 P'_1 上的模板切削,记作$\overline{\Pi}_m[p'_1,p'_2,\ldots,p'_n](M(O_1))$;(5) $M(O_1)$在被粘贴点 p_1 对于 $M(O_2)$在粘贴点 p_2 的模板粘贴,记作 $M(O_1)\otimes_m[p_1,p_2]M(O_2)$.

可以证明:$M(O_1)\oplus_m M(O_2)$是 O_1,O_2 对象并的模板. $=M(O_1)$既是 O_1,O_2 对象差的模板,也是 O_1 的对象选择的模板. $\Pi_m[p'_1,p'_2,\ldots,p'_n](M(O_1))$是 O_1 在 P'_1 的同类路径集 $TLP(P'_1,O_1)$上对象投影的模板. $\overline{\Pi}_m[p'_1,p'_2,\ldots,p'_n](M(O_1))$是 O_1 在 P'_1 的同类路径集 $TLP(P'_1,O_1)$上对象切削的模板. $M(O_1)\otimes_m[p_1,p_2]M(O_2)$是 O_1 在被粘贴点 $TLP(p_1,O_1)$与 O_2 在粘贴点 $TLP(p_2,O_2)$上对象粘贴的模板. 证明详见文献[12].

根据以上结论,OIM 对象操作的模板可由相应的模板操作构成,因此,利用局部模板之间的操作可以构造出全局 OIM 对象模板. Versatile 系统的全局动态字典就是利用局部动态字典的模板操作构造而成的.

3 全局查询的分解和优化策略

3.1 数据源查询能力的描述

Versatile 的查询处理器接受来自客户端的查询,将查询分解成若干子查询,交给各包装器,包装器将用公共语言表达的查询转换成相应数据源能接受的查询语句,再将数据源返回的结果转换成由公共模型表达的对象,以便集成来自各异构数据源的数据. 如果仍然采用传统分布式数据库系统或多库系统中的查询分解和优化方法,各种数据源的包装器必须实现 OIM 对象代数操作的全部功能,对一些查询能力较弱的数据源来说,支持 OIM 对象代数操作需要相当复杂的包装器.

为了简化包装器的设计,Versatile 系统由各种数据源描述其查询能力. 数据源的查询能力用查询能力向量 QCV(query capability vector)表示. 假设 $op_1,op_2,op_3,op_4,op_5,op_6$ 分别表示 OIM 对象代数的 6 种操作,QCV 是六维的,表示成$(c_1,c_2,c_3,c_4,c_5,c_6)$,其中 $c_k=1$ 或 $c_k=0,k=1,\ldots,6$. $c_k=1$ 表示数据源经包装后支持 op_k;$c_k=0$ 表示数据源经包装后不支持 op_k. 一个数据源及其包装器仅需支持其 QCV 描述的查询能力,而无需支持所有 OIM 的对象代数操作.

局部数据源在向 Versatile 系统注册时提供其查询能力向量 QCV,该向量描述数据源及其包装器对 OIM 对象代数操作的支持与否,QCV 存放于元数据仓库*中,查询处理器在查询分解时考虑各数据源的能力,仅向数据源发送其能支持的查询.

3.2 OIM 对象缓存

Versatile 需要集成多媒体信息及 WWW 上的 HTML 文件,这些数据的检索速度慢,每次从局部数据源检索的效率太低. 利用实视图进行数据集成虽然可提高查询速度,但维护的代价又太大.

Versatile 系统利用缓存提高全局查询的速度. 它将用户对全局视图的查询结果存放于缓存中,用 LRU 算法淘汰最近不常用的 OIM 对象. 由于缓存中仅保留全局视图的常用部分,与实视图相比,维护的工作量大大减少.

Versatile 系统利用全局视图进行数据集成,全局视图可由局部视图通过 OIM 对象代数的任何操作构造而

* Versatile 的元数据仓库管理数据集成时生成的全局视图定义、全局视图模板、各种数据源的查询能力向量以及安全性管理所需的授权等信息.

成.终端用户的每次查询仅针对某个全局视图,即仅使用 OIM 对象代数的单目操作,该限制符合实际情况,也为缓存的实现提供了方便.

为了判断用户的查询结果是否在缓存中,除保留过去的查询结果外,缓存还需要保留查询中用到的全局视图名、选择条件、投影(切削)路径,以下称这 3 项为缓存的入口.如果用户的查询与缓存的某个入口相符,查询结果可由缓存直接读取,这是最理想的情况.如果缓存某个入口相应的 OIM 对象 O 包含*用户查询的结果,则对 O 进行适当的对象选择、投影(切削)后可得查询结果,由于 O 从缓存中取得,整个查询的速度要比从各数据源取数再汇总快得多.事实上,如果缓存某个入口相应的 OIM 对象 O 包含于用户查询的结果,也可将 O 作为查询的部分结果输出.如果用户并不满足于部分结果,则将该查询分解送各数据源,以获取整个查询结果.当部分查询结果即可满足用户要求时,这样的处理方法其效率显然较高.

判断两个查询是否具有包含关系,必须判断它们的选择条件是否具有蕴含关系以及投影(切削)路径是否具有包含关系,算法比较复杂,可能会影响查询速度.Versatile 系统允许全局 DBA 提供查询关系向量 QRV(query relation vector),由 QRV 明确指出查询结果之间的等同或包含关系,以简化判断过程,提高整个系统的效率.QRV 是四维向量(vn,e_1,e_2,r).vn 表示全局视图名.e_1,e_2 均是针对 vn 的查询表达式,由投影(切削)路径集 P 和选择条件 f 两部分组成,表示对全局视图 vn 执行选择条件为 f 的对象选择操作以及投影(切削)路径集为 P 的对象投影(切削)操作,投影路径用[ ]括起,切削路径用![ ]括起,选择条件用( )括起.在 e_1 或 e_2 中,P 与 f 同时缺省表示取全局视图 vn 的所有的 OIM 对象;r 表示 e_1,e_2 之间的关系,e_1,e_2 之间可有"="和"⊇"两种关系.当 r 为"="时,对全局视图 vn 执行 e_1 表示操作的结果与执行 e_2 表示操作的结果是等同的 OIM 对象;当 r 为"⊇"时,对全局视图 vn 执行 e_1 表示操作的结果包含执行 e_2 表示操作的结果.

例 2:假设如图 1 所示的研究生_教师库是一个全局视图,对于以下两个查询:

查询 1. 取得导师名为"周文"的研究生姓名和年龄.

首先选择导师名为"周文"的研究生对象,然后投影取得其姓名及年龄,该查询用 OIM 对象代数操作表示成

$$\Pi[\text{研究生:姓名,研究生:年龄}()](\sigma[\text{研究生:教师:姓名}=\text{"周文"}](\text{研究生\_教师库})).$$

查询 2. 列出导师名为"周文"的研究生除生日外的其他情况.

首先选择导师名为"周文"的研究生对象,然后用切削操作去除生日情况,该查询用 OIM 对象代数操作表示成

$$\overline{\Pi}[\text{研究生:生日}](\sigma[\text{研究生:教师:姓名}=\text{"周文"}](\text{研究生\_教师库})).$$

顺便指出,Versatile 系统将根对象名作为相应的 OIM 对象名,所有子对象均由原始路径表达式表示.原始路径表达式描述从根对象到该表达式表示对象的路径,由一组对象名通过":"连接而成.OIM 对象模型支持方法,方法对象在原始路径表达式中表示成"方法名(参数$_1$,参数$_2$,...)"的形式.

显然,查询 2 的结果 OIM 对象包含查询 1 的结果 OIM 对象.它们之间的关系可用 QRV 表示成

(研究生 教师库,![研究生:生日](研究生:教师:姓名="周文"),

[研究生:姓名,研究生:年龄()](研究生:教师:姓名="周文"),⊇).

全局 DBA 提供的查询关系向量 QRV 存放于元数据仓库中,供查询处理使用.

缓存的使用改善了系统的性能,然而,当缓存中的 OIM 对象所对应的局部视图发生变化时,该 OIM 对象与数据源的数据之间就出现了不一致,必须刷新.Versatile 系统根据不同的应用需求分别采取定期刷新(periodical refresh)和按需刷新(by-demand refresh)两种方式.

(1) Versatile 定期检查缓存中 OIM 对象所涉及的局部视图是否改变,根据变化情况修改缓存中的内容.对那些要求快速响应的用户查询,可以从缓存直接读取数据,当然,缓存中内容不能反映两次刷新之间局部视图的改变情况.

(2) 对于那些要求准确响应的用户查询,定期刷新就显得不合适了.Versatile 系统对这种情况采取按需刷

---

* OIM 对象之间存在着等同和包含关系,这些关系的详细定义见文献[13],本文限于篇幅,不一一赘述.

新方式,首先检查缓存中OIM对象所涉及的局部视图是否改变,若没有改变,直接从缓存读取数据;若有改变,则将查询分解到各数据源,从中读取最新数据,并修改缓存的内容.

3.3 基于缓存和数据源能力的查询分解和优化策略

在Versatile系统中,客户端发出的查询经过查询处理器的分析程序进行词法和语法分析,得到一棵OIM对象查询树T,OIQL语句生成的OIM对象查询树类似于SQL语句生成的查询树,叶结点表示操作数(OIM对象),非叶结点表示操作(OIM对象代数操作)及其属性.假设该查询的操作数是全局视图v,选择条件为f,投影(切削)路径集为P,则有

(1) 若v,f,P与缓存的某个入口相符,查询结果可由缓存直接读取.

(2) 若(1)的条件不满足,但存在查询关系向量(vn,e_1,e_2,r),其中$vn=v$,$r=$"$=$",e_1(或e_2)与该查询相符,且v和e_2(或e_1)表示的选择条件、投影(切削)路径集与缓存的某个入口相符,则查询结果可由缓存直接读取.

(3) 若(1)、(2)的条件均不满足,但存在查询关系向量(vn,e_1,e_2,r),其中$vn=v$,$r=$"$\supseteq$",e_2与该查询相符,且v和e_1表示的选择条件、投影(切削)路径集与缓存的某个入口相符,则该入口相应的OIM对象O包含原查询结果.在这种情况下,可将查询树T的操作数(原为v)用O替换,得到一棵新查询树T_1,T_1与T执行的OIM对象代数操作虽然相同,但由于T_1的操作数来自缓存,执行T_1的速度显然比T要快得多.

(4) 若(1)、(2)、(3)的条件均不满足,但存在查询关系向量(vn,e_1,e_2,r),其中$vn=v$,$r=$"$\supseteq$",e_1与该查询相符,且v和e_2表示的选择条件、投影(切削)路径集与缓存的某个入口相符,则该入口相应的OIM对象O包含于原查询结果,即原查询的部分结果可由缓存直接读取.在这种情况下,从缓存读取O之后交给用户,并询问用户对部分结果的满意程度.若满意,则原查询完成,否则按下面(5)的方法将查询分解至各数据源执行.

(5) 若上述条件均不满足,则从元数据仓库中取出全局视图的定义(也是一棵OIM对象查询树,设为T'),用T'取代原查询树T中的叶结点(存放全局视图名),得到一棵新查询树T_2.对T_2作代数优化,代数优化所得查询树经后序遍历分解为若干子查询树.查询树在的分解时,不仅要考虑局部OIM对象所在的数据源,还要考虑该数据源的查询能力.各数据源支持的子查询树发往相应数据源执行,子查询结果在全局结点汇总.

4 结束语

一个通用异构数据源集成系统需要集成各种各样的数据源,有些数据源,例如WWW,既无规则的模式结构,又无强有力的查询功能,给全局查询的分解和优化造成一定的困难.Versatile系统引入模板和动态字典的概念统一描述各种异构数据源的模式,利用局部动态字典的模板操作构造集成系统全局动态字典,为全局查询的分解和优化奠定了基础.此外,Versatile利用缓存存放用户对全局视图的查询结果,利用元数据仓库保存DBA提供的查询关系向量QRV和各种数据源的查询能力向量QCV.在处理全局查询时,根据QRV判断查询结果是否包含在缓存中,凡结果包含在缓存中的,则尽量从缓存读取.若结果不在缓存中,或部分结果在缓存中而用户对这部分结果并不满足,则根据数据源的能力,将查询分解到各源执行.Versatile的查询处理器不仅能充分利用数据源的查询能力,而且能利用缓存提高全局查询的速度.

参考文献

1 Bright M W *et al*. A taxonomy and current issues in multidatabase systems. IEEE Computer, 1992,25(3):50~59

2 Sheth Amit P. Federated database systems for managing distributed, heterogeneous, and autonomous databases. ACM Computing Surveys, 1990,22(3):182~236

3 McHugh J, Abiteboul S, Goldman R *et al*. Lore: a database management system for semistructured data. SIGMOD Record, 1997,26(3):39~53

4 Roy Goldman, Jennifer Widom. DataGuides: enabling query formulation and optimization in semistructured databases. In: Dayal U, Gray P M D, Nishio S eds. Proceedings of the 23rd International Conference on Very Large Data Bases. San Francisco, CA: Morgan Kaufmann Publishers, Inc., 1997. 436~445

5 Mary Tork Roth, Peter Schwarz. Don't scrap it, wrap it! A wrapper architecture for legacy data sources. In: Dayal U,

Gray P M D, Nishio S eds. Proceedings of the 23rd International Conference on Very Large Data Bases. San Francisco, CA: Morgan Kaufmann Publishers, Inc., 1997. 266～275

6 Kevin Chen-Chuan Chang, Hector Garcia-Molina. Boolean query mapping across heterogeneous information sources. IEEE Transactions on Knowledge and Data Engineering, 1996,8(4):515～521

7 Gupta A, Mumick I, Subrahmaman V. Maintaining views incrementally. In: Buneman P, Jajodia Sushil eds, Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data. New York: Academic Press, 1993. 157～166

8 Zhuge Y *et al*. View maintenance in a warehousing environment. In: Widom J ed, Proceedings of 1995 ACM SIGMOD Conference. New York: Academic Press, 1995. 316～327

9 Wang N, Chen Y, Yu B Q, Wang N B. Versatile: A scaleable CORBA-based system for integrating distributed data. In: Department of Computer Science and Technology of Tsinghua University ed. Proceedings of the 1997 IEEE International Conference on Intelligent Processing Systems. Beijing: International Academic Publishers, 1997. 1589～1593

10 Randy Otte, Paul Patrik, Mark Roy. Understanding CORBA. Englewood Cliffs, NJ: Prentice Hall, Inc., 1996

11 Wang Ning, Xu Hong-bing, Wang Neng-bin. A data model and algebra for object integration based on a rooted connected directed graph. Journal of Software, 1998,9(12):894～898
(王宁,徐宏炳,王能斌.基于带根连通有向图的对象集成模型及代数.软件学报,1998,9(12):894～898)

12 Wang Ning, Xu Hong-bing, Wang Neng-bin. Construction of global dynamic dictionary in heterogeneous data integration system. Chinese Journal of Computers, 1999,22(1):103～107
(王宁,徐宏炳,王能斌.数据源集成系统中动态字典构造方法研究.计算机学报,1999,22(1):103～107)

13 Wang Ning, Xu Hong-bing, Wang nenig-bin. Construction of global dynamic dictionary in heterogeneous data integration system. Chinese Journal of Computers, 1999,22(1):31～38
(王宁,徐宏炳,王能斌.异构数据源集成系统中基于数据源能力的查询分解和优化策略,计算机学报,1999,22(1):31～38)

Query Decomposition and Optimization in Heterogeneous Data Integration System

WANG Ning[1] WANG Neng-bin[2]

[1](*Power System Control Corporation Nanjing Automation Research Institute Nanjing* 210003)

[2](*Department of Computer Science and Engineering Southeast University Nanjing* 210096)

Abstract A heterogeneous data integration system can integrate a broad range of data sources including WWW. Decomposition and optimization for query are very difficult because some data sources have neither regular schemata nor strong query capabilities. To help query decomposition and optimization, a dynamic dictionary is proposed as unified schemata for various heterogeneous data sources, and the way for the construction of a global dynamic dictionary by operating on dynamic dictionaries of local data sources is given. Furthermore, to simplify the design of wrappers for data sources with limited capabilities and reduce responding time for query, an approach, which can take advantage of all powers of various data sources and results of prior queries, is proposed for query decomposition and optimization.

Key words Heterogeneous data source, data integration, dynamic dictionary, query decomposition, query optimization, query capability, cache.