多数据库查询接口的设计与实现*

陈 微 刘 瑞 李昭原

(北京航空航天大学信息工程研究所 北京 100083)

摘要 MQI(multi-database query interface)是作者研制的、用 SQL 对多个 RDBMS(relational database management system)数据源进行综合查询的客户端工具. 该文主要讨论其逻辑结构与算法设计.

关键词 多数据库,综合查询.

中图法分类号 TP311

随着计算机技术的不断进步和应用领域的不断拓展,现代管理信息系统的数据构成呈现出多元和分布的趋势,即一个应用涉及到的数据不仅存在于不同的地点,还可能由不同的数据库管理系统(DBMS)进行管理,这就导致了多数据库系统的产生和发展. 与单一的 DBMS 相比,多数据库系统具有更好的开放性、可扩充性和灵活性. 它能够针对不同的领域、不同的部门、不同的需求,使用不同的数据库技术,从而能够提供更好的性能/价格比. 然而,多数据库系统也给信息系统的设计和实现带来了更大的复杂性. 其中,如何实现对多数据库的查询操作是一个亟待解决的问题. 与对单一 DBMS 的查询相比,对多数据库的查询操作必须隐藏各个 DBMS 之间的差异和它们在系统中的位置,使用户能够通过一个统一的查询界面,透明地访问所需的数据.

鉴于当前大多数多数据库系统均由关系数据库构成,本文提出和设计了一个适用于由多种 RDBMS(relational database management system)组成的多数据库综合查询接口软件 MQI(multi-database query interface). 它在不同的 RDBMS 之上提供了一个新的抽象层面,该抽象层面向下可映射到各种具体的 RDBMS,向上(对应用)则隐藏各 RDBMS 的差异,从而构成了一个虚拟的 RDBMS. MQI 以 SQL 作为查询语言,能够支持对分布在多个数据库系统中的数据的查询,具有较好的透明性、可扩充性和容错性.

1 MQI 的应用环境

目前,市场上支持多数据库系统的商品软件有 Ingres/Star 和 Sybase OmniSQL Gateway 等,但它们都是服务器一级的软件,对硬件要求高,安装和维护工作量大,且价格昂贵. 鉴于此,MQI 的设计原则是小巧和易于使用,它在基于客户/服务器的信息系统中的位置如图 1 所示. 这种结构有两个特点:(1) MQI 驻留在客户端,为一个客户机上的应用服务,避免产生新的瓶颈问题,从而降低了对硬件的要求. (2) MQI 位于负责网络事务的中间件之上,与具体网络无关.

2 MQI 的逻辑结构

MQI 的模式设计主要参考了 Amit P. Sheth 和 James A. Larson 在文献[1]中给出的描述 FDBS 的 5 级参考模型. 根据实际需要,我们略去了成员模式,并将外模式与联邦模式合二为一,如图 2 所示.

Component DBS:每种 DBMS 可看作一个 Component DBS. 在 MQI 中,Component DBS 的种类可以不一样,但必须是关系的. Local Schema:各成员数据库的概念模式. Export Schema:Local Schema 的子集. 针对不同的 MQI 用户,每个成员数据库可有不同的 Export Schema. External-Federated Schema:是综合了一个 MQI 用户所有的 Export Schema 得到的,它刻画了一个 MQI 用户所能利用的所有数据的结构.

通过这 3 层模式,MQI 实现了对多 RDBMS 的综合查询. 下面给出加入处理过程的系统结构图如图 3 所示.

---

* 本文研究得到国家航空基金和航空预研项目基金资助. 作者陈微,女,1970 年生,博士生,主要研究领域为数据库,软件工程. 刘瑞,1970 年生,讲师,主要研究领域为数据库理论,数据库应用开发工具. 李昭原,1938 年生,教授,主要研究领域为数据库工程,软件工程,管理工程.

本文通讯联系人:刘瑞,北京 100083,北京航空航天大学信息工程研究所

本文 1997-05-16 收到原稿,1997-06-23 收到修改稿

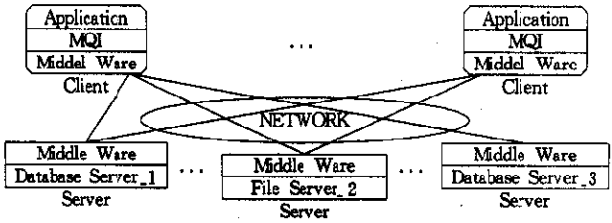

图1

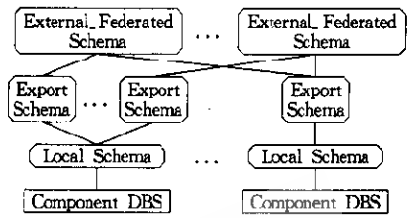

图2

Filter Processor:允许合法的命令和数据通过.它对命令作语法和存取控制检查,对数据作类型转换.

Constructing Processor:通过分割或复制,把一个任务分解为多个任务分别去完成,然后将每个任务产生的数据综合为一个数据集返回给调用者.它通过内部数据字典提供位置和数据源类型的透明性.它主要完成下面的功能:(1) 模式集成:将多个输出模式集成为一个外部_联邦模式.(2) 查询分解与优化:分解和优化建立在外部_联邦模式上的查询.

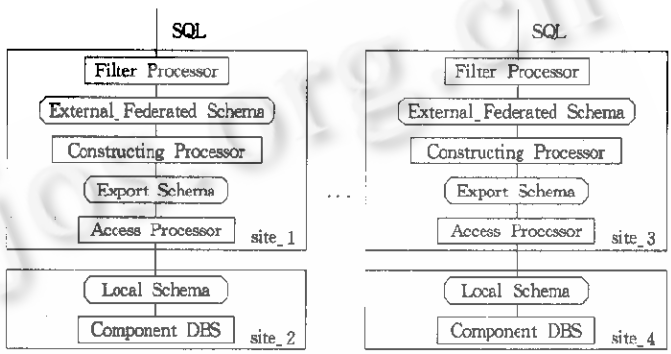

图3

Access Processor:接受命令,按命令要求从数据库中检索出相应的数据.Construction Processor是整个系统实现的关键部分,下一节将详细讨论查询任务分解的算法.

3　MQI的查询分解算法

(1) 求出查询的基本数据集合的SELECT表达式

关系代数和元组演算是SQL查询的数学基础,是证明各种分解算法正确性的手段;但关系代数却不一定能表达SQL表示的查询.这是因为SQL融进了很多关系代数所没有的特征,如分组、排序、聚集等.为便于算法的设计与证明,我们提出基本数据集合的概念:给定SELECT语句的基本数据集合是能用关系代数表达,且能产生SELECT的最终结果的最小关系.这里,最小是指,去掉任何一行或一列都不足以得到最终结果.每个基本数据集合总能由一个SELECT语句表达(因为能用关系代数表达的总能用SELECT表达),这个SELECT语句称为原SELECT语句的基本数据集合的SELECT表达.

查询分解的第1步就是去掉SELECT语句中那些不能由关系代数表达的特征,得到与之对应的基本数据集合的SELECT表达.在MQI支持的形如

```
SELECT select_list
FROM table_list
WHERE where_list GROUP BY group_list HAVING having_list
ORDER BY order_list
```

的SELECT语句中,不能由基本关系代数运算表达的特征有:列别名、表达式、聚集函数、分组、排序.下面逐步除去这些特征,得到基本数据集的SELECT表达:

①去掉select_list中的聚集函数,保留作为被去掉的聚集函数的参数的列名(如果和已有列不重复的话).

②去掉出现在select_list中的表达式并将表达式中的列名添到select_list中(如果和已有列不重复的话).

③去掉出现在select_list中的别名.

④将出现在where_list中,却不在select_list中的列加到select_list中.

⑤对group_list,having_list,order_list重复④中对where_list的操作,并去掉相应的子句.

(2) 化一般查询为用UNION连接的多个合取查询

合取查询是WHERE句中的条件只包含AND和NOT两种连接符,而且每个NOT作用域内部都没有AND的

查询. J. D. Ullman 在文献[2]中指出，合取查询是一类很自然的查询，它包括了关系代数"选投连"核心的部分. 实际中，大多数查询都属于合取查询. 合取查询的等价性问题是可判定的，所以，大多数系统在设计其查询分解优化的算法时，主要工作都在对合取查询的优化上. MQI 的分解优化策略也主要针对合取查询.

当某个查询不是合取查询时，我们将把它变为多个合取查询来处理. 根据德·摩根定理，每个布尔表达式都可以规范为合取式的析取式形式，即具有 A_1 OR ... OR A_n(其中 $A_i(1\leqslant i\leqslant n)$是一个合取式)，则任何查询的基本数据集合的 SELECT 表达都可表示为

SELECT select_list FROM table_list WHERE A_1 OR ... OR A_n

可以证明，它与下述语句等价.[3]

SELECT select_list FROM table_list WHERE A_1
UNION .....
UNION
SELECT select_list FROM table_list WHERE A_n

然后对每个已成为合取查询的 SELECT 语句分别处理，最后将其合并为一个关系.

(3) 分割每一个合取查询

分割合取查询的目的是把涉及到多个 DBMS 的数据的查询分解为一组能在单个 DBMS 上完成的子任务和综合这些子任务的结果的子任务.

①合取查询中"选投连"的可重组性

合取查询中"选投连"的可重组性是指 SELECT 语句的这样一种特性：设有合取查询

SELECT select_list FROM from_list WHERE where_list

其中，所有在 where_list 中出现的列都出现在 select_list 中. 现将 from_list 分为 r 组，相应地，把属于 from_list_i 的列分到 select_list_i 组，把只与 select_list_i 中的列相关的 where_list 中的条件都分在 where_list_i 中，将 where_list 中不属于任何 where_list_i 中的条件分到 where_list_0 中，忽略属性在元组中出现的先后次序，原 SELECT 语句改写为

SELECT select_list_1,select_list_2,...,select_list_r
FROM from_list_1,from_list_2,...,from_list_r
WHERE where_list_1 and ... and where_list_r and where_list_0

其中 where_list_i 可能为空，则有如下结论.

SELECT select_list_i FROM from_list_i WHERE where_list_i 结果存于 T_1 {1}

...

SELECT select_list_r FROM from_list_r WHERE where_list_r 结果存于 T_r {r}

SELECT select_list FROM $T_1,T_2,...,T_r$ WHERE where_list_0 {$r+1$}

{1}～{r+1}步执行的结果与原 SELECT 语句结果相同.[3]

②查询分解中的分割原则

合取查询中"选投连"的可重组性解决了子任务的执行结果是否与原来的查询结果等价的问题. 分组时，应遵循两个原则：①每个分组只能包含存在于同一数据源上的表/视图；②每个分组的查询的连接图必须是连通的.

连接图用于描述查询中涉及到的表/视图之间的连接关系. 连接图的顶点是 table_list 中的每个表/视图，如果两个表/视图有连接关系，则在对应二顶点间画一条边. 语句

SELECT $C.attrc, A.attra$
FROM A,B,C,D,E,F
WHERE $A.ab=B.ba$ and $B.bc=C.cb$ and $C.ce=E.ec$ and $D.de=E.ed$ and $D.df=F.fd$ and $F.fe=E.ef$

的连接图如图 4(a)所示.

假设 A,B,D,E,F 在数据源 α 中，C 在数据源 β 中，如果以是否在同一个数据源上为分组的唯一原则，无疑，A,B,D,E,F 在同一组，而 C 在另一组. 在查询被分解的同时，原查询图也被分为两个子图(图 4(b)、(c)). 图 4(b)是在数据源 α 上执行的查询的连接图. 由于这个图不是连通的，所以必然产生笛卡尔积，而笛卡尔积能使结果数据成几何速度增长，这是查询中最糟糕的情况. 在一般情况下，应避免使用包含笛卡尔积(查询连接图为不连通图)的查询. 在例子中，用户给出的查询连接图是连通的，只是由于使用了不合适的分割方法而被分割成不连通的子图，从而使查询效率

大受影响.如果将 A,B,D,E,F 进一步划分为 A,B 和 D,E,F 两个组,则可避免上述问题.即使在用户给出的查询连接图不连通的情况下,也要尽量推迟笛卡尔积的操作,以减少网络通讯量和对其后操作的影响.通过以上分析,我们给出分组的第2条原则.

③分割步骤

我们用一个四元组 $\langle C,T,S,D\rangle$ 来描述每一个分割出的组,其中 C 是属于该组的需要输出的列的集合,T 是该组的表/视图的集合,S 是该组选取和连接条件的集合,D 是临时结果库名.对一个合取查询分割的结果,就是产生若干这样的四元组,再由四元组组装 SELECT 语句.不失一般性,设合取查询有如下形式.

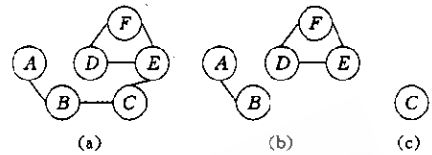

图4

$$\text{SELECT } a_1,\ldots,a_n \text{ FROM } t_1,\ldots,t_m \text{ WHERE } B_1 \text{ and } \ldots \text{ and } B_r$$

分割过程如下:

① 创建 $P_0(C,T,S,D)$,其中 $C=\{a_1,\ldots,a_n\}$,创建 $P_i(C,T,S,D)(1\leqslant i\leqslant m)$,其中 $P_i.T=\{T_i\}$,$P_i.C=\{a_1,\ldots,a_n$ 中属于 t_i 的列$\}$,$P_i.S=\{B_1,\ldots,B_r$ 中只包含了 $P_i.C$ 中元素的布尔表达式$\}$.

②对于每个不属于任何 P_i 的 B_k,设 A,B 是该 B_k 涉及到的两个表/视图,若 A,B 在同一数据源上,设 A 所在的分割是 P_i,B 所在的分割是 P_j,且 $i<j$,则将 P_j 中的内容添加到 P_i 中去,即令 $P_i.C=P_i.C\cup P_j.C$;$P_i.T=P_i.T\cup P_j.T$;$P_i.S=P_i.S\cup P_j.S\cup\{B_k\}$,然后使 P_j 无效,否则,将 B_k 加到 P_0 中,即 $P_0.S=P_0.S\cup\{B_k\}$.

③过程②的结果使有的 P_i 无效了.对每一个有效的 P_i 分配一个唯一的表名 TN_i,即使 $P_i.D=TN_i$,并使

$$P_0.T=\bigcup_{1\leqslant i\leqslant n,P_i\text{有效}}\{TN_i\}.$$

④为每个有效的 P_i 生成相应的 SQL 语句,即

$$\text{SELECT } P_i.C \text{ FROM } P_i.T \text{ WHERE } P_i.S$$

⑤还原在①中被去掉的特征

第3步所产生的 SQL 序列只是查出了基本数据集合,要得到用户想要的结果,还要加上被去掉的如列别名、GROUP、ORDER 子句等特征,具体方法是

SELECT select_list FROM $P_0.D$ GROUP By group_list

HAVING having_list ORDER BY order_list

其中 select_list,group_list,having_list,order_list 和用户所给出的相同.

4 小　结

MQI 是国家航空基金和国防科工委基金的资助项目——"开放-分布-异构和基于客户/服务器结构的数据库应用快速开发平台"的子项目.它支持对多数据库的综合查询,既可嵌入在"开发平台"中使用,也可独立使用.目前,我们已经在 Windows 环境下实现了 MQI 综合查询系统.今后,我们将作更进一步的理论探索和实践,使 MQI 的算法更加完善,从而提高效率和支持更多的 SQL 特征.

参考文献

1 Sheth Amit P, LARSON James A. Federated database system for managing distributed, heterogeneous, and autonomous databases. ACM Computing Surveys, 1990,22(3):183~236

2 Jeffrey D Ullman 著,张作民译.数据库系统原理.北京:国防工业出版社,1984
(Jeffery D Ullman. Principles of Database Management System. Rockville, MD: Computer Science Press, 1980)

3 陈微.多数据库查询接口的设计与实现[硕士论文].北京航空航天大学,1995
(Chen Wei. Design and implementation of a multi-database query interface [M. S. Thesis]. Beijing University of Aeronautics and Astronautics, 1995)

Design and Implementation of a Multi-database Query Interface

CHEN Wei　LIU Rui　LI Zhao-yuan

(*Information Engineering Institute　Beijing University of Aeronautics and Astronautics　Beijing*　100083)

Abstract　MQI (multi-database query interface) is a client tool which the authors developed for querying with SQL on several different RDBMS (relational database management system) data sources. Its logical architecture and algorithm design are mainly discussed in this paper.

Key words　Multi-database, synthetic query.