第35卷 第20期 计 算 机 工 程 2009年10月
Vol.35 No.20 Computer Engineering October 2009
·软件技术与数据库· 文章编号：1000—3428(2009)20—0066—02 文献标识码：A 中图分类号：TP311.12

分布式环境下数据共享中的多表查询转换算法

邬建锋，彭宇行
(国防科技大学计算机学院并行与分布处理国家重点实验室，长沙 410073)

摘 要：不同数据源之间的数据表示方法不同且结构上存在冲突，导致分布式环境下的异构数据源查询成为一个难点。提出一种分布式环境下的多表查询转换算法，该算法在查询转换过程中对源查询进行分解、在目标数据源上进行目标查询的转换和重构，解决数据共享中的多表查询转换问题。实验结果证明了该算法的有效性。

关键词：数据共享；查询转换；多表

Multi-table Query Conversion Algorithm in Data Sharing Under Distributed Environment

WU Jian-feng, PENG Yu-xing
(National Key Laboratory for Parallel & Distributed Processing, College of Computer, National University of Defense Technology, Changsha 410073)

【Abstract】 The conflict of data structure and heterogeneous representation between different data sources makes it becomes a difficult problem to query heterogeneous data sources under distributed environment. This paper proposes a multi-table query translation algorithm under distributed environment. This algorithm decomposes the source query during query transition process, transforms and reconstructs target query on the target data source, and solves the problem of multi-table query transition problem in data sharing. Experimental results show that the algorithm is effective.

【Key words】 data sharing; query conversion; multi-table

1 概述

数据库技术的广泛应用为企业和科研机构积累了大量以不同形式存储、依赖不同数据库管理系统的数据。在提高资源利用率、降低二次开发成本等目的驱动下，人们对相关领域不同机构之间共享现有数据资源的需求与日俱增。但由于此类数据库系统是在不同应用背景下，基于不同数据模型独立设计的，因此它们之间可能存在数据结构、表示方法等多方面的差异和冲突，限制了数据共享。

异构数据库间的数据共享系统按各数据共享端之间耦合程度的不同分为紧耦合型和松耦合型2种。紧耦合型系统将各局部数据源通过逻辑上的全局模式联合起来，用户对全局数据库的查询被转换并重构为符合各局部数据源子查询的形式[1]。但其可扩展性较差，数据源模式的修改、新节点的加入会对全局模式产生影响。在松耦合型系统中，不需要存在唯一的全局模式，各节点分别维护自身局部模式和节点之间的映射规则，对异地数据的请求查询只要在源和目标2个节点间进行[2]。在分布式环境下的数据共享中，由于对共享的灵活性、多样性、动态性需求，松耦合型系统具有更大优势。本文算法基于松耦合型系统模型设计。

在局部数据源之间共享数据时，需要指明表、属性之间的语义映射规则，在此基础上获取异地相关数据的过程称为查询转换[3]。由于已有数据源之间存在的各种异构，特别是表结构之间的异构，因此数据共享需要在语义映射关系的基础上对多表之间的查询转换问题进行处理，进而解决异地数据的获取问题。

本文提出一种分布式环境下的多表查询转换算法(Query Translation among Multi-tables, QTM)，该算法对源查询进行分解后，基于映射规则在目标模式上进行查询转换和重构，在保证正确性和有效性的前提下解决了多表之间的查询转换问题。

2 QTM算法设计思想

若B, B’分别表示不同数据库系统中的表，F, F’分别表示表B, B’中的属性，F, F’记录了相同语义的数据。异构数据库系统之间共享数据，即在B表中查询F数据可以延续到B’表中查询F’数据。在本文中，以映射规则 m: B.F→B’.F’表示不同数据库表中相同语义的数据，映射规则的集合用 M 表示。

数据库表间的延续查询问题，即多表之间的查询转换问题。先对查询语句 Q^S 进行分解，得到待转换的表名和投影项集合，然后按映射规则转换成目标数据库上对应的表名和投影项，最后在异地的目标数据库上生成相同语义的查询语句 Q^T。

一般地，一个查询 Q^S 通常定义为

SELECT select_list FROM table_reference1 WHERE where_clause1
[UNION(INTERSECT/MINUS)
SELECT select_list FROM table_reference2 WHERE where_clause2][4]

其中，select_list表示查询 Q^S 中所要选取的投影项，即“<投影项>或<投影项><运算符><投影项>组成的集合”,<投影项>表示“<表名/别名>.<属性名>”；table_reference表示查询 Q^S 涉及的引用的表，即“<表名> [<别名>]组成的集合”；

基金项目：国家“863”计划基金资助项目(2006AA01Z332)
作者简介：邬建锋(1982－)，男，硕士研究生，主研方向：异构数据库，数据共享；彭宇行，研究员、博士
收稿日期：2008-09-01 **E-mail：**j.f_wu@yahoo.com.cn

where_clause 表示查询 Q^S 中的条件子句，即“ <投影项 1><连接符><投影项 2/常量>组成的集合 ”。

对查询 Q^S 在目标模式上做查询的延续，先要分解并获取查询 Q^S 的 select_list, table_reference 和 where_clause 中对源模式中表名和投影项引用的集合。具体而言，分解和获取按以下规则得到：

(1) 集合操作的分解。如果查询中存在 UNION, INTERSECT 和 MINUS 等操作，则先去掉这些操作，将 Q^S 分解为几个子查询的形式再分别处理。

(2)处理 table_reference。分解为不同的数据库表名。

(3)处理 select_list。按照“ <表名/别名>.<属性名> ”的形式分解投影项。

(4)处理 where_clause。分解为条件子句，并进一步将每个条件子句分解为<连接符>和<连接符>两端的项(按“ [<投影项 1>，<投影项 2>/常量] ”的形式)。

对源查询 Q^S 进行以上分解后，在映射规则 M 的指导下，将分解得到的源模式的表名和属性名分别转换为与之对应的目标模式的表名和属性名，从而重构得到异地模式上的新的 select_list, table_reference 和 where_clause(分别标记为 new_select_list, new_table_reference 和 new_where_clause)，形成最终的目标查询语句 Q^T。

例如，Q^S：SELECT $A.s_1$, $B.s_4$, $B.s_5$*$C.s_8$ FROM A, B, C WHERE ($A.s_1$=$B.s_3$) AND ($B.s_4$=$C.s_6$) AND ($C.s_6$='value')。

处理 table_reference：{A, B, C}；

处理 select_list：{$A.s_1$, $B.s_4$, $B.s_5$, $C.s_8$, B,s_3}；

处理 where_clause：{[$A.s_1$, $B.s_3$], [$C.s_6$, 'value']}, {[=] AND [=]}。

与 Q^S 相关的映射规则为

$M=\{m_i \mid i=1,2,\cdots,5\}$

其中，m_1:$A.s_1 \rightarrow E.t_1$；m_2:$B.s_3 \rightarrow F.t_3$；m_3:$B.s_4 \rightarrow E.t_2$；m_4:$B.s_5$*$C.s_8 \rightarrow F.t_7$；m_5:$C.s_6 \rightarrow F.t_6$；

上述 2 个数据库模式结构如图 1 所示。

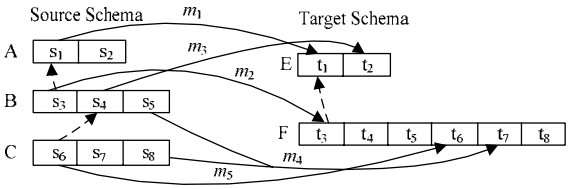

图 1　2 个数据库模式结构

生成的目标如下：

new_select_list：{ $E.t_1$, $E.t_2$, $F.t_7$}

new_table_reference：{E, F}

new_where_clause：{($E.t_1$=$F.T_3$) AND ($F.t_6$='value')}

最后得到 Q^T：SELECT $E.t_1$, $E.t_2$, $F.t_7$ FROM E, F WHERE ($E.t_1$=$F.T_3$) AND ($F.t_6$='value')。

3　QTM 算法流程

QTM 算法流程如下：

输入 源请求查询 Q^S，映射规则集合 M

输出 目标模式上的查询 Q^T

If 查询 Q^S 中存在集合操作 Then

将 Q^S 从集合操作符处分解为单独的查询 Q；

Endif

从 Q^S 中获取 select_list, table_reference 和 where_clause 列表中的项；

For table_reference 中的每一个表名

分解为不同数据库的表名，并进行缓存；

Endfor

For select_list 中的每一个投影项

按照“ <表名/别名>.<属性名> ”的形式分解投影项，并进行缓存；

Endfor

将 where_clause 列表分解为条件子句的集合；

For 条件子句集合中的每一个查询条件子句

按照“ [<投影项 1>，<投影项 2>/常量] ”形式从连接符处分解，并对前述的对象组和对应的连接符进行缓存；

Endfor

获取查询 Q^S 相关映射规则集合 M；

对上述 3 个列表中的元素在目标模式上替换；

生成目标模式上的 new_table_reference 列表；

生成目标模式上的 new_select_list 列表；

生成目标模式上的 new_where_clause 列表；

最终形成目标查询 Q^T。

4　实验结果

在如下 2 个同为记录设备资料的实例数据库上对本文共享算法进行实验。2 个实例数据库参与实验的表结构分别如图 2 和图 3 所示(分别记为 A 和 B)，其中，虚线表示数据库表之间的外键关联关系。

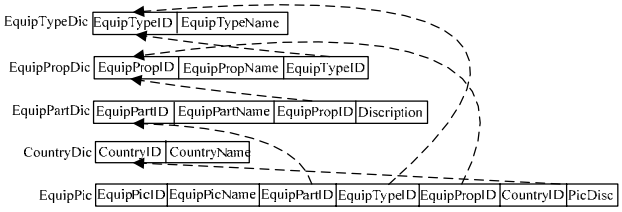

图 2　数据库 A 的表结构

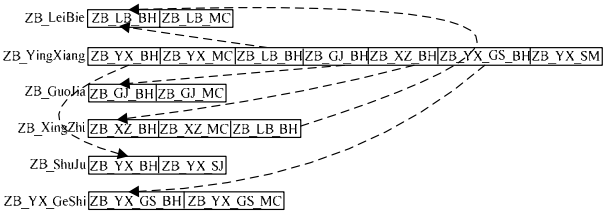

图 3　数据库 B 的表结构

在实验中，先根据 A 和 B 中各表和属性的语义关系建立 2 个数据库模式之间的语义映射关系规则，例如，EquipPic.EquipPicName→ZB_YingXiang.ZB_YX_MC,EquipPic.EquipData→ZB_ShuJu.ZB_YX_SJ 等，以此来屏蔽由于 A 和 B 之间命名方式和表结构不同带来的差异，为两者之间的数据共享做好准备。

对以下 2 对查询进行实验，其中，SQL1 和 SQL2 由 A 共享端产生，SQL3 和 SQL4 由 B 共享端产生，SQL1 和 SQL3 针对单表进行查询转换，SQL2 和 SQL4 针对多表进行，实验查询语句如下：

SQL1: SELECT EquipPic.EquipPicName, EquipPic.PicDisc FROM EquipPic WHERE (EquipPic. EquipPicName='F-117A');

SQL2: SELECT EquipPic.EquipPicName, EquipTypeDic.EquipTypeName, CountryDic.CountryName FROM EquipPic, EquipTypeDic, CountryDic WHERE (EquipTypeDic.EquipTypeID=EquipPic.EquipTypeID) AND (CountryDic.Country ID='01');

SQL3: SELECT ZB_XingZhi.ZB_XZ_BH, ZB_XingZhi. ZB_

(下转第 70 页)