系。Agent 的一个组织关联图如图5所示。

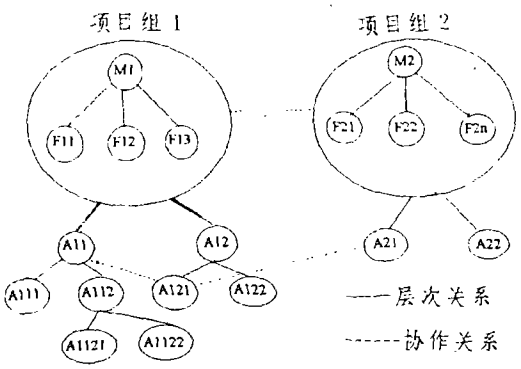

图5 Agent 的组织关联图

4.2 问题求解组织结构

问题求解组织结构是 agent 为了协同完成一个任务而形成的一种动态组织结构。各 agent 之间的关系是动态变化的，一但具体任务完成，这种关系就自行消失。当新的任务到来后，会形成由其它 agent 构成的新的问题求解结构。例如，对于合同网式的问题求解结构[4]，只有当某个 agent 接受任务时，它才有可能成为 manager，其它 agent 根据自身能力因素决定自己是否成为该 manager 的 contractor，从而构成 manager 与 contractor 的关系。任务完成，这种关系也随之消失。

结束语 本文分析了人类智能行为的活动过程及信息处理的四个层次，并在此基础上提出了软件 agent 的基本结构及多 agent 系统的组织结构。目的是通过对人类智能活动的模拟，构筑具有自主、协作及智能的软件 agent，使计算机更好地为人类服务。我们的下一步任务是开发相应的构筑环境。

（下转第48页）

（上接第81页）

中进一步发展了这个思想用于面向对象多数据库系统（主要是增加了导航查询的局部处理代价的获取能力）。很显然，使用这种方法必须知道每个具体局部系统所支持的存取方法，[8]中认为这明显地违背了局部自治性。因此，在[8]中提出了一组方法，其中之一是查询取样技术。其中心思想是对于局部场地的查询根据查询的特点、操作数的基数、可用的索引信息，将局部系统的特点进行分类。对于每一类查询设计一个取样查询以获得这一类查询的参考代价参数。使用这种方法的关键是查询的分类技术。目前该项研究的局部系统全部为关系数据库。

4.3 探索策略

对于一个给定的查询，其全局执行计划不是唯一的，而且往往可以有大量的执行计划可以选择。用代价模型采用穷举法分别地评价每一个执行计划，从中选择一个最优的执行计划将极大地恶化查询执行效率。因此需要一些探索策略，来尽可能减少枚举的次数。由于异构性，各局部场地的处理速度、语言的表达能力相差很大，因此这样的探索策略是很难设计的。

目前关于这方面的研究非常少。[7]中列举了一些在设计这样的探索策略时应考虑的一些问题和可选的策略，包括：(1)贪婪地分配投影、选择和利用半连接策略；(2)在有大量的连接次序可以考虑时，采用带参数的爬山式启发策略；(3)过滤不能在局部场地执行的操作（如连接）。其中策略1，2是传统分布数据库系统探索策略的延续，策略3只是异构分布环境下应考虑的最明显的策略。而对于异构分布环境下导航查询应采取什么样的探索策略成为今后应考虑的重要问题。

5 局部查询处理

在局部处理步骤执行的单场地查询必须经局部优化来选择有效的存取路径；经优化的单场地查询被转化为局部数据库管理系统的数据语言。

目前关于查询转换的研究较多，而关于局部优化的研究较少。其原因是局部优化也面临一个与全局优化同样的问题，即无法取得需要的全部优化参数。局部优化是多数据库系统的设计者而不是局部数据库管理员解决的问题，有关的研究可参见[12]。

总结 由于异构性和自治性，多数库系统中的查询处理是十分复杂的，但目前关于这个问题的研究还比较少，而关于引入面向对象技术的多数据库系统中的查询处理的研究就更少。尽管已有一些研究探讨了查询修改、查询优化、局部查询处理这些问题的解决方案，但对于每一个问题都没有一个理想的解决方案，因此在国家863资助项目“基于 CORBA 的面向对象的多数据库集成平台系统 SCOPE/CIMS”中，关于这些问题的解决策略是我们正在进行的研究内容之一。（参考文献共14篇略）

计算机科学1998 Vol. 25№. 2

多数据库系统中的查询处理*)

Query Processing in a Multidatabase System

石祥滨　　张　斌　于　戈　郑怀远

（辽宁大学计算机系　沈阳110036）（东北大学计算机系　沈阳110006）

摘　要　Because of heterogeneity and autonomy, the query processing in a multidatabase is very complicated. Some problems need be resolved, mainly including query modification, query optimization and local query processing. The paper attempts indicate the future research direction about these problems through analyzinkg existing researches.

关键词　Multidtatabase system, Query processing, Object-orientation

1　前言

在多数据库系统中，为实现局部系统的互操作，通常需要建立局部系统的集成视图。多数据库系统的用户针对集成视图表达全局查询请求后，多数据库系统的查询处理器经多级转换和处理后将全局查询请求等价地转换为优化的针对局部模式的一组局部子查询，由局部系统取得的部分查询结果经组装后取得最终的查询结果。

尽管多数库系统中的查询处理与传统的分布数据库系统中的查询处理有相似之处，但多数据库系统中的查询处理面临以下新的挑战：

(1)由于成员系统的数据在数据模型和语义上的异构性，为了消除这些差异需建立成员系统的集成视图。在查询处理阶段根据集成视图有效地实现数据集成成为一个非常复杂的问题。

(2)由于局部自治性，成员系统不可能暴露其全部信息。在执行一个全局查询时，因为缺乏局部信息将使得全局查询处理器难以选择一个理想的优化策略来执行一个全局查询。此外，由于成员系统在数据表达能力上的差异，也将使得全局查询优化更加复杂。

(3)参加联邦的局部系统可能有模式(如关系数据库系统)，也可能无模式(如文件系统)，模式的多样性将使得局部查询处理更加繁琐。

(4)面向对象公共数据模型和面向对象查询语言的引入使得多数据库系统的设计者在实现全局查询处理器时必须很好地解决异构分布环境下的导航查询、方法调用等问题。

现有的关于多数据库系统的研究[1-5]主要是针对多数据库系统的系统结构和模式集成的，而关于查询处理的研究却相对少一些。有的多数据库原型系统如[3-5]中采用了简单的、低效的查询处理策略。因此，有效地实现多数据库系统中的查询处理是目前未能很好地解决的异构信息集成的问题之一。

根据已有的研究[6-10]多数据库系统中的查询处理需解决的主要问题有：查询修改、查询优化、局部查询处理。针对这些问题，本文通过对现有研究的分析以确定对这些问题今后的研究方向。

2　多数据库系统中的基本问题

本文这部分讨论影响多数库系统中查询处理的几个最基本的问题，包括公共数据模型和查询语言、模式结构、模式集成策略和数据集成策略。

2.1　公共数据模型和查询语言

由于面向对象的数据模型具有丰富的语义，现有的多数据库系统多采用各种形式的对象模型。随着ODMG-93成为面向对象数据库的事实上的工业标准，最近几年出现的多数据库原型系统如[4,10]中基本上都采用了ODMG-93标准规定的对象模型和查询语言OQL。目前商品化的面向对象数据库系统如O2, ONTOS等大都是遵循这个标准的产品。当采用这些产品的成员系统参加联邦时，从查询处

*)863/CIMS主题资助项目，项目号：863—511—9611—0006。石祥滨　在职博士研究生，副教授，主要研究领域为面向对象数据库，信息集成技术，软件工程。张斌　博士，副教授。于戈　博士，教授。郑怀远　教授，博士导师。

理的角度，由于全局查询语言的语言特征与部分成员系统的语言特征基本相同，因此方便了全局查询向局部查询的转换。现有的很多关于面向对象查询处理的研究，包括：对象演算、对象代数及查询优化等可以作为采用了对象模型和对象查询语言的多数据库系统查询处理的基础，但这些问题需要根据异构分布和集成的特点进行适当地扩充和修正。

2.2 模式结构

在多数库系统中，所采用的模式结构决定了查询转换的层次和查询处理的流程。现有的多数据库系统大都采用了与[1]中的五级模式结构相同或类似的模式结构，包括：局部模式；成员模式(COMPONENT SCHEMA)；输出模式(EXPORT SCHEMA)；全局模式或联邦模式，即集成模式或集成视图；外模式。本文的讨论只涉及前4层模式。

针对这样的模式结构查询处理的流程大致如下：在全局查询场地，针对集成视图的查询首先经查询修改处理被转换为针对输出模式的查询；经查询优化处理针对输出模式的查询被等价分解为可优化执行的针对成员模式的一组子查询，每个子查询只涉及单一的场地；在每个成员系统的局部场地，每个针对成员模式的子查询被转换为针对局部模式的查询，局部查询的结果在全局查询处理场地完成组装后形成最终的查询结果。

2.3 模式集成

模式集成的主要问题有：冲突的解决、不同模式的类之间语义关系的确定和建立、集成模式中实例的导出方式。尽管现有的多数据库系统解决这些问题的方法不同，但基本上都使用合并和概括、特化、聚合语义关系来建立和组织集成模式。在集成模式中，既存在与输出模式中的类有直接对应关系的类，称为基类，又有在集成过程中由基类产生的新类，称为虚类。下面通过一个简单的例子来说明模式集成的问题，这个例子中只涉及概括语义并且与集成视图具体的描述方式无关。

假定两个待集成的类为教师 teacher{ss #, name, salary}，教学管理人员 staff{ss #, name, salary}，类 teacher 和 staff 的实例存在重叠，即教师可以兼作教学管理人员，在这种情况下，教师可以得到两份工资，并且名字属性的值以 teacher 中对象的属性值为准(即 teacher 中对象的名字属性值更可靠)。为简化讨论，我们略去了它们之间可能存在的其它冲突。在集成模式中可通过下面的集成视图 V1 以概括语义定义这两个类的超类(虚类)employee {ss #, name, salary}。

generalize (teacher, staff, employee, ss # = ss #)ss #:chooseany(ss #$_1$, ss #$_2$),

name: choosebiased (name$_1$, name$_2$), salary. sum (salary$_1$, salary$_2$)……V_1

其中 ss # = ss # 为 teacher 和 staff 中重叠对象的合并条件，chooseany、choosebiased、sum 为 employee 的属性值的导出函数，这样的函数，在[7]中称为聚集(Aggregate)函数，在[11]中称为判定(Decision)函数，本文称为属性值导出函数。事实上，属性值导出函数可以是任意的函数(方法)，但是为了使查询处理过程便于控制，在[7,11]中定义了各种常用的属性值导出函数。

研究表明：不同模式中存在的表示相同现实世界实体的有冲突的对象(等价对象)是使查询处理复杂化的一个重要因素。

2.4 数据集成策略和方法

通常有两种数据集成策略可以选择：一是在查询之前进行数据集成；二是在查询的过程中进行数据集成。策略1通常称为集成视图的实例化(materializing)，即在用户进行全局查询之前，根据集成视图的定义，完成数据的集成。在这种情况下，多数据库系统查询处理问题就转化通常的查询处理问题，其难点是实例化视图的维护问题以保证全局数据与局部数据的一致性。

策略2是将用户的全局查询与集成视图的定义进行组合，经一系列的查询处理(查询修改、查询优化、局部查询转换)来确保以最小的代价取得用户查询所需数据的集成。

由于多数据库查询处理，特别是优化的复杂性，一些系统采用了混合的数据集成策略。其主要特点是在用户发布全局查询以后，多数据库系统的查询处理器根据用户查询所涉及的虚类，有选择地将难以优化的虚类在查询优化之前进行实例化。这个问题见后面有关的讨论。

数据集成的另外一个问题是数据集成的方法问题。目前广泛采用的是外连接方法，但也有采用其它方法的，如文[12]中采用基于语义标识符的数据集成方法。对于与下面的讨论相关的概括操作，如采用外连接的方法，对于集成视图 V1可表达如下：

πss #:chooseany(ss #, ss #1), name:choosebiased (name1, name2), salary: sum (salary1, salary2)

(teacher outjoinss #1=ss #2 staff)

3 查询修改

查询修改的主要任务就是将针对集成视图的查询转换为针对输出模式的查询。在某种意义上，查询修改可以认为是模式集成过程的逆过程。在传统的分布数据库系统由于不存在异构性，只涉及数据的分片问题，这个问题的解决不是很复杂的。

在多数库系统中，对于一个给定的查询Q的查询修改过程通常按以下的步骤进行：(1)将所有虚类递归地替换为其在集成视图中的定义；(2)根据查询Q中的特点将Q转换为一系列子查询的并，同时消除不能为查询Q提供答案的或冗余的子查询；(3)对于每个子查询中的属性替换为在同构过程中引入的属性值导出函数。后面的查询优化处理将分别地优化每个子查询。这三个步骤中最难处理的是步骤2，所以本文不具体讨论步骤1，3。从步骤2可以看出，查询修改的目标之一是根据集成语义使回答全局用户所需查询最小化，因此在一定的意义上，查询修改具有语义优化的作用。最近有关查询处理的研究[12,13]将步骤2要解决的问题作为查询优化处理的一部分。

为了描述查询修改的过程，我们首先对类teacher和staff的实例做如下划分：teacher中的全部实例，简记O(teacher)；staff中的全部实例，简记为O(staff)，teacher中独有的实例，简记为EO(teacher)；staff中独有的实例，简记为EO(staff)，teacher和staff中的重叠实例，简记为OO(teacher，staff)。

假设有一个针对集成视图V1的简单的查询Q：πnameσsalary＞5000employee

从逻辑上讲employee的实例集是teacher和staff中的实例集的并，因而一种错误的做法是将查询Q直接修改为以下两个针对O(teacher)和O(staff)的子查询的并：Q1：πname σsalary＞5000teacher　　Q2：πss＃，nameσsalary＞5000staff

如果teacher和staff中的实例集不存在重叠，则以上查询修改是正确的。但是在实例集存在重叠时以上查询修改则是错误的。从语义上看，上述查询修改存在两个错误：(1)修改后的两个子查询丢失了相同对象的判定条件ss＃＝ss＃；(2)基于属性值导出函数的选择(salary＞5000)将产生错误结果。

下面讨论综合了[5,6,12]正确的查询修改方法。为了计算Q中的name属性(基于choosebiased函数)要求的划分为O(teacher)，OO(teacher，staff)；为了计算Q中的salary属性(基于sum函数)要求的划分为EO(teacher)，EO(staff)，OO(teacher)，因为针对划分EO(teacher)，EO(staff)，OO(teacher，staff)的三个子查询的结果包含了另外两个划分的子查询的结果，所以最后将Q修改为三个子查询的并。

下面我们通过另外一个例子进一步说明查询修改的问题。假定两个待集成的类为employeel{ss＃，name}，employee2{ss＃，name，salary}，两个类的实例可能重叠，在集成模式中应通过合并定义这两个类的合并类(虚类)employee{ss＃，name，salary}。从逻辑上讲employee的属性为两个待集成类的属性集的并，实例集为两上待集成类的实例集的并。employee可定义为：

merge (teacher，staff，employee，ss＃＝ss＃)ss＃：
　　chooseany(ss＃1，ss＃2)，
name：chooseany (name1，name2)，salary. salary2…
　　V_2

同样假定为V2的查询是前面的Q，根据Q中的属性要求的划分为EO(employee1)，EO(employee2)，OO(employee1，employee2)，但是由于EO(employee1)中无salary属性，因此针对EO(employee1)的子查询不会产生任何查询结果，因此最后确定回答Q的两个子查询为针对EO(employee2)的查询和针对OO(employeel，employee2)的查询。

从前面的讨论我们可以看到这样的问题：(1)对象的重叠使查询修改很复杂；(2)在全局查询涉及多个虚类并且每个虚类都由多个类(前面的讨论只有两个类)形成时，将产生大量的子查询(当一个虚类由n个类形成时，将产生2^n-1个子查询)，每个子查询都可能包含连接操作，分别优化每个子查询，可能产生无效的查询计划。

为减少这两个问题所带来的复杂性和系统性能的恶化，一种合理的解决策略是采取前面介绍的混合实例化方法。对于前面的集成视图V1(实例集存在重叠的情况)和全局查询Q，[3]中采取的方法是将全局查询所涉及的虚类employee在查询优化之前替换为前面在数据集成部分所讨论的外连接形式，将teacher和staff的全部实例直接送往全局场地，在全局场地通过外连接计算employee的结果，即完整地实例化employee，然后计算查询Q；而[7]提出了另外的一种处理方法，对employee的实例化采取特殊的优化策略(见后面优化策略部分的说

明)。

现有的有关查询修改的研究只能处理简单的语言特性。用像OQL这样的面向对象查询语言发布的全局查询中,可能出现路径表达式、各种聚集类型和SFW结构的正交嵌套,这对查询修改算法提出了新的挑战。

4 查询优化

多数据库系统中全局优化的目标是根据修改后的查询产生一个高效的全局执行计划。在全局执行计划中包括单个场地的局部处理步骤、数据在场地间移动的通讯步骤、组合局部场地执行结果的后处理步骤以及这些步骤的执行次序。

全局优化需要解决的问题有:(1)优化什么类型的查询;(2)采取什么样的优化策略;(3)代价模型的选择;(4)采取的探索策略。事实上,问题1在查询修改时也是一个需要考虑的问题。目前在讨论各种类型的数据库系统的优化策略时一般都是优化合取查询,所以我们主要讨论后三个问题。

4.1 优化策略

在传统的分布数据库系统中通过一些优化策略减少数据在不同场地之间数据移动的数量,常用的策略有:(1)在执行跨场地的连接之前首先执行单场地的投影和选择;(2)分配在集合并上的投影、选择和连接;(3)使用半连接减少连接运算的数据移动量。在多数据库系统中减少不同场地之间数据移动的数量同样应该是系统的设计者追求的优化目标之一。因此,在大多数系统如[10]中沿用了前面的优化策略,而在[7]中根据外连接和属性值导出函数的特点对这些策略进行了扩充。

在第二节的查询修改部分已经提到查询修改可能会产生大量的子查询,分别优化每个子查询,可能产生无效的查询计划。为此[7]中提出了一种半外连接优化策略。对于前面的集成视图V1和查询Q,查询修改后保留外连接定义如下:

πname:choosebiased(name1,name2)

σsum (salary1+salary2)>5000(teacher outjoinss#1=ss#2 staff)

首先在teacher所在场地运用半连接策略 teacher semijoin ss#1=ss#2 staff 同时计算EO(teacher),O(teacher)—EO(teacher),然后计算以下两个子查询的并

Q1:πname:name1 σsalary1>5000 EO(teacher)

Q2: πname: choosebiased (name1, name2) σsum (salary1 + salary2) > 5000 ((O (teacher)-EO (teacher))outjoinss#1=ss#2 staff)

[7]中讨论的重点是在涉及外连接和属性值导出函数时的优化策略,实际上是概括操作的优化策略,是基于表的优化策略。其缺陷是:(1)无法满足优化其它集成操作的实例化需求;(2)无法满足导航查询(全局对象标识的处理、复杂对象的组装)的优化需求;(3)无法满足优化多级集成视图的需求。据我们所知,目前还没有关于这些问题的全面研究。

4.2 代价模型

一个代价模型是一组数学公式的集合,用于计算一个执行计划的代价。查询优化器需要利用代价模型在所有可选的执行计划中选择效率最高的一个执行计划。在传统的分布数据系统中代价模型的重要参数是数据移动的代价,而忽略了局部系统的处理代价,这是因为局部系统往往具有等同或接近的数据处理能力。但是在多数据库系统中,由于异构性局部系统的数据处理能力相差很大,因此局部系统的处理代价成为代价模型的重要参数。

然而由于局部自治性,局部系统不可能暴露全部全局层所需要的代价参数,如基数(cardinality)可选性(selectivity),对象尺寸(object size),存取方法等,因而获取局部系统的处理代价是一项难度很大的任务。从理论上讲,取得精确的局部系统的处理代价是不可能的,也是没有必要的。因此,大多数系统特别是大规模信息集成系统如[12-13]中并没有将这个问题作为一个重要的问题加以考虑,这些系统优化处理的重点是基于语义的优化问题。以下我们简要地介绍有关局部代价模型的研究。

在[7]中采取了一种比较简单的局部代价计算模型。这种模型将局部代价定义为将要在局部场地执行的操作(投影,选择,连接,半连接等)的函数,这个函数的一个重要参数是依赖于局部场地的加权函数。但要获得能比较精确反映局部处理代价的加权函数是十分困难的。在[9]中提出了另外一种方法,称为校准方法(calibrating method)。这种方法的中心思想是在每个局部场地构造一个特定的局部校准数据库,这个校准数据库包含各种具体的数据库系统(如ORACLE)所支持的存取方法包括索引、聚族(clcuster)。对于校准数据库设计一组称为取样查询(Sample query)的一组典型查询以取得局部系统对于各种代数操作(如连接)的参考处理代价。[9]中提出的方法是针对关系型多数据数据库系统的,[10]

(下转第28页)