计算机科学2002Vol. 29№. 3

一种适合多数据库系统的查询表示方法

An Approach of Query Representation for Query Processing in Multidatabase Systems

娄勤俭　李瑞轩　卢正鼎

（华中科技大学计算机科学与技术学院　武汉430074）

Abstract　A multidatabase system gives users a global schema. A user must use global query language to query the multidatabase system. So, query transformation from global query to local queries must be needed and middle queries must be used during this transformation. This paper presents an approach of query representation for query processing in multidatabase systems and gives the properties of equivalent transformation of the query. An algebra foundation of middle class operations is also introduced to achieve the algebra optimization during the course of transformation from global query to middle queries.

Keywords　Query representation, Query processing, Middle query, Multidatabase systems

1　引言

随着分布计算和网络技术的不断发展，传统的数据库技术已越来越不能满足数据共享和互操作的需要。同时，已有的数据库系统又不可能全部丢弃，因而研制能同时访问和处理来自多个数据库中数据的多数据库系统已成为必然趋势。多数据库系统是解决已存的、异构的、分布的多个局部数据库系统之间数据共享和集成的问题。由于多数据库系统具有异构性、分布性和局部自治性的特点，使得多数据库查询处理与传统数据库查询处理有很大的不同。

多数据库系统呈现给用户的是全局模式，用户使用全局查询语言提交对多数据库的查询，而所需的数据又必须从各局部数据库获得，所以必须将全局查询转换成与局部数据库对应的局部查询。在全局查询转换为局部查询的过程中，需要经过中间查询的转换。本文使用查询树来表示多数据库系统中的查询，给出了查询的等价转换性质，并扩充了用于中间类操作的代数基础，以在全局查询转换为中间查询的过程中实现查询的代数优化。

2　多数据库系统中的类模式

2.1　类模式

多数据库系统的模式结构决定了查询处理的流程。现有的多数据库系统大多采用四级模式结构或类似的模式结构，它包括以下四种模式：局部模式、输出模式、全局模式和外模式。这里用类来定义模式中的所有信息，类的概念类似于抽象数据类型。在多数据库中，通常有很多相似的对象，对每个对象单独进行定义是很浪费的，因此可以将相似的对象分组形成一个类，类中的每个对象称为类的一个实例，一个类中的所有对象共享一个接口定义。输出模式和全局模式中的类与普通对象类一样，由一个接口和一组实例（或对象）组成，所不同的是，它们并不直接存储其对象，而是由其他类的对象导出自己的对象。

多数据库系统全局模式中包含一组全局类，这里给出与全局类模式有关的两个定义。

定义1　多数据库系统的类模式是一个多元组C(U, D, INFO, Q, M, IS-A, IS-PART-OF)，其中，C是类名；U是组成C的有限属性集；D是U中属性的值域；INFO是类C的对象所响应的消息的集合；Q是类C的对象所满足的限定条件集；M是类C的模式映射信息的集合；IS-A是类C继承的父类的集合；IS-PART-OF是类C所包含的类的集合。

定义2　一个类是相应于类模式C(U, D, INFO, Q, M, IS-A, IS-PART-OF)按模式映射M组织起来的从属性集U到值域D上所有满足条件Q的元素的集合，其中每个元素称为类C的对象。每个类有主键$K \subseteq U$。

全局类模式描述了全局类的结构及语义约束，它可以按一定的条件Q转换成中间类模式。类是类模式在某一时刻的当前值。为了讨论方便，我们把类模式简化成C (U, Q, M)，有时也用C(U, Q, M)或类名C表示类。

2.2　全局类、中间类和局部类

在多数据库系统中，一个类C (U, Q, M)可分为三种：全局类、中间类和局部类。

定义3　全局类(GC)是指在多数据库系统中对全局用户可见的类。在多数据库系统中，全局类是虚拟的，它并不具有实际的对象，而是由若干中间类和局部类按模式映射M组成的，其中Q＝真(True)，$M \neq \varnothing$。

定义4　中间类(EC)是指全局类在某个输出模式中的映射，其中$M \neq \varnothing$，中间类也是虚拟的。全局类与中间类通过中间映射进行联系，它们是1:n的联系。

定义5　局部类(LC)是指中间类映射到某个局部数据库上的基本类，其中$M = \varnothing$。中间类与局部类通过局部映射进行联系，它们是1:1或1:n的联系。

基于上述三种类，多数据库系统中有相应的三种数据库：①全局数据库：由多数据库中所有全局类组成的数据库，即多数据库，记为MDB；②中间数据库：由所有参与多数据库的中间类组成的数据库，记为EDB；③局部数据库：由所有参与多数据库的局部类组成的数据库，记为LDB。

2.3　模式映射

多数据库系统呈现给用户的是一组全局类，这组全局类实际上是由若干个输出模式中的中间类所组成，而中间类又是由若干个局部类组成的，那么，在这些模式之间必定存在一种映射机制将这些类维系起来，这就是模式映射。

定义6　类C(U, Q, M)的模式映射M是全局类与中间类、中间类与局部类之间联系的集合。

模式映射描述了多数据库中全局类的对象最终是如何从

局部数据库中获取数据的。它又可分为中间映射和局部映射。

定义7 中间映射 EM 是全局类与中间类之间联系的集合。

由定义7有：EM(MDB)=EDB。

定义8 局部映射 LM 是中间类与局部类之间联系的集合。

由定义8有：LM(EDB)=LDB。

3 查询表示及等价转换性质

3.1 多数据库查询处理

多数据库系统呈现给用户的是全局模式，用户只能在全局类上完成查询，而其实际数据又必须从各个局部数据库中获得，在全局查询和局部查询之间还需要经过中间查询的转换。按2.2节的概念，给出多数据库查询对应在三种数据库上的三种查询：①全局查询(GQ)：是用户对多数据库(MDB)提交的查询；②中间查询(EQ)：是全局查询对应于中间数据库(EDB)上的查询；③局部查询(LQ)：是中间查询对应于局部数据库(LDB)上的查询。这三种查询之间有一定的联系，这种联系依赖于中间映射和局部映射定义。可用下述定理进行描述。

定理1 对于任一全局查询 GQ，相应的中间查询为 $GQ\cdot EM^{-1}$，相应的局部查询为 $GQ\cdot EM^{-1}\cdot LM^{-1}$。

证明：由2.3节中间映射定义，有 $MDB=EM^{-1}(EDB)$，所以，有 $GQ(MDB)=GQ(EM^{-1}(EDB))=GQ\cdot EM^{-1}(EDB)$；同样由局部映射定义，有 $EDB=LM^{-1}(LDB)$，因为 $EM(MDB)=EDB$，$LM(EDB)=LDB$，所以有 $LDB=LM\cdot EM(MDB)$。同理，由 $MDB=EM^{-1}(EDB)$，$EDB=LM^{-1}(LDB)$，所以有 $MDB=EM^{-1}\cdot LM^{-1}(LDB)$，因而，有 $GQ(MDB)=GQ(EM^{-1}\cdot LM^{-1}(LDB))=GQ\cdot EM^{-1}\cdot LM^{-1}(LDB)$，证毕。

在多数据库系统中，对全局查询的处理一般存在两次转换：全局查询到中间查询的转换和中间查询到局部查询的转换。图1给出了多数据库系统中的查询转换。查询转换有时也叫查询分解。从图中我们可以给出多数据库查询处理的定义。

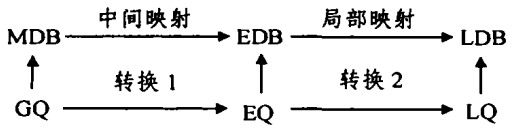

图1 多数据库系统中的查询转换

定义9 多数据库系统的查询处理 Q 是一算法：算法的输入是全局查询 GQ，算法的输出是相应的局部查询 LQ，算法的功能是将全局查询按照每个全局类的模式结构转换成一个最优的局部查询。

3.2 查询表达式

多数据库用户使用全局查询语言来表达全局查询，但要得到查询结果，必须对数据库中的类进行具体操作。这里使用对象查询语言 OQL 作为多数据库的全局查询语言，使用对象代数作为查询的内部表示方法。OQL 的语法结构基本取自于 SQL，只不过 SQL 主要是以集合为基础的，OQL 则提供了更为丰富的数据集，如集合、表或包等。对象代数几乎包含了所有的关系代数操作，并提供了一些对其他数据集的操作。目前有许多关于对象模式下代数形式的建议，这里主要针对集合操作对在多数据库查询中用到的代数基础进行扩充，将要用到的类的操作符包括，并(∪)、交(∩)、差(−)、笛卡尔积(×)、选择(σ)、投影(Π)、连接($\underset{ij}{\infty}$)、自然连接(∞)、半连接(⋉)和外连接(*=*)。下面用一个例子来进行说明。

例1 假设对全局类 employee 有如下 OQL 查询表达式：

SELECT ename，dno FROM employee WHERE dno = '101'； (1)

其相应的代数表达式为：

$$\Pi_{ename,dno}(\sigma_{dno='101'}\,employee) \quad (2)$$

或表示为

$$\sigma_{dno='101'}(\Pi_{ename,dno}\,employee) \quad (3)$$

式(1)、(2)、(3)表达了相同的查询语义，但式(2)和(3)表达了不同的操作次序。由此可得出：不同的表达式可表示出相同的结果；代数表达式可表达一定的操作次序。我们利用这个特性来讨论查询转换，并给出等价性定义。

定义10 如果两个查询表达式 E1和 E2的查询结果是相同的，则称它们是等价的，记为 E1=E2。

利用这个等价定义，可以得到一组非常有用的代数等价变换规则。

3.3 查询树

为了将查询表达式转换成类的操作系列，这里使用查询树来表示查询的内部结构。

定义11 查询树是一棵树 T=(V,E)，其中，(1)V 是节点集，每个非叶节点是类操作符，叶节点是类名；(2)E 是边集，两节点有边(V_1,V_2)，当且仅当 V_2是 V_1的操作分量。

例2 现有查询 Q1："查询在北京地区所属部门工作的职员的工号"，使用 OQL 语句表达的查询 Q1如下：

SELECT emp_id FROM employee e，department d

WHERE e.dno=d.dno AND d.address = 'Beijing'；

其相应的代数表达式为：

$$E1=\Pi_{emp\_id}\sigma_{address='Beijing'}(employee \underset{dno=dno}{\infty} department)$$

其相应的查询树如图2所示。

更一般地，如查询表达式 $E=\Pi_A(C_1\infty\sigma_P(C_2))\cup\Pi_A(C_1\infty C_2\infty C_3)$，其查询树如图3所示。

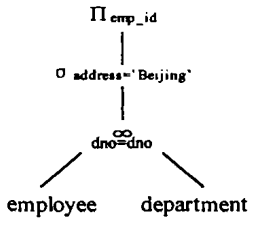

图2 E1的查询树

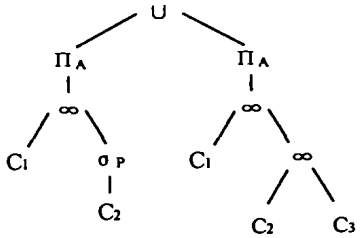

图3 E 的查询树

3.4 等价变换规则

为了方便，这里将3.2节给出的代数操作按操作数的个数分为一元操作(用 U 表示)和二元操作(用 B 表示)。一元操作主要有 σ 和 Π，其余为二元操作，由此可以得到以下等价变换规则。

1. 单个操作的变换规则

$C\times\varnothing\equiv\varnothing$ $C\cup C\equiv C$ $C\cup\varnothing\equiv C$ $C\cap C\equiv C$ $C\cap\varnothing\equiv\varnothing$

$\varnothing-C\equiv\varnothing$ $C-C\equiv\varnothing$ $C-\varnothing\equiv C$ $C\infty C\equiv C$ $C\infty\varnothing\equiv\varnothing$

$C\ltimes\varnothing\equiv\varnothing$ $\varnothing\ltimes C\equiv\varnothing$ $\sigma_P(\varnothing)\equiv\varnothing$ $\Pi_A(\varnothing)\equiv\varnothing$

2. 多个操作的变换规则 设有多个类 C,C_1,C_2,C_3，在一定条件下有如下规则：

①一元操作交换律：$U_1(U_2(C))\equiv U_2(U_1(C))$

②二元操作交换律：$(C_1)B(C_2)\equiv(C_2)B(C_1)$

③二元操作结合律：$(C_1)B((C_2)B(C_3))\equiv((C_1)B(C_2))B(C_3)$

④一元操作幂等律：$U(C)\equiv U_1U_2(C)$，其中 $U\equiv U_1U_2$

⑤一元操作对二元操作的分配律：$U((C_1)B(C_2))\Rightarrow U(C_1)BU(C_2)$

⑥一元操作的因式分解律：$U(C_1)BU(C_2)\Rightarrow U((C_1)B(C_2))$

使用这些规则可以改变查询树中操作的次序，以对查询树进行优化。

4 多数据库查询中代数基础的扩充

4.1 扩充的中间类代数规则

在多数据库系统中，查询转换中第一次转换是将对全局类的查询转换为对中间类的查询，所以，需要进一步讨论中间类的性质。由2.2节可知，中间类是对全局类进行 σ、Π、∞等操作而得到的。从对类的操作而言，中间类是带有一定限定条件的对象类。中间类的限定条件一般是谓词或属性集。定理1指出，有一个全局查询则必有相应的对中间类的查询和对局部类的查询。其中，对中间类的查询就是对有限定条件的类的操作。显然，在处理对象类以外还要对限定条件进行操作。

这里主要讨论可求值的限定条件，并给出中间类的简化表示：$[C:Q]$，其中 Q 是中间类应满足的谓词条件。对$[C:Q]$的操作是集合代数的一种扩充，其中使用中间类作为操作数。将集合代数操作作用到中间类上有如下扩充规则：

规则1 $\sigma_P[C:Q_C]\Rightarrow[\sigma_P(C):P\wedge Q_C]$

这一规则表示对一个全局类 C 进行选择操作（谓词为 Q_C）得到的中间类再做选择操作（谓词为 P），相当于对全局类做了一次选择操作，其谓词为 $P\wedge Q_C$，即谓词具有合取性，表示 P 在 Q_C 所选定的对象中的限定。

规则2 $\Pi_A[C:Q_C]\Rightarrow[\Pi_A(C):Q_C]$

对中间类投影出某些属性（A），即使计算谓词条件的属性不在 A 中，所得到的中间类的谓词不会改变，仍为 Q_C。

规则3 $[C:Q_C]\times[S:Q_S]\Rightarrow[(C)\times(S):Q_C\wedge Q_S]$

两个中间类的笛卡尔积同样有谓词合取性。

规则4 $[C:Q_C]-[S:Q_S]\Rightarrow[(C)-(S):Q_C]$

两个中间类的差操作是不对称的。

规则5 $[C:Q_C]\cup[S:Q_S]\Rightarrow[(C)\cup(S):Q_C\vee Q_S]$

两个中间类的并操作，其谓词具有析取性。

规则6 $[C:Q_C]\cap[S:Q_S]\Rightarrow[(C)\cap(S):Q_C\wedge Q_S]$

两个中间类的交操作，其谓词具有合取性。两个中间类的交操作可由规则4即差操作推导出来。

规则7 $[C:Q_C]\underset{J}{\infty}[S:Q_S]\Rightarrow[(C)\underset{J}{\infty}(S):Q_C\wedge Q_S\wedge J]$

两个中间类的连接操作也具有合取性，这可由规则3和规则1导出。

规则8 $[C:Q_C]\underset{J}{\propto}[S:Q_S]\Rightarrow[(C)\underset{J}{\propto}(S):Q_C\wedge Q_S\wedge J]$

两个中间类的半连接操作是投影与连接操作的导出操作，半连接操作也具有谓词合取性。

根据以上八个对中间类的操作规则，可以得到扩充的代数表达式转换的等价性质。

定义12 当两个中间类的基础类是等价的，且其限定条件都表示了相同的真值函数（即对同一对象用两个限定条件时，能得到相同的真值），则称这两个中间类是等价的。

由此，可以得到如下的用于中间类的命题。

命题1 所有集合代数具有的等价转换性质同样适用于中间类。

证明从略。

4.2 利用谓词合取性质进行查询优化

在上面讨论的中间类变换规则的选择和连接中有谓词合取性质，如 $Q_C\wedge P$，$Q_C\wedge Q_S\wedge J$，这种合取性本身可能引起一些矛盾。例如一个全局查询的某个选择操作可能在某中间类上并无与其对应的属性，那么该选择操作在查询转换过程中将为空。即当中间类的谓词合取时具有矛盾的限定条件时，实际上将是一种空操作。这种性质称为谓词合取可能为空。它对查询转换很有用，可以根据中间类所具有的内涵性质，利用其操作可能产生一些表达式为空的情况以简化查询的执行。

当然，要想利用谓词合取为空的性质，则要求在模式映射中考虑操作数谓词的正确性。3.4节讨论的对集合的单个操作变换规则中的空操作均适用于谓词合取性有矛盾的情况，这种空操作虽然作用于查询执行时，但在查询转换时就应先考虑进去。

（下转第136页）

第三届 Web 时代信息管理（WAIM'2002）国际会议将于2002年8月11-13日在北京举行

第三届 Web 时代信息管理国际会议（WAIM'02）将于2002年8月11日至13日在北京召开。该会议由中国计算机学会数据库专业委员会和 ACM SIGMOD 主办，中国人民大学和清华大学协办。会议论文集由德国 Springer 出版社作为《Lecture Notes in Computer Science》发表，优秀论文将推荐到 JCST 杂志上发表。会议得到国家自然科学基金委的支持。WAIM 国际会议旨在促进中国数据库界与国际上的交流，向世界介绍中国最新的研究成果，并使国内的研究水平保持与世界同步。会议将邀请国际知名学者做辅导讲座和特邀报告。

本届会议欢迎国内外同仁莅临，有关大会信息请向中国人民大学信息学院孟小峰教授垂询。

邮编：100872　Tel/Fax：010-6251 5153　Email：xfmeng@public.bta.net.cn；

大会网址：http://www.cs.ucsb.edu/~waim02；http://www.cs.ust.hk/waim/

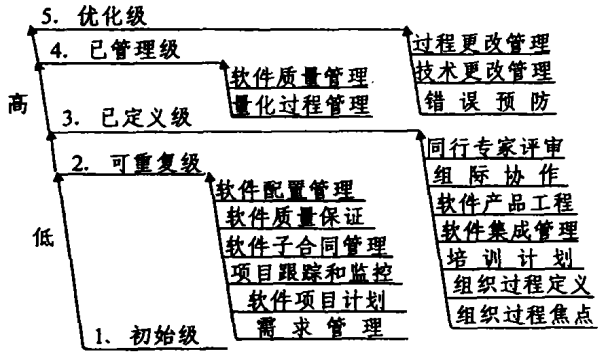

图5 CMM 的成熟度等级和各等级实施的关键过程域

CMM 的缺点之一是从起点级1级"初始级"到2级"可重复级"之间的 KPA(关键过程域)较多,台阶太高,跨度太大,实施周期长而见效慢,在一定程度上挫伤了软件机构的积极性;而 SPICE 模型要求软件机构从起点级0级"不完整级"开始首先建立基本的软件工程过程,树立质量意识,实现 PA 1.1 过程实施属性(Process Performance Attribute)要求后升到1级"可实施级",然后按照 PA 2.1实施管理属性(Performance Management Attribute)和 PA 2.2 产品管理属性(Work Product Management Attribute)的要求进一步规范和改进软件过程而达到2级"已管理级",……软件能力等级划分更为合理,更有利于软件开发组织循序渐进地逐步提升软件能力等级。

CMM 的缺点之二是关于等级4"已管理级"和等级5"优化级"的阐述显得较为空泛,与等级2"可重复级"和等级3"已定义级"相比,有关的 KPA 及其所包含的关键实践定义得还不够完整和具体,增加了操作歧义和困难。这一问题在 ISO/IEC 15504中有了较大的改善。

从逻辑修辞上看,ISO/IEC 15504在软件能力等级、过程、实践等方面的定义和描述用语比 CMM 要更为准确和恰当。另外,SPICE 的结构层次划分也比 CMM 更为合理和完整。除此之外,CMM 中所涉及的软件生存周期过程描述与国际标准 ISO/IEC 12207 不太一致,也是 CMM 不利于国际化推广的一个缺点。

4.2 Bootstrap 和 Trillium 评估标准

除此而外,还有一些影响较为有限的评估标准,如欧洲 Bootstrap 和加拿大 Trillium 等。Bootstrap 标准主要为欧洲的某些组织所采用,Bootstrap 3.0由 Bootstrap Institute 于1995年发表,其评估方法趋向于符合 ISO 15504标准要求并借鉴 CMM 对过程能力和成熟度的定义,其特点是较为灵活实用,专门设计了适用于不同情况的三种方式:自我评估、完全评估和选择性评估。

结束语 ISO/IEC 15504吸收了 SEI CMM 等早期标准的优点,在完整性、合理性、可操作性等方面有较大改进和发展,是一个集大成的软件过程评估标准。ISO/IEC 15504评估标准不仅可供用户(需方)对软件开发组织(供方)的开发能力及水平进行考察评估,同时也为开发机构提升软件过程能力水平提供了依据。许多软件开发组织多年的运用经验表明,SPICE 模型及相应的一套评估方法的确能帮助软件开发组织迅速改进软件开发过程,提高软件工程能力。统计数据说明随着软件开发过程的改进,所开发的软件产品中所含缺陷数明显减少;费用进度控制情况也随过程改进而改进。

在软件开发组织按有关标准对本组织的软件过程进行考察和改进时,可以使用美国 Rational 公司的 RUP、Intersolv 公司的 PVCS 和 CA 公司的 ADvantage 等软件开发管理工具来辅助和加强软件开发管理,营造一个规范而高效的管理机制,许多开发组织成功地采用上述工具而通过了 ISO 或 CMM 的评估认证。

可以预见随着 ISO/IEC 15504标准的完成,SPICE 模型及其方法将为软件开发组织不断改进软件工程过程、提高过程能力和产品质量指明可行的科学途径。

参考文献

1 International Organization for Standardisation (ISO). Information technology - Software process assessment: [Technical Report]. ISO/IEC TR 15504[S]:1998

2 http://www.software.org:ISO 15504 (SPICE)Description[EB/OL],2001.8

3 Rational Software Corporation. Assessing the Rational Unified Process against ISO/IEC15504[R],2000

4 International Organization for Standardisation (ISO). Information technology - Software Life Cycle Process, ISO/IEC 12207[S]: 1995

5 International Organization for Standardisation (ISO). Quality management and quality assurance standards Part 3: Guidelines for the application of ISO 9001 to the development, supply and maintenance of software, ISO 9000-3[S]:1997

6 Magnani G, Garro I. Software Process Improvement towards Business Improvement[J]. METHODS & TOOLS, ISSN 1023-4918, Fall 1999 (Volume 7 - number 3)

7 Paulk Mark C. Capability Maturity Model for Software, Version 1.1[S]. Carnegie Mellon University, Software Engineering Institute: CMU/SEI-93-TR-25, Feb. 1993

8 http://www.sei.cmu.edu: Capability Maturity Model (SW-CMM)for Software[EB/OL],2001.8

9 何新贵. 软件能力成熟度模型 CMM 的框架与内容[J]. 计算机应用,2001(3)

10 Herron D, Garmus D. Estimating Software Earlier and More Accurately[J]. METHODS & TOOLS, ISSN 1023-4918, Fall 1999 (Volume 7 - number 3)

(上接第82页)

需要指出的是,以上所给出的等价变换规则、扩充的代数规则及一些相应的性质,不仅可以用来在全局查询转换为中间查询的过程中进行代数优化,它们同样适用于中间查询向局部查询的转换,从而实现全局查询的优化处理。

参考文献

1 Sheth A P, Larson J A. Federated Database System for Managing Distributed, Heterogeneous, and Autonomous Database, ACM Computing Surveys, 1990, 22(3): 183～236

2 Chen M S, Yu P S. Using combining join and semijoin operations for distributed query processing. IEEE Trans. Knowledge and Data Engineering, 1993, 5(3): 534～542

3 Lee C, Chen C J. Query Optimization in Multidatabase Systems Considering Schema Conflicts. IEEE Trans. on Knowledge and Data Eng., 1997, 9(6): 941～955

4 Zhu Q, Larson P.-A. Solving local cost estimation problem for global query optimization in multidatabase systems. Distributed and Parallel Databases, 1998, 6(4): 373～420