计算机科学 2002Vol.29№.10

半结构数据的存储模型和查询执行*)

Storage Models and Query Execution of Semi-Structured Data

冯建华[1] 王钦克[1] 周立柱[1] 孟宪虎[2]

(清华大学计算机科学与技术系 北京 100084)[1] (运城高等专科学校计算机系 运城 044000)[2]

Abstract Semi-structured data are generally modeled as labeled graphs. Data in such models are self-describing and dynamically typed, and capture both schema and data information. Such models, although flexible, evoke severe efficiency penalties compared to querying structured database, such as relational databases. In order to improve the efficiency of data manipulation by utilizing structure information, we present a hybrid method capable of reorganizing semi-structured data on the basis of their structural degrees. The method extracts data with high degrees of structure and stores them in relations while leaves the rest part in its original graph form. This paper gives the algorithms for generating and dynamic updating storage model of the method, illustrates how queries could be executed based on the storage model and analyzes its improvement in processing queries, comparing with common execution methods. It also gives an algorithm that converts queries on semi-structured data to relational calculus, which provides a way to utilize query optimization techniques in relational database systems.

Keywords Semi-structured data, Relational database, Storage model, Query execution

1 引言

半结构数据是指区别于语音和图像文件等"原始数据",具有一定程度的结构,又不像传统的数据库系统那样存在严格模式的数据[1,2]。半结构数据广泛存在于各种电子数据源,特别是 Internet 当中。以 WWW 为例,其 HTML 文件格式本身就是由标签和锚点等结构单元组成的,因此文件中的数据常常具有明显的结构。但同时数据的结构又非常不规范,不符合传统数据库的要求,因此不能简单地应用现有的数据库技术和工具对其进行处理,需要研究和开发对半结构数据进行描述和处理的新技术、新工具。

1.1 问题的提出

由于半结构数据具有结构复杂、不规范和易变等特点,研究人员普遍采用灵活的图或树形结构(在半结构数据的相关讨论中,通常不加区分地统称为树)来设计半结构数据模型。边标记图[3]是其中具有代表性的一种。其他的半结构数据模型,如 TSIMMIS[4],LORE[5,6]等系统中所采用的模型,都可以转化为边标记图模型。在数据模型的基础上,研究人员又提出了若干半结构查询语言,如 UnQL[3]和 LOREL[5,6],这些查询语言主要以正规路径表达式为基础,以便能够递归地搜索任意深度的数据路径。

边标记图模型具有很强的表达能力,能够灵活地表示网络上各种格式的数据,但在描述高度结构化的数据时存在大量的模式信息冗余,而且不能利用高效率的传统数据处理技术。这是由于数据所具有的结构完全隐含在数据表示当中,边标记图模型无法对数据中不同程度的结构进行明确的描述和概括。另外,半结构数据的查询执行完全依赖于对数据的遍历,数据的处理效率是难以令人接受的。

1.2 本文的工作

为了解决这一问题,研究者提出了许多描述和利用半结构数据中的结构信息的方法,包括 Data Guides[7]、Graph Schemas[8]、Schema Definition Language[9]以及 T-indexes[10]等。本文提出了根据半结构数据具体的结构程度,重新组织和存储数据的实际方法:对结构性较高的数据进行组织,并保存在关系当中,而结构性较弱的数据,仍保留其图数据的存储形式。本文简要分析了这种存储模型的优点,并给出了具体的生成算法。另外,还介绍了在该存储模型的基础上三种常用的查询执行策略的实现方法,并与其他的查询执行方法进行了比较。最后,本文给出了将半结构数据查询转换为关系运算的一般方法,提出了利用关系查询执行技术求解半结构数据查询的一般思路。

1.3 相关工作

目前描述和利用半结构数据的结构的方法主要可以分为两类。第一类方法根据预先定义的数据模式(或 XML 文件[11]的 DTD)将半结构数据(或 XML 数据)转化为关系或面向对象数据,再利用传统的数据库技术对数据进行处理。采用这种方法,数据处理的效率在很大程度上取决于数据模式的指定。如果预定义的模式过于严格,就会限制半结构数据可能的变化,这与半结构数据技术的主要对象——网络数据的灵活变化的特点是不相适应的。而如果数据模式对数据的限制过于宽松,对提高数据处理效率的作用不大。到目前为止,对数据模式的定义及其与数据处理效率的关系,还没有具体的规范和结果。

另外一些研究人员提出了从实际的数据出发描述和利用数据结构的方法,典型的如 Data Guides[7]。但数据指南随着数据的动态变化进行更新比较困难,而且数据指南的规模在理论上有可能达到原始数据节点数目的指数级。本文提出的存储模型,有效地解决了数据指南所面临的指数级规模和更新困难的问题,为半结构数据的查询执行提供了新的途径。

下面介绍半结构数据的数据模型和查询语言,接着提出其存储模型,然后给出关系存储的生成算法,并且介绍常用的三种查询执行策略的实现方法,最后给出将半结构数据查询

*)本文得到国家"973"重点基础研究发展规划项目(G1998030414)的支持。

转换为关系运算的具体算法。

2 半结构数据基础

2.1 数据模型

边标记图是常用的描述半结构数据的模型。边标记图中存在一个根节点，图中所有的数据节点都可以从根节点访问到，并且在数据库中有唯一的节点 ID。通常以根节点的 ID 来表示一棵树，因此边标记图可以形式化为成对的标记和树的集合：$\{l_1 \Rightarrow t_1, l_2 \Rightarrow t_2, \cdots, l_n \Rightarrow t_n\}$，其中 $l_1, l_2, \cdots, l_n$ 为标记，$t_1, t_2, \cdots, t_n$ 是其他的树。图 1 是一个数据库研究小组主页的图数据库，一般的图数据中还会存在回路。

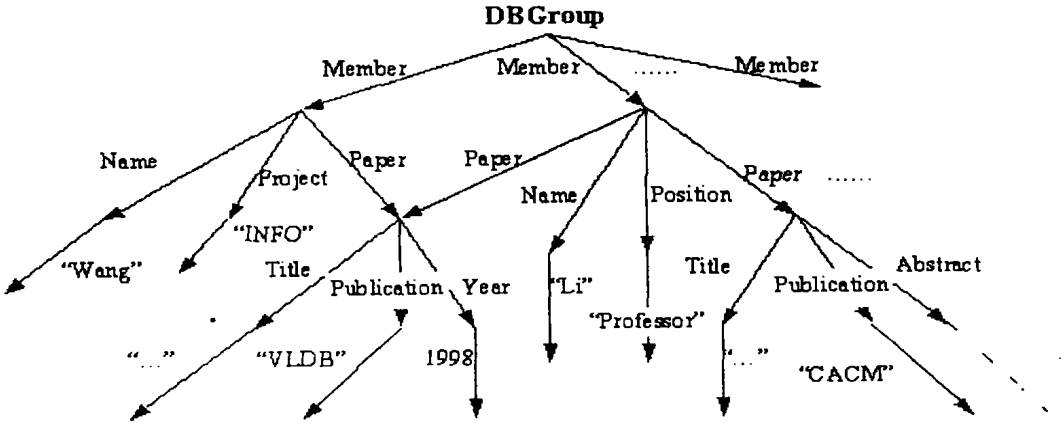


图 1 边标记图实例

本文采用的半结构数据模型是对边标记图的扩展。在实际情况下，有意义的基本类型的属性值往往出现在边标记图中叶节点相关的边上。因此这里限定基本类型的数据值只存在于叶节点，而内节点的标记只是一般的符号。该数据模型可以形式化地表示如下：

type BASE=INT|REAL|STRING|…

type TREE=BASE|set(SYMBOL×TREE)

下面给出一些与数据模型相关的定义和符号表示，以便在下面的讨论中能够清晰地对存储模型和查询执行方法进行描述和说明：

基本定义：

1）*路径*：以树 t 为起点，经过一系列交替的边和节点 $l_1, t_1, l_2, t_2, \cdots, l_n, t_n$，可唯一地确定一条 t 到 t_n 的路径，记为 $t.l_1 \Rightarrow t_1.l_2 \Rightarrow t_2.l_3 \Rightarrow \cdots \Rightarrow t_n$；

2）*路径表达式*：路径表达式由一系列标记 $l_1, l_2, \cdots, l_n$ 组成，表示由标记依次为 $l_1, l_2, \cdots, l_n$ 的边逐个连接而成的路径。以树 t 为起点，符合路径表达式 l 的所有实际路径的集合，记为 $t.l$；

3）*路径表达式 l 在树 t 中的目标集合* target(t.l)：定义为 $t.l$ 中所有路径的终点组成的集合，即 $\{t_n \mid t.d_1 \Rightarrow t_1.d_2 \Rightarrow t_2.d_3 \Rightarrow \cdots \Rightarrow t_n \in t.l\}$。

2.2 查询语言

在讨论半结构查询的执行时，本文使用了一个简单的包含正规路径表达式的查询语言，该查询语言可以用下面的文法表示：

Q::=(select Q where C,…,C)|var

C::=R⇒var in var

R::=Pred|(R⇒R)|(R|R)|R*

查询语句“select Q where …”的执行结果是图数据库中的一系列节点的集合。文法中的变量 var 表示图数据中的节点；有时需要用符号 ε 来表示一个空树。形如“R⇒x in y”的条件表示从节点 y 到节点 x 存在符合路径表达式 R 的路径。另外，正规路径表达式中还包括对标记和原子数据的谓词条件 Pred。

3 存储模型

本文所提出的半结构数据存储模型，其主体部分仍然是一个有标记的图，图中有两类节点：数据节点和模式节点。存储模型中的数据节点类似于图数据库中的数据节点，用来表示源数据中结构松散的半结构数据。数据存储中的模式节点对应于源数据中由类型相近的数据节点组成的集合，并且每个模式节点有一个主关系和若干从属关系，用来保存集合中的源数据节点的相关信息。源数据节点在模式节点的主关系中唯一对应于一个元组，因此可以用数据节点的 ID 作为主关系的主键。主关系的其他属性域用来保存节点集合中多数节点共有的属性值，而那些少量节点独有的属性域仍然作为数据节点存储。由于主关系中每个节点只有一个元组，具有多个属性值的节点属性，需要存储在单独的从属关系中，因此在从属关系中，源数据节点通常对应于多个元组，数据节点的 ID 是从属关系的外键。节点 ID 作为主关系和从属关系的主键和外键，在数据访问中，经常用来确定关系元组，因此在主关系和从属关系的 ID 属性域上建立哈希索引。另外还需要有一个全局索引 Map，保存源数据节点到模式节点的对应关系。

在前面介绍的数据库组的边标记图数据 DBGroup，可以如图 2 所示存储。

上图中源数据库的 Member 节点集合{2,3,…}对应于存储中的模式节点 22，多数 Member 节点所共有的属性 Name 和 Position 的属性值则保存在模式节点 22 的主关系 Member_Main 中。由于一个 Member 节点可以具有多个 Paper 属性，因此 Paper 属性值保存在模式节点 22 的一个从属关系 Member_Paper 中。另外，Member 节点 2 具有一个特殊的属性 Project，由于该属性并不是多数 Member 节点所共有的，因此相关数据仍然保留其图数据的表示和存储方式。类似地，源数据中的 Paper 节点集合{7,10,…}对应于模式节点 23，Paper 节点的多数属性值保存在模式节点 23 的关系 Paper_Main 中。总的来说，源数据中一个节点的相关属性值，在数据存储中可以保存在对应模式节点的主关系和从属关系中，也可以作为节点的相关图数据存在。

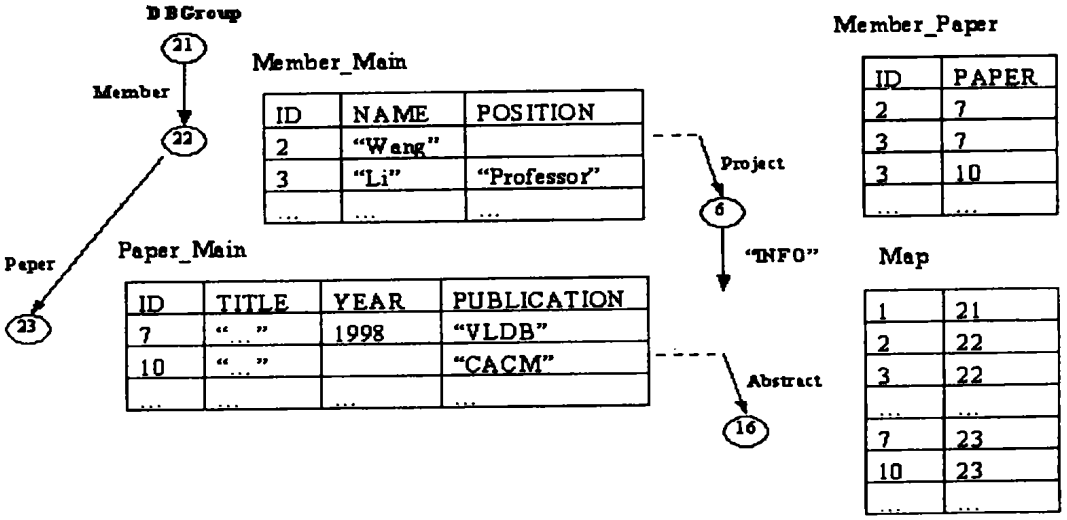


图 2 DBGroup 的模式和关系存储

这种关系与图数据相结合的存储方法，为利用半结构数据中的结构信息提供了有效途径。数据库中结构性较强的半结构数据能够在很大程度上转化为关系数据，并且可利用传统的数据库技术来提高半结构数据的处理效率。另一方面，这种存储方法还保留了边标记图灵活的表达和更新能力。半结构数据具有结构复杂、灵活易变的特点，将半结构数据完全转换为结构性的关系数据，会不可避免地形成大量的小规模的关系，而且难以随着数据的变化动态地进行更新，而以关系的大量连接操作完全代替对图的遍历，数据处理的效率也难以提高。试想一个结构比较松散的半结构数据源，完全存储于关系是很难得到令人满意的结果的，而本文所提出的存储方法可以保留其中大部分数据的图数据的形式。

4 模式和关系存储的生成

4.1 数据节点与模式节点的对应关系

为了尽可能地提高数据存储和处理的效率，对应于某个模式节点的数据节点集合应具有相近的语义类型，从而具有类似的属性集合和属性值类型。在图数据库中，边的标记说明了相应数据节点的语义。图数据库中，从根节点到数据节点的路径表达式相同的所有节点，通常具有相同的语义和属性，因此可以将源数据库中从根节点开始路径表达式相同的若干路径归结为模式中的一条路径，并将路径表达式的目标节点集合——对应于该路径上的模式节点[12,13]。采用深度优先的遍历算法可以实现上述简单的对应关系。但通常的半结构化数据并不是图论中严格定义的树。在半结构数据中，可能存在多条到达同一节点的路径，也就是说源数据中的节点有可能属于多个路径表达式的目标节点集合，从而对应于模式中的多个节点，因此按照这种简单的对应关系，存储会存在一定程度的冗余；模式的规模在理论上有可能可达到节点数目的指数级[7]。

为了避免在多个模式节点存储同一个节点信息所形成的冗余，可以将目标节点集合按照与其他节点集合的相交关系进一步划分。例如，两个不同的路径表达式的目标节点集合 A 和 B 存在着交集，那么可以将目标节点集合划分为 A—B、A∩B、和 B—A 三个不相交的节点集合，从而将两个路径表达式均可达到的节点与只有单个路径表达式可达到的节点区分开，进一步强化了节点集合所表达的语义信息。采用这种方法，数据节点唯一对应于一个模式节点，虽然在通常情况下所形成的模式节点数目可能多于简单的对应关系，但却避免了最坏情况下指数级模式节点数目的出现。

4.2 关系数据的生成

前面将具有类似语义的数据节点对应于一个模式节点，并在该模式节点的数据结构中保存数据节点的相关信息。在此之前，需要对这些节点的属性数据所具有的结构程度进行简单的判断。首先，根据集合中节点的数目决定是否将该节点集合转化为关系数据。只有当节点数目大于某个阈值 T1 时，才会创建模式节点的主关系。阈值 T1 的确定必须合理：如果 T1 的取值太小，有可能形成大量元组数目较小的关系，从而影响数据存储和处理的效率；T1 的取值过大，则不能充分地发挥出这种存储模型的优势。

确定对节点集合进行转化后，还要选择节点属性值的具体存储方式。首先需要区分属性值为基本类型的数据值和属性值为节点 ID 的不同属性。如果某个属性的属性值既包括基本类型的数据值，也包括节点 ID，那么应将该属性作为不同的两个属性处理。另外还要区别只能具有单个属性值和可以具有多个属性值的不同属性。

对于可以具有多个属性值的属性 ATTR，如果属性值的数目大于阈值 T1，则创建单独的从属关系(ID，ATTR)存储属性值。对于只能具有单个属性值的属性 ATTR，如果具有该属性值的节点数目与总的节点数目的比值较大(大于某个阈值 T2)，则在主关系中添加属性域 ATTR，这样可以保证主关系中大多数的元组具有该属性域，避免了存储的浪费；如果具有该属性值的节点数目与总的节点数目的比值小于阈值 T2，并且属性值的数目大于阈值 T1，则需要创建单独的从属关系。

阈值 T2 的确定，需要综合考虑数据存储和数据处理的效率。从存储的角度来看，假设标记所需存储长度为 l，节点 ID 或是基本数据类型的长度为 d，则阈值 T2 应取大于 d/(l+d)的某个值。阈值 T2 越低，存储的浪费就越大。但从数据处理的角度来看，将属性值存储在主关系，可以避免对单独的从属关系进行连接，数据处理的效率较高，因此又需要适当地降低阈值 T2。

4.3 存储模型的生成算法

算法主要包括三个部分，首先采用深度优先的搜索方法

生成模式节点及其所对应的数据节点集合。在搜索过程中，对于新形成的目标节点集合，如果已经存在相同的节点集合，则不需要创建新的模式节点；如果该节点集合与已有的节点集合存在交集，则需要暂停搜索过程，转而对相交的节点集合及其能够达到的所有节点集合进行划分。

划分过程也应采用深度优先的搜索方法，沿着已有的模式部分进行。由于数据节点间复杂的连接关系，有可能出现需要对已划分的节点集合再一次划分的情况。划分的循环过程一直持续到所有的已有节点集合都不再需要划分时停止。

形成模式和数据节点集合后，根据阈值 T1 和 T2，对于符合条件的节点集合的属性值，选择适当的存储方式，转化为关系数据。需要注意的是，为了提高数据处理的效率，通常避免出现节点集合的关系数据和图形数据有相同属性域的情况。因此，要对划分后的若干节点集合一并进行转化。最后，消除无用的模式节点，即对应节点集合不符合转化条件没有创建主关系的模式节点。

根据上述模式节点的生成方法可知，算法的复杂度在最坏情况下可以达到节点数目的指数级。另外，在生成节点集合的回溯过程中，可以同时消除部分无用的节点集合，从而减少划分过程的开销。

5 在存储模型上采用不同策略的查询执行方法

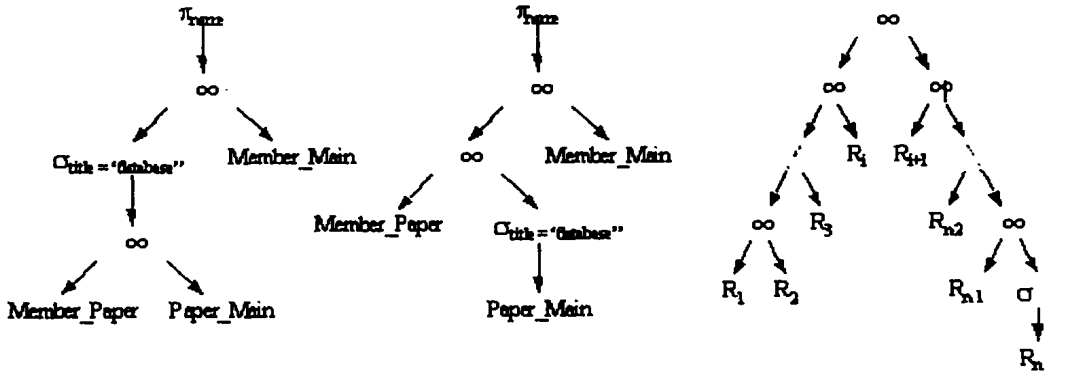
对于半结构数据的查询语句，通常有三种执行策略：从上到下的查询执行策略，从下到上的查询执行策略和混合式查询执行策略[14]。在本文所提出的半结构数据存储模型的基础上，也可以采用这三种策略来实现查询执行。在本节的讨论中，主要以下面的查询语句为例，具体描述采用不同策略的查询执行过程，并与其他存储方法上的查询执行在处理效率上进行了比较。在引言介绍的半结构数据 DBGroup 中，查询标题为“Database”的论文的作者，相应的查询语句如下：

```
select M. NAME
where member⇒M in DBGroup,
      paper⇒P in M,
      P. TITLE="database"
```

5.1 从上到下的查询执行策略

在图数据库的基础上执行上述查询语句，最简单的方法是从根节点开始，对图数据进行遍历，沿着路径表达式 DBGroup⇒member，找到数据库研究小组的所有成员节点。对每个成员 M，查找是否存在路径符合表达式 M⇒paper⇒title⇒“database”⇒ε。对符合条件的成员节点 M，返回其属性值 M. name。在这种查询执行方法中，从根节点开始，从上到下对路径表达式进行匹配。对图数据库来说，这种从上到下的查询执行策略，具体实现为深度优先的图遍历算法。

在本文提出的半结构数据存储模型的基础上，也可以实现类似的查询执行方法。参见图 2 中 DBGroup 的存储模式。显然，可以用模式路径上关系的依次连接来模拟对图数据宽度优先的遍历过程：在模式上匹配路径表达式 DBGroup⇒member⇒paper，并将路径上的关系依次连接，得到 Member_Paper∞Paper_Main。然后对匹配表达式所得到的 Paper 节点应用选择条件 TITLE="database"，并返回满足条件的相应 Member 节点的 Name 属性值。参见图 3(a)中的查询树。注意按照惯例，连接操作的左子节点是外关系，而右子节点是内关系。由于在主关系和从属关系的 ID 属性域上存在索引，因此关系从上到下的依次连接与图的遍历在时间复杂度上是相当的。而考虑到路径表达式的匹配，基于存储模型的查询执行有明显的优势。



(a)从上到下的查询执行计划　(b)从下到上的查询执行计划　(c)混合式的查询执行计划

图 3 不同策略的查询执行计划

5.2 从下到上的查询执行策略

利用路径索引，可以采用另外一种方法来执行上述查询。例如，DataGuide 在模式节点上标注了与之对应的目标节点集合。因此，可以在 DataGuide 上匹配路径表达式 DBGroup⇒member⇒paper⇒title，获得所有 Title 节点的 ID。然后对 Title 节点应用条件 TITLE="database"。对于满足条件的 Title 节点，在图数据上从下到上逆向匹配路径表达式，找到对应的 Member 节点。这种查询执行方法可以迅速地减少匹配路径的数目，对于近似树形的图数据查询优化的作用较大。但边标记图模型并不支持反向的指针，在图数据库中逆向遍历，需要额外的指向父节点的数据结构，例如文[14]中沿指定标记指向父节点的索引 Lindex。另外，对于复杂的图数据，逆向匹配表达式可能会沿着目标节点非根节点的错误路径进行，因此其开销也可能较大。

基于图 2 中的数据存储，也可以实现从下到上的查询执行方法。类似地，在模式上匹配路径表达式 DBGroup⇒member⇒paper，找到存储 Paper 节点信息的关系 Paper_Main，并应用选择条件 TITLE="database"，直接得到符合条件的 Paper 节点 ID。然后，沿着模式中的匹配路径，从下到上依次连接路径上的关系：Paper_Main∞Member_Paper，找到满足条件的 Paper 节点所对应的 Member 节点，并返回其 Name 属性值。参见图 3(b)中的查询树。

将路径上的关系从下到上地依次连接，类似于在图数据库中逆向遍历的操作。但关系的逆向连接不能利用主关系和从属关系在节点 ID 上的索引，因此连接操作的开销可能会大于利用逆向的边或索引进行遍历的开销。与图数据库和路径

索引相结合的查询执行方法相比，在模式的匹配路径上逆向连接关系，避免了对图数据库逆向匹配路径表达式所带来的标记比较和搜索错误路径的开销。

5.3 混合式查询执行策略

比较上述两种查询执行方案，从上到下的查询执行可以利用节点 ID 上的哈希索引完成关系的连接，而从下到上的查询执行则采用了类似下移选择操作的关系查询优化方法，并且有可能利用在基本类型的属性域上的索引检查选择条件。适当地结合两种方法，可以得到更好的优化效果。利用从上到下的方法，从模式的根节点开始匹配路径表达式，并依次连接路径上的部分关系，得到"有效"的数据节点集合。然后再利用从下到上的方法，直接判断基本类型属性上的选择条件，并逆向连接到相同位置的模式节点。对两个"有效"节点集合求交，最终得到查询的执行结果。这就是混合式查询执行的基本思想。

假设对一个简单的查询 Q = select M where $l_0 \Rightarrow$ M in DB，$l_1 \Rightarrow l_2 \Rightarrow \cdots \Rightarrow l_n \Rightarrow$ T in M，σ(T)，并且匹配路径表达式得到的关系依次为 $R_1, R_2, \cdots, R_n$，则按照混合式查询执行方法可以得到图 3(c)中的若干执行方案。最后根据估算的执行计划的开销，选择其中最小开销的方案执行。

6 查询执行的一般方法

从上节的讨论可以看出，在本文提出的存储模型的基础上，将半结构数据的查询求解转化为关系运算后，就可以应用传统的关系查询优化的思想和技术来选择具体的执行计划。为讨论方便起见，这里忽略了数据存储中的图数据，而只考虑其中的存储模式和关系数据。

首先给出根据存储模式 S，对形如"R⇒x in y"的查询条件进行转化的方法。根据路径表达式 R，构造等价的不确定自动机 A，其状态之间的转移上标注着谓词。假设 A 有 p 个状态 $a_1, a_2, \cdots, a_p$，模式 S 有 n 个状态 $s_1, s_2, \cdots, s_n$，构造 n * p 个状态的自动机 S×A。对模式 S 中的转移 $s \xrightarrow{l} s$ 和 A 中的转移 $a \xrightarrow{P} a$，如果 P(l) = TRUE，则在 S×A 中存在转移 $sa \xrightarrow{R} sa$，式中关系 R 为模式节点 s 包含属性域 l 的主关系或者从属关系。自动机的起始状态为所有的(s_y,a)，其中 s_y 为变量 y 所对应的模式节点，a 为自动机 A 的起始状态；类似地，其终止状态为所有(s,a)，其中 a 为自动机 A 的终止状态。创建自动机 S×A，并消除其中的无用状态后，采用类似于将自动机转换为正则表达式的方法，得到等价于条件"R⇒x in y"的关系运算：

1) 对形如 $s \xrightarrow{R} s \xrightarrow{R} s$ 的转移，变换为 $s \xrightarrow{R \times R} s$；

2) 对形如 $S \underset{R}{\overset{R}{\rightrightarrows}} S$ 的转移，变换为 $s \xrightarrow{R \cup R} s$；

3) 对形如 $\xrightarrow{R} s \xrightarrow{R} s \xrightarrow{R}$ 的自环，变换为 $\xrightarrow{R} s \xrightarrow{CLS\ R} s \xrightarrow{R}$，式中状态 s' 为临时状态，$CLS\ R = I \cup R \cup R > < R \cup \Lambda$ 为关系 R 连接操作的闭包。如果关系 R = I，即模式节点 s 所对应的所有数据节点存在自环或环路，这时可以忽略上面的自环，因此在构造查询语句时，应尽量避免这种情况出现。

上述变换一直持续到只剩下起始状态和终止状态时为止，这时两个状态之间的转移即为所求的关系运算。由于自动机 A 的状态数目与路径表达式 R 的规模相当，因此变换可以在 R 和 S 的规模的行列式时间内完成。

对于一般的查询语句，首先构造节点变量之间的关系图。图中的每个节点，代表着查询中的一个变量，并且对于每个形如"R⇒x in y"的查询条件，在图中有边 $y \xrightarrow{R} x$ 存在。从起始节点 DB 开始，逐步将变量间的路径表达式转换为关系运算，然后按照变量关系图将关系运算连接起来。将半结构数据的查询求解转化为关系运算后，可以利用传统的关系查询执行技术来具体地求解查询。

讨论 本文提出了关系和图数据相结合的半结构数据存储模型，按照数据具体的结构程度组织和存储数据，不仅为利用半结构数据中的结构信息提供了有效途径，而且保留了边标记图灵活的表达和更新能力。论文对在该存储模型上三种常用的查询执行策略的实现方法和其他存储结构上的查询执行方法的优缺点进行了比较。可以看出，存储模型上的查询求解提供了更多可选的执行方案，并且可以采用关系查询优化的思想来处理查询执行过程中的关系运算。对半结构数据的存储模型和查询执行，还有许多需要进一步研究的问题。例如，查询求解过程中出现的关系运算，有其自身的特点，在查询优化的开销估算和执行计划空间等方面都需要有特殊的考虑。另外，存储生成时采用的数据节点与模式节点的对应方法有时会产生不是非常必要的划分。例如数据库小组所发表论文的第一作者集合和其他作者集合通常存在交集，从语义上理解，可以形成包括所有论文作者在内的节点集合，而采用上述对应方法，将论文作者划分为只作为第一作者，只作为其他作者，以及既是第一作者，又是其他作者这样三个集合，过于详细的划分反而会影响数据更新和查询处理的效率，因此可以考虑采用适当的合并节点集合的方法[15]。

参考文献

1 Abiteboul S. Querying semi-structured data. In: Proc. of the Intl. Conf. on Database Theory, Deplhi, Greece, Springer-Verlag, 1997. 1～18

2 Buneman P. Tutorial: Semistructured data. In: Proc. of ACM Symposium on Principles of Database Systems, 1997. 117～121

3 Buneman P, et al. A query language and optimization techniques for unstructured data. In: Proc. of ACM-SIGMOD Intl. Conf. on Management of Data, 1996. 505～516

4 Papakonstantinou Y, Garcia-Molina H, Widom J. Object exchange across heterogeneous information sources. In: IEEE Intl. Conf. on Data Engineering, March 1995. 251～260

5 Abiteboul S, et al. The Lorel query language for semistructured data. International Journal on Digital Libraries, 1997, 1(1): 68～88

6 McHugh J, et al. Lore: A database management system for semistructured data. SIGMOD Record, 1997, 26(3): 54～66

7 Goldman R, Widom J. DataGuides: enabling query formulation and optimization in semistructured databases. In: Proc. of Very Large Data Bases, Sep. 1997. 436～445

8 Buneman P, et al. Adding structure to unstructured data. In: Proc. of the Intl. Conf. on Database Theory, Deplhi, Greece, Springer Verlag, 1997. 336～350

9 Beeri C, Milo T. Schemas for integration and translation of structured and semi-structured data. In: Proc. of the Intl. Conf. on Database Theory, 1999

10 Milo T, Suciu D. Index structures for path expressions. In: Proc. of the Intl. Conf. on Database Theory, 1999

11 World Wide Web Consortium. Extensible markup language (xml) 1.0, 1998. http://www.w3.org/TR/REC-xml

12 Wang Qinke, Zhou Lizhu. Schema-based Rearrangement of Semi-structured Data. In: Proc. of the 16th National Conf. of Database, Aug. 1999. 581～586

13 Wang Qinke, Zhou Lizhu. Schema Based Data Storage and Query Optimization for Semi-structured Data. In: Proc. of the First Intl. Conf. of Web-Age Information Management (Lecture Notes in Computer Science, Vol. 1846, Springer), Jun. 2000. 389～398

14 McHugh J, et al. Indexing semistructured data: [Technical report]. Stanford University, 1998

15 Goldman R, Widom J. Approximate DataGuides. In: Proc. of the Workshop on Query Processing for Semistructured Data and Non-Standard Data Formats, Jerusalem, Israel, Jan. 1999