一个面向 Internet 数据管理的系统模型*)

A System Model of Data Management for Internet Applications

韩立新　恽　爽　陈道蓄　谢　立
(南京大学计算机软件新技术国家重点实验室　南京210093)
(南京大学计算机科学与技术系　南京210093)

Abstract Since nowadays there are a great deal of semistructured data on the Internet and heterogeneous data integration systems, in-depth research on this kind of data is very important. In this paper, we propose a system model, Webmanager, for Internet information management. In Webmanager, classifier can process semistructured data and structured data respectively, thus can decrease system overhead. Moreover, we propose a set of strategies for query optimization including Max-Min algorithm for mining association rules. Based on these strategies, we also provide some of the optimizing algorithms for part of the basic query operations. These algorithms may improve query efficiency.

Keywords Semistructured data, System model, Data mining, Agent, Query optimization, Optimizing query operations

1. 引言

随着 Internet 的发展，查询网上信息变得越来越重要，常用的方法是使用诸如 Yahoo，Infoseek 等搜索引擎来查询信息。一旦用户提出需要查询内容的关键词，搜索引擎就根据关键词来确定查询的内容。不足之处是因为搜索引擎忽略了网页的内部结构，所以会导致用户不能获得准确的信息。许多搜索引擎有如下明显的缺点：(1)网页中较为具体的信息不能直接获取。(2)查询语言描述能力差。鉴于上述缺陷，研究人员借鉴一些数据库技术进行 WWW 上信息的查询。但它的不足之处是：WWW 上的数据差异大，数据结构不规则，因此关系数据库或面向对象数据库缺乏足够的灵活性来表示 WWW 上的数据。

目前正在研究的第二代网上查询语言能够支持对网页内部结构的查询，并且能够处理半结构化数据。影响较大的是 Stanford 大学的 TSIMMIS 项目[1~4]。TSIMMIS 的目标是提供一个框架和工具集，从位于 Internet/Intranet 上的多个相关联的异构数据资源上进行数据获取和集成。它采用自描述的对象交换模型 OEM 作为公共数据模型，使用轻量级 Lorel 语言查询半结构化数据，然后发送到由 MSL 语言编写的中介器，最后由包装器将 MSL 语言变换为本地数据源上的查询语言。此外，Roma Tre 大学的 ARANEUS 项目[5]采用页模型作为公共数据模型，同时页模型扩充了异构联合类型。它的缺点是半结构化数据灵活性较差，并且功能有限。

我们认为上述系统有以下不足：(1)查询速度有待进一步提高。(2)用传统的联邦数据库所采用的 Wrapper 对数据源进行包装，由 Mediator 向上提供一致的数据模型来对数据进行处理的方法，从而造成数据分布对用户不能完全透明。(3)无论半结构化数据还是结构化数据都必须经过公共模型进行处理，从而开销较大。

为解决上述问题，本文提出一个 Webmanager 系统模型。该模型是对从 Internet 上收集的有用信息进行模式抽取后所获得的数据进行管理，主要提供存储、查询等多种功能。此外，经少量修改，该模型也可以运用到异构数据源集成中。Webmanager 系统模型的主要特点如下：

(1)目前数据挖掘在数据库中的应用大多是利用数据挖掘工具自动地为分析人员搜索数据库，在不需要预先设想或假设的情况下，寻找出隐藏在数据背后的模式。而本文主要是将数据挖掘技术运用到查询优化中以提高效率。此外，本文还提出了另外一些查询优化策略，并在此基础上，给出基本查询操作的实现算法，从而较好地提高了查询速度。

(2)改变异构数据源普遍采用 Wrapper 对数据源进行包装，由 Mediator 向上提供一致的数据模型来对数据进行处理的方法。因此，分布式数据库的内部结构、数据分布等对用户是完全透明的，用户完全不必了解数据分布情况。

(3)考虑到 Internet 上的有些信息有相同的结构，而有些信息结构差异较大，因此使用分类器将半结构化数据和结构化数据分开进行处理。也就是结构化数据由广泛使用的关系数据库管理系统来进行处理，而半结构化数据使用较为广泛的对象交换模型 OEM 来存储并进行处理。这样做可以减少开销，提高效率，同时也可以充分利用已有的关系数据库管理系统。

(4)对已有关系数据库管理系统进行扩充来支持半结构化数据的处理，从而只需对关系数据库管理系统作少量修改就可以实现。

(5)提出新的多 Agent 系统的管理模式。在多 Agent 系统中主要存在一个通讯和协调的问题，很多现有的多 Agent 系统都采用集中式的管理模式。即在系统中有一个固定的管理器，所有 Agent 的通讯或请求都要通过这个管理器来协调。一旦这个管理器崩溃，那么整个多 Agent 系统也就瘫痪了。我们为了避免这个缺陷，对多 Agent 系统的管理采用的是完全分布式的管理模式。每台加入到该多 Agent 系统中的机器上都有一个管理程序，该程序负责 Agent 的加入，退出，信息注册等。每当某个 Agent 加入或退出时，都向本机的管

*)本研究得到国家八六三技术研究发展计划(编号863-306-ZT02-0301)和国家自然科学基金(编号69803005)资助。韩立新　博士研究生。恽　爽　硕士研究生。陈道蓄　博士生导师。谢　立　博士生导师。

理程序发出消息，而管理程序在更改本地信息的同时通知其它的管理器该 Agent 的动作。

2. Webmanager 系统模型

2.1 Webmanager 系统结构

图1描述了 Webmanager 的系统结构。

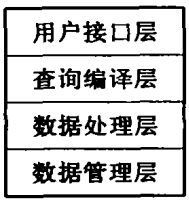

图1 Webmanager 系统结构

2.2 Webmanager 系统功能

Webmanager 系统由以下几层组成：

(1)用户接口层

·交互 Agent：提供给用户一个图形界面以方便用户提出查询请求，同时将查询请求转发给分析 Agent。当获得查询结果后，将反馈的结果显示给用户。

(2)查询编译层

·分析 Agent：将用户提出的查询语句转换为语法分析树，同时将生成的语法分析树转给查询计划 Agent 进行处理。

·查询优化 Agent：实现一些查询优化策略，例如采用路径索引以及链接索引和值索引相结合等方法。

·查询计划 Agent：同查询优化 Agent 相互协作，将接收到的分析 Agent 的结果进行优化，最终生成查询计划，并且将生成的查询计划转给基本查询操作 Agent。

(3)数据处理层

(a)半结构化数据的处理。对象管理 Agent：它包括管理对象的一些基本原语，例如创建对象、删除对象、取对象、比较对象；基本查询操作 Agent：它主要是利用查询计划 Agent 所生成的查询计划，完成查询的一些基本操作，例如选择操作、投影操作、连接操作和归并操作。

(b)结构化数据的处理。关系数据库管理系统：它用来对结构化数据进行处理。

(4)数据管理层

·数据管理 Agent：对已分类的信息进行管理，从而保证无论半结构化数据还是结构化数据的分布对用户都是透明的。

·数据分类 Agent：采用自学习的方式对已经存放的数据进行分析，将从搜索引擎获得的信息按半结构化数据和结构化数据的不同情况分开进行处理。

3. Webmanager 系统中的一些关键技术的实现

本节将介绍该系统中一系列关键技术的实现算法，并与相关算法进行相应的比较。

3.1 半结构化数据的查询优化

未优化的查询处理过程实质上是对图的遍历过程。如果采用这种方法进行查询，必然遍历很多结点，从而查询花费时间长，这在存储的数据量大时，尤为突出。

为了减少有向图的搜索范围，本文采用一些优化措施。目前普遍采用的优化措施是聚集和索引技术。在这里本文着重讨论基于索引的优化技术。

3.1.1 路径索引优化技术 本文提出以下几种路径索引的优化技术：

关联规则挖掘算法 Max-Min 是一个实现路径索引优化的算法。Max-Min 算法是从最大的频繁项目集出发，使用了具有很大影响的 Apriori 算法所提出的修剪策略，即一个频繁项目集的任一子集必定也是频繁项目集，因而 Max-Min 的最终结果实际上是简明地隐含了全部频繁项目集。Max-Min 算法的主要思想是：若较长项目是频繁项目集，则不需判定它的子项目是否是频繁项目集，并对子项目进行修剪。若较长项目不是频繁项目集，则按它的子项目的支持度大小进行排序，这是由于支持度大的项目更有可能是频繁项目集的一部分。

与具有很大影响的 Apriori 算法不同的是，Max-Min 算法采用对搜索空间进行“自顶向下”遍历的方法来代替 Apriori 算法所采用的“自底向上”遍历的方法。这主要是因为 Apriori 算法采用对搜索空间进行“自底向上”遍历的方法，而这种方法对较短频繁项目集处理效率较好，而对较长频繁项目集处理效率较差。然而这里的路径一般相对较长，因此为了弥补 Apriori 算法的这一不足，Max-Min 算法放弃了对搜索空间进行“自底向上”遍历的方法，而采用“自顶向下”遍历的方法。

Max-Min 算法的处理步骤如下：

```
输入:路径集
输出:统计出路径的使用情况
{统计所有路径的使用频率;
 While 路径集()nil
{ 取路径集中的最长路径 l_n;
  if l_n 是频繁项目集
  then 从路径集中删除这一路径的所有子路径;
  else { 按访问路径的频率对长度为 n-1的所有路径 l_{n-1}进行排序;
         While { l_{n-1}}()nil //{ l_{n-1}}是长度为 n-1的所有路径的集
           合
         {if l_{n-1}是频繁项目集 // l_{n-1}是长度为 n-1的某一路径
         then 从路径集中删除这一路径的所有子路径;
         else { l_{n-1}} = { l_{n-1}}-l_{n-1};
         if { l_{n-1}} = nil
         then n=n-1;
         }
       }
  }
}
利用频繁项目集生成具有最小可信度的关联规则;
统计出路径的使用情况;
}
```

Hashtree 算法 目前路径索引大多采用 hash 表来实现，而本文提出的 Hashtree 算法使用 Hash-tree[5]来实现路径索引，该 Hash-tree 的内部结点都和指向的 Hash 表相联，在 Hash 表中包含指向子节点的链，叶子节点用来保存路径，从而能更有效地进行路径索引。这种方法特别适用于含统配符的路径。

Hashtree 算法的处理步骤如下：

```
输入:查找的路径
输出:查找的结果
  {node =取路径中的第一个结点;
  While node<>nil
  { if Hash-tree 中不存在查找的路径结点
    then node 插入 Hash-tree 中;
    node =next(node);//取下一节点
    }
    获得查找的结果;
}
```

3.1.2 链接索引和值索引相结合的索引优化技术

Linkvalue 算法 是一个实现链接索引和值索引相结合的优化算法，在具体描述 Linkvalue 算法之前，首先给出以下函数：

(1)child(x,l,y)--通过 x 的指针把结点 x 的子结点放入

y 中。

(2)parent(x,l,y)--把结点 y 的父结点放入 x 中。

(3)val(类型,操作符,值,x)--输入操作符和值,输出的是满足操作符等于输入特定值的所有原子对象,并把它放入 x 中。在这里,如果类型是数值型,那么操作符为!=,=,<,>,<=,>=。如果类型是字符串型,那么操作符为 like,=,!=。

本文通过上述函数来实现 Linkvalue 算法,该算法避免了目前普遍采用的链接索引或值索引方法所造成效率低下的问题,并且能提高程序的并行性。

Linkvalue 算法的处理步骤如下:

```
{ if 无 join 操作
then 调用 val 函数找出满足条件的所有值;
else { 找出与 join 操作有关的根结点;
       在含有这些根结点的图中,调用 val 函数找出满足条件的所有值;
     }
  根结点调用 child 函数向下遍历到叶结点;
  child 函数所得的结果与调用 val 函数所得的结果相交,最终获得查询结果;
}
```

3.2 半结构化数据的处理

3.2.1 对半结构化数据基本操作的优化 在上述索引优化策略基础上,实现了在查询中归并操作、选择操作、投影操作等部分基本操作的优化。下面简单介绍归并操作、选择操作、投影操作等一些基本操作的实现过程。

·归并操作。由于半结构化数据的结构可能是不规则的或不完整的,因此易于造成数据源结构上出现冗余。本文解决冗余的方法是将冗余的数据源结构进行归并,从而减少数据冗余,提高查询效率。与普遍使用的归并操作算法不同的是这里提出的归并操作算法具有更好的并行性。

在具体描述算法之前,首先给出如下定义:

定义1 设 O_i 是对象,a_i 是属性,$i=0,\cdots,m$,则把 $p=O_0.a_0,\cdots,O_m.a_m$ 称为从起始对象 O_0 到终止对象 O_m 的路径。

定义2 设有 $P_i=(O_{i1}.a_{i1},\cdots,O_{in}.a_{in})$,$P_j=(O_{j1}.a_{j1},\cdots,O_{jn}.a_{jn})$,其中 $oid(O_{i1})=oid(O_{j1})$,$a_{i1}=a_{j1},\cdots,a_{in}=a_{jn}$,则称 P_i 和 P_j 是相似路径。

定义3 在一个图中存在多对相似路径,如果这些相似路径有相同的起始对象,那么把这些相似路径合并成一条路径叫做前缀归并。

定义4 在一个图中存在多对相似路径,如果这些相似路径有相同的终止对象,那么把这些相似路径合并成一条路径叫做后缀归并。

定义5 将原始数据源中的冗余结构进行合并的操作称为归并操作。

归并算法的处理步骤如下:

```
输入:原始数据源
输出:目标模式集
1)循环,直至该结点无子树;
2)找出根结点的子树;
3)判断各子树路径是否能进行前缀归并,如果可能,那么进行前缀归并;
4)判断各子树路径是否能进行后缀归并,如果可能,那么进行后缀归并;
5)取出各子树的根节点,回到1);
```

·选择操作。与普遍使用的选择操作算法不同:这里提出的选择操作算法采用前文中链接索引和值索引相结合的优化技术。

在具体描述算法之前,首先给出如下定义:

定义6 在给定的对象集中选取满足条件的路径集而进行的操作称为选择操作。

选择操作算法的处理步骤如下:

```
输入:存放关键词索引的 hash 表
输出:执行选择操作后满足条件的路径集
condition = find(); // 调用 find 函数,在 hash 表中找出满足条件的
    终结点对象集合。
ai= first(condition);//取出第一个终结点的对象
k=1;
while condition<>nil// 当终结点对象集不为空时循环
{ condition = condition-{ ai};
while aj<>根结点
{
{ aj}=parent({ ai});//取出 ai 的父结点
pathk = pathk+{ aj} ; //pathi 存放满足条件的第 i 条路径
}
k=k+1;
ai=next(condition);//取出下一个终结点的对象
}
```

·投影操作。在具体描述算法之前,首先给出如下定义:

定义7 在选择操作的基础上,从满足条件的路径集中选取从根对象到投影点所构成的路径集合而进行的操作称为投影操作。

投影操作算法的处理步骤如下:

```
输入:指定的属性集和满足条件的路径集
输出:投影操作后获得指定的路径集
//qattribute 存放指定的属性集;
path ={path1,…,pathn} ;//path 存放满足条件的路径集
i=1;
result={};
while pathi∈path // 当 pathi 为满足条件的路径集时循环
{
if finial(pathi)∈qattribute //如果 pathi 的终结点为指定属性
then result=pathi+ result;//result 存放指定的路径集
i=i+1;
}
```

4. 实例研究

4.1 原型系统 semistr

原型系统 semistr 的硬件实验平台为三台微机,其配置为 Intel P Ⅱ 400MHz CPU,64MB 内存,8.4GB 硬盘,操作系统为 Windows NT 4.0。该原型系统由用户界面、查询编译、数据处理、数据管理、信息搜索等部分组成。我们使用 Java 编程语言进行了实现。该原型系统主要完成存储关系数据库或面向对象数据库所不能存储的 Internet 网上信息,以及查询 Internet 网页中搜索引擎所不能查询的一些细节信息,并且使用查询优化策略来提高查询效率。在实验中,信息经搜索引擎收集后,使用抽取方法抽取出文档中的有用信息,并将抽取的信息使用数据分类 Agent 将半结构化数据和结构化数据分开进行处理,这些信息经处理后,半结构化数据存放在 OEM 模型中,结构化数据仍保存在 Sybase 数据库中。由用户输入需要查询的内容,经查询编译处理后,获得查询结果。实验结果表明,该原型系统取得了良好的效果。

4.2 实例分析

该原型系统使用分类器将半结构化数据和结构化数据直接分开进行处理,从而减少了中间环节,避免文[2,6]中所采用的无论半结构化数据还是结构化数据都必须经过公共模型进行处理的的缺陷。这样做可以减少系统开销,提高系统效率。由于在一些基本查询操作中运用了一些查询优化策略,例如在选择操作中运用了链接索引和值索引相结合的优化技术,查询数据的速度明显加快。此外,将冗余的数据源结构进行归并,减少了数据冗余,提高了查询速度。

4.3 性能评价

我们的实验环境是建立在以155M ATM 相连接的3台 PC 机上,PC 机的配置为 Intel P Ⅱ 400MHz CPU,64MB 内存,8.4GB 硬盘,操作系统为 Windows NT 4.0。我们使用 Ja-

va 编程语言进行了实现。

我们从多个网站随机抽取了一些网页，分别建立了包含50个网页信息和包含500个网页信息的实验数据库，然后使用本文提出的查询优化策略和 Stanford 大学的 Lore 系统中所采用的自顶向下查询策略进行了查询速度的比较，所得结果如图2所示。

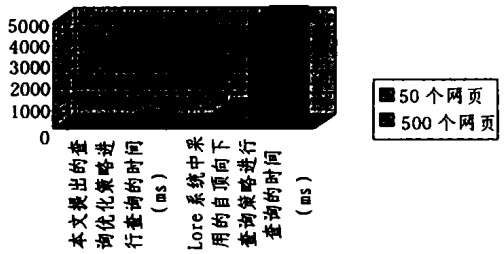

图2 测试结果比较

从图2可以看出：随着实验数据库中数据量的增加，本文提出的查询优化策略比使用 Lore 系统中所采用的自顶向下查询策略有越来越好的查询效果。因为随着实验数据库中数据量的增加，本文提出的查询优化策略进行查询的时间增加较少，而使用 Lore 系统中采用的自顶向下查询策略进行查询的时间增加较多。这主要是由于 Webmanager 系统使用了一些索引优化策略，在此基础上，还给出部分基本查询操作的优化算法的缘故。如在选择操作中运用了链接索引和值索引相结合的优化算法 Linkvalue，使得查询数据的速度明显加快。又如，将冗余的数据源结构进行归并，减少了数据冗余，也提高了查询速度。

结束语 鉴于异构数据源集成，特别是 Internet 中出现的大量难以存储的信息，本文提出了利用多 Agent 技术对 Internet 上的信息进行管理的系统模型 Webmanager。该模型是将半结构化数据和结构化数据分开进行处理，半结构化数据存放在 OEM 模型中，结构化数据仍保存在关系模型中。这样可以充分利用关系数据库管理系统的查询能力，并且只需将关系数据库做少量修改，从而减少了系统开销。从已发表的一些国内外文献来看，虽然在此方面已做了一些工作，但是我们的系统是通过对一系列关键技术的改进，从而使得系统的性能得到明显的提高。例如在路径索引优化技术方面，本文将挖掘技术运用到查询优化中，提出了基于关联规则的 Max-Min 算法以及使用 Hash-tree[5]来实现路径索引的 Hashtree 算法。此外，还提出了一个实现链接索引和值索引相结合的优化算法 Linkvalue。在此基础上，本文给出归并、选择、投影等部分基本查询操作的优化算法，从而大大提高了查询速度。目前该系统的设计思想已运用在国家863高科技项目"Java 开发环境的开发"的研制过程中，并且这一项目已通过了国家863专家组组织的鉴定和验收。

进一步的工作包括如何更好地处理动态信息，以及当循环路径[1]出现时，如何更好地进行处理。

参 考 文 献

1 Goldman R, Widom J. Approximate DataGuides. Proceedings of the Workshop on Query Processing for Semistructured Data and Non-Standard Data Formats, Jerusalem, Israel, January 1999
2 Abiteboul S, et al. The Lorel Query Language for Semistructured Data. International Journal on Digital Libraries, 1997, 1(1): 68～88
3 Papakonstantinou Y, Garcia-Molina H, Ullman J. Medmaker: A mediation system based on declarative specifications. In: Proc. ICDE Conf. 1996
4 Goldman R, Widom J. Interactive Query and Search in Semistructured Databases. In: Proc of the First Intl. Workshop on the Web and Databases (WebDB '98), Lecture Notes in Computer Science 1590, Springer-Verlag, Berlin, March 1998. 52～62
5 Agrawal R, Srikant R. Fast Algorithms for Mining Association Rules. In: Proc. of the 20th Int. Conf. on Very Large Databases (VLDB'94), Santiago, Chile, Sep. 1994. 478～499. Expanded version available as IBM Research Report RJ9839, June 1994
6 Atzeni P, Mecca G, Merialdo P. To Weave the Web. In: Proc. of the 23rd Intl. Conf. on Very Large Databases (VLDB'97), 1997
7 Goldman R, McHugh J, Widom J. From Semistructured Data to XML: Migrating the Lore Data Model and Query Language. In: Proc. of the 2nd Intl. Workshop on the Web and Databases (WebDB '99), Philadelphia, Pennsylvania, June 1999
8 McHugh J, et al. Indexing Semistructured Data: [Technical Report]. Stanford, Jan. 1998
9 Spertus E, Stein L A. Squeal: A Structured Query Language for the Web. In: Proc. of the Ninth Intl. World-Wide Web Conf. Amsterdam, May 2000
10 Konopnicki D, Shmueli O. WWW Information Gathering: The W3QL Query Language and the W3QS System. ACM Transactions on Database Systems, Sept. 1998
11 Spertus E, Stein L A. Just-In-Time Databases and the World-Wide Web. In: Proc. of the Seventh Intl. ACM Conf. on Information and Knowledge Management, Nov. 1998.
12 Deutsch A, Fernandez M F, Suciu D. Storing semistructured data with STORED. In: Proc. of ACM SIGMOD Conf. on Management of Data, 1999. 431～442
13 Florescu D, et al. Query optimization in the presence of limited access patterns. In: Proc. of ACM SIGMOD Conf. on Management of Data, 1999. 311～322
14 Johnson T. Performance measurements of compressed bitmap indices. In: Proc. of the Int. Conf. on Very Large Data Bases (VLDB), 1999. 278～289
15 Shanmugasundaram J, et al. Relational databases for querying XML documents: Limitations and opportunities. In: Proc. of the Int. Conf. on Very Large Data Bases (VLDB), 1999. 302～314
16 Galhardas H, et al. AJAX: An extensible data cleaning tool. In: Proc. of ACM SIGMOD Conf. on Management of Data, 2000
17 Buneman P, et al. Constraints for Semistructured Data and XML. SIGMOD Record, 2001, 30(1)

（上接第134页）

参 考 文 献

1 Nederlof L, et al. End-to-End Survivable Broadband Networks, IEEE Comm. Mag., 1995(Sep.): 63～70
2 Gryseels M, et al. Common Pool Survivablity for ATM over SDH Ring Networks. In: Proc. 8th Int'l. Symp. Network Planning, Sorrento, Italy, Oct. 1998
3 Demeester P, et al. Resilience in Multilayer Networks. IEEE Comm. Mag, 1999(Aug.): 70～76
4 Keping L, Yuehui J, Shiduan C. A New SDH-Based ATM Network Survivability Escalation Mechanism. High Technology Letters, 2000, 6(1): 27～32
5 赵季红，李增智，曲桦. 多层传送网的生存性策略，电信科学，2000(10): 14～17
6 Wu T H. Emerging Techologies for Fiber Network Survivability. IEEE Comm. Mag., 1995, 33(2): 58～74
7 Gryseels M, et al. Common pool survivability for meshed SDH-based ATM networks
8 Crochat O, Boudec J Y L, Gerstel O. Protection Interoperability for WDM Optical Networks. IEEE/ACM Trans. on networking, 2000, 8(3): 384～395
9 Grover W D, Bilodeau T D, Venables B D. Near Optimal Spare Capacity Planning In A Mesh Restorable Network, IEEE Globecom'91, pp2007～2012