基于数据网格面向服务的查询算法

朱 青[1)] 王 珊[1)] 丁博麟[2)] 张 孝[1)] 蔡宏艳[1)] 姚佳丽[1)]

[1)](中国人民大学数据与知识工程教育部重点实验室 北京 100872)

[2)](香港中文大学工程学院 香港)

摘 要 面向服务的框架(SOA)为用户的服务提供了一个标准的平台,实现服务的提供、发现、配置和集成,以帮助用户查询和处理信息.数据网格是面向服务的架构,为用户进行分布式远程数据查询服务提供了保障.对网格环境下 Hidden Web 数据库的研究与开发逐渐成为人们关注的焦点问题.要回答用户的查询,数据集成系统需要解决网格上的需求语义分析和关键字查询、建立数据查询模型.将数据库抽象为无向图,节点对应数据库中的元组,边对应"主-外码"的关系.查询的结果是与元组连接的答案树,它与查询的关键字相匹配.针对以上这些问题提出了一个新的查询算法,将改进的动态规划算法用于查询模型,保证 Top-1 答案树最优,Top-K 答案树近似最优;给出了实验测试和评估结果.

关键词 网格计算;SOA;数据集成;动态规划;查询算法

中图法分类号 TP311

Service-Oriented Search Algorithm on Data Grid

ZHU Qing[1)] WANG Shan[1)] DING Bo-Lin[2)] ZHANG Xiao[1)] CAI Hong-Yan[1)] YAO Jia-Li[1)]

[1)](*Institute of Data and Knowledge Engineering of Ministry of Education, Beijing* 100872)

[2)](*School of Engineering, Chinese University of Hong Kong, Hong Kong*)

Abstract A service-oriented architecture provides a standards-based platform that allows services to be provided, discovered, configured and integrated, to facilitate the creation of a business process. Data Grid is a service-oriented architecture that provides the coordinated search services for data distributed across remote resources. Research and development activities relating to the Grid have generally focused on application where data is stored in database which is called Hidden Web. To answer user queries, a data integration system employs a set of request semantic analyzing, keywords searching and model creating on Grid. This paper presents a novel model of searching, which the database is an undirected graph, of which each node correspond to a tuple of the database, and each edge correspond to a "primary key—foreign key" link. Results to a query are modeled as answer trees connecting tuples that match individual keywords in the query. This paper also presents a novel and efficient searching algorithm of dynamic programming, and the algorithm is employed in authors model to ensure the Top-1 answer tree optimal and Top-K answer trees nearly optimal. Finally, the algorithm's performance is tested and evaluated.

Keywords grid computing; SOA; data integrating; dynamic programming; searching algorithm

收稿日期:2006-02-17;**修改稿收到日期**:2006-05-18.**本课题得到国家自然科学基金**(60473069)、**教育部下一代互联网示范工程项目基金**(CNGI-04-15-7A)**和中国人民大学科研基金**(30206102.202.307)**资助**.**朱 青**,女,1963 年生,博士研究生,副教授,主要研究方向为网格计算、分布式算法、语义 Web 服务和高性能数据库. E-mail: zq@ruc.edu.cn. **王 珊**,女,1944 年生,教授,博士生导师,CCF 高级会员,主要研究领域为网格计算、高性能数据库、数据仓库和知识工程. **丁博麟**,男,1983 年生,硕士研究生,主要研究方向为高性能数据库和算法设计. **张 孝**,男,1972 年生,副教授,主要研究方向为网格计算、高性能数据库. **蔡宏艳**,女,1981 年生,硕士研究生,主要研究方向为高性能数据库. **姚佳丽**,女,1979 年生,硕士研究生,主要研究方向为高性能数据库.

1 引　言

数据集成与关键字查询[1]普遍用于 Internet 的信息检索. 计算机用户利用搜索引擎 Google、百度等,可以方便地从网上获取他们需要的信息. 大约 80%的数据被存储在网络各结点(peer)的数据库中无法发现,即为 Hidden Web[2]数据. 因此,具有挑战的任务是,如何集成分布式数据集,以满足数据密集性应用的需求;如何透明地、快速查询异地异构数据库中的数据.

开放网格服务体系结构(OGSA)定义了一种面向服务基础设施所需的基本体系结构和机制. 它可以直接应用于解决数据集共享、描述和分析等引发的挑战. 网格环境下数据的集成和关键词查询系统,即网格数据库信息检索 GIRDB(Grid IR for DataBase)是面向用户查询服务的应用研究.

本文研究的主要内容是接收用户的关键词进行逻辑语义分析,形成候选网生成规则;依据规则,从网格数据库中自动抽取信息构造模式图. 基于模式图,论文提出一种基于动态规划的有效查询算法,这个算法在生成的答案树打分函数控制下,进行 Top-K 排序可以输出近似最优结果.

本文第 2 节介绍相关工作;第 3 节提出系统模型,即数据集成抽象图和答案树模型;第 4 节给出改进的动态规划查询算法;第 5 节给出实验和算法效率评估;第 6 节总结及指出将来的工作.

2 相关工作

数据的集成和关键词查询是面向数据服务、信息服务的体系结构(SOA)应用研究的热点问题. 与我们系统研究相类似的代表有 BANKS[3]系统、DISCOVER[4]系统和 DBXplorer[5].

文献[3]建立了"主码-外码"抽象查询模式图;评分函数考虑点与边的分数和比例确定;查询算法采用启发式向后扩展查询,但查询效率低于 GIRDB. 文献[4]依据关键字生成候选网络,查询算法采用贪心算法.

文献[5]将数据库抽象成为无向图,其答案是数据图的一棵子树,但是构造答案是以模式图为基础的. 算法首先以宽度优先遍历的方法枚举模式图所有可能的子树(JoinTree),然后以 JoinTree 为"模板",构造 SQL 语句对模式图中的表做连接,从而生成所有的答案. 其不足之处在于只能处理 AND 语义的查询,另外,文献[5]生成所有的答案后再评分排序,耗时较多.

3 系统查询模型

系统 GIRDB 查询处理过程:系统对接收到的关键词进行语法解析语义分析. 抽象为无向图的数据模型实现数据集成,根据答案树评分函数进行打分,通过基于动态规划的有效查询算法生成 Top-K 查询结果.

系统核心技术包含关键字语义解释模块、查询模型生成模块、答案树评分模型和下一节重点介绍的基于动态规划的有效查询算法.

3.1 关键字语义解释

由于数据库管理系统 DBMS 不支持语义查询,在网格环境下提供服务的数据库需要借助网格中间件实现语义分析. GIRDB 采用本体和领域知识来支持语义查询,提供半自动的图形接口生成关系名和字段名语义上匹配的关键字. 关键字语义生成器接收从元数据抽取器传递来的数据(包括关系名和字段名),然后采用系统部署的本体和相关的领域知识,为关系和字段生成相匹配的关键字,并存储在相应的数据表中. 语义关键字的生成,为高效的查询提供了前提条件.

定义 1(查询语句). 设有 m 个关键词 $q_1, q_2, \cdots, q_m$,关键词 q 可以是字符串或多媒体参数、属性、源数据. GIRDB 接受的查询语句 $L(q_1, q_2, \cdots, q_m)$是一个以关键词为参数,AND/OR 为运算符的布尔运算表达式,其中运算符定义如下:

q_1 AND q_2:关键词 q_1 和 q_2 都必须存在查询结果中;

q_1 OR q_2:关键词 q_1 或者 q_2 需要存在查询结果中.

例如,查询语句表示为

(("Data Mining" AND "Jim Gray") AND ("Alexander Szalay" OR "Jan vandenBerg")),

简化为

$((q_1 \text{ AND } q_2) \text{ AND } (q_3 \text{ OR } q_4))$,

它表示的含义是:用户需要查询作者 q_2 = "Jim Gray"发表的关于 q_1 = "Data Mining"的论文,并且是与作者 q_3 = "Alexander Szalay"或作者 q_4 = "Jan vandenBerg"合作完成的.

目前 GIRDB 部署了比较简单的本体,随着语

义和本体研究的不断发展,我们可以在系统中部署各类本体,采用更多的语义查询技术增加系统性能,应用信息检索技术更好地满足用户的需要.

3.2 查询模型生成

为了能够方便有效地提供网格的数据资源服务,GIRDB 访问控制器通过关键字语义生成器的规则结果,访问系统配置表和 DBMS 的系统表来获取数据信息.候选网络生成器主要用来发现网格数据库中各个表之间的连接关系,并根据这些元数据之间的约束定义(主外码约束等),重新构建关系模式图.这样在进行关键字检索时,查询算法就可以根据关系模式图生成候选答案树,从而使用户得到更丰富的相关信息.候选网络生成器的具体做法是,把网格上的数据库抽象成为数据图(DataGraph),并对重组的数据库模型抽象图进行规范定义.

定义 2(数据图 DataGraph). 网格数据库的抽象图 $G(V,E)$ 是带权无向图,其中,节点 $v(t)\in V$:数据库中的一个元组 t 对应 G 中的一个节点 $v(t)$,并且是一一映射.边 $(v_1,v_2)\in E$:如果数据库中元组 t_1 和元组 t_2 之间存在"主码-外码"联系,那么在 G 中存在边 $(v_1,v_2)\in E$,其中 $v_i=v(t_i)$.

定义 3(有效节点集合). 对于数据图 $G(V,E)$,节点集合 $S(q)\subseteq V$ 被称为关键词 q 的有效节点集合,当且仅当 $S(q)$ 由所有包含关键词 q 的元组对应的节点组成,即 $Vt(q)\in S(q)$,t 是数据库中包含关键字 q 的任意元组.

3.3 答案树评分模型

以往的查询结果的评分函数,有的只是单纯地考虑答案中节点和边的数目(DataSpot[6]),有的引入了图的结构和节点的度(BANKS[3]),有的考虑全文索引中关键词与数据库属性列的相关程度(DISCOVER[4]),还有的引入概率和期望概念,例如文献[7,8],我们考虑对 Top-K 答案和语义理解的精确程度,建立答案树评分模型.

定义 4(答案树). 如果把定义 3 中的查询语句 L 看成是以 AND/OR 为运算符,以有效节点 $v_1,v_2,\cdots,v_m\in V_T$ 为参数的布尔函数,那么当 $L(v_1,v_2,\cdots,v_m)$ 是有效节点集合时,$T(V_T,E_T)$ 是一棵答案树.

在查询部分(query phase),GIRDB 通过 Parser 获取用户输入的查询语句中的 AND/OR 语义信息;再通过搜索引擎从数据库中得到有效节点(元组)集合;查询处理器根据查询语义信息和有效节点集合,从数据库、数据图中获取必要信息,运行查询算法得到 Top-K 近似最优答案树.

如图 1 所示的数据图,节点 A,B,C,D 和 E 上分别包含关键字 A,B,C,D 和 E.如果输入查询"A AND B AND C",正确的输出是答案树 1(如图 2(a))和答案树 2(如图 2(b)).

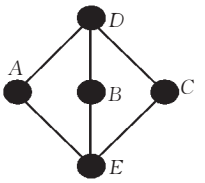
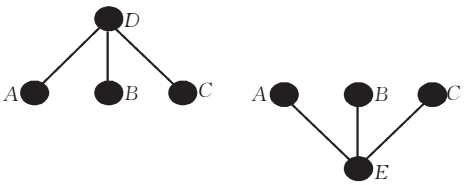


图 1 数据图



(a)答案树 1 (b)答案树 2

图 2 答案树模型

答案树评分模型系统采用一种基于邻近程度的答案树评分函数,分值越低表示邻近程度越高,结果越优.答案树的分值由树中的节点分值和边分值两部分组成.节点和边的分值又由两部分合成,分别是:(1)节点和边的类型决定的分值;(2)节点的度决定的分值.

定义 5. 节点的分值 $w_v(v)$:假设 v 的度为 $d(v)$,数据图 $G(V,E)$ 中节点的度的最大值为 $d_{\max}$,节点 v 的基本分值为 $b(v)$,那么它的分值定义为

$$w_v(v)=b(v)\cdot\frac{d(v)}{d_{\max}},$$

其中基本分值为 $b(v)$ 是由节点 v 的类型确定的,或者由 v 对应元组所在的关系类型确定,可以由用户进行设置.$w_v(v)$ 的值和 v 周围节点的邻近程度成反比.

定义 6. 边的分值 $w_e(e)$:假设 $e=(v,u)$,v 的对应元组在关系 R_1 中,u 的对应元组在关系 R_2 中,那么边的权定义为

$$w_e(e)=s(R_1,R_2)\cdot\frac{d(v)\cdot d(u)}{d_{\max}^2},$$

其中 (R_1,R_2) 确定了边 e 的类型,$s(R_1,R_2)$ 表示不同关系之间的邻近程度,它也能由用户进行设置.$w_e(e)$ 的值和 v 与 u 之间的邻近程度成正比.

定义 7(答案树的评分函数). 答案树 T 的分值通过评分函数定义如下:

$$Score(T)=\lambda\sum_{v\in V_T}w_v(v)+(1-\lambda)\sum_{e\in E_T}w_e(e),$$

其中 λ 是一个参数,用于调整答案树分值中节点分值和边分值合成的比例. 分值越低,答案树与关键词之间的关系越紧密,就越接近用户需要的查询结果.

节点和边的类型决定分值的引入理由,不同类型的边和节点所反映的紧密程度是不一样的,况且不同用户的偏好是不一样的,有时候他们需要自己设定不同类型的边和节点的基本分值,例如在 DBLP 数据库中,有的用户可能认为 CITE 关系更紧密,有的用户可能认为 WRITE 关系更紧密.

节点的度决定的分值说明如下:如果一个节点连接了很多其它节点,那么它应该具有很高的分值,也就是说它周围的节点邻近程度很低. 另一方面包含节点的答案树反映的关键词之间的关系不紧密.

GIRDB 系统目标就是向用户输出 $Score(T)$ 函数得到的分值最低的 K 个答案树.

定义 8(最优子树 $MT(r,s)$, $MT(r,s,h)$). 为了方便以下的算法描述,下面定义最优子树 $MT(r,s)$ 和 $MT(r,s,h)$:

$T(r,s)$ 为以 r 为根,包含了关键词集合 s 的、数据图 G 中的子树的集合;

$T(r,s,h)$ 为以 r 为根,高度$\leq h$,包含了关键词集合 s 的、数据图 G 中的子树的集合;

$MT(r,s)$ 代表 $T(r,s)$ 中分值最低的那一棵,即 $MT(r,s)\in T(r,s)$ 对应

$$Score(MT(r,s))=\min(Score(T(r,s))\mid T\in T(r,s)),$$

记为 $SMT(r,s)$;

$MT(r,s,h)$ 代表 $T(r,s,h)$ 中分值最低的那一棵,即 $MT(r,s,h)\in T(r,s)$ 对应

$$Score(MT(r,s,h))=\min(Score(T(r,s,h))\mid T\in T(r,s,h)),$$

记为 $SMT(r,s,h)$.

4　改进的动态规划查询算法

GIRDB 所处理的关键词查询问题是一个 NP 完全问题,当查询的关键字个数不多时,其算法复杂度可以接受. 假设用户查询的关键字个数控制在 6～8 个以内,其关键词集合的幂集至多包含 64～256 个元素. 本文在此基础上进行研究.

4.1　传统的动态规划算法

动态规划查询算法描述为:依据用户的输入和评分函数得到的 Top-1 答案树理论上最优(分值最低),而按分值从低到高还可以得到后面 Top-K 近似较优的答案树. 算法的关键是用动态规划方程来解所有的最优子树 $MT(r,s,h)$.

算法思想:答案树的构造从叶节点开始,逐步由叶节点合并成小的子树,再由小的子树合并成为大的子树最终得到答案树. 如果要求答案树是最优的,那么合并得到它的子树也必然是最优的;反之,最优的子树的合并却不是总能得到一棵更大的最优答案树,因此枚举所有子树的组合和合并的方案,可以得到最优的答案树. 这就是动态规划的"最优化原理".

求解 Top-K 答案树的算法流程:

1. 获取查询语句 $L(q_1,q_2,\cdots,q_m)$;

2. 通过进行全文检索得到关键词 $q_1,q_2,\cdots,q_m$ 的有效节点集合 $S(q_1),S(q_2),\cdots,S(q_m)$ 之后,就可以确定一部分 $MT(r,s)$ 或者 $MT(r,s,h)$ 的值了,包括 $MT(r,\{q_i\})$ 或者 $MT(r,\{q_i\},1)$(其中 $r\in S(q_i)$),初始值是只包括一个节点 r 的退化了的子树;

3. 按分值从小到大确定所有最优子树 $MT(r,s)$,每确定一个最优子树后就判断它是否为答案树,如果是则输出,这样输出前 K 个最优子树就作为近似的 Top-K 答案树.

最优子树 $MT(r,s,h)$ 的动态规划方程为

$$\begin{cases} MT'(r,s,h)=\min\Big\{\sum_{i=1}^{k}[MT(p_i,s_i,h-1)\oplus(p_i,r)]\mid \\ \qquad\qquad \forall k\geq 1;\ \forall\{p_i:i=1,2,\cdots,k\}\subseteq N(r);\\ \qquad\qquad \forall\{s_i\mid i=1,2,\cdots,k,\bigcup s_i=s\}\Big\}\\ MT(r,s,h)=\min\{MT(r,s,h-1),MT'(r,s,h)\}\end{cases}$$

其中 $N(r)$ 是节点 r 的相邻节点集合,$MT(p_i,s_i,h-1)\oplus(p_i,r)$ 表示在子树 $MT(p_i,s_i,h-1)$ 中加入边 (p_i,r),式中的求和符号 $\sum$ 表示子树的合并,函数 $\min()$ 表示取子树集合中分值最小的答案树.

当 $h\to\infty$ 时,最优子树 $MT(r,s,h)$ 为答案树. 即,给定 k 棵子树 $MT(p_1,s_1,h-1)$, $MT(p_2,s_2,h-1)$,…, $MT(p_k,s_k,h-1)$,添加节点 r 和 k 条边 $(p_1,r),(p_2,r),\cdots,(p_k,r)$,这样得到的子树根节点是 r,它上面包含的关键词集合为 $s=\bigcup s_i$,而它的高度小于等于 h. 反过来说,为了得到这样的子树中分值最小的那棵 $MT(r,s,h)$,枚举所有可能的 k 值($k\leq r$ 的度 $d(r)$),$\{p_i\}$ 以及满足 $s=\bigcup s_i$ 的 $\{s_i\}$,组合起来计算分值最小的那棵就可以得到 $MT(r,s,h)$.

此时还不能直接得到 Top-K 的答案树,计算完后,还需要把所有满足答案树条件的最优子树 $MT(r,s)$ 按分值排序以选取其中分值最低的 Top-K 个答案,排序所有满足答案树条件的最优子树甚至需要花费更多的时间.

传统的动态规划算法的计算是以数据图边和节

点规模为底数的,考虑到计算复杂性,须对动态规划方程进行改进. 当一个节点的度为 d,枚举所有可能的相邻节点集合 $\{p_i\}$ 就需要 $O(2^d)$ 的时间复杂度,这是算法复杂度升高的基本原因,再加上答案子树排序所花的时间会降低算法的效率. 因此可以对算法进行优化,得到一个改进的动态规划查询算法.

4.2 查询算法的改进

改进算法的核心思想:因为 $MT(r,s,h')\leq MT(r,s,h)$ 对于所有 $h'\geq h$ 成立,并且数据图 G 的大小是有限的,所以存在一个 h_0,使得 $MT(r,s,h_0)=MT(r,s,\infty)$,即 $MT(r,s,h')=MT(r,s,h_0)$ 对于所有 $h'>h_0$ 成立且 $MT(r,s)=MT(r,s,h_0)$. 为了得到 h_0 和 $MT(r,s)$ 的正确取值,采用迭代更新,把 $MT(r,s_1)$ 与 r 相邻的每个节点 p 上的所有最优子树 $MT(p,s_2)(s_2=1,2,\cdots,2^m)$ 进行合并,得到新的最优子树 $MT(p,s=s_1|s_2)$ 及对应分值,最后得到的极小值为查询结果.

对应 $MT(r,s)$ 的 h_0 来确定这个状态上的迭代更新如何停止是一个关键问题. 在每一次迭代被更新了的 $MT(r,s)$ 中,分值最小的 $MT(r_0,s_0)$ 将不会再被更新. 我们采用求最小堆的方法,每一次迭代时把堆顶元素取出,如果它满足答案树条件则构造相应的答案树进行输出,否则,把它和相邻节点上所有的最优子树进行合并更新,同时对堆进行调整.

对于用户输入的一个查询函数,通过进行全文检索得到关键词 $q_1,q_2,\cdots,q_m$ 有效节点集合 $S(q_1),S(q_2),\cdots,S(q_m)$ 之后,$MT(r,s,h)$ 的初始值可以设定如下

$$\begin{cases} MT(r,\varnothing,1)=r, & \forall i, r\notin S(q_i) \\ MT(r,s,1)=r, & \forall q_i\in s, r\in S(q_i) \\ MT(r,s,1)=\varnothing, & \exists q_i\in s, r\notin S(q_i) \\ MT(r,s,h)=undecidable, & \text{其它} \end{cases}.$$

相应地,$SMT(r,s,h)$ 的初始值为

$$\begin{cases} SMT(r,\varnothing,1)=w_v(r), & \forall i, r\notin S(q_i) \\ SMT(r,s,1)=w_v(r), & \forall q_i\in s, r\in S(q_i) \\ SMT(r,s,1)=\infty, & \exists q_i\in s, r\notin S(q_i) \\ SMT(r,s,h)=undecidable, & \text{其它} \end{cases},$$

令 $h\to\infty$,得到下面式子:

$$MT(r,s)=\min\{[MT(p,s_1)\oplus(p,r)]+MT(r,s_2)\mid \forall p\in N(r),\forall s_1,s_2 \text{ s.t. } s_1\cup s_2=s\}.$$

改进的动态规划查询算法如图 3 所示.

```
Make Heap TreeHeap(r,s);
//TreeHeap is a Min-Heap of MT(r,s);
//TreeHeap is according to value of SMT(r,s);
Initialize SMT(r,s);
For all r in V and for all s do
Begin
  SMT(r,s)=SMT(r,s,1),
  If SMT(r,s)<∞ then
  Put MT(r,s) into TreeHeap
End;
While TreeHeap is not Null do
Begin
Get the Top of TreeHeap: MT(r,s1),
If s1 is Acceptable Keyword Set then
  Output MT(r,s1);
//Update adjacent states from MT(r,s1)
//as following:
For each e=(r,p) in E do
For s2=0 to 2^m-1 do
If SMT(r,s1)+w(r,p)+SMT(p,s2)<SMT(p,s1|s2) then
Begin
  SMT(p,s1|s2)=SMT(r,s1)+w(r,p)+SMT(p,s2);
  If MT(p,s1|s2) is in TreeHeap
  then update TreeHeap;
  else Put MT(p,s1|s2) into TreeHeap;
End;
Delete MT(r,s1) from TreeHeap;
End;
```

图 3 改进的查询算法

因为堆最大为 $O(|V|2^m)$,一次堆操作需要时间 $O(m\log|V|)$. s_1 和 s_2 的可能取值均是从 0 到 2^m-1,它们的组合有 4^m 种可能. 改进算法的时间复杂度总计为 $O(4^m m|E|\log|V|)$,空间复杂度与基本算法一样为 $O(2^m|V|)$.

5 实验与测试

这一节,通过在不同大小的数据库上运行各种查询比较 GIRDB 系统与同类关键词检索系统 BANKS 的效率.

在实验中,采用了 DBLP 文献数据库,从中按会议或者时间抽取大小不同的子数据库进行实验,数据库大小从 40K 到 900K 不等,参见表 1. 而实验所采用的查询则是我们随机构造的,一般一项实验均有 50 个左右的查询语句,实验结果取其平均值. 在比较效率的时候采用了

"q_1 AND q_2 AND $\cdots$ q_n"

这样简单的 AND 语义查询. 实验中采用 ORACLE 9i RMDBS,硬件环境为 Pentium(R) Ⅳ 2.4GHz CPU /1GB RAM. 另外,我们采用 C++/ODBC/ORACLE Text Index 同一环境下实现了 GIRDB 和 BANKS 系统. 在保证查全率与查准率相当的情况下,重点比较两个系统的查询效率.

表 1　实验用的数据图大小

节点数(K)	边数
40	66590
100	124864
200	250482
600	708298
900	1028020

实验 1.　在 200KB 的 DBLP 数据库上进行,运行 2～6 个关键词的查询分别 15 个,图 4 是 GIRDB 与 BANKS 系统得到 Top-1 答案树的平均响应时间. BANKS 的响应时间从 2 个关键词开始均大于 GIRDB 的响应时间,这是因为 BANKS 所使用的"后向扩展算法"耗费的时间空间和 D^m 成正比(m 是关键词的个数,D 是包含任意关键词的元组的个数),而 GIRDB 中的算法耗费的时间空间只是与 2^m 成正比,所以随着关键词个数的增加,BANKS 的查询响应时间增加得会更多.

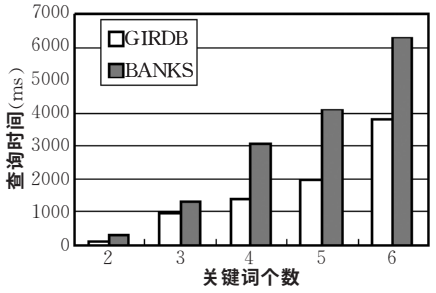


图 4　关键词数量对查询的影响

实验 2.　图 5 是两系统得到 Top-1 答案树的平均响应时间. 不同数目关键词的查询在实验中出现的频率基本与人们日常的查询一致. BANKS 的响应时间随数据库规模的增大变化比较剧烈,这主要是因为 BANKS 查询算法的搜索空间远远大于 GIRDB 查询算法的搜索空间,特别是当数据库规模增大的时候. 事实上,在"动态规划最优化原理"的保证下,GIRDB 的查询算法避免了很多状态空间中的重复搜索,这就是动态规划算法的最大的优势. 同时,随数据库规模增大,D(包含关键词的元组个数)也会增加,而 D^m 的增加就更加剧烈了.

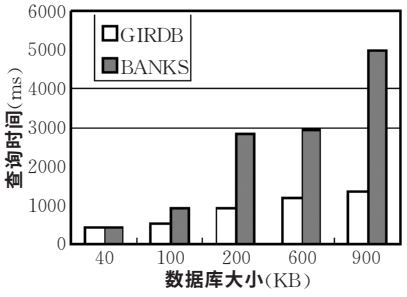


图 5　数据库规模对查询的影响

在与 GIRDB 系统进行比较的实验中,保证查准率和查全率的情况下,发现 BANKS 对于查询处理效率普遍偏低,这主要是因为 BANKS"后向扩展算法"的特点. BANKS 的查询算法相对 GIRDB 使用的算法消耗内存比较多,当内存消耗光而使用到外存缓冲的时候,查询处理时间就会急剧上升.

6　总　结

我们提出的 GIRDB 系统模型,是一种网格环境下的数据集成与关键字查询方法. 包括四个核心部分. 首先,对用户的查询语句进行分析和语义解释;之后产生最小的查询候选网络,保证查询正确有效地进行;然后采用有效的动态规划改进算法,创建近似最优的答案树,并依据评分原则,选择 Top-K 答案. 最后,实验对 GIRDB 与 BANKS 系统做了查询效率的性能比较.

我们计划进一步研究新的优化技术用于 GIRDB,例如,增加动态、可适应、自配置技术生成候选网络并对其优劣加以评估;对网格环境下的多种异构数据源进行集成. 总之将数据库技术与网格计算技术结合起来,最终能实现面向用户的高效服务.

参　考　文　献

1　Wang Shan, Zhang Kun-Long. Searching database with keywords. Journal of Computer Science & Technology, 2005, 20(1): 55～62

2　Florescu D., Levy A. Y., Mendelzon A. O.. Database techniques for world wide Web: A survey. Sigmod Record, 1998, 27(3): 59～74

3　Bhalotia G., Hulgeri A., Nakhe C., Chakrabarti S., Sudarshan S.. Keyword searching and browsing in databases using BANKS. In: Proceedings of the 18th International Conference on Data Engineering, 2002

4　Hristidis V., Papakonstantinou Y.. DISCOVER: Keyword search in relational databases. In: Proceedings of the 28th International Conference on Very Large Data Bases, 2002

5　Agrawal S., Chaudhuri S., Das G.. DBXplorer: A system for keyword-based search over relational databases. In: Proceedings of the 18th International Conference on Data Engineering, 2002

6　Dar S., Entin G., Geva S., Palmon E.. DTL's DataSpot: Database exploration using plain language. In: Proceedings of the International Conference on Very Large Databases, New York, 1998, 645～649

7　Chaudhuri S., Das G., Hristidis V., Weikum G.. Probabilistic ranking of database query results. In: Proceedings of the

28th International Conference on Very Large Data Bases, 2004
8 Balmin A., Hristidis V., Papakonstantinou Y.. ObjectRank: Authority-based keyword search in databases. In: Proceedings of the 30th International Conference on Very Large Data Bases, 2004
9 Sarda N. L., Jain A.. Mragyati: A system for keyword-based searching in databases. IIT Bombay, India: Report No. cs. DB/011052 on CORR, 2001
10 Hulgeri A., Bhalotia G., Nakhe C., Chakrabarti S., Sudarshan S.. Keyword search in databases. IEEE Data Engineering Bulletin, 2001, 24(3): 22～32

ZHU Qing, born in 1963, associate professor. Her research interests include Grid computing, distributed algorithms, semantic Web service and high performance database.

WANG Shan, born in 1944, professor, Ph. D. supervisor. Her research interests include Grid computing, high performance database, data warehouse and knowledge engineering.

DING Bo-Lin, born in 1983, M. S. candidate. His research interests include high performance database and algorithm design.

ZHANG Xiao, born in 1972, associate professor. His research interests include Grid computing, high performance database.

CAI Hong-Yan, born in 1981, M. S. candidate. Her research interests is in high performance database.

YAO Jia-Li, born in 1979, M. S. candidate. Her research interests is in high performance database.

Background

This work is supported by the National Natural Science Foundation of China under grant No. 60473069 namely, A New Research of Information Searching for Database on Grid Environment. The project aims to build a service platform and provide an environment for user searching on information Grid and Knowledge Grid. The authors have made researches on data integrating, keyword searching and its application on information Grid. Currently, they have made some progress in keyword searching on Grid, Web service composition, service discovery, data integrating and data mapping.