SEEKER:基于关键词的关系数据库信息检索*

文继军+, 王 珊

(中国人民大学 信息学院,北京 100872)

SEEKER: Keyword-Based Information Retrieval over Relational Databases

WEN Ji-Jun+, WANG Shan

(Information School, Renmin University of China, Beijing 100872, China)

+ Corresponding author: Phn: +86-10-84909436, E-mail: wenjj@ruc.edu.cn, http://www.ruc.edu.cn

Received 2005-01-04; Accepted 2005-03-11

Wen JJ, Wang S. SEEKER: Keyword-Based information retrieval over relational databases. *Journal of Software*, 2005,16(7):1270−1281. DOI: 10.1360/jos161270

Abstract: Traditionally, SQL is the main interface to access data from relational databases. However, it is difficult for inexperienced end users to learn the complicated syntax of SQL. Enabling keyword-based information retrieval over relational databases will allow users to acquire information from databases without any knowledge of SQL and the underlying database schema, just like the way of common search engines. This paper describes the design and implementation of SEEKER, a system supporting keyword-based information retrieval over relational databases. While there have been some existing systems that support searching text attributes in relational databases, SEEKER can also search database metadata and numeric attributes. Moreover, SEEKER employs an improved ranking function and supports Top-*k* queries. Experimental results show that SEEKER can achieve good retrieval performance.

Key words: relational database; keyword query; information retrieval; Top-*k* query

摘 要: 传统上,SQL 是存取关系数据库中数据的主要界面.但是,对于没有经验的用户来说,学习复杂的 SQL 语法是一件困难的事情.实现基于关键词的关系数据库信息检索,将使用户不需要任何 SQL 语言和底层数据库模式的知识,用搜索引擎的方式来获取数据库中的相关数据.描述了一个基于关键词的关系数据库信息检索系统 SEEKER 的设计和实现.现有的关系数据库关键词查询系统只能检索关系数据库中的文本属性,而 SEEKER 还可以检索数据库元数据以及数字属性.并且,SEEKER 采用了更合理的排序公式,支持 Top-*k* 查询.实验结果显示,SEEKER 具有良好的查询性能.

关键词: 关系数据库;关键词查询;信息检索;Top-*k* 查询

中图法分类号: TP311 文献标识码: A

* Supported by the National Natural Science Foundation of China under Grant Nos.60473069, 60496325 (国家自然科学基金); the National High-Tech Research and Development Plan of China under Grant No.2003AA4Z3030 (国家高技术研究发展计划(863)); the Science and Technology Plan of Beijing of China under Grant No.H030130060011 (北京市科技计划)

作者简介: 文继军(1971－),男,四川西充人,博士生,主要研究领域为数据库,信息检索;王珊(1944－),女,教授,博士生导师,主要研究领域为数据库,知识工程.

关系数据库是一种主要的信息存储机制.SQL 语言是关系数据库的标准查询语言,对普通用户来说,SQL 语言既难学习,又难使用.并且,用户在查询关系数据库中的数据时,必须知道这个数据库的数据模式.当关系数据库隐藏在网页后面或在 P2P 网络里共享时,用户很难获知它们的数据模式,因此无法用 SQL 语言进行查询.关键词查询是查询文档和网页的最简单、最流行的信息检索技术,关键词查询直观、易用,不需要学习查询语言,也不需要知道查询对象的底层结构.实现基于关键词的关系数据库信息检索,将使用户不需要任何 SQL 语言和数据库模式的知识,像使用 Google 那样通过提交关键词来获取数据库中的相关数据.

本文实现了一个基于关键词的关系数据库信息检索系统 SEEKER,与现有的关系数据库关键词查询系统相比,SEEKER 不仅可以检索关系数据库里的文本属性(CHAR,VARCHAR 等),还可以检索数据库的元数据(关系名、属性名等)以及数字属性(DATE,TIME,INT,FLOAT,NUMERIC 等);并且,SEEKER 采用了更合理的评分公式对查询结果进行排序,将 Top-*k* 结果返回给用户.

本文第 1 节介绍与本文相关的工作;第 2 节概略描述 SEEKER 的基本方法;第 3 节详细介绍 SEEKER 的体系结构和算法;第 4 节用实验来评测 SEEKER;第 5 节总结全文并概述未来的研究方向.

1 相关工作

文本文档和网页的关键词检索已经是成熟的研究领域.近年来,结构化和半结构化数据的关键词查询开始引起研究人员的兴趣.Oracle[1],DB2[2],SQL Server[3]等关系数据库都提供了文本搜索扩展,可以为数据库中的文本建立全文索引(倒排表索引),这为实现关系数据库的关键词查询提供了一个基础.已经有几个关系数据库关键词查询系统被开发出来:BANKS[4],DBXplorer[5],DISCOVER[6],IR-Style[7],EKSO[8]和 ObjectRank[9].这些系统的基本思路是一致的,即将数据库看做是由数据库中的元组(顶点)通过主码-外码关系(边)连接而成的图.当用户给出一个关键词查询时,则通过全文索引从图中找出含全部关键词的最小子图作为查询结果.

数据库关键词查询系统可以分为在线查询和离线查询两大类.在线系统将关键词查询转换为 SQL 查询,通过实时查询数据库来生成查询结果.由于转换而成的 SQL 查询中有大量的连接操作,所以在线系统的查询速度较慢.在线系统提高查询速度的关键在于减少数据库,尤其是磁盘的访问次数.BANKS 解决查询速度慢的方法是将整个数据库都放在内存中,显然这种方法只适用于中小数据库.当用户提交一个关键词查询时,DISCOVER 首先用 Oracle 的全文索引为每一个关系生成含不同关键词组合的元组集合,然后根据数据库模式图生成元组集合模式图,再从中找出包括全部关键词的子图(树),将子图转换为 SQL 查询,最后查询数据库得到查询结果.IR-Style 对 DISCOVER 进行了改进,利用信息检索的文本相关排序策略对查询结果进行排序,将 Top-*k* 查询结果返回给用户.IR-Style 提出了几种算法来提高查询效率,但效果并不理想.

离线系统通过预先计算生成中间结果,当用户提交关键词查询时,根据中间结果生成查询结果,因此,离线查询系统的响应时间较短.但是,由于离线系统需要一个预先计算过程,并且在数据库更新后必须重复这个“昂贵”的计算过程,所以离线系统仅适用于更新很少的数据库.EKSO 和 ObjectRank 是典型的离线系统.

XML 数据的关键词查询是对 XML 查询语言的一个有益扩充.Goldman 等人[10]研究了存储在关系数据库中的 XML 数据的关键词查询.Florescu 等人[11]对 XML 查询语言进行了扩展,实现了元素粒度的关键词查询.Xkeyword[12]将 XML 数据视作有向标记图,查询结果是 XML 片断连接成的树,树中包含全部关键词,根据树中边的数量多少进行排序.XRANK[13]用关键词查询 XML 文档,返回结果是元素形成的树,树中包含所有的关键词,并使用一个排序公式对查询结果进行排序.

现有的关系数据库关键词查询系统的不足之处主要有:(1) 不支持对数据库的元数据的关键词查询.数据库中的一些元数据,比如关系名和属性名,是可以用关键词来查询的.(2) 不支持对非文本属性的关键词查询.现有的系统很好地支持了文本属性的关键词查询,但是关系数据库中还存在着大量的非文本属性.例如,要从图 5 所示的 DBLP 数据库里查询 Jim Gray 在 1990 年后发表的关于 Transaction 的论文,用 SQL 语言表示这个查询:

```
SELECT a.Name, p.Title, p.Year FROM AUTHOR a, PAPER p, WRITE w
WHERE a.AuthorId=w.AuthorId AND p.PaperId=w.PaperId AND
        a.Name='Jim Gray' AND p.Title LIKE '%Transaction%' AND p.Year>=1990
```

现有的系统无法进行这种查询.(3) 做了一些不必要的限制.例如,现有的系统都要求查询结果包含所有的关键词,这样可以提高查询效率.但是,如果数据库里没有包含某个关键词,用户将得不到任何查询结果.包含部分关键词的查询结果可以帮助用户改进查询.本文的研究目标是构建一个能够有效解决上述问题的关系数据库关键词查询系统.

2 方法概述

2.1 问题定义

一个包含 n 个关系 $R_1,\ldots,R_n$ 的数据库 DB,可以将它看成是一个有向图 G,DB 中的每一个关系就是 G 的一个顶点,每一个主码-外码关系就是 G 的一条有向边,方向由主码所在的关系 $R_i\in G$ 指向外码所在的关系 $R_j\in G$,记为 $e(R_i,R_j)$,将 G 称为模式图(schema graph),例如图 5 所示的 DBLP 数据库模式图.模式图中允许自环和多重边存在.实际上,有向边可以是两个关系间任意一种连接关系,在 SEEKER 中暂时只考虑主码-外码连接关系.一个关键词查询 Q 由一组关键词 $kw_1,\ldots,kw_m$ 组成,表示为 $Q(kw_1,\ldots,kw_m)$.

定义 **1**. 连接元组树 TT 是以数据库 DB 中的元组为结点的一棵树,树中任意两个相邻的元组 $t_i\in R_i,t_j\in R_j$,它们之间的边 $e(t_i,t_j)=e(R_i,R_j)\in G$ 且 $(t_i\triangleright\!\triangleleft t_j)\in(R_i\triangleright\!\triangleleft R_j)$.$TT$ 的大小是它拥有的元组的数量,记为 sizeof(TT).

定义 **2**. 查询结果是含有至少一个 Q 中的关键词且每个叶结点都含有至少一个 Q 中的关键词的连接元组树.

定义 2 和 BANKS[4],DISCOVER[6]以及 IR-Style[7]的定义相类似,但 SEEKER 不要求查询结果包含全部关键词,只要求查询结果包含关键词的一个非空子集.

2.2 查询语言

SEEKER 要实现对数据库的元数据和数字属性的关键词查询,必须对查询语言进行扩展.SEEKER 的查询语言包含 3 类关键词:(1) Keyword;(2) Keyword:Keyword;(3) Keyword:⟨*op*⟩Value.第 1 类关键词用来查询文本属性.第 2 类关键词用来对文本属性进行包含元数据的查询,":"前的 Keyword 用来匹配关系或属性,":"后的关键词用来对匹配上的关系或属性进行关键词查询.例如,在关键词查询"Author:'Jim Gray'"里,对"Jim Gray"的查询将被限制在和"Author"相匹配的关系或属性上.第 2 类关键词可以帮助用户更精确地表达查询要求.例如,查询"Knuth Algorithm"和"Author:Knuth Algorithm"会产生不同的查询结果,原因是 Knuth 既可能是作者名,也可能出现论文的标题中.采用第 2 类关键词将对 Knuth 的查询限制在 Author 关系上,有效地消除了这种模糊性.第 3 类关键词用于对数字属性的查询,":"前的 Keyword 用来匹配属性,"⟨*op*⟩ Value"部分对匹配上的属性进行条件查询.⟨*op*⟩是关系操作符,可以是=,>,<,≥或≤,Value 是属性的值.第 1 节中的 SQL 查询可以用 SEEKER 的查询语言表示为"Jim Gray Transaction Year:>=1990".第 3 类关键词将数据库中的数字属性纳入到关键词查询的范围内,使用户可以更精确地表达查询要求.

3 系统体系结构

SEEKER 由以下几个模块组成(如图 1 所示):语法分析、IR 引擎、候选评分表连接树集合的生成和 Top-k 查询结果生成.本节将详细描述这几个模块,并用关键词查询"Author:'Jim Gray' Transaction Year:>=1990"作为实例来说明 SEEKER 的查询过程.

3.1 语法分析

如第 2.2 节所述,用户提交的查询是 3 类关键词的组合,这就需要对查询进行语法分析,将不同形式的关键词识别出来,分别存储,以便于 IR 引擎采用不同的方法进行处理.对第 1 类关键词,SEEKER 的语法分析器将它们识别出来,放入一个数组中存储.SEEKER 允许用户用双引号或单引号将多个单词合并为一个关键词,例如输入"Jim Gray",SEEKER 将把它作为一个单一的关键词来处理.对第 2 类的关键词,":"前后的关键词被分别放入不同的数组中,关键词的识别方法与第 1 类关键词相同.对第 3 类关键词,Keyword,⟨*op*⟩和 Value 被分别识别出来,

放入不同的数组中存储.对于 Value,还要识别出它的数据类型,辅助 Keyword 匹配相关属性.如果有两个以上的第 3 类关键词,它们的 Keyword 相同,语法分析器将把它们合并为一个关键词.例如,对查询“Jim Gray Transaction year:>=1990 year:<=2000”,语法分析器将“year:>=1990”和“year:<=2000”这两个关键词合并为一个单一的关键词.

查询实例在语法分析后得到 3 个关键词:第 1 类关键词“Transaction”;第 2 类关键词“Author,Jim Gray”;第 3 类关键词“Year,GE,1990,INT”,INT 表示整数.

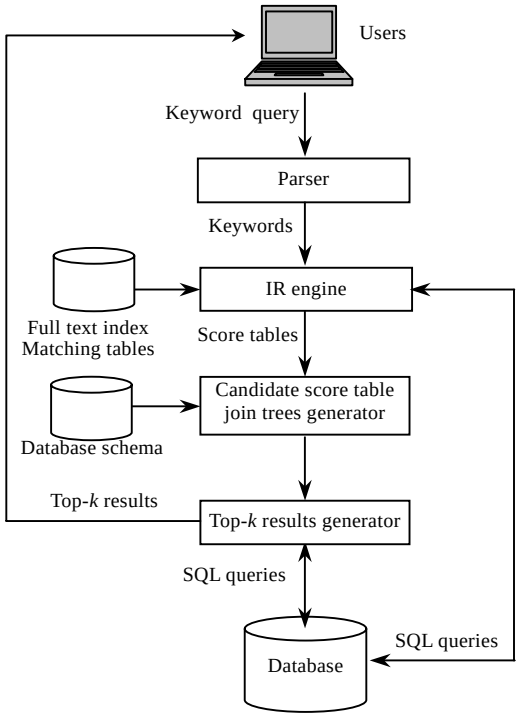


Fig.1 Architecture of SEEKER

图 1 SEEKER 体系结构图

3.2 IR引擎

SEEKER 使用 Oracle 9i 的全文索引来对文本属性进行关键词查询.Oracle 9i 不仅可以为关系中的单个属性建立全文索引,还可以为元组建立全文索引,采用的方法是将元组里每一个加入全文索引的分量用属性名进行标注,然后将这些分量串接起来,作为一个虚拟文档进行索引,关系中的所有虚拟文档作为文档集.这种索引同时支持元组级和属性级两种粒度的全文检索.

3.2.1 元数据的关键词查询

为了支持对元数据的关键词查询,在数据库中建立了两个匹配表(如图 2 所示),分别用于关系名和属性名的关键词匹配.这两个表上也建立了全文索引.关系匹配表的 TableName 列存储数据库中所有关系的名称,Keywords 列给出描述关系的关键词,这两列都在全文索引中.属性匹配表的 AttributeName 列存储数据库中的所有属性的名称,TableName 列存储属性所属关系的名称,Keywords 列给出描述属性的关键词,Type 列给出属性的类型,其中 AttributeName 列和 Keywords 列在全文索引中.用户在对元数据进行关键词查询的时候,可以猜测关系名或属性名,SEEKER 将在关系名、属性名以及描述它们的关键词中进行匹配.SEEKER 通过全文索引来进行关系和属性的匹配,采用的方法和对文本属性的关键词查询完全一样,匹配表中所有全文索引给出分数大于 0 的关系或属性都被认为是和关键词相匹配的.

对第 2 类关键词,关系的匹配优先于属性的匹配:如果只匹配上了某个属性,关键词查询只对该属性进行;如

果只匹配上某个关系,查询该关系;如果同时匹配上了某个关系和该关系的属性,查询关系.对第 3 类关键词,只匹配属性,因为只有确定属性,才能对数字属性进行条件查询.对这类关键词,还需要用关键词里“value”的数据类型对匹配上的属性进行校核,只有当匹配上的属性的数据类型和“value”的数据类型兼容时,才可以确认该属性和关键词是匹配的.

Relation matching table

TableName	Keywords
AUTHOR	writer, …
PAPER	thesis, …
WRITE	edit, …
CITE	reference, …

Attribute matching table

TableName	AttributeName	Keywords	Type
AUTHOR	AuthorId	id, …	INT
AUTHOR	Name	editor, writer, …	VARCHAR
…	…	…	…
PAPER	Year	date, …	INT
…	…	…	…

Fig.2 Matching tables (DBLP database)

图 2 匹配表(DBLP 数据库)

3.2.2 文本属性评分

文本属性的评分将由 Oracle 的全文索引给出.Oracle 的评分算法采用了逆文档频率(inverse document frequency)算法[1].文档的分数与查询词在该文档中的出现频率成正比,而与文档集中的出现频率成反比.Oracle 给出的分数范围为[0,100].

3.2.3 数字属性评分

为了保持和 Oracle 的评分算法的一致性,SEEKER 采用了下面的公式[14]来给数字属性评分:

$$\omega_{i,j}=\frac{freq_{i,j}}{\max_l freq_{l,j}}\times\log\frac{N}{n_i} \tag{1}$$

其中,$\omega_{i,j}$为关键词 k_i在文档 d_j中的权重,$freq_{i,j}$为 k_i在 d_j中的词频,$\max_l freq_{l,j}$是 d_j中最高的词频,N 是文档的总数,n_i是文档集中出现 k_i的文档数.

SEEKER 将整个数字属性作为文档集,属性中的每一个分量作为一个文档.这里,k_i 是第 3 类关键词“Keyword:⟨*op*⟩ Value”,d_j 是数字属性中的某个分量,N 是数字属性中的分量的数量,n_i 是数字属性中满足条件“⟨*op*⟩ Value”的分量的数量.显然,$\max_l freq_{l,j}$=1;对于满足条件“⟨*op*⟩ Value”的 d_j,$freq_{i,j}$=1;对于不满足条件“⟨*op*⟩ Value”的 d_j,$freq_{i,j}$=0.将$\omega_{i,j}$的值映射到区间[0,100]上,设定当 n_i=1,N=1 000 000 时,$\omega_{i,j}$=100.因此,式(1)变为

$$\omega_{i,j}=100\times\frac{1}{6}\times freq_{i,j}\times\log\frac{N}{n_i} \tag{2}$$

如果 n_i=0,则$\omega_{i,j}$=0;如果 n_i=N,则$\omega_{i,j}$=1;如果式(2)计算出的$\omega_{i,j}$>100,则$\omega_{i,j}$=100.

3.2.4 基本评分表和评分表

ROWID	SCORE
Rowid 1	Score 1
Rowid 2	Score 2
…	…

Fig.3 Basic score table

图 3 基本评分表

对于关键词查询 $Q(kw_1,\ldots,kw_m)$中的每一个关键词 kw_j,如果 kw_j 为第 1 类关键词,IR 引擎为数据库中的每一个关系创建一个基本评分表;如果 kw_j 为第 2 类或第 3 类关键词,只为匹配上的关系或匹配上的属性的所在关系创建基本评分表.基本评分表有两列(如图 3 所示),第 1 列是 Oracle 为每一个元组赋予的一个虚列:ROWID,ROWID 唯一标示一个元组,可以根据 ROWID 直接存取一个元组;第 2 列 SCORE 是 IR 引擎根据元组和 kw_j 的相关性评出的分数(由 Oracle 的全文索引或式(2)给出).IR 引擎为关系中每一个得分大于 0 的元组在基本评分表中插入一行,所有为空的基本评分表都将被丢弃.

IR 引擎对每个关系的基本评分表集合进行组合操作生成评分表集合,评分表和基本评分表的结构是一样的.设关系 R 的基本评分表集合为 $BSTSet\{BST(kw'_1),\ldots,BST(kw'_p)|0\le p\le m\}$,其中 $BST(kw'_j)$是关键词 kw'_j 对应的 R 的基本评分表.首先生成 R 的关键词集合 $KWSet(kw'_1,\ldots,kw'_p)$的全部子集,子集的数量为 2^p,然后为每个子集 $KWSet_i(i=1,\ldots,2^p)$生成一个评分表 ST_i,该评分表的 ROWID 列标示的 R 中的元组含且只含 $KWSet_i$ 里的全部关键

词,ST_i 的 SCORE 列的值为参与生成 ST_i 的所有基本评分表的 SCORE 列的值的和.根据下面的公式由 *BSTSet* 生成 R 的评分表集合 *STSet*:

$$ST_i = \bigcap_{k\in KWSet_i} BST(k) - \bigcup_{k\in KWSet \wedge k\notin KWSet_i} BST(k), \qquad KWSet_i \neq \varnothing \tag{3}$$

由定义 2 可知,R 中不含任意一个关键词的元组可以作为桥梁将含关键词的元组连接起来,这些元组的数量通常超过含关键词的元组的数量,因此,将它们在 R 中的补集存入评分表中.

$$ST_i = \bigcup_{k\in KWSet} BST(k) \qquad KWSet_i = \varnothing \tag{4}$$

显然,任意两个评分表之间没有交集,数据库中的任意一个元组只能存在一个评分表中.在最坏的情况下,一个关系的评分表数量为 2^m 个.但在一般情况下,关键词的数量并不会太大.当关系中不含某个关键词时,这个关键词对应的基本评分表为空,会被删除,不会用来产生评分表.并且,当评分表的元组数为空时,也会被删除.因此,评分表的数量通常不会太大.

查询实例经过 IR 引擎后生成的基本评分表是:AUTHOR$^{\{\text{“Jim Gray”}\}}$,PAPER$^{\{\text{“Transaction”}\}}$,PAPER$^{\{\text{“Year:>=1990”}\}}$;生成的评分表是:AUTHOR$^{\{\text{“Jim Gray”}\}}$,AUTHOR$^{\{\}}$,PAPER$^{\{\text{“Transaction”\&\&“Year:>=1990”}\}}$,PAPER$^{\{\text{“Transaction”}\}}$,PAPER$^{\{\text{“Year:>=1990”}\}}$,PAPER$^{\{\}}$,WRITE$^{\{\}}$,CITE$^{\{\}}$.

3.3 候选评分表连接树集合的生成

在本模块,首先根据数据库模式图 G 以及 IR 引擎生成的评分表集合 *STSet* 来创建评分表图 G_{ST}.评分表图 G_{ST} 是有向图,顶点是评分表.如果在数据库模式图 G 中,关系 R_i 和 R_j 之间有一条边 $e(R_i,R_j)$,那么在 R_i 的任意一个评分表 $ST_{i,p}$ 和 R_j 的任意一个评分表 $ST_{j,q}$ 间添加一条边 $e(ST_{i,p},ST_{j,q})$;如果 $i=j$,即在自环的情况下,不添加边 $e(ST_{i,p},ST_{j,p})$.因此,在评分表图 G_{ST} 中可能存在多重边,但没有自环.图 4 是查询实例生成的评分表图.

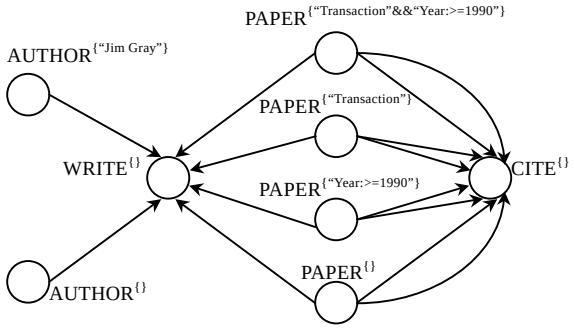


Fig.4 Score table graph of query example

图 4 查询实例的评分表图

定义 **3**. 候选评分表连接树是评分表图的一棵子树,它至少含一个 Q 中的关键词且所有叶结点都至少含一个 Q 中的关键词.候选评分表连接树 *CT* 的大小是 *CT* 中的评分表的数量,记为 sizeof(*CT*).

因为评分表图允许多重边和回路的存在,必须对候选评分表连接树的大小进行限制,否则将会产生过多的候选评分表连接树.SEEKER 用宽度优先搜索(BFS)算法来生成候选评分表连接树集合.

算法 **1**. 候选评分表连接树集合 *CTSet* 的生成算法(BFS 算法).

输入:评分表图 G_{ST},候选评分表连接树最大允许大小 Max*CTSize*,关键词查询 $Q(kw_1,\ldots,kw_m)$.

输出:候选评分表连接树集合 *CTSet*.

V:队列,用来存储已经访问过的评分表.

Q:队列,用来存储中间结果.

Begin

1 **for** 评分表图 G_{ST} 中的每一个评分表 ST_i,i=1,…,n **do**

2 将 ST_i 插入 *V* 的队尾

```
3    将 ST_i 插入 Q 的队尾
4    while Q 不为空 do
5      从 Q 的队头取出一个中间结果 CT
6      if CTSet 中没有 CT then
7        if CT 满足定义 3 then 将 CT 加入 CTSet 中
8        if CT 的大小<MaxCTSize then
9          for CT 中的每一个评分表 ST_j do
10           for ST_j 上的每一条边连接的邻居评分表 AdjST_k do /*存在多重边*/
11             if CT 中没有 AdjST_k AND V 中没有 AdjST_k then
12               将 CT 复制到 CT′,将 AdjST_k 连接到 CT′上,然后将 CT′插入 Q 的队尾
End
```

算法 1 产生的候选评分表连接树集合 *CTSet* 是一个有序数组,排序的规则是:含关键词数量多的候选评分表连接树排在前面;如果关键词数量相同,较小的候选评分表连接树排在前面.可以证明,算法 1 满足以下定理:

定理 **1**(完全性和无重复性). 算法 1 生成的候选评分表连接树集合 *CTSet* 里无重复地包含所有满足条件 sizeof(*CT*)≤Max*CTSize* 的候选评分表连接树 $CT \in G_{ST}$(证明略).

查询实例生成的候选评分表连接树有 $\text{AUTHOR}^{\{\text{“Jim Gray”}\}} \triangleright\triangleleft \text{WRITE}^{\{\}} \triangleright\triangleleft \text{PAPER}^{\{\text{“Transaction”\&\&“Year:>=1990”}\}}$, $\text{PAPER}^{\{\text{“Transaction”\&\&“Year:>=1990”}\}}$,$\text{AUTHOR}^{\{\text{“Jim Gray”}\}} \triangleright\triangleleft \text{WRITE}^{\{\}} \triangleright\triangleleft \text{PAPER}^{\{\text{“Transaction”}\}}$等 13 个(Max*CTSize*=4 时).

3.4 Top-*k*查询结果生成

Top-*k* 查询结果生成模块的主要功能是将候选评分表连接树转换为 SQL 查询,从数据库中查询出数据,将 *k* 个得分最高的结果返回给用户.

3.4.1 评分公式(scoring function)

SEEKER 只将 Top-*k* 查询结果返回给用户,因此需要给查询结果评分,然后根据分数高低来对查询结果排序.查询结果所含关键词的数量不尽相同,显然,含关键词越多,得分应该越高.SEEKER 采用了以下的评分公式:

$$Score(TT,Q)=\frac{1}{\text{sizeof}(TT)}\cdot\left(\frac{m'}{m}\right)^{a}\cdot\sum_{i=1}^{\text{sizeof}(TT)}\sum_{j=1}^{m}Score(T_i,kw_j) \tag{5}$$

其中,*TT* 是构成查询结果的一个连接元组树;sizeof(*TT*)是 *TT* 中所含元组的个数,它与该结果的得分成反比;T_i 是 *TT* 中的元组;*Q* 是一个关键词查询,*m* 是 *Q* 中关键词的个数,$kw_j \in Q$;*m*′是 *TT* 中所含关键词的数量;*a* 是常数,取大值(例如 10)来保证含关键词多的查询结果比含关键词少的查询结果的得分高;$Score(T_i,kw_j)$是元组 T_i 对于关键词 kw_j 所得的分数,$\sum_{j=1}^{m}Score(T_i,kw_j)$ 是元组 T_i 所在评分表的 SCORE 列的值.

推论 **1**(查询结果比较). 对关键词查询 *Q* 产生的两个查询结果 TT_1 和 TT_2,如果 TT_1 所含的关键词数量小于 TT_2 所含的关键词数量,则 $Score(TT_1,Q)<Score(TT_2,Q)$.

DBXplore[5]和 DISCOVER[6]采用了简单的评分策略:查询结果包含的元组越少越好.IR-Style[7]的评分公式不仅考虑了查询结果的大小,还考虑了查询结果和关键词查询的相关性,但由于要求查询结果包含全部关键词,因此没有考虑含部分关键词的查询结果的评分.BANKS[4]和 ObjectRank[9]采用了类似 PageRank 的评分方法,PageRank 对于网页排序是非常有效的,但对关系数据库中的元组排序的有效性还有待验证.

3.4.2 Top-*k* 查询结果生成算法

候选评分表连接树 $CT(ST_1,\ldots,ST_n)$ 的最大可能分数是组成它的所有评分表的最大分数之和,即 $\text{Max}Score(CT,Q)=\sum_{i=1}^{n}\max\{\pi_{score}(ST_i)\}$,$\pi$是投影关系运算.在查询过程中,假设当前的 Top-*k* 查询结果中的第 *k* 个结果的分数为 $Score_k$,如果 $\text{Max}Score(CT,Q)\le Score_k$,那么 *CT* 可以安全地移除而不会对 Top-*k* 查询结果有任何影响.为了提高连接操作的效率,应该尽可能地减少参与连接操作的元组数.对参与连接的某一个评分表 ST_j 来说,要产生比 $Score_k$ 得分更高的查询结果,只有满足以下条件的行 *row* 才有可能:$\pi_{score}(row) > Score_k -$

$\sum_{i=1,i\neq j}^{n}\max\{\pi_{score}(ST_i)\}$.同理,对于两个相邻的评分表 ST_p 和 ST_q,它们的行 row_p 和 row_q 应该满足以下条件 $\pi_{score}(row_p)+\pi_{score}(row_q)>Score_k-\sum_{i=1,i\neq p,q}^{n}\max\{\pi_{score}(ST_i)\}$.这些条件都被加入 SQL 查询中来以加快查询速度.查询结果根据分数高低来排序.

算法 **2**. Top-k 查询结果生成.

输入:有序的候选评分表连接树集合 $CTSet$,关键词查询 $Q(kw_1,\ldots,kw_m)$.

输出:Top-k 查询结果.

$Results[1,\ldots,k]$:有序数组,用来存储查询结果,数组里的查询结果按分数高低来排序.

$Results[k].kws$:$Results[k]$所含 Q 中关键词的数量,查询结果数<k 时,为 0.

$Results[k].score$:$Results[k]$的分数,查询结果数<k 时,为 0.

Begin

1 **for** 候选评分表连接树集合 $CTSet$ 中的每一个候选评分表连接树 $CT_i,i=1,\ldots,n$ **do**

2 **if** CT_i 所含关键词数量<$Results[k].kws$ **then** 算法结束,返回 Results /*推论 1*/

3 **if** CT_i 所含关键词数量=$Results[k].kws$ **AND** CT_i 的最大可能分数≤$Results[k].score$ **then goto** 1

4 将 CT_i 转换为 SQL 查询

5 执行 SQL 查询 /*查询结果按分数高低排序*/

6 **while** 查询结果不为空 **do**

7 取出查询结果 TT

8 **if** $Score(TT,Q)>Results[k].score$ **then** 将 TT 插入 Results

9 **else goto** 1

10 返回 Results

End

表 1 是 SEEKER 给出的查询实例的 Top-5 结果.

Table 1 Top-5 results of query example

表 **1** 查询实例的 Top-5 结果

No.	Score	AUTHOR	WRITE	PAPER
1	8.33	Jim Gray…	…	Database and transaction processing benchmarks. 1992…
2	8.33	Jim Gray…	…	The benchmark handbook for database and transaction systems (1st Edition). 1991…
3	8.33	Jim Gray…	…	The benchmark handbook for database and transaction systems (2nd Edition). 1993…
4	8.33	Jim Gray…	…	Database and transaction processing performance handbook. 1993…
5	8.33	Jim Gray…	….	Transaction processing: Concepts and techniques. 1993…

4 实 验

4.1 实验设置

SEEKER 是在 Oracle 9i 上开发的,采用 JAVA 语言编写,语法分析部分用 JAVACC 编写,和 Oracle 的接口是 JDBC.我们用实验评测了 SEEKER,评测是在 DBLP[15]数据集上进行的.我们将 DBLP 网站提供的 XML 文件按照图 5 所示的模式分解成 4 个关系:AUTHOR 中有 294 062 个元组;PAPER 中有 446 409 个元组;WRITE 中有 1 000 099 个元组;CITE 中有 111 357 个元组.AUTHOR 和 PAPER 上建立了全文索引,所有的外码和数字属性上也都建立了索引.实验是在一台有 512M 内存的 Pentium IV 2.5 GHz 的微机上进行的,操作系统是 Windows XP.

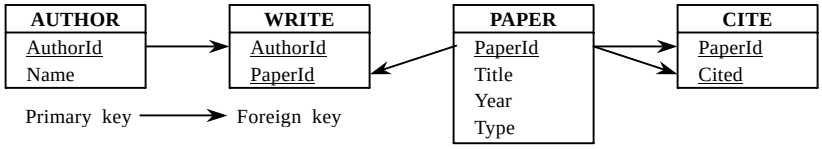


Fig.5 The DBLP database schema graph

图 5 DBLP 数据库模式图

4.2 查询效果

本节用查准率(precision)和查全率(recall)来衡量 SEEKER 的查询效果,查准率和查全率的计算公式如下:

$$Precision = \frac{|R_a|}{|A|} \tag{6}$$

$$Recall = \frac{|R_a|}{|R|} \tag{7}$$

其中,$|R_a|$是返回的查询结果集中与查询相关的连接元组树的数量;$|A|$是查询结果集中的连接元组树的数量;$|R|$是数据库中与查询相关的连接元组树的数量.

4.2.1 元数据对查询效果的影响

我们构造了一个包含 10 个 SQL 查询的测试集,其中的 SQL 语句都是查询某个作者发表的关于某个主题的论文.在实验中,我们用关键词查询来完成 SQL 查询同样的任务,以测试集的查询结果作为基准来计算关键词查询的查准率和查全率,即将 SQL 查询的结果作为$|R|$.表 2 和表 3 是根据测试集中的 SQL 查询给出的两组关键词查询,表 2 用第 1 类关键词表述,表 3 用第 1 类和第 2 类关键词共同表述.

Table 2 Keyword queries without metadata

表 2 不考虑元数据的关键词查询

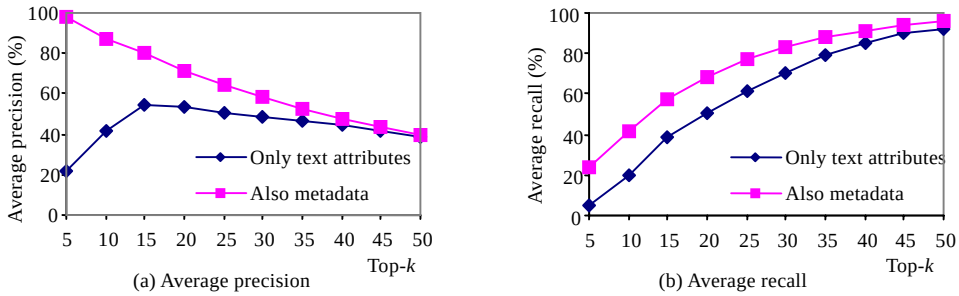
Knuth algorithm	Dijkstra programming
Turing article	"Jim Gray"Article
Codd relational	Salton "Information Retrieval"
Fagin proceeding	"C. J. Date" database
Hartmanis NP	"Marvin Minsky" article

Table 3 Keyword queries with metadata

表 3 考虑元数据的关键词查询

Author: Knuth algorithm	Author: Dijkstra programming
Author: Turing article	Author: "Jim Gray" article
Author: Codd relational	Author: Salton "Information Retrieval"
Author: Fagin proceeding	Author: "C. J. Date" database
Author: Hartmanis NP	Author: "Marvin Minsky" article

分别执行测试集中的 SQL 查询和两组关键词查询.图 6 给出了 Top-k 取不同值时,两组关键词查询的平均查准率和平均查全率.从图中可以看出,实现对元数据的关键词查询,提高了查准率和查全率,尤其是在 Top-k 较小的时候更加明显.因此,第 2 类关键词可以帮助用户更精确地表达查询要求,减少查询模糊性.



(a) Average precision (a) 平均查准率 (b) Average recall (b) 平均查全率

Fig.6 Metadata's impact on query effectiveness

图 6 元数据对查询效果的影响

4.2.2 数字属性对查询效果的影响

我们构造了一个包含 10 个 SQL 查询的测试集,测试集中的 SQL 语句都是查询某个作者在某段时间内发表的关于某个主题的论文,将 SQL 查询的结果作为$|R|$.表 4 和表 5 是根据测试集中的 SQL 查询给出的两组关键词查询,表 4 用第 1 类关键词表述,表 5 用第 1 类和第 3 类关键词共同表述.

分别执行测试集中的 SQL 查询和两组关键词查询.图 7 给出了 Top-k 取不同值时,两组关键词查询的平均查准率和平均查全率.实验结果证明,采用第 3 类关键词可以提高查准率和查全率,尤其是在 Top-k 较小的时候更为明显.第 3 类关键词增强了关键词查询的表达能力,将数据库中的数字属性纳入到关键词查询的范围内,帮助用户更精确地表达查询要求.

以上两个实验证明了 SEEKER 对关键词查询语言的扩展是有效的.

Table 4 Keyword queries over text attributes

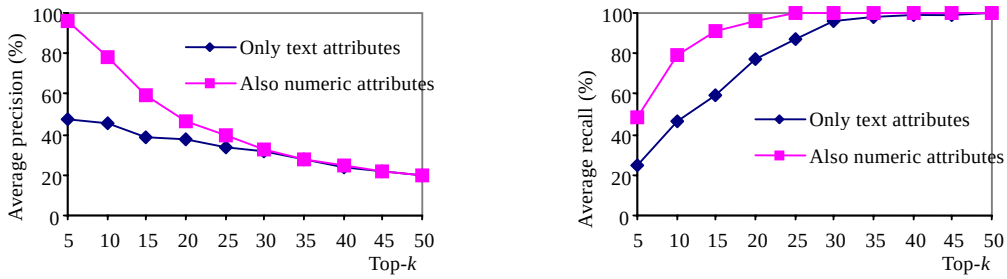
表 4 只考虑文本属性的关键词查询

Dewitt database	Foster grid
Molina 'peer-to-peer'	Stonebraker object
'Jennifer Widom' database	
'Jiawei Han' 'Data Mining'	
Bernstein concurrency	
Ullman database	
'Rakesh Agrawal' 'Data Mining'	
'Juris Hartmanis' complexity	

Table 5 Keyword queries over text and numeric attributes

表 5 考虑文本和数字属性的关键词查询

Dewitt database year:>1990	Foster grid year:=2003
Molina 'peer-to-peer' year:>=2002	Stonebraker object year:>1993
'Jennifer Widom' database Year:>1995	
'Jiawei Han' 'Data Mining' Year:>=2000	
Bernstein concurrency year:>=1970 year:<=1985	
Ullman database year:>1980 Year:<2000	
'Rakesh Agrawal' 'Data Mining' year:>=1997	
'Juris Hartmanis' complexity year:>1990	



(a) Average precision (a) 平均查准率 (b) Average recall (b) 平均查全率

Fig.7 Numeric attribute's impact on query effectiveness

图 7 数字属性对查询效果的影响

4.3 各参数对查询响应时间的影响

以下 3 个实验的目的是分析 SEEKER 的几个主要参数对查询响应时间的影响,并比较查询过程中各模块耗费的时间.实验中所用关键词都是从 DBLP 数据库中随机抽取出来的.

4.3.1 候选评分表连接树最大允许大小 Max*CTSize* 对查询响应时间的影响

图 8 显示了候选评分表树最大允许大小对查询响应时间的影响,响应时间是 500 次查询的平均值,查询的关键词数量为 2,Top-k 的 k 值为 10.从图中可以看出,随着 Max*CTSize* 的增大,查询响应时间也随之增加,这主要是因为候选评分表树数量增大的缘故.我们注意到,当 Max*CTSize* 分别为 1 和 2,3 和 4,5 和 6 时,响应时间几乎相同,原因是对 DBLP 数据库,算法 1 生成的候选评分表连接树集合 *CTSet* 几乎是相同的.

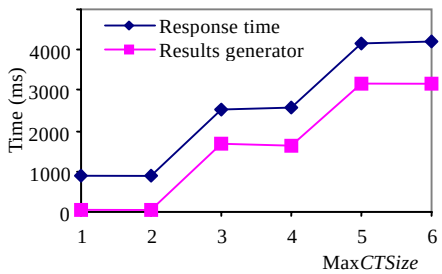


Fig.8 Effect of maximum allowed *CT* size

图 8 候选评分表树最大允许大小的影响

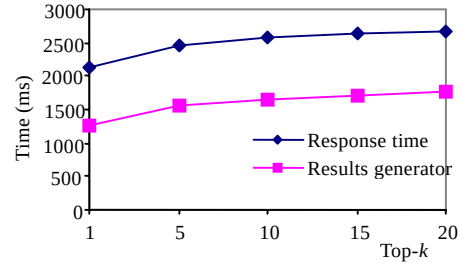


Fig.9 Effect of Top-k

图 9 Top-k 的 k 值的影响

4.3.2 Top-*k* 的 *k* 值对响应时间的影响

图 9 是 Top-*k* 的 *k* 值对查询响应时间的影响,响应时间是 500 次查询的平均值,查询的关键词数量为 2,候选评分表树最大允许大小为 4.*k* 值的增大使得查询时间随之增大,主要原因是用户要求的查询结果越多,算法 2 需要执行的 SQL 查询也越多.

4.3.3 查询关键词数量对响应时间的影响

图 10 是关键词数量对查询响应时间的影响,响应时间是 500 次查询的平均值,候选评分表树最大允许大小为 4,Top-*k* 的 *k* 值为 10.从图中可以看出,随着关键词数量的增加,响应时间迅速增加.关键词数量对响应时间的影响要更大一些,这是因为关键词数量的增加使得基本评分表和评分表的数量也随之增加,在最坏的情况下呈指数增长,使得候选评分表树数量迅速增加,造成查询时间的迅速增加.

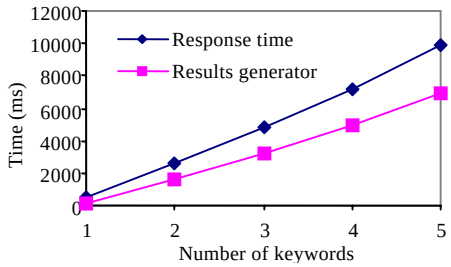


Fig.10 Effect of number of keywords

图 10 关键词数量的影响

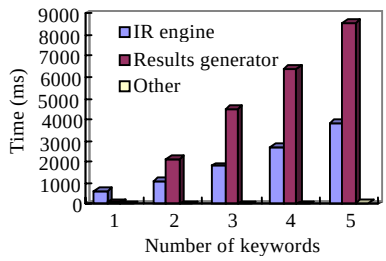


Fig.11 Time consumed by different query steps

图 11 查询过程各阶段耗费时间比较

4.3.4 查询过程中各模块耗费时间的分析比较

图 10 和图 11 是同一个实验的结果.从图中可以看出,查询过程中最耗费时间的模块是 Top-*k* 查询结果生成,其次是 IR 引擎.这两个模块都包含对数据库的大量查询和操作,其他模块的工作主要在内存中进行,很少访问数据库,耗费的时间与之相比可以忽略不计.显然,要想进一步减少查询响应时间,应该在这两个模块上下功夫.

5 结论和未来工作

本文描述了一个基于关键词的关系数据库信息检索系统 SEEKER 的设计和实现.SEEKER 不仅可以检索关系数据库里的文本属性,还可以检索数据库的元数据以及数字属性.SEEKER 虽然是在 Oracle 9i 上实现的,但经过简单修改,就可以用于其他关系数据库.

我们计划在未来的工作中完成以下工作:提高算法效率,使系统能够更快地响应用户的查询;实现对数字属性的模糊查询,例如用“Jim Gray Transaction Year:1990”这样的查询来获取 Jim Gray 在 1990 年左右发表的关于 Transaction 的论文,论文发表的年代越接近 1990 年,排名应该越靠前.

References:

[1] Oracle text. 2004. http://otn.oracle.com/products/text/index.html

[2] DB2 text information extender. 2004. http://www-3.ibm.com/software/data/db2/extenders/textinformation/index.html

[3] SQL server full-text query. 2004. http://msdn.microsoft.com/library/default.asp?url=/library/en-us/architec/8_ar_sa2_0ehx.asp

[4] Halotia G, Hulgeri A, Nakhey C, Chakrabarti S, Sudar-Shan S. Keyword searching and browsing in databases using BANKS. In: Agrawal R, *et al*., eds. Proc. of the 18th Int'l Conf. on Data Engineering. San Jose: IEEE Press, 2002. 431−440.

[5] Agrawal S, Chaudhuri S, Das G. DBXplorer: A system for keyword-based search over relational databases. In: Agrawal R, *et al*., eds. Proc. of the 18th Int'l Conf. on Data Engineering. San Jose: IEEE Press, 2002. 5−16.

[6] Hristidis V, Papakonstantinou Y. DISCOVER: Keyword search in relational databases. In: Bernstein PA, *et al*., eds. Proc. of the 28th Int'l Conf. on Very Large Data Bases. Hong Kong: Morgan Kaufmann Publishers, 2002. 670−681.

[7] Hristidis V, Gravano L, Papakonstantinou Y. Efficient IR-style keyword search over relational databases. In: Freytag JC, *et al*., eds. Proc. of the 29th Int'l Conf. on Very Large Data Bases. Berlin: Morgan Kaufmann Publishers, 2003. 850−861.

[8] Su Q, Widom J. Indexing relational database content offline for efficient keyword-based search. Technical Report, Stanford: Stanford University, 2003. http://dbpubs.stanford.edu/pub/2003-13

[9] Balmin A, Hristidis V, Papakonstantinou Y. ObjectRank: Authority-Based keyword search in databases. In: Nascimento MA, *et al*., eds. Proc. of the 30th Int'l Conf. on Very Large Data Bases. Toronto: Morgan Kaufmann Publishers, 2004. 564−575.

[10] Goldman R, Shivakumar N, Venkatasubramanian S, Garcia-Molina H. Proximity search in databases. In: Gupta A, *et al*., eds. Proc. of the 24th Int'l Conf. on Very Large Data Bases. New York: Morgan Kaufmann Publishers, 1998. 564−575.

[11] Florescu D, Kossmann D, Manolescu I. Integrating keyword search into XML query processing. In: Albert V, *et al*., eds. Proc. of the 9th Int'l World Wide Web Conf. Amsterdam: ACM Press, 2000. 119−136.

[12] Hristidis V, Papakonstantinou Y, Balmin A. Keyword proximity search on XML graphs. In: Dayal U, *et al*., eds. Proc. of the 19th Int'l Conf. on Data Engineering. Bangalore: IEEE Press, 2003. 367−378.

[13] Guo L, Shao F, Botev C, Shanmugasundaram J. XRANK: Ranked keyword search over XML documents. In: Halevy AY, *et al*., eds. Proc. of the 2003 ACM SIGMOD Int'l Conf. on Management of Data. San Diego: ACM Press, 2003. 16−27.

[14] Baeza-Yates R, Riberio-Neto B. Modern Information Retrieval. New York: Addison-Wesley, 1999. 29.

[15] DBLP bibliography. 2004. http://www.informatik.uni-trier.de/~ley/db/index.html