ISSN 1000-9825, CODEN RUXUEW
Journal of Software, Vol.21, No.10, October 2010, pp.2454−2476
doi: 10.3724/SP.J.1001.2010.03913

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel/Fax: +86-10-62562563

基于关系数据库的关键词查询*

林子雨[1+], 杨冬青[2], 王腾蛟[2], 张东站[1]

[1](厦门大学 计算机科学系,福建 厦门 361005)

[2](北京大学 信息科学技术学院,北京 100871)

Keyword Search over Relational Databases

LIN Zi-Yu[1+], YANG Dong-Qing[2], WANG Teng-Jiao[2], ZHANG Dong-Zhan[1]

[1](Department of Computer Science, Xiamen University, Xiamen 361005, China)

[2](School of Electronics Engineering and Computer Science, Peking University, Beijing 100871, China)

+ Corresponding author: E-mail: ziyulin@xmu.edu.cn

Lin ZY, Yang DQ, Wang TJ, Zhang DZ. Keyword search over relational databases. *Journal of Software*, 2010,21(10):2454−2476. http://www.jos.org.cn/1000-9825/3913.htm

Abstract: First, the research background of keyword search over relational databases is presented and is followed by a detailed description of two solutions to this problem, i.e., data graph based and schema graph based methods, and a discussion of the principles, advantages and disadvantages of these methods is also mentioned. Finally, some future trends in this area are discussed.

Key words: keyword search; relational database; information retrieval; data graph

摘 要: 介绍了基于关系数据库的关键词查询问题的研究背景;阐述了解决该问题的两大类方法,即基于数据图的方法和基于模式图的方法,并详细介绍了各种方法的原理以及各自的优缺点;最后展望了未来的研究方向.

关键词: 关键词查询;关系数据库;信息检索;数据图

中图法分类号: TP311 **文献标识码**: A

数据库(database,简称 DB)已经广泛地应用于人们的生产和生活中,它可以高效地支持结构化数据的存储和查询.关系数据库是当前数据库的主流形式,它采用结构化查询语言进行内容检索,并要求用户掌握一定的查询语言和数据库模式知识.与此相反,目前蓬勃发展的互联网中的信息检索(information retrieval,简称 IR)则采用了另一种完全不同的、属于 IR 风格的内容检索方式,即关键词查询(查询通常是数据库的专用术语,但是与大多数其他研究一样,本文将混用查询和搜索这两个术语).在这种查询方式中,只要用户输入关键词,网页就会为用户返回包含该关键词的相关结果.结构化查询支持针对结构化数据的高效检索,并具备了完善的查询优化技术.关键词查询则具有简便、易用的特点,支持针对文本文档的快速检索.二者在各自的应用领域都取得了极大的成功.

随着互联网的发展,越来越多的普通用户需要访问在线数据库,这些用户通常不具备查询语言和数据库模

* Supported by the National Natural Science Foundation of China under Grant No.50604012 (国家自然科学基金); the National High-Tech Research and Development Plan of China under Grant No.2009AA01Z150 (国家高技术研究发展计划(863))

Received 2009-09-22; Accepted 2010-07-19

式知识[1].同时,关系数据库里存储了越来越多的文本数据,企业需要实现文本数据和结构化数据的无缝集成.由此就产生了一个很自然的需求,即让关系数据库支持高效的关键词查询.目前,这一方面的研究已经成为数据库领域的热点,具有广阔的应用前景.通过基于关键词的查询,企业可以建立针对大规模数据的、快速而便捷的信息发布和搜索方式,让企业用户无技术障碍地访问企业内部各种关系型数据,帮助企业更好地利用数据产生价值.

本文第1节介绍基于关系数据库的关键词查询问题的相关研究背景.第2节详细阐述解决基于关系数据库的关键词查询问题的两大类方法,即基于数据图的方法和基于模式图的方法.第 3 节进行总结并展望未来的研究方向.

1 基于关系数据库的关键词查询问题概述

这部分内容首先对 IR 领域的关键词查询和 DB 领域的结构化查询进行描述,然后指出二者结合的必要性以及存在的困难与挑战,最后概括基于关系数据库的关键词查询的研究历史与现状.

1.1 结构化查询和关键词查询

关系数据库通常使用 SQL(structured query language)语言进行结构化查询,用户需要在 SQL 语句中指定要查询的列,系统会把该列的内容与查询的关键词进行匹配,并最终返回结果.下面我们介绍一个关于结构化查询的实例.

如图1所示,数据库中包含4个表,分别是*Author*,*Paper*,*Citation* 和 *Paper-Author*.其中:*Author* 表记录了作者的标识(*AID*)和姓名(*Name*);*Paper* 表记录了论文的标识(*PID*)和标题(*Title*);*Paper-Author* 表记录了论文(*PID*)和作者(*AID*)之间的对应关系,*PID* 和 *AID* 都是外键,分别引用了 *Paper* 表的 *PID* 属性和 *Author* 表的 *AID* 属性;*Citation* 表记录了引用论文(*Cite*)和被引用论文(*Cited*)之间的对应关系,*Cite* 和 *Cited* 也都是外键,都引用了 *Paper* 表的 *PID* 属性.

Author

AID	*Name*
a_1	Tom
a_2	Jack

Citation

Cite	*Cited*
t_1	t_2
t_2	t_3
t_4	t_3
t_6	t_5

Paper

PID	*Title*
t_1	XML database
t_2	Database optimization
t_3	SQL language
t_4	Data mining
t_5	Keyword search over database
t_6	Proximity search over database

Paper-Author

PID	*AID*
t_1	a_1
t_2	a_1
t_3	a_1
t_5	a_1
t_5	a_2
t_6	a_2

Fig.1 An example of database

图1 一个数据库实例

下面是一个采用 SQL 语句书写的结构化查询:

```
SELECT*FROM Paper P
WHERE CONTAINS (P.title,'database',1)>0
ORDER BY score(1) DESC
```

如果在 ORACLE 9.1 上执行这个查询,系统会访问 *Paper* 表中的行,并使用关键词 database 在 *Title* 字段上进行匹配,然后根据积分对匹配结果进行排序,最终返回结果.很显然,这个过程需要查询指定某些列进行关键词匹配.从普通用户角度而言,这种方法不仅显得复杂,而且灵活性也不强.因为在某些时候,可能需要对多个表进行连接操作才能得到结果,让用户自己去了解每个表和列的作用是比较困难的.比如,如果我们要查询包含关键词 keyword search by Jack 的记录,那么就需要对 *Paper* 表、*Author* 表和 *Paper-Author* 表进行连接操作,才能得到查询结果.

很显然,由上面的例子我们可以得知,如果想要利用结构化查询从关系数据库中获得满意的结果,就需要用

户熟悉结构化查询语言以及数据库模式的知识,这对大多数普通用户而言都是一件比较困难的事情.与此相反,关键词查询则不需要用户了解这些专业化的知识,用户只需给出一个关键词集合 $K=\{k_1,k_2,\ldots,k_m\}$,系统就会返回包含关键词的查询结果.这种方法简单、易用,在互联网世界中表现出了强大的生命力,获得了用户的广泛认可.因此,在用户越来越需要在线访问关系数据库的今天,结构化查询已经不能很好地满足用户的要求,在关系数据库中引入关键词查询则具有很大的必要性和重要性.

1.2 关系数据库和关键词查询结合的必要性

对于企业应用而言,大都同时需要结构化数据和文本文档.因此,两种信息的融合就成为一个核心的问题.可以说,结构化数据和文本的无缝集成是许多企业的美好愿望,并且企业可以从中获得实实在在的收益.而结构化数据和文本的无缝集成,是通过关系数据库和关键词查询技术的相互结合来实现的.没有二者的有效集成,就无法实现针对结构化数据和文本的自由查询,这是由二者各自的优缺点所决定的,具体如下:

(1) 关系数据库

目前,许多成熟的商业数据库系统(比如 Microsoft SQL server,Oracle 和 IBM DB2)都使用一个扩展功能来处理文本文档搜索.但是,这种文本扩展功能存在一些不足,主要表现在两个方面:第一,这种功能通常是面向单个文本属性的.例如,Oracle 9i Text[2],IBM DB2 Text Information Extender[3]和 Microsoft SQL Server 2000[4]都可以利用标准的 SQL 语句为关系表中的文本属性创建全文索引,并使用 *contain*(*A*,*k*)函数返回关键词查询结果.其中,*A* 是属性名称,*k* 是一个用户给定的关键词.但是,在关系数据库中,由于数据库的规范化处理,回答关键词查询所需要的信息通常分散在多个表中,需要对多个表进行连接操作才能得到查询结果.因此,这种针对单个文本属性创建的全文索引,无法有效支持针对全数据库的关键词查询;第二,这种文本扩展功能通常是一个独立的引擎,并不是真正地和数据库系统集成在一起.在 API(application programming interface)层面,针对文本的查询谓词与 SQL 相比,二者遵循不同的概念和语法,系统有时甚至需要用户使用特定的语法来引导查询处理器如何使用索引和其他优化方法.实际上,如果系统支持真正的数据独立性,这些工作对于用户而言应该是不可见的.另外,作为数据库的扩展功能,由于缺乏先进的相关性排序机制,它们也很难提供灵活的评分(score)和排序(rank)功能.因此,在这种平台基础上构建一个灵活的、可定制的企业网搜索系统几乎是不可能的[5].

(2) 关键词查询

互联网搜索引擎(比如 Google 和百度)普遍采用了关键词查询技术,它为文本搜索提供了强大的功能和灵活性,也取得了巨大的商业化成功.这些搜索引擎主要面向文档,不过,最近这些系统也开始更多地关注结构化数据,甚至还提供了一些对 XML(extensible markup language)的支持.但是,这类 IR 系统也存在两个主要不足:第一,这类 IR 系统只能提供有限的结构化数据查询能力[1].在互联网中,为了实现针对后端数据库的有限查询,许多 Web 站点或者把数据库中的数据导出成 HTML(hypertext markup language)文档,或者使用表单(或 HTML 模板)来查询数据库.对于前者,必须要求数据是静态的,一旦数据发生更新,就要重新为数据库中的新数据生成新的 HTML 文档,这个维护工作很繁琐.对于后者,给用户和开发者都带来了不小的麻烦.这种做法的另一个明显缺点就是,它会导致数据库模式所包含的语义信息的丢失.因此可以说,这类 IR 系统所能提供的结构化数据查询能力是非常有限的;第二,这类 IR 系统通常缺乏数据库系统所具备的查询优化机制[5].因此,开发者经常需要考虑采用何种比较有效的搜索策略,或者依赖那些具有次优性能的默认策略.当应用涉及连接操作或更加复杂的查询,而且又需要同时访问结构化数据和文本时,这个问题就变得更为严峻.

总的来说,关系数据库技术缺少对文本的足够支持,关键词查询技术没有针对高级查询的优化机制.二者各有所长,也各有所短,而且存在很强的互补性.因此,非常有必要设计一个平台来对关系数据库和关键词查询技术加以整合,使其具有灵活和开放的评分和排序方法,支持对文本和结构化数据的自由搜索.学术界和业界都已经广泛认识到,信息检索(IR)和数据库(DB)的集成将会给用户带来更多的高质量的服务.

1.3 困难与挑战

IR 的目标是,从文本数据库中寻找与给定关键词相关的文档.而对于基于关系数据库的关键词查询而言,目

标就不仅仅是寻找包含给定关键词的相关文档或文档片段,而是要发现关键词之间的语义关系.这是由关系数据库不同于文本数据库的特点所决定的,也正是由于二者的区别,导致了关系数据库和关键词查询技术的集成会面临以下主要挑战:

(1) 如何发现关键词之间的语义关系:满足用户要求的答案并不只是来自单个元组,很可能是由来自多个表的多个元组的连接得到的,这些元组构成一个元组连接树(参见本文第 2.2.2.2 节中的定义 9),这棵树描述了关键词之间的语义关系.但是,在关系数据库中寻找这些元组连接树并不容易.由于数据库的规范化,信息的逻辑单元可能被分片存储到不同的物理表当中.对于一个给定的关键词集合,可能需要对多个关系表进行即席连接操作才能得到匹配的行集,即包含关键词的元组连接树.仅从这一点而言,基于关系数据库的关键词查询和基于文档的关键词查询就存在很大的不同,前者在搜索时每次要处理一个或多个表中的多个属性,而后者在搜索时每次只需处理一个文档.因此,我们不能把文档搜索中的成熟技术直接移植到关系数据库中.由于搜索结果来自于关系数据库的多个元组,一个结果反映了不同元组之间的相互关联,因此,文献[6]把这种问题称为“结构化关键词查询”,而把数据库自身提供的针对单文本属性的搜索,称为“全文关键词查询”;

(2) 如何得到最相关的结果:一个用户查询的结果可能包含了大量元组连接树,为了评估它们与给定查询的相关性,就需要为每个元组连接树单独评分,这些评分可以把最相关的结果排在尽可能高的位置.在文本数据库中,用户搜索的基本信息单元是文档,对于一个关键词查询,IR 系统为每个文档计算一个评分,然后根据评分对文档进行排序,排在最前面的文档就会作为结果返回给用户.但在关系数据库中,信息的存储形式是表和列,以及主外键关联.用户所需答案的逻辑单元,不仅仅局限于单个列上的值,或单个元组,它还可能是由多个元组连接得到的.因此,就需要为每个元组连接树单独评分;

(3) 如何处理结果中的重复和冗余信息:关系数据库比文本数据库具有更加丰富的结构,容易产生信息重复和冗余问题,系统生成的搜索结果中的重复冗余信息会使用户感到困惑.

挑战可能不止上述几个方面,但是,即使对于以上几个问题,现有的系统中大多数也都只是解决了一部分,尤其是信息冗余问题,只有少数研究[1,7]提出初步的解决方案.

1.4 研究历史与现状

对信息检索和数据库这两者进行结合比较早的研究工作是文献[8],作者把数据库看成一个图,元组作为图中的节点,元组之间的关系作为图中的边.一个用户查询指定两个对象集合——*Find* 集合和 *Near* 对象集合,它们可以通过使用两个不同的关键词集合来得到.然后,系统就会根据 *Find* 集合中对象到 *Near* 集合中对象的距离来对 *Find* 集合中的对象进行排序.

文献[8]发表于 1998 年,当时的网络并没有在世界范围内普及,网络应用也比较有限.因此,这类研究并没有引起学术界的共鸣,此后几年没有出现多少相关的研究.在世纪之交,互联网迅速发展,人们开始大范围接触各种互联网应用.尤其是互联网搜索引擎的帮助,使得用户可以简单、方便地获得大量的信息,让用户感受到网络的巨大价值.到了 2002 年,互联网和数据库都已经发展到非常成熟的阶段.这时的用户已经不满足于只从互联网文档内容中获得信息,而是希望能够以类似的方式在数据库中进行检索.现实的需求促成了针对关系数据库的关键词查询的研究,DBXplorer[9],DISCOVER[10]和 BANKS[1]等具有开创意义的成果都是于 2002 年发表的.此后,在国际顶级学术会议(VLDB,SIGMOD,ICDE 等)上,大量研究不断涌现[6,11−22].

对于关系数据库的关键词查询,绝大多数研究都可以归为两大类,即基于数据图的方法和基于模式图的方法.由于关系数据库、XML 数据都可以建模成数据图,因此,基于数据图的方法既可以处理针对关系数据库的关键词查询,也可以处理针对 XML 数据的关键词查询.

在研究初期,关键词查询的目标是发现数据库元组之间存在的“元组连接树”.随着研究的深入,目前一些研究工作的目标已经转向寻找元组之间更加复杂的结构,比如 *r*-半径 Steiner 树[23]和社区[22],并且有研究开始寻找诸如“频繁共现词语(frequent co-occurring term,简称 FCT)”[24]等与查询关键词密切相关的内容.同时,关键词查询研究的范围也越来越广,扩展到了包括关系数据库、XML[25−27]、数据流[28,29]、分布式数据库[30−32]、空间数据库[33,34]、OLAP[35,36]、工作流[37]和不确定数据[38]等在内的各种环境.

此外,大多数采用关键词查询来访问数据库的方式只采用语法匹配,而没有采用语义匹配,这样就无法充分利用数据之间的语义关系,影响了算法的查全率和查准率.比如,同名异义词的存在会降低结果的查准率,而同义词的存在则会降低结果的查全率.为此,文献[39]把关键词查询方式和基于本体的关系数据库语义检索相结合,设计了一个扩展关系数据库关键词检索系统,取得了较好的查询效果.

这里需要特别指出的是,目前已有一些与本文类似的综述文章[40,41]对该领域的相关研究问题进行了阐述,但是本文内容和已有文献相比仍然具有很大的差别.比如,与 Wang 等人[40]在 2005 年发表的文章相比,本文对问题的讨论更加全面和深入,并介绍了许多 2005 年以后的研究成果;与 Chen 等人[41]在 2009 年 SIGMOD 大会的专题报告相比,本文侧重于讨论基于关系数据库的关键词查询,而前者讨论了结构化数据和半结构化数据(如 XML)的关键词查询,二者在文章组织结构和讨论侧重点上也都截然不同.

2 基于关系数据库的关键词查询

这部分内容首先概括基于关系数据库的关键词查询问题的核心思想;然后详细阐述解决该问题的两大类方法,并对两类方法进行了比较;接下来论述了该问题需要关注的几个方面,包括评分和排序、结果冗余、结果呈现和索引等问题.

2.1 核心思想

目前,基于关系数据库的关键词查询问题已经存在不少具有代表性的研究成果,比如 DBXplorer[9], DISCOVER[10]和 BANKS[1,42]等等.这些研究工作解决问题的方法虽各有千秋,但总的来说,它们的核心思想是一样的,即这些方法都基于图和“简化子树(reduced subtree)”[13]的概念.因为图中的节点和边都关联了文本内容,因此,图就成了关系数据库、半结构化数据和 Web 数据的一种比较好的公共表示形式.关键词查询研究的核心问题就是从数据图中高效地寻找到少量最好的结果树.

定义 1(关键词查询). 形式化地,一个关键词查询就是一个关键词的集合 K.一个关键词查询的结果是给定的数据图 G 的一个子树 T,从而使得 T 是关于给定关键词集合 K 的简化.也就是说,T 包含了 K,但是,不会再有 T 的子树包含 K.

DBXplorer,BANKS 和 DISCOVER 都符合上述思想,并且都采用启发方法来减少搜索空间[15].在这些系统中,一个数据库可以被看成一个数据图 G,图 G 以元组和关键词作为节点.如果两个元组可以通过外键进行连接,那么二者之间就存在一条边.如果元组 t 包含关键词 k,那么 t 和 k 就存在连接.这样,一个关键词查询的结果就是一个图 G 的子树,这个子树是图 G 关于关键词 K 的简化.

不同的研究对简化子树的称呼也不大相同,比如元组连接树(join tree of tuples)[9]、有根有向树(rooted directed tree)[1,12]、元组连接网络(joining network of tuples)[10]和 K-片段(K-fragments)[43]等等.

简化子树可以包含 3 种不同的类型:无向子树、有向子树和强子树(文献[43]中称为“强 K-片段”),其中,强子树是无向子树,并且树中的所有关键词节点都是叶子节点.关键词查询的结果或者是强子树,或者是有向子树.文献[1,12,44]等研究都生成有向子树,而文献[9−11,18,45]等研究则生成无向子树.

另外,关键词查询的结果对用户而言并非具有同等意义.因此,就需要根据重要性对查询结果进行排序.当前面的几个查询结果已经可以满足要求时,查询处理器就应该及时停止查询,而不是继续给出所有的查询结果.这就是 top-k 查询优化工作需要解决的问题.

定义 2(top-k 关键词查询). 一个 top-k 关键词查询 Q 是一个关键词集合 K.一个 top-k 关键词查询的结果是一个由 k 个元组连接树组成的列表 T,并且对于 Q 而言,这 k 个元组连接树的评分 $Score(T,Q)$是最高的.当存在评分平局情况时,可以用任意的方式打破平局.查询结果是根据评分的降序来排列.

文献[11,13,15,18,19,22]等研究工作都关注如何寻找 top-k 个查询结果.

2.2 简化子树生成方法

枚举简化子树的算法对于不同类型的关键词查询方法而言都是非常重要的.在目前已有的研究工作中,简

化子树的生成主要采取两种策略[45]:

(1) 基于数据图的方法:直接对数据图进行处理,从中枚举简化子树;

(2) 基于模式图的方法:利用数据库模式创建连接表达式,然后在 DBMS(data base management system)上执行连接表达式对应的 SQL 语句并得到结果.

基于模式图的方法只能应用于关系型数据的搜索,而基于数据图的方法则可以用于关系型、XML 和 HTML 数据,因为这些数据都可以用数据图的形式来表示.

2.2.1 基于数据图的方法

本节首先简要介绍基于数据图的方法,然后讨论该方法的数据和查询模型、查询过程以及相关研究,最后讨论与该方法相关的几个问题.

2.2.1.1 概 述

许多研究[1,12,13,15,16,18,24,43,45,46]都采用基于数据图的方法来生成简化子树,即直接对数据图进行处理,从中枚举简化子树.在处理关键词查询时,基于数据图的方法主要包括两个步骤:首先把数据库看成一个带权重的数据图,并且假设数据图已经被物化;然后,充分利用数据图中节点(元组)和边(元组之间的主外键关联)的权重来为关键词查询找到 top-k 个代价最小的简化子树.这里,节点/边的权重的分配可以采用文献[17,47]中的方法.

2.2.1.2 数据和查询模型

定义 3(数据图)[43]. 一个数据图 G 包含一个节点集合和一个边的集合.图 G 中存在两种类型的节点,即结构化节点和关键词节点.关键词节点只有入射边,而结构化节点既有入射边也有出射边.因此,一条边不能连接两个关键词.图 G 中存在两种类型的边:一种是前向边(u,v),表示节点 u 和 v 之间存在主外键关联;一种是后向边(v,u),它与边(u,v)相对应,并且只有在图中存在前向边(u,v)时才存在对应的后向边(v,u).一个数据图 G 的边可以具有权重,权重函数 w_G 为每条边 e 分配一个正的权重 $w_G(e)$.数据图 G 的权重,用 $w(G)$表示,就是图 G 中所有边的权重之和.一个数据图 G 是有根的,如果它包含某个节点 r,并且从图 G 中的任何节点都可以通过一条有向路径到达节点 r,这个节点 r 就被称为图 G 的根.

数据图的边具有方向性,可以反映不同方向上连接的强弱,因为两个节点之间的连接在不同方向上的连接强度不是对称的.比如,在反映主外键关联的边中,外键到主键方向的边与其反方向的边具有不同的重要性.

数据图中的节点和边都可以具有权重,这些权重都是预先被赋值的,从而可以更好地支持关键词查询.BANKS[1]引入了节点权重的概念,并借鉴了 Google 的 PageRank[47]思想,即美誉度(prestige)的概念.当某个节点具有更多的指针指向它时,它就具有更高的美誉度.在 BANKS 的实施方案中,节点的美誉度就是它的入度.更高的节点权重意味着更高的节点美誉度.比如,对于一篇论文而言,如果有很多篇论文都指向(引用)它,那么这篇论文显然更加重要.BANKS 综合利用节点权重和边权重来计算树的相关性评分,这一点在第 2.3 节会有论述.在设计简化子树枚举算法时,通常暂时不考虑节点权重,而是只考虑边的权重.之所以使用边权重,是因为简化子树枚举算法中都采用了最短路径的思想,在寻找最短路径时,必须以边权重为基础.这一点在第 2.2.1.3 节有详细论述.对于边的权重而言,更高的权重意味着更弱的相关性.下面来看边的权重是如何计算的.这里介绍的是文献[18]中所采用的方法,其他研究基本上也大同小异.

对于数据图中的节点 v,$N(v)$表示节点 v 的邻居节点集合,$|N(v)|$表示 $N(v)$的尺寸,即集合中元素的个数,可以使用公式(1)来计算一个无向图中的边的权重:

$$w_e((u,v))=\log_2(1+\max\{|N(v)|,|N(u)|\}) \tag{1}$$

这里需要指出的是,在无向图中,$w_e((u,v))=w_e((v,u))$.对于有向图,可以使用公式(2)和公式(3)来为有向图中的边分配权重.对于一个 u 到 v 的主外键关联,(u,v)的权重是公式(2),(v,u)的权重是公式(3):

$$w_e((u,v))=1 \tag{2}$$

$$w_e((v,u))=\log_2(1+N_{in}(v)) \tag{3}$$

其中,$N_{in}(v)$表示引用 v 的节点的数量.公式(1)~公式(3)所包含的语义是,如果一个节点有更多的邻居,那么入射到这个节点的边所反映的元组之间的关系就比较弱.

关系型数据和XML数据都可以用数据图来表示.对于XML数据而言,XML元素就是数据图的结构化节点.而对于关系数据库而言,结构化节点则代表了元组.一个数据库可以被看成一个数据图 G,图 G 以元组和关键词作为节点.如果两个元组可以通过外键进行连接,那么二者之间就存在一条边.如果元组 t 包含关键词 k,那么 t 和 k 就存在连接.图 2 显示了图 1 中的数据库所对应的数据图,这里不包括边的方向和权重.

下面将要给出“基于数据图的方法枚举元组连接树”这个问题的形式化定义.目前,大多数基于数据图的方法[1,12,13,15,18,46,48]都把简化子树枚举问题看成 Steiner 树[49]问题.因此,我们下面来分析这样两个问题:

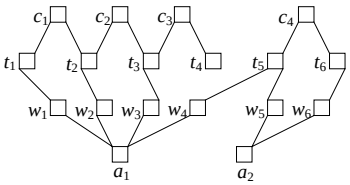


Fig.2 An example of data graph

图 2 一个数据图的实例

- 什么是最小 Steiner 树问题?
- 简化子树枚举问题如何转化成 Steiner 树问题?

首先我们来关注第 1 个问题,即最小 Steiner 树问题.最小 Steiner 树问题的任务如下:对于给定的一个顶点集合 V,使用一个最短的路径把这些顶点连接起来,其中,路径的总长度是各条边的路径的长度之和.Steiner 树问题和最小生成树问题看起来十分相似,但是二者实际上有着本质的不同,根本区别在于:在 Steiner 树问题中,为了减少生成树的路径总长度,可以在图中增加额外的顶点(不属于 V)和边.下面给出最小 Steiner 树问题的形式化定义,它对无向图和有向图都适用.文献[18,48]等给出了 Steiner 树问题的相关定义,这些定义大同小异,我们这里给出的定义以这些文献中的定义作为基础.

定义 4(最小 Steiner 树). 给定一个图 $G(V,E)$,以及 $V'\subseteq V$,如果 T 是图 G 中的一棵连通子树,并且 T 包含了 V'中的所有节点,那么我们就说 T 是图 G 中关于 V'的一棵 Steiner 树.假设用 $c(T)=\sum_{e\in E(T)} w_e(e)$ 表示 T 的代价,其中,$E(T)$表示树 T 中边的集合,$w_e(e)$表示边 e 的权重.如果在所有图 G 中关于 V'的 Steiner 树中,$c(T)$最小,那么就说 T 是一棵最小 Steiner 树.

最小 Steiner 树的一种扩展形式是最小分组 Steiner 树,它可以用来表示基于数据图的关键词查询方法所得到的结果,即图 G 的简化子树,从而把关键词查询问题转化成最小分组 Steiner 树问题.

定义 5(最小分组 Steiner 树). 给定一个图 $G(V,E)$和分组 $V_1,V_2,\ldots,V_n\subseteq V$,如果 T 是一棵最小 Steiner 树,并且 T 中包含了分组 $V_i(1\leqslant i\leqslant n)$中至少一个节点,那么就说 T 是图 G 中关于这些给定分组的最小分组 Steiner 树.

下面我们来解释第 2 个问题,即简化子树枚举问题如何转化成 Steiner 树问题.假定一个查询包含 m 个关键词 $k_1,k_2,\ldots,k_m$.为了回答这个查询,第 1 步工作就是定位与查询关键词匹配的节点,这个可以通过符号表技术[9]或者全文索引技术[10]来获得.一个节点与查询关键词是相关的,如果它所对应的元组的属性值或者元数据(表、列或视图名称)包含了这个关键词.比如,属于关系 *Paper* 的所有元组都被认为是与关键词 PAPER 相关的.对于查询中的每个关键词 k_i,我们需要找到与 k_i 相关的节点集合 S_i.一个查询的答案是一棵有根有向树,并且树中包含了来自每个 S_i 的至少一个节点.很显然,这里的集合 S_i 就对应于定义 4 中的分组 V_i.因此,查询的结果就是一棵最小分组 Steiner 树.简化子树枚举问题,实际上就是最小分组 Steiner 树问题.这里需要指出的是,查询结果树中包含的某些节点也可能不属于任何 S_i,这也正好符合 Steiner 树的特性.因为在最小 Steiner 树问题中,为了减少生成树的路径总长度,可以在图中增加额外的顶点.

最小分组 Steiner 树问题可以给出代价最小(与用户给定的关键词集合 K 相关性最大)的简化子树,当需要返回 top-k 个代价最小的简化子树时,这个问题被称为“top-k 分组 Steiner 树问题”,具体定义如下:

定义 6(top-k 分组 Steiner 树问题). 给定一个关键词查询 $k_1,\ldots,k_m$,为该查询寻找 top-k 个代价最小的分组 Steiner 树 $T_1,T_2,\ldots,T_k$,这些树之间根据代价函数 c 进行排序,并且有 $c(T_1)\leqslant c(T_2)\leqslant\ldots\leqslant c(T_k)$.

通过上面的论述我们可以发现,对于基于数据图的简化子树枚举方法,它与 Steiner 树问题具有相似性.由此,简化子树枚举问题就被转换成了相关的 Steiner 树问题.Steiner 树问题是一个 NP-hard 问题[1],对此,我们很自然地会关心两个问题:第一,如果查询的尺寸是受限的,那么 Steiner 树问题就是容易处理的.此时,查询能够被高效处理吗?第二,为了给 top-k 查询寻找近似的答案,能够使用 Steiner 树的大量近似算法吗?幸运的是,相关研究工作[15]已经证明,这两个问题的答案都是肯定的.由于 Steiner 树[49]问题已经具有多年的研究历史,也产生了丰

富的研究成果[50].因此,可以考虑直接利用已有的算法来解决简化子树枚举问题.

2.2.1.3 查询过程

在许多文献中,分组 Steiner 树问题已经被证明是 NP-hard 问题[49].由于需要计算每棵树的整体相关性,就需要考虑节点的权重,这个问题就变得更加复杂了.对于这个问题,通常采用近似算法,这种算法主要包含 3 类:覆盖和清除(spanning and cleanup)、d^*-Steiner 树(d-star Steiner tree)和 i-层次树(i-level tree).文献[18]中有关于这 3 类算法的简单介绍.目前,不少研究工作(比如 BANKS[1]和 BANKS-II[12])都采用了 d^*-Steiner 树方法.本质上,BANKS 和 BANKS-II 等所寻找到的最小分组 Steiner 树,都是由从树中的叶子节点到根节点的最短路径构成的.

总的来说,给定一个关键词集合,基于数据图方法的查询过程主要包含两个步骤:第 1 步,查找倒排表形式的关键词索引,获得节点 ID.这些节点包含了一个或多个查询关键词,被称为“关键词节点”;第 2 步,运行图搜索算法,寻找到那些连接了上述关键词节点的树(有根树),并且对结果树进行排序.

比较具有代表性的图搜索算法是反向搜索(backward search)[1,12,51]和动态编程方法[18,45].这里主要介绍反向搜索算法,它是由 BANKS 系统最先提出来的,之后的研究(诸如 BANKS-II[12]和 BLINKS[51]等)扩展了该算法.概括地说,反向搜索算法的工作过程如下[51]:

(1) 在反向搜索的任何时刻,令 E_i 表示当前已知的可以到达关键词节点 k_i 的节点集合,其中,E_i 被称为关于 k_i 的簇.

(2) 在最初始阶段,E_i 被定义成直接包含 k_i 的节点集合.这个集合被称为“原始簇”,它的成员节点为关键词节点.

(3) 在每一步的搜索过程中,都从以前访问过的节点,比如说节点 v 开始,选择一条入射边,然后沿着这条边反向访问它的源节点,比如说节点 u.任何包含节点 v 的 E_i,现在都被扩展到节点 u.一旦一个节点已经被访问,那么搜索算法就可以获知它的所有入射边的信息,搜索就可以在以后步骤中访问这些边.

(4) 如果对于每个簇 E_i 都有或者节点 x 属于 E_i,或者存在一条从 x 到 E_i 中某个节点的边,则意味着已经发现了一个答案的根节点 x.

2.2.1.4 相关研究

BANKS 系统使用了一种图搜索算法,即反向扩展搜索(backward expanding search)算法,直接对数据图进行搜索.概括地说,它首先从与关键词匹配的每个节点开始,然后朝着汇合的根部节点向上搜索,每当它发现一个节点已经被每个关键词所到达时,那么就输出一棵结果树.但是,当一个查询关键词匹配量非常大的节点时,或者当算法遭遇一个入度非常大的节点时,反向扩展算法的性能就会非常糟糕.为此,BANKS-II[12]提出了一种新的搜索方法——双向搜索,它允许从可能的根节点出发朝着叶子节点进行前向搜索,从而改进了后向搜索的性能. BANKS-II 把查询结果树看成是路径的集合,每一个关键词对应一条路径,每条路径从根出发到达包含关键词的节点,结果树的边积分就是路径长度的总和.BANKS-II 可以在多项式时间延迟内回答查询,同时可以避免生成大量具有同一个根的相似结果.

BLINKS[51]采用了一个基于代价均衡扩展(cost-balanced expansion)的反向搜索策略,并且使用了额外的双层索引来加快搜索速度.双层索引明显减少了运行最优反向搜索算法的开销,并且可以支持前向搜索,从而有效地实现类似 BANKS-II 中的双向搜索.与 BANKS-II 双向搜索方法相比,BLINKS 在搜索过程中可以实现更大的前向跳跃,加快了搜索速度.

Kimelfeld 等人也做了许多重要的研究工作[13,15,43,46].文献[13]描述了一个关于结构化数据的关键词查询的形式化框架,该形式化框架明确定义了枚举简化子树问题的 3 个变种,其中,有一个变种用来处理有向简化子树,另外两个变种用来处理无向简化子树.Kimelfeld 等人发现了简化子树的枚举和 Steiner 树优化问题这二者之间存在的复杂关系,由于 Steiner 树已被研究了很多年,诞生了大量的研究结果,因此这些研究可以被用来开发枚举简化子树的高效算法.文献[46]首先描述了如何在多项式时间延迟内枚举出所有的“K-片段”(即简化子树).之后,文献[15]又给出了高效的枚举算法,它可以采用精确的或者近似的权重升序顺序进行枚举(本文第 2.2.1.5.3 节将会详细讨论枚举顺序问题).对于近似顺序的情况,即使查询 K 的大小(即关键词的数目)非常大,枚举算法也

具有很高的效率.而且,它可以把那些为 Steiner 树而设计的近似算法作为自己的插件直接使用.而文献[43]提出的枚举算法则可以采用任意顺序进行枚举,而且文中把枚举问题分解成几个小的子问题,从而可以提前对每个子问题进行测试,判断它的结果是否非空,最后对子问题的结果进行组合从而得到原问题的答案.与文献[15]中的方法相比,文献[43]中的算法具有两个优点:首先,后者只需要多项式空间,而前者所需要的数据结构会随着枚举工作的进行而不断膨胀,数据结构的大小和输出结果的大小成正比;其次,文献[43]给出的实践经验表明,后者要比前者更加简单和容易实现.

文献[18,45]提出了一种动态编程方法,它可以近似地给出 top-k 个元组连接树.

上述针对结构化数据的关键词查询研究都是在数据图中寻找代价最小的连接树,即 Steiner 树,并且采用了 Steiner 树问题的近似算法来寻找连接树.但是,由于 Steiner 树问题是一个 NP-hard 问题,在一个大型数据图中寻找所有的 Steiner 树是非常困难的.而且 Steiner 树很难适应复杂数据图的情形,比如复杂的生物数据库,因为算法只能发现简单的树结构,而无法确定一些更加复杂并且有意义的图结构,比如环路.因此,EASE[23]提出了“r-半径 Steiner 树”的概念,把关键词查询问题转化成为“r-半径 Steiner 树问题”.形式化地,给定一个数据图 G 和输入关键词集合 K,r-半径 Steiner 树问题就是寻找图 G 中的所有 r-半径 Steiner 图,这些图包含了全部或者部分查询关键词,并且根据与 K 的相关性进行排序.尤其需要指出的是,EASE 可以适用于非结构化数据(文本数据)、半结构化数据(XML 文档)、结构化数据(关系数据库)和图数据.但是,EASE 可能丢失高度相关的结果,而且可能生成重复的结果[22].

文献[22]指出,简化子树还不能充分表达不同元组之间的关系,因此提出了“社区(community)”的概念(借鉴了 Web 社区的概念),它是一个数据图中的多中心有向图.文献[23]指出,“社区”比简化子树更能代表用户的查询兴趣.实际上,文献[23]中的“r-半径 Steiner 树”可以看成是多中心图的一个特例,即单中心图.文中提出了一种社区枚举算法,它可以在查询和数据复杂性条件下,在多项式时间延迟内枚举所有社区,并且算法所寻找到的社区是完整和无重复的.所谓完整性,就是指算法可以找到所有的社区.文中还提出了一种高效的索引方法来减少搜索空间,加快算法效率.

2.2.1.5 讨 论

对于基于数据图的方法而言,为了实现高效的搜索,简化子树枚举算法必须具备 3 个基本的属性:第一,不能丢失相关的结果;第二,必须是高效的;第三,生成的答案的顺序必须与用户期望的顺序高度接近.那么,现有的方法是否都满足这些属性呢?另外,基于数据图的方法大多假设数据图可以一次性放入内存,但是,如果在一些复杂的大型数据图的情况下,这个假设并不成立,即数据图存储空间大于内存空间,这时该如何处理这个问题呢?下面我们分别讨论这几个问题.

2.2.1.5.1 丢失相关结果

文献[7,43]等研究指出,BANKS,BANKS-II,NUITS 和文献[18]等研究工作中没有一种算法能够找到所有的简化子树.这些算法有可能丢失高度相关的结果,反而产生一个不是很相关的结果.比如,文献[18]中的方法首先生成 top-1 个简化子树,然后对此进行扩展,继而生成其他答案.但是,它不能保证接下来生成的 k–1 个结果就一定属于 top-k 中的结果,许多答案根本就不会被生成,即使算法已经运行到不会产生任何结果的时候.文献[18]中的算法也不能产生一些高度相关的结果.与文献[18]中的算法相比,文献[1,12]可以找到更多高度相关的结果,但仍然也不能找到所有结果.

枚举算法会丢失高度相关的元组的原因,可以参见第 2.4 节中相关内容的论述.其中,我们给出了一个关于 BANKS 系统的实例.通过这个实例,我们既讨论了结果重复问题,也解释了结果丢失问题.目前,有少量研究已经可以给出查询的所有相关结果,比如文献[7]提出的搜索引擎就可以返回所有 top-k 结果.

2.2.1.5.2 算法高效性

算法的高效性是由输出两个连续答案之间的时间延迟来衡量的.我们说一个算法 E 是可以在多项式时间内进行枚举的,如果存在一个多项式 $p(n)$,其中,n 是输入(即数据图 G 和关键词查询 K)的尺寸,从而使得生成下一个答案(以前已经生成一个答案)的时间延迟总是位于多项式边界 $p(n)$内.特殊情况下,如果没有任何答案,那

么算法在经历了多项式步骤以后必须停止[7].

对于枚举算法的设计,需要重点考虑两个问题[13]:第 1 个是效率问题,即是否无论以任意顺序或者确定顺序都可以高效地生成简化子树?如果采用确定的顺序,那么当关键词集合具有无限大尺寸时,NP-hard 问题决定了这种简化子树的枚举工作不会很有效率.但是,对于固定尺寸的关键词集合K而言,仍然有可能找到高效的算法;第 2 个问题就是近似性,需要研究以一种近似的顺序来枚举简化子树是否可行和高效.关于近似顺序问题,我们在下面一节中会加以讨论.

考察算法高效性,通常需要看它们是否能够高效地完成以下 3 项任务:(1) 寻找最优的结果;(2) 计算 top-k 个结果;(3) 以排序的方式枚举出所有结果.

2.2.1.5.3 枚举顺序

对于简化子树枚举算法而言,生成答案的顺序必须和用户期望的顺序高度接近.如果排序函数选择得当,则排序后的结果顺序就是用户期望的顺序.在理想情况下,枚举算法生成答案的顺序应该与排序的顺序一致.

所谓的排序顺序(ranked order),就是选择一个理想的评分函数,使得它可以真实地反映查询结果与用户的相关性,然后根据评分函数对所有可能的答案进行相关性评分,最后按照相关性评分大小对所有答案进行降序排序.这样得到的答案之间的顺序就是排序顺序.理想情况下,系统应该采用这个顺序来枚举答案,即首先把所有可能的答案中相关性最高的答案枚举出来,然后重复执行同样的工作.但是,这个任务是无法高效完成的,主要原因在于简化子树枚举问题是 Steiner 树问题,而 Steiner 树问题又是一个 NP-hard 问题,即使是寻找很少的答案也是非常耗费时间的[7].为了保证效率,一些研究对枚举工作采用了启发的顺序,亦即采用一个与排序顺序比较相似的顺序,但是又无法保证这个顺序和排序顺序完全一致.比如,BANKS 系统生成的结果的实际顺序就不是由权重决定的排序顺序,不过系统没有说明这种实际顺序和排序顺序到底差距有多远.

基于数据图的方法大都按照相关性的近似顺序生成答案,并且必须在一个中间的堆栈中临时保存这些答案.以 BANKS[1]系统为例,它为元组连接树维护一个固定大小的堆,堆中的元素根据相关性进行降序排序.当生成一棵新的树时,算法不会直接输出这棵树,而是先把它加入到堆中.当堆已满而同时又要增加一棵新树时,算法就把堆中最高相关性的树输出来,然后把新树增加到堆中.当系统不再生成新的树时,算法就根据相关性降序排序,把堆中的所有树都输出来.虽然这个启发算法不能保证以排序顺序生成树,但是它可以很好地工作,即使堆的尺寸很小时,也可以获得不错的性能.

需要强调的是,这些算法在任何阶段都可以提供一个“边评分(edge score)”的边界 L_E,它保证不会在未来时刻再生成任何具有更高边评分的结果,并且所有具有更高边评分的结果都已经被生成.通过计算边评分边界以及给定关键词集合最大的可能节点评分,就可以得到一个关于结果总评分的边界 L.对于一棵结果树而言,如果它的评分大于边界L,那么它就可以被输出.随着算法的进行,边界L会逐渐减小,这就使得 top-k 结果以评分降序的顺序输出[20].

此外,对于近似枚举问题,一些研究工作[7,15]还引入了“θ-近似”的概念,并研究了采用θ-近似顺序的枚举算法.因此,这里我们需要给θ-近似一个形式化定义.

一个近似的质量,即近似顺序和排序顺序的接近程度,是由一个称为“近似比”的参数θ来衡量的.其中,$\theta>1$,并且θ可能是一个输入 x(关键词集合)的函数.对于一个给定的输入 x,一个最优答案的“θ-近似”是由答案 $app\in A(x)$($A(x)$表示关于输入 x 的查询结果)构成的,并且对于所有的答案 $a\in A(x)$,都有$\theta.rank(app)\geq rank(a)$.对于 top-$k$ 查询而言,它的θ-近似是一个集合 $AppTop$,它包含了 $\min(k,|A(x)|)$个答案,并且对于所有的 $a\in AppTop$ 和 $a'\in A(x)\backslash AppTop$,都有 $\theta.rank(a)\geq rank(a')$.对于所有答案的序列 $a_1,\ldots,a_n$,如果对于所有的 $1\leq i\leq j\leq n$,都有 $\theta.rank(a_i)\geq rank(a_j)$,那么这个序列就采用了$\theta$-近似顺序.

关于θ-近似有一个重要发现[7],即对于一个枚举算法而言,如果它能在多项式时间延迟内以θ-近似的顺序枚举答案,那么这个算法也能够高效地寻找到 top-k 答案的θ-近似.方法很简单,只需要算法在枚举了前 k 个结果以后就停止运行即可,这里,算法的运行时间和k呈线性关系,与输入的大小具有多项式关系.尤其需要指出的是,算法可以高效地寻找到最优答案的θ-近似.此外,文献[15]的研究也发现,为了寻找最优答案的“θ-近似”而设计

的高效算法,也完全能够用来在多项式时间延迟内解决如下两个问题:第一,以“(θ+1)-近似”的顺序枚举答案;第二,计算 top-k 结果的“(θ+1)-近似”.

文献[7]提出了一个新的搜索引擎,它采用了增量式算法,以高度升序(increasing height)的 2-近似(θ=2)顺序来枚举子树.算法具有多项式时间延迟,并且可以返回所有 top-k 结果.大量的实验结果显示,搜索引擎在生成答案的顺序和排序顺序之间具有很好的相关性.

总的来说,就现有的简化子树枚举算法而言,虽然它们输出的结果是按照相关性排序的,但是没有一种算法是以排序顺序生成结果,因为这些算法有可能丢失高度相关的结果(文献[7,43]等研究工作已经明确指出了这一点).

2.2.1.5.4 数据图无法放入内存

基于数据图的方法大多假设数据图可以一次性放入内存.对于大多数数据库而言,这种物化的数据图的内存空间消耗确实很小,可以直接放入内存[18].因为图中不需要存储实际数据,对于数据量达到上千 MB 的数据库,对应的数据图包含几百万个节点,可以被存储到几十 MB 的内存当中[1,12].但是对于海量数据库而言,庞大的数据图可能具有几十亿个节点,就无法放入有限的内存了,这时就会带来大量的 I/O 开销.文献[20]研究了这种情形的处理方法,提出了一种新的图表示技术,它融合了超节点图(图的浓缩版本,总是可以放入内存)和缓存中的细节图,从而形成多粒度的图表示方法.文献[20]还对现有的基于数据图的搜索算法进行扩展,提出了可以利用多粒度图的“反复扩展(iterative expansion)”和“增量扩展(incremental expansion)”搜索算法.这两种算法可以通过把搜索过程引导到图中可能产生结果的区域来最小化 I/O 开销.

2.2.2 基于模式图的方法

本节首先简要介绍基于模式图的方法,然后讨论该方法的数据和查询模型、查询过程以及相关研究.

2.2.2.1 概 述

DBXplorer[9],DISCOVER[10]以及文献[11,16,19,28]等都采用了基于模式图的方法,它们很好地利用了数据库模式,从而使回答关键词查询时更加快速,因为数据库模式中包含的结构化约束对查询处理过程是很有用的.

基于模式图的方法主要包括 3 个步骤:第 1 步,利用数据库模式来枚举可能包含查询结果的所有连接表达式;第 2 步,根据一系列规则把连接表达式转换成 SQL 语句,并在数据库上执行,得到所有可能的候选结果;第 3 步,对结果进行排序并把相关内容返回给用户.在第 1 步枚举连接表达式时,会对表达式尺寸(表达式包含连接的数目)进行限制,因为如果表达式尺寸太大(包含很多连接),两个元组距离太远(通过很多中间连接而发生关系),那么即使二者存在连接关系,实际意义也不大.在第 2 步中,执行 SQL 的方式可以分为两种[6]:一种是使用 SQL 语句直接在 RDBMS(relational database management system)上执行;另一种是采用中间件方法,在中间件上执行 SQL 语句,而中间件层位于 RDBMS 层之上.由于需要处理大量的关系代数表达式,许多现有的研究[11,19,28]都采用基于中间件的方法,而没有充分利用 RDBMS 的能力,只有早期的少量研究[9,10]采用直接在 RDBMS 上执行 SQL 语句.文献[6]证明,当前的商业数据库系统足够强大,完全可以高效地支持关键词查询,而不需要增加和维护额外的索引.因此,文献[6]采用了直接在 RDBMS 上执行 SQL 语句的方法,并且用实验证明了其可行性.

2.2.2.2 数据和查询模型

下面给出数据库模式图、连接树、元组连接树和元组连接网络等概念,这些概念参照了文献[9,10]中的相关论述.

定义 7(数据库模式图 G_S). 假定一个数据库有 n 个关系 $R_1,\ldots,R_n$,每个关系 R_i 有 m_i 个属性,$a_1^i,\ldots,a_{m_i}^i$.数据库模式图 G_S 是一个有向图,它反映了数据库模式中的主外键关联.数据库中的每个关系 R_i 在图 G_S 中都存在与之对应的节点,如果关系 R_i 的属性集合{$a_{b_1}^i,\ldots,a_{b_l}^i$}和 R_j 的属性集合{$a_{b_1}^j,\ldots,a_{b_l}^j$}之间存在主外键关联,即对于 k=1,…,l 都有 $a_{b_k}^i \equiv a_{b_k}^j$,那么,在图 G_S 中就存在与之对应的边 $R_i \to R_j$.

数据库模式图通常要比数据图小得多,并且不需要维护任何额外的关系.图 3 就是本文在前面给出的数据库(如图 1 所示)所对应的数据库模式图.文献[9–11,16]等研究都利用数据库模式来枚举可能包含查询结果的所有连接表达式,然后在数据库上执行这些表达式得到元组连接树[9]或元组连接网络[10](二者都是简化子树的

别名).

下面我们来了解一下连接表达式的概念.对于这个概念,不同的研究使用不同的名称,但本质上却是一致的.比如,连接表达式的概念在 DISCOVER 中被称为候选网络,在 DBXplorer 中被称为连接树.

定义 8(连接树). 连接树 J 是一棵以数据库表为节点的树,并且对于 J 中每对相邻的节点 R_i^K 和 R_j^M ,在图 G_u 中都存在与之对应的边(R_i,R_j).其中,G_u 是由数据库模式图 G_S 的无向图版本.

这里,我们给出一个实例来介绍连接树.如图 4(a)所示,有一个数据库模式图 G_S,该图描述了 7 个表 $T_1,T_2,T_3,T_4,T_5,T_6,T_7$ 之间的关系.假定查询关键词集合是$\{k_1,k_2,k_3\}$,图中黑色的方框表示了那些可以与关键词匹配的节点(表),其余节点则用白色方框表示.假设 k_1,k_2,k_3 都出现在 T_1 当中,并且出现在 T_1 的不同列中,k_2 都出现在 T_4 当中,k_3 都出现在 T_7 当中.图 4(b)给出了 3 个可能的连接树,这 3 棵树包含的节点集合分别是$\{T_1\}$,$\{T_1,T_4\}$和$\{T_1,T_4,T_5,T_7\}$.此外,也可以看出,由$\{T_4,T_5,T_7\}$构成的树不可能是连接树,因为这些表不可能包含所有关键词 k_1,k_2 和 k_3.

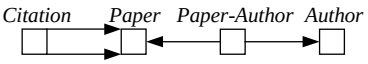
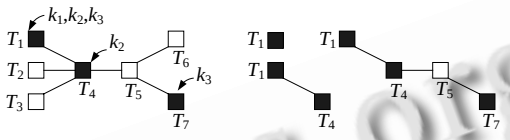


Fig.3 An example of database schema

图 3 一个数据库模式实例



(a) Database schema (b) Join trees

(a) 数据库模式 (b) 连接树

Fig.4 Join trees

图 4 连接树

在关系数据库上执行连接表达式,即连接树或连接网络,就可以得到元组连接树或元组连接网络.

定义 9(元组连接树). 给定一个数据库模式图 G_S,一个元组连接树 T 是一棵元组树.其中,T 中的每一条边(t_i,t_j)($t_i\in R_i,t_j\in R_j$)满足以下两个属性:

(1) $(R_i,R_j)\in G_S$;

(2) $t_i\infty t_j\in R_i\infty R_j$.

这里,我们给出一个关于元组连接树(或元组连接网络)的简单例子.假设有 4 个查询关键词 k_1=XML,k_2=SQL,k_3=Tom,k_4=mining,我们使用这 4 个关键词在图 1 所示的数据库中进行搜索,可以得到图 5 所示的一棵可能的元组连接树,它包含了全部的 4 个关键词,从图中我们可以看出树中每个节点所包含关键词的具体情况.

基于以上定义,下面我们给出采用“基于模式图的方法”时的关键词查询模型.

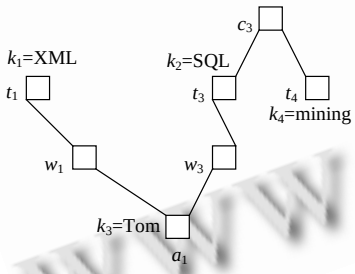


Fig.5 An example of join tree of tuples

图 5 元组连接树实例

定义 10(关键词查询). 一个关键词查询是一个关键词集合$K=\{k_1,k_2,\ldots,k_m\}$,关键词查询的结果是所有可能的元组连接树(连接网络)的集合,并且这些元组连接树满足以下条件:

- 完整性:每个关键词都至少出现在元组连接树中的一个元组上;
- 最小性:无法做到从元组连接树中移除任何元组,同时还能使剩下的元组继续组成一个具备完整性的元组连接树.

通过以上定义可以看出,元组连接树(连接网络)本质上就是数据图 G 关于关键词 $k_1,\ldots,k_m$ 的简化子树.

DISCOVER 和 DBXplorer 等研究都采用了类似元组连接树的概念(虽然对这个概念的命名各不相同).实际上,这种树模型也被用来寻找存在关联的网页,这些网页作为一个整体包含了所有查询关键词.

2.2.2.3 查询过程

总的来说,在关系数据库中,当采用基于模式图的方法时,主要需要 3 个步骤来处理关键词查询:(1) 生成所有的候选答案,每个候选答案都是一个元组连接树,它们是由来自多个表的多个元组连接得到的;(2) 为每个答案计算评分,评分函数应该使得最相关的答案尽可能地排在最前面;(3) 最后返回具有语义的结果.

下面我们介绍一个具有代表性的关键词查询系统(即 DISCOVER)的搜索过程.

首先,用户向系统提交一个关键词集合 $K=\{k_1,k_2,\ldots,k_m\}$.系统根据关键词,并利用关系数据库主索引,对每个关系 R_i 都查找出包含这些关键词的元组集合 $R_i^{k_1},\ldots,R_i^{k_m}.R_i^{k_j}$ 的每个元组的属性值中都包含了关键词 k_j;其次,系统计算所有的候选网络(与上面介绍的连接树的概念类似).亦即根据主外键关联而创建的连接关系表达式,候选网络集合可以保证生成所有的最小连接网络;然后,系统对这些候选网络进行评估,并使用计划生成器生成一个执行计划,这个计划在评估候选网络过程中可以计算并使用中间结果;最后,DISCOVER 为每一行执行计划都生成 SQL 语句,并把这些 SQL 语句提交给 DBMS 执行,DBMS 返回元组连接网络,并进行排序,这就是关键词查询的结果.

在枚举连接树过程中,DBXplorer,DISCOVER 和 DISCOVER-II[11]等方法都使用了宽度优先的图搜索策略,而文献[44]提出的算法 FIS(find initial subgraphs)则采用了最好优先的图搜索策略.

2.2.2.4 相关研究

在基于模式图的方法中具有代表性的研究成果包括 DBXplorer[9],DISCOVER[10]以及文献[6,11,16,19,24,52]等,它们主要关注枚举简化子树的高效性,采用各种方法避免生成无用的结果.同时,采用高效的算法减小时间和空间复杂性.

对于给定的查询关键词集合,DBXplorer[9]返回数据库中所有包含了全部查询关键词的行,这些行或者来自单个表,或者来自多个表的连接操作.DBXplorer 通过两个步骤完成查询工作:第 1 步被称为“发布”的预处理,它会为数据库构建符号表和辅助结构,从而使其支持关键词查询;第 2 步被称为“搜索”,它从发布的数据库中获得匹配的行.但是,DBXplorer 没有考虑两个元组来自同一个关系表的情况,并且只考虑精确匹配,也就是说,关键词必须和一个属性值完全匹配.此外,它也没有考虑连接树的可重用性[10].

DISCOVER[10]面向关系数据库,并且同样不需要用户了解数据库模式和 SQL 语言知识,只要使用关键词就可以进行查询,极大地便利了用户的信息检索工作.通过利用数据库模式,DISCOVER 可以无重复性地发现所有候选网络,但候选网络的尺寸是受数据限制的.同时,文献[10]也证明了最优执行计划的选择是一个 NP-hard 问题,并提供了一种贪婪算法,它可以提供近似最优的查询计划执行开销.而且,DISCOVER 直接对数据库进行操作,因此它不存在主存空间限制.

DBXplorer 和 DISCOVER 都采用了非常简单的结果排序机制(关于结果排序的问题,本文第 2.3 节会有更加详细的论述),即根据元组连接树中所包含的元组数目的升序顺序来对它们进行排序.当两个元组连接树包含相同数目的元组时,就采用任意方式决定二者的顺序.因此,所有具有单个元组的元组连接树都会排在具有多个元组的元组连接树之前.此外,二者都没有设计有效的 top-k 查询优化机制,因而影响了查询的效率.即它们都尽力去捕捉包含全部关键词的所有元组,而不是在前 k 个结果生成以后就停止算法的执行.为此,DISCOVER-II[11]首先引入了 IR 排序公式来解决搜索效率问题,同时也提出了几种有效的专门针对 top-k 结果进行优化的查询执行算法.文献[11]采用了两种算法,即稀疏(sparse)算法和全局管道(global pipeline)算法,它们可以及时停止元组连接树的枚举工作,只返回 top-k 个结果.DISCOVER-II 建立在一个中间件基础之上,这个中间件位于 RDBMS 上面, DISCOVER-II 会通过中间件发起 SQL 查询来访问数据.

DBXplorer 和 DISCOVER 关注的是算法的效率,而忽视了算法的有效性.算法得到的有些结果虽然也包含了所有关键词,却未必是用户感兴趣的,那么这些结果就是无效的.因此,文献[16]重点研究了算法的有效性,它也是第 1 个采用大量实验来验证算法有效性的研究.同时,文献[16]也提出了一个新的排序策略,通过采用精炼的加权模式对排序公式进行了优化,它可以对结果进行有效排序,并返回具有基本语义的结果.

文献[19]主要讨论在关系数据库中如何支持高效的 top-k 关键词查询.分析了以前排序方法的不足,并提出

了一种新的排序方法.它根据虚拟文档模型对现有的排序方法和理论进行了改进,并在排序中考虑了诸如结果的完整性和尺寸等因素.由于大多数关于 top-k 查询的优化工作都假设排序聚集函数具有单调性,为此,文献[19]研究了一种新的有效的查询处理方法 SPARK,它可以对非单调排序函数进行优化.文献[19]提出了 skyline 扫描算法,通过对非单调排序函数使用一个单调的积分上限函数,它可以取得最小的数据探查量.另外,还提出采用另一个非单调上限函数来进一步限制不必要的数据访问,并据此设计了相应的算法.文献[19]通过大量实验证明,SPARK 可以比 DISCOVER-II[11]取得更好的性能.

文献[6]使用 SQL 为一个给定的关键词集合寻找到 3 种类型的元组连接结构,并且可以对它们的尺寸进行控制,同时不需要增加和维护额外的索引,也不需要中间件的支持.这 3 种元组连接结构分别是:

(1) 所有特定尺寸的元组连接树:如前所述,绝大多数简化子树枚举方法都是采用元组连接树作为查询结果的表示方式;

(2) 所有从根元组出发在半径范围内可以到达的元组集合:实际上就是本文第 2.2.1.4 节中提到过的 r-半径 Steiner 树;

(3) 所有在指定半径范围内的多中心子图:就是本文第 2.2.1.4 节中提到过的由文献[22]提出的“社区”.

上述基于模式图的方法大都采用即席的方式确定元组之间的关系,由于元组之间存在大量关系,这种方式通常效率较低.为此,文献[52]提出了元组单元(tuple unit)的概念,并为关键词查询增加了数据预处理阶段.在这个离线处理阶段,需要搜索元组单元并对这些单元进行物化,在关键词查询时,就可以直接利用这些元组单元加速关键词查询的在线处理.

此外,文献[24]还提出了一个和关键词查询相关的问题,即“频繁共现词语(frequent co-occurring term,简称 FCT)”查询.给定一个关键词集合 K 和一个整数 k,一个 FCT 查询可以返回 k 个不在 K 中的词语,但是这些词语却在 K 的查询结果中频繁出现.FCT 查询可以发现与 K 密切相关的概念,并且可以作为一种有效的工具对传统的关键词查询的关键词集合 K 进行二次优化.

2.2.3 两种方法的比较

基于数据图的方法必须对整个数据图进行遍历,而数据图的大小通常比模式图大好几个数量级.另外,为了找到了包含关键词的元组之间的连接关系,算法要执行大量的元组连接操作.因此,基于数据图的方法通常会面临可扩展性的问题.当数据图过大时,无法一次性放入内存,则需要额外的方法来处理图分区[20]问题,这也增加了算法开销.基于数据图的方法还存在一个明显的不足,那就是必须对数据图进行物化和维护,而且一旦数据图被构建,不仅底层的数据库更新无法及时反映到数据图中,而且所有由数据库模式提供的结构信息都会被忽略,从而在算法运行阶段无法使用这些信息,而数据库模式中包含的结构化约束对查询处理过程是很有用的.此外,正如本文第 2.2.1.5.1 节讨论的那样,一些基于数据图的方法[1,12,18]还可能丢失高度相关的结果.

基于模式图的方法可以很好地利用数据库模式,加快了算法执行速度.但是,这种方法也有其自身的不足.对于基于模式图的方法,绝大多数研究,比如 DBXplorer,DISCOVER 和 DISCOVER-II,都会生成所有可能包含查询关键词的连接表达式.如果数据库模式较小且结构简单,那么生成所有连接表达式的过程就会非常迅速.但是通常来说,对于实际应用中的数据库来说,许多连接表达式可能最终只生成空结果[7,43],这就降低了整个查询过程的效率.文献[43]的研究指出,在两个连续的查询结果之间消耗的时间,尤其是生成第 1 个查询结果所需要的时间,在最坏情况下,通常与数据库大小呈指数关系.因此,空结果会影响查询过程的效率.

对于选择基于模式图的方法还是基于数据图的方法这个问题,边的权重是否重要是一个很重要的考虑因素[24].当必须考虑边的权重时,就应该选择基于数据图的方法,因为基于模式图的方法只知道模式图级别的主外键关联,而不知道元组级别的主外键关联.而当边的权重无关紧要时,基于模式图的方法则具有更好的性能,因为它们可以利用关系数据库强大的 SQL 执行引擎,通过一个简单的连接操作就可以得到多个元组连接树.

2.2.4 其他方法

目前,绝大多数简化子树枚举方法都可以归纳为基于数据图的方法和基于模式图的方法,但是也有少数研究采用了不同的思路.ObjectRank[17,21]采用了不同于上述两种方法的查询模型,它把基于影响力(authority)的排

序方法应用到了数据库的关键词查询中.在该方法中,数据库被建模成一个标记有向图,图中的每个节点都有一个标记,标记表明了节点的类型.同时,每个节点还具有一个关键词集合,这个集合构成了节点的内容.每个节点权重表示了节点的影响力(authority),边权重则表示了边所连接的两个节点之间可以传播的影响力的大小.从概念上来看,影响力会产生于标记图中包含关键词的节点(对象),并可以传播到与它具有语义连接关系的对象上.每个节点的排序是根据与关键词相关的影响力来决定的,在对结果进行排序时,需要综合考虑 3 个方面的因素,即结果与查询的相关性、确定性和重要性.对于影响力如何在数据库中传播这个问题,Hristidis 等人提出了预计算和利用现有的数据库模式的方法.虽然文献[24]把 ObjectRank 归类到基于数据图的方法,但是它与其他基于数据图的方法存在很大区别.首先,ObjectRank 与 BANKS 等基于数据图的方法不同,它返回的结果中只包含一个元组,而 BANKS 等返回的结果是一棵元组连接树;其次,BANKS 等方法只需要使用数据图,而 Object 则需要使用“影响力传播模式图”.另外需要强调的是,ObjectRank 的搜索算法具有很高的执行代价[44].

2.3 评分和排序

对由关键词查询得到的结果(元组连接树)进行排序是把最终结果提交给用户之前所需要做的非常重要的工作,因为满足关键词匹配条件的元组连接树可能很多,但是用户通常只对最相关的结果感兴趣.

排序函数是以评分函数为基础的,评分函数为每个元组连接树赋予一个分数,然后,排序函数根据这个分数按照降序对元组连接树进行排列.对于一个给定的关键词查询 K,DBXplorer[9]和 DISCOVER[10]等基于模式图的方法都采用如下方式为元组连接树 T 评分:

$$Score(T,K)=\begin{cases}\dfrac{1}{Size(T)}, & \text{如果}T\text{中包含了查询}K\text{的所有关键词}\\ 0, & \text{其他情况}\end{cases},$$

其中,$Size(T)$是 T 中元组的数目,或者可以是 T 中连接的个数加 1,二者其实是一致的.比如,如果 T 包含了 1 个连接,那么 T 就包含 2 个元组;而如果 T 包含 2 个连接,那么 T 就包含 3 个元组.连接的个数反映了两个包含关键词的元组之间的距离.很显然,如果一个元组和另一个元组发生关联,那么,所需要的连接个数越多,这两个元组的距离就越远,二者之间的相关性就越小,二者所在的元组连接树 T 对于用户的重要性也就越低.因此,T 就应该被评分函数赋予更低的分数.文献[30]指出,元组之间的距离不是依赖于数据库模式,而是取决于特定的关系实例.在一些关系实例中,两个元组可能通过 3 个以上的中间元组发生关联.实际上,这种结果对于用户已经不重要了.对于 top-k 查询而言,这种结果更不可能进入用户的视线.因此,为了减少搜索空间,一些启发方法会设置一个距离门槛值,当元组连接次数超过这个值时,就不会做后续的连接操作.

文献[11]的研究指出,采用以上方法得到的评分方法虽然捕捉到了查询结果的大小和结构,但却没有很好地利用近些年来由 IR 研究群体发展起来的相关性排序策略.为此,文献[11]提出了一种新的评分策略.在这种策略中,一个查询的元组连接树的评分取决于以下两个方面:

(1) 每个文本属性 $a_i\in T$ 的单属性 IR 类型相关性评分 $Score(a_i,K)$,这个是由 RDBMS 中的 IR 引擎决定的;

(2) 一个函数 $Combine$,它可以综合众多的单属性评分,从而为 T 得到最后的评分.

令向量 $A=\langle a_1,\ldots,a_n\rangle$ 都具有文本属性值,定义 $Score(T,K)=Combine(Score(A,K),Size(T))$,其中,$Score(A,K)=\langle Score(a_1,K),\ldots,Score(a_n,K)\rangle$.一个简单的 $Combine$ 函数定义如下:

$$Combine(Score(A,K),Size(T))=\frac{\sum_{a_i\in A}Score(a_i,K)}{Size(T)}.$$

但是,文献[19]用实例证明了上述排序方法可能存在一个问题,即系统给出的排序结果可能会与用户的真实需求相背离.为此,他们提出了“完整性系数(completeness factor)”的概念,对 IR 排序评分加以修正,从而获得了更好的排序结果.

BANKS 等基于数据图的方法采用了不同的评分方法,对于这些方法,一个简化子树的相关性评分是从节点的相关性评分和边的相关性评分这二者综合计算得到的.BANKS 的评分方法如下[1]:

(1) 计算节点评分:使用公式 $N_{score}(v)=N_v/N_{\max}$ 计算单个节点的评分,其中,$N_{\max}$ 是图中的最大节点权重;然后,采用所有节点评分的平均值,作为所有节点的总体评分 N_{score};

(2) 计算边评分:使用公式 $E_{score}(e)=w(e)/w_{\min}$ 计算每条边的评分,其中,$w_{\min}$ 表示图中最小的边权重;然后,总体边评分就是 $E_{score}=1/(1+\sum_e E_{score}(e))$;

(3) 计算树的总体相关性评分:把总体边评分和节点评分进行综合就可以得到一个树的总体相关性评分,综合方式或者可以采用相加的方法,即 $(1-\lambda)E_{score}+\lambda N_{score}$;或者可以采用相乘的方法,即 $E_{score}\times(N_{score})^{\lambda}$,其中,$\lambda$是一个属于[0,1]的系数.

另外,Liu 等人[16]提出了一种新的排序策略,使用了 4 个标准化系数(元组树尺寸、文档长度、文档频率和文档间权重)来提高搜索结果的有效性.并且查询越复杂,效果改善越明显.

2.4 结果冗余问题

一些关于 XML 关键词查询的研究[53]探讨了结果冗余问题,但是目前还不清楚这些方法如何应用到关系数据库的关键词查询中.对于 XML 关键词查询,冗余问题很好定义,如果一个结果中同时包含了一个元素及其子元素,那么这个结果就包含了冗余信息.但是对于基于关系数据库的关键词查询而言,结果冗余问题会更加复杂.

关系数据库具有更加丰富的结构,复杂的数据库所对应的数据图中会包含很多回路,从而容易产生信息重复和冗余问题.系统生成的搜索结果中的重复冗余信息会使用户感到困惑.比如,假设查询中包含了 3 个关键词“李明”、“数据库”和“计算机”.在每对关键词之间会存在许多关联,比如,李明是计算机系的主任,李明编写了一本关于数据库的教材,李明教数据库课程,数据库是计算机系的一门课程,李明是计算机系的一名教师等等.不仅如此,以上关联的任意组合仍然是查询相关的答案.所以,如果一个系统不能处理这个问题,无法对答案进行正确的排序,那么它会为用户提交所有可能的答案,把用户淹没在信息的海洋中.

文献[7]的研究工作提出了处理结果冗余问题的方案.系统采用“引擎”+“排序器(ranker)”的架构,引擎会生成候选答案集合,排序器负责对答案进行评分.排序器采用了一个新的技术来消除重复信息,它的核心思想是:排序函数在初始时是基于总权重的,在排序过程中它会对自身进行动态调整,从而解决冗余性.更确切地说,在生成了 top-i 答案以后,如果其他答案包含了与 top-i 答案相似的信息,排序函数会惩罚该答案,降低该答案的评分.文献[7]考虑了几种不同的惩罚策略,并比较了不同惩罚策略的性能差别.

BANKS 系统也可能生成重复的结果,它需要反向跟踪,消除这些重复结果.在 BANKS 中,包含重复信息的结果被称为重复树[1].对于两棵有向树(BANKS 的查询结果是有向树)而言,如果它们对应的无向图是同构的,那么这两棵树就是重复树.系统只保留具有最高相关性的结果,并丢弃其他结果.为了探测重复树的存在,系统会为所有生成的结果树维护一个列表,当新的树被生成并要被加入到堆中时,如果堆中已经存在一棵重复树,并且相关性评分比新树小(之所以新树比老树的相关性大,是因为枚举算法并不是按照排序顺序枚举答案,这一点已经在第 2.2.1.5.3 节有过论述),那么系统就删除堆中的重复树,把新树加入到堆中.由于系统维护一个输出结果的列表,因此系统可以探测到一棵重复树是否已经被输出过.如果已经被输出,即使新树比老树相关性大,系统也会丢弃这棵新树.这也是为什么 BANKS 系统可能会丢失高度相关的结果的原因.

2.5 结果呈现

基于关系数据库的关键词查询面临的一个很大的挑战就是,如何改进查询结果的呈现方式,从而帮助用户快速浏览查询结果[27,40,54,55].这个问题之所以比较重要,主要基于以下两个原因:

(1) 对于用户而言,查询结果应该具有语义.查询过程的效率不仅取决于搜索算法的效率,还取决于用户对查询结果的理解程度.一个查询结果如果不能让用户一目了然,那么就会增加用户搜索过程所耗费的时间.因此,查询结果具有语义可以加速用户与搜索引擎的交互,提高搜索效率.但是,目前大多数研究工作的查询结果或者是一个元组或者是元组连接树,这对于用户而言并不是很容易理解;

(2) 应该返回最能满足用户需求的少量结果.在搜索引擎中返回一大堆结果会使用户显得茫然无措,尤

其是当搜索结果中包含了很多相似的内容时,就更加浪费了用户的很多宝贵时间.

针对结果呈现问题,BANKS[1]系统提供了丰富的界面来支持用户对数据库的内容进行浏览.浏览系统会自动地为数据库关系和查询结果生成可浏览的视图,不需要编程和用户干预.BANKS 还提供了模板功能,允许用户把数据显示方式预先定义成模板,以便以后多次随时使用.

把查询结果组织成簇[56]也是一种有效的结果呈现方式,用户可以通过检查描述信息来判断一个簇是否相关.这样,用户就只需要进一步浏览相关的簇而忽略其他簇,从而可以提高用户的浏览效率.但是,传统的簇方法不能直接用于解决基于关系数据库的关键词查询的结果呈现问题.为此,文献[54]提出了一种新的簇方法,即TreeCluster,该方法包含两个主要步骤:第 1 步,使用标记来表示每个结果树的模式信息,从而把聚类问题转换成“判断标记树是否同构”的问题;第 2 步,根据用户的查询关键词在数据库中的频率对关键词进行排序,然后根据关键词节点对簇作进一步的分区.TreeCluster 技术在 NUITS[45]系统中得到了应用.类似地,文献[55]提出了一种基于数据库模式的结果展现方法 S-CBR(schema-based classification,browsing and retrieving),它结合了按结构聚类和按内容聚类这两种方法,将查询结果组织成两级类别,从而帮助用户快速浏览结果.

此外,CourseCloud 系统[57]采用了数据云(data cloud)作为针对结构化数据的查询结果的呈现方式.数据云借鉴了 IR 领域的标记云(tag cloud)的概念,后者是一种对文本内容进行可视化描述的有效方式,已在 Flickr 等众多网站中得到了应用.所谓标记,就是指底层数据库内容中最重要、最普遍的词语,它们通常以字母顺序进行排列,并且根据重要性,以不同的字体和颜色来显示.在 IR 领域,标记云可以用于针对非结构化数据的导航浏览和可视化,因为它可以突出显示底层内容中最重要的概念和隐藏的关系.CourseCloud 融合了关键词查询的灵活性和标记云的概括、可视化能力,可以帮助用户更好地理解数据库内容,并引导用户进行二次查询优化(query refinement).

2.6 索 引

为了减少寻找可连接元组的时间,提高搜索的效率,搜索算法在计算过程中通常会使用索引.索引有两种方式:一种是 DBMS 自身提供的索引,另一种是搜索算法构建的索引.

目前,绝大多数商业化 DBMS 都提供了针对文本属性的全文索引,比如 Microsoft SQL Server Full-Text Search[4],Oracle 9i Text[2],IBM DB2 Text Information Extender[3],这些索引通常作为数据库的外部文件进行保存.这种全文索引可以有效支持子串查询,用户可以采用函数 *Contain*(*A*,*k*)在某个单个属性(列)*A* 中查询所有包含关键词 *k* 的记录.DISCOVER[10]就使用了数据库自身提供的索引功能,对每个关系 R_i 都查找出包含查询关键词 k_i 的元组集合 $R_i^{k_1},...,R_i^{k_m}$.

文献[6]证明了这样一个观点,即当前商业数据库系统的强大功能足以高效地支持关键词查询时,则不需要增加和维护额外的索引.不过,为了加快搜索过程,不少研究[1,9,23,51,58−60]仍然通过扫描数据库来构建额外的索引.这种索引构建通常作为一种数据预处理工作事先完成,当数据库内容发生更改时,需要对索引进行维护.而且,这些索引方法都面临两个挑战:一个是如何控制索引内容的粒度,另一个是如何从索引内容中高效地寻找匹配的结果.下面介绍一些采用额外索引的研究工作.

DBXplorer[9]采用了称为“符号表”的索引结构,符号表会把关键词映射到表、列和行,通过查找符号表,就可以快速找到那些包含关键词的表、列和行.BANKS[1]使用了两种索引结构:一种是存储在磁盘上的关键词索引,可以把关键词映射到记录标识符 RID;另一种是存储在内存中的节点索引,可以把 RID 映射到图中的节点.DbSurfer[58]把每个元组的文本内容看成虚拟的 Web 文档,然后构建索引.文献[48]最早把结构感知(structural-aware)的索引集成到了数据库中,并且以关系表的形式对索引进行物化.Verify[59]会扫描数据库的内容,从而为关键词查询构建一个额外的索引,提高搜索效率.EKSO[61]提前扫描关系数据库,从而对虚拟文档构建文本索引.这些虚拟文档和数据库内容相对应,通过这种方式建立的文本索引可以很好地支持关键词查询.但是,EKSO 的索引空间开销会很大[20].

ITREKS[60]提出了一种新的方法来索引元组关系,它通过计算相互连接的关系数据库的全析取(full disjunction)来构建基本的元组关系 FDJT(full disjunction join tuples).文献[60]提出了 FDJT 元组索引表以对元组

之间的关系进行索引.通过制作元组关系索引来提前做连接计算工作,在搜索时就可以节省大量的计算工作,提高了搜索效率.

文献[62]指出,基于数据图和模式图的方法都需要在一些不可能生成最后结果的操作上耗费大量的处理时间.通过研究,作者发现生成检索结果的可能性和连接序列(join sequence)之间的连通性紧密相关.因此,文中提出了可到达性索引(reachability indexing).这种数据结构可以有效地消除无用的数据库遍历和连接树,加速查询过程.

上述方法大都假设数据图可以一次性放入内存,但是对于一些大型数据图而言,这个假设是不成立的.因此,文献[23,51]等研究了图分区策略,并对分区构建索引.

BLINKS[51]提出了双层索引来加快搜索速度,它是一种基于分区的索引,具体方法是:BLINKS 把一个图分割成许多子图(或者块),双层索引在块级别上存储了关于块的汇总信息,从而指导块间搜索.同时,还为每个块存储了更加详细的块内信息来加速块内搜索.

EASE[23]中的方法不是搜索 Steiner 树,而是搜索 *r*-半径图(或称为"*r*-半径 Steiner 树").为了加速寻找 *r*-半径图,它对图进行了分区,具体方法是:首先对 *r*-半径图进行聚类,从而得到多个簇;然后,基于簇对整个图进行分区,每个簇都对应图的一部分.为了加快搜索速度,EASE 为 *r*-半径图设计了一个有效的图索引 EI-Index(extended inverted index),而不是采用传统的倒排索引.因为传统的倒排索引对于基于文本和文档的搜索而言是很有效的,但是对于基于结构化和半结构化数据的关键词查询而言,传统的倒排索引无法发现最好的答案,尤其是无法发现最好的 *r*-半径图.在传统的倒排索引中,每个索引项是单个关键词;而在图索引 EI-Index 中,每个索引项是"关键词对"(即两个关键词的组合),该索引项对应的值是包含该"关键词对"的 *r*-半径图及其评分.实验结果表明,这种图索引在确定图结构化信息方面是非常高效的.

2.7 脏关键词问题

关键词查询降低了用户检索信息的技术难度,但是与结构化查询(如 SQL 查询)相比,却也降低了查询结果的准确度.查询所用关键词的选择直接影响到查询结果的质量,因此,选择合适、高效、准确的查询关键词对于提高查询效率和有效性尤为重要.但是对于普通用户而言,一方面,由于他们通常无法准确知道选择什么样的关键词进行查询是最合适的,因而给出的关键词也许和数据库内容根本无关;另一方面,用户给出的关键词可能存在拼写错误,或者关键词本身并不出现在数据库中,但是和数据库中的一些内容是语义等价的.上述情况就是所谓的"脏关键词(dirty keyword)问题"[63],这个问题与前面章节的内容具有相对独立性,因此我们放在这里讨论.

处理脏关键词问题是查询净化(query cleaning)问题的主要研究内容,它需要在处理关键词查询时增加一个额外的预处理步骤,即设计一个查询净化算法,从查询关键词集合 *K* 中过滤出与数据库内容无关的词汇,确定拼写错误词汇以及发现同义词.这样做不仅可以提高查询结果的质量,而且可以显著减少关键词查询算法的搜索空间.

文献[63]提出了查询净化问题的解决方案,定义了一个关键词查询的质量标准,并且提出了许多算法来净化关键词查询.文中证明了最优查询净化是一个 NP-hard 问题,但是如果数据库中术语的长度存在边界,那么这个问题就可以在多项式时间内得到解决.文献[63]还提出使用动态编程方法来解决基本的最优查询净化问题,对基本的算法作进一步扩展之后,可以用来解决增量查询净化问题以及 top-*k* 最优查询净化问题.

3 总结与研究展望

基于关系数据库的关键词查询是数据库研究领域的一个热点问题,目前已有不少相关的研究成果发表.在本文中,我们介绍了基于关系数据库的关键词查询问题,并讨论了该问题的研究背景以及困难与挑战.我们对该问题的解决方案进行了分类介绍,并讨论了每种方法的基本原理和优缺点.基于关系数据库的关键词查询问题是一个简化子树(Steiner 树)枚举问题,并且已经被证明是 NP-hard 问题.因此,本文介绍的方法大都关注算法的效率.

纵观现有的基于关系数据库的关键词查询的相关研究,我们认为以下几个方面是该问题未来的研究方向:

(1) 查询更加丰富的元组结构;(2) 良好的结果呈现机制;(3) 多种环境下的关键词查询;(4) 支持汇总查询;(5) 性能评价机制.

3.1 查询更加丰富的元组结构

元组结构代表了元组之间的语义连接关系.最初的关系数据库自带的全文搜索功能,查询结果只包含单个元组,无法反映元组之间的连接关系.一些元组虽然各自不包含全部查询关键词,但是它们连接后得到的元组连接树可以满足要求,因此元组连接树就成为表示关键词查询结果的简单而有效的方式,寻找元组连接树也就自然成为早期大多数研究的目标.但是后来的研究人员开始发现,元组连接树很难适应复杂数据图的情形,比如复杂的生物数据库,因为这些算法只能发现简单的树结构,而无法确定一些更加复杂并且有意义的图结构,比如环路.因此,一些研究开始寻找元组之间蕴藏的其他有意义的结构,比如 *r*-半径 Steiner 树[23]、社区[6,22]和频繁共现词汇[24]等等,这些元组结构都可以解决实际应用中存在的某类问题.目前,这类研究也只是在最近两年才出现,但是查询更加丰富的元组结构必将是一个未来的研究重点.这类研究的难点在于,如何从实际应用中遇到的问题里提炼出相应的元组结构,并设计寻找特定元组结构的高效算法.

3.2 良好的结果呈现机制

关系数据库中存储了大量的数据,即使对于简单的关键词查询也可能返回非常多的相关结果.用户的理解和接受能力毕竟有限,在一大堆简单罗列的结果面前通常会显得茫然无措,甚至会失去查询的兴趣.这时,后台所使用的关键词查询技术无论多么先进,对于用户而言都是毫无意义的.为此,需要设计一个良好的结果呈现机制来帮助用户浏览查询结果,同时尽可能地减少结果中的冗余数据.从目前发表的成果来看,这也是一个比较容易让研究人员忽略的问题,只有 NUITS[45]和 CourseCloud[57]等少数系统提出了较好的解决方案.其中:NUITS 把查询结果组织成簇,用户只需要浏览相关的簇而忽略其他簇;而 CourseCloud 则采用数据云的方式呈现结果.实际上,根据我们的调研,已经有不少相关的研究[56]关注关系数据库 SQL 查询结果的呈现问题,但是这些研究无法直接用于一些复杂元组结构的呈现上,比如 *r*-半径 Steiner 树和社区的结果呈现.因此,针对关系数据库的关键词查询,设计良好的结果呈现机制是不可缺少的一项研究工作,相信未来会出现更多的相关成果.

3.3 多种环境下的关键词查询

在基于关系数据库的关键词查询研究的带动下,一些研究开始努力把关键词查询技术应用到更加具有挑战性的环境中,比如数据流[28,29]、分布式数据库[30−32]、空间数据库[33,34]和 OLAP[35,36]等.以分布式数据库环境为例,在分布式环境下存在多个对等节点,它们互相合作提供服务.但是一个关键词查询的结果可能只包含在少量数据库节点中,为了避免由搜索大量无关数据库而带来的开销,文献[30]提出了一种新的方法——G-KS,它可以根据数据库节点包含给定查询答案的可能性大小来选择 top-*k* 个候选者.G-KS 用一个关键词关系图来表示一个数据库.其中,节点代表了关键词,边表示它们之间的关系.G-KS 可以利用关键词关系图来计算每个数据库和关键词查询之间的相似性,从而使得在查询过程中只对最有希望包含查询结果的数据库进行搜索.总的来说,多种环境下的关键词查询,可以让 IR 类型的查询技术为更多领域的用户带来全新的体验,这方面的研究具有很好的应用前景.

3.4 支持汇总查询

关键词查询极大地改善了用户从关系数据库中获取信息的方式,但是从目前已有的研究来看,绝大多数研究都只能支持简单类型的关键词查询,而无法支持比较复杂的汇总查询(aggregate query).用户为了获得汇总查询结果,就仍然需要学习 SQL 语言和数据库模式知识,从而构造正确的 SQL 汇总查询语句.实际上,数据库查询中有不少汇总查询,为了更好地满足用户需求,就要求关键词查询也能够很好地支持这类查询.目前,这方面已有少量具有影响力的研究出现.在 2008 年的 SIGMOD 会议上,Tata 等人[64]给出了这类问题的解决方案,他们提出了一个称为“SQAK(SQL aggregates using keywords)”的框架,允许用户通过关键词实现汇总查询,并且不需要或需要很少的数据库模式知识.SQAK 不需要对数据库引擎作任何的改变,并且适用于现有的任何数据库产品.

在2009年的EDBT会议上,Zhou等人[36]提出了使用最大连接方法和关键词图索引方法来处理汇总关键词查询问题.

3.5 性能评价机制

目前已有大量基于关系数据库的关键词查询研究,但却缺少统一的性能评价机制,许多研究大都采用不同的数据集和不同的性能指标,并从不同角度来衡量算法或系统的综合性能.缺少一个统一的性能评判机制,就使得一些性能结果对比测试很难具有普遍说服力.因此,Chen等人[41]在2009年SIGMOD大会的专题报告上指出了建立性能评价机制的必要性,从而有助于今后的系统设计.主要内容是建立一套测试基准(benchmark)和形式化评估体系.

References:

[1] Bhalotia G, Hulgeri A, Nakhe C, Chakrabarti S, Sudarshan S. Keyword searching and browsing in databases using BANKS. In: Proc. of the 18th Int'l Conf. on Data Engineering (ICDE 2002). San Jose: IEEE Computer Society Press, 2002. 431−440. [doi: 10.1109/ICDE.2002.994756]

[2] Dixon P. Basics of Oracle text retrieval. Bulletin of the Technical Committee on Data Engineering, 2001,24(4):11−14.

[3] Maier A, Simmen DE. DB2 optimization in support of full text search. Bulletin of the Technical Committee on Data Engineering, 2001,24(4):3−6.

[4] Hamilton JR, Nayak TK. Microsoft SQL server full-text search. Bulletin of the Technical Committee on Data Engineering, 2001, 24(4):7−10.

[5] Chaudhuri S, Ramakrishnan R, Weikum G. Integrating DB and IR technologies: What is the sound of one hand clapping? In: Proc. of the 2nd Biennial Conf. on Innovative Data Systems Research (CIDR 2005). Asilomar, 2005. 1−12. http://www.cidrdb.org/cidr2005/papers/P01.pdf

[6] Qin L, Yu JX, Chang LJ. Keyword search in databases: The power of RDBMS. In: Cetintemel U, Zdonik SB, Kossmann D, Tatbul N, eds. Proc. of the 2009 ACM SIGMOD Conf. on Management of Data (SIGMOD 2009). Providence: ACM, 2009. 681−694.

[7] Golenberg K, Kimelfeld B, Sagiv Y. Keyword proximity search in complex data graphs. In: Tsong J, Wang L, eds. Proc. of the 2008 ACM SIGMOD Conf. on Management of Data (SIGMOD 2008). Vancouver: ACM, 2008. 927−940. [doi: 10.1145/1376616.1376708]

[8] Goldman R, Shivakumar N, Venkatasubramanian S, Garcia-Molina H. Proximity search in databases. In: Gupta A, Shmueli O, Widom J, eds. Proc. of the 24th Int'l Conf. on Very Large Data Bases (VLDB'98). New York: Morgan Kaufmann Publishers, 1998. 26−37.

[9] Agrawal S, Chaudhuri S, Das G. DBXplorer: A system for keyword-based search over relational databases. In: Proc. of the 18th Int'l Conf. on Data Engineering (ICDE 2002). San Jose: IEEE Computer Society Press, 2002. 5−16. [doi: 10.1109/ICDE.2002.994693]

[10] Hristidis V, Papakonstantinou Y. DISCOVER: Keyword search in relational databases. In: Proc. of the 28th Int'l Conf. on Very Large Data Bases (VLDB 2002). Hong Kong: Morgan Kaufmann Publishers, 2002. 670−681. http://www.informatik.uni-trier.de/~ley/db/conf/vldb/vldb2002.html

[11] Hristidis V, Gravano L, Papakonstantinou Y. Efficient IR-style keyword search over relational databases. In: Freytag JC, Lockemann PC, Abiteboul S, Carey MJ, Selinger PG, Heuer A, eds. Proc. of the 29th Int'l Conf. on Very Large Data Bases (VLDB 2003). Berlin: Morgan Kaufmann Publishers, 2003. 850−861.

[12] Kacholia V, Pandit S, Chakrabarti S, Sudarshan S, Desai R, Karambelkar H. Bidirectional expansion for keyword search on graph databases. In: Böhm K, Jensen CS, Haas LM, eds. Proc. of the 31st Int'l Conf. on Very Large Data Bases (VLDB 2005). Trondheim: ACM Press, 2005. 505−516.

[13] Kimelfeld B, Sagiv Y. Efficient engines for keyword proximity search. In: Doan AH, Neven F, McCann R, eds. Proc. of the 8th Int'l Workshop on the Web & Databases (WebDB 2005). Baltimore: ACM Press, 2005. 67−72.

[14] Wen JJ, Wang S. SEEKER: Keyword-Based information retrieval over relational databases. Journal of Software, 2005,16(7): 1270−1281 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/16/1270.htm. [doi: 10.1360/jos161270]

[15] Kimelfeld B, Sagiv Y. Finding and approximating top-*k* answers in keyword proximity search. In: Vansummeren S, ed. Proc. of the 25th ACM SIGACT-SIGMOD-SIGART Symp. on Principles of Database Systems (PODS 2006). Chicago: ACM Press, 2006. 173−182. [doi: 10.1145/1142351.1142377]

[16] Liu F, Yu CT, Meng WY, Chowdhury A. Effective keyword search in relational databases. In: Chaudhuri S, Hristidis V, Polyzotis N, eds. Proc. of the 2006 ACM SIGMOD Int'l Conf. on Management of Data (SIGMOD 2006). Chicago: ACM, 2006. 563−574. [doi: 10.1145/1142473.1142536]

[17] Balmin A, Hristidis V, Papakonstantinou Y. ObjectRank: Authority-Based keyword search in databases. In: Nascimento MA, Özsu MT, Kossmann D, Miller RJ, Blakeley JB, Schiefer KB, eds. Proc. of the 30th Int'l Conf. on Very Large Data Bases (VLDB 2004). Toronto: Morgan Kaufmann Publishers, 2004. 564−575.

[18] Ding B, Yu JX, Wang S, Qin L, Zhang X, Lin XM. Finding top-*k* min-cost connected trees in databases. In: Proc. of the 23rd Int'l Conf. on Data Engineering (ICDE 2007). Istanbul: IEEE Computer Society Press, 2007. 836−845. http://www.informatik.uni-trier.de/~ley/db/conf/icde/icde2007.html

[19] Luo Y, Lin XM, Wang W, Zhou XF. Spark: Top-*k* keyword query in relational databases. In: Chan CY, Ooi BC, Zhou AY, eds. Proc. of the 2007 ACM SIGMOD Conf. on Management of Data (SIGMOD 2007). Beijing: ACM, 2007. 115−126.

[20] Dalvi BB, Kshirsagar M, Sudarshan S. Keyword search on external memory data graphs. PVLDB, 2008,1(1):1189−1204.

[21] Hristidis V, Hwang H, Papakonstantinou Y. Authority-Based keyword search in databases. ACM Trans. on Database Systems, 2008,33(1):1−40. [doi: 10.1145/1331904.1331905]

[22] Qin L, Yu JX, Chang LJ, Tao YF. Querying communities in relational databases. In: Proc. of the 25th Int'l Conf. on Data Engineering (ICDE 2009). Shanghai: IEEE Computer Society Press, 2009. 724−735. http://www.informatik.uni-trier.de/~ley/db/conf/icde/icde2009.html

[23] Li GL, Ooi BC, Feng JH, Wang JY, Zhou LZ. EASE: An effective 3-in-1 keyword search method for unstructured, semi-structured and structured data. In: Tsong J, Wang L, eds. Proc. of the 2008 ACM SIGMOD Conf. on Management of Data (SIGMOD 2008). Vancouver: ACM, 2008. 903−914.

[24] Tao YF, Yu JX. Finding frequent co-occurring terms in relational keyword search. In: Kersten ML, Novikov B, Teubner J, Polutin V, Manegold S, eds. Proc. of the 12th Int'l Conf. on Extending Database Technology (EDBT 2009). Saint Petersburg: ACM Press, 2009. 839−850. [doi: 10.1145/1516360.1516456]

[25] Florescu D, Kossmann D, Manolescu I. Integrating keyword search into XML query processing. Computer Networks, 2000, 33(1-6):119−135. [doi: 10.1016/S1389-1286(00)00069-4]

[26] Guo L, Shao F, Botev C, Shanmugasundaram J. XRANK: Ranked keyword search over XML documents. In: Halevy AY, Ives ZG, Doan AH, eds. Proc. of the 2003 ACM SIGMOD Conf. on Management of Data (SIGMOD 2003). San Diego: ACM, 2003. 16−27.

[27] Hristidis V, Papakonstantinou Y, Balmin A. Keyword proximity search on XML graphs. In: Proc. of the 19th Int'l Conf. on Data Engineering (ICDE 2003). Bangalore: IEEE Computer Society Press, 2003. 367−378. [doi: 10.1109/ICDE.2003.1260806]

[28] Markowetz A, Yang Y, Papadias D. Keyword search on relational data streams. In: Chan CY, Ooi BC, Zhou AY, eds. Proc. of the 2007 ACM SIGMOD Conf. on Management of Data (SIGMOD 2007). Beijing: ACM, 2007. 605−616. [doi: 10.3745/KIPSTD.2009.16D.6.859]

[29] Qin L, Yu JX, Chang LJ, Tao YF. Scalable keyword search on large data streams. In: Proc. of the 25th Int'l Conf. on Data Engineering (ICDE 2009). Shanghai: IEEE Computer Society Press, 2009. 1199−1202. [doi: 10.1007/s00778-010-0190-x]

[30] Vu QH, Ooi BC, Papadias D, Tung AK. A graph method for keyword-based selection of the top-*k* databases. In: Tsong J, Wang L, eds. Proc. of the 2008 ACM SIGMOD Conf. on Management of Data (SIGMOD 2008). Vancouver: ACM, 2008. 915−926. [doi: 10.1145/1376616.1376707]

[31] Yu B, Li GL, Sollins KR, Tung AK. Effective keyword-based selection of relational databases. In: Chan CY, Ooi BC, Zhou AY, eds. Proc. of the 2007 ACM SIGMOD Conf. on Management of Data (SIGMOD 2007). Beijing: ACM, 2007. 139−150. [doi: 10.1145/1247480.1247498]

[32] Sayyadian M, LeKhac H, Doan AH, Gravano L. Efficient keyword search across heterogeneous relational databases. In: Proc. of the 23rd Int'l Conf. on Data Engineering (ICDE 2007). Istanbul: IEEE Computer Society Press, 2007. 346−355. http://www.informatik.uni-trier.de/~ley/db/conf/icde/icde2007.html

[33] Felipe ID, Hristidis V, Rishe N. Keyword search on spatial databases. In: Proc. of the 24th Int'l Conf. on Data Engineering (ICDE 2008). Cancún: IEEE Computer Society Press, 2008. 656−665. http://www.informatik.uni-trier.de/~ley/db/conf/icde/icde2008.html

[34] Zhang DX, Chee YM, Mondal A, Tung AK, Kitsuregawa M. Keyword search in spatial databases: Towards searching by document. In: Proc. of the 25th Int'l Conf. on Data Engineering (ICDE 2009). Shanghai: IEEE Computer Society Press, 2009. 688−699. http://www.informatik.uni-trier.de/~ley/db/conf/icde/icde2009.html

[35] Wu P, Sismanis Y, Reinwald B. Towards keyword-driven analytical processing. In: Chan CY, Ooi BC, Zhou AY, eds. Proc. of the 2007 ACM SIGMOD Conf. on Management of Data (SIGMOD 2007). Beijing: ACM, 2007. 617−628. [doi: 10.1145/1247480.1247549]

[36] Zhou B, Pei J. Answering aggregate keyword queries on relational databases using minimal group-bys. In: Kersten ML, Novikov B, Teubner J, Polutin V, Manegold S, eds. Proc. of the 12th Int'l Conf. on Extending Database Technology (EDBT 2009). Saint Petersburg: ACM Press, 2009. 108−119. [doi: 10.1145/1516360.1516374]

[37] Shao QH, Sun P, Chen Y. WISE: A workflow information search engine. In: Proc. of the 25th Int'l Conf. on Data Engineering (ICDE 2009). Shanghai: IEEE Computer Society Press, 2009. 1491−1494. http://www.informatik.uni-trier.de/~ley/db/conf/icde/icde2009.html

[38] Song XM, Li GL, Feng JH, Zhou LZ. Effective fuzzy keyword search over uncertain data. In: Zhou XF, Yokota HR, Deng K, Liu Q, eds. Proc. of the 14th Int'l Conf. on Database Systems for Advanced Applications (DASFAA 2009). Brisbane: Springer-Verlag, 2009. 66−70. [doi: 10.1007/978-3-642-00887-0_6]

[39] Wang S, Zhang J, Peng ZH, Zhan J, Du XY. Ontology-Based semantic search over relational databases. Journal of Frontiers of Computer Science and Technology, 2007,1(1):59−78 (in Chinese with English abstract). [doi: 10.3778/j.issn.1673-9418.2007.01.005]

[40] Wang S, Zhang KL. Searching databases with keywords. Journal of Computer Science Technology, 2005,20(1):55−62.

[41] Chen Y, Wang W, Liu ZY, Lin XM. Keyword search on structured and semi-structured data. In: Cetintemel U, Zdonik SB, Kossmann D, Tatbul N, eds. Proc. of the 2009 ACM SIGMOD Conf. on Management of Data (SIGMOD 2009). Providence: ACM, 2009. 1005−1010.

[42] Hulgeri A, Bhalotia G, Nakhe C, Chakrabarti S, Sudarshan S. Keyword search in databases. Bulletin of the Technical Committee on Data Engineering, 2001, 24(3):22−32.

[43] Kimelfeld B, Sagiv Y. Efficiently enumerating results of keyword search over data graphs. Information System, 2008,33(4-5):335−359. [doi: 10.1016/j.is.2008.01.002]

[44] Simitsis A, Koutrika G, Ioannidis YE. Précis: From unstructured keywords as queries to structured databases as answers. The VLDB Journal, 2008,17(1):117−149. [doi: 10.1007/s00778-007-0075-9]

[45] Wang S, Peng ZH, Zhang J, Qin L, Wang S, Yu JX, Ding BL. NUITS: A novel user interface for efficient keyword search over databases. In: Dayal U, Whang KY, Lomet DB, Alonso G, Lohman GM, Kersten ML, Cha SK, Kim YK, eds. Proc. of the 30th Int'l Conf. on Very Large Data Bases (VLDB 2006). Seoul: Morgan Kaufmann Publishers, 2006. 1143−1146.

[46] Kimelfeld B, Sagiv Y. Efficiently enumerating results of keyword search. In: Bierman GM, Koch C, eds. Proc. of the 10th Int'l Symp. on Database Programming Languages (DBPL 2005). Trondheim: Springer-Verlag, 2005. 58−73. [doi: 10.1007/11601524_4]

[47] Brin S, Page L. The anatomy of a large-scale hypertextual Web search engine. Computer Networks, 1998,30(1-7):107−117. [doi: 10.1016/S0169-7552(98)00110-X]

[48] Li GL, Zhou XF, Feng JH, Wang JY. Progressive keyword search in relational databases. In: Proc. of the 25th Int'l Conf. on Data Engineering (ICDE 2009). Shanghai: IEEE Computer Society Press, 2009. 1183−1186. [doi: 10.1007/978-3-540-70504-8]

[49] Reich G, Widmayer P. Beyond Steiner's problem: A VLSI oriented generalization. In: Nagl M, ed. Proc. of the 15th Int'l Workshop on Graph-Theoretic Concepts in Computer Science (WG'89). Castle Rolduc: Springer-Verlag, 1989. 196–210.

[50] Robins G, Zelikovsky A. Improved Steiner tree approximation in graphs. In: Proc. of the 11th Annual ACM-SIAM Symp. on Discrete Algorithms (SODA 2000). San Francisco: ACM, 2000. 770−779. http://www.informatik.uni-trier.de/~ley/db/conf/soda/soda2000.html

[51] He H, Wang HX, Yang J, Yu PS. BLINKS: Ranked keyword searches on graphs. In: Chan CY, Ooi BC, Zhou AY, eds. Proc. of the 2007 ACM SIGMOD Conf. on Management of Data (SIGMOD 2007). Beijing: ACM, 2007. 305−316.

[52] Li GL, Feng JH, Zhou LZ. Retune: Retrieving and materializing tuple units for effective keyword search over relational databases. In: Li Q, Spaccapietra S, Yu ES, Olivé A, eds. Proc. of the 27th Int'l Conf. on Conceptual Modeling (ER 2008). Barcelona: Springer-Verlag, 2008. 469–483. [doi: 10.1007/978-3-540-87877-3]

[53] Kazai G, Lalmas M. Extended cumulated gain measures for the evaluation of content-oriented XML retrieval. ACM Trans. on Information System, 2006,24(4):503−542. [doi: 10.1145/1185877.1185883]

[54] Peng ZH, Zhang J, Wang S, Qin L. TreeCluster: Clustering results of keyword search over databases. In: Yu JX, Kitsuregawa M, Leong HV, eds. Proc. of the 7th Int'l Conf. on Advances in Web-Age Information Management (WAIM 2006). Hong Kong: Springer-Verlag, 2006. 385−396. [doi: 10.1007/11775300]

[55] Peng ZH, Zhang J, Wang S. S-CBR: Presenting results of keyword search over databases based on database schema. Journal of Software, 2008,19(2):323−337 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/19/323.htm. [doi: 10.3724/SP.J.1001.2008.00323]

[56] Chakrabarti K, Chaudhuri S, Hwang SW. Automatic categorization of query results. In: Weikum G, König AC, Deßloch S, eds. Proc. of the 2004 ACM SIGMOD Conf. on Management of Data (SIGMOD 2004). Paris: ACM, 2004. 755−766. [doi: 10.1145/1007568.1007653]

[57] Koutrika G, Zadeh ZM, Garcia-Molina H. Data clouds: Summarizing keyword search results over structured data. In: Kersten ML, Novikov B, Teubner J, Polutin V, Manegold S, eds. Proc. of the 12th Int'l Conf. on Extending Database Technology (EDBT 2009). Saint Petersburg: ACM Press, 2009. 391−402.

[58] Wheeldon R, Levene M, Keenoy K. DbSurfer: A search and navigation tool for relational databases. In: Williams MH, MacKinnon LM, eds. Proc. of the 21st British National Conf. on Databases (BNCOD 2004). Edinburgh: Springer-Verlag, 2004. 144−149.

[59] Raghavan P. Structured and unstructured search in enterprises. Bulletin of the Technical Committee on Data Engineering, 2001,24(4):15−18.

[60] Zhan J, Wang S. ITREKS: Keyword search over relational database by indexing tuple relationship. In: Ramamohanarao K, Krishna PR, Mohania MK, Nantajeewarawat E, eds. Proc. of the 12th Int'l Conf. on Database Systems for Advanced Applications (DASFAA 2007). Bangkok: Springer-Verlag, 2007. 67−78. [doi: 10.1007/978-3-540-71703-4]

[61] Su Q, Widom J. Indexing relational database content offline for efficient keyword-based search. In: Proc. of the 9th Int'l Database Engineering and Applications Symp. (IDEAS 2005). Montreal: IEEE Computer Society, 2005. 297−306. [doi: 10.1109/IDEAS.2005.36]

[62] Markowetz A, Yang Y, Papadias D. Reachability indexes for relational keyword search. In: Proc. of the 25th Int'l Conf. on Data Engineering (ICDE 2009). Shanghai: IEEE Computer Society Press, 2009. 1163−1166. http://www.informatik.uni-trier.de/~ley/db/conf/icde/icde2009.html

[63] Pu KQ, Yu XH. Keyword query cleaning. PVLDB, 2008,1(1):909−920.

[64] Tata S, Lohman GM. SQAK: Doing more with keywords. In: Tsong J, Wang L, eds. Proc. of the 2008 ACM SIGMOD Conf. on Management of Data (SIGMOD 2008). Vancouver: ACM, 2008. 889−902.

附中文参考文献:

[14] 文继军,王珊.SEEKER:基于关键词的关系数据库信息检索.软件学报,2005,16(7):1270−1281. http://www.jos.org.cn/1000-9825/16/1270.htm. [doi: 10.1360/jos161270]

[39] 王珊,张俊,彭朝晖,战疆,杜小勇.基于本体的关系数据库语义检索.计算机科学与探索,2007,1(1):59−78. [doi: 10.3778/j.issn.1673-9418.2007.01.005]

[55] 彭朝晖,张俊,王珊.S-CBR:基于数据库模式展现数据库关键词检索结果.软件学报,2008,19(2):323−337. http://www.jos.org.cn/1000-9825/19/323.htm. [doi: 10.3724/SP.J.1001.2008.00323]

林子雨(1978－),男,博士,助理教授,CCF 会员,主要研究领域为数据库,实时主动数据仓库,数据挖掘.

王腾蛟(1973－),男,博士,副教授,CCF 高级会员,主要研究领域为数据库,数据仓库,Web 数据集成,数据挖掘.

杨冬青(1945－),女,教授,博士生导师,CCF 高级会员,主要研究领域为数据库,数据仓库,Web 数据集成,移动数据挖掘.

张东站(1974－),男,博士,副教授,CCF 会员,主要研究领域为数据库理论与应用.