关系数据库的关键词检索①

Key Search over Relational Databases

戴经国 李运智 谢 东 (湖南人文科技学院 计算机科学技术系 湖南 娄底 417000)

摘 要： SQL 查询语言被用于检索关系数据库中的数据，但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。本文总结了基于关键字的数据库检索工作的关键技术和研究进展，展望了今后的研究方向。

关键词： 关系数据库 关键字查询 信息检索

1 引言

数据库系统和信息检索系统的数据查询技术，两者的解决方法是不同的。数据库系统通过复杂的查询语句查询结构化数据，结果是确定和完整的；信息检索系统通过关键字查询非结构化数据，结果通常是不精确和不完整的，某些结果要比其他结果更相关。两者在查询方面的区别可以用图 1 形象描述。

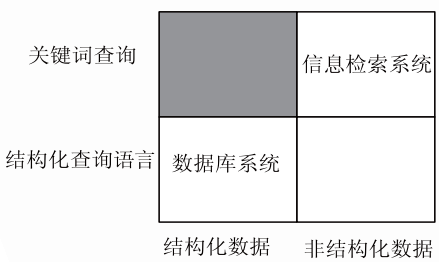

图 1 数据库系统 vs 信息检索系统

近几年来，结构化和半结构化数据的关键词查询逐步引起研究人员的兴趣。Oracle，DB2，SQL Server 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的关键词查询系统被开发出来：BANKS[1]、DBXplorer[2]、DISCOVER[3]、IR-Style[4]、EKSO[5]和 ObjectRank[6]、SEEKER[7]。这些系统的基本思路是一致的，即都将数据库看做是由数据库中的元组(顶点)通过主键-外键的关系(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[8]。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(EKSO，ObjectRank)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，因此响应时间较短。但是，由于离

① 基金项目:湖南省自然科学基金(07JJ3119,07JJ6113);湖南省教育厅科研基金(08B040)
收稿时间:2008-12-04

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。

XML 数据的关键词查询是对 XML 查询[9,10]语言的一个有益扩充。Goldman 等[11]研究了存储在关系数据库中的 XML 数据的关键词查询。Florescu 等[10]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。

不同的查询语句被用来查询不同种类的数据，如 SQL 查询关系数据库，XQuery 用来查询 XML 文件和文本文件。一个系统处理多种类型的数据就必须提供统一的查询语句，而关键字查询可以做到这一点。同时现在流行的关系数据库系统都为数据库中的文本建立了全文索引，这就为实现基于海量数据基础上方便的实现关键词查询提供了基础。

关系数据库上的关键词检索主要是在数据库中发掘包含关键词的元组及它们之间的关系。在现有的系统中，建模方法或使用有向数据图，如图 2，是关于 DBLP 的数据图描述；或使用无向模式图，如图 3，是关于 DBLP 数据库的模式图描述。这两张图形象地描述了模式图与数据图的区别，简单地说，数据图和模式图的关系就是“值”与“型”的关系。数据图中的结点对应关系表中的元组，边表示元组之间的主外码参照关系[14,15]。模式图的结点对应数据库中的关系，边表示模式定义的主外码约束。就图 2 和 3 而言，第一行结点描述对应图 3 中的 Paper 关系表，第二行和第三行分别对应关系表 Write 和 Author。查询结果表示方式或是一个结点或是元组连接树。基于建模方法，系统还可以分为在线和离线两类，在线系统通常将数据库表示为模式图，最终通过 SQL 查询得到数据库“最近的一致性状态”下的查询结果，但速度较慢。离线系统则使用数据图表示数据库，在内存执行基于数据图的快速查询扩展算法，并通过使用预处理生成数据图的方式提高查询速度。

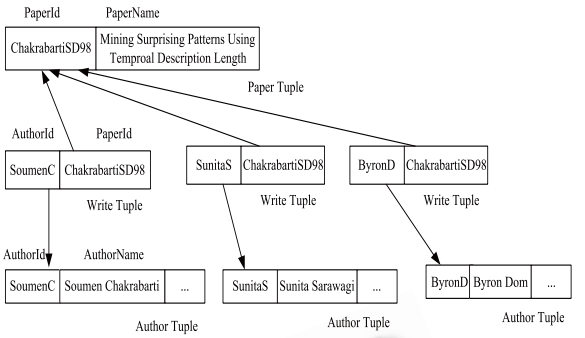

图 2　DBLP 有向数据图(主外键参照)

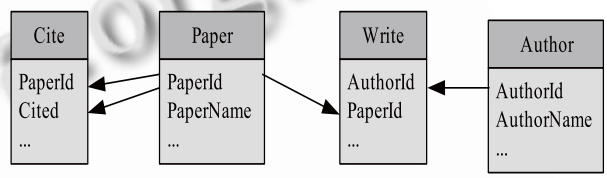

图 3　DBLP 数据库模式图

目前，BANKS、DISCOVER、DBXplorer 等主要是简单的实现关系数据库的精确查询，但是它们还存在有不足的地方，不支持对非文本属性的关键词查询等。如要从图 3 所示的 DBLP 里查询 Jim Gray 在 1990 年后发表的关于 Transaction 的论文，用 SQL 表示为：

```
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
```

针对上述 SQL 描述的查询，已有系统还不能实现这样复杂的参与数字属性关键词的查询。王珊在解决原有系统不足的基础上一起构建了 SEEKER 查询系统，有效改善了某些方面。为了兼顾查询快和结果准确的优点，开发了基于关系数据库通用的在线关键词查询系统 DETECTOR[16]。下面就针对各个系统所用到的关键技术加以阐述。

2　关键技术

2.1 系统模块

数据库检索系统有预处理模块和查询模块。

预处理模块是负责在执行用户第一个查询前进行

预处理，产生需要被查询模块加速查询处理的数据，产生的数据存储在内存或存储在硬盘。在 BANKS 中，数据表被存储于内存中；在 DBXplorer 中，标记表被存储在硬盘中。预处理模块的复杂性取决于数据库检索系统；在 DISCOVER 中，数据库系统的全文搜索能力被开发去做预处理；在 ObjectRank 中，大多数计算都由预处理完成的。

查询模块负责查询过程，分析用户的输入，找出查询关键字和查询语义，然后执行搜索算法产生结果。在这个阶段完成的工作也取决于数据库检索系统。在 ObjectRank 中，计算每个相应查询对应节点的分数来产生最后的结果；在 DISCOVER 中，产生元组集合图、列举候选网络和执行查询；SEEKER 使用数据库系统的全文索引对文本属性进行关键词查询，同时支持元组级和属性级两种粒度的全文检索。

2.2 数据模型与查询机制

大多数数据库检索系统把数据库模拟成数据库图，数据库图表可以是有向或无向的，由一个模式图和一个数据图组成。在模式图中，节点代表关系，而边代表主键－外键在关系间的主外键约束；在数据图中，节点代表元组，边代表元组的主外键约束。图表示数据库有明显的优势，Web 和 XML 文件可以用图表示。Web 图和数据库图的类似表明，搜索引擎的搜索算法可以被数据库检索系统利用。因为，XML 文件和数据库都被模拟为图，数据库检索系统(如 BANKS 和 DISCOVER)可以扩展检索 XML 文件。

有向图进行关键字查询，节点代表数据库中的元组，有向边代表每个主键-外键连接的两个元组。通常一个查询包含 $n(n\geqslant 1)$种查询条件 t_1，t_2，…，t_n。每个查询条件 t_i 对应于一个节点 S_i。当一个节点将查询条件作为它的属性值或元数据的一部分时，这个节点就对应于这个查询条件，相应的节点 t_1，t_2，…，t_n 对应于 S_1，S_2，…，S_n。BANKS 把查询结果定义为连接树，连接树是一棵至少包含一个节点的有向树，根节点叫做信息节点。BANKS 使用启发式搜索算法去搜索所有的信息节点，从每个叶子开始使用 Dijkstra 的单源点的最短路径算法[17]来移动数据，每一个单源点最短路径算法都会访问根节点。这种搜索算法可以大大缩短信息采集时间。

DBXplorer 采用无向图来进行关键字查询，把包含所有关键字的元组连接作为查询结果。1) 标记表是一个在查询前由数据库预处理创建的反向列表，寻找标记表，然后查找包含查询关键字的数据库的列和行；2) 依照模式图列举连接树。一个连接树是模式图的子树，对应于表的叶子节点至少包含一个查询关键字，且每个查询关键字属于一个叶子节点。如果所有表都在连接树中被连接，则所有的关键字都在结果包含的行中；3) 通过去掉不包括任何关键字的叶子节点而产生新的模式图 G'，在 G'中的叶子节点是通过一个启发性算法选择的，再从选择的叶子节点开始 G'的广度优先遍历导出了所有的连接树。SQL 语句连接树中的表且选择包含所有关键字的行，最后结果被排序并提交给用户。

DISCOVER 改进了 DBXplorer，它的列举算法也是基于广度优先搜索算法的，但它没有首先裁剪元组集合图。元组集合图是建立在模式图的基础上的，它包括与元组集合相关的关键字。数据库系统允许用户在单个属性上创建全文索引去执行关键字查询，DISCOVER 利用了数据库系统的全文搜索能力去查找包含查询关键字的关系和元组，而且也利用了 IR-style 中的排序功能。

ObjectRank 定义一个结果为数据模式图的结点，每个结点有一个初始状态，但最后整个系统会到达一个稳定的，一致的状态。尽管一个结点在初始状态是不重要的，但在最后的状态可能变得重要，ObjectRank 能找到相关的不包含任何查询关键字的结点。ObjectRank 通过基于被 Google 使用的 PageRank 算法[18]使得每个结点到达最终稳定一致的状态。PageRank 算法核心是互相连接的结点有许多有价值的信息。

SEEKER 定义的模式图中允许自环和多重边存在，有向边也只考虑主外键连接关系。连接元组树是以数据库中的元组为节点的一棵树，树的大小(用 size(T) 表示)是它拥有的元组的数量。与 BANKS、DISCOVER 和 IR-Style 相似，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，且以连接元组树表示的结果中要求每个叶结点都包含至少一个关键词。SEEKER 根据数据库模式图创建评分表图，然后用宽度优先搜索算法来生成子树候选评分表连接树集合。

虽然在数据图比在模式图上搜索有更好性能，但数据图的可度量性是一个问题。大量存储空间被用来

存储数据图，而内存容量有限。另外数据库更新会引起数据图产生变化。

2.3 查询语言

数据库检索系统需要定义它支持的语言的语法和语义。最基本的关键字查询形式就是单字查询，其他基于关键字查询的形式包括短语查询、近似查询、真假查询和模式匹配。

AND-语义的查询要求查询结果包含所有查询条件，但 OR-语义的查询要求一个查询结果包含至少一个查询条件。BANKS 支持的仅有 AND-语义的查询，DBXplorer 支持 AND-语义查询和子字符串的匹配，DISCOVER 和 ObjectRank 既支持 AND-语义也支持 OR-语义查询。

默认的查询范围包括所有的文本属性 (CHAR，VARCHAR)和关系。虽然多数系统声称查询范围可以轻易的扩展到包括元数据(关系名、属性名等)，但问题依然存在。在 BANKS 中，如果一个关键字匹配一个关系名，那么这个关系所有元组都会与这个关键字相关。因为 BANKS 为每个关键字匹配节点启动一个线程，如果这个关系元组太多，那么线程也越多。扩展的查询范围包括非文本属性，如数字和日期属性(DATE,TIME,INT,FLOAT,NUMERIC 等)。Agrawat[19] 给出了匹配数字属性的查询方法。

尽管违背了关键字查询是不需要用户知道数据库模式的规则，但有时在查询中让用户给出模式信息是有用的。如用户用“月：12”声明属性名“月”和属性值“12”。查询语言可被扩展到包括非文本的数字表达式，如查询“月:<12”即是查找“月”属性值小于 12 的元组。SEEKER 对查询语言进行扩展，实现了对数据库的元数据和数字属性的关键词查询。包含 3 类关键词;(1)Keyword,用来查询文本属性;(2)Keyword:Keyword，用来对文本属性进行包含元数据的查询；(3) Keyword:<op>Value，用于对数字属性的查询，但局限于对包含操作符的精确查询。第 2 类关键词前者匹配关系或属性，后者匹配在关系或属性上的关键词查询，帮助用户更精确地表达查询要求，有效地消除模糊性。如“Author: ‘Jim Gray’”，“Jim Gray”将被限制在和“Author”相匹配的关系或属性上。第 3 类关键词，前者匹配属性，后者在匹配上属性后进行条件查询，<op>是关系操作符，如≥，≤等，使用户更精确地表达查询要求。

2.4 查询结果排序

查询结果可以被定义为关系元组集合。在 ObjectRank 中，结果被定义为对应于查询且可以不包含任何查询关键字的元组；在 DBXplorer 中，结果被定义为多个连接元组集合且它必须包含所有查询关键字。把查询结果定义为有意义的信息单元，因而查询结果元组的连接有确定的含义，提交给终端用户时方便理解。

查询结果排序有三个因素：1)属性值的 IR 分数根据属性值包含的关键字数量计算。在 DISCOVER 和 SEEKER 中，属性值 IR 分数是由数据库系统计算的；2)结果树如 BANKS 的连接树和 SEEKER 的候选评分表连接树等通过它的大小计算分数，此外通过语义计算结果树是很有用的，在 DBLP 中检索作者更重要些，则 Author-Write-Paper 树比同样大小的 Paper-Cite-Paper 树有更高的分数；3)一个节点的分数取决于它和其他节点的连接，ObjectRank 考虑了这种因素。

每个查询结果都被分配了一个相应的分数，并以分数的递减顺序提交。分数函数与查询结果的定义有关。如果一个查询结果包含多个节点，则属性值的 IR 分数和结果树的结构需要考虑。如果一个查询结果只包含一个节点，那么结果树的结构是不必考虑的。ObjectRank 仅当计算结果分数时才考虑连接语义。BANKS 的每个树都被分配一个适当的分数，以分数的大小顺序排列，树的边进行了加权，计算所有边的分数、节点分数和树的分数。DBXplorer 的分数函数与连接数量有关，连接树的分数是与它的大小成比例的。DBXplorer 和 DISCOVER 的评分策略是查询结果包含的元组越少越好。IR-Style 的评分策略考虑了查询结果的大小，以及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 和 ObjectRank 采用了类似用于网页排序的 PageRank 评分方法。SEEKER 只将 Top-k 查询结果返回给用户，根据分数高低来对查询结果排序，查询结果含关键词越多，得分则越高。

2.5 查询结果提交

简单地把查询结果提交用户，可能造成用户的难以理解。由于数据库有多个物理表，信息可能被分散而产生逻辑连接信息，不容易被用户理解，因此不能

把查询结果简单地提交给用户。假定作者 W1 的论文引用了作者 W2 的论文，查询包含关键字 W1 和 W2，会产生多个具有相同结构的结果，太多类似的结果会让用户忽视最重要的结果。大多数用户花费大量的时间重构查询去完成有效的信息检索，适当的查询反馈是很好的策略。在反复反馈过程中，用户得到结果后检查它们，并标记适当的结果再进行查询重构。

最后用户需要浏览结果。BANKS 显示查询结果在一个嵌套表中，表示中间节点的表(包含一个代表叶节点的表)被嵌套在代表根节点的表中，并且支持浏览存储在数据库中的数据。DBXplorer 用简单的用户接口提交结果，用文本描述连接树，并在表中显示连接结果。DISCOVER 和 SEEKER 均以连接元组树的形式表示查询结果，然后通过转化为 SQL 语句查询从数据库中返回相应的关系表，方便用户浏览。

2.6 执行性能

为了有效的执行查询，不同的系统用不同的算法产生查询结果。数据库检索系统中一个重要的问题是如何有效的执行关键字查询。在查询过程中，用户只对少量的结果集 k(k 是输出最匹配的前 k 个结果集)，通过避免所有查询结果来有效地执行 Top-k 查询。其中一种方法是首先产生所有的结果，然后排序输出最先的 k 个结果，而这样效率是很低的。在 DISCOVER 中，作为中间结果的候选网络产生多个 SQL 语句，利用普通的连接表达式(如 a1c1b1)提高 SQL 语句的执行速度。选择查询[20]与 Top-k 连接查询[21]是很接近的问题，而 DISCOVER 通过产生最小最佳中间结果元组树，来重构以其为子表达式的其它元组树，这样在转化为 SQL 查询时可以加快访问数据库的速度。SEEKER 在执行 Top-k 查询时，为能快速从数据库中查找到符合用户需求的结果集，只对满足得分比第 k 个结果得分高的元组进行查询处理。相关限制条件的加入可以提高查询执行的效率。

3 总结与展望

本文对数据库检索系统已发展的一些原型系统进行了总结，关键词查询研究下一步的工作需要在如下方面开展:

(1) 实现对文本属性的模糊查询，可以更复杂地从文本之间的语义关系考虑。如果仅从词扩展算法方面去研究，可能导致同义词查询使得该查出来的结果没有查出来，而同名异义词往往导致不该查出来的结果却查出来了。这些问题是需要在今后的工作中不断考虑和做出改进的。

(2) 性能问题。如果数据库检索系统的查询性能很低，是不会被用于实际的。目前大量的测试都在简单模式和低容量数据下进行，当在更复杂数据库模式和海量数据的情况下，如何提高查询性能是个重要的问题。同时基于基准测试的实验是必要的，如有许多参考选项用于评价检索系统，有必要建立一个参考数据库用于评价数据库检索系统的性能。

(3) 寻找隐藏网页。大量数据以 Hidden Web 形式存储于数据库中，不能被搜索引擎检索到。用同样的关键词界面做搜索引擎，下一步是搜索引擎必须决定哪个关键字应该 用于数据库中的搜索。一个普通的方法是在字典中搜索所有的词，这个方法导致许多无用的搜索。因此，有必要寻找一个更好的方法用数量最少的关键字检索数据库。另一个简单的方法让搜索引擎对每个数据库检索系统发送搜索请求，并且综合所有返回结果。

(4) 基于本体的关系数据库语义检索[22]。随着本体和语义 Web 技术的快速发展和应用。未来工作中，将从关系数据库关键词检索技术出发，结合经济学领域示范语义平台的建设，研究和开发基于本体的关系数据库语义检索技术。

(5) 关键词提交搜索仍是用户与系统之间最重要的接口，如果能整合数据库检索、信息检索和 XML 检索系统，将有助于用户检索。

参考文献

1 Halotia G, Hulgeri A, Nakhey C, et al. Keyword Searching and Browsing in Databases Using BANKS. Agrawal R, et al, eds. Proc. of the Int'l Conf. on Data Engineering, 2002:431－440.

2 Agrawal S, Chaudhuri S, Das G. DBXplorer: A System for Keyword-Based Search over Relational Databases. Agrawal R, et al, eds. Proc. of the Int'l Conf. on Data Engineering, 2002:5－16.

3 Hristidis V, Papakonstantinou Y.DISCOVER: Keyword Search in Relational Databases. Bernstein PA, et al, eds. Proc. of the Int'l Conf. on Very Large Data Bases, 2002:670－681.

4 Hristidis V, Gravano L, Papakonstantinou Y. Efficient IR-style Keyword Search over Relational Databases. Freytag JC, et al, eds. Proc. of the Int'l Conf. on Very Large Data Bases, 2003:850－861.

5 Su Q, Widom J. Indexing Relational Database Content Offline for Efficient Keyword-Based Search. Technical Report, Stanford: Stanford University, 2003.

6 Balmin A, Hristidis V, Papakonstantinou Y. ObjectRank: Authority-Based Keyword Search in Databases. Nascimento MA, et al, eds. Proc. of the Int'l Conf. on Very Large Data Bases, 2004:564－575.

7 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.

8 Wang S, Zhang KL. Searching Databases with Keywords. Journal of Computer Science and Technology, 2005,20(1):55－62.

9 Cohen S, Mamou J, Kanza Y, et al. XSEarch:A Semantic Search Engine for XML. Very Large Data Bases, 2003: 45－56.

10 Florescu D, Manolescu I, Kossmann D. Integrating Keyword search into XML Query Processing. Albert V, et al, eds. Proc. of the Int'l Conf. on World Wide Web Conference. Amsterdam:ACM Press, 2000:119－135.

11 Goldman R, Shivakumar N, Venkatasubramanian S, et.al. Proximity Search in Databases. Gupta A, et al, eds. Proc. of the Int'l Conf. on Very Large Data Bases. New York: Morgan Kaufmann Publishers, 1998:564－575.

12 Hristidis V, Papakonstantinou Y, Balmin A. Keyword Proximity Search on XML Graphs. Dayal U, et al, eds. Proc. of the Int'l Conf. on Data Engineering. Bangalore: IEEE Press, 2003:367－378.

13 Guo L, Shao F, Botev C, et al. XRANK: Ranked Keyword Search over XML Documents. Halevy AY, et al, eds. Proc. of the Int'l Conf. on Management of Data. San Diego: ACM Press, 2003:16－23.

14 Wheeldon R, Levene M, Keenoy K. DbSurfer: A Search and Navigation Tool for Relational Databases. The Annual British National Conference on Databases. Edinburgh: Springer Berlin, 2004:144－149.

15 Dar S, Entin G, Geva S, et al. DTL's DataSpot: Database Exploration Using Plain Language. Proc. of the Int'l Conf. on Very Large Data Bases. San Francisco: Morgan Kaufmann Publishers Inc., 1998: 645－649.

16 蔡宏艳,姚佳丽,王珊.DETECTOR：基于关系数据库通用的在线关键词查询系统.计算机研究与发展, 2007,44(1):119－125.

17 Dijkstra.http://student.zjzk.cn/course_ware/data_structure/web/tu.htm.

18 Brin S, Page L. The Anatomy of a Large-Scale Hypertextual Web Search Engine. Proc. of the Int'l Conf. on World Wide Web Conference, 1998:107－117.

19 Agrawal R, Srikant R.Searching with Numbers. Proc. of the Int'l Conf. on World Wide Web Conference, 2002:855－870.

20 Fagin R, Lotem A, Naor M. Optimal Aggregation Algorithms for Middleware. Journal of Computer and System Sciences, 2003,66(4): 614－656.

21 Ilyas I, Aref W, Elmagarmid A. Supporting Top-k join Queries in Relational Databases. Proc. of the Int'l Conf. on Very Large Data Bases, 2004:207－221.

22 王珊,张俊,彭朝晖等.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):59－77.