Top-k keyword search using Skyline Sweeping and Improved Rank Function

[1]P. Vardhani, [2]S. Uma Maheswara Rao, [3]N. Tulasi Raju
[1,2,3]Dept. of CSE, Swarnandhra College of Engineering and Technology, Narsapur, AP, India

Abstract
Searching keywords in databases is complex task than search in files. Information Retrieval (IR) process search keywords from text files and it is very important that queering keyword to the relational databases. Generally to retrieve data from relational database Structure Query Language(SQL) can be used to find relevant records from the database. There is natural demand for relation database to support effective and efficient IR Style Keyword queries. This paper describes problem of supporting effective and efficient top-k keyword search in relational databases also describe the frame word which takes keywords and K as inputs and generates top-k relevant records .The results of implemented system with Skyline Sweeping (S.S) Algorithm shows that it is one effective and efficient style of keyword search

Keywords
Top-k, Keyword Search, Relational Database, Information Retrieval

I. Introduction

Internet search engines have popularized keyword based search. Users submit keywords to the search engine and a ranked list of documents is returned to the user. A significant amount of the world's enterprise data resides in relational databases. It is important that users be able to seamlessly search and browse information stored in these databases as well. Searching databases on the internet and intranet today is primarily enabled by customized web applications closely tied to the schema of the underlying databases, allowing users to direct searches in a structured manner. Examples of such searches within, say a bookseller's database may be "Books → Travel → Lonely Planet → Asia", or "Books → Travel → Rough Guides → Europe". With the growth of the World Wide Web, there has been a rapid increase in the number of users who need to access online databases without having a detailed knowledge of schema or query languages; even relatively simple query languages designed for non-experts are too complicated for such users. increasing amount of text data stored in relational databases, there is a demand for RDBMS to support keyword queries over text data. As a search result is often assembled from multiple relational tables, traditional IR-style ranking and query evaluation methods cannot be applied directly. This paper, Describes the effectiveness and the efficiency issues of answering top-k keyword query in relational database systems. We propose a new ranking formula by adapting existing IR techniques based on a natural notion of virtual document.

The rest of the paper is organized as follows: Section 2 Related work. Section 3 presents Problem Description Section 4 Frame works and algorithms optimized for efficient top-k retrieval. Experimental results are reported in Section 5.Section 6 concludes the paper.

II. Related Work

The FFF search mechanism at the websites that provides facts and figures may be augmented by DBXplorer technology. DataSpot is a commercial system that supports keyword-based searches by extracting the content of the database into a hyper base. Thus, this approach duplicates the content of the database, which makes data integrity and maintenance difficult. Microsoft's English Query provides a natural language interface to a SQL database. DISCOVER has proposed a breadth-first CN enumeration algorithm that is both sound and complete. The algorithm is essentially enumerating all sub graphs of size k that does not violate any pruning rules. The algorithm varies k from 1 to some search range threshold M. Three pruning rules are used and they are listed below. issue an SQL query for each CN and union them to find the top-k results by their relevance scores. DISCOVER2 introduce two alternative query evaluation strategies: sparse and global pipeline algorithms, both optimized for stopping the query execution immediately after the true top-k-th result can be determined

II. System Description

Consider a relational schema R as a set f relations $\{R_1, R_2, \ldots, R|R|\}$. These relations are interconnected at the schema level via foreign key to primary key references and denote Ri → Rj if Ri has a set of foreign key attribute(s) referencing Rj 's primary key attribute(s), following the convention in drawing relational schema graphs. For simplicity, we assume all primary key and foreign key attributes are made of single attribute, and there is at most one foreign key to primary key relationship between any two relations and do not impose such limitations in our implementation. A query Q consists of (1) a set of distinct keywords, i.e., Q = {w1,w2, . . . ,w|Q|}; and (2) a parameter k indicating that a user is only interested in top-k results ranked by relevance scores associated with each result. Ties can be broken arbitrarily. A user can also specify AND or OR semantics for the query, which mandates that a result must or may not match all the keywords, respectively. The default mode is the OR semantics to allow more flexible result ranking

A result of a top-k keyword query is a tree, T, of tuples, such that each leaf node of T contains at least one of the query keyword, and each pair of adjacent tuples in T is connected via a foreign key to primary key relationship. We call such an answer tree a joined tuple tree (JTT). The size of a JTT is the number of tuples (i.e., nodes) in the tree. Note that we allow two tuples in a JTT to belong to the same relation. Each JTT belongs to the results produced by a relational algebra expression — we just replace each tuple with its relation name and impose a full-text selection condition on the relation if the tuple is a leaf node. Such relational algebra expression (or its SQL equivalent) is also termed as Candidate Network (CN) [16]. Relations in the CN are also called tuple sets. There are two kinds of tuple sets: those that are constrained by keyword selection conditions are called non-free tuple sets (denoted as RQ) and others are called free tuple sets (denoted as R). Every JTT as an answer to a query has its relevance score, which, intuitively, indicates how relevant the JTT is to the query. Conceptually, all JTTs of a query will be sorted according to the descending order of their scores and only those with top-k highest scores will be returned.

IV. Frame Work and Algorithm

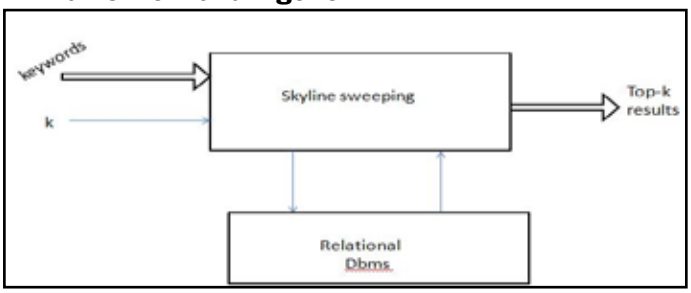

Fig. 1:

Sky line Sweeping algorithm designed to minimize the number of join checking operations, which typically dominates the cost of the algorithm. This intuition is that if there are two candidates x and y and the upper bound score of x is higher than that of y, y should not be checked unless x has been checked. Therefore, we should arrange all the candidates to be checked according to their upper bound scores. A na¨ıve strategy is to calculate the upper bound scores for all the candidates, sort them according to the upper bound scores, and check them one by one according to this optimal order. This will incur excessive amount of unnecessary work, since not all the candidates need to be checked.

Algorithm : Skyline Sweeping Algorithm
1: Q.push((m z }| {1, 1, . . . , 1),
calc uscore((m z }| { 1, 1, . . . , 1)))
2: top-k ← ∅
3: while top-k[k].score < Q.head().uscore do
4: head ← Q.pop max()
5: r ← executeSQL(formQuery(head))
6: if r 6= nil then
7: top-k.insert(r, score(r))
8: for i ← 1 to m do
9: t ← head.dup()
10: t.i ← t.i + 1
11: Q.push(t, calc uscore(t)) {According to Equation (4)}
12: if t.i > 1 then
13: break
14: return top-k

A result list,
top-k, contains no more than k results ordered by the descending real scores. The main data structure is a priority queue, Q, containing all the candidates (which are mapped to multi-dimensional points) according to the descending order of their upper bound scores. The algorithm also maintains the invariant that the candidate at the head of the priority queue has the highest upper bound score among all candidates in the CN. The invariant is maintained by (a) pushing the candidate formed by the top tuple from all dimensions into the queue (Line 1), and (b) whenever a candidate is popped from the queue, its adjacent candidates are pushed into the queue together with their upper bounds (Lines 8–13). The algorithm stops when the real score of the current top-k-th result is no smaller than the upper bound score of the head element of the priority queue; the latter is exactly the upper bound score of all the unprocessed candidates.

A. Rank Function

$$score(T, Q) = \sum_{t \in T} score(t, Q)$$

$$score(t, Q) = \sum_{w \in t \cap Q} \frac{1 + \ln(1 + \ln(tf_w(t)))}{(1 - s) + s \cdot \frac{dl_t}{avdl_t}} \cdot \ln(idf_w) \ ,$$

$$\text{where } idf_w = \frac{N_{Rel(t)} + 1}{df_w(Rel(t))} \ ,$$

tfw(t) denotes the number of times a keyword w appears in a database tuple t, dlt denotes the length of the text attribute of a tuple t, and avdlt is the average length of the text attribute in the relation which t belongs to (i.e., Rel(t)), NRel(t) denotes the number of tuples in Rel(t), and dfw(Rel(t)) denotes the number of tuples in Rel(t) that contain keyword w. The score of a JTT is the sum of the local scores of every tuple in the JTT.

CN	$t \in CN$	tf_{maxtor}	$tf_{netvista}$	$Score_t$	$Score_T$	Our Score
$c_3 \rightarrow p_2$	c_3	1	1	2.0	3.0	1.13
	p_2	0	1	1.0		
$c_1 \rightarrow p_1$	c_1	0	1	1.0	2.0	0.98
	p_1	1	0	1.0		
$c_2 \rightarrow p_2$	c_2	0	1	1.0	2.0	0.44
	p_2	0	1	1.0		

Fig. 2:

V. Results

The below diagrams represent keyword search results normal as well as top-3 key word searching hanks 2001.

1. 2001: HAL's Legacy (2001) (TV)
2. Gigantic Skate Park Tour: Summer 2002 (2002) (TV)
   aka *"Tony Hawk's Gigantic Skate Park Tour: Summer 2002"* - USA
3. TV Hunks and Babes 2006 (2006) (TV)
4. Danish Music Awards 2001 (2001) (TV)
5. Norah Jones & the Handsome Band: Live in 2004 (2004) (V)

Table 1: Top-3 Search Results on Our System

1	**Movies**: "Primetime Glick" (2001) Tom Hanks/Ben Stiller (#2.1)
2	**Movies**: "Primetime Glick" (2001) Tom Hanks/Ben Stiller (#2.1) ← **ActorPlay**: Character = Himself → **Actors**: Hanks, Tom
3	**Actors**: John Hanks ← **ActorPlay**: Character = Alexander Kerst → **Movies**: Rosamunde Pilcher - Wind über dem Fluss (2001)

running example (shown in Figures above). In the example, R = {P,C,U}.1 Foreign key to primary key relationships are: C → P and C → U. A user wants to retrieve top-3 answer to the query "maxtor netvista". Some example JTTs include: c3, c3 → p2, c1 → p1, c2 → p2, and c2 → p2 ← c3. The first JTT belongs to CN CQ; the next three JTTs belong to CN CQ → PQ; and the last JTT belongs to CN CQ → PQ ← CQ. Note that
c3 → u3 is not a valid JTT to the query, as the leaf node u3 does not contribute to a match to the query. A possible answer for this top-3 query may be: c3, c3 → p2, and c1 → p1. We believe that most users will prefer c1 → p1 to c2 → p2, because the former complaint is really about a IBM Netvista equipped with a Maxtor disk, and
that it is not certain whether Product p2 mentioned in the latter JTT is equipped with a Maxtor hard disk or not.

COMPLAINTS

rid	prodId	custID	date	comments
c_1	p121	c3232	6-30-2002	disk crashed after just one week of moderate use on an IBM Netvista X41
c_2	p131	c3131	7-3-2002	lower-end IBM Netvista caught fire, starting apparently with disk
c_3	p131	c3143	8-3-2002	IBM Netvista unstable with Maxtor HD

PRODUCTS

rid	prodId	manufacturer	model
p_1	p121	Maxtor	D540X
p_2	p131	IBM	Netvista
p_3	p141	Tripplite	Smart 700VA

CUSTOMERS

rid	custId	name	occupation
u_1	c3232	John Smith	Software Engineer
u_2	c3131	Jack Lucas	Architect
u_3	c3143	John Mayer	Student

Schema: $P^Q \leftarrow C^Q \rightarrow U$

Size	CN ID	CN	Valid?	Violates
1	CN_1	P^Q	Y	
1	CN_2	C^Q	Y	
2	CN_3	$P^Q \leftarrow C^Q$	Y	
2		$C^Q \rightarrow U$	n	Rule (2)
3		$P^Q \leftarrow C^Q \rightarrow U$	n	Rule (2)
3		$P^Q \leftarrow C^Q \rightarrow P^Q$	n	Rule (3)
3	CN_4	$C^Q \rightarrow P^Q \leftarrow C^Q$	Y	
3		$U \leftarrow C^Q \rightarrow U$	n	Rules (2, 3)
4	⋮	⋮	⋮	

VI. Conclusion

This paper, studied supporting effective and efficient top-k keyword queries over relational data bases.and proposed a new ranking method that adapts the state-of-the art IR ranking function and principles into ranking trees of joined database tuples. ranking method also has several salient features over existing ones. We also studied query processing method tailored for our non-monotonic ranking functions. Two algorithms were proposed that aggressively minimize database probes. We have conducted extensive experiments on large-scale real databases. The experimental results confirmed that our ranking method could achieve high precision with high efficiency to scale to databases with tens of millions of tuples.

References

[1] S. Agrawal, S. Chaudhuri, G. Das. DBXplorer,"A system for keyword-based search over relational databases", In ICDE, pp. 5–16, 2002.

[2] H. Bast, D. Majumdar, R. Schenkel, M. Theobald, G. Weikum,"Io-top-k: Index-access optimized top-k query processing", In VLDB, pp. 475–486, 2006.

[3] G. Bhalotia, A. Hulgeri, C. Nakhe, S. Chakrabarti, S. Sudarshan,"Keyword searching and browsing in databases using BANKS", In ICDE, pp. 431–440, 2002.

[4] S. B¨orzs¨onyi, D. Kossmann, K. Stocker,"The skyline operator", In ICDE, pp. 421–430, 2001.

[5] K. C.-C. Chang, S. won Hwang,"Minimal probing: supporting expensive predicates for top-k queries", In SIGMOD, pp. 346–357, 2002.

[6] S. Chaudhuri, R. Ramakrishnan, G. Weikum,"Integrating db and ir technologies: What is the sound of one hand clapping?", In CIDR, pp. 1–12, 2005.

[7] G. Das, D. Gunopulos, N. Koudas, D. Tsirogiannis, "Answering top-k queries using views", In VLDB, pp. 451–462, 2006.

[8] B. Ding, J. X. Yu, S. Wang, L. Qin, X. Zhang, X. Lin, "Finding top-k min-cost connected trees in databases", In ICDE, 2007.

[9] R. Fagin,"Combining fuzzy information from multiple systems", J. Comput. Syst. Sci., 58(1), pp. 83–99, 1999.

[10] R. Fagin, A. Lotem, M. Naor,"Optimal aggregation algorithms for middleware", In PODS, 2001.

[12] R. Goldman, N. Shivakumar, S. Venkatasubramanian, H. Garcia-Molina,"Proximity search in databases. In VLDB, 1998.

[13] T. Grabs, K. B¨ohm, H.-J. Schek,"Powerdb-ir – information retrieval on top of a database cluster", In CIKM, pp. 411–418, 2001.

[14] P. J. Haas, J. M. Hellerstein,"Ripple joins for online aggregation", In SIGMOD 1999, pp. 287–298, 1999.

[15] V. Hristidis, L. Gravano, Y. Papakonstantinou,"Efficient IR-Style Keyword Search over Relational Databases", In VLDB, 2003.

[16] V. Hristidis, Y. Papakonstantinou,"DISCOVER: Keyword search in relational databases", In VLDB, pp. 670–681, 2002.

[17] I. F. Ilyas, W. G. Aref, A. K. Elmagarmid,"Supporting top-k join queries in relational databases", VLDB Journal, 13(3), pp. 207–221, 2004.

[18] V. Kacholia, S. Pandit, S. Chakrabarti, S. Sudarshan, R. Desai, H. Karambelkar,"Bidirectional expansion for keyword search on graph databases", In VLDB, pp. 505–516, 2005.

[19] B. Kimelfeld, Y. Sagiv,"Efficient engines for keyword proximity search", In WebDB, pp. 67–72, 2005.