International Journal of Applied Engineering Research
ISSN 0973-4562 Volume 10, Number 10 (2015) pp. 26629-26640

http://www.ripublication.com

Predicting the Hard Keyword Queries over Relational Databases

B. Rupesh and M.Sunil Kumar

PG Scholar, Dept of CSE, SVEC, Tirupati,
Mail id: rupesh.seheri@gmail.com
Assistant Professor(SL), Dept of CSE, SVEC, Tirupati,
Mail id: sunilmalchi1@gmail.com

Abstract:

Keyword queries based on database gives an solution to data. Even though it suffers due to low ranking quality. So, a database provides best alternatives to such type of hard queries. The characteristics of hard keyword queries over Relational Databases are analyzed for the novel composition to evaluate the amount of problems, and content of the Relational Databases is proposed and then hard keyword query prediction is evaluated against INEX and SemSearch benchmarks. In this work the hard keyword queries over Relational Databases is predicted by using Structured Robustness algorithm with Skyline Sweeping Algorithm and Joined Tuple Tree Technique.

Keywords: Keyword search, Relational Databases, Ranking function, Query score, top-k.

1 Introduction:

Basically there is no particular keyword query search in Relational databases, but our work has to be done this work and satisfying the user requirement. Predicting the difficult keyword queries in Relational Databases is nothing but to predict the hard keyword queries based on the score and rankings of queries. A Keyword query interface (KQIs) is an interface between users to Relational databases. It is nothing but collection of structured, unstructured and semi structured data in Relational databases. Here databases have pulled in much consideration in the most recent decade because of their adaptability and usability in browsing the data. Considering that any element throughout information the set which contains the particular query key phrases is often a probable answer, keyword queries commonly have number of conceivable solutions. Keyword query interface must recognize the details wants

behind keyword queries as well as get ranking the particular answers. with the goal that the solutions wanted to show up at the highest point of the list. A Relational database include entities and entities include characteristics (attributes) that will take characteristic (attributes) valuations..

A portion of troubles are given an answer to queries as below steps:

- First, normally a user do not specify the correct schema elements for each query words. For this, a query Q1: Cricket World Cup matches on the ICC tournaments. A ICC database does not know if the user is really curious in sports as their title is World Cup distributed by the International Cricket Council. The interface (KQIs) should discover the preferred characteristics (attributes) related to each expression within the query.
- Next, this schema on the result isn't given, i.e, users usually do not allow plenty of data to select exactly their desired entities.

For instance, a Query Q1 may return Tournaments as well as Players as well as Records of cricket and presents a more detailed analysis of the resources of issues and also ambiguity in hard queries on Relational databases. In this work without interfering of MAP (Mean Average Precision) method we can calculate the MAP values of a query by using Ranking function.

Here introduce the problem of predicting the degree of the issues for queries over Relational databases and using some technologies we can calculate the Query Score(QS), which measures the issues of user query based on variations between the rating of the same query over Relational databases and performing Ranking Function(RF) i.e., Skyline Sweeping Algorithm to compute the queries in ascending order. In this work we are using a *dominate* relationship among queries, it shows the upper bound scores and lower bound scores of a queries.

2 Related Work:

The success and also the productivity difficulties regarding giving an answer to top-k keyword problem inside relational repository methods[5]. A fresh ranking formulation through aligning recent IR(Information retrieval) techniques dependant on an organic and natural notion regarding electronic file. Weighed against preceding techniques, the new ranking procedure is simple still efficient, along with will abide by people awareness plus research productive problem finalizing options for the new ranking procedure, along with suggest algorithms that contain minimum accesses towards the repository. Carried out comprehensive findings with large-scale true directories using a couple well-liked RDBMSs[2][3]. The fresh final results display substantial enhancement towards the alternative techniques in terms of access success along with productivity. For example business programs for example Customer Relation Management System (CRM), along with private or public programs, for example World-wide-web blogs along with wiki web-sites. Since dominant way of querying free textual content is usually through keyword seek, there exists a normal

requirement regarding relational directories to back up efficient along with productive IR-style keyword queries.

Mostly focus on the challenge regarding supporting efficient along with productive top keyword search inside relational databases[2]. Even though many RDBMSs help full text search, many people just allow rescuing relevant tuples coming from inside the very same relationship. An original characteristic regarding keywords search over RDBMSs is usually which search result can be set up coming from relevant tuples in numerous contact such that they are interconnected along with collectively relevant to this keyword query.

A Joined Ttuple Tree(JTT) is one of the final results created by a new relational algebra phrase and have absolutely used quantity of keyword queries with number of queries[3]. The main topic in this document is usually Tips on how to report ranking operates along with how you can produce along with get this SQL queries with the candidate networks. A ranking perform is usually nothing but come back a new ranking value for each line in the partition, depending a great this perform that is certainly used plus a ranking operates are generally no deterministic.

For instance, imagine the person enthusiastic about the actual competition inside the 1984-1985 Olympic World cup issues the actual question of World cup resistant to the TRCE AP87 collection of information articles or blog posts[5]. If those two queries are the only evidence to the system has about what the user means, it's simply impossible for the system to come back the actual Olympic articles continually larger inside the ranked list than the articles with regards to World cup chess tournaments. Articles regards to World cup chess tournament are prevailing, inside the decided on assortment among the articles that utilize the query terminology frequently. Though anyone may possibly not have recognized which there is a World cup inside anything other than Olympic, the user might get a ranked list along with chess articles prevailing is actually Olympic articles interspersed occasionally throughout.

2.1 Motivations:

A problem of predicting the keyword queries over Relational Databases is not effective. This observation has motivated recent research on free form keyword query search over Relational Databases with the help of Clarity score, Unstructured Robustness Method, Weighted Information Gain, Normalized Query Commitment.

3 Problem Domain:

More recently days the keyword query search can be a process employed by all kind of organizations. Throughout Relational databases the keyword query search is utilized to search for the tuples by giving queries, but majority of techniques that have small efficiency and also more space for storing intended for saving the results. So, we find the efficient way of process for keyword query search in relational databases.

3.1 Problem Definition:
To reduce the low ranking quality of keyword queries over Relational databases and produce an effective and efficiency top keyword query search in Relational databases.

3.2 Statement of the Problem:
The hard keyword queries over Relational Databases is predicted by using Structured Robustness Algorithm with Skyline Sweeping Algorithm and Joined Tuple Tree Technique.

4 Problem Formulation:
To predict the keyword based queries over Relational databases and reduce their low ranking quality and produce an effective top keyword queries by using the implementation of Structured robustness algorithm with Skyline sweeping algorithm and Joined tuple tree Technique.

Assume,

i. A Relational Database contains a large amounts of datasets.
ii. A keyword queries in Relational Database is Q = $\{q_1, q_2, \ldots\ldots q_m\}$.
iii. A retrieval function based on the given Relational Database and query Q is e (E, Q, RD).
iv. A keyword search system returns a ranked list of entities in Relational Database is RL (Q, e, RD).
v. A Ranking score of a document returned by the retrieval systems for an input query may estimate the similarity of the query over Relational Databases.
vi. Robustness based that the results of an easy query are relatively stable against the un filled queries.
vii. Ranking function contains all words in Query Q.
viii. T contains all the words in Q (query)

Notations:
Present work will be done by the following notations,
Q: Query
T: Number of tuples
$d_r(T)$: Relations of the tuples of T
$d_w(T)$: Weight of the tuples of T
$d_p(T)$: Weight of the edge of T
e: Retrieval function
d_i: Non free tuple sets
Sim: Spearman rank
$x_{RD:}$ Frequency tuples in Relational Database
A: $X_{RB:}$ A set of Relational Database attribute values
P: Number of edges
N: Number of noisy iterations

RL: Ranked list
a, b: Assume two queries
RD: Relational Database
$t_{i:}$ Keyword term.

4.1 Mathematical model

Structure robustness query score(QS score) for query Q in relational database RD given retrieval function e is:

$$QS\ (Q, e, RD, X_{RD}) = P\{Sim(RL(Q, e, RD), RL(Q, e, X_{RD}))\} \quad (1)$$

Where, e = Retrieval function i.e., e(E, Q, RD) where E is a Number of entities.

Applying the number of different tuple sets in the above equation we can reduce the low ranking quality of a query:

$$QS(Q, e, RD, X_{RD}) = \sum Sim\ (RL(Q, e, RD), RL(Q, e, f x_{RD})) \quad (2)$$

Where,
'Sim' denotes Sperman rank correlation b/w displayed ranked lists,
'$f x_{RD}$ ·denotes total number of tuple set under the data.

If there are two keyword queries a and b, to calculate the higher bound scores for these keyword queries, Using *dominate* relationship among those queries is: $a.d_i \leq b.d_i$

Where,
d_i is non free tuple sets
(Qscore(a) ≤ Qscore(b))

Then calculate the most approximated search keyword of a query result in the score i.e., Ranking function is:

Queryscore(T, Q) = $\frac{1}{size(T)}$ if T contains all words in query Q, otherwise it will be nil 0.

$$\text{Ranking Function F => } Qscore(T, Q) = d_r(T) + d_w(T) + d_p(T) \quad (3)$$

Where,
$d_r(T)$ = Relations of the tuples of T
$d_w(T)$ =Weight of the tuples of T
$d_p(T)$ =Weight of the edge of T

Finally combines (1)(2)(3)equations by using the joined tuple tree technique:

$$\text{Combine(QS(T, Q), size(T))} = \frac{\sum \text{QS(Q,e,RD,XRD)}, \text{Queryscore(T,Q)}}{size(T)}$$

4.2 Algorithms:

4.2.1 SR(Structured Robustness) Algorithm

The Structured Robustness Algorithm computes the definite QScore, based on the top keyword result entities. The world entries are stored in B (metadata) and IL (inverted indexes) in the SR Algorithm. SR Algorithm computes the clamor in the Relational databases.

Input: Query Q, Top k result list L of Q, ranking function g, Metadata B, Inverted indexes IN, Number of corruption iteration N.

Output: QS score for Q.

Step 1: QS←nil; W← {}; //W is caches for keyword query in Q
Step 2: for i =1→ n do
Step 3: IN^1 ← I: B^1← B: L^1 ← L; //Corrupted copy of IN, B and L
Step 4: for each result R in L do
Step 5: for each character value A in R do
Step 6: A^1 ← A; //Debased versions of A
Step 7: for each keyword query j in Q do
Step 8: Compute # of k in A^1
but not in W, calculate and cache them
Step 9: if # of k varies in A^1 and A then
Step 10: Upgrade A^1, B^1 and entry of k in IN^1 ;
Step 11: Sum A^1 to R^1 ;
Step 12: Sum R^1 to L^1 ;
Step 13: Rank L^1 using e, which returns L, based on IN^1, B^1 ;
Step 14: QS += Sim(L, L^1) ; //Sim computes Sperman correlation
Step 15: returns QS← QS/n ;

SR Algorithm loops every attribute value in each top keyword result and tests whether it must be corrupted. As noted before, one entity may have hundreds of attribute values. Note that the attribute values that do not contain any query term still must be corrupted for the 2nd and 3rd levels. This is an account of their traits(attribute) or substance (entity) sets may contain some query keywords. This will largely increase the number of attribute values to be corrupted.

4.2.2 Skyline sweeping Algorithm

To calculate the rankings of a existed queries based on their scores we have to implement skyline sweeping algorithm. By implementing this algorithm we can easily sorting the queries based on score. It will predict the queries automatically depending on the scores and arrange in increasing order.

Step 1: Q.push((1, 1, ….1), calc_Qscore((1, 1, …1)))
Step 2: Top k←0
Step 3: If n ≠ nil then
Step 4: Top k.insert(n, score(T, Q))

Step 5: for IN ←1 to m do
Step 6: H← top()
Step 7: g.i ← g.i+1
Step 8: Q.push(g, calc_Qscore(T, Q))
Step 9: if g.i >1 then
Step 10: Break
Step 11: Return top keyword.

The algorithm shows, A result list top keyword, contains no more than top results ordered by the decreasing the Query scores. Pushing the Query to the list and it will find out the score of a query. Initially the ranking score is 0, then inserting a query based on number of words that contained in query. In step 7 the score will be increased based on scores and a search menu displayed number of query results that is user requirements. In step 9 when the query score matched and the value is less than 1 it will break the function. Finally returns the top keywords based on user requirements.

4.2.3 Verification and Validation

Input Keyword: "*Ranjith"*.

- First calculate the score of an keyword related to "Ranjith".
- First of all the score of an keyword "Ranjith" is initially null, to calculate the QScore for "Ranjith(Q)" over relational database, then the retrieval function of Q is 'e' as follows:

QS (Ranjith, e, RD, X_{RD}) = P{Sim(RL(Q, e, RD), RL(Q, e, $f x_{RD}$))}

Here (RL(Q, e, RD) is Relational database and RL(Q, e, X_{RD}) is Noisy version of Relational database. Q = Keyword query is a set { q_1, ……..$q_{|Q|}$} of terms, Where |Q| is the number of terms in Q.

- X_{RD} is nothing but combination of Attributes and entity sets in a Noisy version of Relational database, i.e., $X_{RD}=(X_A, X_T, X_E)$

Here X_T is a set of attributes in T and X_E is Set of entity sets in E, then X_A (set of attribute values)= (X_T, X_S).

- Retrieval function e (E, Q, RD) where E Є RD to Q.

Here 'E' is nothing but entity set, then a ranked list is RL(Q, e, RD)=List of entities in Relational database. Then the retrieval function 'e' is (college, Ranjith, RD).

- A 'Ranjith' keyword may appear in any attribute value of attribute set like student name, faculty, Head of the Department, watchmen etc…
- Finally calculating the score of an query is by combining all the equations as shown in above,

QS(Q, e, RD, X_{RD}=∑Sim(RL(alone, e, RD), RL(Ranjith, (college, Ranjith, database, $f x_{RD}$)))

Here $f x_{RD}$ is a non negative integer.

Now, we can set those scored keywords by ranking wise by applying the skyline sweeping algorithm i.e.

Step 1: Ranjith. push((1, 1, ….1), calc_Qscore((1, 1, …1)))
Step 2: Don't know the top keyword so we assume that Top k←0
Step 3: If r ≠ nil then
Step 4: Top k. insert(1, score(Term, Ranjith) //n value is 1, 2…..
Step 5: for IN value is 1 to m do //Inverted index (IN) and Metadata (m)
Step 6: Term← top()
Step 7: term g. i ← g.i+1 //term. score(1)+1
//term. score(2)+2
// term. score(3)+3 etc
Step 8: Ranjith. push(term g, calc_Qscore(Term, Ranjith)) //$Ranjith_9$
//$Ranjith_8$
//Ranjith etc
Highest priority attribute values should be displayed on top of the list.
Step 9: if term g.i(value) >1 then //The process is executed//
Step 10: Break;
Step 11: Return top keyword.

Then finally combines scored and ranked list by using joined tuple tree technique,

$$\text{Combine(QS(T, Q), size(T))} = \frac{\sum \text{QS(Q,e,RD,XRD),Queryscore(T,Q)}}{size(T)}.$$

The process executed is shown below in fig 1,

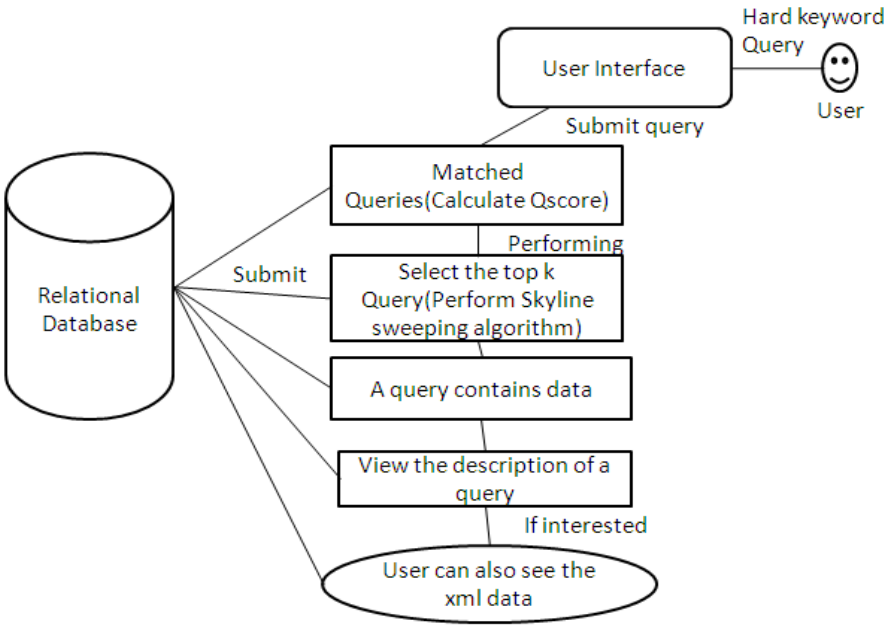


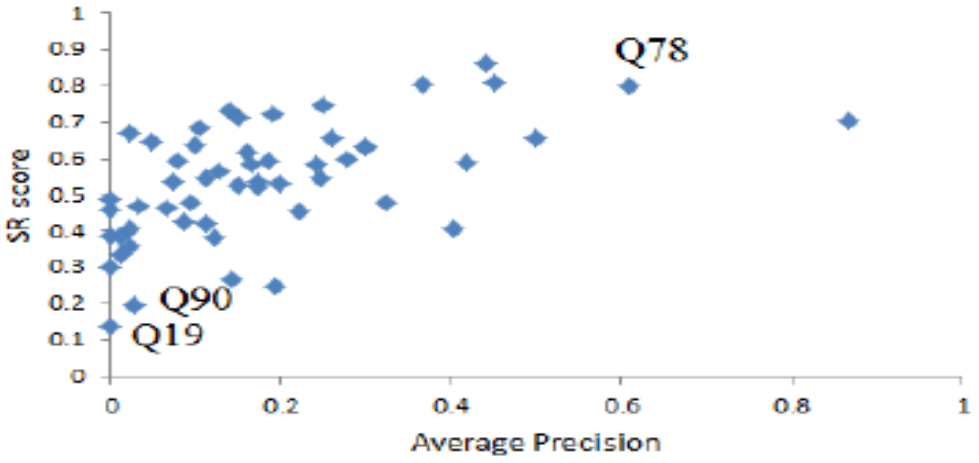
Fig 1: Framework of Proposed Model

5 Result:
Our experimental results shown below,

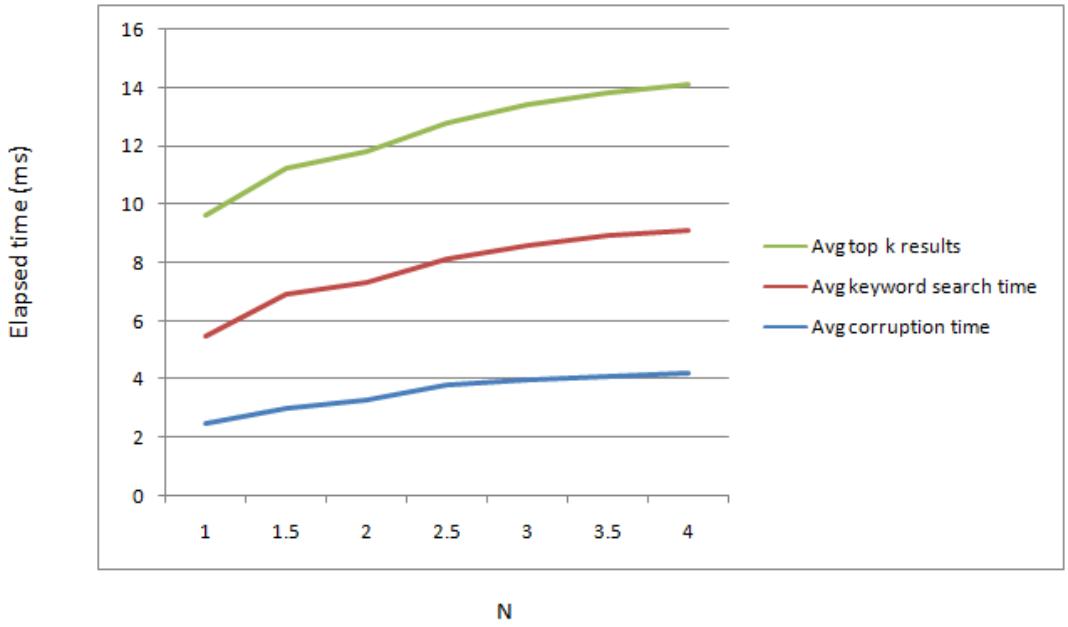
Table 1: - Experimental result

Format	Input	Output
Query	Connect a database using query, which contain data.	Database is connected successfully.
Keyword Search	Enter the keyword query as per the user requirement.User needs the details of Ranith HOD cse name, but user simply enters the Ranjith. Ex: Ranjith	List of required records are displayed. Our extensive process executed internally and the results list shows like. Ex: Dr.k.Ranjith (cse HOD) Ranjith (student mc) Ranjith (student cse) Ranjith (Plytech)
	Select the top keyword result based on user requirements. Ex: Ranjith(cse HOD)	A keywords can be displayed based on the ranking wise. Ex: Dr.k. Ranjith (cse HOD)
	View the keyword results.	User can view the entire description of the keyword. Ex: Name: Prof Dr.K. Ranjith Head of CSE dept Phno: 9845632159 Qualifications: Mtech, Phd Address: Vijayawada

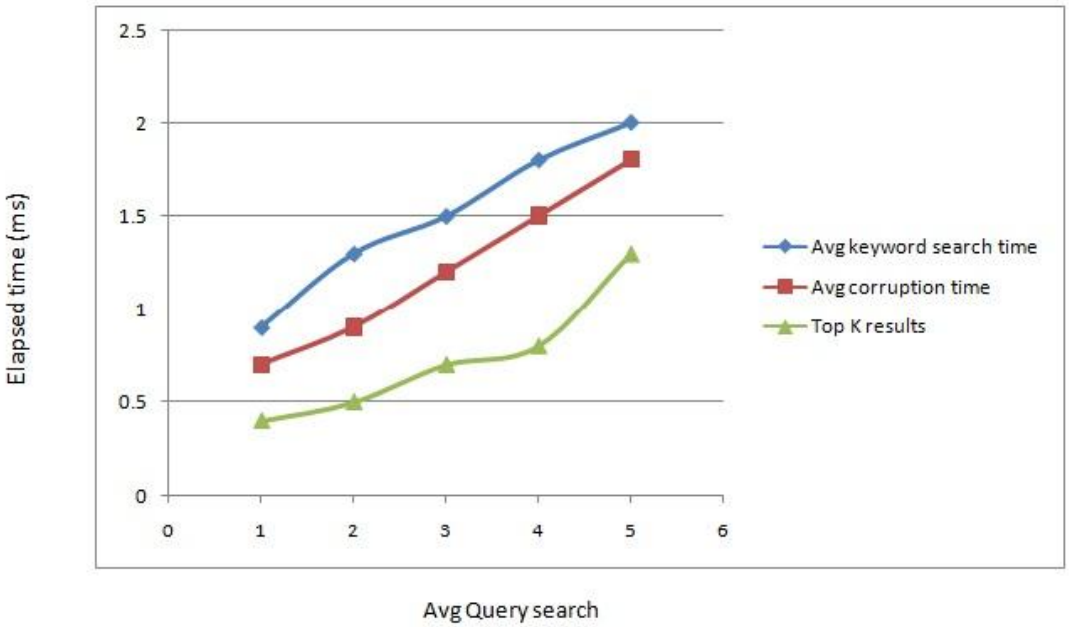
6 Comparison of results:
The results of using INEX and SemSearch benchmarks respectively,



The structure robustness algorithm showing the time to compute the SR score only depends on the top K results, here the top K results are corrupted and re-ranked and to increasing the data set size will only increase the query processing time,



Our experimental results shows the Skyline sweeping algorithm with joined tuple tree technique shown in belowto reducing the keyword query search time and the average corruption time and top k results re-rank process in fraction, if seconds based on the clarity score and rankings,



7 Conclusion:

In this paper, we studied predicting the keyword queries and supporting effective and efficient top keyword queries over relational databases. We added a new ranking algorithm that getting ranking function and principles into ranking sub trees of joined tuples in relational database. We have led test on college real databases. The experimental results also reduce the low ranking of a keyword query and satisfy the user requirements on relational databases.Our future enhancement is to calculate the query size and weight, depending on the number of characters that mingles in keyword queries.

8 Acknowledgments

I have taken efforts in this paper. However, it would not have been possible without the kind support and help of my guide and organization. I would like to extend my sincere thanks to all of them. I am highly indebted to ***Mr. M.Sunil Kumar***(Senior Assistant Professor) for their guidance and constant supervision as well as for providing necessary information regarding the project and also for their support in completing the project. I would like to express my gratitude towards my parents and member of ***Sree Vidyanikethan College of Engineering*** for their kind co-operation and encouragement which help me in completion of this project.

9 References:

[1] V. Hristidis, L. Gravano, and Y. Papakonstantinou, "*Efficient IRStyle Keyword Search over Relational Databases*, " in *VLDB 2003*.

[2] Y. Luo, X. Lin, W. Wang, and X. Zhou, "*SPARK: Top-k Keyword Query in Relational Databases*, " in *SIGMOD*, 2007.

[3] S. C. Townsend, Y. Zhou, and B. Croft, "*Predicting Query Performance*, " in *SIGIR*, 2002.

[4] Y. Zhou and B. Croft, "*Query Performance Prediction in Web Search Environments*, " in *SIGIR*, 2007.

[5] S. Cheng, A. Termehchy, and V. Hristidis, "*Predicting the Effectiveness of Keyword Queries on Databases*, " in *ACM CIKM*, 2012.

[6] Y. Zhou and B. Croft, "*Query Performance Prediction in Web Search Environments*, " in *SIGIR*, 2007.

[7] Liu.F., Yu.C., Meng.W., and Chowdhury.A. (2006) "*Effective Keyword Search in Relational Databases*, " ACM *SIGMOD* Int'l Conf. Management of Data, 563574.

[8] Sharmili C and Rexie J.A.M, (2014) "*Efficient Keyword Search Methods In Relational databases"* in IJERA 2014.

[9] B. He and I. Ounis"Query performance prediction" *Inf. Syst., vol. 31, no. 7, pp. 585-594, Nov., 2006.*

[10] A. Shtok, O. Kurland and D. Carmel "Predicting query performance by query-drift estimation" *Proc. 2nd ICTIR, pp. 305-312, 2009, Heidelberg, Germany.*

[11] J. Han, M. Kamber and J. Pei Data Mining: Concepts and Techniques *2011, Morgan Kaufmann, San Francisco, CA.*

[12] E. Mittendorf and P. Schauble "Measuring the effects of data corruption on information retrieval" *Proc. SDAIR 96 Conf., pp. 179-189, 1996, Las Vegas, NV, USA.*