International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 3 Issue 5, May - 2014

New Schema Free Method for Text Search

Mahesh Mali (Author) Computer Department Vidyalankar Institute of Technology Mumbai, India

Pankaj Vanwari (Author)
Computer Department
Vidyalankar Institute of Technology
Mumbai, India

Vipul Dalal (Author)
Computer Department
Vidyalankar Institute of Technology
Mumbai, India

***Abstract* -All real-world databases often have extremely complex schemas may be due to size or design of database. Database schema with lots of entities and their relationships, each with a multiple attributes. It is challenging for new users to explore the data and formulate queries in order to get knowledge from database. Schema free text search can address this issue by allowing users with less or no knowledge of the schema to formulate database queries. The paper will show that most current Schema Free Query Interfaces provide a very limited degree of design independence the proposed method. Proposed paper also introduces a novel and improved Duplication and Relationship Aware Coherency Ranking (DRA-CR) based on information-theoretic relationships weight among the data items in the database, and shows that DRACR is design independent. The study using multiple real world data sets shows that the ranking quality of improved DA-CR is better than or equal to that of current ranking systems.**

Keywords - Database Text Search, Schema Free Text Search, Design Independent Text Search

I. INTRODUCTION

This Knowledge of Database Schema is extremely crucial for writing queries to fetch data from database using some query [3], [5], [6]. Generally relational database schema consists of lot of tables which may contain data in multiple columns [5]. Data may be related with the other data present in same table or may be data present in other table. Therefore, it is very difficult for end users (generally with to very less or no technical knowledge) to understand the database schema [3]. Database schema is always evolving over time, so changing schema is a great challenge even for expert computer user to formulate correct query for fetching required data[5], [6].

The Schema free query interfaces (SFQI) are proposed solution for these problems for web databases [3], [5], [6] (XML). Such interfaces will accept the query as search keywords and returns answer by applying some concepts on keywords. For example, if user submits a query to find link between Mahesh and IT department in universities database, the SFQI should return there relation as answer with all intermediate nodes or relations (or tables) and if keywords are not related it should display it. This operation of text search here onwards we will refer as Schema Free Text Search (SFTS).

The DBA (Database Administrator) may change the database design over the period of time [5], [6]. The main reason for changes is to normalize database or may be for doing time/space optimization. Such database changes may requires some or more changes in query. The proposed SFTS should manage such changes. So there is no effect on query for achieving same query answers. For example DBA may add a node Topics to existing schema or may delete the node Library which may or may not contain further sub tree. This schema change should not affect answer of above query.

The big success of web search engines like Google, Yahoo etc. makes the keyword search very popular search option for the computer end user for searching data in huge dataset [3].

This paper will explore the design independence along with schema independence to SFQI which has not done previously in many papers. SFTS is a simple way to query any databases since it allows users to formulate queries without the knowledge of complex query languages and the database schema. An important feature of SFTS text search is that it ranks the query answer so that the most relevant results appear first [2], by applying filtering techniques and then by using improved IR style ranking which can exactly capture the hierarchical structure of data and resolve ambiguity. Besides, the popularity of query results is designed to distinguish the results with comparable relevance scores [9]. At last the final ranked list of results will be displayed to the user [3].

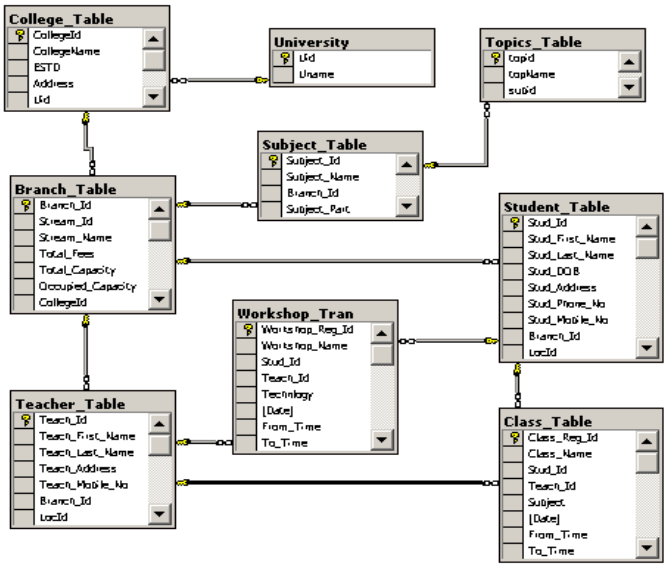


Figure 1. Sample University Databases.

II. BACKGROUND

A. *Basic Definitions*

SFQI [3] paper is describing model of DB [3] as a tree T = (r, V, E, L, C, A), where V is the *set of nodes* in the tree, E is the set of edges between members of V, r is the root element, C is a subset of the leaf nodes (i.e. C ⊂ V) of the tree called *content nodes*, L assigns a *label* to each member of V-C and A assigns data value to each content node [3]. Parent of content node now we call it as *attribute*. Each *sub tree* U = (r_u, V_u, E_u, L_u, C_u, A_u) of tree T is a tree such that $r_u \subseteq r$, $V_u \subseteq V$, $E_u \subseteq E$, $L_u \subseteq L$, $C_u \subseteq C$ and $A_u \subseteq A$ [3].

In SFTS we will describe model of DB [3] as a graph G = (V, E, L, C, A), where V is the *set of nodes* in the graph, E is the set of edges between members of V, C is a subset of the leaf nodes (i.e. C ⊂ V) of the graph called *content nodes*, L assigns a *label* to each member of V-C and A assigns data value to each content node [3]. Parent of content node now we call it as *attribute*. Each *sub graph* S = (r_S, V_S, E_S, L_S, C_S, A_S) of tree T is a tree such that $r_S \subseteq r$, $V_S \subseteq V$, $E_S \subseteq E$, $L_S \subseteq L$, $C_S \subseteq C$ and $A_S \subseteq A$.

The graphs G_1 and G_2 are (label) isomorphic, If the nodes of G_1 can be mapped to the nodes of G_2 in such a way that node labels are preserved and the edges of G_1 are mapped to the edges of G_2 [5]. A **pattern** is total number of isomorphic graphs. The pattern can be obtained from the prefix string by removing the content. For Example *FY* and *DOT.Net* has two instances or patterns. P_1 contains Class, Student and Workshop while P_2 contains Class, Student, Branch, Student and Workshop. **Size** of pattern is number of leaf nodes it contains. A pattern P_1 is a **sub pattern** of pattern P if each of P_1 instances is a *sub graph* of one of P's instances. The value of *sub graph* is list of content associated with leaves. For Example the value of *sub graph* with pattern *class, Teacher, Workshop* is ("FY","DOT.Net").

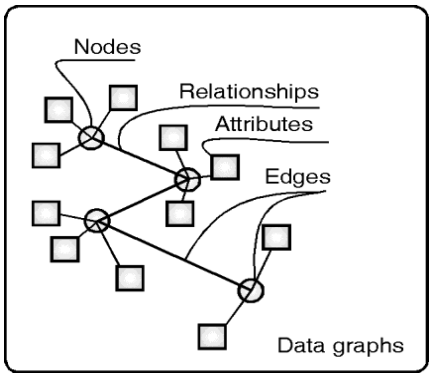


Figure 2. Data in Graphs.

The sub graph S is called as **candidates answer (CA)** to query (*Q)* if each of its content nodes contains at least one instance of each keyword term in *Q* [5].

The IR community working on retrieval techniques for text-centric data [5], where structures are simple. Extracting metadata (e.g. College, Student, Class etc.) from text centric content, in such cases structure plays an important role [5].

B. *Related Work*

In tree data model, LCA (lowest common ancestor) semantics is proposed and studied in [10] to find XML nodes, each of which contains all query keywords within its sub trees. Afterwards, SLCA (smallest LCA [11]) is proposed to find the smallest LCAs that do not contain other LCAs in their sub trees. Such approach also been taken to apply keyword search in XML documents (e.g., XSearch [14] and DISCOVER [15]).

Ranking mechanisms [13], [16] have been applied to the search results such that results with received higher relevance are returned to the user first and so on. SFQI technique can be used to manage logical data independence of database [3]. Many of the SFQIs not well explore about design independence property of SFQIs. The main goal of SFQI is to identify schema mappings where query over the source schema can be manually translated to a new query over the target schema. Our goal is to identify mappings where SFQIs can return the same answers without changing the query, and for achieving this we proposing new SFTS engine.

We are going to develop schema free text search (SFTS) engine that will returns the same answers for different designs of the same DB content. This approach provides good logical data independence. For Example, if the DB designer changes attributes' labels of *Ename* to *EmployeeName*, our SFTS will still returns the same query answers over the old and new DB.

This paper contributes new SFTS engine which is based on concept of newly introduced Duplication and Weighted Relationship Aware Coherency Ranking (DRA-CR) with some improvements. Also explained about structure preserving, value preserving and weak value structure preserving transformations.

III. DESIGN INDEPENDENCE

A transformation T over Database D is a function that modifies D to generate a new Database as $D_1=T(D)$ [3]. This paper suggested that any SFQI operating over the transformed Database should not be dependent on the original DB [3], [5]. So, we are introducing a new Schema Free Text Search (SFTS) engine which is not be dependent on the original DB.

If any two CAs are isomorphic with corresponding leaf nodes also contains the same content then they are called as label-content isomorphic [3]. Such CAs can represent same answers over original and transformed database [3]. Hence we consider our Schema free text search is basically Design independent.

Therefore, they consider Schema free text searching to be design independent if it returns the same list of label content isomorphic answers for every query over the original and modified Databases [3]. If Database is modified SFQI may not be able to return exactly the same list of CAs for a query over the old and new Database.

A. *Preserving Content*

The answers of same query should contain the same content on both original and transformed databases. For Example, if the content of different node were different, users would consider the CAs to be different [3]. Consider a transformation that removes some nodes which are children of the same node and creates a single new child node which contains the merged content nodes. Then CAs for original and transformed DBs will have similar content [3]. The two value members equal if they are having same length and contain the same data [3]. Since users consider all CAs that a Schema Free Text Search returns for a query, so this paper defines value equality between the lists for the same query over different databases [3] with same data contents.

B. *Preserving Structure*

The nodes which do not contain any value represent structural relationships between content nodes. If any database transformation renames schema or removes schema elements or introduces new schema elements, it may change the structural relationship between these content nodes [3]. Hence, Schema Free Query Interfaces (SFQIs) cannot always deliver answers with exactly the same structure over the original and new DB [5]. Users can always translate the structure of the old answers to the structure of the new answers manually [3]. Our newly introduced Schema Free Text Search (SFTS) engine can returns structurally similar answers for both DBs, as SFTS uses some schema information to rank and/or filter CAs, we must only consider transformations that do not lose any information of the schema of the original DB [3].

C. *Weak Design Independence*

*T*he equations are an exception to the prescribed specifications of this template. You will need to determine whether or not your equation should be typed using either the Times New Roman or the Symbol font (please no other font). To create multileveled equations, it may be necessary to treat the equation as a graphic and insert it into the text after your paper is styled.

IV. SCHEMA FREE TEXT SEARCH INTERFACES

Schema Free Text Search (SFTS) engine enables users to query data with partial knowledge (or zero knowledge) of the schema that they have. If they know the full schema, they can write regular SQL. If they do not know the schema at all, they can just specify keywords as intention to search data by using the SFTS approach [2] we can search any information in database.

The basic technique we are going to use for answering queries is also called as LCA method [10], [11], returns all CAs. The LCA method is design independent. However, this approach returns all the relevant CAs. In order to improve the effectiveness of results we must filter out irrelevant CAs using filtering methods [13], [14], [12]. Once the irrelevant results are filtered we can rank the answers using a novel Duplication and Weighted Relationship Aware Coherency Ranking (DRACR) method explained in this paper.

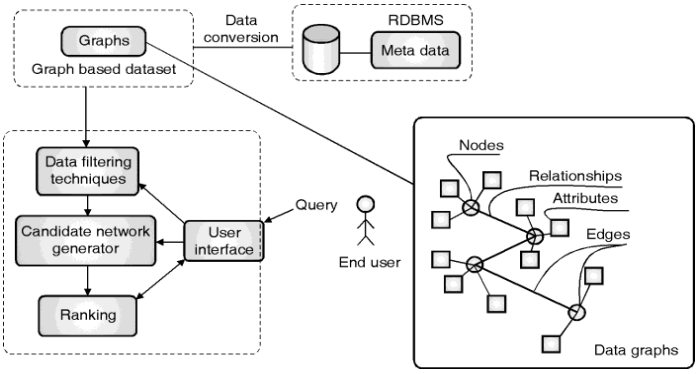


Figure 3. Architecture of Schema Free Text Search (SFTS) Engine.

Filtering Techniques

The obtained results are further processed to remove CAs which is less important. This filtering can be done using various filtering techniques. In SFTS we are going to make use of two such techniques as given below,

1.1. *Distance based Filtering Technique* [5]

The CA is considered to be irrelevant if its sub graph is relatively large. This filtering method assumes that only the closest nodes are meaningfully related with each other. This technique also removes all redundant CAs which are already covered by its sub graph.

In this technique, if CA G1 shares a leaf node with another CA G2 and the LCA of G1 is an ancestor of the LCA of G2, they filter out G1.

For Example,

Query Q2: Sachin Sarita having two different LCAs as below,
Sub Graph G1: Sachin and Sarita are in SFIT College
Leftmost data is having multiple CAs.

Sub Graph G1: LCA is node SFIT (College) covers following path,

Sachin (Students) -> IT (Branch) -> SFIT (College) -> IT (Branch) -> Sarita (Students)

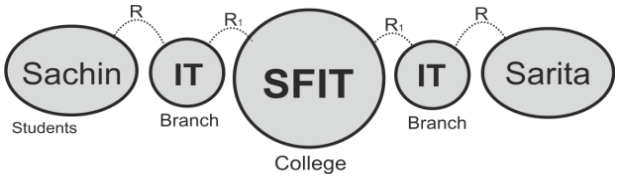


Figure 4. *G_1: Both in same collage*

Sub graph G_2: Sachin and Sarita are in IT department of same college
Leftmost data is having multiple CAs
Sub graph G_2: LCA is node IT (Branch) covers following path, *Sachin (Students) -> IT (Branch) -> Sarita (Students)*
So, G_2 is LCA of G_1 there for we can filter out G_1.
So this filtering technique helps us to filter relatively larger sub graphs.

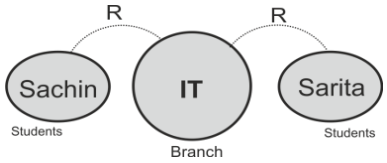


Figure 5. G_2: Both in same Department

The Distance based Filtering method is design independent method of SFQI [5].

1.2. Label based Filtering Technique [5]

All Candidate Answers (CA) having non attribute nodes with same label (name) will be removed in order to avoid redundancy. So, this filtering of CA automatically removes repetition of similar CAs. The basic idea behind is that nodes are instances of the same entity type if they have duplicate labels, and there is no interesting relationship between entities of the same type.
For example,
Q_3: Sachin and IT
Leftmost data is having multiple CAs
Sub graph a_1: LCA is node 1 University (MU) covers following path,
Sachin (Student) -> IT (Branch) -> SFIT (College) -> MU(University)-> SFIT (College) -> IT (Branch)

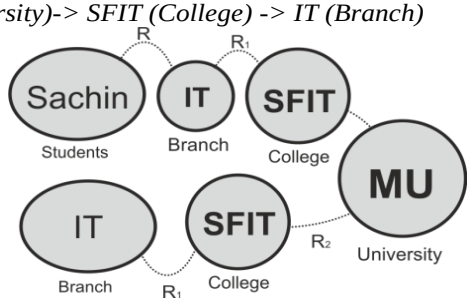
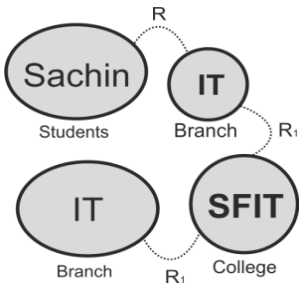


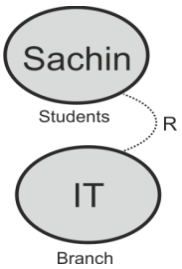
Figure 6. a_1 : Sachin and IT department are in same university (MU) and also in same college (SFIT)

Sub graph a_2: LCA is node 1 University (MU) covers following path,
Sachin (Student) -> IT (Branch) -> SFIT (College) -> IT (Branch)



a_2 : Sachin and IT department are in same college(SFIT)

Sub graph a_3: who has IT department itself as LCA node.
Sachin (Student) -> IT (Branch)



a_3 : Sachin is in IT department

These methods filter out a_1 because it contains duplicate labels like Branch and college while method also filters a_2 as it contains duplicate labels like Branch. Generally, SFQIs which use the label based heuristic are not design independent [3].

A. Ranking Technique

Ranking is important step as it decides the display of results in some meaningful order. Ranking techniques generally considers the depth and number of nodes in Candidate Answers (CA) [2]. The ranking techniques like XSearch can rank the CAs according to the number of nodes in the candidate sub graph [3].However, as it was mentioned, VS preserving transformations can add or remove nodes from a DB. Thus, they can change the number of nodes for different CAs if they are no instances of the same pattern [2]. XReal uses the depths of the LCAs and attributes nodes of candidate subtree to find and rank the CAs [7].

In order to calculate better ranking scheme we first compute entropy. Entropy is defined as predictability of expected pattern in multiple possible CAs.

The probability of value *a* in pattern *p* is *P(a).*

$$P(a) = \frac{1}{n}(count(a))$$

Where,
count(a) = Number of instances of p with value a
n = Total number of instances of p in the DB.

Entropy of a random variable indicates how predictable it is. The entropy of a pattern p having values a_1, a_2 . . . a_n with probabilities $P(a_1)$, . . . , $P(a_n)$, respectively

$$H(p) = \frac{1}{n}(\sum_{1 \le i \le 1} P(a_i) \lg(\frac{1}{P(a_i)}))$$

Normalized total correlation (**NTC**) measures the correlation of a pattern; its value for a pattern t with root-paths p_1; . . . ; p_n, $n \geq 1$ is;

$$NTC(q) = 1 - \frac{H(q)}{\sum_{1 \le i \le n} H(p_i)}$$

If size of pattern is one. CR sets value of NTC to H(q) [3]. For a given pattern t with paths p_1; . . . ; p_n, $n \geq 1$, p's *normalized set total correlation* (**NSTC**) is the NTC of its set of pattern values. In case of Schema Free Text Search technique can use TF-IDF methods for IR-style ranking [16]. Assume a transformation maps path p in D to T(p) in T(D). Since there could be different numbers of instances of p and T(p), the DF of the terms in the values of p and T(p) will be different.

Current TF-IDF methods are not design independent under transformations. Thus, it is required to redefine the DF of a term w in pattern p to be the number of distinct values of p that contain w [3]. With the redefined DF, we determine the contextual rank *Score (t,Q)* of a CA t with pattern p for query q as :

$$Score(t,Q) = \sum_{w \in q,t} \frac{1 + \ln(1 + \ln(t.f(w)))}{(1-s) + s(\mathrm{el_t} / \mathrm{avel_p})} \times qt.f(w) \times (\frac{N_p + 1}{df_p})$$ Here,

t.f (w) = Number of occurrences of w in t
qt.f (w)= Number of occurrences of w in qs
el_t = Total length of the content of t
$avel_p$= Average length of the distinct values of p
N_p = Count of distinct values of p
df_p = Number of distinct values of p that contain w.
s = Constant
(IR community has found that 0.2 is the best value for s [3])

Now they combine above computed *score* with NSTC on a sliding scale as given below,

$$rank(t,Q) = \propto NSTC(p) + (1 - \propto) \times Score(t,Q)$$

Where p is t's pattern and α is a constant that determines the relative weight given to structural versus contextual information [5]. They determine the value of α empirically [5].

2.1. Duplication Aware Coherency Ranking Technique (DA-CR) [3]

The ranking which can be used for Schema free Text Search can be DA-CR (Duplicate Awareness Coherency Ranking) which can be applies in similar way as IR Style ranking [16]. To solve some problems involved in ranking scheme such as duplications of similar pattern, we group patterns with equal values for NSTC and the same set of paths before query time. After finding the CAs at query time, now they find the equivalence class of the pattern of each CA and consider only CA(s) with the smallest patterns in each class. If there is more than one such pattern, this technique will break the ties arbitrarily. The new approach is called as Duplicate Aware CR (DA-CR) [5].

$$F_0 = DACR(t,Q) = newScore(t,Q)$$

Where,
newScore(t,Q) = NSTC Score after avoiding all duplicates in results

This ranking scheme DACR has two problems [5]
1. User has to scan through all duplicate patterns in the list of candidate answers.
2. In IR style ranking may ranks larger patterns in the same equivalence class higher, as they have more paths and therefore may contain more query terms.
3. Importance of relationship is not considered while ranking.
4. In case of duplicate, No other factor considered while selecting candidate answer.

2.2. Duplication and Relationship Aware Coherency Ranking Technique (DAR-CR)

To solve problem of DACR technique given above, we can use a level of importance for specific relationships by measuring the occurrence of such relationship between various entities of candidate answer (CA).

The *inverse frequency of association (relevance)* will give us the importance of results found by any searching keywords by given Query. The CAs will then organize by above rank (t, Q) and then, it also considers the inverse frequency of relationships included by selected pattern. For each pattern found by keyword search we will compute the inverse relevance weight,

$$IRW(Rv) = \sum_{i=0}^{n} \frac{p}{T_P}$$

Here,
Rv = Patterns Searched by keywords
p = Number of occurrences of pattern p
T_p = Total Number of occurrences of patterns in database

Now for each pattern p we will compute the *weighted average of inverse relationship* [1] in order to identify its relationship weightage or importance,

$$F_1 = RWF(t,Q) = \sum_{i=0}^{n} \frac{IRW(Rv)}{N_p}$$

Here,
p = Number of occurrences of pattern p
N_p = Number of relation occurrences of w in qs

The *inverse frequency of relations* will give us the importance of any entity searched by given Query. The CAs will then organize by above rank (t, Q) and then, it also considers the inverse frequency of entities included by selected pattern. For each relation in pattern we will compute the inverse relationship weight,

$$IRW(Ent) = \sum_{i=0}^{n} \frac{e}{T_e}$$

Here,
Ent= Entities occurred in t of pattern p
e = Number of occurrences of e in t
T_e = Number of occurrences of entities in qs
N_f = Total length of the relationship and involved entities

Now for each pattern p we will compute the *weighted average of inverse entities* in order to identify its relationship weightage or importance,

$$F_2 = RWF(t,Q) = \sum_{i=0}^{n} \frac{IRW(Ent)}{N_r}$$

Here,
e = Number of occurrences of entities in t
N_e = Number of entities occurrences of w in qs

The *inverse frequency of relationship* will give us the importance of any relationship searched by given Query. The CAs will then organize by above rank (t, Q) and then, it also considers the inverse frequency of relationships included by selected pattern. For each relation in pattern we will compute the inverse relationship weight,

$$IRW(\mathrm{Re}l) = \sum_{i=0}^{n} \frac{r}{T_r}$$

Here,
Rel = Relations occurred in t of pattern p
r = Number of occurrences of *Rel* in pattern p
T_r = Total Number of occurrences of *Rel* in database

Now for each pattern p we will compute the *weighted average of inverse relationship* [1] in order to identify its relationship weightage or importance,

$$F_3 = RWF(t,Q) = \sum_{i=0}^{n} \frac{IRW(\mathrm{Re}l)}{N_r}$$

Here,
r = Number of occurrences of *Rel* in pattern p
N_r = Number of relation occurrences of w in qs

So After applying all factors on computed result the new ranking formula considers multiple parameters while ranking results. As given below,

$$Rank(t,Q) = F_0 * K_0 + F_1 * K_1 + F_2 * K_2 + F_3 * K_3$$

Where,
K_0 = weightage of old distance based ranking scheme
K_1 = weightage of relevance of results
K_2 = weightage of entities included in results
K_3 = weightage of relationships included in results

The practical implementation of system shows for above constants $K_0 = 0.4$; $K_1 = 0.3$; $K_0 = 0.1$; $K_0 = 0.1$ will be the best values.

EXPERIMENTS

In this paper, we studied and tried to solve the problem of Schema Free Text Search techniques. This problem also includes the efficient data search and ranking results in meaningful order in the presence of keyword search ambiguities. We have implemented SFTS using some research papers based on SFQI [2] and evaluated the SFTS system on two aspects:

1. Search Quality: Search Quality is evaluated using both a standard RDBMS benchmark and a heterogeneous data collection using XML data set.
2. Search Performance: We measure the overhead caused by evaluating schema-free query versus the schema-aware query. This is done by recording time and relativeness measures of query results.

TABLE I.
RESULT OBTAINED BY OLD DISTANCE BASED RANKING SCHEME

Keyword No	DRACR			CR		
	N	N	D	E	D	E
1	5,5	2	375	5,5	2	250
2	3,5	3	750	3,5	3	560
3	5,5	2	*375*	5,5	2	*250*
4	2,2	3	234	2,2	3	328
5	3,5	3	375	3,5	3	234
6	3,5	4	328	3,5	4	50
7	3,5	2	450	3,5	2	343

WHERE,
D = Distance
(Minimum distance from all results, Maximum distance from all results)
N = Number of Results produced by Query Keyword
E = Execution time required for query to obtain results (in milliseconds)

The results shows that the DRACR newly introduced scheme is more effective when it comes to ranking quality as it displays more relevant results at the start by placing their occurrence in database as a primary weight and then distance of their relation.
As we are also searching for all possible sub tree and super tree of all CAs therefore the time required is more than the old scheme.

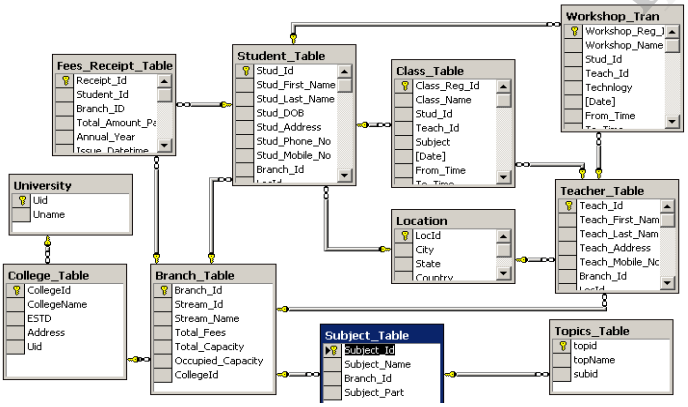


Figure 7. Modified Sample University Databases.

Modified Sample University data sets shows that the pattern P1 having comparatively bigger sub tree than pattern P2, But still pattern P1 is ranked before path P2 as the importance of sub graph P1 is much more than sub graph P2.
The possibility of Mahesh and Aayush know each other by pattern P1 (i.e. Using departments of college VIT) is more likely than pattern P2, although sub graph for P1 is bigger than sub graph for pattern P2.
Example, if SFTS accepts Keywords as Mahesh Aayush
Sub graph P_1:
Mahesh(Students) → IT(Branch) → VIT(College) → CMPN(Branch) → Aayush(Students)
Sub graph P_2:
Mahesh(Students) → Mumbai(Location) → Aayush(Students)

Effectiveness
The quality of a search using SFTS system was measured in terms of accuracy and completeness using standard precision and recall metrics, where the correct results are the answers returned by the corresponding schema-aware Query. Precision measures accuracy, it refers to the fraction of results in the query answers that are correct. While, Recall measures completeness, indicating the fraction of all correct results actually retrieved in query answer. The F factor records (2 * Precision * Recall)/(Precision + Recall).

TABLE II.
PRECISION, RECALL AND F-MEASURE COMPARISON WITH OTHER APPROACHES

Technique	Precision	Recall	F Measure
DRACR	0.98	0.99	0.69
CR	0.98	0.99	0.99
MLCA [1]	0.71	0.68	0.69
XQuery [1]	0.82	0.88	0.84

As discussed above, Always higher design independence cannot produce effectiveness, i.e., better recall and precision. However, previous work has already shown that CR delivers better ranking quality than the old methods discussed in this paper [2].

ACKNOWLEDGMENT

This work is supported Vidyalankar institute of Technology (VIT), Mumbai and greatly supported and encouraged by my guides.

REFERENCES

[1] M. Mali, P. Vanwari and V. Dalal, "Towards Schema Free Text Search : A Survey", Published in "TECHNOCON-2013" ISSN 0974-0678 in National Conference on "INNOVATIVE TRENDS FOR TECHNOLOGY DEVELOPMENTS" at Datta Meghe Institute of Engineering Technology and Research, Wardha, Maharashtra State, India held on 17th Dec.2013.
[2] M. Rammohanrao, M. Brahmaiah, E. Ramesh and V. Rajesh, "Schema and Design Free Keyword Search Interfaces for XML Databases", International Journal of Engineering Research and Development, e-ISSN: 2278-067X, p-ISSN: 2278-800X, www.ijerd.com,Volume 5, Issue 9, January 2013
[3] A. Termehchy, M. Winslett, Y. Chopathumwan and A. Gibbons, "Design Independent Query Interfaces", IEEE transaction on knowledge and data engineering, vol. 24, no. 10, October 2012
[4] L. Han, T. Finin, A. Joshi, "Schema-Free Structured Querying of DBpedia Data" in CIKM, November 2012.
[5] A. Termehchy, M. Winslett and Y. Chopathumwan, "How Schema Independent Are Schema Free Query Interfaces?" ICDE '11 Proceedings of IEEE 27th International Conference on Data Engineering, October 2011

[6] A. Termehchy and M. Winslett, "Effective, Design-Independent XML Keyword Search" in CIKM, 2009.
[7] A. Termehchy and M. Winslett, "Using Structural Information in XML Keyword Search Effectively," TODS, vol. 36, no. 1, 2011.
[8] Y. Luo, X. Lin, W. Wang, and X. Zhou, "SPARK: Top-k Keyword Query in Relational Databases," in SIGMOD, November 2007.
[9] A. Termehchy and M. Winslett, "Keyword Search for Data-Centric XML Collections with Long Text Fields," in EDBT, 2010.
[10] G. Li, J. Feng, J. Wang, and L. Zhou, "Effective Keyword Search for Valuable LCAs over XML Documents," in CIKM, 2007.
[11] Y. Xu and Y. Papakonstantinou, "Efficient Keyword Search for Smallest LCAs in XML Databases," in SIGMOD, 2005.
[12] Y. Li, C. Yu, and H. V. Jagadish, "Schema-Free XQuery," in VLDB, 2004.
[13] L. Guo, F. Shao, C. Botev, and J. Shanmugasundaram, "XRANK: Ranked Keyword Search over XML Documents," in SIGMOD, 2003.
[14] S. Cohen, J. Mamou, Y. Kanza, and Y. Sagiv, "XSearch: A Semantic Search Engine for XML," in VLDB, 2003.
[15] V. Hristidis and Y. Papakonstantinou, "DISCOVER: Keyword Search in Relational Databases" Proceedings of the 28th VLDB Conference, Hong Kong, China, 2002
[16] V. Hristidis, L. Gravano, Y. Papakonstantinou, "Efficient IR-Style Keyword Search over Relational Databases" Proceedings of the 29th VLDB Conference, Berlin, Germany, 2009
[17] F. Liu, C. T. Yu, W. Meng, and A. Chowdhury, "Effective keyword search in relational databases." in SIGMOD, 2006, pp. 563–574.