Supporting Ranking Queries for Search-As-You-Type in Databases using WIP

Kaushik G. Vaghani
PG-Scholar
Computer Science & Engineering Department
Parul Institute of Technology,GTU
Vadodara,Gujarat,India

Pratik A. Patel
Assistant Professor
Computer Science & Engineering Department
Parul Institute of Technology,GTU
Vadodara,Gujarat,India

ABSTRACT
Most of the search engines in Internet have simplified keyword-based search concept. The relational database management systems do not allow keyword-based search while they offers powerful query languages. Existing search systems that are based on keyword in relational database require users to submit a complete query to compute answers. Often users have limited knowledge about the data, and have to use a try and see method to modify queries and find the answers.

Search-as-you-type is a search system that allows the user to submit the prefix of the keyword and system will compute the answer as user type keyword character by character for data stored in a relational database management systems. A main challenge is how to influence existing database functionalities to achieve high-performance in searching speed and how to support ranking queries that provide the most frequently searched results at top position in computed result. The proposed technique shows how to use weights of records stored as an auxiliary tables to increase search performance. I have proposed solutions for single-keyword queries and develop a new technique, weighted index-based technique called WIP-based searching that supports ranking queries for searching records based on prefix of keywords by using additional weight table stored as auxiliary table. My main approach is to push the weight constraints into index-based techniques. By this new WIP-based technique of search-as-you-type, query result gives the records based upon frequency of usage.

Keywords
Search-As-You-Type, Type-ahead Search, Keyword Search, Databases, SQL, WIP, Ranking Queries

Abbreviations
WIPTables technique – Weighted, Inverted-Index, Prefix Table technique

1. INTRODUCTION
It has become extremely popular to provide users with flexible ways to search information over databases as simple as keyword search like Google search. Now a day, the relational databases are broadly used by applications from different areas and different search paradigms needed by different users. Knowledgeable users, such as database administrators, need a search paradigm that can provide them accurate and fully functional accessing abilities. In contrast, most inexperienced users, as casual Internet users expect to search databases as easily as possible. In addition, some users, such as systems analysts, call for new paradigms for search that influence usability and functionality.

A search-as-you-type system calculates the responses on-the-fly as a user types in a keyword query character by character. I study how to support search-as-you-type on data residing in a relational DBMS using database language, SQL.

Most information systems currently improve user search experiences by providing immediate feedback as users create search queries. Many search engines and online search forms support auto completion, which shows suggested queries or even answers “on the fly” as a user types in a keyword query character by character. For example, In Web search interface at Netflix,1 that permits a user to search for movie details. If a user types partial keyword “mad,” the search interface shows movies with a title matching this keyword as a prefix, such as “Madagascar” and “Mad Men: Season 1”. The instantaneous answer helps the user not only in formulating the query, but also in understanding the underlying data[5]. This type of search is generally called search-as- you-type or type-ahead search.

2. PROBLEM STATEMENT
In order to provide excellent keyword based search-as-you-type speed at a minimal time, all information systems needs diagnostic procedures that are fast, efficient, and accurate. In addition, the procedures should not be requiring additional overhead in relational database management systems.

Because most search systems keep their information in a backend relational Database management system, a question arises obviously: how to bear search-as-you-type on the data stored in a DBMS? Some databases already support prefix search such as Oracle and SQL server, and we might use this feature to do search-as-you-type. But, all databases do not provide this quality. Because of this cause, we require new methods that can be used in all databases.

One approach is to develop a separate application layer on the database to construct indexes, and implement algorithms for answering queries. Even this concept has the advantage of achieving a high performance; its main drawback is it is not work for ranking queries[5]. A new technique, Search-as-you-type with weight attribute can support ranking queries for searching based on prefix of keyword in database systems. We added weight constraint in previously searched records by maintaining additional auxiliary table to find frequently searched records accurately in minimum time. By this technique, We can reduce the programming efforts to support search-as-you-type that supports ranking queries. In addition, the solution that is developed for one database using this technique is portable to other databases supporting the same standard.

3. RELATED WORK

Search based on keyword is a well studied problem in the world of Internet search engines and text based documents. There have been many studies on keyword search in relational databases. Most of them employed tree based methods[10,12,13]. Other methods[11,14] generated answers composed of relevant tuples by generating and extending a candidate network following the primary-foreign-key relationship. The main objective to develop new techniques is all about turning intentions into actions in the blink of an eye. In online retail, having a quick and user-friendly website it helps to increase sales and conversion rates on merchant websites[6]. Search-as-you-type is a user-friendly feature which can reduce the efforts of users to process their queries by returning the results immediately as users type keyword character by character.

Keyword Search over relational databases uses DBXplorer[1], DISCOVER, BANKS[4] system that support keyword search on relational databases.

A main requirement of search-as-you-type on huge amounts of relational data is the need of a high interactive speed for searching. Every keystroke from a user can invoke a query to the system that needs to calculate the answers within the milliseconds. Many few techniques have been implemented for Search-as-you-type for the data stored in the relational database systems. Recently type-ahead search on relational databases uses TASTIER approach[2] and another techniques are by using existing functionality of query engine of database systems as much as possible and requires additional index structures stored as auxiliary tables[5].

4. SEARCH-AS-YOU-TYPE FOR NON RANKING QUERIES

4.1 No-Index Methods

One simple way to support search-as-you-type is to execute a SQL query that scan every record and checks whether the record is an answer of the query. There are two methods to do so :

a. *Calling User-Defined Functions (UDFs)*

We can add functions into databases to verify whether a record contains the query keyword.

b. Using the LIKE predicate

Databases provide a LIKE predicate that allow users to achieve keyword matching. We can use LIKE predicate to ensure whether a record contains the query keyword.

This two no-index methods do not require additional space in database but they may not use because they need to scan all records in database table.

4.2 Index-Based Method

This method uses additional index structure stored as auxiliary tables to facilitate prefix search. This method can be used in all databases. A description of the additional auxiliary tables is as follows:

Inverted-index table: Given a table T with assign unique ids to the keywords in table T, following their alphabetical order. Inverted-index table I_T with records in the form <kid, rid>, where kid is the id of the keyword and rid is the id of a record that contains the keyword.

Prefix table: Given a table T, for all prefixes of keywords in the table, a prefix table P_T with records in the form <p,lkid,ukid>, where p is a prefix of a keyword, lkid is the smallest id of those keywords in the table T having p as a prefix, and ukid is the largest id of those keywords having p as prefix.

The example of these tables based on records in sample dblp Table 1 is shown in below Table 2.

Table 1 dblp: A Sample Publication Table (about "Privacy")

ID	Title	Authors	Booktitle	Year
r1	K-Automorphism: A General Framework for Privacy Preserving Network Publication	Lei Zou, Lei Chen, M. Tamer Ozsu	PVLDB	2009
r2	Privacy-Preserving Singular Value Decompositon	Shuguo Han, Wee Keong Ng, Philip S. Yu	ICDE	2009
r3	Privacy Preservation of Aggregates in Hidden Databases: Why and How?	Arjun Dasgupta, Nan Zhang, Gautam Das, Surajit Chaudhuri	SIGMOD	2009
r4	Privacy-Preserving Indexing of Documents on the Network	Mayank Bawa, Roberto J. Bayardo, Rakesh Agrawal, Jaideep Vaidya	VLDBJ	2009
r5	On Anti-Corruption Privacy Preserving Publication	Yufei Tao, Xiaokui Xiao, Jiexing Li, Donghui Zhang	ICDE	2008
r6	Preservation of Proximity Privacy in Publishing Numerical Sensitive Data	Jiexing Li, Yufei Tao, Xiaokui Xiao	SIGMOD	2008
r7	Hiding in the Crowd: Privacy Preservation on Evolving Streams through Correlation Tracking	Feifei Li, Jimeng Sun, Spiros Papadimitriou, George A. Mihaila, Ioana Stanoi	ICDE	2007
r8	The Boundary Between Privacy and Utility in Data Publishing	Vibhor Rastogi, Sungho Hong, Dan Suciu	VLDB	2007
r9	Privacy Protection in Personalized Search	Xuehua Shen, Bin Tan, ChengXiang Zhai	SIGIR	2007
r10	Privacy in Database Publishing	Alin Deutsch, Yannis Papakonstantinou	ICDT	2005

Table 2: The inverted index table and prefix table

(a) Keywords

kid	keyword
k1	icde
k2	icdt
k3	preserving
k4	privacy
k5	publishing
k6	pvldb
k7	sigir
k8	sigmod
k9	vldb
k10	vldbj
…	…

(b) Inverted-index Table

kid	rid
K2	r10
K5	r6
K5	r8
K5	r10
K6	r1
K7	r9
K8	r3
k8	r6
k9	r8
k10	r4
…	…

(c) Prefix Table

prefix	lkid	ukid
ic	k1	k2
p	k3	k6
pr	k3	k4
pri	k4	k4
pu	k5	k5
pv	k6	k6
pvl	k6	k6
sig	k7	k8
v	k9	k10
vl	k9	k10
…	…	…

For example, The inverted-index table has a tuple <k8,r3> since keyword k8 ("sigmod") is in record r3. The prefix table has a tuple <"sig",k7,k8> since keyword k7 ("sigir") is the minimal id of keywords with a prefix "sig", and keyword k8 ("sigmod") is the maximal id of the keywords with a prefix "sig". The ids of keywords with prefix "sig" must be in the range [k7,k8].

Given a partial keyword w, we first get its keyword range [lkid,ukid] using the prefix table P_T, and then find the records that have a keyword in the range through the inverted-index table I_T as shown in Figure 1. We use the following SQL to answer the prefix-search query w[5]:

SELECT *T*.* FROM P_T, I_T, *T*
WHERE P_T.prefix = "*w*" AND
$P_T.ukid >= I_T.kid$ AND $P_T.lkid <= I_T.kid$ AND
$I_T.rid = T.rid$

Below fig 1 shows that how index based method works to find the records by using these additional tables.

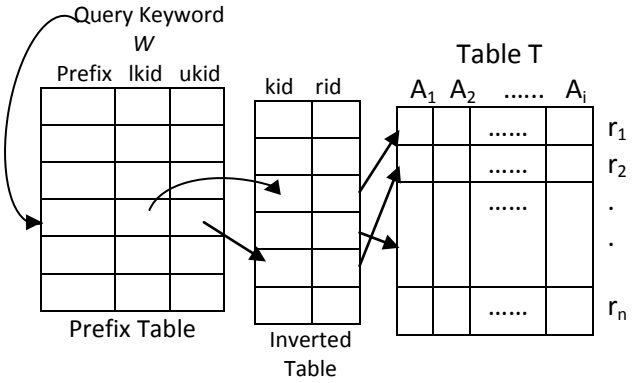

Fig 1: Using inverted-index table and prefix table to support search-as-you-type[5]

This method does not requires to scan whole records in databases. So execution time for query is very small as compared to No-index based methods but this method does not support ranking queries.

5. WIP-BASED SEARCH-AS-YOU-TYPE

To support prefix matching, we showed solutions that use auxiliary tables as index structures and SQL queries to support search-as-you-type. There are several open problems with these solutions to support search-as-you-type using SQL. However these techniques do not support ranking queries efficiently.

Ranking queries means when user enters keyword character by character, system computes the most relevant answers first based on prefix entered by the user. The records that are most likely searched by users for particular prefix of keyword have higher ranks. The top ranking records are then shown to the users.

For support the ranking queries for search-as-you-type from relational database I proposed a new technique that add one additional table, Weight table W_T that contains information in form <rid, weight>, where rid is record id in table T and weight is the overall rank of the record in table T.

Now when user enter a partial keyword *w* system first get its keyword range [lkid,ukid] using the prefix table P_T, and then find the records that have a keyword in the range through the inverted-index table I_T. It then looks up the weight table W_T to get rank. The system then returns top ranked records first followed by records that have lower ranks. User then selects the appropriate record from listed records and based on that weight table is refine to modify the rank of selected record. We will use the following SQL to answer the prefix-search query *w*:

SELECT *T*.* FROM P_T, W_T, I_T, *T*
WHERE P_T.prefix = "*w*" AND
$P_T.ukid >= I_T.kid$ AND $P_T.lkid <= I_T.kid$ AND
$I_T.rid = T.rid$ AND $T.rid=W_T.rid$
ORDER BY $W_T.weight$ *DESC*

5.1 WORKFLOW OF WIP APPROACH

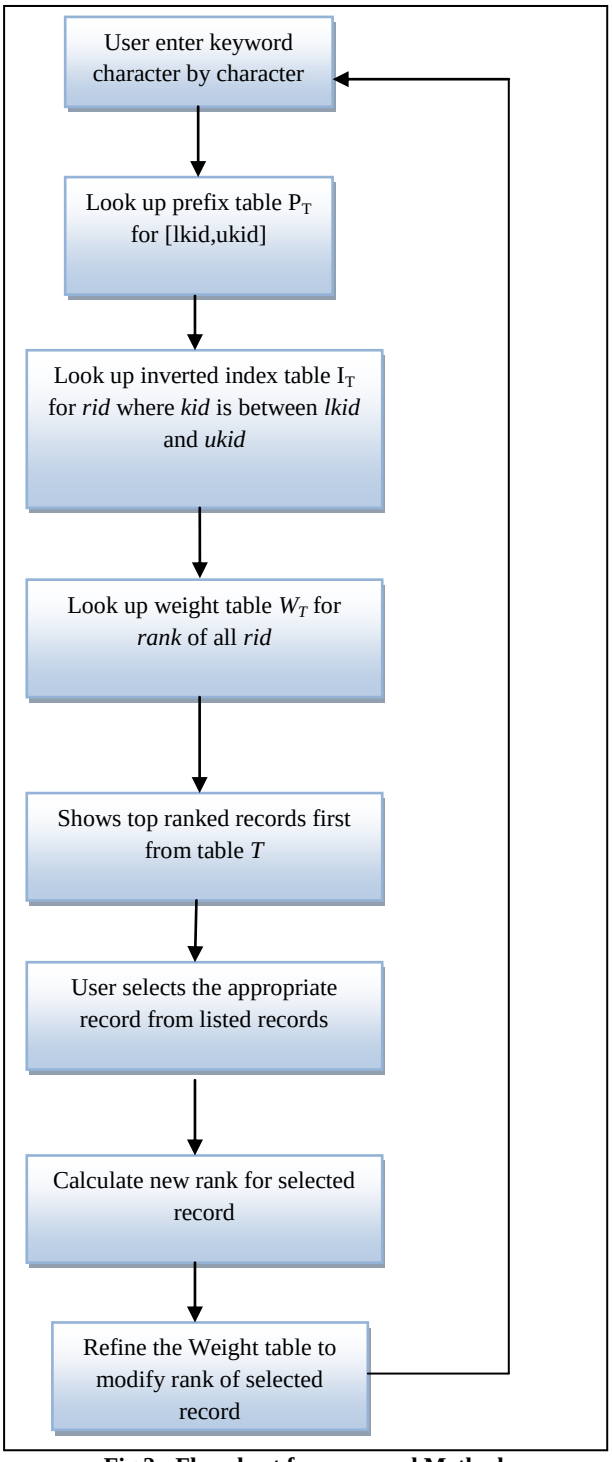

Fig 2: Flowchart for proposed Method

A flowchart shown in above figure 2 depicts step by step procedure for WIP approach of search-as-you-type that is enhanced technique of index-based method of searching over databases. First user enters keyword in to the search field of any system character by character. System will then fire the query on each key stroke of the user into backend database and look up the prefix table P_T for lower and upper rang of *kid* and then find all *rid* for that *kid* range from Inverted-index table I_T. Now we have all the *rid* of those records which contains keywords with prefix that is entered by user. To support ranking queries we have added one additional table Weight table W_T in which all records have some particular rank. By using this rank system will show top ranked records from result records to the users at higher position followed by lower ranked records. User will select appropriate record from the given searched records and based on that weight table is again refined to increment the rank of that particular record by one.

5.2 ANALYSE AN WIP APPROACH OF SEARCH-AS-YOU-TYPE BY EXAMPLE

To understand how the proposed WIP based technique that supports ranking queries for search-as-you-type is design and implemented, we gone through following example. Suppose here we again consider publication table of dblp database that is shown in Table 1.

Now when user enter a partial keyword ‘Sig’ system first get its keyword range [lkid,ukid] using the prefix table P_{dblp}, and then find the records that have a keyword in the range through the inverted-index table I_{dblp}. It then looks up the weight table W_{dblp} to get rank. The system then returns top weighted records first followed by records that have lower weight. User then selects the appropriate record from listed records and based on that weight table is refine to modify the weight of selected record and increment its weight by one. The schematic diagram for this example is shown in below figure 3. Also we have shown the query that is fired into backend database to find top ranked records with the use of these auxiliary tables.

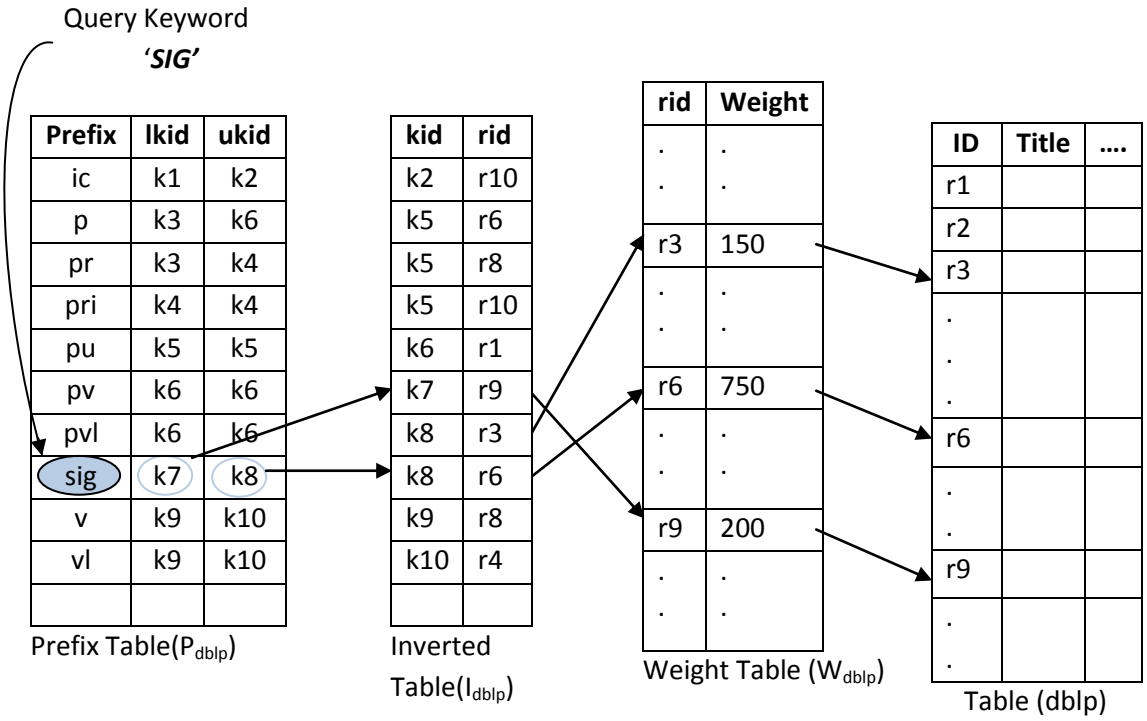

Fig 3: WIP Approach - Using weight table to support ranking queries for search-as-you-type

We will use the following SQL to answer the prefix-search query 'Sig':

SELECT *dblp*.* FROM P_{dblp}, $W_{dblp,}$ I_{dblp}, *dblp*
WHERE P_{dblp}.prefix = "*sig*" AND
P_{dblp}.*ukid* >= I_{dblp}.*kid* AND P_{dblp}.*lkid* <= I_{dblp}.*kid* AND
I_{dblp}.*rid* = W_{dblp}.*rid AND* W_{dblp}.*rid* = *dblp.rid ORDER BY* W_{dblp}.*weight* DESC.

After execution of the above query system will returns records in order of r6, r9 and r3. This order is based on their weight that is r6(750), r9(200) and r3(150). So in this way user will get higher weighted records first in the list.

6. EXPERIMENTAL STUDY

We have implemented existing and proposed methods on one real dataset zipcode: It includes 42,741 records of zip codes. It is database of zipcodes of countries of USA.

We implemented the technique in JAVA that demonstrates the use of my proposed technique for search-as-you-type that uses weight constraint to support ranking queries. We used MYSQL as database in the experiments. We will compare different techniques in below section 6.1. The sample table of dataset zipcode is shown in below figure 4.

6.1 RESULT ANALYSIS

The result analysis of proposed technique is shown into figure 5 as a graph. We have taken the results for all three techniques that is LIKE, Index-based and WIP for different prefix of keywords of dataset ZIPCODE.

rid	ZIP	City	State	Country	Type
1	00501	HOLTSVILLE	NY	SUFFOLK	UNIQUE
2	00544	HOLTSVILLE	NY	SUFFOLK	UNIQUE
3	00601	ADJUNTAS	PR	ADJUNTAS	STANDARD
4	00602	AGUADA	PR	AGUADA	STANDARD
5	00603	AGUADILLA	PR	AGUADILLA	STANDARD
6	00604	AGUADILLA	PR	AGUADILLA	STANDARD
7	00605	AGUADILLA	PR	AGUADILLA	STANDARD
8	00606	MARICAO	PR	MARICAO	STANDARD
9	00610	ANASCO	PR	ANASCO	STANDARD
10	00611	ANGELES	PR	UTUADO	PO BOX ONLY

Fig 4: Sample zipcodes table of zipcode dataset

The comparative result analysis for all three techniques is clearly shown in to below graph. As we can show from graph that LIKE predicate based technique takes more time than Index based technique (IPTable) and WIP based technique because it need to scan all records into database table to find the answers. The index-based technique and WIP technique almost takes same time to execute the query. In some cases WIP based technique takes some milliseconds more time then Index-based technique but WIP technique gives search result that is based on records frequency of search as we discussed in above section.

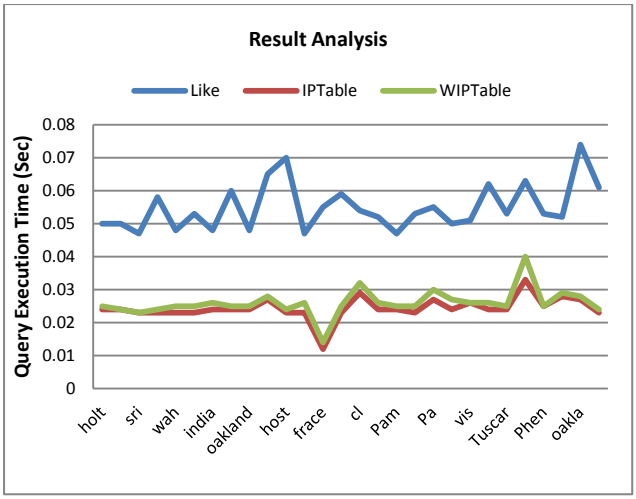

Fig 5: Query execution time for different techniques

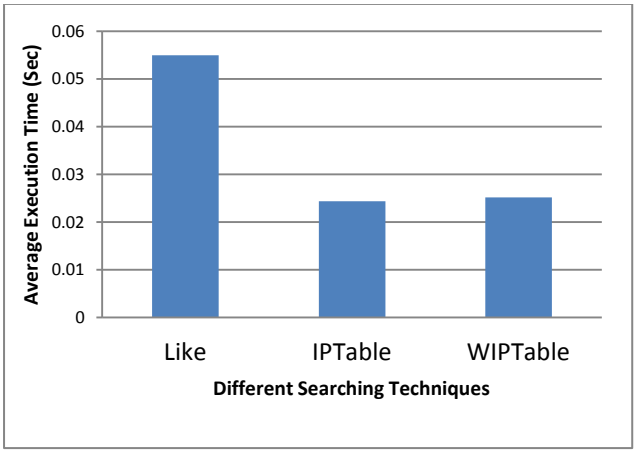

Fig 6: Average Query execution time for different techniques

The above figure 6 shows average time to execute queries by different searching techniques. The difference of query execution time between IPTable and WIP-based technique is nearly one to two milliseconds that can be neglected but WIP technique gives efficient results and supports ranking queries that index-based technique does not support.

7. CONCUSION AND FUTURE WORK

Various techniques have been studied to support search across the database and each technique has its own advantage and limitation. Experiment on large, real data set show that the existing techniques enable DBMS systems on a commodity computer to support search-as-you-type on tables with millions of records have many open problems like supporting ranking queries. We have studied the problem of supporting ranking queries using SQL to support search-as-you-type for data stored in relational databases. We used additional weight table as auxiliary table in existing technique that allows the user to get answers which are mostly searched when user type keyword character by character. A proposed WIP based Search-as-You-Type approach provides instant answer of user's queries on each key stroke of user based on frequency of search.

In future work, an approach can be done to develop this technique for multiple tables and to reduce some more time in execute queries.

8. ACKNOWLEDGMENTS

With the cooperation of my guide, I am highly indebted to Asst. Prof. Pratik Patel, for his valuable guidance and supervision regarding this work as well as for providing necessary information regarding materials.

9. REFERENCES

[1] S. Agrawal, S. Chaudhuri, and G. Das, "DBXplorer: A System for Keyword-Based Search over Relational Data Bases," Proc. 18th Int'l Conf. Data Eng. (ICDE '02), pp. 5-16, 2002.

[2] G. Li, S. Ji, C. Li, and J. Feng, "Efficient Type-Ahead Search on Relational Data: A Tastier Approach," Proc. 35th ACM SIGMOD Int'l Conf. Management of Data (SIGMOD '09), pp. 695-706, 2009.

[3] Hao Wu. "Search-As-You-Type in Forms: Leveraging the Usability and the Functionality of Search Paradigm in Relational Databases." VLDB (PhD Workshop), 36-41. 2010. Accessed February 7, 2012.

[4] G. Bhalotia, A. Hulgeri, C. Nakhe, S. Chakrabarti, and S. Sudarshan. Keyword searching and browsing in databases using banks. In ICDE, pages 431–440, 2002.

[5] Guoliang Li, Jianhua Feng, Chen Li, "Supporting Search-As-You-Type Using SQL in Databases," IEEE Transactions on Knowledge and Data Engineering, vol. 25, no. 2, pp. 461-475, Feb. 2013

[6] L. Qin, J.X. Yu, and L. Chang, "Keyword Search in Data Bases: The Power of Rdbms," Proc. 35th ACM SIGMOD Int'l Conf. Management of Data (SIGMOD '09), pp. 681-694, 2009.

[7] Daniel Suelmann, "Keyword-based Search in a Relational Database" Bachelor's Thesis, Department of Information Science Faculty of Arts University of Groningen August 2009

[8] F. Liu, C.T. Yu, W. Meng, and A. Chowdhury, "Effective Keyword Search in Relational Data Bases," Proc. ACM SIGMOD Int'l Conf. Management of Data (SIGMOD '06), pp. 563-574, 2006.

[9] H. Bast and I. Weber, "Type Less, Find More: Fast Autocompletion Search with a Succinct Index," Proc. 29th Ann. Int'l ACM SIGIR Conf. Research and Development in Information Retrieval (SIGIR '06), pp. 364-371, 2006.

[10] M. Hadjieleftheriou, A. Chandel, N. Koudas, and D. Srivastava, "Fast Indexes and Algorithms for Set Similarity Selection Queries," Proc. IEEE 24th Int'l Conf. Data Eng. (ICDE '08), pp. 267-276, 2008.

[11] C. Li, J. Lu, and Y. Lu, "Efficient Merging and Filtering Algorithms for Approximate String Searches," Proc. IEEE 24th Int'l Conf. Data Eng. (ICDE '08), pp. 257-266, 2008.

10. AUTHOR'S PROFILE

Kaushik Vaghani received the Bachelor of Engineering degree in Computer Engineering from Sarvajanik College Of Engineering & Technology under Veer Narmad South Gujarat University in 2011, Gujarat. During 2011-2012, he worked for Triz Innovation Pvt Ltd. as a software developer. He is pursuing Master of Engineering degree in Computer Science & Engineering from Parul Institute of Technology under Gujarat Technological University.