NOVEL FRAMEWORK SUPPORTING SEARCH IN DATABASES BASED ON SQL

K Nagarjuna[1], A Ramaswamy Reddy[2]

[1] *M.Tech Scholar (CSE), Nalanda Group of Institutions, Siddharth Nagar, Guntur, A.P, (India)*
[2] *Associate Professor & HOD (CSE), Nalanda Group of Institutions, Guntur, A.P, (India)*

ABSTRACT

The search as you type system computes answers on the fly as the user type in the keyword query character by character. We study in the support search as you type on the data residing in relational DBMS. We completely focus on how to the support this type of search using native database language. The main scope is how to force existing database functionalities to meet the high performance requirement to the achieved the interactive speeds. We study how to use the auxiliary indexes deposited as in the tables to the surge search the representation. We present the solution for together keyword queries and the multi keywords queries and the developed novel techniques for fuzzy search using SQL by allowing mismatches among query keywords and answers. Present techniques to answer first-N queries and discuss how to the support updates efficiently. The experiments on big real data sets show that our techniques enable DBMS systems on the commodity computer to support search-as-you-type on tables with millions of the records.

Index Terms Are:-Search As You Type and Databases SQL Fuzzy Search

I. INTRODUCTION

More information systems currently improved the user search experiences by providing instant feedback asthe users verbalize search query. Frequently search engine, online search forms support sedan completion which are shows recommended queries or even answers on fly as the user types in the keyword query character by the character. Since instance consider Web search interface at the Netflix which tolerates the user to search for the movie information. Whether the user types in the partial query mad system shows movies with the title matching this keyword as the prefix such as Madagascar and Mad Men The instant feedback helps the user not only in the formulating the query then also in understanding underlying data. This is type of the search generally called search as you type or type onward search.Thereforeadditional search systems store their information in the Backend interpersonal DBMS question arises naturally how to the support search as you type on data residing in the DBMS. Some databases such as the Oracle and SQL server already support prefix search and we could use this feature tosearch as you type. However entire databases provide this feature. We study new method that can be used in all databases. Once the methodology is to the developed the separate application layer on to the database to construct indexes and the implement algorithm is for the answering queries. However this approach has the advantage of the achieving the high performance it is main drawbacks are duplicating data and the indexes resulting in the additional hardware cost. The alternative methodologies are to the use database extenders such as the DB2 Extenders Informix Data Blades Microsoft SQL Servers Common Language. Runtime (CLR) integration and Oracle Cartridges which isalloweddevelopers to the implement novel functionalities to DBMS. In this type of approach isn't feasible for databases that don'tprovide such extender interface such asMysql database. Then itis needs to utilize registered interfaces giving bydatabasevendor

solution for oncedatabase might portable tothe others. The extender based solutionsare especially those implemented in the C or the C++ could cause the serious reliabilityand the security problems are to the database engine.We study how to a support search as you typeon the DBMS systems using native query language (SQL) Structure queries language. We want to the use SQL to find responses to searchquery as user types are in the keywords character by the character.Our aim is toutilize built in query engine of databasesystem as moreas the possible. We can decreaseprogramming efforts to the support search as you type. Solutions are developed on one the database usingthe standard SQL techniques areportable to another databasewhich is supportsimilar standard. Correspondingly the observations are also completed by Gravanoetand Jestes which is use SQL to the support similarity join indatabases.

II. PRELIMINARIES

2.1 Problem Formulation

Let T be theinteractive table with attributes $A1, A2, \ldots, A$.let$R = \{r1, r2, \ldots, r_n\}$be collection of the records in $T$and$ri[Aj]$denote the content of the record ri in the attribute Aj. Let Wbe set of the tokenized keywords in the R. Search-as-You-Type for the Single keyword QueriesExact Search As the user types in the single partial prefixkeyword w character by the character search-as-you-type onflyfinds records that is contain keywords with the prefixw. We are call this is search paradigm prefix search. Deprived of loss ofgenerality every tokenized keyword is in the data set andthe queries are assumed to use the lower case characters.

Table 1, A_1= title, A_2 = authors, A_3 = booktitle, and A_4 = year. R = {r_1 ,r_2 ... ,r_{10} }. r3[booktitle]= ‘sigmod’.

W = {Privacy, Sigmod ,…}

III. DIVERSEMETHODS FOR SEARCH AS YOU TYPE

Possible different methods are to support search asyou typeand give their advantages and the limitations.Method first is to use the separate application layer inwhich can achieve the very high performance as it is can usevarious programming languages and the complex data structure.Howeverthe insulated from DBMS systems.Method is to use the database extender. For thatthis is extension based method not safe to the queryengine which is could causereliability and security problemto database engine. This is methoddepend on API ofthe specific DBMS being are used and different DBMS systemshave transposed API. Nevertheless this method doesn’t workuncertaintyDBMS systems has no this is extender feature.Method based on SQL ismore compatible whether it is using standard SQL. MySQL database is third methods to use SQL. So ifthe DBMS systems don’t provided search as you typeextension feature indeed on the DBMS systems provide are suchan extension SQLbased method can also to be used in thiscase. So SQL based method is much portable to thedifferent platform than start two methods.We takeattentionon the paper on the SQL-based method andthe developed various techniques to theachieved high cooperativespeed.

IV. EXACT SEARCH FOR SINGLE KEYWORD

4.1 No Index Methods

Straightforward way to support the search-as-you-type is tothe issue the SQL query that scans every record and verifies ifthe record is the answer to query. There are two main ways to the checking:

1) Calling User Defined Functions. We can add functions into a databases to the verify if record contains the query keyword

2) UsingLIKE predicate databases provide LIKE predicate toallow user to perform string matching. We can use LIKEpredicate to check if record haveencompasses querykeyword. This isthe method might the introduced incorrect positivesfor examplekeyword 'publication' contains the query string 'ic' but keyword doesn't have query string 'ic' by way of the prefix.We can eliminate these deceptive positives by the calling UDF. The methods are needs no additional space but they mayn't scale subsequently they need to scan entire records in table.

4.2 Index-Based Method

We propose this scenario to buildauxiliary tables as the indexstructures to facilitate the prefix search. Particular databases such asOracle and then SQL server already support prefix search and we could use this feature to do prefix search. However not entire database provide this feature. For this motive we develop the newfangled method that can be used in entire databases. In addition our experiments in this show that our method performs are prefix search much efficiently.Given a table T we assign uniqueids to the keywords in the table T following their sequentialorder. We create an inverted index table for IT with records in the form ikid and ridi, where kid is the id of keyword and rid is the id of the record that is encompasses keyword. Given the complete keyword we can use an inverted index table to the find records with keyword.Prefix table:The assumed the table T for the entire prefixes of keyword we build the prefix PT with the recordsinformhp; lkid; ukidi, where p is a prefix of a keyword kid is smallest id of this keywords T having p as the prefix, andukid is the largest id of this keywords having p as the prefix. The interesting observation is that the complete wordWiththe p as the prefix must and should have the ID in the keyword range $[lkid; ukid]$ and every complete word in T with an IDin this keyword range must and should have the prefix p. So given the prefix keyword w we can use prefix table to find therange of the keywords with prefix. $SELECT\ T.*\ FROM\ p_T, l_T, T\ WHERE\ p_T\ .prefix = \text{w}\ AND\ p_T.ukid \geq l_T$.kid AND $P_T.lkid \leq l_T.kid$ AND $l_T.rid = T.rid$.

For example assuming a user types in the partial query "sig" on the table dblp issue on the following SQL.

$SELECT\ dblp.*\ FROM\ p_{dblp}, l_{dblp}, dblp\ WHERE\ p_{dblp}\ .prefix = \text{"Sig"}\ AND\ p_{dblp}.ukid \geq l_{dblp}$.kid AND $P_{dblp}.lkid \leq l_{dblp}.kid$ AND $l_{dblp}.rid = dblp.rid$.

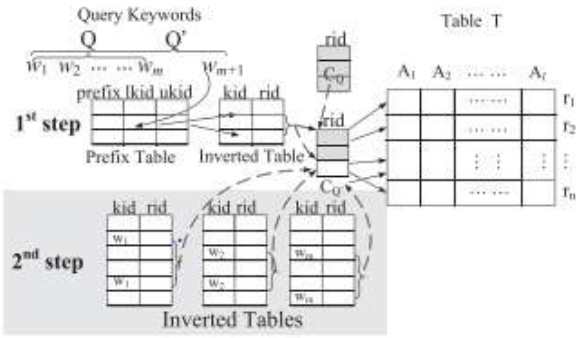
V. SUPPORTING MULTIKEYWORD QUERIES

Computing Answer from Scratch:

The given multikeyword query Q with m the keyword $w1, w2, \ldots, wm$.

1) Using the 'INTERSECT' Operator:- The straightforward wayis to the first compute records for every keyword using previous method and then use the 'INTERSECT' operator to join these records for the different keywords to compute the answers. 2) Using completetext Indexes: we first use full-text indexes (example CONTAIN command) to find records are matching the first m 1 complete keyword and then use ourmethods are to discovery records matching the latter prefix keyword. Lastly we join the results. These two methods cannot usethe recomputed results and may lead to the low performances. To the address this is problem we propose the incremental computation method.*Word-Level Incremental Computation:* We can usethe previously computed results are to the incrementally answer the query. Presumptuous the user has typed in the query Q with the keywords $W_1, W_2, W_3, \ldots, W_n$ we generate the temporary table CQ to the cache records id of the query Q.

Whether the user types in the new the keyword wmþ1 and submits the new query Q0 withthe keywords$W_1, W_2, W_3, \ldots, W_m . vmp_1$ we use the temporary table CQ to the incrementallyanswer the new query.



ig. 5. Incrementally computing first-N answers.

VI. SUPPORTING FIRSTN QUERIES

Previous methods are focus on the computing all answers.

As the user types are in the query character by the character, we usually give user first N resultsis as instantFeedback. This section discusses how to the compute first Nresults.Exact first N queries result, for we exact search could use 'LIMIT N' syntax in databases toreturn first N results, MYSQL uses 'LIMIT n1: n2' to return n2 rowsstarting from the n1 row.Method based on inverted index table and to the prefix table. Our techniques can be easilyextended to other method.The single keyword query we can use for 'LIMIT 0 and N find first N answers. Assume the user types in the keyword query 'sig' to compute first 2 answersand issue the following SQL:

```
SELECT dblp.* FROM Pdblp, Idblp, dblp WHERE Pdblp.prefix = "sig" AND Pdblp:ukid
        ≥ Idblp:kid AND Pdblp:lkid ≤ Idblp:kid AND Idblp:rid = dblp:rid LIMIT 0; 2,
```

VII. SUPPORTING UPDATES EFFICIENTLY

We can use generate to the support data updates. We consider Insertion and deletions of the records. The Insertion. Take as a record is inserted. We first assign it innovative record ID's. For each keyword in record we insert keyword into the inverted index table. For every prefix of keyword whether prefix is not in the prefix table we add the entry for prefix. For keyword range encoding of each prefix we can be reserve extra space for the prefix idtotheaccommodated future insertions. We can only the necessary to do globalre-ordering whether the reserved space of insertion is the consumed. Deletion: Assume that the record is deleted. For every keywordin records are inverted index table we use the bitto denote if the record is deleted. Here we can use the bit tothe mark record to be deleted. We don't update tableuntil we can need to the rebuild index. For example range encoding of theevery prefix we can usethe deleted prefix ids for future Insertion.The range of the ids is assigned based inverse document frequency (idf) of the keyword. We use the bigger range for the keyword with smaller idf. In many cases we can use andkept extra space for the update. Nevertheless in the worst case we needto the rebuild index. The problem of ranges selection andanalysis is beyond the scope of this scenario.

Data Updates:

We are tested cost of updates on the DBLP data set. We first constructed indexes for 1 million records and then inserted 10000 records at all-time. We compared the performance of three method are on inserting 10000 record. It took more than 40 seconds tore-index data while our incremental indexing method only took0.5 seconds.

Exact Search:

In the single keyword queries, we implemented three methodsFor the single keyword queries:

1) Exhausting UDF. 2) Exhausting the LIKE predicate.

3) Exhausting inverted index table and prefix table known as the 'IPTables'. We are compared the performance of this three methods to the compute the first N answers. Unless otherwise specified N=10. Method had low search performance as theyare necessary to the scan records. IPTables achieved a high performanceby using index Keyword length augmentedperformance of the Start two method decreased the subsequently keywordsuitedmuch selective and the two methodsrequiredscanning more records in order to find the samenumber (N) of answers. As the keyword length increasedIPTables had the higher performance since there were fewer wholekeywords for the query and query necessaryfewer join operations.Multikeyword queries. We implemented six methods for multi-keyword queries using UDF;2. Using the LIKE predicate. 3. Using full-text indexes and UDF known as "FI+UDF". 4. Usingadequate text indexes, LIKE predicate known as 'FIpLIKE'.

5. Using the inverted-index table and prefix table(IPTables);

VIII. EXISTING SYSTEM

In the existing system strings are search which given the set of strings and the query string all strings in the set that are similar to query string. Likeness joins are the extensively studied which is given two sets of the string find entire similar string pairs from two sets. Uses are the existing built in the functionalities for example full-text indexes and CONTAINS command in Oracle and the SQL Server. This system follows q-gram algorithms. The keyword search is happening as taking query keywords as the complete keywords.Existing technique are the Q gram-based technique.

Disadvantages:

1. Firstly they are needed to searchsimilar introducesof keyword from scratch.
2. Secondly they may be essential to call UDFmany times.

3 Concentration on the computing entire the answers.

IX. PROPOSED SYSTEM

In this propose system we consideredas the problem of the using SQL to support the search-as-you-type in thedata bases. The focused is on the challenge of how to leverage existing DBMS functionalities to the meethigh-performance requirement to achieve interactive speed. To the support prefix matching we proposed the solutions that use the auxiliary tables as the index structures and the SQL queries to support search-as-you type. We are protracted techniques are to case of the fuzzy query and suggested vary many techniques to improve the query presentation. We proposed the incremental computation techniques toanswer multi keyword queries and deliberate how to the support first N queries and the incremental updates. Our experimental results on the large real data sets showed that proposed techniques could allowthe DBMS systems to the support search as you type on big tables.

Propose Technique**:** The Incremental computation techniques&the novel techniques for the fuzzy search.

X. ADVANTAGES

- Improving Performance Using the Indexes.
- we propose efficient techniques to the support
- Multi keywords queries.
- Word Level Incremental Computation (WLIC) write the user in the query character by we usually give the user first N (any-N) results as imperative feedback.
- We can use trigger to support the data update and weare consideringthe insertion and deletion of record by using the Fuzzy Search.

XI. CONCLUSION

In this scenario we focused on challenge of how to the leverage existing DBMS functionalities to meet the high-performance requirement to the achievedthe interactive speeds. To support the prefix matchingare proposed solutions are that use auxiliary tables as the index structures and SQL queriessupport to search as you type. Long drawn outtechniques to case of the fuzzy queries and anticipatedseveral techniques to the improve query performance.In the proposed we are incrementalcomputation techniquestothe answer ofmulti keyword queries and the studied how tosupport the first N queries and the incremental update. Our main aim to experimental results on large real data sets showed that proposed techniques can beallowed DBMS systems tothe support search-as-you-type on the big tables.Many problem support SQL suppose asyou typeusing SQL, one is how to support ranking queriesefficiently and other is how to support the multiple table.

REFERENCES

[1] S. Agrawal, K. Chakrabarti, S. Chaudhuri, and V. Ganti, "Scalable Ad-Hoc Entity Extraction from Text Collections," Proc. VLDB Endowment, vol. 1, no. 1, pp. 945-957, 2008.

[2] 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.

[3] A. Arasu, V. Ganti, and R. Kaushik, "Efficient Exact Set-Similarity Joins," Proc. 32nd Int'l Conf. Very Large Data Bases (VLDB '06), pp. 918-929, 2006.

[4] H. Bast, A. Chitea, F.M. Suchanek, and I. Weber, "ESTER: Efficient Search on Text, Entities, and Relations," Proc. 30th Ann. Int'l ACM SIGIR Conf. Research and Development in Information Retrieval

[5] (SIGIR '07), pp. 671-678, 2007.

[6] 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.

[7] H. Bast and I. Weber, "The Complete Search Engine: Interactive,Efficient, and Towards IR & DB Integration," Proc. Conf. Innovative Data Systems Research (CIDR), pp. 88-95, 2007.

[8] R.J. Bayardo, Y. Ma, and R. Srikant, "Scaling up all Pairs Similarity Search," Proc. 16th Int'l Conf. World Wide Web (WWW '07), pp. 131- 140, 2007.

[9] G. Bhalotia, A. Hulgeri, C. Nakhe, S. Chakrabarti, and S. Sudarshan, "Keyword Searching and Browsing in Data Bases Using Banks," Proc. 18th Int'l Conf. Data Eng. (ICDE '02), pp. 431- 440, 2002.

[10] K. Chakrabarti, S. Chaudhuri, V. Ganti, and D. Xin, "An Efficient Filter for Approximate Membership Checking," Proc. ACM SIGMOD Int'l Conf. Management of Data (SIGMOD '08), pp. 805- 818, 2008.

[11] S. Chaudhuri, K. Ganjam, V. Ganti, R. Kapoor, V. Narasayya, and T. Vassilakis, "Data Cleaning in Microsoft SQL Server 2005," Proc. ACM SIGMOD Int'l Conf. Management of Data (SIGMOD '05),pp. 918-920, 2005.

[12] S. Chaudhuri, K. Ganjam, V. Ganti, and R. Motwani, "Robust and Efficient Fuzzy Match for Online Data Cleaning," Proc. ACM SIGMOD Int'l Conf. Management of Data (SIGMOD '03), pp. 313-324, 2003.

AUTHOR PROFILE

Nagarjuna Kolluru is currently pursuing M.Tech in the Department of Computer Science & Engineering, from Nalanda Institute of Technology (NIT), siddharth Nagar, Kantepudi(V), Sattenapalli (M), Guntur (D), Andhra Pradesh , Affiliated to JNTU-KAKINADA.

A Ramaswamy Reddy (M.Tech, Ph.D) working as Associate Professor & HOD (CSE) in Nalanda Institute of Engineering & Technology (NIET), siddharth Nagar, Kantepudi(V), Sattenapalli (M), Guntur (D), Andhra Pradesh , Affiliated to JNTU-KAKINADA.