**INTRODUCTION**

The project’s objective was to create a Browsing and Keyword search application for the relational databases. It is useful for searching of data within a database without a prior knowledge of the Structured Query Language. The results which were displayed to the user were rows and were ranked according to the priority( from highest to lowest).

**ANALYSIS**

Below is my first algorithm for this application

A table which would contain relations in decreasing order of their priority.

A Dictionary table which would contain all the values of the database in it along with table and column name to which it belongs.

A join table which would consist of all the tables in one column and the tables with which they have a relationship in another column.

While the string is processed word by word.

Take a word and search it in the dictionary table.

If found then extract the table and the column name and store it in a data structure.

Use the join table to apply the join operations on the table name extracted.

Apply the query  ## select \* from Ri where Ci = value ## where Ri is the view producd using join operation and Ci is the column name.

Rank the results using the table which contains priority of the relations. The join view created using maximum number of highest priority tables is ranked the highest.

Then I found that the dictionary table in the above algorithm would take a lot of space and therefore I constructed another algorithm which is a better version of the above algorithm.

The algorithm would consist of 1 graph, 1 tree and  2 tables.

Tree - **Patricia trie** for implementing dictionary for the database.

Graph- Which shows the relationship among the tables of the database ( there can be many disconnected graphs and each node in the graph would consist of tablename and a flag value).

Table 1- Maps table name and attribute name to table number and attribute number.

Table 2- Maps table name to the address of the node in the graph.

The algorithm would run as follows:

1. User enters the query.

2. While the string is processed word by word. Take a word and search it in Patricia trie.

3. The table name and attribute name are retrieved for the words found in Patricia trie.

4. The graph is traversed and flag is set for all the table names found.

5. Then join operation is applied on the tables which are related.

6. Finally tuples are retrieved and the results are displayed to the user.

For the ranking of the results in the above algorithm , the strategy used was that a string with maximum number of joins would have the minimum rank. This is because , the more the number of joins , the less related the data would be in the result and therefore the less relevant it would be to the user. The best result would be a row which would have all the words of the string entered by the user.

Initially I was using java as a platform and mysql as the dbms. Then later I found that the project required that the relationship between two unmatching values have to be shown as well. For example  let user insert string ‘abc xyz’ in a friends database. Let 'abc' is friends with 'mno' which in turn is friends with 'jkf' which in turn is friends with 'xyz'. So I want to show this relationship as well. The application should search all such relationships , if they exist and not just of friendship type relationship but all the  others that can exist and display them. Another exapmle of relationship can be in database in which a student and a teacher are related through the papers that they have published together or two students are related through a common professor that they have worked with.

Before going on to understand on how to implement the above concept, through my search on the internet I was able to find some suitable ways of implementing the basic searching of values corresponding to a string of words and created a stored procedure as follows :

CREATE PROC SearchAllTables

(

@SearchStr nvarchar(100)

)

AS

BEGIN

DECLARE @Results TABLE(ColumnName nvarchar(370), ColumnValue nvarchar(3630))

SET NOCOUNT ON

DECLARE @TableName nvarchar(256), @ColumnName nvarchar(128), @SearchStr2 nvarchar(110)

SET @TableName = ''

SET @SearchStr2 = QUOTENAME('%' + @SearchStr + '%','''')

WHILE @TableName IS NOT NULL

BEGIN

SET @ColumnName = ''

SET @TableName =

(

SELECT MIN(QUOTENAME(TABLE\_SCHEMA) + '.' + QUOTENAME(TABLE\_NAME))

FROM INFORMATION\_SCHEMA.TABLES

WHERE TABLE\_TYPE = 'BASE TABLE'

AND QUOTENAME(TABLE\_SCHEMA) + '.' + QUOTENAME(TABLE\_NAME) > @TableName

AND OBJECTPROPERTY(

OBJECT\_ID(

QUOTENAME(TABLE\_SCHEMA) + '.' + QUOTENAME(TABLE\_NAME)

), 'IsMSShipped'

) = 0

)

WHILE (@TableName IS NOT NULL) AND (@ColumnName IS NOT NULL)

BEGIN

SET @ColumnName =

(

SELECT MIN(QUOTENAME(COLUMN\_NAME))

FROM INFORMATION\_SCHEMA.COLUMNS

WHERE TABLE\_SCHEMA = PARSENAME(@TableName, 2)

AND TABLE\_NAME = PARSENAME(@TableName, 1)

AND DATA\_TYPE IN ('char', 'varchar', 'nchar', 'nvarchar')

AND QUOTENAME(COLUMN\_NAME) > @ColumnName

)

IF @ColumnName IS NOT NULL

BEGIN

INSERT INTO @Results

EXEC

(

'SELECT ''' + @TableName + '.' + @ColumnName + ''', LEFT(' + @ColumnName + ', 3630)

FROM ' + @TableName + ' (NOLOCK) ' +

' WHERE ' + @ColumnName + ' LIKE ' + @SearchStr2

)

END

END

END

SELECT ColumnName, ColumnValue FROM @Results

END

==============================================================================

The above stored procedure has not been executed but the searching that it implements would require the same concept as this. However this can be run only on Microssoft Server SQl DBMS and not on mysql DBMS. As MSSQL provides keywords which improves the flexibility in writing stored procedure . Thus we see that stored procedure did reduce the overhead of programming which would be caused if such an application would have been implemented in java. The time complexity of the stored procedure has not been calculated as the code could not be implemented. The reason behind the unexecuted code was that MSSQL 2008 was not compatible with my laptop as it supported windows 7 and not windows 8. The new version , which is MSSQL 2012 has a lot of versions and thus I was unable to decide which version would be supported by windows 8.

However the above stored procedure does not show the relationship among the results which is an important feature of the original application which we had in mind. So I did a search on how to implement it and that too using mysql as database. Through my search I understood that this could not be implemented using mysql alone. Also, in order to show the relationship among the results , I had to visualize the database as a graph in which the tables were the nodes and the relationship between them were the edges. Neo$j and Orientdb are two DBMS which can be used for graphical implementation of databases. These also provide inbuilt commands which help in traversing these databases like a graph. But we required a generic code which can be implemented using any DBMS and should not be DBMS specific.

Then when the search got exhausted and no such queries could be found which can show the relationship, I created an algorithm which is database specific. For the above example of Friends database, the algorithm would run as follows :

Consider that you have a table which has many columns with one column as the 'person' and other column as 'friends'.

We can use queue as a data structure here.

Execute an SQL query to get friends of 'abc'.

Add 'abc' to a HashSet

Push all the friends in a queue

While there exists any friend in the queue

         Get a friend from the queue

         Execute an SQL query to get the friends of the above friend

         Push all the friends above which are not present in the HashSet into the queue

         Add all the currently pushed elements to the HashSet

         Print the friend and the relationship

You can always get the friends from the result set of a SQL query.

You can use this generic proc to find not only the friends relationship but also any other kind of relationship. For finding all relationships only the SQL query needs to be modified. Say if I have columns in my table as friends, brother, mother, father then do "SELECT friends brother mother father FROM relationships WHERE name = 'abc' ". You can get all the people from the result set of this query and push all these people in the queue mentioned in the above proc.

However, the above algorithm remains database specific and may not do the entire job. Finally I found **Semantic Web** where anything can be expressed as a triple which is of the form <subject,predicate,object> . Assume I have an employee table with emp\_id, emp\_name and salary as the columns. I can convert the above form as <node1 emp\_id 1000>, <node1 emp\_name Shrey>, <node1 salary 2000000>. This way we can convert a relational table into the triples format. Now if I execute a query of the form <node1 ?x ?y>, it not only gives me the details but also the relationships which are the predicates in the triple format. It is also possible to get all the relationships of a person irrespective of the column names but we need to convert our relational table into the triples format.

I have still a lot to learn about the semantic web in order to develop a code which is suitable for my application but due to the time constraint I would have to leave it till this point. The above is just an example of one of the many uses the semantic web can have. Further exploration is required from my side on this subject.

**ACKNOWLEDGMENTS**

1. My mentor Rupesh Nasre whose guidance helped me through thick spots during my internship.
2. BANKS paper by IIT Bombay
3. DBXplorer paper by Microsoft Research.
4. PageRank paper by Larry Page.