Journal of Theoretical and Applied Information Technology
15th March 2012. Vol. 37 No.1

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

FFQI-FAST FORMULATION QUERY INTERFACE FOR DATABASES

[1]R.SHOBANA, [2]D.VENKATESAN

[1]Dept of Computer Science and Engineering, SASTRA University
[2]Asst. Prof., Department of Computer Science and Engineering, Sastra University
E-mail: [1]shob29@gmail.com , [2] venkatgowri@cse.sastra.edu

ABSTRACT

We present a new query formulation interface called FFQI (Fast Formulation Query Interface) which is based on a semantic graph model. The query formulator allows the users with limited IT skills to query and explore the data source easily and efficiently. Here the user inputs are formulated based on the graph search algorithm by using the probabilistic popularity measure. The query ambiguity has been resolved through the ranking technique. We formulated the SELECT-PROJECT-JOIN queries using the aggregate functions. In additional to that we also implemented a formulation technique for image databases. Thus this interface allows user to interact with relational graph-type databases in an effective and easier way.

Keywords: *Query Formulation, Semantic Graph, Relational Database, Select-Project-Join (SPJ) queries*

1. INTRODUCTION

Many organizations rely on heterogeneous and distributed information systems for managing immense quantities of data. Because of the vital structural complexity of the associated databases the query formulation within such systems has been too hard. The major challenge is to make a proposal so that the end users can effortlessly search and consume structured data which receives in recent times a great consideration from web 2.0 and the data web communities. Rapid intensification of structured data on web created a high demand in order to make this content more reusable and consumable. Companies therefore race on gathering structured content and making it public and also they persuade people to reuse and profit from that content. Numerous database applications require users to formulate specific queries instead of invoking precompiled and stored queries. Thus it's vital to develop an intelligent query interface which let users to query and explore the data source easily. And the Query formulation should be fast and should not require programming skills.

Formulating complex queries is the toughest job. In our application a general graph search approach has been developed to formulate SPJ queries from incomplete user input. Through aggregate functions our query formulator formulates SPJ queries. In this approach, a semantic graph is used to model the objects in the database and user-defined relationships have been semi-automatically generated from a database schema. A given input may possibly have multiple path sets, hence the system ranks the candidates based on amount of information present in the nodes and links of the paths.

This paper is organized as follows: Next Section reviews the related work, Section 3 describes about the semantic data model, Section 4 describes about graph searching technique, Section 5 describes about Query Generating for Conditions, Section 6 describes about Query Generation for Join Operations, Section 6 describes about Query Generation for Image Database and Section 7 describes about Experimental Result and Section 8 concludes the paper with future work.

2. RELATED WORK

Universal relation model[1,2] and Steiner tree approach[3] are the various techniques for formulating simple SELECT-PROJECT-JOIN(SPJ) queries. The exclusive hypothesis of relationships in order to mitigate users from the load of specifying

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

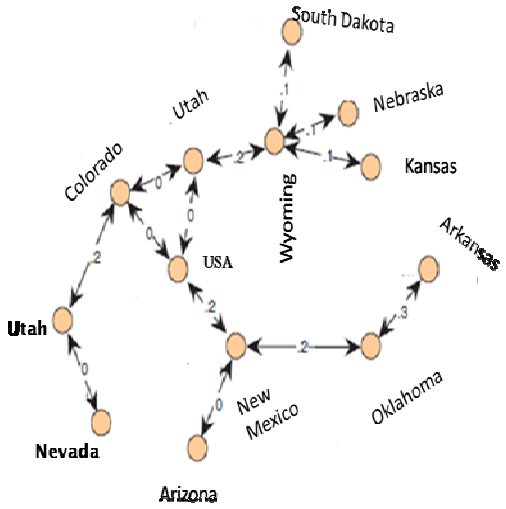
joins which is given by the universal relation model[1,2].However illogical user-defined concepts were not allowed by this model, which limits its applicability. The conversion of cyclic schemas into a tree schema by its maximal object theory may limit the queries that can be formulated. Through Steiner tree problem the query completion has been formalized by Wald and Sorenson [3], and a search algorithm for partial 2-tree graphs has been provided. A deterministic directed cost for the edges was used by them, which provides the cardinality of relationships to measure the complexity of queries.

Path expression completion in object-oriented queries with a partial order relationship between different paths has been used for ranking. Three high-level interfaces to database systems are discussed in [4].In [5] the use of disambiguating queries is developed. Through dialogue tree dialogue is carried out with the user to determine what additional attributes the user is interested in.

In [6] [7], Query-By-Form which is a simple querying method is provided. A form is needed to be developed for each query, and a query-change implies changing that whole form. In [8], Query-By-Example which is a known approach in databases where users formulate queries through filling tables is given. In [9] the query formulation for relational database using high level concepts is given. In [10] the basic concepts for designing database are given.

3. SEMANTIC GRAPH MODEL

The semantic graph model for relational databases contains nodes to represent relations and links to represent the joins between them. The semantic constructs represented by them are;



Extra underneath features for user query interfaces and query formulation mechanisms are provided by our semantic model .In general a semantic graph can be defined as a weighted undirected graph G=(V,E),where V corresponds to a relation and E corresponds to a join between that relations .

3.1 Popularity Measure of Nodes and Links:

For a given user input multiple queries may be formed, in the query formulation. The popularity of nodes and links is used for selecting and ranking query candidates. The popularity measure for an element 'k' (a node or link) in a semantic graph measures the popularity content of 'k';

$$Pop(k) = P(k);$$

where P (k) is the probability of using 'k' in queries The definition is consistent with that used in popularity theory which represents the frequency of usage of a node or link. The measure gives the popularity content of a node (or) link. A larger value of Pop (k) means a larger P (k) thus it will more likely appear in queries.

In computing the popularity of a sub graph for simplicity, we assume that all the nodes and links are independent. For a sub graph with a set of elements (nodes and links) A= ($\{a_i \mid i = 1...n\}$), the independence assumption implies that the popularity measure is additive, that is

$$Pop(K) = P(\{k_i | i\text{-}1....n\}) = \sum_{k=0}^{n} P(K_i) \quad (1)$$

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

3.2 Popularity Measure Update:

The popularity measure for nodes and links can be computed from their relative frequency. Let c_i be the number of times that k_i is used in queries and c be the total number for all the elements used in a set of queries then,

$$Pop\ (k_i) = \frac{ci}{c}$$

The popularity measure of element k_i can be updated by the definition

3.3 Initial Popularity Measure Assignment:

If a large collection of queries are available at the beginning, initial counting can be performed. But if the query set is not available or is too small to be statistically significant, we can assign an equal initial popularity measure to all the nodes and assign popularity measures to links based on the link types and their specificity

4 QUERY FORMULATION AS A GRAPH SEARCH PROBLEM

For query formulation, an incomplete query topic T_0, a characteristic C and a constraint set X can be processed from the given user input. The links specified by the user input, the nodes involved in the links and the nodes in C and X can be included in T_0.

We need to choose additional links and relevant nodes from the semantic graph to extend T_0 to form a connected sub graph for the query topic. Now these links and nodes can be called as query completion candidate for T_0.

4.1 Property of a query completion candidate

Given a semantic graph G=(V,E),to formulate a query from an input query topic $T_0=(V_0,E_0)$,where $V_0 \subseteq V$ and $E_0 \subseteq E$, is to find a query finishing point candidate $T_F=(V_F,E_F)$ for 'T_0' such that the query topic $T=T_0 \cup T_C=(V_0 \cup V_F, E_0 \cup E_F)$ is a connected sub graph of G, where $V_F \subseteq V, E_F \subseteq E, V_F \cap V_0=\emptyset$, and $E_F \cap E0=\emptyset$.

That is V_F is a set of nodes and E_F is a set of links needed to complete a connected sub graph to formulate a query. There can be exist more than one query completion candidate for the same input, if the semantic graph is cyclic. We use the following popularity information principle for ranking the candidates.

4.2 Maximum Popularity (MP) Principle:

The query completion candidate T_F (the missing links and nodes) for an incomplete input topicT_0 contains the maximum popular information; *i.e.maxPop (T_F).*Based on equation 1, Pop (T_F) can be computed from the information of the nodes and links as follows.

$$\text{Pop } (T_F) = \sum_{v \in VF} Pop(v) + \sum_{e \in EF} Pop(e)$$

Thus this MP principle provides us the measure for ranking the query completion candidates. The larger the Pop (T_F), user's query intention are more likely to be met by the completion candidate. Links and nodes of most popular information have a higher probability of being used in queries, thus are more likely to be in the intended query. The probability value used in the ranking is an approximation, since the assumption can be done independently. Therefore, a set of completion candidates could be found out, from which the users are allowed to select one.

Based on the MP principle, the completed graph's end point must be from the user input. Thus our MP principle is consistent with the formulation of the query completion. And according to this query completion a graph search problem is NP-complete.

A query auto completion algorithm (QAC) gets user input x, which denotes the sequence of characters typed by the user in the search engine's search box. Typically a prefix of a complete query q that the user intends to enter will be the user input. If the completion c equals the query q that the user was about to enter, then we can say that as a hit. It is relatively easy to estimate hit rates when inspecting search logs, so that here in this paper we will mainly focus on hits as the main measure of success for QAC algorithms. A QAC algorithm may be flourishing even it returns a completion that is different from the query that user was about to type but that describes the same information need.

4.3 The framework of QAC algorithm:

A query database is built in an offline phase. A large collection of queries with high quality and which represent the intents of the search engine's users will

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

be there in the database. This database could be built from their query logs by extracting the most frequently searched queries. For example, Michael Jackson is a proper completion of the input bar. This advanced QAC algorithm also support non-proper completions, like mid-string completions (e.g., jack->michael jackson) and spell corrections (e.g., michel->michael jackson).

```
UserInput(x)
db = ddatabase.SelectedItem.Value
tb = dtables.SelectedItem.Value
For i = 0 To dfields.Items.Count - 1
        q=dfields.Items (i).Value
        If q (i). Contains(x) Then
                Item (i) = q (i)
End loop
Big=Popularity (Item (i))
For i= 0 To dfields.Items.Count - 1
If Popularity (Item (i)) <Popularity (Item (i+1))
Then
Big=Popularity (Item (i+1))
a =Big
s= "select” &” " & a & “from” & tb
return(s)
```

5 GENERATING QUERY FOR CONDITIONS

This FFQI can do additional task like applying conditions between table in a faster way.There is no need of any manual operation except the selection of tables and the operation to be done between them.The algorithm for this operation is given below.

```
Select_Condition()
{
    column1 = ddfirstcol_SelectedItem_Value
    column2 = ddsecondcol_SelectedItem_Value
    condition = ddcondition_SelectedItem_Value
    condition1=ddcondition1_SelectedItem_Value
    operation = ddoperation_SelectedItem_Value
If (con = "EQUAL") Then

      condition = "="
       ElseIf (con = "NOT EQUAL") Then
      condition = "!="
       ElseIf (con = "LESS THAN") Then
      condition = "<"

   ElseIf (con = "GREATER THAN") Then
      condition = ">"
End If
If Popularity (Item (i)) <Popularity (Item (i+1))
Then
Big=Popularity (Item (i+1))
a =Big
For i = 0 To column1_Items_Count - 1
        If (i = column1_Items_Count - 1) Then
          a = a + column1_Items_Value(i)
        Else
          a = a + column1_Items_Value(i) + ","
        End If
End Loop
squery = "select " + a + " from " + table + " where
" + column1 + "  " + condition + " '" +first_Text +
"' " + operation + " " + column2 + condition1 + " '"
+ second_Text + "'  "

return(squery)
}
```

6 QUERY GENERATION FOR JOIN OPERATIONS

In a similar way to the condition operation this join can be performed in an easy and a faster way. The algorithm for this operation is given below.

```
Join ()
{

 column1 = ddfirstcol_SelectedItem_Value
  column2 = ddsecondcol_SelectedItem_Value
 operation =Joinoperation_SelectedItem_Value
 For i = 0 To ItemsList1_Count - 1
  If (i = ItemsList1_Count -1) Then
          a = a + ItemsList1_Value(i)
        Else
       a = a + ItemsList1_Value(i) + ","
        End If
 End Loop
If Popularity (Item (i)) <Popularity (Item (i+1))
Then
Big=Popularity (Item (i+1))
a =Big
For j = 0 To ItemsList2_Count - 1
        If (j = ItemsList2_Count - 1) Then
          b = b + ItemsList2_Value(j)
        Else
          b = b + ItemsList2_Value(j) + ","

       End If
 End Loop

sqlQry = "Select " + table + "." + a + "," + table1
+ "." + b + "  from " + table + " " +
DropDownList1_SelectedItem_Value + " " + table1
+ " on " + table + "." + column + "=" + table + "."
+ column1
return(sqlQry)
```

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

}

7. GENERATING QUERY FOR IMAGE DATABASE

In an additional way we also generated query for image database. And the major fact here is it can be done in a faster way with just few selection operations. The algorithm for this operation is given below.

Query_Image ()
{
Im1=Get_Input_Image ()
Upload_Image (Im1)
If Popularity (Item (i)) <Popularity (Item (i+1))
Then
Big=Popularity (Item (i+1))
a =Big

img=SqlCommand("SELECTimagename,ImageID from [a] where ImageName='" +TextValue1 + "'")
dt=datatable
daimages_Fill(dt);
Images_DataSource = dt;
Images_DataBind();
img_Attributes_Add("bordercolor", "black");
Return(img)}

8 EXPERIMENTAL RESULT

A database interface system using the above query formulation techniques has been implemented. This tool can easily contain operations like, Query generation for join operations, Query generation for conditions, Query Auto Completion and Query Generation for image database. This tool needs offline datasets which can be automatically loaded in the designed tool. Any number of databases could be accessed by our interface. For image database the images can be chosen by the user which can be automatically uploaded in the database of that offline schema.

Figure1 Generating query for conditional operations

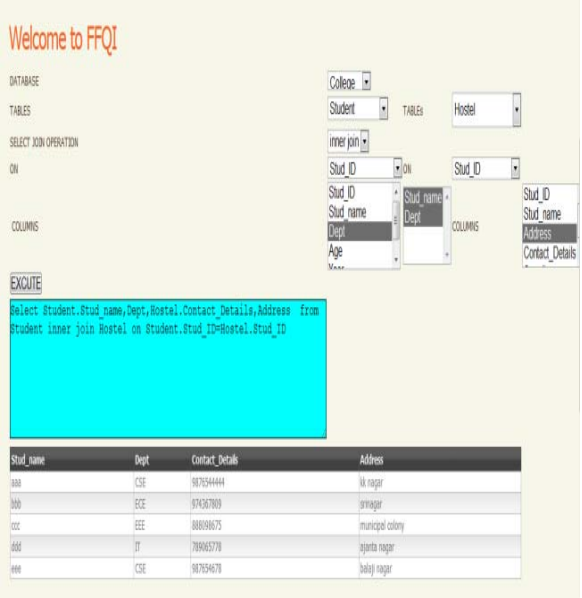


Figure 2 Query Generation for Join Operations

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Figure 3 Query Auto Completion

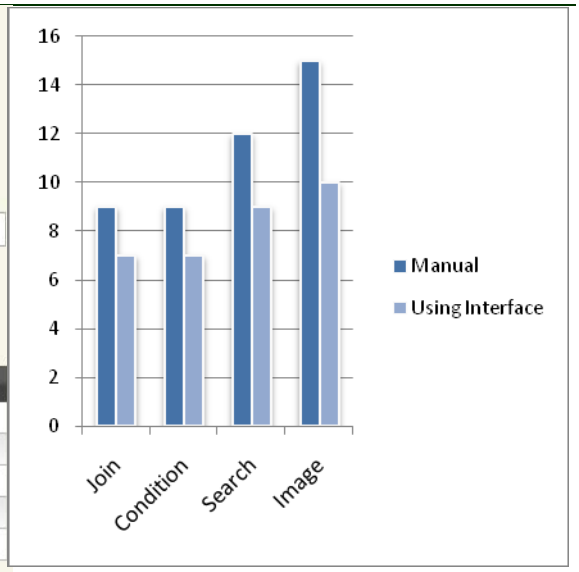


Figure 5 Manual vs Query Interface

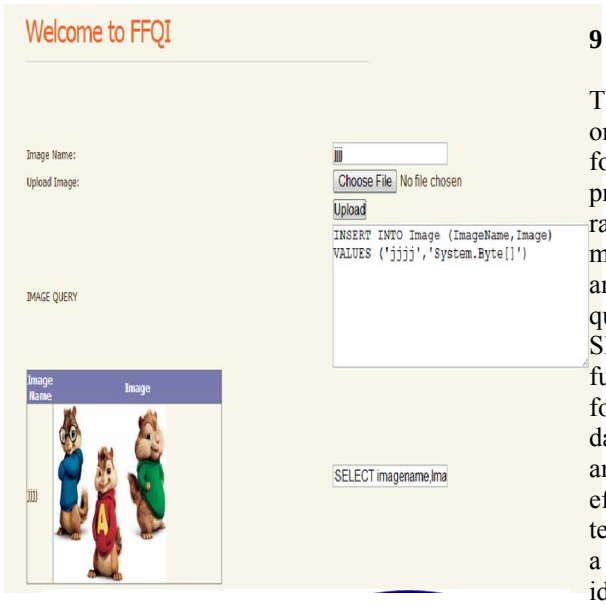


Figure 4 Generating Query for Image Database

Thus finally a graph which provides the overall time complexity for each operation in our query tool has been designed. Thus it provides an overall performance analysis of our query interface tool. This graph provides the comparison between manual and our interface for doing these operations.

9 CONCLUSION

Thus a new query formulation device which is based on a semantic graph model is presented. Here query formulation as a graph search which uses probabilistic popularity measure for searching and ranking query candidates. From a given user input multiple queries are formulated then that query ambiguity can be resolved through ranking. Our query formulator algorithm can formulate the SELECT-PROJECT-JOIN queries with aggregate functions. Additionally we also formulated queries for image database, which could manage the image database by adding and retrieving images in an easy and effective way. Thus we have constructed an effective prototype system using the above technique with simple point-and-click interface. As a future enhancement we have planned to extend our idea in distributed databases. By applying formulation for databases on different system in different environment for example intranet or internet application.

REFERENCE

[1] Jeffrey D.Ullman: Principles of database and knowledge-base systems, Vol.II. Computer Science Press, 1988

[2] Mosche Y.Vardi: The universal-relation data model for logical independence. IEEE software, pages 80-85, March 1988.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

[3] Joseph A. Wald and Paul G. Sorenson: Resolving the query interface problem using Steiner trees.ACM Transactions on Database Systems, 9(3):348-368, September 1984.
[4] Amihai Motro. A tri database user interfaces for handling vague retrieval requests. IEEE Data Engineering Bulletin, 12(2), 1989.
[5] A.D'Atri, P.Di Felice, and M.Moscarini. Dynamic query interpretation in relational databases. Information Systems, 14(3):195-204, 1989.
[6] Jayapandian M, Jagadish H: Automated Creation of a Form-Based Database Query Interface, VLDB 2008.
[7] Jayapandian M.Jagadish H: Expressive Query Specification through Form Customization, EDBT 2008.
[8] Zloof. M: Query-by-Example: Data Base Language, IBM Systems 16(4).
[9] Guogen Zhang: Query Formulation from High-Level Concepts for Relational Databases, IBM Santa Teresa Lab, San Jose, CA 95141,USA.
[10] Michael V. Mannino: Database Design, Application Development and Administration, fifth edition, University of Colorado, Deniver.

AUTHOR PROFILES

R.Shobana received the Bachelor of Engineering in Computer Science and Engineering from PREC, Anna University, in 2010. Currently she is doing her Master's Degree in Computer Science and Engineering in Sastra University. Her interests are in Data Mining, Formulating Queries and Information Retrieval

D.Venkatesan received the M.Sc. degree in Applicable Mathematics and Computer Science from the Bharathidasan University, in 1988.He received the M.Phil.degree in Computer Science from the Alagappa University in 2004. Currently, he is a Assistant Professor in SOC, SASTRA University, Hisr research interests are in soft computing, databases and data mining.