Volume 3, Issue 12, December 2013 ISSN: 2277 128X

International Journal of Advanced Research in Computer Science and Software Engineering

Research Paper

Available online at: www.ijarcsse.com

# A Study on the Existing Techniques for Query Interface Integration

**Sherlin Mary Koshy*, G. Naveen Sundar**

*Dept of CSE, Karunya University,*

*India*

***Abstract— A large amount of the information available on the World Wide Web today is accessible only by the use of query interfaces. A query interface is often the only means of extracting information from certain kinds of web based databases. There are often several query interfaces in existence for a single database, hence the integration of these query interfaces is essential to ensure swifter and more efficient querying. Some of the techniques currently available for the integration of query interfaces have been presented.***

***Keywords— Query Interface, Interface Integration, Interface Representation, Attributes, Unified Interface***

## I. INTRODUCTION

Querying a web database is an oft used technique for extracting information from the World Wide Web. Entering a term into the search field of a search engine, booking flight tickets online, online shopping etc all involve the querying of databases. The querying is done by means of query interfaces which are essentially HTML forms with elements such as text boxes, radio buttons, selection lists and so on. The values of these elements are combined to generate a query that searches through the web database to retrieve the necessary results. However it is often the case that there are several query interfaces in existence for databases in a single domain. For example if one were attempting to book flight tickets and find the airline offering the most affordable rates, one would have to repeatedly type the same information into several query interfaces belonging to different travel operators. Hence from a user point of view a single query interface would be highly beneficial.

Though it may seem straightforward there are several nuances that need to be considered. The single unified query interface is replacement only from the point of view of the user and not for the underlying web database. Hence any generated unified interface should retain the characteristics of the constituent query interfaces. Therefore there is a need to carefully identify these characteristics and ensure that they are preserved in the unified interface.

## II. TECHNIQUES FOR INTERFACE INTEGRATION

There are several issues that need to be considered before integration, these include indentifying the various attributes and constraints imposed on them by the individual interfaces, these constraints need to be maintained by the unified interface as well. Sometimes interfaces in the same domain may use different names for the same element, hence attribute matching also needs to be considered. Once a unified interface is generated the name to be given to the representative element should also be taken into account. These issues have been dealt to varying degrees in the techniques presented. The remaining sub-sections first provide an overview of the suggested technique followed by a detailed description.

### *A. WISE-Integrator*

This method [1]-[3] suggests a twostep technique for the integration of interfaces. WISE Integrator is the abbreviation for Web Interface Search Engine Integrator. The first step is clustering which is a technique of positive matching where attributes are grouped into clusters based on exact matching, clusters are then merged based on semantics and a representative name is chosen for each cluster based on a majority rule. The second step Weight Based Match is a predictive match where the knowledge gained in the first step is utilized. Given a local attribute its representative in the cluster is looked up and added to an intermediate interface, the process is repeated until all local attributes have been exhausted. Maintenance of the generated interface [2] is considered since interfaces may be required to be added or removed.

In this interface representation each interface is said to contain a list of ordered attributes with each attribute having one or more associated elements. Each attribute has a descriptive text linked with it called a label which in turn becomes the attribute name with which it is identified. Each attribute has a definite format for the input. Four types of formats are usually defined: radio button, check box, selection list and textbox. Each element has a domain which specifies the values with which the element can be instantiated when queries are being formed. For example the textbox allows the user to input any value, while selection list restricts him to a few choices. Apart from the domain, the domain type of the attribute is also defined. Four domain types have been identified range, finite, infinite and Boolean. The format of the attribute determines it type, for example the text box would be of type infinite since the user can enter any value. Elements or groups of elements can also have default values. The value type of an attribute is also defined which can be one of the following date, time, datetime, currency, number, char and id. Every attribute is finally also described

 

by its layout position on the local interface. The value of the position is decided by the layout order of the attributes on the interface, more significant attributes are usually placed before the less significant ones.

The problem of interface extraction has also been considered, two steps are involved as follows. Firstly, given an interface logically related labels and elements are grouped together in such a way that the grouping reflects a logical attribute of the underlying database. Each group is then identified with an attribute label. Once this is done meta information of the attributes required for interface integration is extracted. The meta information that is extracted includes the domain type, value type, default value and the unit, where unit refers to the units of currency, weight etc.

The precursor to integrating interfaces is the identification of matching attributes. This is necessary since interfaces are created by independent developers hence in the book domain one interface may use the term author, while another specifies writer. Both these term refer to the same element, therefore attribute matching is vital for integration to be done. In order to increase the chances of a match the following steps are taken: every name or value string is converted to its equivalent lower case, all content within parentheses and the parentheses themselves are removed, any non-alphanumeric character is replaced a space character, each name or value is tokenized using space, abbreviations and acronyms are replaced and finally non content words such as and, the etc are removed. The next stage involves identifying semantic relationships between attributes; there are three types of identified semantic relationships synonymy, hypernymy and meronymy. Synonymy refers to terms that are similar in meaning such as author and writer. Hypernymy refers to generic terms, i.e., tree is a hypernym of maple. Meronymy refers to term that are part of another for example first name is meronym of name.

Matching of the attributes is performed using their corresponding meta-information. There are two types of matches positive matches and predictive matches. Positive match clustering involves first grouping the attributes into clusters based on exact matching of attributes in all interfaces; once this is done all attributes in the same cluster will have the same name. These clusters are then merged into larger clusters based on the matching of between attribute values and the semantic matches of attribute names between clusters. A representative attribute name for each merged cluster is determined; this name is usually decided based on generality and majority rule. Predictive match involves clustering all potentially matching attributes together and then generating the global attribute for each group of matching attributes. Initially there is no existing cluster; all attributes in the first considered local interface form their own clusters. Each attribute is looked up in the results of positive match performed previously to create a attribute cluster thesaurus. For every new attribute this thesaurus is first looked up, to determine which cluster the attribute belongs to. Once the clusters have been formed a global attribute name for the clusters have to be determined, each cluster corresponds to one attribute on the global interface. Hence the global attribute's name, domain type and value should also be determined. This is often determined based on majority rule.

The final step is the generation of the integrated interface. For which three issues need to be considered the style of presentation of the attributes, layout of the attributes on the interface and the selection of which attributes need to be in the interface. Determination of attribute location is based on the aggregation of the positions of the local attributes. Attribute selection is done by trimming the less important attribute values.

*B. Merging Based on Clustering Aggregation*

This method [4] represents a query interface as an ordered tree of elements with the leaf nodes corresponding to an attribute on the interface. The proposed algorithm returns the root of the unified schema given the set of attributes of the component schemas. The algorithm creates clusters among the attributes based on their parents, sub trees are formed for each cluster, and the sub trees are then merged to generate the unified schema. Depiction of the interface is done as an ordered tree of elements, where every leaf node is analogous to an attribute on the interface. Internal nodes have a set of sub nodes, where the sub node could be a leaf node or another internal node. Every interface imposes two types of constraints of the attribute, structural constraints which are defined as ancestor descendent relationships on the lowest common ancestor of the attributes, this confines the structure of the unified schema. The other constraint imposed is the precedence constraint which restricts the ordering of the elements in the integrated schema.

The integration problem is defined as follows, given the set of interfaces, a set of distinct attributes can be determined. A unified schema needs to be developed such that the leaf elements in the global schema are part of the distinct set of attributes, the structural constraints of the initial set of interfaces are preserved in the unified schema and the number of precedence constraints of the local schema satisfied in the global schema is maximized. The algorithm Lmax is presented which forms recursive partitions over a set of attributes such that structural constraints from the constituent schemas are satisfied. The working of the algorithm is as follows, the algorithm is first given a set of unique attributes from all the interfaces being considered. At the first iteration the algorithm generates a root node, forms partitions over the given set of attributes, and creates a list of child nodes for the root where each child corresponds to a cluster in the partition. This process is recursively applied to each child of the root, given the attributes in the cluster corresponding to that child.

*C. Merging Based on Web Databases*

This technique [5] proposes an algorithm that generates a unified tree after considering potential groups of attributes. Given the set of query interfaces clusters are formed for the attributes based on frequency of their occurrences in constituent schema. The merging is performed with two interfaces at a time. Query interface is represented as an ordered tree of elements with the leaves analogous to the elements of the interface, internal nodes correspond to groups of elements and the order of the sibling nodes represents their order in the interface. Input to the integrating algorithm

 

consists of the set of query interfaces in the domain being considered and a mapping which characterizes the semantic matching between comparable elements in the interfaces being considered.

Given the set of attributes, groups need to be formed among them for which the following issues are considered: granularity mismatch i.e., 1:m mapping relationships can exist, for example one schema in the airfare domain may specify a single fields for travellers while another may separate fields for adults and children, clusters need to be formed for a domain which model the way in which fields are grouped in the unified interface. Potential groups need to be formed which capture the natural way in which fields are organized in the query interface. The algorithm being developed needs to form partitions over the clusters that occur in the potential groups such that each set of the partition is a group. This is done as follows: for the given set of interfaces all internal nodes apart from the root are identified such that they have at least two adjacent leaves, these adjacent leaves form a potential group, if one of the children is itself an internal node then its children are considered and so on. Sets of potential groups depending on their frequencies form groups, which then form the fields in the unified interface.

The actual merging of the interfaces is performed with two interfaces at a time in order to preserve the ancestor descendent relationships, for which the necessary groups need to be generated in advance. Two schemas are considered at a time, one of which is designated the target schema, for each element in the other schema, determine whether it matches some element in the target schema, if yes they are combined if not this new element is inserted into the target schema such that both ancestor descendant and grouping constraints are satisfied.

*D. Keyword Matching and Similarity Matching*

In this approach [6] two techniques Keyword Matching and Similarity Computing are considered for unified interface construction. Considering two interfaces A and B that need to be integrated to form C. Keyword Matching: for each of the attributes of A and B keyword matching is performed successful matches are put into C. Semantic Similarity: remaining attributes are matched based on similarity. For the representation of the interfaces, the interfaces themselves are classified into three categories: structured query interfaces which have two or more text boxes for querying multiple attributes, semi structured interfaces which have a radio button or a selection list that allow the querying of only one attribute at a time, unstructured query interfaces which have only a single text box. Only structured and semi structured interfaces are considered for integration.

There are two core modules for integrating, these are keyword matching module and semantic similarity computing model. As specified earlier if two interfaces A and B are being integrated into an interface C, then keyword matching is performed between the interfaces A and B, matched attributes are put into C. For those attributes which could not be matched using the above technique, matching is done using semantic similarity computing and attributes with some value greater than a certain threshold are placed in C. Those which could not be matched with either of these methods are directly into C.

Keyword matching involves the following steps: extracting all attributes of query interfaces into a universal set, extracting non verb keywords from the universal set, deleting repeated keywords from the universal set, combining the non verb keywords into a set. Once this is done keywords having the same semantics are classified into one class based on domain knowledge, keywords in each class are sorted based on frequency in the universal set. Keyword matching is essentially string matching, temporary files are created to hold the matched keywords. For all the remaining attributes semantic similarity is computed.

*E. Integration Based on Attribute Constraints*

This approach [7] defines three primary constraints of query interfaces, hierarchical constraints, grouping constraints and precedence constraints. These three constraints of all interfaces should be preserved in the integrated interface. Query interfaces can be represented as attribute constraint matrices and these matrices capture all three kinds of constraints imposed. The matrices are then merged as a precursor to the unified interface. The process of merging is likely to violate precedence constraints, hence score vectors are generate that help preserve precedence. Finally the unified schema can be generated by appropriately manipulating the merged matrix called the attribute constraint such that constraints are not violated. Starting with the ordered tree representation as discussed previously, an attribute constraint matrix is developed; this matrix is capable of capturing all the constraints imposed on the attributes by the interface. This attribute constraint matrix is a symmetric matrix whose rows and columns correspond to leaves of the ordered tree, hence the attributes themselves. The matrix contains integer elements where the values of the diagonal represent the depth of the node from the root, and the remaining values represent the distance between the corresponding attributes. The diagonal specifies the depth of each element from the root and hence captures the hierarchical or ancestor-descendent relationships. Two attributes demonstrate a sibling relationship or grouping constraint if the distance between them is 2. The precedence constraint is represented by the order of the attributes in the matrix. Using these generated matrices it is possible to reverse engineer the original ordered tree.

Once attribute constraint matrices have been generated for all the interfaces that need to be integrated, the process of integration simply involves the merging of these matrices. The first step in this process is to extend each matrix to one that contains all the attributes of all the interfaces being considered without duplication of attributes, the values in the initial matrices are repeated as appropriate; the distance to attributes that don't exist in the specific interface is set to zero. Once these matrices are formed they are merged to form the average attribute constraint matrix. This takes an average over all the individual extended matrices based on the number of non-zero elements. This average matrix is the ideal value of the unified matrix from which the structure of the integrated interface can be developed. Several

 

operations are now performed on this average matrix including the rounding of the decimal values in order to construct the integrated interface.

After the average matrix has been generated, the integration algorithm iterates as follows: the diagonal elements of the average matrix are rounded off and the maximum diagonal value is determined. This maximum value will be the maximum depth of the integrated interface and determines the number of iterations of the algorithm. A set is created containing the row/column numbers of the maximum diagonal elements; these correspond to the attributes at the maximum depth. This set is then divided into several disjoint sets such that the elements in each disjoint set are siblings. If there exist elements in the original set that don't fall into any of the disjoint sets, these elements do not have any siblings which implies that their parents are redundant hence their depth can be reduced by one.

Two different elements in the same disjoint set will have their values in the average matrix set to two since they are siblings, different elements in different disjoint sets will be updated to a value of four, since this is the minimum distance between nodes of the same depth having different parents. Elements not in any disjoint set are rounded to be the current maximum depth plus one, which is the distance to that node from a node that is part of some disjoint set. This process is repeated by reducing the value of maximum depth. Any remaining decimal values are rounded to be as close to the average matrix as possible. This obtained matrix is the unified matrix from which the ordered tree structure of the unified interface can be developed. It has been shown that the attribute constraint matrix representation and the ordered tree representation of the interface are related, i.e., one can be derived from the other. Hence with the developed final unified matrix a final unified tree can be generated from which the exact structure of the final unified interface can be obtained. Score vectors are used to maintain the precedence constraints among the attributes. These score vectors represent the position of the attributes in the considered interfaces, and a score vector for the average matrix is also generated. These score vectors will be used in the process of generation of the unified tree to ensure that the precedence among the attributes is maintained.

## III. CONCLUSION

Due the increasing dependence of the general public on online access to commercial establishments, query interface integration has grown to be of immense importance. Interface integration will help the user to find the best offers through a single access point. Some of the existing techniques for the integration of query interfaces were discussed. Issues such as the representation of the interface, attribute mapping etc., were dealt with in some of the techniques. The most common means of representation of the interfaces was in the form of an ordered tree, which is an efficient means of representation for it can be easily modified to perform integration. There is a need for further enhancement on the existing techniques since no single technique is capable of complete resolution of all the issues. Query interface integration is a promising field of research with possible commercial advantages.

## ACKNOWLEDGEMENT

The authors would like to thank the School of Computer Science and Technology and especially the Department of Computer Science and Engineering, Karunya University, Coimbatore, India for giving them the opportunity to work on this report and providing them with the necessary infrastructure to do so.

## REFERENCES

[1] H. He, W. Meng, C. Yu and Z. Wu, *WISE-Integrator: an automatic integrator of web search interfaces for e-commerce*, VLDB'03 (2003).

[2] H. He, W. Meng, C. Yu and Z. Wu, *Automatic integration of web search interfaces with WISE-Integrator*, The VLDB Journal 13 (2004) 256–273.

[3] H. He, W. Meng, C. Yu and Z. Wu, *WISE-Integrator: a system for extracting and integrating complex web search interfaces of the Deep Web*, VLDB'05 (2005) 1314–1317.

[4] W. Wu, C. Yu and A. Doan, *Merging Interface schemas on the Deep Web via clustering aggregation*,ICDM'05 (2005).

[5] E. Dragut, W. Wu, P. Sistla, C. Yu and W. Meng, *Merging source query interfaces on web databases*, ICDE'06 (2006) 1–10.

[6] F. Yuan, L. Han and Y. Wei, *A Deep Web interface integration approach based on keyword matching and similarity computing*, Journal of Computational Information Systems 6 (2009) 1569–1576.

[7] Y. Li, Y. Wang, P. Jiang and Z. Zhang, *Multi-Objective Optimization Integration of Query Interfaces for the Deep Web based on Attribute Constraints*, Data & Knowledge Engineering 86 (2013) 38-60

 