Deriving Customized Integrated Web Query Interfaces

Eduard C. Dragut #1, Fang Fang #2, Clement Yu #3, Weiyi Meng *4

#*Computer Science Department, University of Illinois at Chicago, USA*
1edragut@cs.uic.edu, 2ffang@cs.uic.edu, 3yu@cs.uic.edu

**Computer Science Department, SUNY at Binghamton, USA*
4meng@cs.binghamton.edu

***Abstract*— Given a set of query interfaces from providers in the same domain (e.g., car rental), the goal is to build *automatically* an integrated interface that makes the access to individual sources transparent to users. Our goal is to allow users to choose their preferred providers. Consequently, the integrated interface should reflect only the query interfaces of these sources. The problem scrutinized in this work is *deriving customized integrated interfaces*. On the hypothesis that query interfaces on the Web are easily understood by ordinary users (*well-designed* assumption), mainly because of the way their attributes are organized (structural property) and named (lexical property), we develop algorithms to construct customized integrated interfaces. Experiments are performed to validate our analytical studies, including a user survey.**

I. Introduction

With each application domain (e.g., real estate) hosting large and increasing number of sources, it is rather unrealistic to expect a user to probe each source individually. One way of enabling users to access these sources uniformly and effectively is the construction of an integrated query interface that acts as a one-stop gateway to disparate relevant sources in the same application domain. Recent research [2], [6], [8], [12] has focused on facilitating users to exploit the content of the numerous web databases as a whole, in contrast to our goal in this work—*allow users to choose the ones relevant to them*. We see this accomplished by *customizing the integrated interface* to a user's needs. Specifically, even though a meta-search engine may cover a large number of providers, give the user the ability to create a customized view that reflects the providers of her choice. We describe a scenario demonstrating the usefulness of such a view in practice.

Companies bundle themselves in groups of interests offering complex services with the goal of tightly coupling their clients with their services so that the clients would not look for alternatives offered by competitors. Marriott Rewards or Holiday Inn's Priority Club are examples of such groups in the hotel industry. These groups run special membership programs (e.g., rewards programs: free nights, miles) to entice customers to use their services. Consequently, many users ignore the low price criterion when it comes to room reservations and prefer to use the same group so that they build up their membership status. This adds a twist to the integration process that the current approaches (regardless of the domain: e.g., query interface, schema integration) [3], [6], [8], [9] do not support: *tailoring on-the-fly the integrated model to the user specific needs*. As an example, if the user wants to search among the partners of Marriott only then the integrated interface should reflect providers like Courtyard, Renaissance, or Fairfield Inn.

After a user specifies the search engines to be integrated, there are two ways to dynamically obtain a customized integrated interface:

1) create it from scratch on demand from the given individual user interfaces, called *bottom-up*,
2) create it by deleting unwanted fields from an existing integrated interface that comprises as many interfaces in the same application domain as possible, called *top-down*.

In the bottom-up approach we can apply any existing integration algorithm [5], [8]; there are a number of disadvantages, though. First, as it will be shown in Section 4, *automatic* labeling of certain attributes of the integrated interface may not be possible for the bottom-up approach, but it may be possible for the top-down approach. Second, building an interface with the bottom-up approach may take even *minutes*, whereas with the top-down approach this can be accomplished in *milliseconds* (see Section 5). Hence, the bottom-up approach is not feasible for real-time applications.

In practice, a hybrid approach should be taken. That is, given an application domain D, we apply the bottom-up approach [5], [6] to construct the integrated user interface for D. The resulting user interface may not be properly designed. Some manual adjustments by a designer may be needed (e.g., adding missing labels to fields). This is a one-time effort. When a user wants a customized user interface for the a subset of interfaces Q in D, the top-down approach is dynamically invoked to produce the desired integrated user interface from the integrated interface of D.

The contributions of this paper are:

- To our knowledge, we are the first to introduce the notion of customizing an integrated user interface and provide a practical way (the hybrid approach) to construct such a user interface.
- We perform experiments on 9 domains comparing the top-down and bottom-up approaches. Experimental results show that the former approach is superior and suggest a hybrid approach is desirable in practice.

The paper is structured as follows. Section 2 discusses related work. Section 3 introduces the main concepts. Section 4 presents the top-down approach. Experimental results are in Section 5. Section 6 is the conclusion.

978-0-7695-3801-3/09 $26.00 © 2009 IEEE
DOI 10.1109/WI-IAT.2009.115

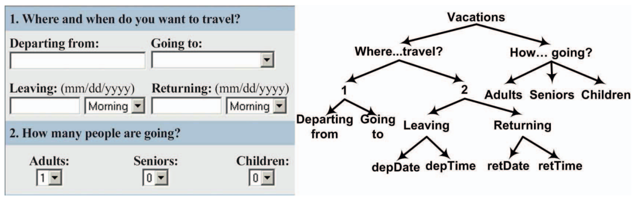

Fig. 1. Example of query interface.

II. Related Work

The goal of recent research projects on Deep Web [2], [6], [7], [8], [10], [12] has been to enable a uniform access to the large amount of data behind query interfaces. The most advocated approach is to undergo integration *domain-wise*. Our work addresses one step among several that need to be completed to make such integration practical. First, query interfaces are identified, extracted from the relevant Web pages [4] and clustered on subject domains [2]. Second, equivalent fields of different query interfaces in the same domain are matched [7], [11]. Third, for each domain a unified interface [5], [6], [8] is constructed. *This is the main focus of the paper*. Fourth, a user's query on the unified interface is translated to queries against interfaces of specific sources [12]. Last, returned data by individual sources needs to be correctly extracted and the results ranked in descending order of desirability (e.g. price).

III. Technical Preliminaries

We now provide the definitions of the structural and lexical properties of query interfaces used in the rest of the paper.

Informally, a **group** [5] is a sequence of adjacent sibling leaves within the integrated interface derived from sets of siblings of the source interfaces. For instance, in Fig. 2, {Company Name, Occupation}, {Name of Company, Time at Company, Business Phone} and {Employer Name, Length of Employment} are sets of adjacent leaves in the Chase, HSBC and NCL, respectively. They lead to the group [Occupation, Company Name, Time at Company, Business Phone] ([ ] denotes a sequence), where all the adjacent leaves in the three sets remain adjacent.

Ancestor-descendant relationships among attributes in source interfaces need to be preserved in the global interface as they convey significant semantic information. For instance, we know that Address (along with its children) mentioned in the interface Chase (Fig. 2) is a company address because Address is a descendant of Employment Information. By preserving this hierarchical relationship within the integrated interface a user would also know that Address refers to the address of a company.

An integrated interface has a **horizonal consistency**, if the labels of leaves in the same group have certain linguistic relationships [5]. For instance, Adults, Seniors, Children are all plurals. An integrated interface has **vertical consistency**, if for any two internal nodes v and w, v an ancestor of w, the label of v is semantically more general than one of w. An integrated interface is **weakly consistently labeled** if it has both vertically and horizontally consistent labeling.

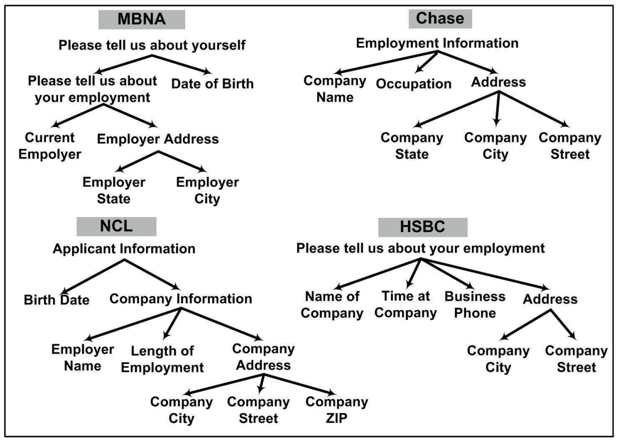

Fig. 2. Example of user interfaces.

The interface is **inconsistently labeled** if it is not weakly consistently labeled.

IV. Deriving Integrated Interfaces

In this section, we establish the following key result: If an integrated query interface can be constructed for a set of interfaces while satisfying the grouping constraint, the ancestor-descendant relationships and the labeling consistencies, then an integrated interface for a subset of interfaces can also be built obeying the two structural constraints and the labeling properties. Thus, customized integrated interfaces can be dynamically obtained in practice. We first analyze the properties of the structural component.

Lemma 1: Suppose an integrated query interface $\mathcal{I}$ for the set of interfaces $\mathcal{QI}$ obeys (1) the grouping constraints and (2) all ancestor-descendant relationships in the individual schema trees. Then for any $Q \subseteq \mathcal{QI}$, $|Q| \geq 2$, there exists an integrated interface obeying the two properties w.r.t Q.

Sketch of Proof: The proof is by construction. We build an integrated interface, called $\mathcal{I}_Q$, for the set of interfaces in Q from the integrated interface of $\mathcal{QI}$, $\mathcal{I}$, and show that this obeys the two structural properties. The construction algorithm has three steps:

1) **Remove** the leaves in $\mathcal{I}$ that do not appear in any schema tree in Q. Also, recursively remove all "fake" leaves. A "fake" leaf is an internal node in $\mathcal{I}$ that becomes a leaf during the removal process.
2) **Collapse** an internal node with its child whenever the child is its sole child. It needs to be applied recursively.
3) **Split up** the groups w.r.t Q that share the same parent node w.r.t QI. Also, split up a group from an isolated leaf whenever they share the same parent node.
   A leaf is isolated if it does not belong to any group and it is not a child of the root in any of the source query interfaces (e.g., node Fa in Fig. 3a) is isolated).

Example 1: To illustrate the algorithm, we create a synthetic example depicted in Fig. 3. The integrated interface is shown on the left. Its groups are {[Da, Db, Dc, Dd, De, Df], [Ea, Eb], [Ha, Hb], [Ga, Gb]}. Suppose the groups w.r.t Q are {[Da, Db], [Dd, De]}. The isolated leaves w.r.t Q are: Df, Hb. Marked in grey are the leaves of the tree that will be removed. After the remove step, the new intermediate

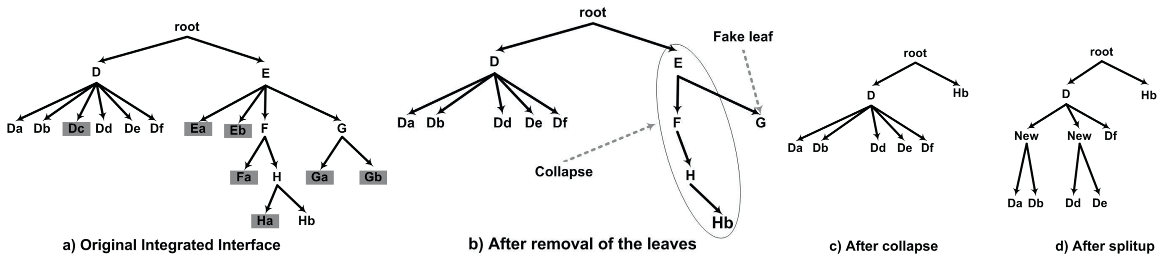

Fig. 3. Example of construction by pruning using Lemma 1.

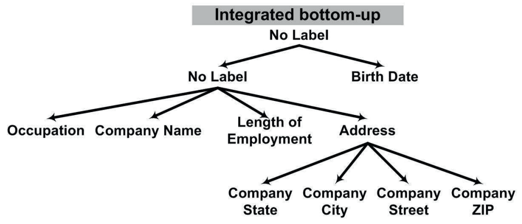

Fig. 4. Example of bottom-up construction.

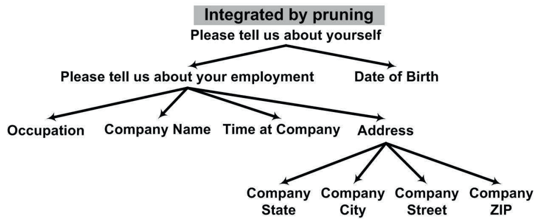

Fig. 5. Example of construction by pruning.

integrated schema tree has only the leaves of the schema trees in Q and the "fake" leaves (Fig. 3b)). All fake leaves need to be removed. Each internal node having only one child needs to be collapsed with its child. The nodes on the path from E to Hb are recursively collapsed. The new intermediate integrated schema tree is shown in Fig. 3c). Since [Da, Db] and [Dd, De] are groups w.r.t. Q, and they share the same parent node, D, they need to be separated. The final integrated interface is shown in Fig. 3d). Note that [Da, Db, Dd, De, Df] may be left as children of the same parent as the larger group was validated w.r.t. the original integrated interface.

The construction algorithm in the proof of the lemma gives us a useful tool in practice. One scenario is when an integrated query interface for a subset of interfaces from a predefined set of interfaces needs to be generated upon user request.

Construction by pruning: We investigate the issue of constructing customized integrated query interfaces by pruning from an existing one, rather than bottom-up. Using the example in Fig. 2 we show why the automatic labeling of certain attributes in an integrated interface may not be possible for a given set of query interfaces. Suppose the application domain consists of two interfaces: Chase and NCL (Fig. 2). We are required to construct a unified interface for them. We will get the interface called Integrated bottom-up (Fig. 4), which does not have labels for its internal nodes. For instance, for the parent node of the field Occupation we have only two choices of labels: Company Information and Employment Information. To choose any of them we need to show that they *semantically cover* [6] the following set of descendant leaves {Occupation, Company Name, Length of Employment, Company State, Company City, Company Street, Company ZIP}. The former label semantically covers all the leaves in the set but Occupation and the latter label semantically covers all the leaves in the set but Length of Employment. The knowledge from the three interfaces is not sufficient (there is no lexical or ancestor-descendant relationship between the two labels) to show that at least one of the labels covers both fields.

Assume now that the application domain is larger, having all four interfaces in Fig. 2, and we are required to derive on-the-fly a unified interface for the subset of interfaces Chase and NCL. With the top-down approach we first build the integrated interface for all four interfaces and, on demand, we derive (extract) from it the unified interface for the subset of interfaces at hand. The construction algorithm is presented in the proof of Lemma 1. The unified interface obtained for the two interfaces with this technique is depicted in Fig. 5. It is obtained by pruning out all elements of MBNA and HSBC that do not appear in the other interfaces. This has better properties than the previous interface, shown in Fig. 4. For example, it has labels for the internal nodes while the other does not.

Proposition 1: Suppose there exists an integrated query interface for $\mathcal{QI}$ that (1) satisfies the structural constraints in the individual schema trees (i.e. grouping and ancestor-descendant) and (2) it has a *(weakly) consistent solution*. Then for any $Q \subseteq \mathcal{QI}$ there exists an integrated interface obeying the structural constraints w.r.t Q and it has a *(weakly) consistent solution* w.r.t $\mathcal{QI}$.

The consistency is validated against $\mathcal{QI}$ and not against Q. This is an acceptable solution given that a consistent solution cannot always be reached by employing only the interfaces in Q. Although the vocabulary increases from Q to $\mathcal{QI}$ it is still within the same application domain and, therefore, the current words remain unambiguous. The labeling can be refined, however, once the pruning is completed since at each internal node of the unified interface of $\mathcal{QI}$ we keep track of all the candidate labels. Therefore, labels that belong to the interfaces of Q can be used.

V. Experiments

We describe a set of experiments that validate the following claims: (1) the integrated interfaces generated by the top-down approach are qualitatively superior to those generated by the bottom-up approach and (2) the top-down algorithm produces customized integrated interfaces on-the-fly. Thus, in practice the bottom-up approach is used to assist a designer to construct

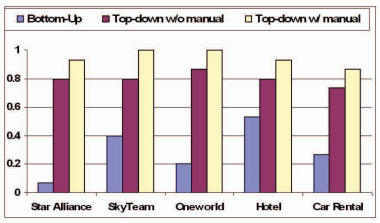

Fig. 6. Easiness

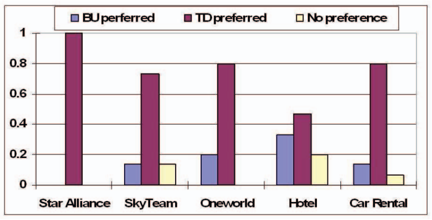

Fig. 7. Preference w/o manual

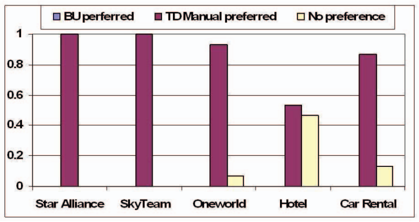

Fig. 8. Preference w/ manual

an integrated interface for an application domain and the top-down approach is employed to dynamically generate the customized integrated user interface for a subset of interfaces in the domain. The bottom-up construction is accomplished with the merge and labeling algorithms depicted in [5], [6].

The testing data set consists of 220 sources over 9 real-world domains on the Web. Two experiments are performed: one to evaluate the efficiencies of the algorithms and the other to evaluate their qualities.

Efficiency: The pruning algorithm generates a unified interface almost instantaneously (average time over 450 runs (50 runs per domain) is 1.6 milliseconds, with the standard deviation of 1 millisecond) while the bottom-up algorithm takes on average 1.8 minutes (the standard deviation is 3.4 minutes) to generate a unified interface. In some domains, the bottom-up may even take between 5 and 10 minutes (e.g., Alliances, Credit Card). This supports our *on-the-fly* statement regarding the top-down customization. The difference is that the bottom-up approach has to compute both the integrated interface and the labels of all the nodes from scratch while the top-down approach does not (Lemma 1 & Proposition 1).

Quality (User Survey): We undertook a user survey to evaluate three algorithms: bottom-up, top-down *with* and *without* manual intervention. 15 people participated. In the top-down approach with manual intervention, the integrated interface of D was modified manually before the pruning algorithm was applied, while in the top-down approach with manual intervention, it was not. The survey was designed as follows. In Car Rental and Hotel, we randomly picked 5 out of 20 and 10 out of 30 source interfaces, respectively. In the Alliances domain, for each alliance, i.e., Star Alliance, SkyTeam and Oneworld, we constructed its unified interface by applying bottom-up and top-down algorithms. This domain has 50 interfaces. There are 5 subsets of query interfaces—one in Hotel, one in Car Rental and three in Alliances. For each subset three integrated interfaces were created with the three algorithms. In total, 15 unified interfaces were used for the survey. They were deployed online [1] and the users were asked the following simple questions. First, each of the 15 interfaces was shown individually and users were asked: "Is the interface easily understood?". The graph in Fig. 6 shows the outcome to this question. The bar on the left represents the percentage of people that found the unified interface constructed bottom-up easy to understand. The bars in the middle and on the right represent similar percentages for the interfaces derived by the top-down approach *without* and *with* manual intervention, respectively. The graph shows an overwhelming easiness of understanding an interface derived top-down over bottom-up, regardless of the manual intervention.

Second, for each of the five scenarios the integrated interfaces constructed bottom-up and top-down *without* manual intervention were shown side by side and the user was asked: "Which one do you prefer?". There are three possible answers: *prefer left*, *prefer right* or *no preference*, where *left* and *right* represent bottom-up and top-down, respectively. The users are not aware what *left* and *right* denote. Fig. 7 summarizes the results. From left to right, the bars represent the percentage of people preferring bottom-up, preferring top-down and with no preference, respectively. The interfaces derived top-down are significantly preferred over the ones constructed bottom-up. Third, we performed the same experiment for bottom-up and the top-down *with* manual intervention algorithms. The outcome of the experiment is shown in Fig. 8. The salient observation is that no user expressed a definite preference for the interfaces created by the bottom-up approach.

VI. Conclusions

The customization (derivation) of integrated query interfaces is the central problem tackled in this paper. We argued that: *The top-down approach yields qualitatively better integrated user interfaces than the bottom-up approach*. We suggest using a hybrid approach in practice, where the bottom-up approach is used to assist a designer to construct an integrated interface for a given application domain and then the top-down approach is employed to produce automatically a customized interfaces.

Acknowledgements: This work is supported in part by the following NSF grants: IIS-0414939 and IIS-0414981.

References

[1] http://www.cs.uic.edu/∼edragut/QIProject.html.
[2] L. Barbosa, J. Freire, and A. Silva. Organizing hidden-web databases by clustering visible web documents. In *ICDE*, 2007.
[3] P. Buneman, S. Davidson, and A. Kosky. Theoretical aspects of schema merging. In *EDBT*, 1992.
[4] E. Dragut, T. Kabisch, C. Yu, and U. Leser. A hierarchical approach to model web query interfaces for web source integration. In *VLDB*, 2009.
[5] E. Dragut, W. Wu, P. Sistla, C. Yu, and W. Meng. Merging source query interfaces on web databases. In *ICDE*, 2006.
[6] E. Dragut, C. Yu, and W. Meng. Meaningful labeling of integrated interfaces. In *VLDB*, 2006.
[7] B. He and K. Chang. Statistical schema matching across web query interfaces. In *SIGMOD*, 2003.
[8] H. He, W. Meng, C. Yu, and Z. Wu. WISE-integrator: An automatic integrator of Web search interfaces for e-commerce. In *VLDB*, 2003.
[9] R. Pottinger and P. Bernstein. Merging Models Based on Given Correspondences. In *VLDB*, 2003.
[10] J. Wang, J.-R. Wen, F. H. Lochovsky, and W.-Y. Ma. Instance-based schema matching for web databases by domain-specific query probing. In *VLDB*, 2004.
[11] W. Wu, A. Doan, and C. Yu. Webiq: Learning from the web to match query interfaces on the deep web. In *ICDE*, 2006.
[12] Z. Zhang, B. He, and K. Chang. Light-weight domain-based form assistant: querying web databases on the fly. In *VLDB*, 2005.