Discovering and Resolving User Intent in Heterogeneous Databases

Chris Fernandes[1] and Lawrence Henschen[2]

[1] Department of Computer Science,
Colby College, Waterville, ME 04901
[2] Department of Electrical and Computer Engineering,
Northwestern University, Evanston, IL 60201
henschen@ece.northwestern.edu

Abstract. We propose a system whereby subtle semantic ambiguity found in queries of distributed heterogeneous database systems can be resolved by considering the user's intentions. Through the use of domain-specific knowledge embedded within a mediator-based architecture, subtleties in meaning can be explicitly modeled. Through the use of dynamic profiles and active dialogue, the system can discover user intent, providing more satisfying query answers.

1 Introduction and Problem Statement

Modern heterogeneous database systems generally require users to issue queries via a global query language. This is because, typically, no one member database of the distributed system has all the concepts of the whole system. Moreover, two identical terms in two different local schemas may have slight semantic differences in the context of the entire network. To avoid the ambiguity caused by this heterogeneity, many systems have a global language containing a vocabulary of terms with exact pre-specified meanings. Wrappers are then used to translate global terms into their meanings within a local schema's context.

This global approach places excess burden on the users to first understand the semantic differences that terms may have in different databases and then to very carefully express queries to precisely reflect their desired semantics. First, note that global concepts are often expressed as generalizations that are broad enough to cover the varying meanings of that concept at the local levels. The user has to be aware of the different nuances that terms might have in the other local repositories and add appropriate constraints to be sure his/her intent is specified. Second, as new databases join the global network, existing terms may take on even more nuances, new terms and concepts may enter the global language, and perhaps even the global schema itself may change. Users will then need to become familiar with the changes, and so user training in a dynamic network is not a one-time affair like many researchers claim. Similarly, applications that had been written for the local database and been modified to adapt to the network at a particular point in time may also need additional updating as the network changes. For the same reason, commonly run queries that were written and saved under an older incarnation of the global schema may also need to be updated.

The opposite approach allows users to specify queries in their own local database language and provides translators to map those queries to a global form that reflects the semantics of that local database. This allows users to issue distributed queries and to correctly interpret answers without having to learn a new language and be continually retrained. This would be a distinct advantage for non-sophisticated users, and it is reasonable to expect a large number of database users to fit into this category. For example, a distributed information system might include car rental companies and meteorological databases, both of which use the term “map” with related but distinct meanings. A counter agent at a car rental location may never care about seeing meteorological maps of the region, only street maps to give to customers renting cars. Such counter agents are probably not database experts and should not be required to know about all the other kinds of maps that may be available and how to specify just street maps when issuing a query. On the other hand, the local-query approach forces any user who wants to take advantage of the variety of related information to again learn the global language and continuously keep up to date as the network evolves. A travel planner in the “maps” example may very well want one or both kinds of maps when planning a group tour to account for both the actual transportation as well as possible weather-related contingencies. Even more, the planner may issue the same query, for example “get a map of the Boston area”, at different times and want different kinds of maps based on what aspect of the tour is being planned at that moment. But, again, travel planners are not likely to be database experts, and it would be advantageous to develop some kind of system that would help such a user cope with a semantically diverse and evolving distributed knowledge base.

We will describe in this paper a system that attempts to merge the best features of each approach and to help users of both kinds. Queries are expressed in local database languages, and the query-processing algorithm uses the query and a global knowledge base in combination with information from an individual user profile and even user dialogue to translate the local query into a global one. Through the use of the profile and dialogue, our system tries to discover the real intent of the user query. In addition, our system attempts to help the user specify the intent when it can’t be guessed or when the user has requested help. The development and use of individual user profiles and the nature of the dialogue are the primary foci of this paper. They are described in Section 3. In Section 2, we differentiate our work from other research in this area, but space limitations preclude any extensive discussion of prior and related work. Section 5 presents concluding remarks and the direction we would like to take in the future.

2 Related Work

We have chosen a mediator-based system for our implementation since it allows for the two most important criteria we desire: the ability to ask queries through local member databases and the presence of a global knowledge base to perform translation on a query-by-query basis. Since intent can change over time, having this latter property is critical. Many other prototypes are also mediated systems, but most do not consider the idea of intent as we have stated it.

Most mainstream prototypes use a single global language to represent queries. These include rule-based languages such as the one used by the HERMES system [7], description logic-based languages such as Loom used by SIMS [1], clause-based languages such as that used by Information Manifold [4], and others such as OQL used by DISCO [8]. Using these global languages provides certain advantages. HERMES, for example, is able to incorporate a degree of probability into its rules, while Information Manifold's use of logical clauses provides a robust translation from its global language into local subqueries. However, they all prevent the user from using a local language with which s/he may already be familiar, and they all prevent the use of applications which already utilize that local language.

A means of adjusting to different user intents is another aspect found in only a few prototype systems. One such system, TSIMMIS [2], is web-based and allows for a limited type of intent clarification by including hypertext in a query result. By this hypertext the user can see more general or more specific information concerning the objects involved in the query answer. This could be extended so that the user could reissue a query after specifying a more appropriate level of object detail or after delineating ambiguous concepts. We propose a dialogue system which would clearly define user intent on a level not seen in these prototypes.

Another important aspect which we include in our system, but which others do not, is conditional attribute equivalence. Essentially, this term means that any two attributes which are considered to be equivalent or at least synonymous for one query may not be considered equivalent or synonymous for another. Consider a distributed system dealing with universities, where one local database keeps track of an undergraduate-only institution while the other holds data of a university with graduates and undergraduates. Two "student" attributes in these two local databases may be considered equivalent for a query dealing with counting up all students but quite dissimilar for a query dealing solely with graduate students. Existing systems such as OBSERVER [5] may use a single ontological term to relate a priori to multiple local attributes. As a result, there is no easy way for these equivalences to change from query to query. Our approach will use constructs called annotations [3, 6] to accomplish conditional attribute equivalence. These annotations will provide for query-dependent processing, which we believe is essential for determining user intent.

3 Discovering a User's Intentions

Our system discovers different possible query interpretations at run-time. Due to space limitations we will not explain our query processing system in detail; the interested reader is referred to [3, 6]. When necessary, we will explain those

Person		
Name	Phone	State
Fred	111-1111	VA
Sally	222-2222	IL
Tom	333-3333	HI

```
SELECT name, phone
FROM Person
WHERE state="IL"
```

Fig. 1. Sample query in an Employee Database

features of the query processor that intersect with our techniques for determining the user's true intentions.

3.1 A Motivating Example

We first illustrate that subtle heterogeneity can creep into even the most innocuous of queries. Figure 1 shows a table from an employee database showing each worker's name, phone number, and place of residence. The accompanying query is asking for the names and phone numbers of those employees residing in IL. While the semantics for such a query would be clear for any one relational database, it becomes more complex if this is a distributed query over many employee tables. Figure 2 shows two local databases containing identically structured employee tables, but differing in their interpretation of the **phone** attribute. In DB1, the phones are all cellular phones. They do not have a permanent location, and they are probably kept with the employee most of the time. However, in DB2, the phones are regular permanent phones, located most likely at the employee's place of residence. This difference is not reflected in either of the two tables, and it produces two different query interpretations. Did the user specify "IL" because s/he intended to call there, perhaps looking for the employee's family? Or did the user intend to contact *people* based in IL, regardless of where their phones are? One's initial solution in a small example like this might be to return both sets of phone numbers so that the user could decide. But this solution is not feasible in general, especially where scalability and time are factors. In an emergency situation such as a plane crash, passenger manifests need to be used to notify family members quickly. In this scenario, permanent phone numbers would be desired, not cell phone numbers. Having the system return hundreds of extraneous tuples that the user must sift through is not practical.

3.2 Finding Ambiguities in Intent

In order to choose the desired interpretations of a given query, a system must be able to explicitly model subtle semantic differences like the one given here.

DB1: cellular phones		
Person		
Name	Phone	State
Fred	111-1111	VA
Sally	222-2222	IL
Tom	333-3333	HI

DB2: regular phones		
Empl		
Name	Regphone	Place
Sally	444-4444	IL
Carl	555-5555	IL
Beth	666-6666	VA

Fig. 2. Subtle Heterogeneity in Logically Identical Tables

The ambiguity in the phone example stems from the fact that there are distinct relationships between the **phone** and **state/place** attributes in DB1 and DB2. We have developed a graph-based mediator [3, 6] that precisely and explicitly maps out concepts and relationships at the global level and relates these to the concepts and relationships in the individual local databases. Although the global representation and the related algorithms are not the focus of this paper (see [3, 6] for full discussions of these), we illustrate briefly how our system models semantic variations.

Figure 3 shows a portion of the mediator for our phone example. Vertices represent concepts, while edges represent relationships. Knowledge is embedded in the mediator in the form of **annotations**. An annotation is an ordered pair $\{x, y\}$ where x and y are local database attributes. If an annotation $\{x, y\}$ is associated with a relationship R, then R **must** hold between local attributes x and y. Thus, the annotation $\{DB2.empl.regphone, DB2.empl.place\}$ shown in Figure 3 means that the relationship *rings* must hold between the **Phone** and **Location** concepts in local database DB2. That is, for DB2, the phone must ring in the specified place. The lack of a similar annotation for DB1 for the *rings* relationship means that *rings* does not hold or is unknown in DB1. When analyzing this section of the graph during query processing, our algorithm can use the presence or absence of certain annotations to detect possible differences in intent.

3.3 Resolving Ambiguities in Intent

Once the source of competing interpretations has been found, the system can then move to the task of resolving the ambiguity. We have developed two methods for determining what the user truly intended: individual user profiles and on-the-fly dialogue.

A **user profile** is a list of preconceived notions that the user has about what local query concepts mean. A profile may contain domain-related information, e.g. the concept *student* means just undergraduates. It may also contain general query processing preferences, e.g. to exclude a particular local database from

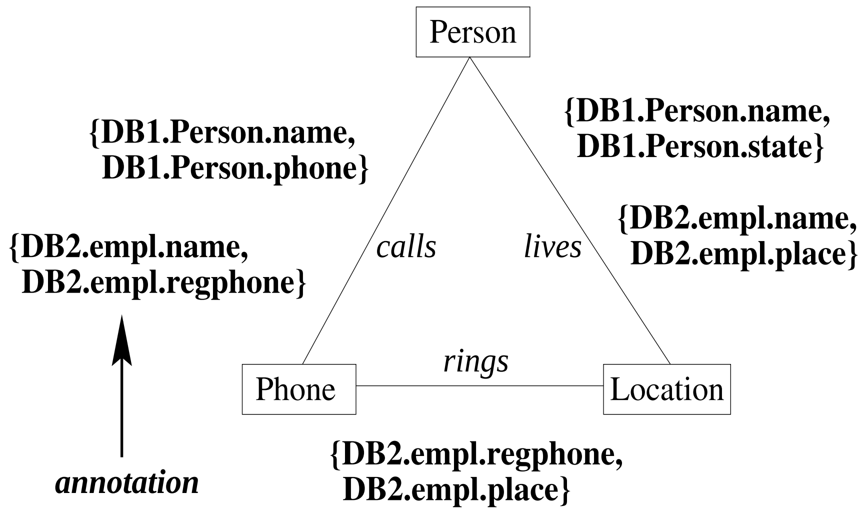

Fig. 3. Section of Mediator used for Phone Example

contributing answers or to allow separate databases to participate in providing partial answers that can be combined into a complete whole (i.e. inter-database joins.)

A profile can be created and modified before query run-time. In the system we are developing, a profile is initially created when a new user logs into the system for the first time. At that point, the user is asked a series of questions about how s/he interprets certain local concepts. A screenshot in our system for this process is shown in Figure 4. The result is a personal knowledge base that the query processor can consult to help determine a user's intent. In our phone example, the query processor could examine this profile when it considered the *rings* relationship shown in Figure 3. If the user had indicated that only non-mobile phones are of interest, then DB1, containing only cell phones, would be eliminated from the search space.

Profiles need not exist only for individual users, however. Profiles for specific classes of individuals could also be created. For example, all incoming business school students at a university or all new secretaries in a corporation could have a class profile that would ensure a uniform query vocabulary. The fact that a local query language is being used would provide a foundational interpretation of terms for creating such a class profile. Similarly, a profile could be set up for an application, extending its access from the original database to a distributed federation of databases. Moreover, as the global schema changes in response to the addition or deletion of member databases, administrators could modify these class profiles to keep end users and applications up-to-date. These changes would be transparent to non-sophisticated users while still allowing more advanced users to tweak their own profiles if they so desire.

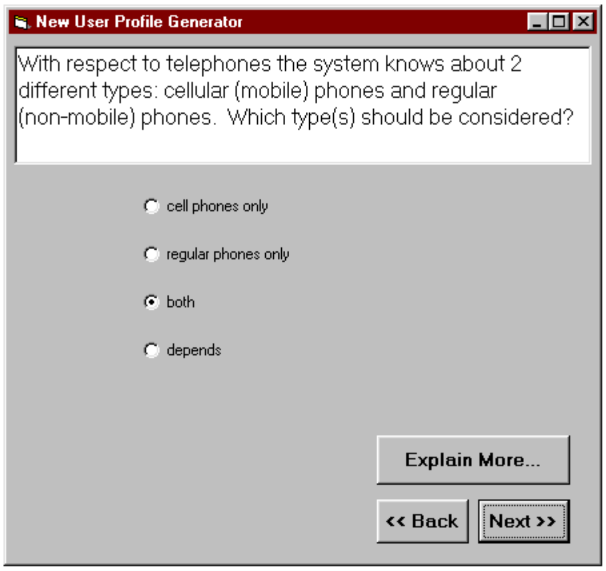

Fig. 4. Creating a User Profile

In addition to a profile generator, we have also implemented a DBA toolkit by which profile generators can be created quickly. This toolkit is illustrated in Figure 5. We envision that both local and global database administrators are in the best position for understanding the subtle semantic differences between the syntactically identical concepts across individual member databases. The administrators can then utilize this tool to form the questions which will make up profile generators. These generators can be tailored by each local DBA so that questions are best expressed to match the level of user expertise and "lingo" in use at that local site.

A query-independent mechanism like a user profile is not sufficient by itself, however. A user may be aware that his/her interpretation of a term may change depending on the query asked. The profile generator accounts for this by allowing the user to pick a "depends" option as shown in Figure 4. This tells the system that when a particular concept is used in a query, its meaning will need to be determined at run-time. In this way, a user would not need to continually update his/her profile just because s/he wishes to execute a different query. Instead the "depends" option signals the query processor to ask the user (say, via a set of choices in a dialogue box) for the user's interpretation of the concept for this query. This type of dialogue is called **on-the-fly dialogue**.

Consider the mediator subset in Figure 6 dealing with an ecommerce network. Here a single member database, DB3, allows two possible interpretations for queries dealing with buyers. The Buyer concept can either represent those who have actually made purchases of a given product (kept in the *Product* table of DB3), or it can represent those who are just in the market to buy said product

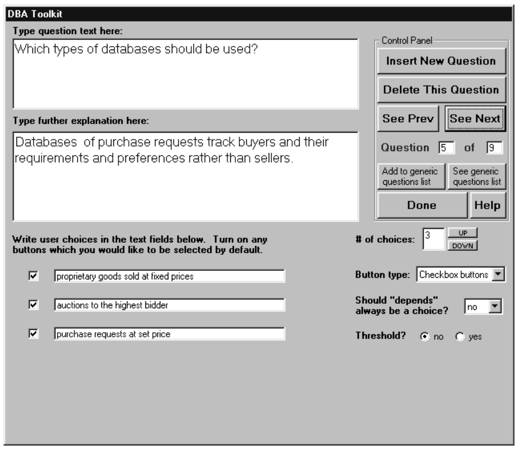

Fig. 5. DBA Toolkit for Creating Profile Generators

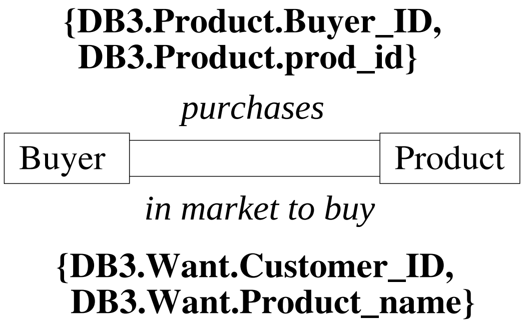

Fig. 6. Subset of Mediator used in an Ecommerce Network

(kept in the *Want* table of DB3.) Such a distinction would be made, for example, in a name-your-own-price website such as *priceline.com.* During processing of a query dealing with buyers, the mediator would detect the two different relationships possible within DB3 and consult the user's profile to find a preference. If one is not found (if the user chose "depends" as an answer, for example) then the user would need to be asked to specify which relationship s/he wanted. Tuples from the correct table in DB3 can then be retrieved.

The gravest potential pitfall of on-the-fly dialogue is its overuse. Since it is an interruption during query processing, we wish to eliminate unnecessary interaction. We have several ways by which our system can deduce a correct interpretation. First, the profiles of other users of the same local database can be used as a guide. If most users interpret a term in the same way, the system could prevent an interruption by assuming the same interpretation. Second, the

DB1: Dan's Good & Services

```
Transaction(PURCHASE #, date, buyer
           product_code, quantity)

Product(PRODUCT_CODE, product_name,
       description, amt_in_stock, unit_price)
```

DB2: Al's Auctions

```
Product(PRODUCT_ID, description
       opening_bid)

Auction(AUCTION_ID, seller_id, product_id,
       high_bid, buyer_id, closing_time, num_bids)

Seller(SELLER_ID, seller_name, seller_email)

Buyer(BUYER_ID, buyer_name, buyer_email,
     current_bid, product_id)
```

DB3: Sam's Shopbot Site

```
Buyer(SAM_ID#, name, email)

Want(SAM_ID#, PRODUCT_NAME, PRICE)

Product(PROD_ID, SAM_ID#, product_name, seller,
       actual_price, qty, time_to_find)
```

Fig. 7. Schemas Used in Experimental Ecommerce Network (keys are in CAPS)

result of an on-the-fly dialogue, no matter what the user answers, may end up being moot. In our phone example, a local database may be eliminated from the search space depending on which phone type the user specified in the dialogue. If that local repository has already been eliminated due to other query constraints, then asking the user for clarification is fruitless. Third, by using the log data of past queries, it is possible to detect when the result of an on-the-fly dialogue would only result in an insignificant increase in the number of returned tuples. In this situation, the interpretation which produced the greater number of answers could be assumed with insignificant effect on processing time or user time when sorting through the results. In general, on-the-fly dialogue could be bypassed if the effect of "guessing wrong" was negligible.

4 Preliminary Results and Future Work

We have already implemented our profile generators, DBA toolkit for making generators, and our annotated mediator. We have also completed preliminary experiments using concepts and queries from an ecommerce domain. In these experiments, both novice and expert computer users issued queries to a set of three databases that each performed on-line buying and selling in a different way. The first sold goods and services at fixed prices, the second auctioned off goods to the highest bidder, and the third was a shopbot-based database that worked in a similar fashion to *priceline.com*. Their schemas are shown in Figure 7.

Table 1. Results from Experiment

Query	User Level	Number of Matches	Number of Mismatches	Probability of Independence
Query 1	novice	3	23	.0008
	expert	0	7	
Query 2	novice	12	14	.7001
	expert	3	4	
Query 3	novice	15	11	.3296
	expert	5	2	
Query 4	novice	9	17	.2538
	expert	4	3	
Query 5	novice	16	10	.1154
	expert	6	1	

Users were first asked to create a profile which recorded their own opinions concerning common terms used in this domain. For example, a user was asked to define the term "Buyer" given the two interpretations presented previously in Section 3.3 and shown in Figure 6. Then, each user was shown the same set of test queries and asked to profess what they considered to be "correct" answers based on their profiles and responses to on-the-fly dialogue. The users then compared the answers returned by the system to the answers they thought would be "correct" to see if the system could properly discern their interpretation of the query. Chi-Square analysis was performed on the raw data to test for a relationship between the two user groups. The results, displayed in Table 1, show the number of users whose query interpretations matched those of the system alongside those whose interpretations did not match.

Several conclusions can be drawn from the data. The low number of matches for the first query is not surprising since the idea of multiple query interpretations is a new one to most users. This query represents a training period for the user. As time went on, the user's interpretation of the query was correctly reflected by the system's interpretation with increasing accuracy. The continually-reducing probability of independence between the two user groups indicates that expert users tended to become adept at using the system in a shorter amount of time than novices. Because non-experts are the target audience for a system of this type, we hope in the future to adjust the wording of questions in the profile generator and to provide a more robust training period using more sample queries. It is hoped that these will provide a ramp to proficiency that is neither too steep nor too long for either type of user.

5 Conclusion

We believe that the discovery of intent in queries is an important aspect in determining if returned answers are indeed "correct" or not. If a system returns tuples based on its own static interpretation of query terms, regardless of how broadly encompassing those terms may be, the lack of consideration for the user's expectations will always leave room for misinterpretation.

We have developed and implemented a system that attempts to take the user's world assumptions into account by using profiles and on-the-fly dialogue to guide the query translation process. Using preferences specified both before and during run-time, a single query can return different sets of answers that correspond to what different users had in mind. Preliminary empirical evidence is favorable, though we hope to do more experiments to increase proficiency and decrease training time.

References

1. Arens, Hsu, and Knoblock. Query processing in the SIMS information mediiator. In Austin Tate, editor, *Advanced Planning Technology*, Menlo Park, CA, 1996. AAAI Press.
2. Garcia-Molina, Papakonstantinou, Quass, Rajaraman, Sagiv, Ullman, and Widom. The TSIMMIS approach to mediation: Data models and languages (extended abstract). *JIIS*, 1997.
3. Henschen, Neild, and Fernandes. An object-oriented graph traversal algorithm for data mediation. In *Proceedings of the 2nd Americas Conference on Information Systems (AIS96)*, Phoenix, AZ, August 1996.
4. Levy, Rajaraman, and Ordille. Query-answering algorithms for information agents. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence (AAAI-96)*, Portland, OR, December 1996.
5. Mena, Kashyap, Sheth, and Illarramendi. OBSERVER: An approach for query processing in global information systems based on interoperation across pre-existing ontologies. In *COOPIS*, Brussels, Belgium, June 1996.
6. Tania Neild. *The Virtual Data Integrator: An Object-Oriented Mediator for Heterogeeous Database Integration.* PhD thesis, Northwestern University, June 1999.
7. Subrahmanian, Adah, Brink, Emery, Lu, Rajput, Rogers, Ross, and Ward. HERMES: A heterogeneous reasoning and mediator system. submitted for publication.
8. Tomasic, Raschid, and Valduriez. Scaling heterogeneous databases and the design of DISCO. In *Proceedings of the International Conference on Distributed Computer Systems*, 1996.