Modeling Data Correspondence in Multidatabase Systems

Lawrence Henschen
Northwestern Univ.
Dept. of ECE
Evanston, IL 60208
Ph: 847-491-3338
Fax: 847-491-4455

Chris Fernandes
Northwestern Univ.
Dept. of ECE
Evanston, IL 60208
Ph: 847-467-2299
Fax: 847-491-4455

Tania Neild
Northwestern Univ.
Dept. of ECE
Evanston, IL 60208
Ph: 847-467-2299
Fax: 847-491-4455

Wen-Syan Li
Rutgers Univ.
Dept. of CIMIC
Newark, NJ 07102
Ph: 201-648-5026
Fax: 201-648-5003

Abstract

An essential part of a multidatabase system is a model which is able to incorporate different database schemas. This model must be capable of relating the concepts within the shared local data. The relationships which may be included may be as straight-forward as a syntax conversion or as complex as conditional attribute equivalence. Moreover, the model should allow queries to be done at the local level while still being guided at the global level.

This paper presents an object oriented graph as one such model. The local database concepts and the object oriented graph concepts are related/linked by annotated edges which are used to algorithmically transform one local query into a series of other local subqueries. Thus, this integration method provides a method for modeling data correspondences across a variety of data models.

Keywords: multidatabase, knowledge-based systems, database management systems

1 Introduction

1.1 Background

Applications in a wide variety of industries require access to multiple heterogeneous databases due to company mergers, the introduction of new database technology, or integrating information across departments. There are two ways to integrate existing information systems: re-engineering and interoperation. In re-engineering, the application logic, data definition, and data from the old systems are transferred into a new system. The advantage of re-engineering is that the new systems are easier to maintain. However, this process is expensive and complicated. The second option is to allow the existing systems to interoperate as a multidatabase system by putting a new, standard interface on existing databases. This preserves the data and applications from the existing databases, yet allows access to the data from new multidatabase applications. A main advantage of this is that query processing by the shared-resource system continues to look to the user as if one homogeneous database were being accessed.

When the user issues a query to a multidatabase system, it is transparently disassembled and reformulated into subqueries that the individual local schemas can interpret. Each local schema can then retrieve answers to its own subquery. The answers to the subqueries are then integrated into the answer for the user's original query. The process by which queries are taken apart and answers put back together is the focus of this research and depends on making accurate connections between data in the different local databases. Much effort has been undertaken to ensure that correspondences between the local data is determined accurately, whether this process be done manually or semiautomatically ([2], [3], and [4]).

1.2 Problem Statement

We believe that data correspondence is a primary bottleneck in integration and query processing in multidatabases. A naming conflict between semantically related objects (such as origin and destination city) may cause queries involving those objects to possibly return incor-

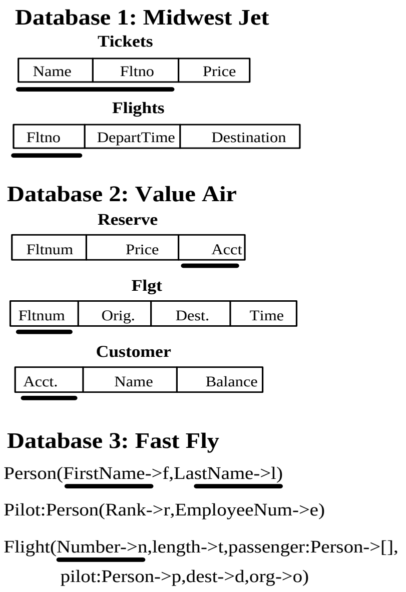

Figure 1: Airline Multidatabase System

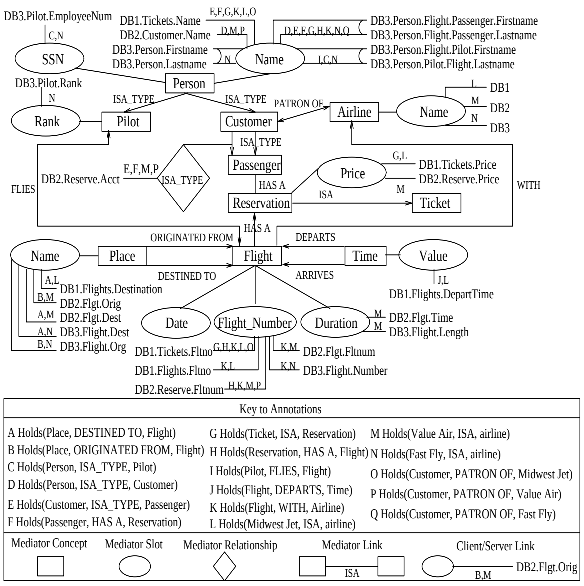

Figure 2: The Mediator

rect answers. Thus, techniques such as extracting semantics, transforming formats, and resolving heterogeneity are essential to intelligent information integration. In this paper, we present an approach for modeling data correspondences for logical database integration. It is robust enough to solve many of the problems associated with interoperability such as naming conflicts, structural and scaling differences, non-empty symmetric differences in data content, differences in scope or level of abstraction, and differing data models (e.g. object-oriented vs. relational.)

2 Model Description

Our model consists of two components. First, we introduce a *mediator* [6], an object oriented graph, which is used to arbitrate between the local schemas by deciding upon the subqueries. Second, we introduce annotated links which relate the local and mediator concepts. For now, we assume the mediator and its links are made *a priori* when each individual schema was admitted to the system.

To illustrate the function of the mediator, a fictitious airline multidatabase system in Figure 1 is used. Three local airline databases comprise this system. Databases 1 and 2 (DB1 and DB2) are relational databases. Database 3 (DB3) is an F-logic database. The underlined attributes in each schema represents the primary key(s) of that local schema. Note that DB2 contains a unique customer account number, *acct*. Additionally, DB2.Customer.Balance represents the amount of money the customer owes the airline. The other attributes are self-explanatory.

2.1 The Mediator

The first component of our schema is a means by which a distributed system can be seen as a logically integrated database. To accomplish this, we create a mediator which is an object oriented graph that stores global knowledge about the distributed system. The mediator for our airline system is shown in Figure 2. The rectangular, oval, and diamond shapes denote mediator objects, slots, and relationships, respectively.

The granularity of the mediator can be as fine or course as desired. For instance, there is no mediator concept corresponding to local concept in DB2.Customer.Balance, and therefore, the DB2.Customer.Balance data is not shared. The granularity of the mediator concept allows for increased management of security and complexity.

2.2 The Links

There are directed links between the mediator objects and slots, between the mediator concepts, and between the mediator slots and local concepts. The local concepts are not always attributes. For example, the Figure 2 link between DB1 and mediator slot Airline.Name represents the knowledge that DB1 holds airline information for the Midwest Jet Company.

All of the links are annotated with the relation by which the nodes at the ends are connected (ISA, HASA, etc.) These annotations have the form Holds(x, y, z) where the relation y holds between x and z. Additionally, the links connecting to the local concepts may contain syntactic formatting and symbolic mappings. For example, the concatenation of DB3.Person's Firstname and Lastname is shown using an arc symbol on the joint link to the mediator's slot Person.Name.

While links into our mediator graph represent knowledge between individual tables, that may not be enough to correctly process a query if attributes within different tables are not entirely equivalent, yet related. In our example, DB1.Flights.Destination represents a city where a Midwest Jet flies. This is related to, but no the same as, DB3.Flight.Orig which is a city where a Fast Fly flight originated. If we wanted to perform a query concerning all cities that have airports, then the distinction between origin and destination is not necessary, and the two attributes are equally relevant. On the other hand, if a query concerning the cities to which a passenger may fly was issued, then DB3.Flight.Org is not included. This information represents general knowledge about cities and airports.

The annotations provide knowledge and additional constraints on how queries concerning nodes at the ends of the links should be processed. Together, the links and their annotations are used to connect the queried concepts. When a query is issued, the links in which the annotations do not result in any inconsistencies are traversed. The link traversal finds all query paths which may produce results. If the knowledge is not needed, the constraint will simply not be used. The formal syntax of the annotation is not addressed here; these annotations represent the functionality and descriptions which must be incorporated into the model.

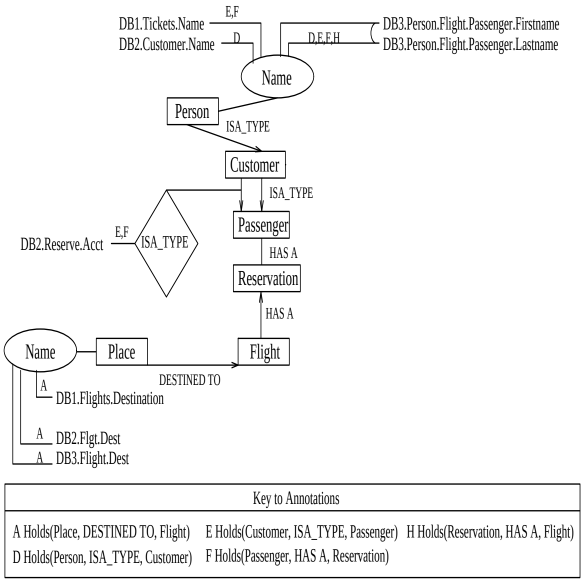

Figure 3: Traversed Links

2.3 An Example

Suppose we issue a query to find the names of all travelers who went to Paris. If DB1 was issuing the query, the standard SQL would be:

SELECT t.Name
FROM Tickets t, Flights f
WHERE t.fltno = f.fltno
and f.destination = "Paris"

This query should be transformed to queries for the other databases, DB2 and DB3. Consequently, DB1 is referred to as the client database, and DB2 and DB3 are referred to as the server databases.

We enter the mediator through the links specified by those attributes mentioned in the original query, the *client query.* These would be Tickets.Name, Flights.Destination, Tickets.Fltno, and Flights.Fltno. The mediator concepts which correspond to the "selected" and "restricted" attributes in the query will be connected via a traversal through the mediator. Therefore, DB1.Tickets.Name and DB1.Flights.Destination need to be connected.

The traversal is done as follows. First, the four attributes mentioned in the client query have a set of annotations associated with them, formed by taking the union of all of the annotations listed on the four client links. This is the set C={A,E,F,G,H,K,L,O} as seen in Figure 2. Note that annotation A has been instantiated with the restriction Holds(Paris, DESTINED TO, Place). Thus, C represents all of the known global facts we have concerning the current *state* of the query.

Second, we traverse the graph to connect the concepts. An edge is traversable if the relationship expressed by that edge is confirmed by a fact in C. Thus, from the mediator concept Person.Name (denoted as Med.Person.Name) we can traverse from Med.Person to Med.Customer to Med.Passenger due to annotation E $\in C$ which says a customer is a passenger (and, therefore, a person, by inheritance.) Similarly, we can traverse to Med.Reservation by F, to Med.Flight by H, and lastly, to our "restricted" concept Med.Place.Name, by A. The resulting path is shown in Figure 3. Let $C' \subseteq C$ be the set of those annotations actually confirmed during the traversal. Here, C' ={A,E,F,H}. Thus, C' is that set of facts known for the queries destined to the server databases, i.e. the *server queries*.

Lastly, we find all sets of server concepts, D', which are along a given traversed path and whose union of annotations contains C'. Each D' is used as a starting point to retrieve answers. This requirement is necessary as it reveals the actual attributes in the server databases which will need to be retrieved as answers (or partial answers.) For this example, there are two D' sets: {DB3.Flight.Dest, DB3.Person.Flight.Passenger.First(Last)name} and {DB2.Reserve.Acct, DB2.Customer.Name, DB2.Flight.Dest}. The first set contains the exact set of attributes which is needed to retrieve answers from DB3. On the other hand, the second set requires at least a join on DB2.Reserve.Acct which was revealed by the complex ISA_TYPE relation between the mediator's Customer and Passenger concepts, denoted by the diamond in Figure 3.

For each D', each server's resulting local concepts may be grouped to form a Universal Relation Window (URW) which may be answered according to a query derivable from [5]. Alternatively, methods like [1] can also be used. Although a local query may require joins which are not explicitly part of the mediator's path or window, [5] or [1] unveil such attributes using the local database's functional dependencies. In this example, DB2's fltnum attribute, which is needed to join Reserve.Acct with Flgt.Dest, is discovered from the primary key dependencies. Additionally, DB2's acct attribute is needed to join Customer.Name with Reserve.Fltnum. Finally, the results for each URW query are retrieved, formatted, and returned.

Some observations are noteworthy. First, note that the link from DB1.Flights.fltno to Med.Flight.Flight_Number is not traversed because the join on DB1's fltno is specific to DB1 attaining the Destination and Passenger Name in the same tuple. Also, the conjunction of annotations along the paths must be satisfied. Thus, the link from Med.Place.Name to DB3.Flight.Org is not used because its annotations do not help confirm any of the facts in C'. Finally, note that there can be more than one path connecting the mediator concepts. As Figure 3 illustrates, there are two such paths which only differ in the edge from Customer to Passenger. For each path, there could be different C' and D' sets. The paths in the example result in the same C', but different D' sets.

2.4 Summary

Once the local concepts and relationships have been identified, the mediator, links, and their annotations can be generated. Given such a mediator, a query issued in one database may be algorithmically transformed into queries at the other local databases. This process involves traversing links to reach related concepts in other databases and modifying the query according to the annotations on those links.

The advantage of using this type of object oriented graph is that it allows for precise and accurate semantic modeling. The object oriented links provide standard relationships like ISA, HASA, AGGREGATE_OF, etc. The client/server links provide additional, open-ended semantics on relationships and usage that are not afforded by the standard ensemble of object oriented connections. These connections model how the individual member databases relate. The degree of closeness between two concepts can be precisely controlled by these link annotations. The ability to have this control is essential given that concepts which are treated as semantically equivalent in one query may not be so in another query. Furthermore, the query itself may not explicitly provide this knowledge.

3 Concluding Remarks

We have introduced theoretical and practical constructs for resolving semantic conflicts involved with database interoperability. Our model provides a methodology for identifying data correspondences that can provide a unified view of a multidatabase system.

There are several elements which invite future work. First, the original determination of relationships among objects has been assumed to be a manual procedure. This step is essentially the same as the "investigation phase" of [7]. In order to obtain a fully automatic dynamic integration, in which new databases can be integrated and old ones modified or deleted on the fly, this investigation phase must be knowledge-based.

Second, the degree to which the mediator has "world knowledge" about the application will ultimately determine how well it can automatically detect correspondences and relationships among features. While the annotations are a first step to provide such knowledge, a formal representation of such information will be required. While our use of the Holds construct to detail annotations in Figure 2 has been sufficient so far, it remains to be seen if all necessary knowledge can be expressed in this way. Rule-based organizations of such knowledge are now being proposed and investigated.

Third, we have mentioned updates to the mediator only with respect to adding a new database to the federation. This need not be the only situation in which the mediator changes. The advantages to having a dynamically changing mediator are numerous. The ability to adjust the granularity of concepts based on the semantics of the queries would lead to a better representation of global knowledge. Furthermore, if the mediator was able to monitor and adjust statistical information with regards to which links are more likely to produce actual answers, then future traversals may be further optimized.

References

[1] Shailesh Agarwal, Arthur M. Keller, Gio Wiederhold, and Krishna Saraswat. Flexible relation: An approach for integrating data from multiple, possibly inconsistent databases. In *Proceedings of the 11th International Conference on Data Engineering*, pages 489–504, Taipei, Taiwan, March 1995. IEEE.

[2] DeMichiel. Resolving database incompatibilitiy: An approach to performing relational operations over mismatched domains. *IEEE Trans. on Knowledge and Data Engineering*, 1989.

[3] Wen-Syan Li and Chris Clifton. Semint: A system prototype for semantic integration in heterogeneous databases. In *Proceedings of the 1995 ACM SIGMOD Conference*, San Jose, California, May23-25 1995.

[4] Lim, Srivastava, Prabhakar, and Richardson. Entity identification in database integration. *Proc. IEEE Intern. Conf. on Data Engineering*, pages 294–301, 1993.

[5] David Maier and David S. Warren. Specifying connections for a universal relation scheme database. *ACM*, pages 1–7, 1982.

[6] Xiaolei Qian and T.F. Lunt. Semantic interoperation: A query mediation approach. Technical Report 02, Computer Science Laboratory, SRI International, April 1994.

[7] Spaccapietra, Parent, and Dupont. Model independent assertions for integration of heterogenous schemas. *VLDB Journal*, pages 81–126, 1992.