Using Metamodels as Part of a Semantic Gateway among Databases

Michael L. Dowell
Larry M. Stephens
Ronald D. Bonnell

March 1995

USC Technical Report ECE-MLD-030-95
Department of Electrical & Computer Engineering
University of South Carolina
Columbia, South Carolina 29208

Table of Contents

I Introduction

Different groups of users within an enterprise have different perceptions and requirements for the organization and representation of semantically overlapping information. This has led schema designers to develop different, often incompatible, schemas of information for each group. This solution has worked well in isolation. However, users needing to combine information from several sources are faced with the problem of locating and integrating relevant information as shown in Figure 1.

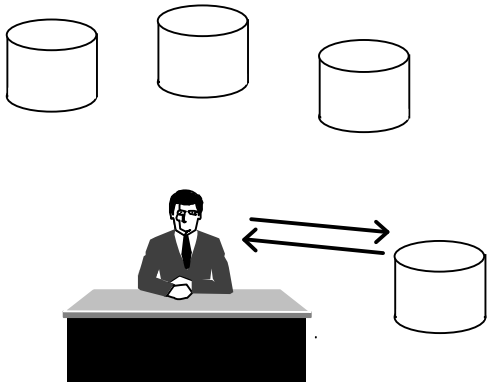

Figure 1: The User's Problem

One possible solution to the user's problem is simply allowing the user access to the relevant database systems, shown in Figure 2; but often a user does not know other databases exist. Even if the user knows about the existence of the other databases, the user probably does not know the schemas of the other databases. This solution also requires the user to learn all the other schemas.

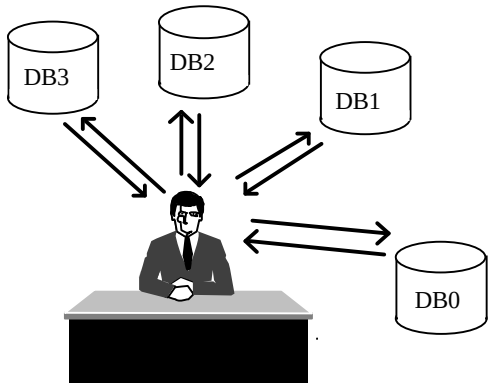

Figure 2: User Access to all Databases

This solution places an unacceptable burden of learning many different schemas upon users needing to integrate enterprise information from several different sources. Current research efforts focus on determining the overall approach to solve this enterprise information integration problem.

One type of approach, the federation approach, determines which information system processes need to interact, the specific purpose of and protocols used for each interaction [IRDS]. It does not deal with integration at the logical or semantic level. All that the information system processes know about the interaction is the way in which they send or receive data from other processes. The semantics of the domain is hidden. In addition, the federation approach employs the notion of encapsulation in which communicating objects need not know anything about what goes on inside the others. This is essentially a short-term solution in which programmers link the two communicating processes together for a specific purpose. Long term problems arise due to the limited and specialized nature of the interaction, which leads to a lack of flexibility. In addition, the lack of access to semantics in the context of the integration activity further restricts the flexibility since the semantics are built-in for each interaction.

Another type of approach is based upon translation [IRDS]. It involves two or more information systems interacting through the use of a common interchange format. The intent is to provide a deeper level of semantic integration than with the federation approach. The common interchange format and deeper semantic integration provides greater flexibility since the interaction is less dependent upon the built-in semantics of a specific interaction. Pair-wise translation quickly reaches its limits in terms of flexibility when applied to any reasonably complex environment where several information systems need to integrate and interact with one another. An alternative method involves defining generic protocols to be used by two or more information systems. This method mitigates the efficiency and flexibility problems; however, it still involves translation from one language to another. There is an ongoing need to extend, revise, and formulate constructs in the common interchange form as the participating information systems change. Translation-based approaches typically use a data dictionary or repository system.

The third type of approach, unification, is an extremely general form of translation but is considerably different in terms of what is being accomplished and how it is accomplished [IRDS]. Unification uses a set of deep representational constructs to model both the Universe of Discourse (UoD) and any representation of the UoD. In other words, using the primitive constructs of this approach, all other more limited representation schemas can be subsumed. Unification-based approaches use an advanced dynamic data dictionary or a repository management system.

This paper proposes a mechanism for a user to query other relevant databases using only the local relational model that the user knows, as shown in Figure 3. This mechanism allows the user to retrieve information from other databases without learning other schemas. In addition, existing databases and programs do not have to be changed.

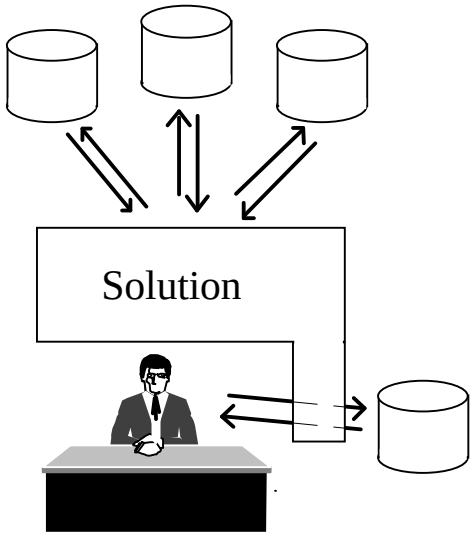

Figure 3: User's View of Proposed Solution

The next section describes the overall model of the Domain Knowledge Repository System and two metamodels. Section III presents the domain knowledge repository. An example query is then introduced in Section IV. Finally, the conclusions and future work are presented in Section V.

II The Domain Knowledge Repository Model

This paper shows how domain knowledge and database metamodels permit users, having only the their knowledge of a local relational model, to query other databases for relevant information. The Domain Knowledge Repository System (DKRS) model is shown in Figure 4.

In this figure, a user queryies a local database using his knowledge of the local database's relational model. The DKRS system captures this query and links it to the relational metamodel that represents the local database. This relational metamodel is linked to the extended entity-relation metamodel that also represents the local database. In addition, the extended ER metamodel is linked to the domain knowledge repository. The domain knowledge repository contains knowledge that is common to the different databases within the DKRS. In this manner, the user's local query is linked to two metamodels and the domain knowledge repository. The domain knowledge repository and the metamodels are described in the next three sections.

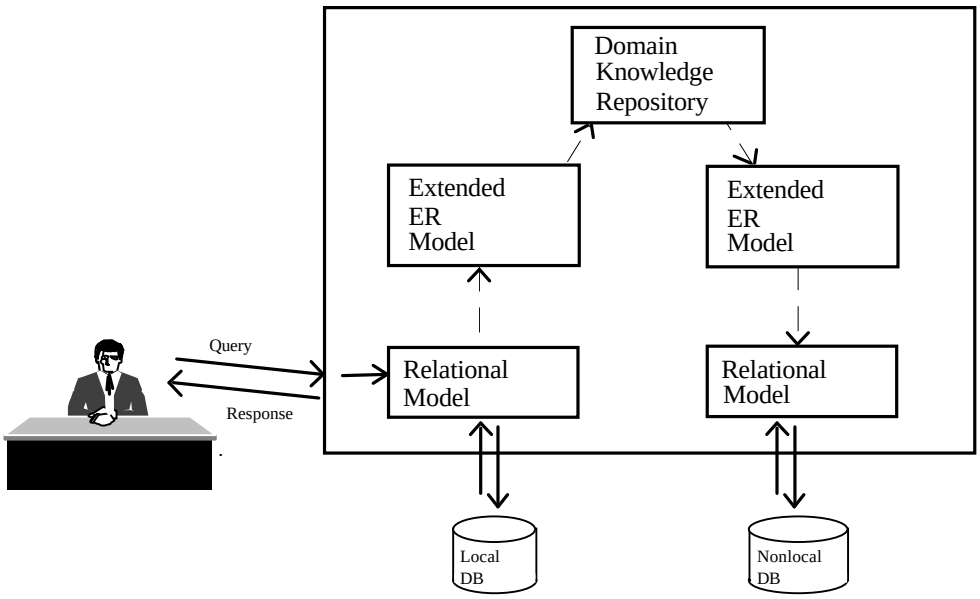

Figure 4: The DKRS model

A. Relational Metamodel

The relational metamodel, shown in Figure 5, is derived from the ideal table metamodel of Blaha *et al.*[Blaha]. Each *Ideal table* is linked to a *Database* since multiple databases are connected to the DKRS. Each *Ideal table* has one or more *Ideal table columns*. Each *Ideal table column* is grouped into one or more *Column lists*. In addition, each *Ideal table* has associated primary, candidate, and foreign key lists.

The information represented by the relational metamodel is at the logical level. Users make queries using tables and columns within the database by modeling the information with this metamodel.

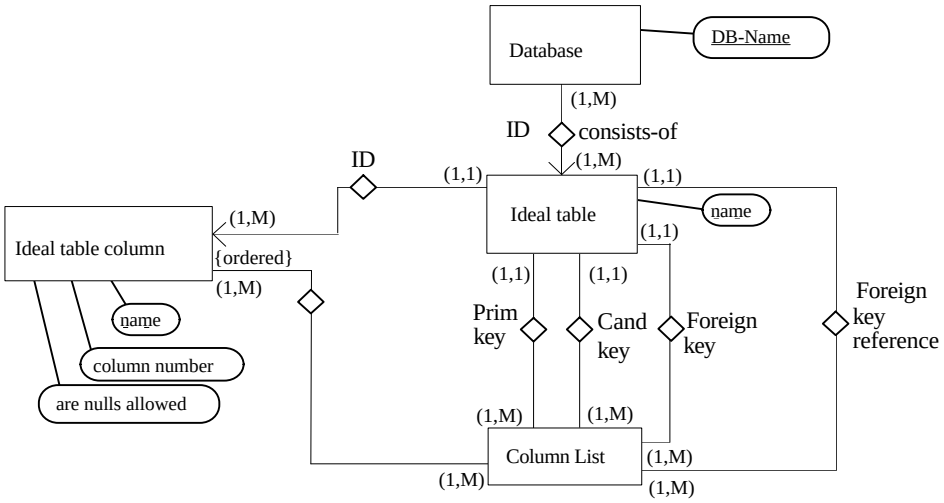

Figure 5: Relational Metamodel

B. Extended Entity-Relation Metamodel

Some of the information represented within the relational metamodel is implicit. For example, a foreign key implicitly represents a relationship between two entities sets. This is one of the reasons the DKRS uses an intermediate representation, the extended entity-relation metamodel, between the relational metamodel and the domain knowledge repository. The implicit information from the relational metamodel is made explicit in the extended entity-relation metamodel of Figure 6.

Furthermore, we have found that the semantic gap between the logical level (relational model) and the semantic level (domain knowledge) is too broad to bridge directly. Finally, the extended ER metamodel provides for a general data model independent of the actual database implementation.

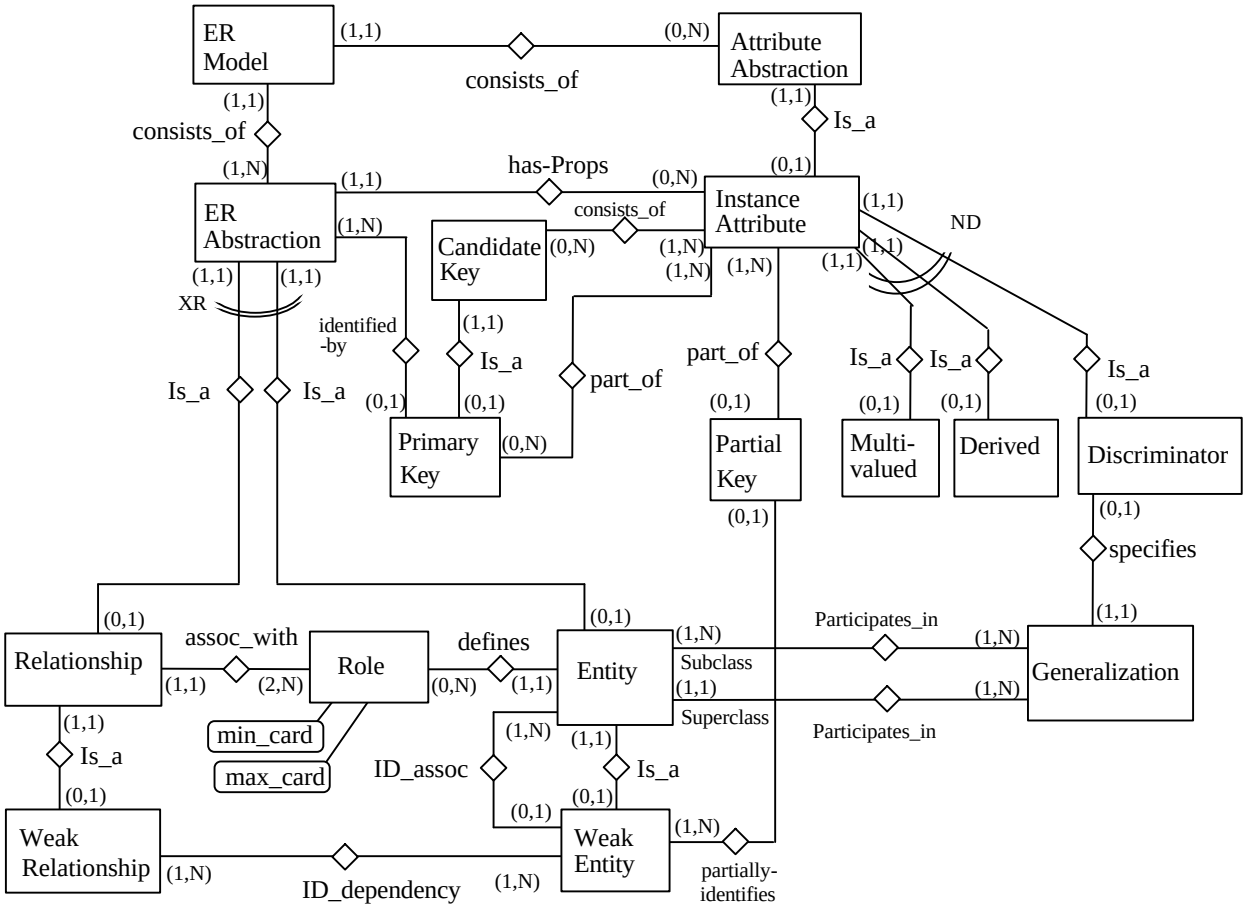

Figure 6: The Extended ER Metamodel

III Domain Knowledge Repository

The domain knowledge repository contains a representation of semantically overlapping information from the different information sources connected to the DKRS. These information sources have different, often incompatible, schemas of the overlapping information. While the structures of the different schemas are represented with the DKRS relational and extended ER metamodels, it is the domain knowledge repository that represents the semantics of the information contained within the different schemas.

The domain knowledge is represented using an ontology of concepts and relations. These concepts are used in the process of semantic unification, which consists of mapping the concepts in each database to concepts in the common ontology. This mapping is done by DKRS users since only the users know the exact meaning of the domain concepts within their database schemas.

IV An Example

For this example, two similar sets from different schemas are linked to different levels of the domain ontology, as shown in Figure 7. A student entity set is stored in database 1 and is represented in schema 1, a extended ER metamodel linked to a relational metamodel representing the information stored in database 1. The metamodels and links to the ontology are done prior to the query. The student table is linked to the student concept in the domain knowledge through relational and extended ER metamodels. In addition, a person entity set in database 2, represented in schema 2, is linked to the person concept in the domain ontology. These entity sets have overlapping information concerning attributes of people since students are also people.

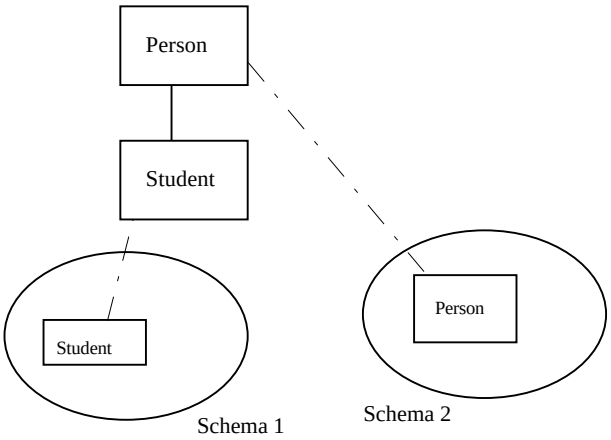

Figure 7: Different Ontology Levels Example

A user, familiar with the schema of database 2, queries the DKRS about attributes of the Person entity set. If there is overlapping information stored in database 1, the DKRS can also retrieve information about the Student entity set in database 1. The query on Schema 1 is a specialization of the query on Schema 2 since the information from Schema 1 is only about students. The query on Schema 1 will return a subset of the tuples

desired since the original query concerns all Persons but database 1 has information only about Students. The user querying about Person should be made aware that the information returned from Schema 1 is only about Students not about all the persons at the university. Specializing queries by moving down the type hierarchy of the domain ontology, as shown in Figure 8, returns relevant information from other information sources.

Query(Person, Local-Schema-2) ==> Query(Student, Local-Schema-1)

Figure 8: Specialization of a Query

A more interesting problem is a query that must move up the type hierarchy of the domain ontology. If a query is about attributes of the student entity set that are also attributes of the person entity set, should a query about persons be generated for Schema 2 even though the original query concerns only students? This second query is a generalization of the original query and will return a superset of the tuples requested. Generalizing queries by moving up the type hierarchy may return information that is not relevant. Deciding when to move up the type hierarchy and how far to move up is an open question. Generalizing a query will return more information and may be an expensive operation since, for example, there may be many more people in database 2 than there are students in database 1. One solution is to ask the user making the query if a specific generalization of the query is desired.

Query(Student, Local-Schema-1) ==> Query(Person, Local-Schema-2)

Figure 9: Generalization of a Query

For the last example, two similar relationships from different schemas are linked to different levels of the domain ontology, as shown in Figure 10. This example involves more of the same type of problems as the previous example but the decision of when to generalize or specialize involves not only entity concepts but also relationship concepts.

In this example, a *tracked-by* relation in schema 1 is linked to the *tracked-by* concept in the domain ontology while a *traced-by* relation in schema 2 is linked to the higher level *traced-by* concept. In addition, each entity concept involved in the schema relations is linked to a domain concept. Generalizing a query involves generalizing not only the relation but also the entities. For instance, a query concerning *tracked-by* in schema 1 can be generalized to include queries about *traced-by* relationships in schema 2. But the two entities linked together by track-by, Response and Coordinator, must also be generalized to two higher level concepts, Document and Person, as shown in Figure 11.

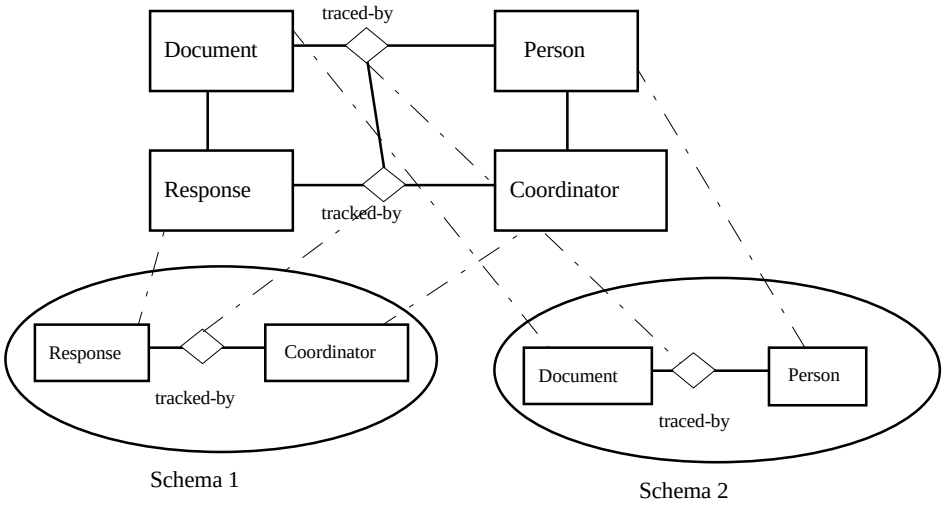

Figure 10: Different Relation Ontology Levels

Query(*tracked-by*, Response, Coordinator, Local-Schema-1) ==>
Query(*traced-by*, Document, Person, Local-Schema-2)

Figure 11: The Required Generalizations for a Relation Query

V Conclusions

Different sources of information have different perceptions of overlapping information. This leads to different representation of overlapping information. The DKRS uses two metamodels to bridge the gap between the logical level (relational model) and the semantic level (domain knowledge). The structural differences are represented and processed using the relational and extended ER metamodel. The semantic differences are represented and processed using the domain knowledge ontology. A prototype to explore the knowledge representation requirements has been implemented using ROCK, a frame-based system designed around the C and C++ programming languages. Future work will look at developing algorithms to generalize and specialize queries.

References

[Blaha]

M. R. Blaha, W. J. Premerlani, and H. N. Shen, "Compiling Object-Oriented Models into Relational Database Schema," *IEEE Software*,

[Davis]

James P. Davis and Ronald D. Bonnell, "EDICT - An Enhanced Relational Data Dictionary: Architecture and Example," *Proceedings of the Fourth International Conference on Data Engineering*, February, 1988, Computer Society Press, pp. 184-191

[IRDS]

[Spaccapietra]
Stefano Spaccapietra and Christine Parent, "View Integration: A Step Forward in Solving Structural Conflicts", IEEE Transactions on Knowledge and Data Engineering, Vol. 6, No. 2, April 1994, pp. 258-274

[Stephens]
Larry M. Stephens and Yufeng F. Chen, "Principles for Organizing Semantic Relations in Large Knowledge Bases", USC Technical Report ECE-LMS-94-08, October 1994, University of South Carolina