Refining Semantic Mappings from Relational Tables to Ontologies

Yuan An[1], Alex Borgida[2], and John Mylopoulos[1]

[1] Department of Computer Science, University of Toronto, Canada
{yuana,jm}@cs.toronto.edu
[2] Department of Computer Science, Rutgers University, USA
borgida@cs.rutgers.edu

Abstract. To support the Semantic Web, it will be necessary to construct mappings between legacy database schemas and ontologies. We have developed a prototype tool which starts from a simple set of correspondences from table columns to ontology components, and then helps derive algorithmically candidate logical mappings between complete tables and the ontology. We report here some refinements of this algorithm inspired by an analysis of the ways in which relational schemas are standardly derived from Extended Entity Relationship diagrams, and relate this to the main heuristic used by the Clio system [6], which maps between relational database schemas.

1 Introduction

In order to make the vision of the Semantic Web a reality, it will be necessary to find semantic mappings between existing databases (the "deep web") and existing ontologies. Building such connections is nontrivial task because: (i) the ontologies and schemas will have been derived *independently*; (ii) the ontologies (and schemas) could be very large; (iii) there will be relatively few people who will be thoroughly familiar with any one of them; (iv) the construction would have to be repeated when encountering new ontologies. For this reason, it would be desirable to have computer tools to help find the logical mappings between database schemas and ontologies.

Specifically, we assume a framework where we are given

1. An ontology, expressed in some language, such as OWL or UML, which has a semantics that can be captured by First Order Predicate Logic through the use of unary and binary predicates, representing concepts and properties. The ontology language should support domain, range and cardinality restrictions on properties and their inverses, and differentiate datatype valued properties ("attributes" in UML).
2. A relational schema, where for each table we have standard information available in SQL DDL declarations, including constraints concerning the primary key, foreign keys, and absence of null values.

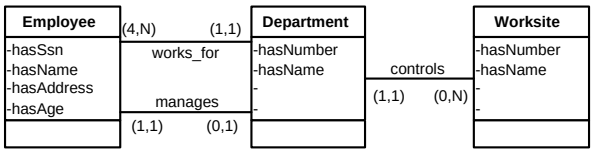

Fig. 1. Company Ontology.

Our general objective is to find a mapping relating predicates in the ontology and relational tables [5]. Currently, we are obtaining for each table $T(A_1, \ldots, A_n)$ a formula ϕ that is a (disjunction of) conjunction of ontology atoms. For example, given the ontology in Figure 1, and relational table schema $Emp(ssn, name, dept, proj)$, we may expect an answer of the form

$$\begin{array}{l}\mathcal{T}{:}Emp(ssn, name, dept, proj){:}\text{-}\\ \quad \mathcal{O} : Employee(x), \mathcal{O} : hasSsn(x, ssn), \mathcal{O} : hasName(x, name),\\ \quad \mathcal{O} : Department(y), \mathcal{O} : works\_for(x, y), \mathcal{O} : hasNumber(y, dept).\\ \quad \mathcal{O} : Worksite(z), \mathcal{O} : controls(y, z), \mathcal{O} : hasName(z, proj).\end{array}$$

where, for clarity, we use prefixes $\mathcal{T}$ and $\mathcal{O}$ to distinguish predicates in the relational schema and the ontology.

2 The MAPONTO Approach.

To achieve the above objective, we envision a 2-step process, where (a) the columns A_i of each table are linked to elements in the ontology (mostly datatype-valued properties); then (b) a formula of the kind described above is proposed by the tool on the basis of heuristics. Since considerable effort has been devoted to step (a) in the database and ontology integration literature, we have concentrated on step (b). Because the answers produced are inherently heuristic, our tool offers a *partially ordered list* of formulae, where more highly ranked proposals are assumed to be more likely.

The basic idea underlying our current tool (detailed in [1]) is to represent the ontology as a graph consisting of nodes (corresponding to concepts) connected by edges (corresponding to properties). Semantic connections in the ontology, expressed in the formula ϕ, are then based on paths in this graph, and we hypothesize a version of Occam's razor: fewer connections are better. This has lead us to look for minimal-cost spanning trees connecting the concepts which have one or more properties corresponding to table columns — called *Steiner trees*. Such a tree is then translated to a logical formula by "joining" the concepts and properties encountered in it. For example, if concepts C and D are connected by the tree consisting of edges p and q, traversing intermediate node G, the formula produced is $C(x), p(x, y), G(y), q(y, z), D(z)$.[3]

[3] The algorithm in [1] is considerably more elaborate; among others, sometimes copies are made of certain nodes in the graph, so that more than one variable can range over a concept.

This paper presents some refinements to this algorithm, their motivation based on techniques for mapping from Extended ER diagrams to relational database schemas, and their relationship to other previous research.

2.1 Related Work

The framework of our approach is clearly inspired by the Clio system [6, 8], which attempts to find mappings between two relational schemas. In fact, in Section 4.2, we will relate the key heuristic underlying Clio to our work.

The general framework for connecting ontologies and relational schemas using logical formulas has a long history [2, 5], although in all previous cases the specification is done entirely by the designer.

Data reverse engineering is the process of taking an existing database schema (and instances), and recovering some corresponding conceptual schema. Various approaches have been proposed (e.g., [7]), with a comprehensive introduction provided by Hainaut [4]. Our problem differs in two ways: we are given an *existing* ontology, which needs to be connected to the database; and the ontology will likely contain much superfluous information that will not appear in the schema. (For example, a $City$ might be $locatedIn$ a $Province$, and a $Province$ $locatedIn$ a $Country$, yet the relational table may only have columns for $cityName$ and $countryName$.) Conversely, we have to face the fact that some aspects of the database (e.g., aritificial identifiers), may not appear in the ontology.

3 From Extended ER to Relational Schema

Our new proposal is based on the methodology of relational schema design from Extended Entity-Relationship (EER) diagrams.[4] The principles behind this widely-understood and practiced technique are to create a small number of tables that are in Boyce-Codd Normal Form (assuming that the only dependencies are those due to keys in the EER diagram), preferring schemas where columns do not have null values, which might waste space. The basic methodology can be summarized as follows:

- For each regular entity type E create an "entity table" whose columns are the attributes of E, and whose primary key is the key of E.
- For each weak entity type E, create an "entity table" whose columns are the attributes of E, together with the key of the owner entity type O, and any attributes of the "identifying relationship". The primary key of the table consists of the concatenation of the key for O and the identifying attribute(s) of E.
- For each relationship type R create a "relationship table" whose columns are the primary keys of the participating entity types, together with attributes of R. The key of the table is determined by the cardinality of relationship: if R connects entities A and B, with keys K_A and K_B respectively then
  - if R is an N:M relationship, then the key is the union of K_A and K_B;
  - if R is a N:1 relationship, then the key is just K_A, while for 1:N relationships the key is K_B;

[4] We assume the reader is familiar with standard EER terminology, e.g., [3].

- if R is 1:1, the key should be that of the entity whose participation in the relationship is *total* (i.e., has cardinality lower bound 1); otherwise the choice of key is arbitrary.

– Repeatedly merge any pair of tables that have the same key by unioning the set of attributes and eliminating the duplicate key attributes. In some cases (e.g., merging a N:1 relationship which is not "total"), the result may have columns that could now be null. Such merges are less desirable.

Finally, an important aspect of the relational schema design is imposing appropriate foreign key and non-null constraints during the above construction.

The above mapping results in a schema that is very natural for humans to understand, and may well be the one encountered in practice, unless denormalized in order to improve query processing.

4 New Heuristics for Finding Mappings

The search for a low-cost Steiner tree[5] may return a number of results, some of which may be intuitively better than others, especially if the table is not "denormalized". For example, in the semantic mapping for the $\mathcal{T} : Emp$ example earlier, the tool should prefer $works\_for$, rather than $manages$, as the connection between $Employee$ and $Department$. On the other hand, for a table $\mathcal{T} : Project(name, supervisor, ...)$, the connection of $\mathcal{O} : Worksite$ with $\mathcal{O} : Employee$ would more likely involve the composition of $controls^{-1}$ with $manages^{-1}$ rather than with $works\_for^{-1}$.

To achieve this, we propose that the spanning sub-trees be ranked according to the following rules:

1. In growing a tree, edges with cardinality upper-bound 1 should be preferred. Edges which also have cardinality lower-bound 1 are given even higher preference.
2. If the relational table has a key that is the composite of foreign keys k_A, k_B,..., and there are single anchor nodes[6] for each of the keys k_A,..., then those trees are preferred in which every pair of anchor nodes is connected by a *path* on which some edge has cardinality upper bound higher than 1. (The simplest case of this is an edge for a property p such that the cardinality upper bound of p and its inverse are larger than 1.)
3. If the relational table has a key $K = K_C + A_1 + ...$, where K_C is a foreign key, and the A_i are not part of a foreign key appearing in K, then the real anchor node is not the node (call it C') which has properties matching K_C; instead, it is some other node D, which has a 1-upper bound path to C', such that D has some attributes for A_j.

[5] From our motivation, minimality is not strictly necessary, and it may be hard to ensure, since the problem is NP-hard.

[6] An anchor node is a concept which has datatype properties corresponding to columns in the table key

4.1 Motivating the New Heuristics

Suppose the ontology corresponds *exactly* to the conceptual model used for database design, and the relational schema is obtained according to Section 3. Furthermore, let us restrict ourselves to the case when the EER diagrams that can be represented directly in ontologies, by mapping entity types into concepts, and binary relationship types and attributes into properties.

An analysis of the algorithm in Section 3 shows that every table produced by it will have key K of the form (i) K_E for an entity table, possibly merged with some N:1 or 1:1 relationship tables involving that entity; (ii) same as (i), but corresponding to N:1/1:1 relationships that were not merged in with the entity because the participation was not total; (iii) $K_A + K_B$, where entities A and B are in an N:M relationship; (iv) $K_E + A_1 + ...$, for a weak entity table, where $A_1...$ do not form a key; (v) the analogue of (ii) for weak entities.

Columns in tables of category (i) and (ii) correspond to datatype properties of a concept C_E (possibly a subclass), as well as properties of entities related to it by N:1/1:1 relationships. All of these appear as properties of C_E with upper bound 1 in the ontology. Moreover, the preference for total participation corresponds to lower-bound 1. Hence case 1 of our heuristic.

Tables in category (iii), which can be recognized from their key structure, and which have columns corresponding to N:M relationships between properties of anchor entities, would be miss-treated by the algorithm, which would prefer N:1/1:1 linking properties. Hence case 2 of our heuristic.

Tables corresponding to weak entities are another source of problems, since weak entities are an artifact of the EER data model, and are not specially marked in standard ontologies. Once again, we recognize weak entities from the table key structure, and then try to find the concept corresponding to the weak entity, and the (chain of) relationships/properties to the strong entity identifying it.

Our algorithm generalizes all these cases to the situation when a relationship in the EER model might be the composition of several properties in the ontology, by permitting the Steiner tree to traverse additional edges, if necessary, as long as their upper bound is 1.

4.2 Relationship to Clio Heuristics

Recall that Clio tries to find a logical mapping from a source to a target relational (or XML) schema, starting from correspondences between columns of tables in them. The core of Clio is the generation, in both the source and target schema, of "logical relations" [8] — maximal sets of logically related schema elements, particularly table columns. This is accomplished as follows: if table R has a foreign key to table T, then (a) R and T are joined over this foreign key to yield a larger table, R', and (b) the process is repeated on R'[7].

First, as we noted, in MAPONTO ontology subtrees give rise to formulas representing joins similar to Clio's logical relations.

Now suppose ontology properties $p(x, y)$ and $q(y, z)$ meet at concept $G(y)$. If these were relational tables, columns y of p and q would be foreign keys for

[7] This is related to the notion of "chase" in relational databases.

G. Clio would only suggest $p(x,y) \bowtie G(y)$ or $G(y) \bowtie q(y,z)$ as joins building alternative logical relations. The reason for avoiding $p(x,y) \bowtie G(y) \bowtie q(y,z)$, is that from practical experience, this could lead to too many alternatives[8]. Step 1 of the new heuristics will make MAPONTO also downgrade such joins **if** the relationship corresponding to q is M:N. Moreover, if q represented a N:1/1:1 relationship then according to Section 3, table q could have been merged with E to yield $E'(y,z) \iff G(y) \bowtie q(y,z)$. In this schema, Clio would in fact join p and E', as would our algorithm, since q has cardinality upper bound 1.

The iterative nature of Clio's algorithm is also captured by our tree growing algorithm.

5 Conclusions and Future Work

Establishing manually semantic mappings between database schemas and ontologies is time-consuming and error-prone, especially, when the mappers are not fully cognizant of the ontology, which could be very large. We are developing a tool, MAPONTO, to support creating such mappings, and have carried out several experiments using it [1]. In this paper, we have presented certain refinements of the algorithm intended to deal with several problems we have encountered, together with explanations tying the heuristics to the well-known mapping of EER diagrams to relational schemas, and the heuristics used in Clio.

We defer to a later paper the treatment of n-ary relationships, the less standard tabular representations of semantic relationships (such as the representation of subclass hierarchies using concept names as values in columns), denormalized relations, and more complex mappings.

References

1. Y. An, A. Borgida, and J. Mylopoulos. Building Semantic Mappings between Database Schemas and Ontologies. Submitted for publication.
2. D. Calvanese, G. D. Giacomo, M. Lenzerini, D. Nardi, and R. Rosati. "Data integration in data warehousing", *J. Cooperative Information Systems. 10(3)*, 2001.
3. R. Elmasri and S. B. Navathe. *Fundamentals of Database Systems.* Addison-Wesley. 2000.
4. J.-L. Hainaut. Database reverse engineering. http://citeseer.ist.psu.edu/article/hainaut98database.html. 1998.
5. J. Madhavan, P.A. Bernstein, P. Domingos, A.Y. Halevy. "Representing and Reasoning about Mappings between Domain Models", *AAAI 2002*: 80-86
6. R. J. Miller, L. M. Haas, and M. A. Hernandez. Schema mapping as query discovery. In 26th VLDB. 2000.
7. V. M. Markowitz and J. A. Makowsky. Identifying Extended Entity-Relationship Object Structures in Relational Schemas. IEEE Transactions on Software Engineering 16(8). 1990.
8. L. Popa, Y. Velegrakis, R. J. Miller, M. Hernandes, R. Fagin. "Translating web data", *VLDB 2002.*

[8] Y.Velegrakis, personal communication, 2004.