Reverse Engineering of Relational Databases to Ontologies: An Approach Based on an Analysis of HTML Forms

Irina Astrova[1], Bela Stantic[2]

[1] Tallinn University of Technology, Ehitajate tee 5,
19086 Tallinn, Estonia
irina.astrova@cellnetwork.com
[2] Griffith University, PMB 50 Gold Coast Mail Center,
QLD 9726 Australia
b.stantic@griffith.edu.au

Abstract. We propose a novel approach to reverse engineering of relational databases to ontologies. Our approach is based on the idea that semantics of a relational database can be inferred, without an explicit analysis of relational schema, tuples and user queries. Rather, these semantics can be extracted by analyzing HTML forms, which are the most popular interface to communicate with relational databases for data entry and display on the Web. The semantics are supplemented with the relational schema and user "head knowledge" to build an ontology. Our approach can be applied to migrating data-intensive Web pages, which are usually based on relational databases, to the ontology-based Semantic Web.

1 Introduction

With the development of Web technology, data-intensive Web pages [1], which are usually based on relational databases, are seeking ways to migrate to the ontology-based Semantic Web [2]. This migration calls for reverse engineering of relational databases to ontologies [3]. However, there are few approaches that consider ontologies as the target for reverse engineering. A majority of the work on reverse engineering has been done on extracting a conceptual schema (such as an entity-relationship) from relational databases. As an attempt to fill gap in this area, we propose a novel approach to reverse engineering of relational databases to ontologies.

2 Related Work

There exist few approaches that consider ontologies as the target for reverse engineering. The existing approaches fall roughly into one of the three categories or their combination:

- *Approaches based on an analysis of relational schema:* E.g. Stojanovic et al's approach [3] provides a set of rules for mapping constructs in the relational database (i.e. relations, attributes, tuples and constraints) to semantically equivalent constructs in the ontology (i.e. classes, attributes, instances and axioms). These rules are based on an analysis of relations, key and non-key attributes, functional and inclusion dependencies. Dogan and Islamaj's approach [4] provides simple and fully automatic (i.e. without user interaction) reverse engineering: relations map to classes, attributes in the relations map to attributes in the classes and tuples in the relational database map to instances in the ontology. However, this approach ignores inheritance, thus building the ontology that looks rather "relational"
- *Approaches based on an analysis of tuples:* E.g. Astrova's approach [5, 6] builds an ontology based on an analysis of relational schema. Since the relational schema often provides very few explicit semantics, this approach also analyzes tuples in the relational database to discover additional "hidden" semantics (e.g. inheritance)
- *Approaches based on an analysis of user queries:* E.g. Kashyap's approach [7] builds an ontology based on an analysis of relational schema; the ontology is then refined by user queries. However, this approach does not create axioms, which are part of the ontology.

The application of the existing approaches to migration of data-intensive Web pages to the Semantic Web can be limited by the completeness of input information and its correctness. On one hand, the complete information about the relational database, such as functional and inclusion dependencies, is usually not available. There are always relations with unknown primary keys and attributes with naming conflicts (e.g. synonyms and homonyms) [8]. Therefore, Dogan and Islamaj's, Stojanovic et al.'s, Astrova's and Kashyap's approaches may require more input information than it is possible to provide in practice. On the other hand, a bad-designed, optimized and denormalized relational schema, and erroneous tuples must also be taken into consideration. However, Astrova's approach assumes that tuples in the relational database contain no errors. Moreover, this approach considers a relational schema in third normal form (3NF) – again an unrealistic assumption, because the relational schema can be in any normal form [9].

By contrast to the existing approaches, our approach is based on the idea that semantics of the relational database can be inferred, without an explicit analysis of the relational schema (i.e. relations, key and non-key attributes, functional and inclusion dependencies), tuples and user queries. Rather, these semantics can be extracted by analyzing HTML forms. The semantics are supplemented with the relational schema and user "head knowledge" to build an ontology.

3 Why to Analyze HTML Forms

There are many reasons why to analyze HTML forms:

- HTML forms are the most popular interface to communicate with a relational database for data entry and display on the Web

- HTML forms are designed to be user-friendly. Therefore, they are usually associated with some descriptive text to help the users understand the semantics. This is an important advantage of HTML forms, as the users have to guide the process of reverse engineering. User interaction is needed, when ambiguities occur or semantics of the relational database cannot be inferred automatically [3]
- HTML forms do not necessarily mirror the structure of the relational database (i.e. the relational schema) behind them. Rather, they serve as communication interface with the relational database. The relational database can be bad-designed and denormalized; it can contain unusual and optimization structures [8]
- HTML forms are structured collections of fields, which are appropriately formatted to communicate with the relational database. Therefore, data contained in the forms is usually structured, while the structure of the relational database is often unknown in advance [10]
- HTML forms often contain instructions (or annotations) that the users should follow to correctly fill in fields or to get relevant information on the entries of fields. These instructions are part of the contextual knowledge [11]
- Field names in HTML forms are usually more explicit and meaningful than the names of corresponding attributes in the relational database.

4 Our Approach

Our approach uses a relational schema and HTML forms as input, and goes through three basic steps: (1) analyzing HTML forms to extract a form model schema that expresses semantics of a relational database behind the forms, (2) transforming the form model schema into an ontology ("schema transformation"), and (3) creating ontological instances from data contained in the forms ("data migration").

4.1 Extracting Form Model Schema from HTML Forms

A form model schema [12] was originally proposed to extract an entity-relationship schema from database forms. It consists of:

- *Underlying source:* This is a structure of the relational database (i.e. a relational schema), which defines relations and attributes along with their data types
- *Form field:* This is an aggregation of name and entry associated to it[1]. A *name* is pre-displayed and serves as a clue to what is to be entered by the user or displayed by the HTML form. An *entry* is the actual data; it roughly corresponds to an attribute in the relational database. We use the term of *linked attribute* for such an entry to distinguish it from other entries, which are computed or simply unlinked with the underlying source

[1] However, we can also identify a form field, where there is no name for the entry.

- *Structural unit:* This is a logical group of closely related form fields. It roughly corresponds to a relation in the relational database
- *Relationship:* This is a connection between structural units that relates one structural unit to another (or back to itself). There are two kinds of relationship: *association* and *inheritance*
- *Constraint:* This is a rule that defines what data is valid for a given form field. A *cardinality constraint* specifies for an association relationship the number of instances that a structural unit can participate in
- *Form type:* This is a collection of empty form fields
- *Form template:* This is a particular representation of form type. Each form template has a *layout* (i.e. its graphical representation) and a *title*, which identifies the HTML form and provides its general description
- *Form instance:* This is an occurrence of form type, when its template is filled in with data.

In reverse engineering of relational databases to ontologies, the first step is analyzing HTML forms to extract a form model schema. Basically, this means identifying structural units, linked attributes and relationships using the wrapper generation techniques [10]. It can be accomplished most successfully by the users (e.g. domain experts), who are knowledgeable about the application domain. Indeed, it is usually easy for the users to parse HTML forms in order to understand their structure and meaning, while performing these tasks is often difficult to automate. E.g. difficulties arise, when the forms contain multiple frames, optional and multi-valued attributes, attributes as values, merged attributes and values, externally factored data, duplicate data and synonyms [10]. Therefore, our approach emphasizes the user "head knowledge" in the analysis of structure of HTML forms, assuming that the forms are user-friendly.

In addition to the structure of HTML forms, data contained in the forms are analyzed to identify constraints. Data analysis includes a strategy of learning by examples borrowed from the machine learning techniques [13, 14]. In particular, it is performed as a sequence of learning tasks from the relational database. Each task is specified by: (1) *task relevant data* (e.g. data contained in the forms), (2) *problem background knowledge* (e.g. the application domain knowledge), and (3) *expected representation of results of learning tasks* (e.g. the first order predicate logic) [12]. The results of learning tasks are related to a current state of the relational database; they are, therefore, generalized into knowledge about all possible states through an induction process. This process combines the semantics extracted from the forms with the application domain knowledge, which is provided by the users. This knowledge controls the learning tasks to come to the best inductive conclusion, the conclusion that will be consistent with all possible states of the relational database.

Several HTML forms are usually used to communicate with the relational database. Each form represents a particular view of the relational database. Therefore, an analysis of these forms will produce several form model schemata that will be merged into a single one through an integration process. First, the schemata are compared for overlaps in structure. This means looking for structural similarities: similar names of structural units and linked attributes, similar structures of structural units and relationships. Second, the schemata are compared for overlaps in meaning. By using their knowl-

edge of application domain, the users look for structural units that correspond to the same real-world objects, but have different names. Third, structural and naming conflicts (e.g. synonyms and homonyms) are resolved. Conflicts can also be in different constraints on the linked attributes and different cardinality constraints on the relationships. By going through these three steps, the integration process makes the schemata consistent with one another and brings them together into a single one that makes sense for all HTML forms.

4.2 Schema Transformation

The second step of reverse engineering is transforming the form model schema into an ontology ("schema transformation"). Basically, this means replacing a construct in the form model schema with a semantically equivalent construct in the ontology using the mapping rules in Table 1.

Table 1. Summary of schema transformation

	Mapping rule
1	Structural unit maps to class
2	Linked attribute in structural unit maps to attribute in class
3	Binary association maps to attribute in class; ternary or higher degree association gets its own class
4	Constraint maps to axiom
5	Inheritance hierarchy of structural units maps to inheritance hierarchy of classes

4.3 Data Migration

The third step of reverse engineering is creating ontological instances from data contained in the HTML forms ("data migration"). Basically, this means assigning values to the appropriate attributes in the ontology using the table understanding techniques [10].

5 Conclusion

We have proposed a novel approach to reverse engineering of relational databases to ontologies. Our approach is based on the idea that semantics of a relational database can be extracted by analyzing HTML forms (both their structure and data). These semantics are supplemented with the relational schema and user "head knowledge" to build an ontology.

Our approach can be applied to migrating data-intensive Web pages, which are usually based on relational databases, to the ontology-based Semantic Web. The main reason for this migration is to make the Web content machine-understandable [3].

Data-intensive Web pages do not exist, until they are generated dynamically from relational databases at the time of user requests [1]. Since data is usually represented through HTML forms for displaying to the users (i.e. intended for human consumption only), machines cannot understand and process data-intensive Web pages.

Acknowledgement

This research is partly sponsored by Estonian Science Foundation (ESF) under the grant nr. 5766.

References

1. P. Fraternali, Tools and Approaches for Developing Data-intensive Web Applications: A Survey, In: ACM Computing Surveys, Vol. 31, No. 3 (1999) 227–263
2. T. Berners-Lee, XML 2000 – Semantic Web Talk, http://www.w3.org/2000/Talks/1206-xml2k-tbl/slide10-0.html (2000)
3. L. Stojanovic, N. Stojanovic and R. Volz, Migrating Data-intensive Web Sites into the Semantic Web, In: Proceedings of the 17th ACM Symposium on Applied Computing (SAC) (2002) 1100–1107
4. G. Dogan and R. Islamaj, Importing Relational Databases into the Semantic Web, http://www.mindswap.org/webai/2002/fall/Importing_20Relational_20Databases_20into_20the_20Semantic_20Web.html (2002)
5. I. Astrova, Reverse Engineering of Relational Databases to Ontologies, In: Proceedings of the 1st European Semantic Web Symposium (ESWS), LNCS 3053 (2004) 327–341
6. I. Astrova, Extracting Ontologies from Relational Databases, In: Proceedings of the 22nd IASTED International Conference on Databases and Applications (DBA) (2004) 56–61
7. V. Kashyap, Design and Creation of Ontologies for Environmental Information Retrieval, In: Proceedings of the 12th Workshop on Knowledge Acquisition, Modeling and Management (KAW) (1999)
8. W. Premerlani and M. Blaha, An Approach for Reverse Engineering of Relational Databases, In: Communications of the ACM, Vol. 37. No. 5 (1994) 42–49
9. J. Hainaut, J. Henrard, J. Hick, D. Roland and V. Englebert, Database Design Recovery, In: Proceedings of the 8th Conference on Advanced Information Systems Engineering (1996) 272–300
10. D. Embley and C. Tao, Automating the Extraction of Data from HTML Tables with Unknown Structure, http://www.deg.byu.edu/papers/dke2003etl.pdf (2003)
11. M. Mannino, J. Choobineh and J. Hwang, Acquisition and Use of Contextual Knowledge in a Form-Driven Database Methodology, In: Proceedings of the 5th International Conference on the Entity-Relationship Approach (1986) 141–157
12. N. Mfourga, Extracting Entity-Relationship Schemas from Relational Databases: A Form-Driven Approach, In: Proceedings of the 4th Working Conference on Reverse Engineering (WCRE) (1997) 184–193
13. R. Michalski, A Theory and Methodology of Inductive Learning, In: Machine Learning: An Intelligence Approach, Vol. 1 (1983) 83–134
14. J. Paredis, Learning the Behavior of Dynamical Systems from Examples, In: Proceedings of the 6th International Workshop on Machine Learning (1989) 137–140