VEBO: Validation of E-R diagrams through ontologies and WordNet

Giuseppe Russo, Francesca Anastasio, Arianna Pipitone, Antonio Gentile, Roberto Pirrone
Dipartimento di Ingegneria Chimica, Gestionale, Informatica, Meccanica
Universita’ degli Studi di Palermo - Viale delle Scienze - 90128 PALERMO (ITALY)
{giuseppe.russo,roberto.pirrone,antonio.gentile}@unipa.it, {ariannapipitone77,francesca.anastasio}@gmail.com

Abstract—In the semantic web vision, ontologies are building blocks for providing applications with a high level description of the operating environment in support of interoperability and semantic capabilities. The importance of ontologies in this respect is clearly stated in many works. Another crucial issue to increase the semantic aspect of web is to enrich the level of expressivity of database related data. Nowadays, databases are the primary source of information for dynamical web sites. The linguistic data used to build the database structure could be relevant for extracting meaningful information. In most cases, this type of information is not used for information retrieval. The work presented in this paper deals with an attempt to enrich a database structure using linguistic information. The purpose is twofold: the proposed approach can be used either to validate the database structure linguistically or at least to enrich information retrieval with structural information. In this paper the first goal is pursued by the construction of the VEBO system that is used to validate database entity-relation diagrams through semi-automatic creation and enrichment of ontology.

I. INTRODUCTION

In this paper we address the issues related to semantic enrichment of a relational database structure using an ontology and WordNet [1]. A relational database is the result of a conceptual modeling process aimed at representing the domain entities and the relations between them. The model is depicted using the so-called as entity-relationship diagram (ERD). The database schema reflects the ERD and includes tables properties, properties datatypes, and length of types. Integrity constraints are semantic constraints applied to data, including entity integrity, referential integrity and user defined integrity. Our work is intended to enrich the overall linguistic quality of the database through a process able to incorporate semantic properties. To this purpose we have used WordNet, which is the largest lexicon database of English language. WordNet groups nouns, verbs, adjectives and adverbs into sets of cognitive synonyms called “synsets” in order to express a particular concept. Synsets are interlinked by conceptual relations able to express relation like hypernymy/hyponymy, antonymy, entailment, and meronymy/holonymy. Our approach involved the creation of a tool able to extract the ontological representation of a knowledge base. The tool, called VEBO (Validate ERD By OWL) allows to read the contents of a relational database, to derive its ontological representation. In turn such representation is processed by the construction of the ontology directed graph whose vertices represent classes, while edges represent the properties and indicate the existence of a relation between different classes. Our work is not intended solely to build the ontology from a database but it is focused on the interaction with a potential user to modify the database structure accordingly. In a few words, the VEBO system cycles through three macro-stages: ontology building from database, interaction with the user to enrich the database structure, and applying a set of rules defined purposely to rebuild the database structure starting from the modified ontology.

Many works have been presented in scientific literature to address the problem of ontology extraction from a relational database [2] [3]. Many of them define a set of rules to obtain the ontology representing the database structure [4], while other approaches are mainly focused on generating the ontology for database integration [5]. Other works use the interaction with WordNet to refine matching [6]. Very Few works define mapping rules from the ontology to a database schema [7] [8].

The rest of the paper is arranged as follows: next section presents the overall system and the approaches that have been adopted along its development. Section III presents some results for the proposed system. Conclusions and future works are reported in the last section.

II. SYSTEM PRESENTATION

Fig. 1 represents the three-steps cycle performed by the presented system.

Fig. 1. The main parts of the system and the output produced at each step

The first step is a reverse engineering process that is able to produce the ontology corresponding to the database structure. In the second step the ontology is analyzed to extract any
information about its structure, and to derive relational objects. In the second phase, users can enhance ontology adding some semantic elements through querying the WordNet database, which allows the user to search semantic relations for each term in the graph. The information produced by users through interactive graph modifying needs to be stored in order to be reused in a subsequent processing of the knowledge base. The system at this step produces an output file containing SQL statements to be used to perform the mapping. In the last step, changes are applied definitively, by executing SQL statements derived from the mapping rules. From now on, the database is ready for a new iteration. The process may be repeated endlessly.

Our approach uses a set of rules for the recognition of parts of the ontology, and a set of mapping rules that allow making changes to the database according to graph changes. All classes in the ontological representation identify entities in the relational model, and are then mapped to the database as tables, where the name of each table is the identification name of the entity that is the class name. Columns are represented in the ontological model as Functional Properties. Object Properties define relationships between entities and in mapping rules are handled as constraints.

A. Ontology extraction from database

The ontology extraction phase relies on a reverse engineering of the database creation process. The reverse engineering of the physical schema ascends to the conceptual model and the ERD that represents the structure of the database [9]. Once obtained the conceptual model, the system proceeds to transform the ERD in an ontological formalism through appropriate translation rules. The starting point consists in the retrieving the logical model of the database. The system queries the DBMS containing the DB to be processed to obtain information about the database schema. Such queries produce the list of tables along with their primary and foreign keys, the name and type of their fields, and the constraints posed on them such as uniqueness, referential integrity conditions, and so on. The next step is to trace the database design process back to the conceptual model, and the related ERD. We defined a ERD data structure that is made by a couple of vectors $E$ and $R$ that contain all the entities and relations respectively along with their features, constraints, and mutual references to each other, that is each $e_i \in E$ contains references to its impinging relations $r_j \in R$. Building the ERD is not a simple task, because what is a table in the logic scheme could be either a table or a relationship in the conceptual one. So the main difficulty in this phase, is to identify conceptual relations from tables in the logic scheme. We devised the following cases.

1) Tables without any foreign key are mapped onto entities in the conceptual scheme. Furthermore, the fields in such a table become attributes of the corresponding entity, and the primary keys of the table become the primary keys of the entity.

2) Tables with one or more foreign keys that differ from the primary key are converted into entities. Unlike the previous case, the foreign key fields are not translated into entity attributes; each of them becomes a one-to-many relationship between the entity that holds the referenced primary key and the entity being processed.

3) Tables where all primary keys are also foreign keys reference different tables. Such tables are converted to many-to-many relationships between the entities referenced by their foreign keys.

4) Tables where a foreign key refers to a primary key of the same table are converted to entities. The foreign key becomes a recursive one-to-many relationship which connects the entity with itself. The fields in the table that are not foreign keys become attributes of the corresponding entity.

Algorithm 1 reports describes the implementation of the whole procedure. Finally, the ERD structure is mapped onto OWL statements using rules derived from the works by Fahad [10] and Myroshnichenko [11]; such rules have been implemented using the Jena APIs.

Algorithm 1 Tables to ERD algorithm

```
1: procedure Tables_to_ERD(T)  \(\triangleright\) T: set of tables
2: \(R \leftarrow \emptyset\)  \(\triangleright\) At first there are no relationships
3: \(E \leftarrow \emptyset\)  \(\triangleright\) At first there are no entities
4: for \(i = 1 \ldots \text{size}(T)\) do
5: \(E \leftarrow E \cup \text{entity}(t_i)\)
6: end for
7: for \(i = 1 \ldots \text{size}(T)\) do
8: \(PK \leftarrow \text{primary_keys}(t_i)\)  \(\triangleright\) find PKs and FKS
9: \(FK \leftarrow \text{foreign_keys}(t_i)\)  \(\triangleright\) for the i-th table
10: if \(FK \neq \emptyset\) then
11: \(\text{drop}(e_i, E)\)  \(\triangleright\) remove the i-th entity
12: if \(FK \equiv PK\) then  \(\triangleright\) a M-M relationship
13: \(\{h, k\} \leftarrow \text{table_ref}(FK)\)
14: \(R \leftarrow R \cup \text{rel}_\text{MM}(e_h, e_k)\)
15: else  \(\triangleright\) a 1-M rel. for each foreign key
16: \(I \leftarrow \text{table_ref}(FK)\)
17: for \(l = 1 \ldots \text{size}(I)\) do
18: \(R \leftarrow R \cup \text{rel}_\text{1M}(e_I(l), e_i)\)
19: end for
20: end if
21: end if
22: end for
23: end procedure
```
the owl:ObjectProperty corresponding to the relationship under investigation. Such statements are inserted in the definition of the property’s range.

The ontology is represented graphically as a directed graph using the JUNG libraries [12]. Entities are drawn as nodes, while relationships are edges. Each relationship in this graph corresponds to an OWL Object Property, and the related edge is oriented from the property domain towards its range. The GUI provides the user with interaction techniques for changing the information represented in the graph by adding/removing its elements and/or querying the WordNet database to obtain rich linguistic descriptions their names. Two working copies of the database are ready for a new iteration.

C. Mapping to RDB

In the final step, the VEBO system uses the algorithm 2 reported in the following to compare the original and working copy of $E$ and $R$ vectors that is $PE$ and $PR$ are used as the intermediate data structure during each editing phase. After finishing to edit the graph they will contain all the modified elements.

Algorithm 2 Map_to_RDB algorithm

```
1: procedure Map_to_RDB(E, PE, R, PR) 
2:     S ← ∅  ▷ S: SQL statements list
3:     for i = 1 . . . size(PE) do ▷ Parse Entities
4:         if i > size(E) then
5:             E ← E ∪ pe_i
6:             S ← S ∪ mk_SQL("Create Table", pe_i)
7:         else if pe_i.name ≠ e_i.name then
8:             e_i.name ← pe_i.name
9:             S ← S ∪ mk_SQL("Rename Table", pe_i)
10:        else if pe_i.name == null then
11:            drop(e_i, E)
12:            S ← S ∪ mk_SQL("Delete Table", pe_i)
13:        end if
14:     end for
15:     for i = 1 . . . size(PR) do ▷ Parse Relationships
16:         if i > size(R) then
17:             R ← R ∪ pr_i
18:             S ← S ∪ mk_SQL("Add Constraint", pr_i)
19:         else if pr_i.name ≠ r_i.name then
20:             r_i.name ← pr_i.name
21:             S ← S ∪ mk_SQL("Rename Constraint", pr_i)
22:         else if pr_i.name == null then
23:             drop(r_i, R)
24:             S ← S ∪ mk_SQL("Delete Constraint", pr_i)
25:         end if
26:     end for
27: end procedure
```

III. APPLICATION EXAMPLES

The system has been utilized to evaluate databases produced in an undergraduate class of “Basi di dati” that is the introductory database course in our Department. Students produced their database; next, they were asked to use VEBO to validate the schema. We analyzed about twenty different databases for educational purposes. Some of them had well-formed ER schemas based on Chens original definition of the ER Model [13] with simple relational (i.e. first-order logic) semantics, while others were not. The tool has been used by students in different moments of their course, so we were able to take into account the modifications in their diagrams as long as they became more skilled in the database definition task. The user interface of the system is shown in the following figure (see Fig. 3) In all these cases, we observed a correct OWL generation. Some examples of the produced results are reported in the following.

1) Definition of a concept from a table. Database Name: Shop; table: Barcode.

```
Table 1 : barcode
Field 1 : barcode_ean(bpchar) pk
Field 2 : item_id(int4) fk[item]
Entity 1 : barcode
Attribute 1 : barcode_ean pk
```
2) Definition of a complex concept from a table. Database Name: Airport; table: Flight.
In this case the system defines the Object Property with the correct domain and range and also the list of attributes with domains and ranges as shown below.

Entity 6 : flight
Attribute 1 : date
Attribute 2 : flightnumber
Attribute 3 : company
Attribute 4 : startcode
Attribute 5 : arrivalcode

In this case the system defines the Object Property with the correct domain and range, and also the inverse Object Property. The produced report is the following:

Table 10 : PersonalBestTimes
Field 1 : member_id(int4) pk fk [Members]
Field 2 : stroke_code(int4) fk [Strokes]
Relation 10 : PersonalBestTimes
Cardinality : M − N

As regards the use of WorNet for enriching the lexicon to define database elements, the students were free to use such resource, and this didn’t affect our experimentation as regards the system operating way. Finally, SQL statements were produced according to the graphical interaction. As an example, we report the automatic definition of the relation execute_branch_id in the database BigCompany between the entities Branch and Transaction.

create or replace function nomechiave()
returns char varying(100000)
as $procS declare result char varying(100000);
begin SELECT into result constraint_name
FROM information_schema.key_column_usage
where column_name =’ execution_branch_id’;
return result;
end;
$procS language plpgsql;
create or replace function fun() returns void as
$declare a char varying(100000);
begins select "Alter table transaction
drop constraint nomechiave()’;
end;
$language plpgsql;
select fun();
drop function nomechiave(); dropfunction fun();

IV. CONCLUSIONS AND FUTURE WORKS

In this work the VEBO system has been presented that is able to enrich semantically a database schema. The systems operates by cycling through three main steps: extracting an OWL ontology from the database schema, adding new linguistic information using WordNet by means of a graphical interaction, and updating the schema according to the OWL modifications. This work, represents a first step for a wider system that will be able to perform deep annotation over the web using the proposed approach. We are building a service oriented architecture aimed at aligning two databases by means of the common ontology built over their schemas, and mapped onto an intermediate database structure. This is an important field of application especially for real world applications like small enterprise networks where users want to share information without modification of the IT infrastructure of the single enterprise. Future works are in the aforementioned direction. The results achieved in this work are currently used for building some of the web services needed by the overall architecture.

REFERENCES