Building Domain-specific Ontology from Data-intensive Web Site: An HTML Forms-based Reverse Engineering Approach

Sidi Mohamed Benslimane
Computer science Department, University of Sidi Bel Abbes, Algeria
Benslimane@univ-sba.dz

Mimoun Malki
Computer science Department University of Sidi Bel Abbes, Algeria
Malki_m@yahoo.com

MustaphaKamal Rahmouni
Computer science Department, University of Es-senia Oran, Algeria
Rahmouni@univ-oran.dz

Djamal Benslimane
LIRIS Laboratory
University of Lyon 1
Villeurbanne, France
Djamal.benslimane@liris.cnrs.fr

1. Introduction

The actual web has been moving away from static, fixed web pages to dynamically-generated at the time of user request. This kind of web site is called data-intensive web site [1], and usually realized using relational databases. Data-intensive web pages are characterized by an automated update of the web content and a simplified maintenance of the web design [2]. Nevertheless they suffer from two limitations. First, they form a hidden web since its content is not easily accessible to any automatic web content processing tools including the search engine indexing robots. Second the content of the database-driven web pages presented by using HTML is not machine-understandable. The next generation of the web, the semantic web, seeks to make information more usable by machines by introducing a more rigorous structure based on ontologies. Lately, ontologies have become the focus for research in several other areas, including knowledge engineering and management, information retrieval and integration, the semantic web, and e-commerce.
In this paper we propose a novel and integrated approach for a semi-automated migration of data-intensive web pages into semantic web and thus, make the web content machine-understandable. The best approach seems to rely on reverse engineering [3] rather than on semantic annotation [4], which is time consuming and error-prone.
Several researches have been done on relational databases reverse engineering, suggesting methods and rules for extracting entity-relationship and object models from relational databases [5, 6, 7, 8]. Recently, some approaches that consider ontologies as the target for reverse engineering have been proposed [2, 9, 10, 11, 12, 13]. Applicability of the existing approaches can be limited by the completeness of input information and its correctness.
This paper is organized as follows: In Section 2, we explain the overall reverse-engineering architecture and Section 3 details our proposed approach, Whereas Section 4 contains conclusion remarks and future works.

2. Our Approach

Our approach enriches the semantics of data by providing additional ontological entities [14]. It uses the information extracted from both HTML-forms and HTML-tables structure and instances as a data-intensive web application reverse engineering input. HTML-forms are often the most popular and convenient interfaces for entering, changing and viewing data in data-intensive web pages and, therefore, important information can be obtained by analyzing them.
The proposed architecture of our approach is depicted in Fig 1. The main components are: *The Extraction Engine* which consists of tree sets of *Extraction Rules*. *The Transformation Engine* which consists of two sets of *Transformation Rules. The Migration Engine* which consists of a set of *data migration rules.*
Our approach articulates around six steps performed by the six set of rules. To illustrate these steps, we'll use an HTML pages in Fig. 2. This returns as the query result for booking flight at http://www.airalgerie.dz.

3. Reverse engineering process

3.1. Analysis of HTML pages structure

The main goal of this phase is to understand the form meaning and explicit its structure by analyzing HTML forms to identify its components and their interrelationships and extract a form model schema. A form model schema was originally proposed, suitable for databases reverse engineering task [15].

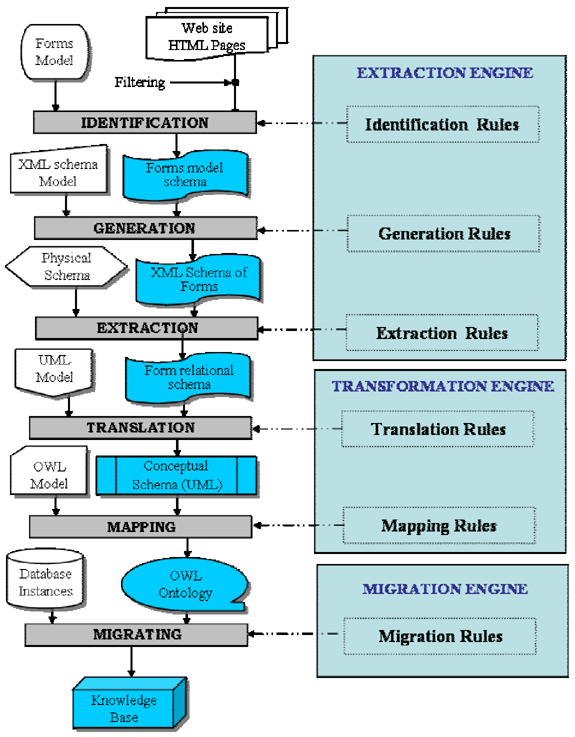
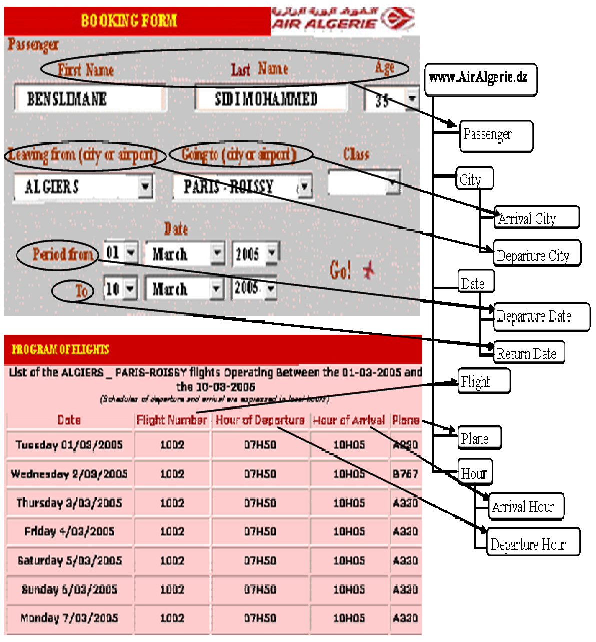


Fig. 1. Data-intensive web application reverse engineering Architecture


BOOKING FORM

Passenger

First Name	Last Name	Age
BENSLIMANE	SIDI MOHAMMED	35

Leaving from (city or airport)	Going to (city or airport)	Class
ALGIERS	PARIS - ROISSY	

Date

Period from	01	March	2005
To	10	March	2005

Go!

PROGRAM OF FLIGHTS

List of the ALGIERS _ PARIS-ROISSY flights Operating Between the 01-03-2005 and the 10-03-2005

(Schedules of departure and arrival are expressed in local hours)

Date	Flight Number	Hour of Departure	Hour of Arrival	Plane
Tuesday 01/03/2005	1002	07H50	10H05	A330
Wednesday 2/03/2005	1002	07H50	10H05	B767
Thursday 3/03/2005	1002	07H50	10H05	A330
Friday 4/03/2005	1002	07H50	10H05	A330
Saturday 5/03/2005	1002	07H50	10H05	A330
Sunday 6/03/2005	1002	07H50	10H05	A330
Monday 7/03/2005	1002	07H50	10H05	A330

Fig. 2. Personalised Ontology extraction from an HTML forms

3.1.1 The form model. The model allows abstracting any database form, that is, to make explicit its components, fields as well as objects, and their interrelationships. This model is similar but not identical to the model presented in [16]. Basically, this model consists of: *Form type*: Is a structured collection of empty fields formatted to communicate with databases. *Structural units*: Is a group of homogeneous pieces of information, that is, an object that groups closely related form fields. *Form instance:* Is an occurrence of a form type. This is the extensional part obtained when a form template is filled in with data. Fig 2 is an instance of the "Booking form" and "Program of flight" forms type. *Form fields:* Is an aggregation of a caption with its associated entry. Caption is pre-displayed on the form and serves as a clue as what is to be filled in by the respondent as well as a guide to enter or read it on the form. *Underlying source:* This is a structure of the relational database (i.e. a relational schema. *Relationships:* this is a connection between structural units that relates one structural unit to another. 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.

3.1.2. Form model schema identification rules. The rules below briefly summarize the transformation rules used to identify the form model constructs, they are part of the extraction engine component in Fig 1.
Rule 1: Identifying form instances. In order to clearly distinguish different kinds of information in the document, the web pages are usually split to multiple areas. Each area is crated using specific tags. For our approach we perform a filtering process and consider both the section between the open and closing <form> tag used to access and updates the relational databases and the section between the open and closing (<table>, <td>,<tr>,<li>,<ul>) tags returned as the query results and representing a particular view of the relational databases.
Rule 2: Identifying linked attributes. Linked attributes are identified by examining the HTML code for structural tags such as <thead> and <th> [17]. If the linked attributes aren't separated with the structural tags (merged data), we can use visual cues [18, 19]. This approach typically implies that there will be some separators that help users split the merged data.
Rule 3: Identifying structural units. To determine a logical structure of HTML page, we can use visual cues [18] E.g. the users might consider the FirstName, LastName, and Age in Fig. 2 as a whole group (passenger), just because they are specifications too.

Rule 4: Identifying relationships. The association can be indicated by the fact that the two structural units appear at the same page. If the two structural units come together, they might be logically related to each other. We would also identify an association relationship between two structural units using hyperlinks. By clicking on a hyperlink in one structural unit, we can go to another structural (possibly at another page).

3.2 Extraction of form XML-schema

Once the structure of the form type is extracted, the corresponding XML-schema is generated based on a set of translation rules between concepts of form models and those of the XML schema.

3.2.1. XML-schema generation rules. The translation is done systematically by the following set of transformation rules.
Rule 1: Each structural unit in the form type is translated as a complexType element in the corresponding XML schema. Example: the structural unit "passenger" is translated as follow:
<xsd: complexType name="passenger">
…
</xsd: complexType>
Rule 2: The rule 1 is applied recursively on the complex structural unit components. Example: The complex field "Period from (Day, month, Year)" in the "date departure" structural unit is translated as a ComplexeType element too.
<xsd: complexType name="PeriodFrom">
…
</xsd: complexType>
Rule 3: Each form field of the structural unit is translated in a sub-element of the corresponding complexeType element. The primitive type of the element is the one of the field. Example: the field "FirstName" is translated as a string type:
<xsd: element name="firstname" type="xsd:string"/>
Rule 4: If the structural unit contains some simple filling fields, the corresponding ComplexeType element takes as occurrence "minOccurs = 1" and "maxOccurs = 1"
Rule 5: If the structural unit contains some multiple filling fields, the corresponding ComplexeType element takes as maximum occurrence "maxOccurs = "*"".
Rule 6: The rules 4 and 5 are applied recursively on the form fields of each structural unit.

3.2.2. Construction of hierarchical structure of forms. In order to have a precise view of the hierarchical relationships of a form and to clearly understand its meaning and facilitate the interpretation and the extraction of the domain semantics, the form XML schema is transformed in a form hierarchical structure without loss of information. Formally, the hierarchical structure of a form is defined by the Tiplet (N, NT, L) where: N: represent no terminal nodes of the hierarchical structure, TN: represent terminal nodes of the hierarchical structure, L: represent the parent-child link between nodes. This process, which is automatic and transparent to the designer, constructs the hierarchical structure in four steps: Defining the root node (with level 0) whose name is the form's title; Transforming all complex elements into no-terminal nodes (with level 1). This step transforms recursively, the complex sub-elements into a no-terminal sub-nodes (with level 2, 3, etc); Transforming all simple elements and attributes into terminal nodes; Identifying the link type (mono-valued or multi-valued) between two nodes of the tree according to the occurrence value (maxoccurs=1 or maxoccurs=n).

3.3. Extraction of the domain semantics

The goal of this phase of extraction is to derive the relational sub-schemas of forms from their hierarchical structure and their instances according to the physical schema of the underlying database.

3.3.1. Form relations extraction. The identification of form relations and their primary keys respectively, consists of determining the equivalence and/or the similarity between structural units (nodes) of hierarchical structure and relations in the underlying database. This is a basis point from a reverse engineering point of view [15]. A node of a form hierarchical structure may be either: Equivalent to a relation in the underlying database, i.e., these two objects (node and relation) have a same set of attributes; Similar to a relation, i.e., its set of attributes is a subset of the one of the relation; A set of relations, i.e., its set of attributes regroups several relations in underlying database. Also, for dependent nodes (or form relation), primary keys are formed by concatenating the primary key of its parent with its local primary key. This process of identification is semi-automated because it requires the interaction with the analyst to identify objects that do not verify proprieties of equivalence and similarity.

While applying this process on the hierarchical structure of "Booking Form" and the physical relational schema of underlying database, we extract the following relational sub-schemas:
Passenger (PassengerID, FirstName, LastName, Age)
City (CityID); DepartureCity (CityID, Name)
ArrivalCity (CityID, Name) ; Date (DeparatueDate)
From the "program flights" form we identify the following relational sub-schemas:
DepartureHour (Dep_HourID, type)
ArrivalHour (Arr_HourID, type); Plane (PlaneID, Capacity) ; Flight (ID, DepartureCityID, ArrivalCityID, Dep_HourID, Arr_HourID, PlaneID)
From the relationships among hierarchical structure of "Booking Form" and "program flight" forms we identify the following relational sub-schemas:
Book (PassengerID, FlightID, DepartureDate, Class)
LeavingFrom (FlightID, DepartureCityID)
GoingTo (FlightID, ArrivalCityID)

3.3.2. Functional dependencies extraction. The extraction of functional dependencies from the extension of database has received a great deal of attention [20, 21, 22] In our approach we use the algorithm introduced by [15] to reduce the time for exacting functional dependencies by replacing database instances with a more compact representation that is, the form instances. While applying this algorithm on the sub-schema of "program of flights" and their instances, one finds the FDs: Flight.ID → DepartureCity.CityId ;.
Flight.ID → ArrivalCity.CityID

3.3.3. Inclusion dependencies extraction. The time of this process is more optimized with regard to the other approaches [15, 5] because the possible inclusion dependencies are verified by analyzing the form extensions which are more compact representation with regard to the database extension. In this algorithm, attributes of dependencies are the primary keys and foreign keys. Thus, the time complexity is reduced to the test of the inclusion dependency on the form instances. The set of the inclusion dependencies extracted is:
Book.FlightID << Flight.FlightID
Book.PassengerID << Passenger.PassengerID

3.4. Transforming the relational sub-schema of form into UML sub-schema

The task of Conceptual Modelling plays a crucial role in the process of information systems development. Conceptual models translate and specify the main data requirements of the user requirements in an abstract representation of selected semantics about some aspects of a real-world domain.

3.4.1 The Transformation process. The transformation is usually a collection of mapping rules that replace constructs in the form relational schema with conceptual entities in the UML model. Our rules are similar to those used in [8] to perform a transformation into an object oriented model.
Rule 1: Identification of object class. The general assumption is that each base relation is mapped into an object class. These object classes have the same attributes as those contained in the relations. The relation *Passenger (PassegerId, FirstName, SecondName, Age*) is translated to class.
Rule 2: Identification of binary association. The foreign keys of class-relation and the corresponding functional dependencies identify a binary association between class-relations. Therefore, this referential link is translated in binary association in the UML model. The target will be, in general, a role attribute typed by the other class.
Rule 3: Identification of association class. For every n-airy class-relation whose primary key is entirely composed of foreign keys, we create an association class between all the classes corresponding to the class-relation that foreign keys refer to. The relation *Book (PassegerID, FilightID, DepartureDate, Class)* is translated into Association-class.
*Rule 4: Identification of inheritance relationships***.** Extracting inheritance relationship from a relational schema usually requires behavioral information. Every pair of relations (R1,R2) that have the same primary key (noted X) and the corresponding inclusion dependencies (i.e., *R1:X << R2:X)* may be involved in an inheritance relationship, i.e., R1 "is-a" R2.
The Relations City, DepartureCity and ArrivalCity have the same primary key (CityID) and the corresponding inclusion dependencies:
DepartureCity.CityID << City.CityID;
ArrivalCity.CityID << City.CityID
Therefore City is a superclass and Departure_city and ArrivalCity are a subclass.

3.4.2 Integration of UML sub-schema. In the precedent phase of reverse engineering using forms as machine-analyzable source, object oriented sub-schemas was derived from relational sub-schemas. These object sub-schemas will be merging into a global object-oriented schema. We assume, in agreement with [23] that the integration schema process consists in two phases: comparison and conforming of schemas, and merging and restructuring of schemas. The comparison phase performs a parities comparison of objects (of the sub-schemas) and finds possible objects pairs, which may be semantically similar with respect to some proprieties, such as synonyms of equal primary key attribute and equivalent of classes. The conforming is a variety of analysts assisted techniques that are used to resolve conflicts and mismatched objects. The merging and restructuring phase generates an integrated schema from two component schemas that have been compared. For more details see [8].

3.5. Mapping of the global UML schema into OWL ontology

In this section we propose a rules set that establish a connection between UML and Web-based ontology language. The rules below briefly summarise the transformation rules used in the mapping between UML and OWL constructs.
Rule 1: Both OWL and UML are based on classes. So, in order to translate the UML class passenger, a class is declared by assigning a name to the relevant type. Example: <owl: class rdf: ID="Passenger"/>.
Rule 2: OWL distinguishes two kinds of properties, so called object properties and datatype properties. First, an instance of class ownedAttribute Property would translate as properties whose domain is Class and whose range is the type of Property. The UML ownedAttribut instance would translate to owl:ObjectProperty if the type of Property were a UML class, and owl:DatatypeProperty otherwise. Second an instance of a binary UML association translates directly to an owl:ObjectProperty.
Rule 3: N-ary relation among types T1...TN is formally equivalent to a set R of identifiers together with N projection functions P1,.., PN, where Pi:R -> Ti. Thereby N-ary UML associations are translated to OWL classes with bundles of binary functional properties.
Rule 4: In UML, a class can exist as a generalisation for one or more other classes. The generalisation element is synonymous with the OWL:subClassOf construct.
Both languages support the subclass and subproperties relationship. The translation from UML to OWL is straightforward. If <S, G> is an instance of an UML association generalisation (S is a subclassifier of G), then if both S and G are classes and TS, TG are respectively the types of the identifying owner property of S, G respectively, the OWL equivalent is the addition of the clause *<rdfs:subClassOf rdf:resource="TG"/>* to the definition of the OWL class TS. Similarly if S and G are both associations, the owl equivalent is the addition of the clause *<rdfs:subPropertyOf rdf:resource="G"/>* to the definition of the OWL object property S.
Rule 5: In OWL, a property when applied to a class can be constrained by cardinality restrictions on the domain giving the minimum *(minCardinality)* and maximum *(maxCardinality)* number of instances which can participate in the relation.

3.6. Migrating Data

Once the ontology is created, the process of *data migration* can start. The objective of this task is the creation of ontological instances (that form a knowledge base) based on the tuples of the relational database. The data migration process has to be performed in two phases based on the following rules: *Rule 1:* First, the instances are created. To each instance is assigned a unique identifier. This translates all attributes, except for foreign-key attributes, which are not needed in the metadata. *Rule 2: S*econd, relations between instances are established using the information contained in the foreign keys in the database tuples. This is accomplished using a mapping function that maps keys to ontological identifiers.

4. Conclusion and perspectives

We have developed a novel, integrated and semi-automated approach for extracting personalised ontology from data-intensive Web applications that can be applied to a broad range of today's business Web sites. The approach starts with transforming the HTML-forms into a form model schema. This model, allows the generation of an XML schema, witch permit the extraction of domain semantic and the construction of an UML conceptual model. Finally a mapping process is done between the conceptual structures and the OWL ontological constructs, which will be used thereafter for the migration of the database content into an ontology-based knowledge base. In the future, a combination between domain ontology and HTML-forms analysis technique can be exploited not only to extracted ontology but also to migrate from the current Web to the semantic Web.

5. References

[1] P. Fraternali, "Tools and approaches for developing data-intensive web applications: a survey", ACM Computing Surveys, Vol.31,227-263, 1999.

[2] L. Stojanovic, N. Stojanovic & R. Volz, "Migrating Data-intensive Web Sites into the Semantic Web", Proc. 17th ACM Symposium on Applied Computing (SAC), Madrid, Spain, 2002.

[3] -M. Erdmann, A. Maedche, H. Schnurr and S. Staab. "From Manual to Semi-automatic Semantic Annotation: About Ontology-based Text Annotation Tools", In: Proceedings of the Workshop on Semantic Annotation and Intelligent Content (COLING), P. Buitelaar &K. Hasida (eds.) (2000)

[4] R. Volz, S Handschuh, S. Staab, L. Stojanovic, N. Stojanovic. "Unveiling the hidden bride: deep annotation for mapping and migrating legacy data to the semantic Web", Journal of Web Semantics: science, services and agents on the Word Wide Web 1 (2004) 187-206.

[5] R.H.L. Chiang , T.M. Barron, V.C. Story . "Reverse engineering of relational databases: extraction of an EER model from a relational database". Data and Knowledge Engineering, 1994.

[6] M. Vermeer, P. Apers. "Object-oriented views of relational databases incorporation behaviour", Proceedings of the 4th International Conference on databases systems for Advanced Application (DASFAA), Singapore, Apr11-13, 1995, pp 26-35

[7] A. Behm, A. Geppert, K. Dittrich. (1997) "On the Migration of Relational Schemas and Data to Object-Oriented Database Systems". In Proceeding of the 5th Int. Conference on Re-Technologies for Information Systems, Klagenfurt, pp. 13-33, 1997.

[8] M. Malki, a. Flory, m.k. Rahmouni "Extraction of Object-oriented Schemas from Existing Relational Databases: a Form-driven Approach', INFORMATICA, Inter. Journal (Lithuanian Academy of Sciences) pp 47-72, Vol. 13(1), 2002.

[9] V. Kashyap, "Design and Creation of Ontologies for Environmental Information Retrieval" , Proc. 12th Workshop on Knowledge Acquisition, Modeling and Management (KAW), Canada, 1999.

[10] I. Astrova, "Reverse Engineering of Relational Databases to Ontologies" , Proc. 1st European Semantic Web Symposium (ESWS), Heraklion, Crete, Greece, LNCS, 3053, 2004, 327–341.

[11] N.Noy & M.Klein, "Ontology evolution: not the same as schema evolution." Report Number: SMI-2002-0926:2002

[12] Y.A Tijerino, D.W. Embly, D.W. Lonsdale, Y. Ding, G. Nagy "Towards Ontology Generation from tables". Kluwer Academic Publishers 2004

[13] I. Astrova, B. Stantic "An HTML Forms driven Approach to Reverse Engineering of Relational Databases to Ontologies", in proceeding of the 23rd IASTED International Conference on Databases and Applications (DBA), eds. M. H. Hamza, Innsbruck, Austria, 2005, pp. 246- 251

[14] S. Benslimane, M. Malki, D. M.K Rahmouni, "From data-intensive Web sites to ontology-based semantic web: A reverse engeineering approach", to appear in IASTED International Conference on Artificial Intelligence and Soft Computing (ASC 2005), September 12-14, 2005, Benidorm, Spain.

[15] M. Malki, M. Ayache, M.K. Rahmouni. "Rétro-ingénierie des Bases de Données Relationnelles: Approche Basée sur l'Analyse de Formulaires ». INFORSID'99. Toulon, France 1999.

[16] N. Mfourga. "Extracting entity-relationship schemas from relational databases: a form-driven approach." In Proc. of Working Conf. on Reverse EngineeringWCRE'97 1997.

[17] D. Embley, "Toward Semantic Understanding – An Approach Based on Information Extraction", Proc. 15th Australasian Database Conference (ADC), Dunedin, New Zealand, 2004.

[18] Y. Yang & H. Zhang, " HTML Page Analysis Based on Visual Cues", Proc. 6th International Conference on Document Analysis & Recognition (ICDAR), Seattle, WA, USA, 2001, 859–864.

[19] J. Wang & F. Lochovsky, "Data Extraction and Label Assignment for Web Databases", Proc. 12th International Conference on World Wide Web (WWW), Budapest, Hungary, 2003, 187–196.

[20] M. Anderson. "Extracting a E.R. schema from a relational database through reverse engineering". In Proc. of the 13th Inter. Conf. on the ERA'94, pp. 403–419, 1994.

[21] H. Mannila et al. *The Design of Relational Databases.* Addison-Wesley publishing, 1994.

[22] J.M. Petit, F. Toumani, J. Kouloumdjian. « Relational database reverse engineering: a method based on Query analysis". Inter. Jour of Cooperative Information System,4(2,3), 287–316, 1995.

[23] C. Batini, M. Lenzerini, S.B. Navathe (1986). "A comparative analysis of methodologies for database schema integration". *ACM Computing Surveys*, 18.