Towards Ontology Extraction from Data-Intensive Web Sites: An HTML Forms-Based Reverse Engineering Approach

Sidi Benslimane[1], Mimoun Malki[1], Mustapha Rahmouni[2], and Adellatif Rahmoun[3]
[1]Evolutionary Engineering and Distributed Information Systems Laboratory, Computer Science Department, University of Sidi Bel Abbes, Algeria
[2]Computer Science Department, University of Es-senia Oran, Algeria
[3]King Faisal University, CCS&IT, Hasa, KSA

Abstract: *The advance of the Web has significantly and rapidly changed the way of information organization, sharing and distribution. However, most of the information that is available has to be interpreted by humans; machine support is rather limited. 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 ontology. In this context we try to propose a novel and integrated approach for migrating data-intensive web into ontology-based semantic web and thus, make the web content machine-understandable. Our approach is based on the idea that semantics can be extracted from the structures and the instances of HTML forms which are the most convenient interface to communicate with relational databases on the current Web. This semantics is exploited to help build ontology.*

Keywords: *information search and retrieval, online information services, applications and expert systems, semantic web, data-intensive web, reverse engineering.*

Received August 4, 2006; Accepted November 20, 2006

1. Introduction

The *Semantic Web* is an extension of the current Web, where information and knowledge is formally well defined. Its goal is to enable machines and people to work in cooperation to achieve higher information processing power, and machines will become much better at processing and "understanding" the data that they merely display currently. 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 (i.e., e-commerce application). 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 HyperText Markup Language (HTML) is not machine-understandable. The simplicity and proliferation of the World Wide Web has taken the availability of information to an unprecedented level. 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, and thus resolve the second problem of data-intensive web pages. Ontology is one of the most important concepts in knowledge representation. It can be generally defined as shared formal conceptualization of particular domain between members of a community of interest, which help them exchange information [3]. Lately, ontologies have become the focus for research in several other areas, including knowledge engineering and management, information retrieval and integration, agent systems, the semantic web, and e-commerce. The availability of formal ontologies is crucial for the success of the semantic web. Nevertheless building ontologies is so costly that it hampers the progress of the semantic web activity. Manual construction of ontologies [4, 5] is a difficult, time-consuming and error-prone task and easily causes a knowledge acquisition bottleneck. Fully automated tools are still at the very early stage to be implemented. Therefore, the use of a semi-automatic ontologies extraction is seen as the practical short terms solution. Reverse engineering technique appears as an interesting solution to reach this objective. It's defined as a process of analyzing a "legacy" system to identify all the system's components and the relationships between them [6].

We propose in this paper a novel approach to reverse engineering data-intensive web application into ontology-based semantic web.

This paper is organized as follows. In section 2, we discuss some of the related works in reverse engineering relational databases into ontologies. Section 3 explains the overall reverse-engineering architecture and section 4 details our proposed approach. Section 5 presents a portal prototype implementation of the ontology construction approach. Finally, section 6 contains concluding remarks and suggests some future works.

2. Related Work

Several researches have been done on relational databases reverse engineering, suggesting methods and rules for extracting entity-relationship and object models from relational databases [6, 7, 8]. Recently, some approaches that consider ontologies as the target for reverse engineering have been proposed. These approaches fall roughly into one of the five categories:

- *Approaches based on an analysis of user queries:* E.g. Kashyap's approach [9] 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.
- *Approaches based on an analysis of relational schema*: e.g., Stojanovic et al's approach [2] provides a set of rules for mapping constructs in the relational database to semantically equivalent constructs in the ontology. These rules are based on an analysis of relations, keys and inclusion dependencies.
- *Approaches based on an analysis of tuples*: e.g., Astrova's approach [10] builds an ontology based on an analysis of relational schema. Since the relational schema often has little explicit semantics [11], this approach also analyzes tuples in the relational database to discover additional "hidden" semantics (e.g., inheritance). However, this approach is very time consuming with regard to the number of tuples in a relational database.
- *Approaches based on an analysis of HTML-table*: e.g., Tijerino's approach [12] based on conceptual modeling extraction technique attempts to understand a table's structure and conceptual content, discover the constraints that hold between concepts extracted from the table, match the recognized concepts with ones from a more general specification of related concepts, and merge the resulting structure with other similar knowledge representations. However, this approach requires auxiliary information including dictionaries and lexical data (WordNet, natural language parsers, and data frames library).
- *Approaches based on an analysis of HTML-forms*: e.g., Astrova's approach [13] constructs an ontology based on an analysis of HTML-forms by analyzing the HTML-forms to extract a form model schema, transforming the form model schema into ontology and creating ontological instances from data contained in the pages. The drawback of this approach is that this approach does not offer any way to the identification of inheritance relationship which is a significant aspect in the ontology construction.

3. Our Approach

To overcome the drawbacks of the approaches described above, we propose a novel approach for reverse engineering data-intensive web sites into ontology-based semantic web. Our approach is based on the idea that semantics of the relational database can be extracted by analyzing the related HTML pages. This semantics are augmented with those captured in the relational schema to build ontology. Unlike [14] that uses frame logic as an ontology description language; this paper adopts the latest standard recommended by World Web Consortium (W3C), namely Ontology Web Language (OWL).

3.1. Motivations

The uses of information extracted from both HTML forms used for sending user queries and HTML-tables[1] returned as the query results can be supported by the following arguments:

- HTML-forms are often the most popular and convenient interfaces for entering, changing and viewing data in the actual data-intensive web pages.
- Studying and analyzing an HTML-forms and their relationship can reveal many data dependencies and mapping.
- HTML-forms are structured collections of fields formatted to communicate with the relational database. Therefore, data contained in the forms is usually structured, while the structure of the relation databases is often unknown in advance [15].
- Field names in HTML-forms are usually more explicit and meaningful than the names of corresponding attributes in the relational databases.
- Often HTML-forms are accompanied with instructions, which provide additional information about organisation's data and their behaviour parts.

3.2. Proposed Architecture

This section describes the ontology building framework. It gives a description of the architecture

[1] In what follows, HTML-forms nominates both HTML-forms and HTML-tables

components, as shown in Figure 1.

- *The Extraction Engine* consists of three sets of extraction rules. The first set of rules analyses the HTML pages to identify constructs in the form model schema. The second set of rules permits the extraction of a form XML schema from the constructs of the form model schema, whereas the third set of rules derives the domain semantics by extracting the relational sub-schemas of forms and their dependencies.
- *The Transformation Engine* consists of two sets of transformation rules. The first set of rules transforms the relational sub-schemas of forms into conceptual schema based on UML model. The second set of rules translates the modelling language constructs into OWL ontological concepts.
- *The Migration Engine* consists of a set of data migration rules responsible of the creation of ontolological instances from the relational tuples.

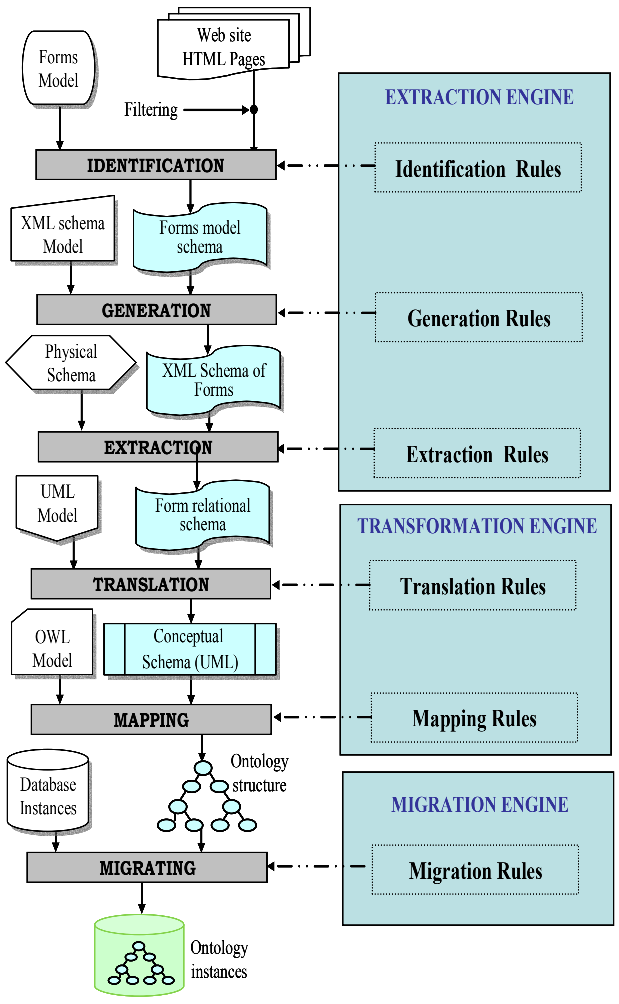
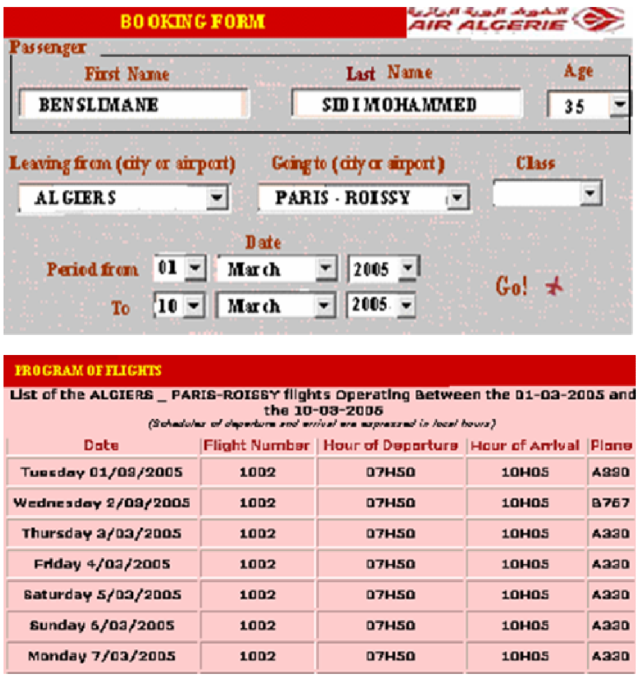


Figure1. Data-intensive web site reverse engineering architecture.

4. Reverse Engineering Process

For illustration purposes, we use the Algerian Airline web site *http://www.airalgerie.dz*. Two HTML pages among several are shown in Figure 2: "Booking" form and "Flight Program" table.



BOOKING FORM

AIR ALGERIE

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 10-03-2006

(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

Figure 2. HTML pages along with HTML-form and HTML-table.

4.1. 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 (both their structure and data they contain) to identify its components and interrelationships and extract a form model schema.

4.1.1. The Form Model

A form model schema was originally proposed, suitable for databases reverse engineering task [16]. The model allows abstracting any database form to make explicit its components, fields, and their interrelationships. Basically, this model consists of:

A. *Form type*: is a structured collection of empty fields that are formatted in a way that permits communication with the database. A particular representation of a form type is called form template that suggests three basic components namely title, captions, and entries.

B. *Structural units*: correspond to objects that closely group related fields in a form.

C. *Form instance*: is an occurrence of a form type. This is the extensional part obtained when a form template is filled in with data. Figure 2 is an instance of the "Booking form" and "Program of flight" forms type.

D. *Form fields*: consists of a caption and its associated entry. Each entry is generally linked to a table's name as per the table names in the underling database. The values that a form field displays/receives are provided by (or stored in) the linked-attribute. We distinguish three types of fields: filling fields (*text, checkbox, radio, textarea* *attributes*); selection fields (*select* attribute); and link fields (*href* attribute).

E. *Underlying source*: corresponds to the structure of the relational database (i.e., *relational* schema) in terms of relations and attributes along with their data types.

F. *Relationships*: 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.

G. *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.

4.1.2. Form Model Schema Identification Rules

The following rules summarize the mechanisms that permit identifying a form model's constructs using a relational schema as input. These rules populate the extraction engine of Figure 1.

A. *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 created 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.
- 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.

B. *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 use visual cues [18, 19]. This approach typically implies that there will be some separators (e.g. blank areas) that help users split the merged data.

C. *Rule 3: Identifying structural unit*. To determine the logical structure of an HTML page (i.e., the real meaning of the page, as it is understood by users), we can use visual cues [18]; e.g., users might consider firstName, lastName, and age in Figure 2 as a whole group (Passenger), just because they all are specifications.

D. *Rule 4: Identifying relationship.* Relationship can be indicated by the fact that two structural units appear in the same page. If both structural units come together, they might be logically related. Since the relational database information typically does not reside in a single HTML page, we try to find relationships using hyperlinks. Hyperlinks can be interpreted, in many cases, as semantic relations between structural units.

4.2. 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.

A. *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 1 is applied recursively on the complex structural unit components.

B. *Rule 2***:** each form field of the structural unit is translated into a sub-element of the corresponding complexeType element. The primitive type of the element is one of the field.
Example: the field "FirstName" is translated as a string type:

<xsd: element name=”firstname” type=”xsd:string”/>

C. *Rule 3*: if the structural unit contains some simple filling fields (e.g., text tag), the corresponding ComplexeType element takes "minOccurs = 1" and "maxOccurs = 1" as occurrence.

D. *Rule 4*: if the structural unit contains some multiple filling fields (e.g., multiple attribute), the corresponding ComplexeType element takes "maxOccurs = "*"" as maximum occurrence.

Rules 3 and 4 are applied recursively on the form fields of each structural unit. While applying the rules mentioned above on the "Booking form" type structure, we obtain the XML-schema as shown in Figure 4.

4.3. 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.

First, the relations and their primary keys are respectively identified with regard to both structural units (nodes) of form and underlying database, then the functional and inclusion dependencies are extracted through both the forms structure and instances.

```
<?xml version=″1.0″?>
<xsd:schema=xmlns:xsd=″http://www.w3.org/2001/XMLSchema″>
<xsd:complexType name=″BookingForm″>
< xsd:attribute name=″class″ type=″xsd:integer″/>
      <xsd:complexType name=″Passenger″ type=
        ‶xsd: ″PassengerID″ maxOccurs=″1″/>
      <xsd:complexType name=″City″ type=
        ‶xsd: ″CityID″ maxOccurs=″1″/>
      <xsd:complexType name=″Date″ type=
        ‶xsd: ″DateID″ maxOccurs=″1″/>
      <xsd:complexType name=″ PassengerID″>
            <xsd:element name=″FirstName″ type=″xsd:string″/>
            <xsd:element name=″LastName″ type=″xsd:string″/>
            <xsd:element name=″Age″ type=″xsd:integer″/>
       <xsd:complexType/>
      <xsd:complexType name=″ CityID″>
            <xsd:element name=″LeavingFrom″ type=″xsd:string″/>
            <xsd:element name=″GoingTo″ type=″xsd:string″/>
      <xsd:complexType/>
      ...
      <xsd:complexType/>
 <xsd:complexType/>
<xsd:schema/>
```

Figure 4. XML schema of "Booking form".

4.3.1. 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 [8].

A node of a form hierarchical structure may be either:

D. Equivalent to a relation in the underlying database, i.e., these two objects (node and relation) have a same set of attributes.
E. Similar to a relation, i.e., its set of attributes is a subset of the one of the relation.
F. A set of relations, i.e., its set of attributes regroups several relations in underlying database.

In addition, for dependent nodes (or form relation), primary keys are formed by concatenating the primary key of its parent with its local primary key. This identification process 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)

4.3.2. 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 [8] 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 functional dependencies:

Flight.ID → *DepartureCity.CityID*
Flight.ID → *ArrivalCity.CityID*

4.3.3. Inclusion Dependencies Extraction

In our approach, we formulate possible inclusion dependencies between relations' key of relational sub-schema of form. The time of this process is more optimized with regard to the other approaches [22, 6] 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*

4.4. Transforming the Relational Sub-Schema of Form into UML Sub-Schema

The transformation is usually a collection of mapping rules that replace constructs in the form relational schema with (semantically equivalent) conceptual entities in the Unified Modeling Language (UML) model, as shown in Figure 5.

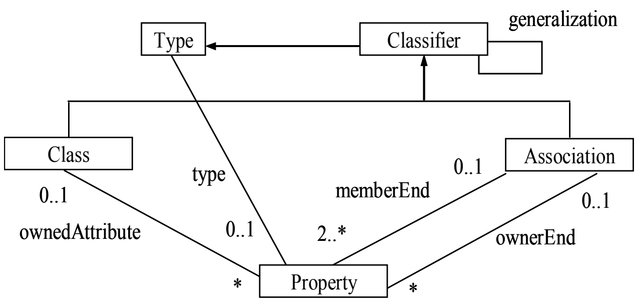


Figure 5. Key aspects of UML class diagram.

Basically, the process uses the constructs generated from the precedent step as the main input (i.e. form relational schema, functional dependencies and inclusion dependencies). It goes through four steps: (1) identification of classes, (2) identification of binary association, (3) identification of *n*-ary association, (4) identification of inheritance relationships. This process is based on the classification of relations. Relation can be classified into one of the three categories.

A. *Base relation*: if a relation is independent of any other relation in a form relation schema.
Example: *Passenger (PassengerID, FirstName, LastName, Age).*

B. *Dependent relation*: if a primary key of a relation depends on another relation's primary key.
Example: *Book (PassengerID, FlightID, DepartureDate, Classe).*

C. *Composite relation*: if it is neither base nor dependent.
Example: *Flight (FlightID, DepartureCityID, ArrivalCityID, Dep_HourID, Arr_HourID, PlaneID).*

4.4.1. The Transformation Rules

Our rules are similar to those used in [8] to perform a transformation into an object oriented model.

A. *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* is translated to class shown if the Figure 6.

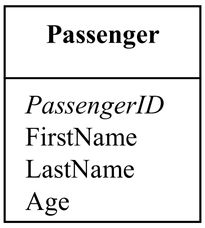


Figure 6. UML class.

B. *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.

While applying this transformation rule on the two class-relations *Flight* and *DepartureCity* and their functional dependencies:

Flight.ID $\rightarrow$ *DepartureCity.DepartureCityID,* We generate the following object schema, as shown in Figure 7.

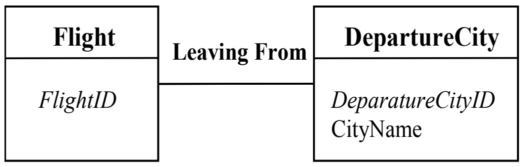


Figure 7. Binary UML association.

C. *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* is translated into association-class as show in Figure 8.

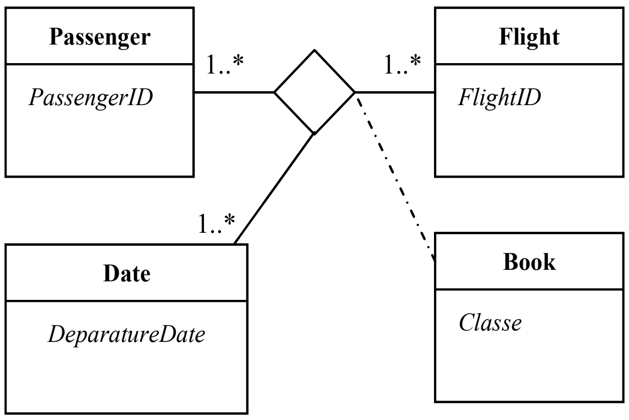


Figure 8. *n*-ary UML association.

D. *Rule 4: Identification of inheritance relationships.* Extracting inheritance relationship from a relational schema usually requires behavioural 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.*
In Figure 9, 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 *DepartureCity* and *ArrivalCity* are a subclass.

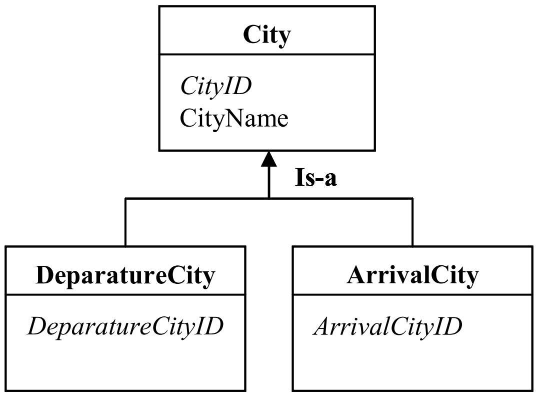


Figure 9. Inheritance relationship.

4.4.2. Integration of UML Sub-Schema

In the precedent phase, relational sub-schemas were transformed into object oriented sub-schemas. These object sub-schemas will be merging into a global object-oriented schema that represents the whole underlying database. However, we apply the techniques of integration schema. We assume, in agreement with [23] that the integration schema process consists in two phases: comparison and merging 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 (name of attribute and class) of equal primary key attribute and equivalent of classes. The merging phase generates an integrated schema from two component schemas that have been compared. Figure 10 presents the integrated UML global schema.

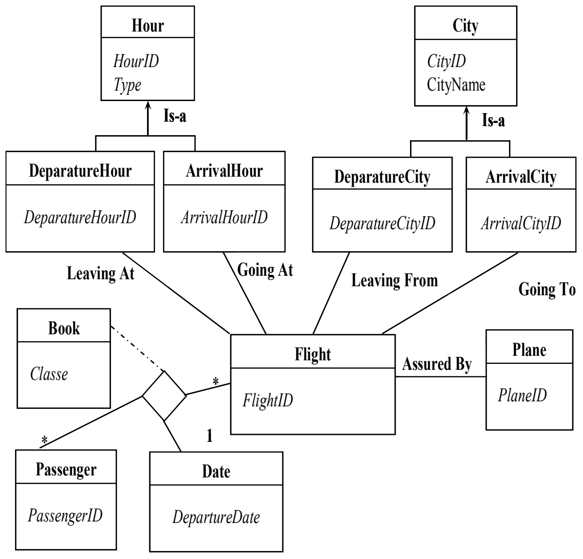


Figure 10. Global UML schema.

4.5. UML into OWL Mapping Rules

UML conceptual models can be translated into other ontology languages like Resource Description Framework Schema (RDFS), Web Ontology Language (OWL) or even in to object oriented database systems. Some proposals which address the problem of reusing knowledge previously specified as UML in a form that allows it to be 'on the Web' and can be reasoned with [24, 25, 26].

The rules below briefly summarise the transformation rules used in the mapping between UML and OWL constructs. OWL is designed for using by applications that need to process the content of information instead of just presenting information to humans. OWL facilitates greater machine interpretability of web content than that supported by XML, RDF, and RDFS by providing a standard language for the representation of ontologies on the World Wide Web.

A. *Rule 1*: Both OWL and UML are based on classes. So, in order to translate the UML class of Figure 6, an OWL class is declared by assigning a name to the relevant type.

Example: *<owl: class rdf: ID="Passenger"/>.*

B. *Rule 2:* By default a property is a binary relation between thing and thing. It comes from two different sources in the UML model:

- 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. Table 1 shows the translation of classes, as shown in Figure 7.

- Second an instance of a binary UML association translates directly to an *owl:ObjectProperty*. The translation of the binary association of Figure 7 is given in Table 2.

Table 1. Classes translation.

UML class	Owned Property	Type of Owned Property	OWL equivalent
Flight	FlightID	Integer	<owl:DatatypeProperty rdf:ID="FlightID"> <rdfs:domain rdf:resource="#Flight"/> <rdfs:range rdf:resource= "http://www.w3.org/2001/XMLSchema#integer"/> </owl:DatatypeProperty>
DepartureCity	CityID	Integer	<owl:DatatypeProperty rdf:ID="CityID"> <rdfs:domain rdf:resource="#DepartureCity"/> <rdfs:range rdf:resource= "http://www.w3.org/2001/XMLSchema#integer"/> </owl:DatatypeProperty>
	CityName	String	<owl:DatatypeProperty rdf:ID="CityName"> <rdfs:domain rdf:resource="# DepartureCity"/> <rdfs:range rdf:resource= "http://www.w3.org/2001/XMLSchema#string"/> </owl:DatatypeProperty>

Table 2. Binary association translation.

UML Association	Member 1 Property Type	Member 2 Property Type	OWL equivalent
LeavingFrom	Flight	DepartureCity	<owl: objectProperty rdf:ID="LeavingFrom"> <rdfs:domain rdf:resource="#Flight"/> <rdfs:range rdf:resource= "#DepartureCity"/> </owl:objectProperty>

C. *Rule 3: n*-ary relation among types $T_1...T_N$ is formally equivalent to a set R of identifiers together with N projection functions $P_1...P_N$, where P_i: R → T_i. Thereby *n*-ary UML associations are translated to OWL classes with bundles of binary functional properties, as shown in Figure 8.

D. *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. The inheritance relationship in the Figure 9 is translated as follow:

<owl:class rdf:about="DepartureCity">
<owl:subClassOf rdf:resource="#City" />
</owl:class>
<owl:class rdf:about="ArrivalsCity">
<owl:subClassOf rdf:resource="#City" />
</owl:class>

E. *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. In UML an association can have minimum and maximum cardinalities (multiplicity) specified for any of its ends. OWL allows individual-valued properties (ObjectProperty) to be declared in pairs, one the inverse of the other. So if a binary UML association has a multiplicity on a navigable end, the corresponding OWL property will have the same multiplicity. If a binary UML association has a multiplicity on its both ends, then the corresponding OWL property will be an inverse pair, each having one of the multiplicity declarations.

4.6. 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:

A. *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.

B. *Rule 2:* Second, 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. Figure 11 illustrates an example result of the data migration process from the Table 3.

Table 3. Relational model instances.

Plane		
Plane ID	Capacity	Company ID
A330	150	1
B767	200	2

Company	
Company ID	Company Name
1	Air Algeria
2	Air France

5. Implementation

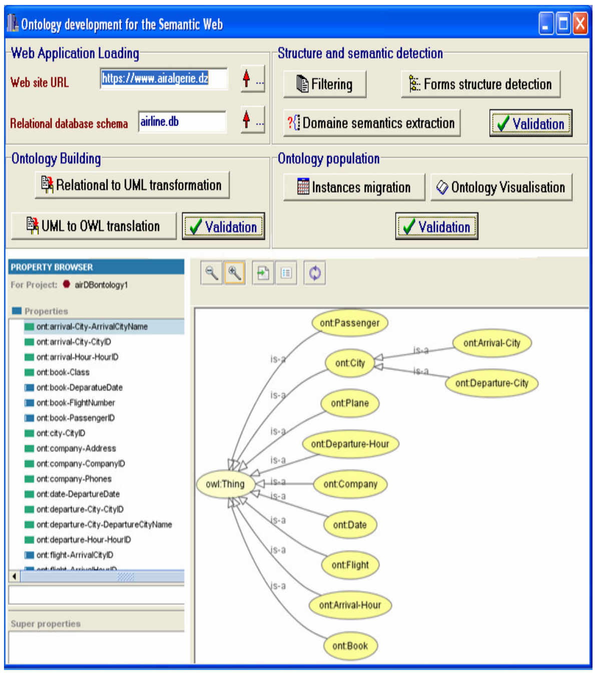
In this section, we present some experiments we performed to assess the effectiveness of the proposed approach to semi-automatically build OWL ontology from relational database using the related HTML-forms. The main purpose of the experiments is to evaluate the effectiveness of the ontology development rules presented in the previous sections, and to verify that the proposed approach can contribute in helping the user in performing the labour intensive ontology development task. Since the construction of OWL ontology from an enriched relational schema is characterized by the specific rules, the generation of the ontology can be automated.

```
<?xml version="1.0"?>
<rdf:RDF
  <owl:Ontology rdf:about=""/>
  <owl:Class rdf:ID="Company"/>
  <owl:Class rdf:ID="Plane"/>
...
  <Company rdf:ID="Company1">
   <CompanyId rdf:datatype=
    "http://www.w3.org/2001/XMLSchema#int"
    >1</CompanyId>
    <CompanyName rdf:datatype=
    "http://www.w3.org/2001/XMLSchema#string"
    >Air Algerie</CompanyName>
  </Company>
  <Plane rdf:ID="Plane1">
    <capacity rdf:datatype=
    "http://www.w3.org/2001/XMLSchema#int"
     >150</capacity>
    <PlaneId rdf:datatype=
    "http://www.w3.org/2001/XMLSchema#string"
     >A330</PlaneId>
    <Possede rdf:resource="#Company1"/>
  </Plane>
...
</rdf:RDF>
```

Figure 11. Ontology instances.

5.1 Prototype

A prototype is developed using Java (j2sdk 1.4.2) and Jena 2.1, the Java API for ontology development and processing, as shown in Figure 12. The prototype has been implemented in order to experiment and verify that the proposed approach is an applicable solution. Our tool has a user friendly GUI to perform the ontology development process and to produce ontology stored in an OWL file.


Ontology development for the Semantic Web

Web site URL: https://www.airalgerie.dz

Relational database schema: airline.db

PROPERTY BROWSER

For Project: airDBontology1

Properties
- ont:arrival-City-ArrivalCityName
- ont:arrival-City-CityID
- ont:arrival-Hour-HourID
- ont:book-Class
- ont:book-DeparatueDate
- ont:book-FlightNumber
- ont:book-PassengerID
- ont:city-CityID
- ont:company-Address
- ont:company-CompanyID
- ont:company-Phones
- ont:date-DepartureDate
- ont:departure-City-CityID
- ont:departure-City-DepartureCityName
- ont:departure-Hour-HourID
- ont:flight-ArrivalCityID

Super properties

Figure 12. Snapshot of the ontology development tool.

The Web site URL, the relational schema and other parameters such as information for the database connection (e.g. JDBC driver, database URL), base URI and ontology URI of the output OWL ontology are provided in an input configuration file. The output ontology can be formalized in the following standard formats: OWL, RDF/XML, RDF/XML-ABBREV, N3 and N-Triples.

5.2. Experimental Evaluation

In order to evaluate our approach, we have performed two experiments on tourism domain. In the first experiment, we analyzed an airlines company Web site[2]. The constructs of the obtained OWL ontology are presented in Table 4. The results are compared to the tutorial ontology for a Semantic Web of tourism[3].

To evaluate the quality of the ontology development process, we compare the OWL ontology constructs (correctly extracted: C, and incorrectly extracted: I) returned by the automatic extraction process with manually determined constructs (M) in the tutorial

Table 4. Results from the ontology development process using an airlines company web site.

OWL Ontology constructs	Constructs in the tutorial Ontology (M)	Constructs extracted Correctly (C)	Constructs extracted incorrectly (	Recall Ratio (C/M)	Precision Ratio C/(C+I)
Classes	30	15	1	0.50	0.94
Objects properties	16	09	1	0.56	0.90
Datatype properties	77	34	3	0.44	0.92

Ontology for a Semantic Web of tourism. Based on the cardinalities of these sets, the following quality measures are computed.

Precision= (C)/(C+I), is the faction of the automatic discovered constructs which are correct.

Recall= (C)/ (M), is the fraction of the correct constructs (the set M) which has been discovered by the ontology development process.

The ontology development process was rather successful, with average recall and precision ratios of 94% and 92% respectively, as shown in Table 5. The results obtained with the use of the second experiment could be much better if more Web sites covering a large part of the tourism activities were used as input. The low recall ratio is not so much a consequence of bad ontology development approach, but much more due to the restricted domain knowledge covered by the Web site itself. In the second experiment, we have conducted experiments on three tourism Web sites related respectively to flights4, hotel[4] and leisure[5] activities.

Table 5. Results from the ontology development process using three Web sites related.

OWL Ontology constructs	Constructs in the tutorial ontology (M)	Constructs extracted correctly (C)	Constructs extracted incorrectly (I)	Recall Ratio (C/M)	Precision Ratio C/(C+I)
Classes	30	28	2	0.93	0.93
Objects properties	16	15	2	0.94	0.88
Datatype properties	77	73	5	0.95	0.94

6. Conclusion and Perspectives

Research on ontology is becoming increasingly widespread in the computer science community. The major difficulties in building ontology are a mass of handwork. So, the use of a semi-automatic ontologies extraction is seen as a practical and good solution. In this paper we focus on the problem of automating the generation of domain ontologies, at least partially, by applying reverse engineering technique. We present the complete details of the process of semi-

[2] http://www.britishairways.com
[3] http://protege.stanford.edu/plugins/owl/owl-library/travel.owl.

[4] http://www.hm-usa.com
[5] http://www.travelandleisure.com

automatically create OWL ontology corresponding to the content of relational database based on the analysis of its related HTML-forms. Our approach can be used for migrating HTML pages (especially those that are dynamically generated from a relational database) to the ontology-based Semantic Web. The main reason for this migration is to make the relational database information that is available on the Web machine-processable, and reduce the time consuming task of ontology creation.

However, in the most circumstances, the obtained ontological structure is coarse. In addition, some semantics of obtained information need to be validated. So refining obtained ontological structure is necessary. Because existing repositories of lexical knowledge usually includes authoritative knowledge about some domains, we suggest as future work refining obtained ontology according to them, especially machine-readable dictionaries and thesauri (e.g., WordNet).

References

[1] Anderson M., "Extracting a E.R. Schema from a Relational Database through Reverse Engineering," *in Proceedings of the 13th International Conference on the (ERA'94)*, pp. 403-419, 1994.

[2] Astrova I., "Reverse Engineering of Relational Databases to Ontologies," *in Proceedings of the 1st European Semantic Web Symposium (ESWS),* Heraklion, Greece, LNCS, 3053, pp. 327-341, 2004.

[3] Astrova I. and Stantic B., "An HTML Forms Driven Approach to Reverse Engineering of Relational Databases to Ontologies," *in Proceedings of the 23rd IASTED International Conference on Databases and Applications (DBA),* Innsbruck, Austria, pp. 246- 251, 2005.

[4] Baclawski M., Kokar M., Kogut P., and Hart L., "Extending UML to Support Ontology Engineering for the Semantic Web," *in Proceedings of the Fourth International Conference on UML (UML'2001),* Toronto, 2001.

[5] Batini C., Lenzerini M., and Navathe S., "A Comparative Analysis of Methodologies for Database Schema Integration," *ACM Computing Surveys,* vol. 18, no. 4, pp. 323-364, 1986.

[6] Behm A., Geppert K., and Dittrich K., "On the Migration of Relational Schemas and Data to Object-Oriented Database Systems," *in Proceedings of the 5th International Conference on Re-Technologies for Information Systems,* pp. 13-33, 1997.

[7] Benslimane S., Malki M., and Amar D., "Automated Migration of Data-Intensive Web Pages into Ontology-Based Semantic Web: A Reverse Engineering Approach," *in Meersman R., Tari Z. et al., (eds.),*ODBASE, vol. 2, LNCS 3761, pp. 1640 – 1649, Springer Verlag, 2005.

[8] Chiang R., Barron T., and Story V., "Reverse Engineering of Relational Databases: Extraction of an EER Model from A Relational Database," *Data and Knowledge Engineering*, vol. 12, no. 2, pp. 107-142, 1994.

[9] Choobineh J., "A form-based Approach for Database Analysis and Design," *Communication of the ACM*, vol. 35, no. 2, pp. 108-120, 1992.

[10] Cranefield S., "UML and the Semantic Web," *in Proceedings of the International Semantic Web Working Symposium*, Palo Alto, 2001.

[11] Embley D., "Toward Semantic Understanding – An Approach Based on Information Extraction," *in Proceedings of the 15th Australasian Database Conference (ADC)*, Dunedin, New Zealand, 2004.

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

[13] Falkovych K., Sabou M., and Stuckenschmidt H., "UML for the Semantic Web: Transformation-Based Approaches," *in B. Omelayenko and M. Klein, (eds.), Knowledge Transformation for the Semantic Web*, pp. 92-106, 2003.

[14] Fraternali P., "Tools and Approaches for Developing Data-intensive Web Applications: a Survey," *ACM Computing Surveys*, vol. 31, no. 3, pp. 227-263, 1999.

[15] Gruber T., "Toward Principles for the Design of Ontologies used for Knowledge Sharing," *Human Computer Studies*, vol. 43, no. 5-6, pp. 907-928, 1995.

[16] Kashyap V., "Design and Creation of Ontologies for Environmental Information Retrieval," *in Proceedings of the 12th Workshop on Knowledge Acquisition, Modeling and Management (KAW)*, Banff, Alberta, Canada, 1999.

[17] Malki M., Ayache M., and Rahmouni M., "Rétro-ingénierie des Bases de Données Relationnelles: Approche Basée sur l'Analyse de Formulaires, " *in Actes du XVIIème Congrès INFORSID,* Toulon, France, 1999.

[18] Malki M., Flory A., Rahmouni M., "Extraction of Object-oriented Schemas from Existing Relational Databases: a Form-driven Approach," *INFORMATICA, International Journal (Lithuanian Academy of Sciences)*, vol. 13, no. 1, pp. 47-72, 2002.

[19] Mannila H. and Räihä K., *The Design of Relational Databases,* Addison-Wesley, 1994.

[20] Noy N. and Klein M., "Ontology Evolution: Not the Same as Schema Evolution," *Knowledge and Information Systems,* vol. 6, no. 4, pp. 428-440, 2004.

[21] Petit J., Toumani F., and Kouloumdjian J., "Relational Database Reverse Engineering: a Method Based on Query Analysis," *International Journal of Cooperative Information System*, vol. 4, no. 2, pp. 287-316, 1995.

[22] Stojanovic L., Stojanovic N. , and Volz R., "Migrating Data-intensive Web Sites into the Semantic Web," *in Proceedings of the 17th ACM Symposium on Applied Computing (SAC)*, Madrid, Spain, 2002.

[23] Tijerino Y., Embley D., Lonsdale D., Ding Y., and Nagy G., *Towards Ontology Generation from Tables*, Kluwer Academic Publishers, 2004.

[24] Yang Y. and Zhang H., "HTML Page Analysis Based on Visual Cues," *in Proceedings of the 6th International Conference on Document Analysis & Recognition (ICDAR)*, Seattle, USA, 2001.

[25] Volz R., Handschuh S., Staab S., Stojanovic L., and Stojanovic N., "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,* vol. 1, no. 2, pp. 187-206, 2004.

[26] Wang J. and Lochovsky F., "Data Extraction and Label Assignment for Web Databases," *in Proceedings of the 12th International Conference on World Wide Web (WWW)*, Budapest, Hungary, 2003.

Sidi Benslimane is a lecterer in the Department of Computer Science, Sidi Bel Abbes University, Algeria. He received the MSc degree in computer science from Sidi Bel Abbes University, Algeria, in 2001. He is a PhD candidate in Computer Science Department at Sidi Bel Abbes University from December 2002. His research interests include semantic web, web engineering, ontology engineering, and information systems.

Mimoun Malki is an assistant professor at the Department of Computer Science at Sidi Bel Abbes University. He received the PhD degree in computer science from Sidi Bel Abbes University, Algeria, in 2003. He heads the Evolutionary Engineering and Distributed Information Systems Laboratory. His research interests include, knowledg management, information retrieval, ontology engineering, semantic web, web services, and soft computing systems.

Mustapha Rahmouni is a professor at the Computer Science Department of the University of Oran Es-Sénia, Algeria. He received the PhD degree in operational research from Southampton University UK, in 1987. He heads the Information Systems Laboratory and the local Doctoral School on STIC. His research interests include formal specifications, information management and integration, process modelling, and knowledge management.

Abdellatif Rahmoun is an associate professor at King Faisal University, KSA. He received the PhD degree in computer science from Sidi Bel Abbes University, Algeria, in 1998. He has been involved in several research projects and teaching in Algeria. His research interests include, logic, genetic algorithms and genetic programming, neural networks and applications, e-learning, e-commerce, and e-business.