ON-THE FLY ANNOTATION OF DYNAMIC WEB PAGES

Mamdouh Farouk
Department of Computer Science,Faculty of Computers and Information, Assiut University, Assiut, Egypt

Samhaa R. El-Beltagy
Department of Computer Science,Faculty of Computers and Information, Cairo University, 5 Tharwat Street, Giza, Egypt

Mahmoud Rafea
Central Lab for Agricultural Expert Systems, Agricultural Research Center, Ministry of Agriculture and Land Reclamation, El-Nour St., Dokki, Giza, Egypt

Keywords: Semantic web, Annotation, DB annotation, XML

Abstract: The annotation of web pages is a critical task for the success of the semantic web. While many tools exist to facilitate the annotation of static web pages, annotation of dynamically generated ones has not been sufficiently addressed. This paper addresses the task of annotating web pages whose dynamic content is derived from a database. The approach adopted is based on annotating a database schema based on public ontologies and using this database annotation to generate a dynamic web page's content annotation on the fly. This paper both presents details about the adopted approach as well as a tool that supports this approach.

1 INTRODUCTION

The semantic web presents a vision in which software agents will be able to understand and use web content to perform tasks that aid Internet users. This will enable web pages as well as databases and other web resources to be 'machine-available' (James, 2002). In order to realize this vision of the semantic web, web resources have to be semantically annotated in a format that enables web agents to understand web contents where semantic annotation refers to the process of providing additional information (metadata) to existing data so as to describe their content and context. As a result, metadata that describe web resources have been identified as key to creating the Semantic Web (Handschuh, 2001). After obtaining metadata, it should represented in a format which facilitates reasoning services from operation over the metadata, thus enhancing their processing power (Kopena, 2003). Using ontologies in the annotation process makes the final annotation more usable.

Annotating web resources is no trivial task due to the current size of these resources. Supporting this task through the development of tools that can facilitate it is imperative for its success. As a result much work has been carried out to make this process an easier one. The aim of this work is to address this issue in relation to pages whose content is dynamically derived from databases.

2 BACKGROUND

During the last few years, many tools [manual and semi-automatic] were developed to facilitate the semantic annotation process. Manual or semi-automatic tools try to extract the 'meaning' of a web page and represent this meaning (metadata) in a format, which can enable search agents to perform reasoning on it. An example of such tools is the MnM tool, which provides both automated and semi-automated support for annotating web pages with semantic contents (Vargas, 2002). However, most of the developed semantic annotation tools are devoted to the task of annotating static web pages (HTML pages) even though a large majority of current web sites are made up of largely dynamic web pages that are generated based on scripts that

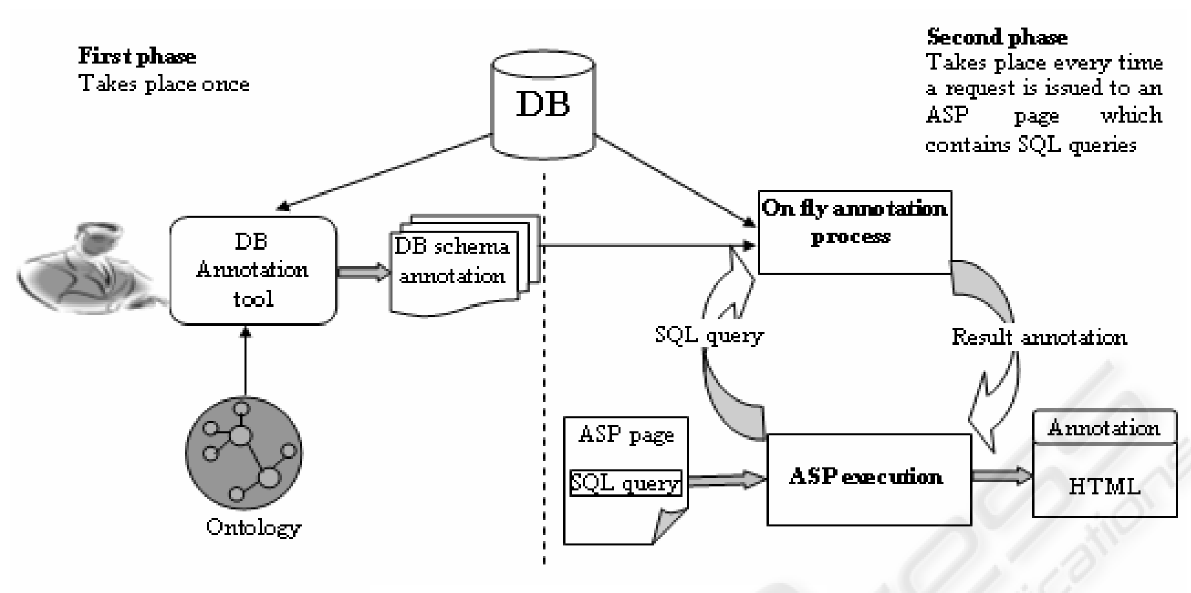


Figure 1: General architecture

access databases and/or respond to specific user variables (Siegfried, 2003). Contents of such pages change according to its user. Failing to annotate these web pages, results in a failure to integrate them as part the semantic web vision. In the following sections, we describe a system/tool that was developed specifically to address this issue and that is capable of supporting the task of creating annotations for web pages, which derive their content from a database.

3 SYSTEM OVERVIEW

In this work, we propose an approach to annotate dynamic web pages that are generated based on data retrieved from a database. Our approach is designed and implemented for Active Server Pages (ASP), but can generalize to any dynamically generated web pages that access a database. The generated annotations are represented in DAML. The technique proposed to annotate dynamic websites depends on dynamic generation of annotations according to SQL queries contained in pages within such sites. . Figure 1 shows the general architecture of our proposed dynamic web page annotation tool.

As shown in figure 1, two phases are proposed for the annotation process. The **first phase** involves annotating the database schema of the DB used by some given application. In this phase, a user manually maps database objects to some predefined ontology classes. This task is facilitated by the developed tool. The output of this phase is an XML document that describes the DB schema. In the **second phase,** on the fly automatic annotations are generated and augmented to a query's result. The input to this process is the SQL query contained in a web page as well the XML file generated in the first phase and the output is the generated annotation. In the next few subsections more details are provided for each of these phases.

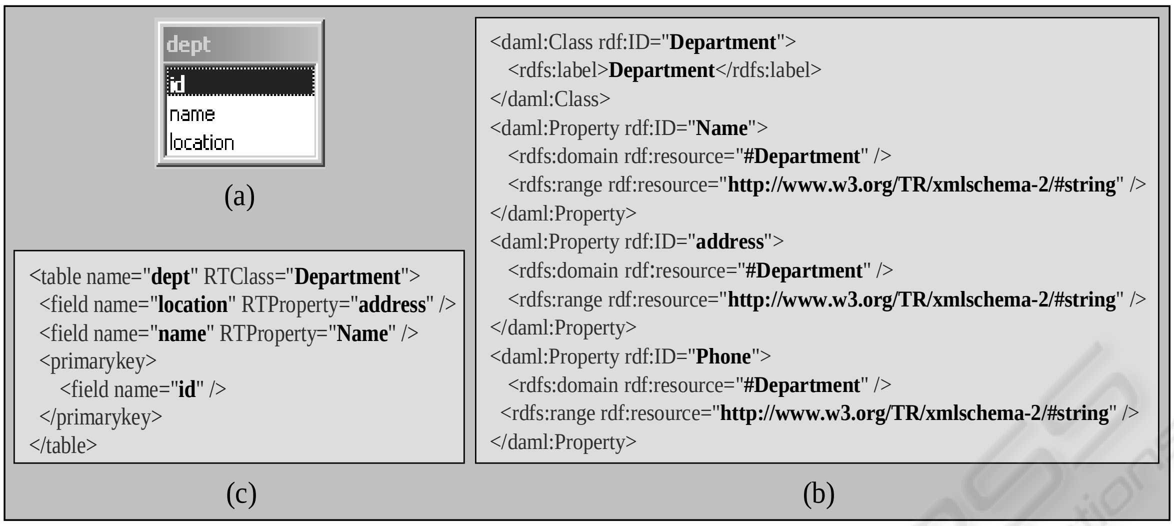
3.1 Annotating the database schema

In our work, automating data annotation relies largely on successfully annotating the schema of the database from which data is to be retrieved. In this process, a user describes various DB objects and relations that exist between these objects based on a predefined ontology. This description is represented in a generic formal structure, (we've chosen XML) so as to be easy to use.

Figure 1 shows the model for database schema annotation process. A database annotator tool (DBA) was implemented to facilitate this process. The DBA accesses the database and loads its schema in a graphical user interface. The DBA also allows a user to select ontology, loads the ontology and displays its structure. Through the graphical user interface, the user can annotate the DB schema using the ontology classes.

The DB schema annotation process is divided into the following steps:

- Annotation of database tables: in this step, a user describes database tables by mapping them to ontology classes and mapping the table's fields to the class' properties.
- Annotation of relations between tables: in this step, a user describes the type of relationship that exists between the database tables.


dept
id
name
location
(a)
(c)
(b)

```
<table name="dept" RTClass="Department">
  <field name="location" RTProperty="address" />
  <field name="name" RTProperty="Name" />
  <primarykey>
     <field name="id" />
  </primarykey>
</table>
```

```
<daml:Class rdf:ID="Department">
   <rdfs:label>Department</rdfs:label>
</daml:Class>
<daml:Property rdf:ID="Name">
   <rdfs:domain rdf:resource="#Department" />
   <rdfs:range rdf:resource="http://www.w3.org/TR/xmlschema-2/#string" />
</daml:Property>
<daml:Property rdf:ID="address">
   <rdfs:domain rdf:resource="#Department" />
   <rdfs:range rdf:resource="http://www.w3.org/TR/xmlschema-2/#string" />
</daml:Property>
<daml:Property rdf:ID="Phone">
   <rdfs:domain rdf:resource="#Department" />
  <rdfs:range rdf:resource="http://www.w3.org/TR/xmlschema-2/#string" />
</daml:Property>
```

Figure 2: table annotation example

3.1.1 Annotation of database tables

To map a DB table to an ontology class, the DBA tool creates a ***table*** element and maps fields in this table to properties of a class selected from the used ontology. The mapping is represented using another nested element called ***field***. This is done for each field in the database table. Sometimes fields in the DB table need not be assigned to any property in the ontology, e.g., a record's auto-increment field.

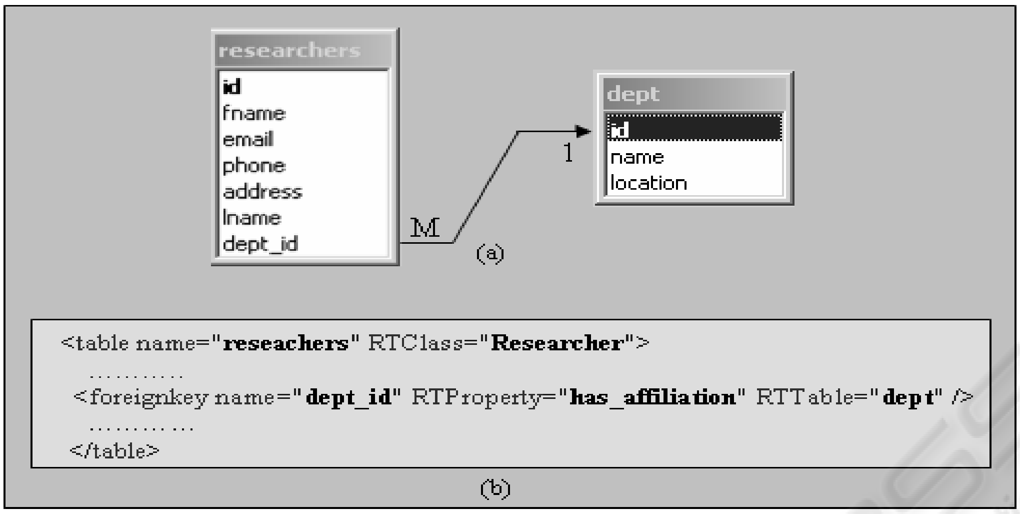
To further illustrate and clarify this process, assume that we have a "dept" table in our database as defined in figure 2 (a). Also, assume that the selected ontology has a "Department" class defined as in Figure 2 (b). Mapping between the table and the ontology will result in the annotation of the "dept" as shown by the XML representation provided in Figure 2 (c). In this figure, the "dept" table is represented with the ***table*** element. The attribute "name" provides the name of the table in the original database, while the attribute "RTClass" provides the name of the ontology class to which this table can be mapped (in this case: Department). In a similar manner, the table's field "location" is mapped to the class property "address" and the table's field "name" is mapped to the class property "name". Finally, a ***primarykey*** element defines the primary key field(s) of a DB table, which is used in the query result annotation process (explained later) to distinguish between different DAML created instances.

3.1.2 Annotation of relations between tables

Annotating the relationships that exist between DB objects is critical to making rich annotations to data that will be retrieved from the database. Of particular interest, is the annotation of one-to-many and many-to-many types of relationships.

Annotating one-to-many relationships: When two tables are related together through a one-to-many type of relationship, one of these tables will contain the primary key of the other table as a foreign key. To annotate this relationship, the tool creates a ***foreignkey*** element to represent the relationship between these tables. Within the ***foreignkey*** element, the name attribute is used to represent the foreign key field within the table element representing the table which contains the foreign key field. The XML representation of the foreign key takes the following structure:
<foreignkey name="ForeignKey" RTProperty=" OntologyClassProperty" RTTable="ReferenceTable"/>

To illustrate further, figure 2 (a) shows a one-to-many relationship between the researcher table and the department table (each researcher can work in only one department and a department will typically have many researchers working within it). A researcher is affiliated to a dept, so within our selected ontology this relationship is called "has_affiliation". To annotate the relationship between these two tables (researcher and dept), a <foreignkey> tag is added under the <table> tag of the researcher table. Within this tag, the primary key of the department in which the researcher works is represented by the name attribute, the name of the relationship is represented by the RTProperty (Refer to Property) attribute (which is a property previously defined in the used ontology) and finally the name of the table to which the researcher is related is represented using the RTTable (Refer To Table) attribute. Figure 3 (b) shows this annotation.



```
<table name="reseachers" RTClass="Researcher">
  ..........
 <foreignkey name="dept_id" RTProperty="has_affiliation" RTTable="dept" />
  ..........
</table>
```

Figure 3: one-to-many relation annotation

Annotating many-to-many relationships: When two tables are related together by a many-to-many type of relationship, a bridge table is created to link the two tables. Within the bridge table, the relationship is further broken into two relations each of which is a one-to-many relation. So, to annotate a many-to-many relationship, the following steps are carried out:

1- Create a <bridgetable> tag to relate the two tables. Relating between the two tables is done through embedding two ***foreignkey*** elements in the bridge table. One of these is used to describe the relationship between the first table to the second, while the second is used to describe the relationship between the second to first.

An example of a many-to-many relationship is that which exists between researchers and publications. A researcher can have many publications, and a publication can be authored by multiple researchers. So, the bridge table created to represent this relationship will have two ***foreignkey*** elements, one that states that a publication has an author which is a researcher, and another that states that a researcher has publications. This can be represented as follows:

*<bridge_table name="**researcher_pub**">*
*<foreignkey name="**researcher_id**" RTProperty="**has_author**" RTTable="**researcher**" />*
*<foreignkey name="**publication_id**" RTProperty="**has_publication**" RTTable="**publications**" />*
</bridge_table>

2- For each table involved in a many-to-many relation (researcher and publications in the given example): add a <has_relation> tag and set the “with” attribute value to the name of the bridge table as follows:

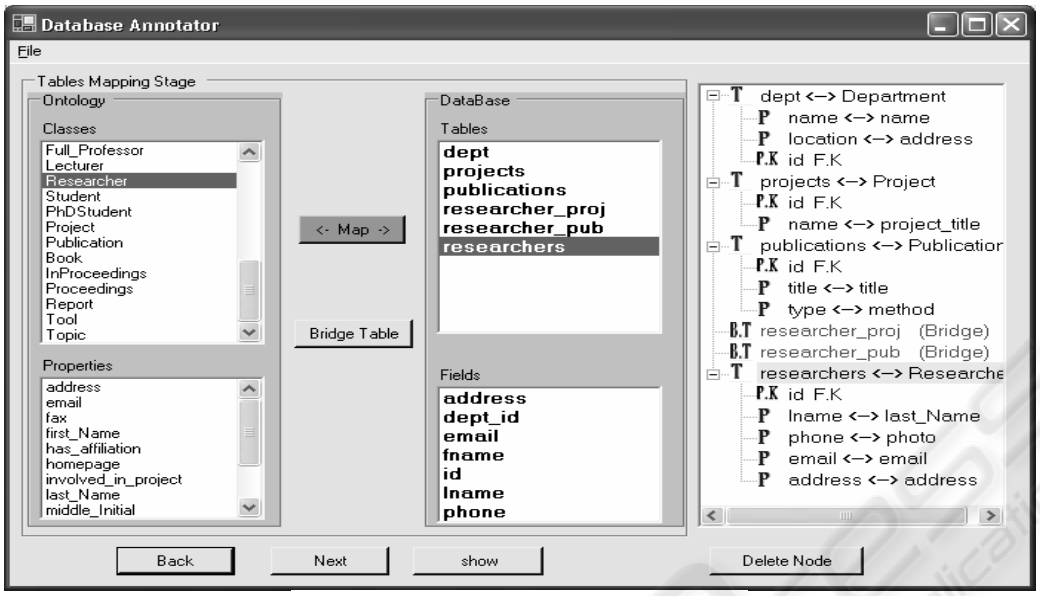
<has_relation with="researcher_publ"/>

A tool has been created to facilitate this annotation process is illustrated in figure 4.

3.2 On the fly automatic annotation of a query’s result

When a request for a dynamic web page is received on the server, the server executes the requested page’s script and returns the HTML result to the client. Our goal is to annotate the result so as to enable semantic manipulation of the page by sending back the annotation embedded in the HTML result. To achieve this goal, when the server executes the requested page, the annotator generates an annotation for the result simultaneously according to the new page’s contents which are determined by the SQL query(s) executed in that page. This process employs the generated DB schema annotation described in section 3.1. The second phase, shown in figure 1, illustrates the process model of “on the fly annotation”. The execution of a web page results in a call to a function that takes the SQL query as input and returns the annotation of the query result as output. The process of annotation generation occurs on the fly as part of the dynamic page’s execution. The following three steps are applied to generate the annotation for a result of an SQL query encountered within a dynamic web page:

- Step 1: Primary keys values of all records contained in the result are fetched



Classes: Full_Professor, Lecturer, Researcher, Student, PhDStudent, Project, Publication, Book, InProceedings, Proceedings, Report, Tool, Topic

Properties: address, email, fax, first_Name, has_affiliation, homepage, involved_in_project, last_Name, middle_Initial

Tables: dept, projects, publications, researcher_proj, researcher_pub, researchers

Fields: address, dept_id, email, fname, id, lname, phone

```
T dept <--> Department
    P name <--> name
    P location <--> address
    P.K id F.K
T projects <--> Project
    P.K id F.K
    P name <--> project_title
T publications <--> Publication
    P.K id F.K
    P title <--> title
    P type <--> method
B.T researcher_proj (Bridge)
B.T researcher_pub (Bridge)
T researchers <--> Researcher
    P.K id F.K
    P lname <--> last_Name
    P phone <--> photo
    P email <--> email
    P address <--> address
```

Figure 4: Database annotation tool

- Step 2: An ontology instance for each retrieved record is created
- Step 3: Relationships between the created instances are identified.

Each of the above steps is detailed in the following subsections

3.2.1 Step 1: Fetching primary key values of all records contained in a query result

The first step to annotate the result of an SQL query is to obtain the primary key values for all records, returned in the query's result. This can be done easily by generating an SQL query from the original incoming query to retrieve the desired primary key values. The generation of this new query depends on the annotation of the DB schema, which is queried to get the primary key fields for each table. So, to generate this query we do the following:

(1) Get the names of the queried tables from the original query (this can be obtained from the *from* clause).

(2) For each table get the primary key field(s) from the XML DB schema annotation.

(3) Construct a *select* query to retrieve primary key values using the primary key field values obtained from the previous step and with the "*from*" and "*where*" clauses corresponding to their counterparts in the original query. The resulting query is in the following form :

    Select (tableName.PK_Field)$^+$ (From clause) (Where Clause);

    The (tableName.PK_Field) is constructed using steps 1 and 2, and the (From clause) and (Where Clause) are obtained from the original query.

For example, given the following **original query:**
*select * from dept where name='CS';*
After applying the above steps, the **generated query will be:**
select dept.id from dept where name='CS';

The result of executing the generated query will be a list of primary key values, which are used in the next step.

3.2.2 Step 2: Creating an ontology instance for each of the retrieved records

In this step, an ontology instance is created for each record retrieved as part of the query result. To create an ontology instance for a specific record, the ontology class corresponding to the table from which the record is retrieved, is obtained from the XML DB schema file. Then, an empty instance of that class is created and its property values instantiated with values from the retrieved record fields according to the mapping between fields and properties in the DB schema annotation. For example, if the retrieved records were obtained from the "researcher" table and have primary keys 3 and 7, in this step, two instances from the ontology class corresponding to the researcher table ("Researcher" class), are created for each record. To do this, a new query is constructed to retrieve all fields' values in these two records from the "researcher" table. The properties' values of the created instances are filled with these retrieved field's values according to the mapping between fields and properties in the DB schema annotation.

3.2.3 Representing relationships between the created ontology instances

In this step we relate the instances created in the previous step based on the relationship that exists between these instances and other objects in the database. In the example given, assuming a relationship exists between researchers and projects where a research can work on multiple projects, we need to represent a relationship between the retrieved researchers and the specific projects they are working on. To do so, the tool performs a search for all relations that can exist between a researcher and other database objects by checking relations that exist between the researcher table and other DB tables in the XML DB annotated schema. For the researcher table in the XML file (represented by a table element), the tool will find that a tag <has_relation_with="research_project"/> exists; this means that there exists a many-to-many relationship between the researcher table described in the "research_project" bridge table. So, if in the previous step, an instance was created for a researcher whose record has a primary key called id with the value of 7, in this step, the tool will create a query on the bridge table to get all projects, that have a researcher with id= 7 (this represents projects in which this researcher is involved in). The tool then creates instances for each project record retrieved from the query. The tool also refines the attributes in each instance. The "persons_involved" attribute in the project instances are set to the URI of the instance of the researcher with id = 7 and the "involved_in_project" attribute in the researcher instance, is set to the created instances.

The final result of the annotation process is the set of related instances generated from the above steps. These instances that contain semantic information about the page content will be added to header of the HTML that is sent to the client.

4 CONCLUSION

The success of the semantic web depends on the easy creation of ontology-based metadata through the use of semantic annotation tools. This work presented an approach whereby annotation of dynamic pages, which derive their content from databases, can occur on the fly. The proposed technique generates annotations according to the retrieved data by annotating a database schema, creating a direct link between the structure of the database (tables and fields of a relational database) and concepts/properties in ontology, and finally using this in the annotation process. The proposed technique is powerful, simple, and easy to implement. Implementing this approach enables semantic manipulation of such web pages thus contributing to the establishment of the semantic web vision.

ACKNOWLEDGMENTS

This work was sponsored by the European Community's FP6 Information Society Technologies programme under contract IST-001935, EVERGROW (www.evergrow.org).

REFERENCES

James, H., Berners-Lee, T., Eric, M., 2002. Integrating Applications on the Semantic Web. In Journal of the Institute of Electrical Engineers of Japan, Vol 122(10), October, 2002, p. 676-680.

Handschuh, S., Staab, S., 2002, Authoring and annotation of Web pages in CREAM, in: Proceedings of the 11th International World Wide Web Conference (WWW), Honolulu, Hawaii, ACM Press, 7–11 May 2002, pp. 462–473.

Vargas-Vera, M., et al., 2002, MnM: Ontology Driven Semiautomatic and Automatic Support for Semantic Markup, In European Knowledge Acquisition Workshop 2002, Springer-Verlag, 2002, pp. 379–391.

Kopena, H., and Willim, C. R., 2003, "DAMLHiessKB: A Tool for Reasoning with the Semantic Web", IEEE Intelligent Systems, MAY/JUNE 2003, pp. 74-77.

Siegfried, H., Raphael, V., Staab, S., 2003, Annotation for the Deep Web, IEEE Intelligent Systems, SEPTEMBER/OCTOBER 2003, pp. 42-48.