PUC

ISSN 0103-9741

Monografia em Ciência da Computação
n° 10/10

W-Ray: A Strategy to Publish Deep Web Geographic Data

Helena Piccinini, Melissa Lemos,

Marco A. Casanova, Antonio L. Furtado

Departamento de Informática

PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO DE JANEIRO

RUA MARQUÊS DE SÃO VICENTE, 225 - CEP 22451-900

RIO DE JANEIRO

Monografias em Ciência da Computação, No. 10/10 ISSN: 0103-9741
Editor: Prof. Carlos José Pereira de Lucena Jun, 2010

W-Ray: A Strategy to Publish Deep Web Geographic Data

Helena Piccinini[1, 2], Melissa Lemos[1], Marco A. Casanova[1], Antonio L. Furtado[1]

[1]Department of Informatics – PUC-Rio – Rio de Janeiro, RJ – Brazil
{hpiccinini, melissa, casanova, furtado}@inf.puc-rio.br
[2]Diretoria de Informática – IBGE – Rio de Janeiro, RJ – Brazil
helena.piccinini@ibge.gov.br

Abstract. This work introduces an approach to address the problem of accessing geographic data from the Deep Web. The approach relies on describing the relevant data through well-structured sentences, and on publishing the sentences as Web pages, following the W3C and the Google recommendations. For conventional data, the sentences are generated with the help of database views. For vector data, the topological relationships between the objects represented are first generated, and then sentences are synthesized to describe the objects and their topological relationships. Lastly, for raster data, the geographic objects overlapping the bounding box of the data are first identified with the help of a gazetteer, and then sentences describing such objects are synthesized. The Web pages thus generated are easily indexed by traditional search engines, but they also facilitate the task of more sophisticated engines that support semantic search based on natural language features.

Keywords: Deep Web, Geographic Data, Natural Language Processing.

Resumo. Este trabalho apresenta uma abordagem para o problema de acesso a dados convencionais e geográficos da Deep Web. A abordagem é baseada nas descrições dos dados relevantes através de sentenças bem estruturadas e na publicação destas sentenças em páginas Web. Seguindo recomendações do W3C e da Google. Para os dados convencionais, sentenças são geradas com a ajuda de visões do Banco de Dados. Para os dados vetoriais, primeiramente são geradas relacionamentos topológicos entre os objetos representados e então as sentenças são sintetizadas para descrever os objetos e seus relacionamentos topológicos. Por último, para os dados raster, os objetos geográficos contidos dentro de uma caixa delimitadora são identificados com a ajuda de um gazetteer e as sentenças que descrevem tais objetos são geradas. As páginas Web geradas são facilmente indexadas por mecanismos de busca tradicionais e também facilitam as tarefas de mecanismos de busca mais sofisticados, ou seja, os que suportam pesquisas semânticas baseadas em linguagem natural.

Palavras-chave: Web profunda, dados geográficos, processamento de linguagem natural.

Summary

1 Introduction 4

2 The W-Ray approach for conventional geographic data 5

2.1 Motivation and overview of the approach 5

2.2 Guidelines for view design 5

2.3 Translating materialized view data to natural language sentences 7

2.3.1 Overview 7

2.3.2 Choice of the external vocabulary 8

2.3.3 Free template definition 10

2.3.4 Default template definition 11

2.3.5 Modifiable default template definition 11

2.4 Guidelines for publishing the sentences as static Web pages 12

3 W-Ray for geographical data in vector format 13

4 W-Ray for raster data 17

5 Conclusions 18

6 References 19

Annex 21

1 Introduction

Unlike the Surface Web of static pages, the Deep Web [2] comprises data stored in databases, dynamic pages, scripted pages and multimedia data, among other types of objects. Estimates suggest that the size of the Deep Web greatly exceeds that of the Surface Web – with nearly 92,000 terabytes of data on the Deep Web versus only 167 terabytes on the Surface Web, as early as 2003. A July 2000 survey [4] estimated 96,000 search sites and 550 billion Web pages in the Deep Web, whereas an April 2004 study [7] estimated 330,000 Deep Web sources with over 1.2 million query forms, reflecting a 3-7 times increase in 4 years. However, Deep Web databases are typically under-represented in search engines due to the technical challenges of locating, accessing, and indexing the databases. Indeed, since Deep Web data is not available as static Web pages, traditional search engines cannot discover data stored in the databases through the traversal of hyperlinks, but rather they have to interact with (potentially) complex query interfaces.

Two basic approaches to access Deep Web data have been proposed. The first approach, called surfacing or Deep Web Crawl [20], tries to automatically fill HTML forms to query the databases. Queries are executed offline and the results are translated to static Web pages, which are then indexed [19, 20]. The second approach, called federated search or virtual integration [6, 22], suggests using domain-specific mediators to facilitate access to the databases. Hybrid strategies, which extend the previous approaches, have also been proposed [26].

Despite recent progress, accessing Deep Web data is still a challenge, for two basic reasons [25]. First, there is the question of scalability. Since the Deep Web is orders of magnitude larger than the Surface Web, it may not be feasible to completely index the Deep Web. Second, databases typically offer interfaces designed for human users, which complicates the development of software agents to interact with them.

This paper proposes a different approach, which we call W-Ray by analogy with medical X-Ray technology, to publish geographic data, in conventional, vector or raster formats, stored in the Deep Web. The basic idea consists of creating a set of natural language sentences, with a simple structure, to describe Deep Web data, and publishing the sentences as static Web pages, which are then indexed as usual. The use of natural language sentences is convenient for three reasons. First, they lead to Web pages that are acceptable to Web crawlers that consider words randomly distributed in a page as an attempt to manipulate page rank. Second, they facilitate the task of more sophisticated engines that support semantic search based on natural language features [8, 29]. Lastly, the descriptions thus generated are minimally acceptable to human users. The Web pages are generated following the W3C guidelines [5] and the recommendations published by Google to optimize Web site indexing [14].

This paper is organized as follows. Section 2 describes how to publish conventional geographic data. Section 3 discusses how to describe geographic data in vector format. Section 4 extends the discussion to geographic data in raster format. Section 5 lists the conclusions. The annex contains the Prolog code that implements a tool developed to support the design and publication processes.

2 The W-Ray approach for conventional geographic data

2.1 Motivation and overview of the approach

The W-Ray approach to publishing conventional geographic data as Web pages proceeds in two stages. In the first stage, the designer manually defines a set of database views that capture which data should be published, and specifies templates that indicate how sentences should be generated. The second stage is automatic and consists of materializing the views, translating the materialized data to natural language sentences, with the help of the templates, and finally publishing the sentences as static Web pages.

Note that metadata, typically associated with geographic data, can be likewise processed.

As an alternative to synthesizing natural language sentences, one might simply format the materialized view data as HTML tables. However, this is not a reasonable strategy for at least two reasons. First, some search mechanisms consider tables as visual objects. Second, tables may be difficult to read, even for the typical user, or at all inadequate for the visually impaired users.

Indeed, the third principle of the W3C recommendation [5] indicates that *"Information and the operation of user interface must be understandable."*, and item 4 of the Google Web page optimization guidelines [14] recommends that *"(Web page) content should be: easy-to-read; organized around the topic; use relevant language; be fresh and unique; be primarily created for users, not search engines"*. This recommendation reflects the fact that Web crawlers may interpret words randomly or repeatedly distributed in a Web page as an attempt to manipulate page rank, and accordingly reject indexing the page.

Finally, we observe that some of the W3C specific recommendations for the visually impaired user in fact coincide with Google's orientations. Comparing the two, it is clear that the difficulties faced by the visually impaired user are akin to those a search engine suffers during the data collection step. As an example, both Google and W3C recommend using the attribute "alt" to describe the content of an image. Naturally, the content of an image is opaque to both visually impaired users and search engines, but an alternate text describing the image can be indexed by a search engine and read (by a screen reader) to the visually impaired user. In general, many W-Ray strategies defined to address the limitations of search engines also apply to the design of a database interface for the visually impaired user.

2.2 Guidelines for view design

As mentioned in Section 2.1, the designer must first select which data will be published with the help of database views.

W-Ray then offers the following guidelines that the designer should follow when defining the views:

G1. Attributes whose values have no semantics outside the database should not be directly published.

G2. Artificially generated primary keys, foreign keys that refer to such primary keys, attributes with domains that encode classifications or similar artifacts, if selected for publication, should have their internal values replaced by their respective external definitions. For example, a classification code should be replaced by the corresponding classification term.

G3. Attributes that contain private data should not be published.

G4. Views should not contain too many attributes; only those attributes that are relevant to help locate the objects and express their relationships should be selected.

Example 1: This example is based on the SIDRA database, a statistical aggregate database which the Brazilian Institute of Geography and Statistics (IBGE) publishes on the Web with the help of HTML forms. We observe that SIDRA is not indexed by any conventional search engine.

We first manually defined a set of database views that captured which data should be published, following the view design guidelines listed in this section.

For example, consider the following question: "What is the population of each territorial unit in Brazil?", a quite frequent query. To answer such queries, we introduced the "*territorial_unit*" view over the SIDRA database, schematically defined in Table 1, where

- the first column identifies the database tables used in the view definition.
- the second column indicates an alias to refer to the tables in the *where clause* of the view definition.
- the third column indicates the attributes used (and their descriptions) in the *where clause* or in the *target clause* of the view definition.
- the fourth column indicates which attributes appear in the *target clause* of the view definition.
- the fifth column associates a variable with each *target clause* attribute, used to define templates in Section 2.3.
- the sixth column indicates the *where clause* of the view definition.

By inspecting Table 1, one may observe that the definition of the "*territorial_unit*" view meets the guidelines G2, G3 and G4. As for G1, it suffices to note that the values of the attributes from the "*v_aggregate_variable*" table that appear in the target list of the view are meaningful names (of territorial units, etc.), and those from the "*v_aggregate_value*" table included in the target list are dates or values of aggregate variables. □

Table 1 – Schematic definition of the *"territorial_unit"* view over the SIDRA database

Table	**Table Alias**	**Attribute Used** ***(Description)***	**Target Clause**	**Var.**	**Where Clause**
v_aggregate_variable	V	name_territorial_unit *(name of the territorial unit)*	yes	***U***	V.code_aggregate_occurrence = T.code_aggregate_occurrence and V. code_survey = T.code_survey and V.date_year = T.date_year and V. date_month = T.date_month and V.no_order_period = T.no_order_period and V. code_aggregate_variable = T. code_aggregate_variable
		name_territorial_level *(level of the territorial unit, such as country, state,…)*	yes	***L***	
		code_aggregate_occurrence			
		name_occurrence_survey *(survey that generated the aggregation variable)*	yes	***S***	
		name_aggregate_variable *(name of an aggregation variable, such as resident population)*	yes	***A***	
		name_measurement_unit *(unit of measure of the aggregation variable)*	yes	***M***	
		code_aggregate_variable			
		code_survey			
		date_year			
		date_month			
		no_order_period			
t_ aggregate_value	T	code_aggregate_variable			
		code_survey			
		date_year (year the aggregation variable was measured)	yes	Y	
		date_month			
		no_order_period			
		code_aggregate_occurrence			
		val_aggregate_occurrence (value of the aggregation variable)	yes	V	

2.3 Translating materialized view data to natural language sentences

2.3.1 Overview

The heart of the W-Ray approach lies in the translation of materialized view data to natural language sentences. Fuchs et al. [12] propose a single language for machine and human users, basically by translating English sentences to first-order logic. Others propose to translate RDF triples to natural language sentences [11, 17], simply by concatenating the triples. Tools to translate conventional data to RDF triples have also been developed [3, 9], which typically map database entities to classes, attributes to datatype properties, and relationships to object properties. The proposals introduced in [11, 17] do not consider sequences of RDF triples, though, which we require to compose simple sentences into more complex syntactical constructions. Therefore, we combine the strategies to synthesize sentences described in [17] with the mapping of conventional data to RDF triples introduced in [3].

The translation of materialized view data to natural language sentences involves two tasks: choice of an appropriate external vocabulary; and definition of templates to guide the synthesis of the sentences.

2.3.2 Choice of the external vocabulary

First observe that view and attribute names are typically inappropriate to be externalized to the database users. The same observation also applies, to some extent, to the values of attribute domains defined by enumeration, such as the names of the days of the week (in some natural language). This problem is further illustrated in Example 6. Naturally, attribute domains defined over commonly used data types, such as dates, do not require further transformations (except perhaps minor transformations to conform with the established conventions to denote numbers and dates), but they typically must be followed by the unit of measure adopted. This implies that the designer must define an external vocabulary, that is, a set of terms that will be used to describe materialized view data to the users.

W-Ray then offers the following guidelines that the designer should follow when describing materialized view data to the users:

G5. View and attribute names should be mapped to one or more terms of controlled vocabularies covering the application domain in question, or of general purpose vocabularies, such as an upper-level ontology or Wordnet.

G6. Attribute domains defined by enumeration should be likewise treated.

G7. Attribute domains defined over commonly used data types should be mapped to a XML Schema data type.

If followed, these guidelines permit defining hyperlinks from the terms of the view data and metadata (such as attribute names) to the terms of the external vocabularies. A similar strategy to synthesize sentences is discussed in [15]. An extension to Wordnet is also proposed in [27] to treat concepts corresponding to compound nouns.

Example 2: Continuing with Example 1, we chose the following external vocabularies to meet guideline G5 (see Table 2):

- OECD (Organization for Economic Cooperation and Development) Glossary of Statistical Terms [23].
- EUROSTAT (statistical organization of the European Commission) Glossary of Statistical Terms [10].

- SUMO Ontology (Suggested Upper Merged Ontology) [28].

As for guidelines G6 and G7, we treated attribute domains as follows (see Table 3):

- The domains of the "*date_year*" and "*val_aggregate_occurrence*" attributes were mapped to XML Schema data types.
- The domain values of the other attributes were mapped to terms of the vocabularies indicated in Table 3. For example, the names of the Brazilian territorial units (the domain of the "*name_territorial_unit*" attribute) come from IBGE's Brazilian gazetteer, which should be aligned with Geonames, a widely known gazetteer available on the Web [13]. By contrast, the names of IBGE surveys (the domain of the "*name_occurrence_survey*" attribute) should be understood as part of IBGE's internal vocabulary. □

Table 2 – External vocabularies for the attribute names of the "*territorial_unit*" view

Attribute Name	**External Term**	**Vocabulary**
name_territorial_unit	territorial unit	EUROSTAT Glossary of Stat. Terms
name_territorial_level	level	EUROSTAT Glossary of Stat. Terms
name_occurrence_survey	survey	OECD Glossary of Statistical Terms
name_aggregate_variable	variable	OECD Glossary of Statistical Terms
name_measurement_unit	unit of measure	EUROSTAT Glossary of Stat. Terms
date_year	year	SUMO Ontology
val_aggregate_occurrence	aggregated value	SUMO Ontology

Table 3 – External vocabularies for the attribute domains of the "*territorial_unit*" view

Attribute Name ***(Description)***	**Attribute Domain**	**Vocabulary**
name_territorial_unit	Names of the Brazilian territorial units, such as "Amazonas" and "Pará"	IBGE internal vocabulary, aligned with Geonames
name_territorial_level	territorial unit levels, such as "unit of the federation"	IBGE internal vocabulary, aligned with Geonames
name_occurrence_survey	Names of the IBGE surveys , such as "Censo Demográfico"	IBGE internal vocabulary
name_aggregate_variable	Names of aggregate variables, such as "resident population"	IBGE internal vocabulary
name_measurement_unit	Units of measure, such as "people"	SUMO Ontology
date_year	(year values in the standard four-digit representation)	XML Schema data type
val_aggregate_occurrence	(numeric values in general)	XML Schema data type

2.3.3 Free template definition

We offer three alternatives to define templates: free template definition; default template definition; and modifiable default template definition. The first alternative, discussed in this section, leaves template definition in the hands of the designer and, thus, may lead to sentences with arbitrary structure. However, the designer should create templates that define simple affirmative sentences.

Very briefly, an *affirmative sentence* contains a *subject* and a *predicate* [24]. The subject of a sentence is the entity about which something is asserted and the predicate is what is declared about the subject. The predicate may be divided into a *verb*, with optionally a *direct object*, or an *indirect object*, or both, and *predicatives*. A predicative may modify the subject, a direct object or an indirect object. The subject of the sentence is typically the value of an identifying attribute of the view; predicatives of the subject are constructed from the values of the other attributes and from the external vocabulary; and the verb is also taken from the external vocabulary.

W-Ray then offers the following guidelines that the designer should follow when designing free templates:

G8. A template should use the external vocabularies selected during view definition, common syntactical elements (articles, conjunctions, etc.) [24] and punctuation marks.

G9. A template should generate a sentence that characterizes an entity through its properties and relationships.

G10. The subject of the sentence should be associated with an identifying attribute of the view.

G11. The predicate of the sentence should be associated with other view attributes that further describe the entity, or that relate the entity to other entities.

Example 3: Returning to our running example, we defined the following template to publish "*territorial_unit*" view data, using the variables in the fifth column of Table 1:

U *is a territorial unit with level "**L**" that has a total of **V M** for the year **Y** and variable "**A**".*

Next, we materialized the view and transformed each line of the resulting table into a sentence, using the template. The following sentence illustrates the result:

<u>Roraima</u> *is a* territorial unit *with* level "**unit of the federation**" *that has a total of* **395.725 people** *for the year* **2007** *and* variable "**resident population**".

Note that: the underlined word is the subject of the sentence; the predicate '*is a* territorial unit *with* level "**unit of the federation"**' qualifies the subject; the words in boldface are view data that play the role of predicatives of the subject, together with the fragments in italics and the external terms in regular font. □

2.3.4 Default template definition

Since template definition sometimes is a tedious task that may lead to ill-formed sentences, we implemented a tool, written in Prolog, to help the designer. The tool generates what we called default templates starting from an entity-relationship model that is a high-level description of the views. The designer's task is therefore limited to creating the entity-relationship model, rather than directly creating templates. The tool also materializes the views and synthesizes the corresponding sentences, which will then have a regular syntactical structure.

Example 4: We repeat Example 3 using the default templates alternative. We started by creating an ER model of the "*territorial_unit*" view, which is quite simple in this case:

```
entity(territorial_unit,name_territorial_unit).
attribute(territorial_unit,name_territorial_level).
attribute(territorial_unit,name_occurrence_survey).
attribute(territorial_unit,name_aggregate_variable).
attribute(territorial_unit,name_measurement_unit).
attribute(territorial_unit,date_year).
attribute(territorial_unit,val_aggregate_occurrence).
```

We then used the design tool to generate the default templates from the entity-relationship model, using the translations of attribute names to terms of the external vocabularies indicated on Table 2 and the variables associated with attribute names from Table 1. Examples of default templates are:

`'There is a territorial unit with name` ***`P`***`'`

`'The level of` ***`P`*** `is` ***`L`***`'`

Using the default templates, the tool then synthesized sentences such as (data in boldface):

`'There is a territorial unit with name` **`Roraima`**`'.`

`'The level of` **`Roraima`** `is` **`unit of the federation`**`'.`

2.3.5 Modifiable default template definition

Default templates, albeit regular, may contain a high level of redundancy. To circumvent this problem, the tool allows the designer to redefine the default templates and to combine several templates into a single template.

Example 5: Using the design tool, we redefined the two templates of Example 4 as follows (where the variables in boldface italics in the new template must occur in the default template):

Default template: `'There is a territorial unit with name` ***`P`***`'`
New template: `'`***`P`***`'`

Default template: `'The level of` ***`P`*** `is` ***`L`***`'`
New template: `'is a` ***`L`***`'`

After redefining the templates, we combined them into a single template as follows (in Prolog notation):

```
facts((territorial_unit(P),level(P,L)).
```

The final template will be quite simple:

```
'P is a L'
```

Using the final template, the tool then synthesized sentences such as (data in boldface):

```
'Roraima is a unit of the federation'
```

2.4 Guidelines for publishing the sentences as static Web pages

In the context of the W-Ray approach, the most relevant recommendations for Web page construction listed by W3C [5] and Google [14] are:

- Create a main Web page with a hierarchy of hyperlinks (Google Recommendation 3).
- Publish an accurate description of the Web page content using the tag <META> (conformance requirements of WCAG 2.0 and Google Recommendation 2).
- Add a Web page title using the tag <Title> (W3C Recommendation 2.4.2 and Google Recommendation 1).
- Structure Web page content using header tags <h*n*> (W3C Recommendation 2.4.6 and Google Recommendation 6).
- Add hyperlinks returning to the main Web page from the top and bottom of each Web page (W3C Recommendation 2.4 and Google Recommendation 3).
- Create hyperlinks between the Web pages to improve data exploration via navigation (W3C Recommendation 1.3.2 and 2.4 and Google Recommendation 3 and 5).
- Create content with well-structured sentences (W3C Recommendation 3 and Google Recommendation 4).
- Use text to describe images when the attribute "alt" does not suffice (W3C Recommendation 1.1.1 and Google Recommendation 7).

In addition to these recommendations, W-Ray offers the following guidelines that the designer should follow when designing Web pages:

G12. The structure of the Web site that will contain the materialized view data should be decided upfront and should reflect the set of views.

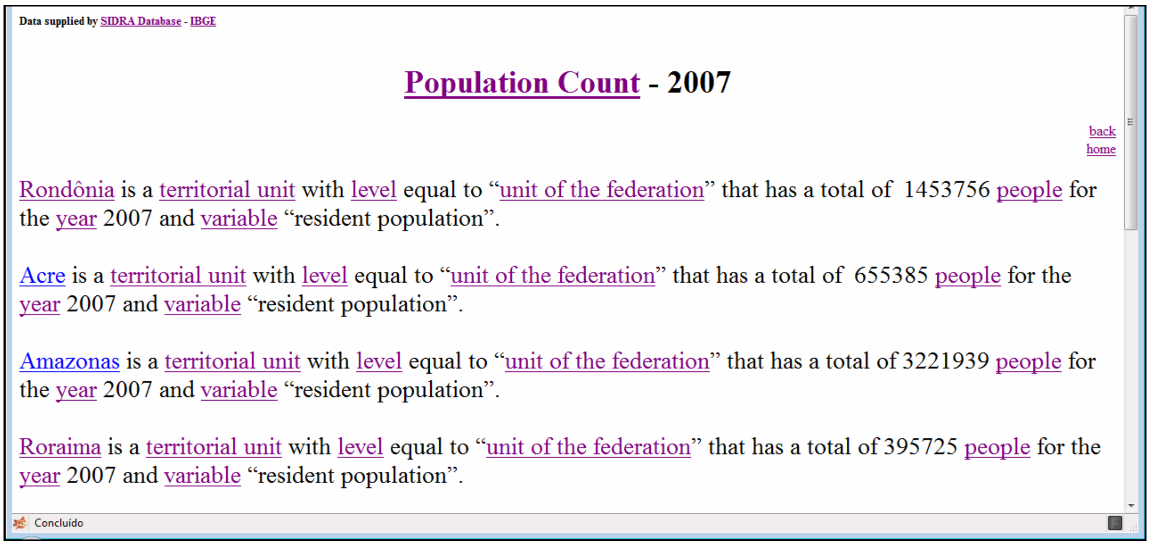
G13. The Web site home page should describe the database and the views to be published; the view and attribute names occurring in the home page should be hyperlinked to the terms of the external vocabularies.

G14. Sentences generated from the same materialized view should be grouped in one or more Web pages, hyperlinked from the home page and among themselves; data coming from domains defined by enumeration should be hyperlinked to the corresponding terms of the external vocabularies.

Example 6: Continuing with our running example, Figure 1 shows a fragment of a Web page containing SIDRA data.

Consider, for example, the first sentence in Figure 1:

- The subject of the sentence, "*Rondônia*" was hyperlinked to an IBGE Web page with further information about the State of Rondônia.
- The terms "*territorial unit*", "*level*", "*unit of federation*", "*people*", "*year*" and "*variable*" were hyperlinked to IBGE Web pages which explain their meaning; these Web pages in turn pointed to the corresponding terms of the external vocabularies, indicated in Table 3. □


Data supplied by SIDRA Database - IBGE

Population Count - 2007

back
home

Rondônia is a territorial unit with level equal to "unit of the federation" that has a total of 1453756 people for the year 2007 and variable "resident population".

Acre is a territorial unit with level equal to "unit of the federation" that has a total of 655385 people for the year 2007 and variable "resident population".

Amazonas is a territorial unit with level equal to "unit of the federation" that has a total of 3221939 people for the year 2007 and variable "resident population".

Roraima is a territorial unit with level equal to "unit of the federation" that has a total of 395725 people for the year 2007 and variable "resident population".

Concluído

Figure 1 – Sentences generated in Example 1 and published as a Web page.

3 W-Ray for geographical data in vector format

We first observe that a number of tools [21] offer facilities to convert geographic data in vector format to dynamic Web pages. However, such dynamic Web pages are typically not indexed by search engines. We also observe that geographic data in vector format is not opaque, as raster images are, since the data is often associated with conventional data and, in fact, with the (geographic) objects stored in the database. A solution to make vector data visible to the search engines would therefore be to publish the conventional data associated with the vector data, as discussed in Section 2. This strategy would however totally ignore the geographic information that the vector data capture.

On a first approximation, the W-Ray strategy for vector data is the same as for conventional data: define a set of database views that capture which data should be published; materialize the views; translate the materialized data to natural language sentences; and publish the sentences as static Web pages.

More specifically, suppose that the vector data is organized by layers. Then, W-Ray offers the following guidelines that the designer should follow when defining the views:

G15. The view definition should combine a small number of layers that contain interrelated objects.

G16. For each layer, the view definition should include a restriction that filters out unimportant objects.

G17. For each layer, the view definition should select a few relevant attributes.

G18. When the view combines several layers,

- G18a. The view definition should specify the priority between the layers.
- G18b. The view definition should specify which topological relationships between the objects of different layers should be materialized.
- G18c. The view definition should indicate in which topological order the objects will be described.

As for conventional data, the designer should select the external names preferably from a controlled vocabulary, such as the ISO19115 Topic Categories [16].

Example 7:

(a) We considered a view consisting of three layers, territorial units, populated places and waterways of Brazil, filtered as follows:

- territorial units: keep only territorial units located in the north region and which are states, with their names, abbreviated names, area and population.
- populated places: retain only populated places located in the north region and which are municipal and state capitals, with their names, political status, area and population.
- waterways: keep only waterways that cross the north region, with their names, navigability and flow conditions.

Furthermore, we assumed that the topological relationship between populated places and states is *'is located in'* and that between waterways and states is *'crosses'*. We also assumed that populated places are listed from north to south and from west to east.

The reader may verify that this informal view definition meets guidelines G15 to G18.

(b) We created the following (free) templates for cities and waterways (the intended meaning of the variables is immediate):

*The city of **C** is located in the state of **S** and has an area of **A** square kilometers.*

*The waterway **W** crosses the state of **S**, with flow **F** and navigability condition **V**.*

Note that the first template refers to a populated place as a city since the view definition retains only municipal and state capitals. Likewise, the first and second templates refer to a territorial unit as a state since again the view definition retains only states.

We then materialized the view and generated sentences such as these:

> *The city of* **<u>Boa Vista</u>** *is located in the state of* **Roraima** *and has an area of* **5,687** *square kilometers.*
>
> *The waterway* **<u>Amazonas</u>** *crosses the state of* **Amazonas,** *with flow* **permanent** *and navigability condition* **navigable**.

The subject of each sentence (underlined words) were hyperlinked to a dynamic Web page with the full information about the city or the waterway, generated by executing a query over the underlying database.

(c) Using the design tool, we generated default templates as follows. First, we created the following entity-relationship model for the views informally defined in Part (a) of the example:

```
entity(territorial_unit,name).
entity(populated_place,name).
entity(waterway,name).
attribute(territorial_unit,population).
Attribute(territorial_unit,abbreviated_name).
attribute(territorial_unit,area).
Attribute(populated_place,level).
Attribute(populated_place,local_area).
Attribute(populated_place,local_population).
Attribute(waterway,flow).
Attribute(waterway, navigability).
Relationship(located_in,[populated_place, territorial_unit]).
Relationship(crosses, [waterway, territorial_unit]).
```

From this model, the design tool generated the following default templates:

```
'There is a populated place with name C'.
'There is a territorial unit with name S'.
'There is a waterway with name W'.
'The flow of the waterway W is F'.
'The navigability of the waterway W is V'.
'The populated place C is related to the territorial unit S by
located in'.
'The waterway W is related to the territorial unit S by
crosses'.
```

Finally, the design tool generated sentences from the default templates and the materialized view data, such as these (with view data in boldface):

```
'There is a populated place with name Boavista'.
'There is a territorial unit with name Amazonas'.
'There is a territorial unit with name Pará'.
'There is a waterway with name Amazonas'.
'The flow of the waterway Amazonas is permanent'.
'The navigability of the waterway Amazonas is navigable'.
```

```
'The populated place **Boavista** is related to the territorial
unit **Roraima** by located in'.

'The waterway **Amazonas** is related to the territorial unit **Ama-
zonas** by crosses'.

'The waterway **Amazonas** is related to the territorial unit **Pará**
by crosses'.
```

(d) The default templates indeed generated regular sentences, but with a fair amount of redundancy. Using just states and waterways as an example, we then used the tool to modify the default templates as follows:

Default template:	`'There is territorial unit with name` ***S***`'`
New template:	`'The state of` ***S***`'`
Default template:	`'The waterway` ***W*** `is related to the territorial unit` ***S*** `by crosses'`
New template:	`'is crossed by the waterway` ***W***`'`
Default template:	`'The flow of the waterway` ***W*** `is` ***F***`'.`
New template:	`'which is` ***F***`'`
Default template:	`'The navigability of the waterway` ***W*** `is` **V**`'.`
New template:	`'and` ***V***`'`

Note again that the new templates took advantage of the fact that the view definition retains only municipal and state capitals from the populated places layer, and only states from the territorial units layer.

Furthermore, we defined the following template composition:

```
facts((territorial_unit(P),crosses(R,P),
       flow(R,S),navigability(R,V))).
```

The tool generated the final template:

```
'The state of S is crossed by the waterway W which is F and V'
```

Finally, the design tool generated sentences from the final template and the materialized view data, such as these (with view data in boldface):

```
'The state of **Amazonas** is crossed by the waterway **Amazonas**
which is **permanent** and **navigable**'.

'The state of **Pará** is crossed by the waterway **Amazonas** which is
**permanent** and **navigable**'.
```

4 W-Ray for raster data

Following Leme et al. [18], the W-Ray strategy describes raster data by publishing sentences that capture the metadata about how the raster data was acquired, and sentences describing the geographic features contained within the bounding box of the raster data. The geographic features might be obtained, for example, from a gazetteer or a geographic database.

For example, one may adopt the ADL gazetteer [1], which holds over 4 million entries and includes a useful Feature Type Thesaurus (FTT) for classifying geographic features. In this case, the view definition would retrieve from the ADL gazetteer all geographic features whose centroids fall within the bounding box of the raster data. The view definition might also restrict the retrieved features to just a few types, using terms from the ADL FTT.

As for vector data, the designer should define a view, this time based on the attributes of the features contained in the gazetteer (or the geographic database). In fact, the guidelines for view definition in the context of raster data are essentially the same as for vector data and will not be repeated here.

The sentences associated with a raster dataset might be automatically generated using the following template

The image I contains the T_i *'s "* F_i^1 *", …, "* $F_i^{n_i}$ *".*

where *I* is and external name for the raster dataset, if one exists, T_i is a term from the thesaurus and F_i^j is a feature name of type T_i .

Example 8: To illustrate the W-Ray approach to raster data, we selected:

- An image fragment of the City of Rio de Janeiro from the Web site "Brazil seen from Space" (see Figure 2), whose metadata indeed indicates the coordinates of its bounding box.
- The ADL gazetteer to provide the geographic features and their classifications.

We then defined a view over the ADL gazetteer that selects geographic features classified as 'hydrographic feature', a topic category of ADL FTT, whose centroid is contained in the bounding box of the image.

We then generated sentences to describe the raster image as follows:

1. The georeferencing parameters were extracted from the image. In this case, the image fragment was consistent with a scale of 1:25.000 and had bounding box defined by *((43°15'W, 22° 52' 30"S), (43° 07' 30"W, 23°S))*.
2. By querying the ADL gazetteer using the georeferencing parameters extracted in Step 1 and the ADL FTT term 'hydrographic feature', nine objects were located, the first three being (see Figure 2):

   a. Feature("Rodrigo de Freitas, Lagoa - Brazil", lakes, contains)
   b. Feature("Comprido, Rio – Brazil", streams, contains)

c. *Feature("Maracana, Rio – Brazil, streams, contains)*

3. The results were translated to sentences describing the image, such as these (using the same conventions as in Section 2.3):

The image of **Rio de Janeiro**, **Brazil**, *contains the* ***lake*** *"***Rodrigo de Freitas***". The image of* **Rio de Janeiro**, **Brazil**, *contains the* ***streams*** *"***Comprido***" and "***Mara-cana***".*

where the underlined words form the subject of the sentence, the words in boldface italics are terms from the ADL FTT (under the general term 'hydrographic feature'), and those in boldface denote the names of the geographic features in the ADL gazetteer.

hydrographic features
. . .
. lakes
. seas
. . oceans
. . . ocean currents
. . . ocean regions
. streams
. . rivers
. . . bends (river)
. . . rapids
. . . waterfalls
. . springs (hydrographic)

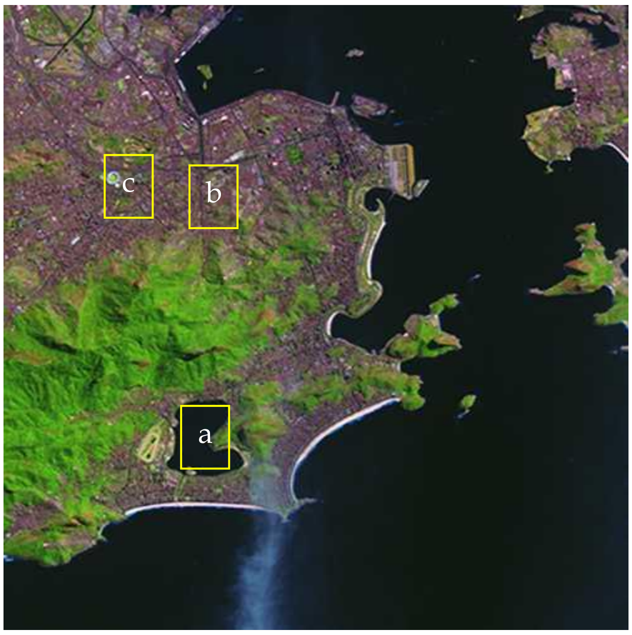


Figure 2. Fragment of ADL Feature Type Thesaurus and image fragment of Rio de Janeiro (Source: Embrapa. Scale: 1:25.000).

5 Conclusions

This paper outlined an approach to overcome the problem of accessing geographic data from the Deep Web. The approach relies on describing the data through natural language sentences, published as Web pages. The Web pages thus generated are easily indexed by traditional search engines, but they also facilitate the task of engines that support semantic search based on natural language features.

We implemented a proof-of-concept Web site, using the design tool to publish a subset of the SIDRA database as natural language sentences. The Web site was then left to be indexed by the Google crawler. Keyword searches were then submitted to Google, which retrieved the correct Web pages from the experimental Web site as expected. Further work is planned to assess which of the three alternatives for generating templates leads to better recall. The experiments will use massive amounts of data from geographic databases organized by IBGE.

Lastly, we remark that the approach can be easily modified to generate RDF triples, instead of natural language sentences, and to cope with multimedia data. In a broader perspective, it can also be used to describe databases of various kinds to the visually impaired users. The challenges here lie in structuring the sentences in such a way to avoid cognitive overload.

Acknowledgements. This work was partly supported by IBGE, for H. Piccinini, and by CNPq under grants 301497/2006-0, 557128/2009-9, and CAPES/PROCAD NF 21/2009, for M.A. Casanova and A.L. Furtado.

6 References

[1] ALEXANDRIA DIGITAL LIBRARY PROJECT. Guide to the ADL Gazetteer Content Standard, v. 3.2 (Feb. 26, 2004). Accessible at: http://www.alexandria.ucsb.edu/gazetteer/ContentStandard/version3.2/GCS3.2-guide.htm

[2] BERGMAN, M. K. The Deep Web: Surfacing Hidden Value. Journal of Electronic Publishing 7, 1 (Aug. 2001).

[3] BIZER, C. and CYGANIAK, R. D2R Server – Publishing Relational Databases on the Web as SPARQL Endpoints. In WWW 2006: Proceedings of the 15th World Wide Web Conference (Edinburgh, Scotland, May 2006), ACM Press.

[4] BRIGHTPLANET. The deep web: Surfacing hidden value. (July 2000). Accessible at: http://brightplanet.com

[5] CALDWELL, B., COOPER, M., REID, L. G. and VANDERHEIDEN, G. Web Content Accessibility Guidelines (WCAG) 2.0. W3C Recommendation (11 December 2008). Accessible at: http://www.w3.org/TR/WCAG20/

[6] CALLAN, J. Distributed information retrieval. In Advances in Information Retrieval, The Information Retrieval Series 7, Springer, USA, pp. 127–150.

[7] CHANG, K. C-C., HE, B., LI, C., PATEL, M. and ZHANG, Z. Structured databases on the web: Observations and implications. SIGMOD Record 33, 3, (Sept. 2004), pp. 61–70.

[8] COSTA, L. Esfinge - Resposta a perguntas usando a Rede. In Proceedings of the Conferencia Ibero-Americana IADIS WWW/Internet (Lisboa, Portugal, 2005).

[9] ERLING, O. and MIKHAILOV, I. RDF support in the virtuoso DBMS. In Proceedings of the 1st Conference on Social Semantic Web (CSSW) (Leipzig, Germany, Sept. 26 - 28, 2007), Lecture Notes in Informatics 113, pp. 59–68.

[10] EUROSTAT. Glossary of Statistical Terms. Accessible at: http://epp.eurostat.ec.europa.eu/statistics_explained/index.php/Glossary:Eurostat

[11] FLIEDL, G., KOP, C. and VÖHRINGER, J. Guideline based evaluation and verbalization of OWL class and property labels. Data & Knowledge Engineering 69, 4 (April 2010), pp. 331–342.

[12] FUCHS, N. E., KALJURAND, K. and KUHN, T. Attempto Controlled English for Knowledge Representation. In Reasoning Web 2008, Lecture Notes in Computer Science 5224 (Springer, Berlin / Heidelberg, 2008), pp. 104–124.

[13] GEONAMES. Accessible at: www.geonames.org

[14] GOOGLE. 2008. Google's Search Engine Optimization Starter Guide, Version 1.1. Accessible at: http://chetangole.com/blog/2008/11/official-googles-search-engine-optimization-starter-guide-pdf/

[15] HOLLINK, L., SCHREIBER, G., WIELEMAKER, J. and WIELINGA, B. Semantic Annotation of Image Collections. In Workshop on Knowledge Markup and Semantic Annotation, KCAP'03 (Sanibel, Florida, USA, Oct., 2003), pp. 41-48.

[16] ISO 19115:2003, Geographic Information – Metadata.

[17] KALYANPUR, A., HALASCHEK-WIENER, C., KOLOVSKI, V. and HENDLER, J. Effective NL Paraphrasing of Ontologies on the Semantic Web. In Workshop on End-User Semantic Web Interaction, 4th International Semantic Web conference (Galway, Ireland, Nov. 2005).

[18] LEME, L. A. P. P., BRAUNER, D. F., CASANOVA, M. A. and BREITMAN, K. A Software Architecture for Automated Geographic Metadata Annotation Generation. Proceedings of the First Brazilian e-Science WorkShop (João Pessoa, Brazil, 2007).

[19] MADHAVAN, J., AFANASIEV, L., ANTOVA, L. and HALEVY, A. Harnessing the Deep Web: Present and Future. In Proceedings of the 4th Biennial Conference on Innovative Data Systems Research (CIDR) (Asilomar, California, USA, Jan. 4-7, 2009).

[20] MADHAVAN, J., KO, D., KOT, L., GANAPATHY, V., RASMUSSEN, A. and HALEVY, A. Google's Deep-Web Crawl. Proceedings of the VLDB Endowment 1, 2 (Aug. 2008), pp. 1241–1252.

[21] MAPSERVER. Accessible at: http://mapserver.org/about.html#about

[22] MENG, W., YU, C. T. and LIU, K. L. Building efficient and effective metasearch engines. ACM Computing. Survey 34, 1 (Mar. 2002), pp. 48–89.

[23] OECD. Glossary of Statistical Terms. 2007. Accessible at: http://stats.oecd.org/glossary/index.htm

[24] PRANINSKAS, J. Rapid review of English grammar. Prentice-Hall, NJ, USA (1975).

[25] RAGHAVAN, S. and GARCIA-MOLINA, H. Crawling the Hidden Web. In Proceedings of the 27th International Conference on Very Large Data Bases (Rome, Italy, Sept. 11-14, 2001), pp. 129–138.

[26] RAJARAMAN, A. Kosmix: High Performance Topic Exploration using the Deep Web. In Proceedings of the VLDB Endowment 2,2 (Au. 2009), pp. 1524–1529.

[27] SORRENTINO, S., BERGAMASCHI, S., GAWINECKI, M. and PO, L. Schema Normalization for Improving Schema Matching. Data & Knowledge Engineering 69, 12 (Dec. 2010), pp. 1254-1273.

[28] SUGGESTED UPPER MERGED ONTOLOGY (SUMO). Accessible at: http://www.ontologyportal.org/

[29] ZHENG, Z. AnswerBus question answering system. In Proceedings of the 2nd International Conference on Human Language (San Diego, California, USA, 2002), pp. 399–404.

Annex

```
/* TEMPLATES FOR ER SCHEMAS */

% version 1.6 - 2010
% distributed with absolutely no guarantee
% --- with database connection ---

:- set_prolog_flag(verbose,silent).
:- use_module(library(clpr)).
:- style_check([-singleton,-discontiguous]).
:- set_prolog_flag(toplevel_print_options,[max_depth(50)]).
:- dynamic utemplate/2.
:- dynamic utemplates/1.
:- op(900,fy,not).

% default templates

template(Einst,T) :-
  entity(E,I),
  Einst =.. [E,Iv],
  name(E,[N1|_]),
  name(aeiou,N),
  (on(N1,N),A = 'an ';
   not on(N1,N),A = 'a '),
  T = ['There is ',A,E,' with ',I,' ',Iv,'.'].

template(Ainst,T) :-
  attribute(E,A),
  Ainst =.. [A,Iv,Av],
  T = ['The ',A,' of ',Iv,' is ',Av,'.'].

template(Rinst,T) :-
  relationship(R,[E1,E2]),
  Rinst =.. [R,Iv1,Iv2],
  T = [Iv1,' is related to ',Iv2,' by ',R,'.'].
```

```
% creates user-defined templates

new_template(E) :-
  entity(E,I),
  name(E,[N1|_]),
  name(aeiou,N),
  (on(N1,N),A = 'an ';
   not on(N1,N),A = 'a '),
  xconc_id(E,Id),
  xclist(['There is ',A,E,' with ',I,' ',Id],Tg),
  nl,write('Default template: '),
  nl,write(Tg),nl,nl,nl,
  write('Please, type sentence with '),
  write(Id),nl,
  write('my choice: '),
  read_line(Tgn),
  (Tgn == stop, !;
   Tgn == nil;
   not (Tgn == nil),
   name(Tgn,Ln),
   repl_n(Id,Ln,Ln1),
   repl_n1([Id],Ln1,Ln2),
   repl_v(Id,V,Ln2,Ln3),
   Ent =.. [E,V],
   (utemplate(Ent,Old), retract(utemplate(Ent,Old));
    not utemplate(Ent,_)),
   assert(utemplate(Ent,Ln3))).

new_template(A) :-
  attribute(I,A),
  xconc('id-',I,E),
  xconc('val-',A,Val),
  xclist(['The ',A,' of ',E,' is ',Val],Tg),
  nl,write('Default template: '),
  nl,write(Tg),nl,nl,nl,
  write('Please, type sentence with both '),
  write(E),write(' and '),write(Val),nl,
  write('my choice: '),
  read_line(Tgn),
  (Tgn == stop, !;
   Tgn == nil;
   not (Tgn == nil),
   name(Tgn,Ln),
   repl_n(E,Ln,Ln1),
   repl_n(Val,Ln1,Ln2),
   repl_n1([E,Val],Ln2,Ln3),
   repl_v(E,V1,Ln3,Ln4),
   repl_v(Val,V2,Ln4,Ln5),
   Attr =.. [A,V1,V2],
   (utemplate(Attr,Old), retract(utemplate(Attr,Old));
    not utemplate(Attr,Old)),
   assert(utemplate(Attr,Ln5))).

new_template(R) :-
  relationship(R,[I1,I2]),
  (I1 == I2, xconc('id1-',I1,E1),
             xconc('id2-',I2,E2);
   not (I1 == I2), xconc('id-',I1,E1),
                   xconc('id-',I2,E2)),
  xclist([E1,' is related to ',E2,' by ',R],Tg),
  nl,write('Default template: '),
  nl,write(Tg),nl,nl,nl,
  write('Please, type sentence with both '),
  write(E1),write(' and '),write(E2),nl,
  write('my choice: '),
  read_line(Tgn),
  (Tgn == stop, !;
   Tgn == nil;
   not (Tgn == nil),
   name(Tgn,Ln),
   repl_n(E1,Ln,Ln1),
   repl_n(E2,Ln1,Ln2),
   repl_n1([E1,E2],Ln2,Ln3),
   repl_v(E1,V1,Ln3,Ln4),
   repl_v(E2,V2,Ln4,Ln5),
   Rel =.. [R,V1,V2],
   (utemplate(Rel,Old), retract(utemplate(Rel,Old));
    not utemplate(Rel,Old)),
   assert(utemplate(Rel,Ln5))).

repl_n(C,[],[]).
repl_n(C,[A|R],V) :-
  name(C,N),
  append(N,V1,[A|R]),
  repl_n(C,V1,V2),nl,
  V = [C|V2].
repl_n(C,[A|R],[A|S]) :-
  name(C,N),
  not append(N,_,[A|R]),
  repl_n(C,R,S).

repl_n1(L,[],[]).
repl_n1(L,[A|R],[Li|S]) :-
  on(Li,L),
  append([Li],V,[A|R]),
  repl_n1(L,V,S).
repl_n1(L,[A|R],[C|S]) :-
  not (on(Li,L), append([Li],_,[A|R])),
  repl_n2(L,[A|R],[],Lc),
  name(C,Lc),
  append(Lc,V,[A|R]),
  repl_n1(L,V,S).
```

```
repl_n2(L,[],[],[]).
repl_n2(L,[Li|R],V,V) :-
  on(Li,L), !.
repl_n2(L,[A|R],V1,[A|V2]) :-
  not (on(Li,L),A == Li),
  repl_n2(L,R,V1,V2).

repl_v(_,_,[],[]).
repl_v(C,V,[T|R],[V|S]) :-
  not var(C),
  T == C,
  repl_v(C,V,R,S).
repl_v(C,V,[T|R],[T|S]) :-
  not var(C),
  not (T == C),
  repl_v(C,V,R,S).

% save, remove or restore the user-defined templates

save_uts :-
  (not utemplates(S),!;
   utemplates(S),
   retract(utemplates(S))),
  findall(U,(U = utemplate(X,Y),U),S),
  assert(utemplates(S)).

remove_uts :-
  (utemplates(_),!;
   save_uts),
  forall(utemplate(X,Y), retract(utemplate(X,Y))).

restore_uts :-
  utemplates(S),
  forall(on(utemplate(X,Y),S), assert(utemplate(X,Y))).

% creating SQL queries

create_queries :-
  forall(entity(E,_), (P =.. [E,_], create_query(P))),
  forall(attribute(_,A), (P =.. [A,_,_], create_query(P))),
  forall(relationship(R,_), (P =.. [R,_,_], create_query(P))).

create_query(T) :-
  sql_query(T,Q),
  assert((T :- Q)).

sql_query(T,Q) :-
  sql_query1(T,Q1),
  B =
  [odbc_connect('XE', _, [ user(prolog), password(prolog),
alias(prolog), open(once) ]),!,
  odbc_query(prolog,Q1,Row),
  T =.. [_|X],
  Row =.. [row|X]],
  conj_list(Q,B).

sql_query1(T,Q) :-
  T =..[E,I],
  entity(E,A),
  (var(I),
   Q1 = ['select ',A,' from ',E];
   not var(I),
   Q1 = ['select ',A,' from ',E,' where ',A,' = ',I]),
  yclist(Q1,Q).

sql_query1(T,Q) :-
  T =..[A,I,V],
  attribute(E,A),
  entity(E,Ai),
  (var(I), var(V),
   Q1 = ['select ', Ai,',',A,' from ',E];
   var(I), not var(V),
   Q1 = ['select ', Ai,',',A,' from ',E,' where ',A,' = ',V];
   not var(I), var(V),
   Q1 = ['select ', Ai,',',A,' from ',E,' where ',Ai,' = ',I];
   not var(I), not var(V),
   Q1 = ['select ', Ai,',',A,' from ',E,' where ',Ai,' = ',I,' and
',A,' = ',V]),
  yclist(Q1,Q).

sql_query1(T,Q) :-
  T =..[R,I1,I2],
  relationship(R,[E1,E2]),
  entity(E1,Ai1),
  entity(E2,Ai2),
  (var(I1), var(I2),
   Q1 = ['select ', E1,',',E2,' from ',R];
   var(I1), not var(I2),
   Q1 = ['select ', E1,',',E2,' from ',R,' where ',Ai2,' = ',I2];
   not var(I1), var(I2),
   Q1 = ['select ', E1,',',E2,' from ',R,' where ',Ai1,' = ',I1];
   not var(I1), not var(I2),
   Q1 = ['select ', E1,',',E2,' from ',R,' where ',Ai1,' = ',I1,'
and ',Ai2,' = ',I2]),
  yclist(Q1,Q).

% single-query execution

sql_query(T) :-
  sql_query(T,Q),
  Q,
  show(T).

% display database tables
```

```
table_info :-
  odbc_connect('XE', _, [ user(prolog), password(prolog),
alias(prolog), open(once) ]),!,
  nl,
  forall(
  (odbc_query(prolog,
  'select us-
er_tab_columns.table_name,user_tab_columns.column_name,user_tab_co
lumns.data_type
  from user_tab_columns, user_tables
  where user_tab_columns.table_name = user_tables.table_name',
  row(T,C,D)), not (T == 'HTMLDB_PLAN_TABLE')),
  (write(T), ttt, write(C), write(' ('),
   downcase_atom(D,D1), write(D1), write(')'), nl)).

ttt :-
  stream_property(S,alias(user_output)),
  stream_property(S,position(Pos)),
  Pos =.. [F,A,B,P,C],
  P1 is 19 - P,
  tab(P1),
  write(' -  ').

% current workspace facts and sql queries

list_ws_facts :-
  forall((entity(E,_),F =.. [E,Ai], clause(F,B), B ==true),
    portray_clause(F)),
  forall((attribute(_,A), F =.. [A,I,V], clause(F,B), B ==true),
    portray_clause(F)),
  forall((relationship(R,_), F =.. [R,I1,I2], clause(F,B), B
==true),
    portray_clause(F)).

list_sql_queries :-
  forall((entity(E,_),F =.. [E,Ai], clause(F,B), not (B==true)),
    portray_clause((F :- B))),
  forall((attribute(_,A), F =.. [A,I,V], clause(F,B), not
(B==true)),
    portray_clause(F :- B)),
  forall((relationship(R,_), F =.. [R,I1,I2], clause(F,B), not
(B==true)),
    portray_clause(F :- B)).

% to display, based on templates, all facts available

facts :-
  nl,
  forall((fact(F), F), (facts(F),nl)), nl.

show :- facts.
```

```
% to display, based on templates, all workspace facts

ws_facts :-
  forall((entity(E,_),F =.. [E,Ai], clause(F,B), B ==true),
    (xtemplate(F,P), xclist(P,S), write(' '), write(S), nl)),
  forall((attribute(_,A), F =.. [A,I,V], clause(F,B), B ==true),
    (xtemplate(F,P), xclist(P,S), write(' '), write(S), nl)),
  forall((relationship(R,_), F =.. [R,I1,I2], clause(F,B), B
==true),
    (xtemplate(F,P), xclist(P,S), write(' '), write(S), nl)).

% to display, based on templates, a fact or a conjunction of facts

show(F) :-
  is_conj(F),!,
  facts(F).
show(F) :-
  not is_conj(F),
  facts((F,nil)).

facts(F) :-
  not var(F), F == nil, !.
facts(F) :-
  is_conj(F),
  findall(F,ck_facts(F),T),
  forall(on(Ti,T),
    (nl,forall(on_conj(Fi,Ti),facts(Fi)))),
  nl, nl.
facts(F) :-
  not is_conj(F),
  (ground(F);
   not ground(F),once(F)),
  xtemplate(F,P),!,
  replace('$null$',undefined,P,P1),
  xclist(P1,S),
  write(' '), write(S).

fact(F) :- not var(F), !,
   F =.. [P|_],
   (entity(P,_);
    attribute(_,P);
    relationship(P,_)),
   current_predicate(P/_).
fact(F) :-
   (entity(P,_), F =.. [P,_];
    attribute(_,P), F =.. [P,_,_];
    relationship(P,_), F =.. [P,_,_]),
   current_predicate(P/_).

ck_facts(F) :-
  is_conj(F),
  conj_list(F,L),
  ck_facts1(F,L).
ck_facts1(F,[]).
ck_facts1(F,[nil]).
ck_facts1(F,[F1|R]) :-
  not (F1 == nil),
  fact(F1),
  F1,
  ck_facts1(F,R).

xtemplate(F,T) :-
  (utemplate(F,T);
   not utemplate(F,_),
   template(F,T)).

% to store a series of sentences in a text file

w_show(F) :-
  is_conj(F),!,
  w_facts(F).
w_show(F) :-
  not is_conj(F),
  w_facts((F,nil)).

w_facts(F) :-
  is_conj(F),
  findall(F,ck_facts(F),T),
  prep,
  forall(on(Ti,T),
    (nl(text),
      forall(on_conj(Fi,Ti),w_facts(Fi)))),
  nl(text), nl(text),
  close(text).

w_facts(F) :-
  not var(F), F == nil, !.
w_facts(F) :-
  not is_conj(F),
  (ground(F);
   not ground(F),once(F)),
  xtemplate(F,P),!,
  xclist(P,S),
  write(text,S), write(text,' ').

% to read a series of pre-recorded sentences

r_facts :-
  open_teste,
  nl,
  r_facts1,
  close(text).
```

```
r_facts1 :-
  read_line(text,A),
  (A = end, !;
   A = nil, r_facts1;
   not (A = nil),write(A),nl,r_facts1).

% UTILITIES

xclist([],'').
xclist([A|R],S) :-
  xclist(R,S1),
  xconc(A,S1,S).

xconc(A,B,C) :-
  name(A,L1),
  name(B,L2),
  append(L1,L2,L3),
  rem_under(L3,L4),
  name(C,L4).

rem_under([],[]).
rem_under([95|R],S) :- !,
  rem_under(R,S1),
  append([32],S1,S).
rem_under([X|R],[X|S]) :-
  rem_under(R,S).

yclist([],'').
yclist([A|R],S) :-
  yclist(R,S1),
  yconc(A,S1,S).

yconc(A,B,C) :-
  name(A,L1),
  name(B,L2),
  append(L1,L2,L3),
  name(C,L3).

xconc_id(B,C) :-
  name('id-',L1),
  name(B,L2),
  append(L1,L2,L3),
  name(C,L3).

on(X,[X|_]).
on(X,[Y|R]) :-
  on(X,R).

replace(_,_,[],[]) :- !.
replace(X,Y,X,Y) :-
  atomic(X), !.
replace(X,Y,X,Y) :-
```

```
  var(X), !.
replace(X,Y,Z,Z) :-
  atomic(Z), !,
  not (X = Z), !.
replace(X,Y,[Z|L],[Z1|L1]) :-
  var(Z), !,
  Z1 = Z,
  replace(X,Y,L,L1).
replace(X,Y,[Z|L],[Z1|L1]) :- !,
  replace(X,Y,Z,Z1),
  replace(X,Y,L,L1).
replace(X,Y,Z1,Z2) :-
  Z1 =.. [F|L],
  replace(X,Y,F,F1),
  replace(X,Y,L,L1),
  Z2 =.. [F1|L1].

is_conj((_,_)).

conj_list(true,[]) :- !.
conj_list(X,[X]) :- not X =.. [',',_,_], !.
conj_list((X , C),[X|Z]) :-
  conj_list(C,Z).

on_conj(X,C) :-
  conj_list(C,L),
  on(X,L).

read_line(A) :-
  rl([],L),
  (L == [], A = nil;
   not (L == []),
   name(A,L)).

rl(L,L1) :-
  get0(A),
  (not (A == 10), !,
   append(L,[A],L2),
   rl(L2,L1);
   L1 = L).

read_line(F,A) :-
  rl(F,[],L),
  (L == end_of_file, !, A = end;
   L == [], A = nil;
   not (L == []),
   name(A,L)).

rl(F,L,L1) :-
  get0(F,A),
  (A == -1, !, L1 = end_of_file;
   not (A == 10), !,
   append(L,[A],L2),
   rl(F,L2,L1);
   L1 = L).

open_teste :-
  open('C:/Users/Helena/PUC/2010Casanova/Furtado/testeW-Ray1.htm',
       read,P,
         [alias(text)]).

reset_teste :-
  set_stream_position(text,'$stream_position'(1,1,1,0)).

prep :-

(stream_property(S,alias(text)),stream_property(S,input),!,close(t
ext);
   true),
  open('C:/Users/Helena/PUC/2010Casanova/Furtado/testeW-Ray1.htm',
       append,P,
         [alias(text)]).

% EXAMPLE STATIC SCHEMA

entity(political_division,pdname).
entity(populated_place,popname).
entity(waterway,wname).

attribute(political_division,population).
attribute(political_division,abbreviated_name).
attribute(political_division,area).
attribute(populated_place,administration_level).
attribute(populated_place,local_area).
attribute(populated_place,local_population).
attribute(waterway,flow).
attribute(waterway,navigability).

relationship(located_in,[populated_place,political_division]).
relationship(crosses,[waterway,political_division]).

:- forall(entity(E,_),(dynamic E/1)).
:- forall(attribute(_,A),(dynamic A/2)).
:- forall(relationship(R,_),(dynamic R/2)).

/* current SQL tables

table political_division(pdname, population, abbreviated_name,
area)
table populated_place(popname, administration_level, local_area,
local_population)
table waterway(wname, flow, navigability)
table located_in(populated_place,political_division).
table crosses(waterway,political_division)
```

```
*/


% EXAMPLE USER TEMPLATES

utemplate(political_division(A), ['<p class=MsoPlainText><span
class=spelle><span lang=EN-US style=''mso-ansi-language: EN-
US''>The ', A]).

utemplate(crosses(A, _), ['is crossed by the ',A, ',']).

utemplate(flow(_, A), ['which is ', A]).

utemplate(navigability(_, A), ['and ', A, '.</span></p><p
class=MsoNormal style=''mso-margin-top-alt:auto;mso-margin-bottom-
alt:auto''><span
lang=EN-US style=''mso-ansi-language:EN-US''></span></p></div>
</div> </body> </html>']).

% DEMOS
% shows on the screen
% demo1 :- show((political_division(P),crosses(R,P),
%  flow(R,S),navigability(R,V))).
% stores in the file testeW-Ray1.htm

demo2 :- w_show((political_division(P),crosses(R,P),
   flow(R,S),navigability(R,V))).
```