Profiles Semantics and Matchings Flexibility for Resources Access

Max Chevalier, Chantal Soulé-Dupuy, Pascaline Laure Tchienehom
IRIT, 118 route de Narbonne, 31062 Toulouse cedex 4, France
Max.Chevalier@irit.fr, Chantal.Soulé-Dupuy@irit.fr, Pascaline.Tchienehom@irit.fr

Abstract

Heterogeneity of resources (information, users, hardware devices, etc.) has raised the problem of defining a generic model, which would be used as a basis for describing them in various applications for resources access. In this article, we propose a profile generic model, which describes the logical structure, the contents and the semantics of any resource. Through the exploitation of profiles semantics we show the matchings flexibility that allows interoperability between different profiles and hence between different applications. For that, we define rules for deducing couples of profiles elements that have a compatible semantics and hence that we can match.

1. Introduction

Resources heterogeneity has led to the definition of various models of resources. These models are of different types (document parts, documents, documents collection, thesis, articles, individual user, users group, mobile devices, working environment, etc.). In information retrieval and filtering, for example, one can restitute information according to users granularity (individual user or users group) and/or according to information granularity (document parts, documents, documents collection). The goal is to discover specialized collections in some specific fields, to discover new information or to find all relevant information units for a given need while adapting to characteristics of each user or users groups. There is a multitude of resources access approaches, which try to solve these problems. The heterogeneity of these approaches and the one of the subjacent models give a greater scale to problems related to generic models definition for the design of systems and for the interoperability between different models and applications.

In this article, we are interested in the definition of a profile generic model, which allows describing any type of profile for resources access. The specificity of this generic model is related to the integration of semantics, which enable the cooperation or interoperability between different profiles models. This profile generic model describes the profiles logical structure, contents and semantics. These semantics will allow the construction of a RDF/RDFS/OWL semantic graph, by combining various profiles instances. The objective of this semantic graph is to identify couples of descriptive profiles elements, which have compatible semantics (couples that we can match). For that, rules based on semantics are clarified. We also show that these rules improve the cooperation between profiles described by different taxonomies (logical structures) through flexible matchings.

Note that what we call resources access here is a broader view of information access where resources are not limited to information (documents) and users but can be extended to any kind of elements depending of the application: mobile devices, working environment, etc.

2. Literature review

Access techniques to information allow an individual to obtain information that meets his needs. We can gather them in two main groups: the pull technique, which needs an explicit request of an individual and the push technique, which does not need an explicit demand to return information to users.

Information Retrieval (IR), which is a pull technique, rests on need expression of an individual through a query formulated in a more or less structured free language [1]. However, in Information Retrieval, the real intention of the user is not always obvious in his manner of formulating his query and that can generate ambiguities on the sense of words that it contains. Many solutions exist for refining the sense of a query through query reformulation [5] [22] [6].

Information Filtering (IF), which is a push technique, is a relatively passive [4] task because the user does not explicitly formulate his needs through a query, as it is the case in IR. In Information Filtering, we rather use a representation of the user called user profile to send information to him.

There are several methods of filtering [19] based on users: interests centers [20]; judgements [15] [5]; demographic data (age, profession, etc.) [17]; or a combination of filtering methods [2]. There are also Context-Aware Applications which take into account the nature of information placed at disposal, the software and hardware used (PC, mobile phone, etc.), the geographic situation of the user [18] [13].

Consequently, there is a multitude of information/resources access methods. They are based on a description of the data handled by processes of retrieval and filtering that are called profile. The profile of an object is a whole of characteristics, which allows to identify and to represent it. The profiles used in information access techniques are of varied nature: user profile, document profile, hardware devices, etc. Their structure can be made up of one or several descriptive elements (or criteria or attributes): centers of interests, data demographic, user preferences, key words, documents metadata, etc. The semantics of these profiles attributes in traditional information access is generally considered as implicit and depends strongly on the application. That poses the problem of profiles co-operation described by different structures and taxonomies (attributes names). Consequently, there is a need of: generic models [16]; semantic models [10]; extensible, flexible, re-usable and interoperable models [3]. Our contribution aims at proposing solutions in this framework for improving resources access. There are existing approaches which also used semantics like CC/PP (*cf.* http://www.w3.org/TR/CCPP-struct-vocab/) or CSCP [7]. Those approaches aim at describing user context through the capability of their devices. The main difference with our proposal is the genericity of our model to any kind of resource.

3. Defining profiles for resources access

In this section, we present a profile generic model for the structure, contents and semantics description of any profiles for resources access. Thereafter, we describe a profiles semantic graph, which combines instances of the generic model, allowing profiles cooperation describe by different logical structures in order to automatically infer attributes couples of compatible semantics.

3.1. Profile generic model

In order to be able to define various profiles, which are reusable, multi-facets, adaptable, extensible and evolutionary, we define a profile generic model.

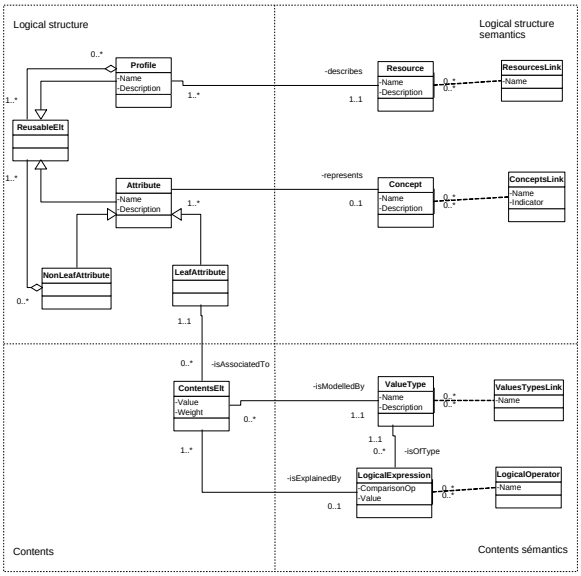

Fig. 1 Profile generic model

The figure *Fig. 1* presents the profile generic model (in UML) proposed. It results from the analysis of various systems of retrieval and recommendation in order to deduce a general model from them. The existing systems are conceived to achieve particular goals according to specificities of their context: recommendation of Web pages according to bookmarks [23], mails filtering [11], electronic trade [9], etc. Contrary to these systems, our model is enough general to be used by various applications.

The profile generic model of figure *Fig. 1* is subdivided into four levels: *the profile logical structure, the profile contents, the profile logical structure semantics and the contents semantics*.

The *logical structure* presents the general structure of a profile. This structure is in the form of a hierarchy of re-usable elements (*ReusableElement* class) that characterize it. This hierarchy is a tree where node or profile elements can be: profiles or attributes. There are two types of attributes: the class *NonLeafAttribute* that represents categories of profiles elements (for example the attribute *user preferences* can be composed of others attributes like: *language, length and date*) and the class *LeafAttribute* that describes leaves attributes to which one can affect values.

Moreover, profiles derived from the generic model can have the following characteristics:

- *re-usable profiles*: in a given profile, a child node can have the structure of another existing profile. For example, a long term user profile can be composed of its various usage profiles (or short term profiles);

- *multi-facets profiles*: profiles can be analysed under various aspects (attributes, sub-profiles). Thus,

each profile or attribute or combination of profiles or profiles attributes can constitute a facet of it. For example, we can analyse a user profile according to facets: *demographic data and judgements, interest centers, etc.*;

- *adaptive and evolutionary profiles*: our profiles can be modified and can evolve in time. For example, a user profile can evolve if many of his short term profiles are different from his long term profile.

The *profile leaves contents* (see class Element) are lists of *Value-Weight* couples. These lists can contain one *Value-Weight* couple (for example the attribute *document size*) or several *Value-Weight* couples (for instance the attribute *document key word*).

The interest of using a generic model to define a given profile is that the basic structure it proposes can be used by any type of application in order to define any type of profiles [8]. The figure *Fig. 2* presents instances of our profile generic model that describe mainly the structure and the contents of a user profile and information profile.

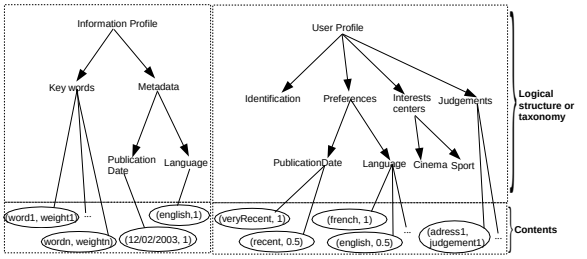

Fig. 2 Information and user profiles examples: structure and contents

The generic model will also enable us to clarify the semantics of a profile logical structure and contents. The *logical structure semantics* of the generic model clarifies what a profile and an attribute represent. A profile is the description of a resource (information, user, etc.) in a given context. Thus, the profiles can relate to users (individual or group) or to information placed at disposal (documents parts, documents, collections, etc.), for instance. Let us note that a user profile can also be: of short term (profiles built over a short period of time) or of long term (profile built over a relatively important period) [24], positive or negative [14].

The figure *Fig. 3* illustrates instances of profiles that are instances of the class “Resource”, with the semantics (instances of the association class “ResourcesLink”), which connects them.

The *attributes semantics* clarifies the characteristic that the attribute describes. The figure *Fig. 4* illustrates an example of profile attributes semantics.

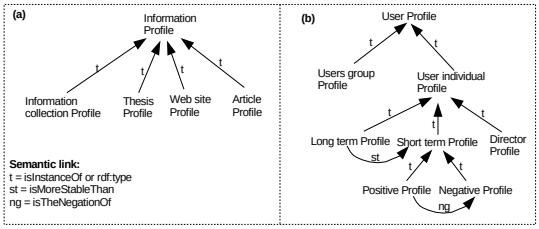

Fig. 3 Semantic relations between instances of the class "Resources"

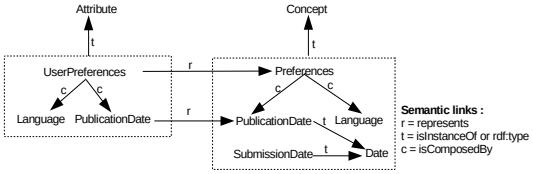

Fig. 4 Instance of profile attributes semantics

The *contents semantics* of a profile clarifies the representation model or type (instance of class *TypeElement*) for contents elements of a leaf attribute (*cf.* XMLSchema element type). The figure *Fig. 5* illustrates semantics instances of contents elements for leaf attributes: *ArticlePublicationDate* and *UserPreferencesPublicationDate*. These two attributes are not represented in the same reference system but it would however be interesting to be able to deduce that we can compare them by extracting year from date (since year is a part of date) and by changing the reference system or vectorial space base. This example shows the interest of clarifying the leaf attributes values or contents semantics which can be done using logical expressions. For instance, in figure *Fig. 5*, a given date *x* is “recent” if *x=2003 OR x=2004 OR x=2005*. All the same, this date is considered “less recent” if *x<2003*.

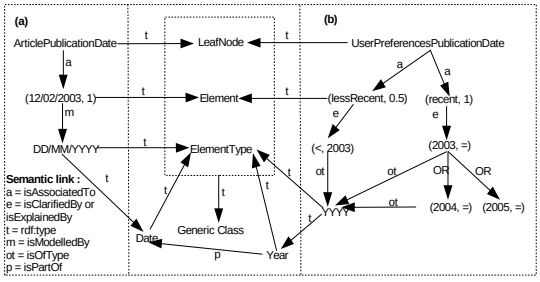

Fig. 5 Instances of leaf attributes values semantics

From the profile generic model, we can derive various profiles structure by applying decomposition rules to *NonLeafAttribute* and *profile* classes.

We can also derive the semantics of profiles, attributes and contents. This will facilitate the co-operation between different profiles in information

access by deducing, through inferences rules, attributes of compatible semantics.

We have chosen RDF/RDFS/OWL descriptive language as a formal framework to combine instances of the profile generic model because those languages are more dedicated for semantics. The RDF/RDFS/OWL formalism is then used to formalize rules for deducing attributes couples of compatible semantics, i.e. that we can match, between two profiles of disjoined logical structure. This aspect of semantics is described and illustrated in the following section.

3.2. Matchings flexibility for resources access

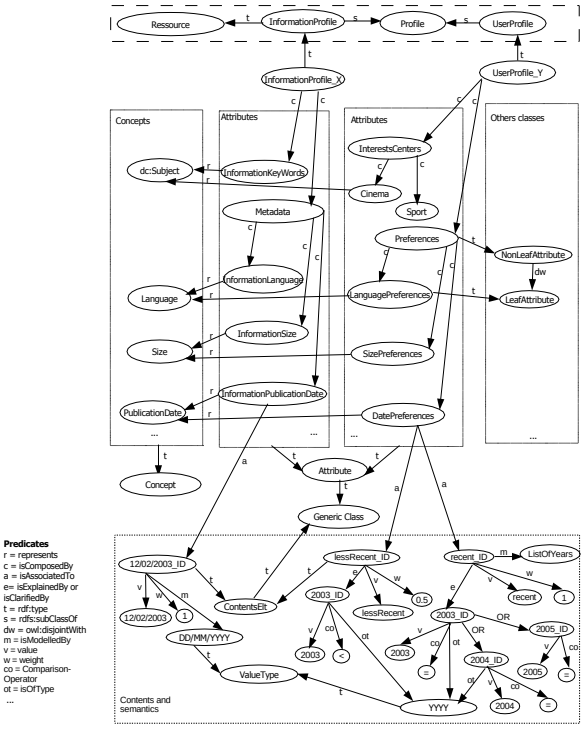

Fig. 6 A semantic graph extract combining profiles instances

To match two different profiles describing various resources instances in different taxonomies (attributes names), it is necessary to be able to determine attributes couples of leaf type that we can match between these profiles. For that, it is necessary to define the semantics of each profile leaf attribute and contents as well as rules that allow the deduction of these couples. The semantics of leaf attributes clarifies the characteristic represented by the attribute while the semantics of a leaf attribute contents elements describes the vectorial space representation and the *values or vectors* (for example, the terms or values lists of the attribute *document key words*) "type" (for instance: string, date, year, different patterns, etc.). We used formalism RDF/RDFS/OWL to formally clarify the inference rules of attributes pairs that we can match. For that, we built a semantic graph, which combines profiles instances derived from the generic model of figure *Fig. 1*. Any semantic relation in our graph is thus defined in the shape of a triplet as follows: *[subject, predicate, object]*.

The figure *Fig 6* presents an extract of semantics description of certain information and user profiles attributes. This extract puts forward the interest of the *Semantic Web* for profiles description [10] and especially for profiles matching. This graph can be seen like task ontology for the profiles matching. The resources of our semantic graph are:

- preset matching classes describing a concept or characteristic represented by an attribute. For that, we generally re-use concepts define in standards like *Dublin Core* or existing ontologies;
- classes representing profiles logical structure and contents;
- classes giving additional information on the representation of leaf attributes contents elements like: the measuring unit used for an attribute evaluation (*terms number or bytes* for the attribute *length*), the representation type of leaf attribute contents element value (text, numerical, date formats, etc.), the reference system representation or vector space (list of elements values), etc. Some contents elements can be clarified through the use of logical expressions in order to avoid the definition of many types that are slightly different.

To describe the semantic relations, we had to define certain predicates as: *represents* (noted *r*) to clarify the characteristic represented by an attribute, *isComposedBy* (noted *c*) to represent the composition link (between attributes and/or profiles), *isModelledBy* (noted *m*) to define the type of contents elements, etc. These predicates are supplemented by RDF/RDFS/OWL predicates which make it possible to typify the various elements of a triplet (*rdf:type* to subsume a class, *rdfs:subClassOf* to subsume all the instances of a class, etc.) and to define constraints on sets of classes (*owl:disjointWith* to define disjunction between two classes, *owl:SymmetricProperty* to set a property as being symmetric, etc).

For defining our rules, we consider that all the contents elements of a leaf attribute are of the same type (i.e they have the same instance of class *ValueType*). In the same way, all the instances of class "LogicalExpression" that clarified the contents elements of this attribute are also of the same type. So, two leaf attributes can be matched if the following rules are verified:

1. *necessary rule*: the two leaf attributes must describe the same concept or matching class (*dc:Subject, Language, Length, etc.*). They have to be connected to the same semantic concept class by the predicate "represents" (noted *r*). Formally, the *necessary rule* for considering a matching between two attributes *x* and *y*, noted *necessary_rule(x, y)*, is written:

Given *A* the profiles attributes set, *C* the concepts classes set, *G* the triplets set of the profiles semantic graph and given $x, y \in A$, *x* and *y* can be matched if and only if, $[x, rdf:type, LeafAttribute], [y, rdf:type, LeafAttribute] \in G$ and $\exists a \in C$ so that $[x, r, a]$ and $[y, r, a] \in G$.

2. *necessary and sufficient rule*: to carry out matching between two attributes, it should be checked that these attributes have the same semantic links or predicates (in term of number and type) which start from these attributes towards the same classes. If the semantic links are the same but are not always defined towards the same classes, it should be checked if the object of the triplet is an instance of class *ContentsElt*. If it is the case, it is necessary to proceed to a *"type verification"* which seeks the semantics of the various contents elements and verifies that they are of the same class or that there is a transformation rule between these content elements semantics, which are instances of class *ValueType*. For example in figure Fig 6, to match the *InformationPublicationDate* attribute and the *DatePreferences* attribute it is necessary that there is a rule that allows to pass from class *DD/MM/YYYY* to class *YYYY*.

Thereafter, it is necessary to verify the dimension (number of contents elements associated to the attribute) and the reference system or vector space (elements values list) of each leaf attribute. There could be either a disjunction or an inclusion or an overlapping as well between the terms (or values) lists of the leaf attributes. In order to perform the matching, it may be necessary to carry out:

- a reference system change (vectorial space change), if one leaf attribute contents elements are clarified through logical expressions which have a type different from the one of contents elements;
- or simply a dimension change in order to reduce the attributes to the same dimension.

Formally, the necessary and sufficient rule for matching two attributes *x* and *y*, noted *necessary_and_sufficient_rule(x, y)*, is written:

Given *A* the set of profiles attributes, *G* the triplets set of the semantic graph, *E* the semantic graph instances set of classes *ContentsElt and LogicalExpression*, *P* the set of predicates, *searching(a, a1, a2)* a method that seeks the classes *a1* and *a2* that are the *ValueType* class instances related to the value *a* (*a* here corresponds to the property *"Value"* of class *ContentsElt*), *TransformationRule(a1, b1)* a rule allowing the transformation of a triplet $[x1, rdf:type, a1]$ into a triplet $[x1, rdf:type, b1]$ or conversely and given $x, y \in A$, *x* and *y* can be matched if and only if, $[x, rdf:type, LeafAttribute] \in G$, $[y, rdf:type, LeafAttribute] \in G$ and $\forall [x, p, a] \in G, \exists [y, p, b] \in G$ so that $p \in P$ and

(a). $a=b$

(b). or $a, b \in E$ and if $\exists [a, rdf:type, ContentsElt], [b, rdf:type, ContentsElt] \in G$ then we execute the methods *searching(a, a1, a2)* and *searching(b, b1, b2)* which return *a1* and *b1* that are *ValueType* class instances of the values *a* and *b* respectively. They also return *a2* and *b2* that are *ValueType* class instances of the values of instances of class *LogicalExpression*, which correspond to the property "Value" of that class and that clarified *a* and *b* respectively. Then:

(i). if $\exists$ *TransformationRule(a1, b2)* or $\exists$ *TransformationRule(a2, b1)*, it is necessary to carry out a reference system change between *A1* (contents elements values list of attribute *x*) and *B1* (contents elements values list of attribute *y*). For example, in figure *Fig. 6*, it is necessary to express the value of *InformationPublicationDate* attribute, initially expressed in the reference system "(12/02/2003)", into the reference system of *DatePreferences* attribute, which is *"(lessRecent, recent)"*;

(ii). If $\exists$ *TransformationRule(a1, b1)*, it is just necessary to change the dimension.

Let us note that the function *searching(a, a1, a2)*, with $a \in E$, is defined as follows:

- $\exists$ *[a, rdf:type, ContentsElt]* and $\exists$ *[a, isModelledBy, a1]* and $\exists$ *[a1, rdf:type, ValueType]* $\in$ G
- *if* $\exists$ *[a, isExplainedBy, v1]* $\in$ *G then* $\exists$ *[v1, rdf:type, LogicalExpression] and* $\exists$ *[v1, isOfType, a2]*

To match two attributes, it is necessary to check the coherence of their semantics (characteristic represented, contents semantics). For that, we have defined some transformations rules for attributes, which have compatible semantics. Among these rules, we can quote: transformation of monovalued attributes into multivalued attributes for a dimension change, reference system change, etc.

Table 1 illustrates a dimension and vector space (or reference system) change using *scalar product* between two attributes p_d and p_u that represent respectively the publication year of an article and the

preferences of a user in terms of article years publication. Initially, p_d is the year "2003" and the user preferences are lists of years represented by the values "*lessRecent*" (for years before 2003) and "*recent*" (for years 2003, 2004 and 2005). In order, to compare those two attributes we must express them in the same reference space, here the base *u* representing the terms "*lessRecent*" and "*recent*". We express p_d in base *u* and we measure the similarity using the *cosine formula*. For writing p_d in *u* base, we evaluate it correspondence to logical expressions linked to values "lessRecent" (one logical expression) and "recent" (disjunction of logical expressions).

Attribute *date*					
***u* base : t_i**	**LessRecent (t_1)**	**Recent (t_2)**			**Similarity**
***d* base : v_i**	**<2003 (v_1)**	**=2003 (v_2)**	**=2004 (v_3)**	**=2005 (v_4)**	
p_d *in d*	*0*	*1*	*0*	*0*	
p_u *in d*	*1*	*1*	*1*	*1*	
p_d *in u*	$w_{d,t1}$=0	$w_{d,t2}$=1			$s(p_d,p_u)$ =0,89
p_u *in u*	$w_{u,t1}$=0.5	$w_{u,t2}$=1			

Table 1. Dimension and vector space (or reference system) change

The interest of a semantic graph of profiles as the one of figure *Fig. 6* is that it makes it possible to give the names which one wishes to profiles attributes without disturbing the matching. We only have to specify their semantics. Moreover, one will be able to match profiles from various applications and/or described by different taxonomies. To determine attributes couples that we can match, we use a parser or RDF analyzer that being given a RDF document, returns all the triplets *[subject, predicate, object]* of this document. It is the set of triplets obtained (*G*) that is analyzed in order to determine attributes couples of compatible semantics. We can also manipulate our profiles using RDF query languages [12]. Examples of figure *Fig. 6* triplets are: [Cinema, represents, dc:Subject], [PublicationDate, rdf:type, Concept], [2003_ID, isOfType, YYYY], etc.

For implementing the rules previously defined, we use an RDF query language (RDQL) that is combined with Java programming language through an API called *Jena* (*cf.* http://jena.sourceforge.net).

3.3. RDF query language for matchings flexibility

The general environment for profiles matching is described in figure *Fig. 7.* For identifying leaf attribute of compatible semantics between two profiles, we use:

- *general ontologies* that are used to verify the semantic compatibility between profiles which have to be matched: equivalence or equality between concepts, relations between types, synonymy between values, etc.;
- *RDF description of profiles* to be matched;
- *indexes* in order to facilitate matchings of some attributes like the *key words of a document* for instance;
- *a dictionary of transformation rules*, between different types of element (instances of class *ValueType*), that describes the methods for moving from one type to another.

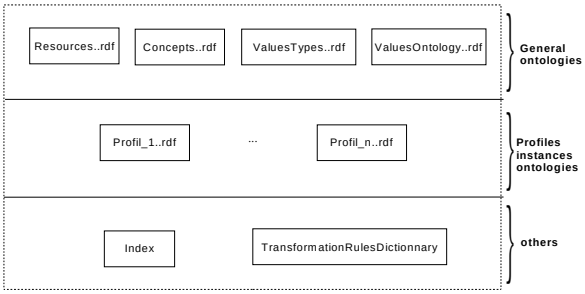

Fig. 7 General architecture for profiles matching

For example, if we want to identify all the couples of leaf attributes of compatible semantics between two profiles *profile_1* and *profile_2*, we can first determine for each leaf attribute (*xi*) its concept (*ci*), its contents elements type (*ai*) and eventually the type of its logical expressions (*bi*). For that, we can made successively the two following RDF query :

```
SELECT ?x1, ?c1, ?a1, ?b1
FROM profile_1.rdf
WHERE (?x1, rdf:type, LeafAttribute)
AND (?x1, sp:represents, ?c1)
AND (?x1, sp:isAssociatedTo, ?a)
AND (?a, sp:isModelledBy, ?a1)
AND (?a, sp:isExplainedBy, ?v1)
AND (?v1, sp:isOfType, ?b1)
USING rdf For <http://www.w3.org/1999/02/22-rdf-
      syntax-ns\#>
      sp For <…>

SELECT ?x2, ?c2, ?a2, ?b2
FROM profile_2.rdf
WHERE (?x2, rdf:type, LeafAttribute)
AND (?x2, sp:represents, ?c2)
AND (?x2, sp:isAssociatedTo, ?a)
AND (?a, sp:isModelledBy, ?a2)
AND (?a, sp:isExplainedBy, ?v2)
```

AND (?v2, sp:isOfType, ?b2)
USING rdf For <http://www.w3.org/1999/02/22-rdf-syntax-ns\#>
sp For <…>

When the concepts of two leaf attributes are not the same (they come from different namespaces or have different names), we can check if they are equivalent or identical. For that, we can use the following query:

SELECT ?x, ?y
FROM concepts.rdf
WHERE (?x, owl:sameAs, ?y)
AND (?x, owl:equivalentClass, ?y)
AND ?x=c1
AND ?y=c2
USING owl For <http://www.w3.org/2002/07/owl#>

Note that the properties *owl:equivalentClass* and *owl:sameAs* are symmetric and that *c1* and *c2* are the names of concepts found for attributes *a1* and *a2* of profiles *profile_1* and *profile_2* respectively.

We can also check the types compatibility of contents elements of two leaf attributes as follow:

SELECT ?x, ?y
FROM values_types.rdf
WHERE (?x, sp:isCompatibleTo, ?y)
AND ?x=a1
AND ?y=a2
USING sp For <…>

In order to detect others types compatibility we can rewrite this query by changing the selection conditions. Thus, the variable *?x* can either be *a1 or b1* and the variable *?y* can be either *a2 or b2.*

The property *sp:isCompatible* is symmetric and *a1, b1, a2* and *b2* are the types names (instance of class *ValueType*) found for attributes *x1* and *x2* contents elements and logical expressions of profiles *profile_1* and *profile_2* respectively. If the elements types are compatible then we check the entry corresponding to those types, in the transformation rules dictionary, in order to have the method description to invoke.

For more matchings flexibility, we can also analyse the values semantics of contents elements, by using values ontology defined, as follow:

SELECT ?v1, ?v2
FROM values_ontology.rdf
WHERE (?v1, sp:isATranslationOf, v2)
OR (?v1, sp:isSynonymousTo, v2)
OR (?v1, sp:isAnAbbreviationOf, v2)
AND ?v1=val_1
AND ?v2=val_2
USING …

Note that the properties *sp:isATranslationOf, sp:isSynonymousTo* and *sp:isAnAbbreviationOf* are symmetric and that *val_1* and *val_2* are two values of contents elements for attributes *x1* and *x2* of profiles *profile_1* and *profile_2* respectively. We use this query when the values lists are disjoined, attributes concepts and contents elements types are compatible. For example, we can have the values *fr* and *french* that represent the same thing since *fr* is an abbreviation of *french*.

The general procedure for matching two profiles is the following:

- identification of leaf attributes of compatible semantics;
- matching of the different couples of leaf attribute;
- aggregation of the different matchings results [8].

4. Conclusion

In this article, we present a generic model of profile that enables us to describe the structure, contents and semantics of various profiles types. We use this generic model to combine instances of profiles through an RDF graph in order to allow interoperability between different profiles.

This graph helps, thanks to some rules, to determine attributes of compatible semantics whatever the profiles taxonomies are and hence optimizes the profiles cooperation. We are now using these inference rules and other general constraints related to the profile generic model in the implementation of an assistant tool for constructing profiles (structure, contents and semantics) and also for performing different profiles matching for recommendations.

10. References

[1] R. Baeza-Yates, and B. Ribeiro-Neto, *Modern Information Retrieval*, First edition, Addison Wesley, ISBN 0-201-39829-X, 1999.

[2] M. Balabanovic, and Y. Shoham, “Fab: Content-Based, Collaborative Recommendations”, *Communications of the ACM*, 1997, vol. 40, n°3, pp. 66-72.

[3] T. Berners-Lee, J. Hendler, and O. Lassila, “The semantic web”, *Scientific american*, 2001.

[4] N. J. Belkin, and W. B. Croft, “Information Filtering and information Retrieval: Two Sides of the same Coin?”, *Communications of the ACM, Information Filtering*, 1992, vol. 35, n°12, pp. 29-38.

[5] M. Boughanem, C. Chrisment, and C. Soulé-Dupuy, “Query modification based on relevance backpropagation in adhoc environment”, *Information Processing & Management Journal*, Elsevier Science, 1999, vol. 35, pp. 121-139.

[6] J. C. Bottraud, G. Bisson, and M. F. Bruandet, “An Adaptative Information Research Personnal Assistant”, *In proceedings of Workshop AI2IA (Artificial Intelligence, Information Access and Mobile Computing) IJCAI'03*, 2003.

[7] S. Buchholz and T. Hamann and G. Hubsch, “Comprehensive Structured Context Profiles (CSCP): Design and Experiences”, *In Proceedings of the Workshop on Context Modeling and Reasoning (CoMoRea'04)*, 2004, pp 43-47/

[8] M. Chevalier, C. Soulé-Dupuy and P. L. Tchienehom, “A profile-based architecture for a flexible and personalized information access”, *IADIS International Conference (IADIS/WWW Internet 2004)*, 2004, vol. 2, pp. 1017-1022.

[9] Y. H. Cho, J. K. Kim, and S. H. Kim, “A personalized recommender system based on web usage mining and decision tree induction”, *Expert System with Applications*, 2002, vol. 23, n°3, pp. 329-342.

[10] P. Dolog, and W.Nejdl, “Challenges and benefits of the Semantic Web for User Modelling”, *In proceeding of AH'03*, 2003.

[11] D.Goldberg, D. Nichols, B. M. Oki, and D. Terry, “Using Collaborative Filtering to weave an Information Tapestry”, *Communications of the ACM, Information Filtering*, 1992, vol. 35, n°12, pp. 61-70.

[12] P. Haase, J. Broekstra, A. Eberhart, and R. Volz, “A comparison of RDF Query Languages”, *In proceedings of the third International Semantic Web Conference ISWC'04*, 2004.

[13] D. Halvatzaras and M. H. Williams, "A context aware user profile for personalization", *IADIS International Conference (IADIS/WWW Internet 2004)*, vol. 1, 2004, pp 452-460.

[14] K. Hoashi, M. Kazunori, I. Naomi, and K. Hashimoto, “Document filtering Method using non-relevant information profile”, *In proceedings of the 23rd Annual International ACM-SIGIR Conference on research and development in information Retrieval*, 2000, pp. 176-183.

[15] J. A. Konstan, B. N.Miller, D. Maltz, J. L. Herlocker, L. R. Gordon, and J. Riedl, “Grouplens: Applying Collaborative Filtering to Usenet News”, *Communication of the ACM*, 1997, vol. 40, n°3, pp. 77-87.

[16] A. Kobsa, “Generic User Modelling Systems”, *User Modelling and User-Adapted Interaction*, 2001, vol. 11, pp. 49-63.

[17] B. Krulwich, “LifeStyle Finder : Intelligent User Profiling Using Large-Scale Demographic Data”, *AI Magazine*, 1997, vol.18, n°2, pp. 37-45.

[18] O. Kwon and K. Yoo and E. Suh, “UbiSS: a proactive intelligent decision support system as an expert deploying ubiquitous computing technologies”, *Expert Systems with Applications*, 2005, pp 149-161.

[19] M. Montaner, B. Lopez, and J. L. D. L. Rosa, “A Taxonomy of Recommender Agents on the Internet”, *Artificial Intelligence Review*, 2003, vol. 19, pp. 285-330, Kluwer Academic Publishers.

[20] M. Pazzani, J. Muramatsu, and D. Billsus, “Syskill & Webert: Identifying interesting web sites”, *In Proceedings of the Thirteenth National Conference on Artificial Intelligence*, 1996, pp. 54-61.

[21] M. Pazzani, “A Framework for Collaborative, Content-Based and Demographic Filtering”, *Artificial Intelligence Review*, 1999.

[22] J. Pitkow, N. Schütze, T. Cass, R. Cooley, D. Turnbull, A. Edmonds, E. Adar, and T. Breuel, “Personalized Search: A contextual computing approach may prove a breakthrough in personalized search efficiency”, *Communications of the ACM*, 2002, vol. 45, n°9, pp. 50-55.

[23] J. Rucker, and M. J Polanco, “Siteseer: Personalized Navigation for the Web”, *Communications of the ACM*, 1997, vol. 40, n°3, pp. 73-75.

[24] D. H. Widyantoro, T. R. Ioerger, and J. Yen, “An Adaptative Algorithm for Learning Changes in User Interests”, *In Proceedings of the Eighth International Conference on Information and Knowledge Management (CIKM'99)*, 1999, pp. 405-412, New York, ACM Press.