AN EXTENSION OF ONTOLOGY BASED DATABASES TO HANDLE PREFERENCES

Dilek TAPUCU[(1)(3)], Yamine AIT-AMEUR[(1)], Stéphane JEAN[(1)] and Murat Osman ÜNALIR[(2)]

[(1)] *LISI/ENSMA,and University of Poitiers, BP 40109, 86961 Futuroscope Cedex, France*
{*dilek.tapucu,yamine,jean*}*@ensma.fr*

[(2)] *Department of Computer Engineering, Ege University, 35100 Bornova, İzmir, Turkey*
murat.osman.unalir@ege.edu.tr

[(3)] *Department of Computer Engineering, Izmir Institute of Technology, 35340 Urla, İzmir, Turkey*

Keywords: User Preferences, Ontology Based Database, Preference based Querying.

Abstract: Ontologies have been defined to make explicit the semantics of data. With the emergence of the SemanticWeb, the amount of ontological data (or instances) available has increased. To manage such data, Ontology Based DataBases (OBDBs), that store ontologies and their instance data in the same repository have been proposed. These databases are associated with exploitation languages supporting description, querying, etc. on both ontologies and data. However, usually queries return a big amount of data that may be sorted in order to find the relevant ones. Moreover, in the current, few approaches considering user preferences when querying have been developed. Yet this problem is fundamental for many applications especially in the e-commerce domain. In this paper, we first propose an extension of an existing OBDB, called OntoDB through extension of their ontology model in order to support semantic description of preferences. Secondly, an extension of an ontology based query language, called OntoQL defined on OntoDB for querying ontological data with preferences is presented. Finally, an implementation of the proposed extensions are described.

1 INTRODUCTION

Nowadays, ontologies are well accepted to describe the explicit semantics of concepts and objects manipulated in a given domain. Domain ontologies are used to provide with definitions and specifications of these manipulated concepts. Several application domains have seen the emergence of ontologies in order to characterise the universe where these models act. Among these application domains, semantic web, databases,ontology engineering are the most well known. Ontologies are described according to ontology models. These models introduce the notion of classes and instances, and a significant amount of classes and instances are defined when ontologies are described. Storing, retrieving and manipulating ontology classes and instances is a major requirement of Ontology engineering. In the last decade, the notion of ontology based database (OBDB) has been developed (Dehainsala et al., 2007; Pierra et al., 2005) in order to offer an infrastructure allowing management of ontologies and their instances (Chong et al., 2005; Petrini and Risch, 2007). At least these models store the ontology and its instances, but some of them also store the ontology model and extensively use meta modeling techniques. However, the currently defined OBDB do not deal with the representation of the non functional aspects related to the ontology models. By non functional aspects, we mean concepts that are capable to describe externally defined properties like quality, preferences or security. Indeed, most of the well known ontology models like Owl, Plib, etc. do not provide with such resources to represent such concepts. Each time non functional concepts are introduced, ad hoc concepts or extensions are introduced in the ontology models like Owl, Plib, etc. or specific attributes like note or remark or a particular property are used to encode these non functional aspects. Rather than extending a specific ontology model, our proposal consists in introducing a side model to describe the non functional concepts together with the ontology model inside an OBDB. The advantage of this approach is the possibility to adopt non functional descriptions to any ontology model keeping its definition unchanged. We particularly study the notion of preference in this context. Technically, this

extension is possible only if the meta-model allows to describe the ontology model can be manipulated. Indeed, such an extension requires to be able to attach ontology model elements to the non functional model element. In our work, with the OntoDB ontology based database, such a manipulation is possible.

This paper describes how a preference model can be attached to an ontology model through the manipulation of the meta-model level. We show how a specific preference model is linked to the concept of class or property of the ontology model. Section 2 is an overview of the material set up in this paper. Section 3 presents the preference model that we have defined, shows how preferences are linked to ontological concepts like classes and/or properties. Handling these preferences in the OntoDB ontology based database is presented in section 4. Section 5 sets up the approach on a case study and finally a conclusion and some perspectives are presented.

2 STATE OF THE ART

This section presents the material needed to set our approach. We first present the ontology based database that support the description of an ontology together with its instances. The second part presents the approaches that have been proposed to handle preferences in the database and semantic web areas.

2.1 Ontology Based Database

In the last years, many OBDB architectures have been proposed. They can be classified in 3 categories according to the number of schemas used.

Type 1 OBDBs. In type 1 OBDBs, information is represented in a single schema composed of a unique triple table (subject, predicate, object) (Chong et al., 2005; Petrini and Risch, 2007; Alexaki et al., 2001; Broekstra et al., 2002; Dehainsala et al., 2007; Pierra et al., 2005). But this approach raises serious performance issues when queries require many self-joins over this table.

Type 2 OBDBs. Type 2 OBDBs store separately ontology descriptions and instance data in two different schemas (Alexaki et al., 2001; Broekstra et al., 2002). The schema for ontology descriptions depends upon the ontology model used to represent ontologies (e.g., RDFS, OWL, PLIB). It is composed of tables used to store each ontology modeling primitive such as classes, properties and subsumption relationships. Separating representation of ontology descriptions and instance data leads to better query response time. However, this approach assumes a fixed ontology model.

Type 3 OBDBs. OntoDB (Dehainsala et al., 2007; Pierra et al., 2005) proposes to add another schema to type 2 OBDBs. This schema called *meta-schema* records the ontology model into a reflexive meta model. For the ontology schema, the meta-schema plays the same role as the one played by the system catalog in traditional databases. Indeed, meta-schema may allow: (1) generic access to the ontology, (2) support of evolution of the used ontology model, and (3) storage of different ontology models (OWL, DAML+OIL, PLIB, etc.). Next, we will present the OntoDB ontological database which we interpreted to extend to handle preferences.

2.2 The OntoDB: Ontology Based Database

In order to set up our approach, is needed to have an infrastructure allowing to encoding ontologies and the defined preference model together with a manipulation language for exploiting the extension to preferences. For our work, the OntoDB ontology based database and the OntoQL language have been chosen. OntoDB ensures models and their instances persistency, whereas OntoQL allows to manage and query ontologies and preferences.

2.2.1 OntoDB Architecture

The OntoDB architecture is composed of four parts presented on Figure 2:

- **The *meta-base* part (1)**. The meta-base, also often called *catalog system*, is a traditional part of databases. It contains system tables used to manage all the data contained in the database. In OntoDB, it contains in particular the description of all tables and columns defined in the three other parts of this architecture.
- **The *data* part (3)**. It represents domain objects described by ontology classes membership and values of properties defined on these classes. These objects are represented according to the table per class approach.
- **The *ontology* part (4)**. It contains ontologies defining semantics of the various domains covered by the database. OntoDB initially supports the PLIB ontology model.
- **The *meta-schema* part (2)**. The meta-schema part records the ontology model used into a reflexive meta-model. For the ontology part, the meta-

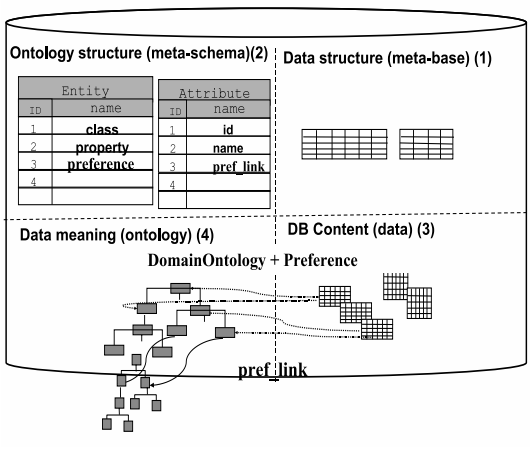

Figure 1: The OntoDB four parts architecture

schema part plays the same role as the one played by the meta-base in traditional databases.

2.2.2 The OntoQL Exploitation Language

The OntoQL exploitation language has been designed to exploit the power offered by the OntoDB architecture (Jean et al., 2005; Jean et al., 2006). Indeed, OntoQL is capable to access and manipulate the model, its instances and its meta-model. For example, when an ontology is stored, OntoQL is capable to manage the ontology (e.g. Hotels, Cities, Prices, rate), itself, its instances (e.g. Ibis_Poitiers, hotel, Sheraton_Paris) and the model that describe the ontologies (e.g. #class, #property, #datatype).

The OntoQL language is equipped with an data definition language DDL (`CREATE, ALTER clauses`),a data manipulation language (`INSERT INTO, DELETE, UPDATE clauses`) and a query language (`SELECT`). These language modules allow to accessing either the ontology model level (by prefixing the accessed date with the # symbol) and the ontology and its instances (when no prefix is available). For example, creating a preference concept at the ontology model level is performed by adding another new concetpt in the `ENTITY` meta-model resource using the following OntoQL clause;

```
CREATE ENTITY #Preference (
oid int,
...
);
```

Then, accessing and querying preferences is performed by;

```
SELECT oid
FROM Preference
WHERE ...
```

2.3 Preferences

Several approaches handling preferences have been proposed in various areas of information systems.

2.3.1 Preferences in databases

Handling preferences in databases has been the subject of several research work (Kieling and Kostler, 2000; Kieling, 2002; Chomicki, 2003; Agrawal and Wimmers, 2000; Koutrika and Ioannidis, 2004; Viappiani et al., 2006). These approaches consider a set of preferences that are evaluated on the logical model of a database. Extensions of the SQL language are defined within a specific preference clause. These approaches are *strongly linked* to the logical model of the database, and therefore it is required for an user to have a good knowledge of this logical model.

2.3.2 Preferences in semantic web

In the context of the semantic Web, preferences have been studied following two main approaches (Siberski et al., 2006), (P. Gursk and Vanekov, 2008), (Toninelli et al., 2008). The first one consists in introducing specific properties in the ontologies, attached to classes. The second one consists in using specific attributes of the ontology model like *note, or definition* to encode the preference. In both approaches, SPARQL queries (SPARQL Query Language for RDF, 2008) take into account property or attribute values. By examining preferences on databases and semantic Web, we identify that they both use definitions of the notion of preferences built on manipulated models themselves. They all introduce the preference notion at the model level. These approaches are static and not flexible enough to handle different preference models.

3 OUR APPROACH

Our approach consists in associating any side preference model to any ontology based database that allows to manipulate the ontology model through its meta-model. Indeed, according to Figure 2, the preference resource concept of the preference model is associated to the class or property concepts available in the ontology model. Let us briefly describe the elements composing these model and link.

3.1 Ontology Resource Definition

The property_or_class resource is introduced in order to attach a preference to an ontology. Moreover, Prop-

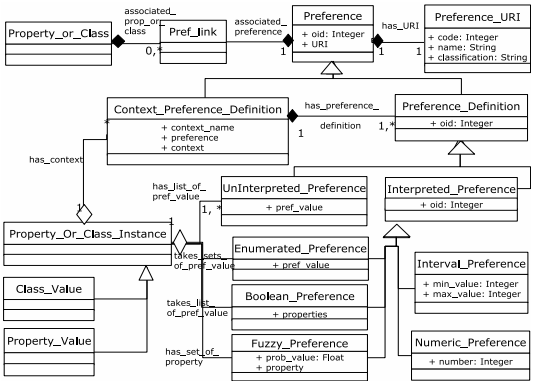

Figure 2: Our Approach.

erty_or_Class_Instance resource is used to define specific instances of the ontology.

3.2 Preference Model

The preference model introduces specific resources allowing to defining preferences. Two categories of preferences are introduced: interpreted and non interpreted ones.

3.2.1 Interpreted preferences

Interpreted preferences are those preferences that can be given an interpretation by means of an evaluation. The nature of their definition depends on the attached interpretation function.

a- *Enumerated preferences* are interpreted by a set of property values or class instances imported from the ontology population. For example a preference can be defined on "*Bob Marley*" as an instance of singer and on "*reggae*" as a property value of a song type. This enumeration is arbitrary defined by the preference modeler.

b- *Numeric preferences* are interpreted by numeric values. For example, the rating of a hotel can be defined as *1, 2, 3, or 4 stars* in a given tourism domain.

c- *Boolean preferences* are interpreted as the presence or the absence of a given feature. For example a preference on hotels can be expressed as the presence of the following triple (*'air conditioning', 'wifi', 'swimming pool'*).

d- *Interval preferences* are interpreted by intervals of values. For example, the *cheap* and *expensive* preferences can be defined respectively by the *[10-20] and [90-100]* intervals.

e- *Fuzzy preferences* are interpreted by a probability rating the presence of a given feature. If we take the previous example, we get (*'air conditioning' 0,9, 'wifi' 0.5, 'swimming pool' 0.2*) meaning that a strongest preference is allowed to air conditioning while having a swimming pool is weaker. Notice that Boolean preference corresponds to a fuzzy preference with a rating value equal 1.

3.2.2 Uninterpreted preferences

Uninterpreted preferences are defined as an enumeration of a set of properties and classes values that are picked from an ontology without any constraint on the chosen values. It corresponds to an ad hoc expressed preference.

3.2.3 Context based preferences

When a preference is defined according to a context, it is possible to specify the context in which a given preference is expressed. For example, the "*cheap*" preference is interpreted differently if we consider the country where the preference is interpreted. Therefore, a specific attribute is added to describe the context where a preference is interpreted.

3.2.4 Ontology Preference Link

Once ontological concepts, resources and preferences are defined, a link pref_link, obeying to the class diagram of UML is described. The role of this link is to attach a preference to a given ontological concept that may be a class or a property. This link is used to reach the preferences attached to ontological data. The availability of a manipulation language allowing to access meta-model data elements, concepts data elements and instances in required. In our study, the OntoQL language is used for this purpose.

4 Handling Preference in OntoDB

One can notice that current ontology models do not offer the possibility to handle preferences expressed on ontology concepts and instances. Since the OntoDB architecture coupled with the OntoQL language, allows to manipulate the ontology model, we suggest to feed the ontology model with the preference model. As a result, we will get, in the same universe, both the ontology model and the preference model. An explicit link between the property_or_class ontology model concept and the preference model concept is defined as well.

4.1 Extension of the OntoDB with Preferences

The extension of OntoDB to handle the preferences consists in describing a set of `CREATE` OntoQL clauses that create all the data elements of the UML data model of preferences defined in Table 1. Table A.1 shows the creation of the Preference root entity, Table A.2. shows the creation of the Preference_URI attached to a preference, and Table A.3. shows the creation of the Preference_Definition as inherited from the preference concept.

Table 1: OntoQL Clause Creation (A.1-A.2-A.3)

`CREATE ENTITY #Preference( #oid int, #URI REF( #Preference_URI));`	`CREATE ENTITY #Preference_URI( #code int, #name string, #classification string);`	`CREATE ENTITY #Preference_Definition UNDER #Preference( #oid int);`

All the other data elements of the UML class diagram of the preference model of Figure 2 are described in Appendix A.1.

4.2 Linking Ontologies and Preferences at the Ontology Model Level

Once the preferences are defined, they need to be linked and attached to the ontological concept they act on. This link, is established by the pref_link class of the UML class diagram of the preference model of Figure 2. The following OntoQL statements allows to create such a link in the OntoDB ontology based database. Practically, to encode composition, the preference link is absorbed by the ontology model concept property_or_class. It becomes an attribute of the property_or_class ontology model element. The following OntoQL statement is defined.

```
ALTER ENTITY #property_or_class ADD ATTRIBUTE #PREF_Link REF
(#Preference) ARRAY
```

4.3 Querying with Preferences

In order to handle the preferences in the OntoQL queries, a preference interpreter has been developed on top of the OntoQL engine. This is materialized by adding a `PREFERRING` clause in the OntoQL `SELECT` clause. An interpretation function is associated to each kind of preference available in the preference model. The form of the `SELECT` clause becomes as follows,

```
SELECT 'selection'
FROM 'tableReference'
PREFERRING 'preferenceIdentifer'
```

5 A CASE STUDY

Let us assume that we have a customer who wants a reservation for a Lodging Service to book an Hotel Room. The customer submits a request to the holiday booking system. It includes the information about the destination, travelling time and maximum budget. The system finds the most suitable hotel based on the information provided by the customers preferences (eg. *standard*, *cheap*).

5.1 Tourism Ontology Instantiation

The tourism ontology described by Figure 3-4 and the corresponding ontology instances are defined in the OntoDB database through two families of OntoQL statements. The first statement consists in creating the classes of the ontology. The next OntoQL statement describes the creation of the Hotel.

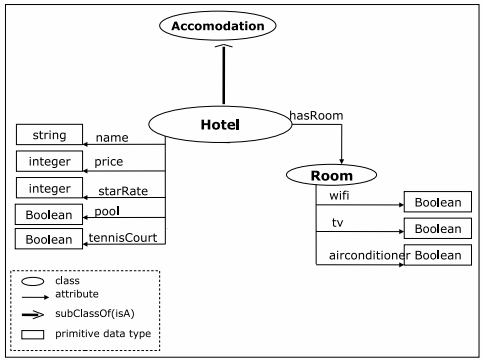

Figure 3: Tourism Ontology Concepts.

```
CREATE Class Hotel ( #id int, #name String, #starRate int,
#price int, #airCond boolean, #tv boolean, #wifi boolean,
#pool boolean, #jakuzi boolean);
```

In order to define ontology instances, the `INSERT INTO` OntoQL clause is used. For example, to define the Kyriad hotel corresponding to the #51 hotel instance, we define the following OntoQL statement.

```
INSERT INTO Hotel (id, name, starRating, price, airCond,
tv, wifi, pool, jakuzi, tennisCourt, casino)
VALUES (51, 'Kyriad ',3,55, yes,yes,yes, no,no,no,no);
```

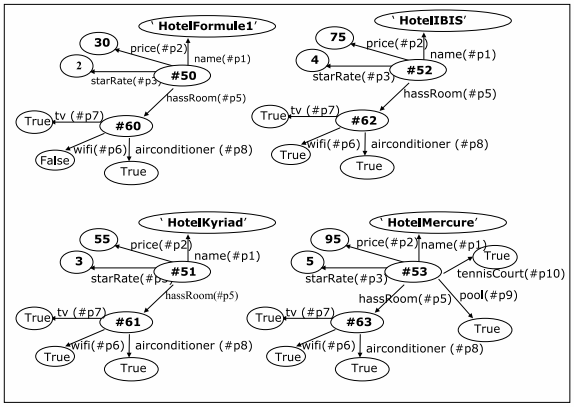

Figure 4: Tourism Ontology Instantiation.

5.2 Defining Preferences in the Tourism Domain

When addressing the tourism domain, qualities of tourism institutions are evaluated by national or international organizations. For our study, we restrict the definition of such quality features to hotels. For example, we are interested in defining the :

- star rating meaning that a hotel has *one*, *two*, *etc stars* in the starRating quality classification. The OntoQL statement that defines a *three star rating* is described by:

```
INSERT INTO #Numeric_Preference( #number_value, #code,
#name, #classification)
VALUES(3, 51,'standard','starRating');
```

- *very cheap*, *cheap*, *expensive*, *very expensive* that is attributed to a price in the cost quality classification. The following OntoQL statement describes the cheap quality as being any number belonging to the interval [45..60].

```
INSERT INTO Interval_Preference( #min_value, #max_value,
#code, #name, #classification)
VALUES(45, 60, 100,'cheap','cost');
```

The other defined preferences related to the case study are defined in Appendix A.2.

5.3 Attaching Preferences in the Tourism Domain to Domain Ontologies

When the preferences and the ontologies are defined, it is possible to link ontology classes to the preferences that are expressed on these classes. For this purpose, the manipulated ontology class is augmented by preferences thanks to the `ALTER` clause. A preference is attached to an instance of a hotel. For example, the preference *cheap* is attached to the Kyriad hotel using the following `UPDATE` OntoQL clause. Here, we put names for readability but identifiers are in fact used.

```
UPDATE Hotel set #pref_link=ARRAY['cheap',
'standard','full board'] where name='Kyriad'
```

Table 2: ALTER Class Hotel ADD Preference.

name	price	star Rating	PREFERENCE (cost,quality,promotion)
Kyriad	55	3	[cheap,standard,full board]

5.4 Querying Domain Ontologies with Preferences

Once the three previous steps are realized, it becomes possible to address queries to the ontology and its instances. Next, we give two examples of queries with quality.

1- The query that gives the 3 stars hotels is written as follows.

```
SELECT name, starRate FROM Hotel PREFERRING 'standard'
```

When the `PREFERRING` clause is interpreted, the query is automatically rewritten as follows,

```
SELECT name, starRate, preference From Hotel WHERE
starRate=3;
```

Table 3: Query Answer.

name	starRate	preference
Kyriad	3	standard

2- The query that asks for cheap hotels is written as follows,

```
SELECT name, price FROM hotel PREFERRING 'cheap';
```

When the `PREFERRING` clause is interpreted, the query is automatically rewritten as follows,

```
SELECT name, price, preference From Hotel WHERE price
BETWEEN 45 and 60;
```

Table 4: Query Answer.

name	price	preference
Kyriad	55	cheap

Notice that all the authorized construction of the OntoQL language can be used in building the query. The last clause is the `PREFERRING` clause. It is used for rewriting the queries into standard OntoQL queries.

6 CONCLUSION

This paper has presented an extension of a database architecture in order to handle preference modeling and querying with preferences not at the database logical model but at the semantic level offered by the ontology. This extension requires:

- the explicit representation of the ontology in the database. As a consequence, we have been able to attach the preferences to the classes and to the properties of the ontology and not to the columns of the logical model of the database where instances or data are stored;
- the possibility to access and to manipulate the ontology model through the access and manipulation to the meta-model and
- finally, the availability of an exploitation language allowing to manipulating both the instances, their classes and the meta-model in the case of ontologies.

These requirements are fulfilled by the OntoDB ontology based database and by the OntoQL exploitation language. The extension of the ontology model with the preference model permitted to attach various types of preferences to classes and/or properties of the ontology. As a consequence, we have been able to describe semantic queries that handle preferences expressed at the semantic level, and thus abstracting from the logical model.

We believe that the possibility to access the meta-model level well adapted to define model extensions that preserve upward compatibility with the extended model. This work has opened several new directions and perspectives. Indeed, such extensions are possible for other different domain characterizations like security, user profiles or model annotations.

REFERENCES

Agrawal, R. and Wimmers, E. L. (2000). A framework for expressing and combining preferences. In *SIGMOD Conference*, pages 297–306.

Alexaki, S., Christophides, V., Karvounarakis, G., Plexousakis, D., and Tolle, K. (2001). Managing Voluminous RDF Description Bases. In *Proceedings of the 2nd International Workshop on the Semantic Web*, pages 1–13.

Broekstra, J., Kampman, A., and van Harmelen, F. (2002). Sesame: A Generic Architecture for Storing and Querying RDF and RDF Schema. In *Proceedings of the 1st International Semantic Web Conference (ISWC'02)*, pages 54–68.

Chomicki, J. (2003). Preference formulas in relations queries. *ACM Transactions on Database Systems*, 28:1–39.

Chong, E. I., Das, S., Eadon, G., and Srinivasan, J. (2005). An Efficient SQL-based RDF Querying Scheme. In *Proceedings of the 31st international conference on Very Large Data Bases (VLDB'05)*, pages 1216–1227.

Dehainsala, H., Pierra, G., and Bellatreche, L. (2007). OntoDB: An Ontology-Based Database for Data Intensive Applications. In *Proceedings of the 12th International Conference on Database Systems for Advanced Applications (DASFAA'07)*, pages 497–508.

Jean, S., Aït-Ameur, Y., and Pierra, G. (2006). Querying Ontology Based Database Using OntoQL. In *Proceedings of On the Move to Meaningful Internet Systems 2006:(ODBASE'06)*, volume 4275 of *Lecture Notes in Computer Science*. Springer.

Jean, S., Pierra, G., and Ait-ameur, Y. (2005). Ontoql: an exploitation language for obdbs. In *VLDB Ph.D. Workshop*.

Kieling, W. (2002). Foundations of preferences in database systems. In Mars, N. J. I., editor, *Knowledge and Data Engineering*, pages 311–322. IOS Press, Amsterdam.

Kieling, W. and Kostler, G. (2000). Preference sql -design, implementation, experience. In Mars, N. J. I., editor, *Knowledge and Data Engineering*, pages 778 – 789. IOS Press, Amsterdam.

Koutrika, G. and Ioannidis, Y. E. (2004). Personalization of queries in database systems. In *ICDE*, pages 597–608.

P. Gursk, T.Horvth, J. J. and Vanekov, V. (2008). User preference web search – experiments with a system connecting web and user. In Mars, N. J. I., editor, *To appear in the Computing and Informatics Journal*, pages 25–32. IOS Press, Amsterdam.

Petrini, J. and Risch, T. (2007). SWARD: Semantic Web Abridged Relational Databases. In *Proceedings of the 18th International Conference on Database and Expert Systems Applications (DEXA'07)*, pages 455–459.

Pierra, G., Dehainsala, H., Aït-Ameur, Y., and Bellatreche, L. (2005). Base de Données à Base Ontologique : principes et mise en œuvre. *Ingénierie des Systèmes d'Information*, 10(2):91–115.

Siberski, W., Pan, J. Z., and Thaden, U. (2006). Querying the semantic web with preferences. In *In Proceedings of the 5th International Semantic Web Conference (ISWC*, pages 612–624.

SPARQL Query Language for RDF (January 2008). SPARQL. http://www.w3.org/TR/rdf-sparql-query/.

Toninelli, A., Corradi, A., and Montanari, R. (2008). Semantic-based discovery to support mobile context-aware service access. *Computer Communications*, 31(5):935–949.

Viappiani, P., Faltings, B., and Pu, P. (2006). Preference-based search using example-critiquing with suggestions. *Journal of Artificial Intelligence Research*, 27:2006.

APPENDIX

Appendix A: OntoQL Statements for Preferences

Table 5: Preference Model Creation (Appendix A.1)

```
CREATE ENTITY #Preference_URI(
#code int, #name string, #classification string);
```

```
CREATE ENTITY #Preference(
#oid int, #URI REF(#Preference_URI));
```

```
CREATE ENTITY #Preference_Definition
UNDER #Preference(#oid int);
```

```
CREATE ENTITY #Interpreted_Pref
UNDER #Preference_Definition(#oid int);
```

```
CREATE ENTITY #Enumerated_Pref
UNDER #Interpreted_Preference(
#pref_value REF #property_or_class_instance ARRAY);
```

```
CREATE ENTITY #Numeric_Pref
UNDER #Interpreted_Preference(#number int);
```

```
CREATE ENTITY #Boolean_Pref
UNDER #Interpreted_Preference( #properties REF(#property) ARRAY);
```

```
CREATE ENTITY #Interval_Pref UNDER #Interpreted_Preference(
#min_value int,#max_value int);
```

```
CREATE ENTITY #Fuzzy_Pref UNDER #Interpreted_Preference(
#prob_value float, #property REF (#property) ARRAY);
```

```
CREATE ENTITY #UnInterpreted_Pref UNDER #Preference_Definition(
#pref_value REF ( #property_or_class_instance)ARRAY);
```

```
CREATE ENTITY #Context_Pref_Definition UNDER #Preference(
#context REF (#property_or_class_instance),
#preference REF (#Preference_Definition), #context_name String);
```

Table 6: Preference Insertion (Appendix A.2)

```
INSERT INTO #Numeric_Pref( #number_value,#code,#name,
#classification) VALUES(2, 50,'low','starRating');
```

```
INSERT INTO #Numeric_Pref( #number_value,#code,#name,
#classification) VALUES(3, 51,'standard','starRating');
```

```
INSERT INTO #Numeric_Pref( #number_value,#code,#name,
#classification) VALUES(4, 52,'middle','starRating');
```

```
INSERT INTO #Numeric_Pref( #number_value,#code,#name,
#classification) VALUES(5, 53,'lux','starRating');
```

```
INSERT INTO Interval_Pref( #min_value,#max_value,#code,#name,
#classification) VALUES(20, 45, 99,'very_cheap','cost');
```

```
INSERT INTO Interval_Pref( #min_value,#max_value,#code,#name,
#classification) VALUES(45, 60, 100,'cheap','cost');
```

```
INSERT INTO Interval_Pref( #min_value,#max_value,#code,#name,
#classification) VALUES(60, 90, 101,'expensive','cost');
```

```
INSERT INTO Interval_Pref( #min_value,#max_value,#code,#name,
#classification) VALUES(90, 100, 102,'very_expensive','cost');
```