Querying Relational Data with Semantic Domains using SQL in Oracle RDBMS

Souripriya Das, Eugene Inseok Chong, George Eadon, Jagannathan Srinivasan
Oracle Corporation
One Oracle Drive, Nashua, NH 03062, USA

Abstract
Ontologies as well as controlled vocabularies, and domain-specific terminologies are increasingly being developed to capture semantics for life science applications. A requirement for database systems is to allow the domain of a table column to be such hierarchical terminologies and to provide query support. The paper addresses this requirement by introducing the notion of a semantic domain for relational data and providing the capability of querying over such relational data. Specifically, 1) OWL ontologies are used to model semantic domains, 2) A set of semantic-match SQL operators, namely, ONT_RELATED, ONT_DISTANCE, and ONT_PATH are introduced, to query over columns with respect to a semantic domain, 3) A new indexing scheme ONT_INDEXTYPE is introduced to speed up such queries, and 4) ONT_EXPAND operator is provided to query ontologies directly. Our approach enables users to query relational data with respect to a semantic domain directly using SQL operators, thereby opening up possibilities of combining with other operations such as joins as well as making semantically rich life science applications easy to develop and efficient. This paper describes the key concepts of querying relational data with semantic domains, discusses the set of SQL semantic match operators, and illustrates its usage via a case study involving National Cancer Institute’s Ontology.

1 Introduction
Ontologies as well as controlled vocabularies, and domain-specific terminologies, are increasingly being developed for life science domains. They range from full-fledged ontologies such as Gene Ontology [1], and BioPAX [2], to controlled vocabularies such as National Library of Medicine’s SNOMED Clinical Terms [3]. A requirement for database systems is to allow the domain of a table column to be such hierarchical terminologies and the ability to query them [4].

To illustrate this requirement, consider a simple restaurant guide application, which recommends restaurants to a user based on her/his preferences. Consider a table `served_food` with `restaurant_id` and `cuisine` columns, which describes the types of cuisines served at restaurants. The `cuisine` column’s domain is a hierarchical terminology, which can be best represented as a cuisine ontology shown in Figure 1.

![Cuisine Ontology]

---

**Table served_food**

<table>
<thead>
<tr>
<th>restaurant_id</th>
<th>cuisine</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>American</td>
</tr>
<tr>
<td>2</td>
<td>Mexican</td>
</tr>
<tr>
<td>14</td>
<td>Portuguese</td>
</tr>
</tbody>
</table>

*Figure 1: a) A Cuisine Ontology: Each node represents an Individual and each edge represents a transitive ObjectProperty ‘IS_A’. and b) The served_food Table*

For querying on such columns whose domain is hierarchical terminology, the traditional syntactic matching via the ‘=’ operator is not very useful. For example consider the following query posed by a user interested in restaurants that serve ‘Latin American’ Cuisine:

```
SELECT * FROM served_food WHERE cuisine = 'Latin American';
```

This query generates no rows since none of Cuisine values in the table will match ‘Latin American’. In contrast, the user can get more meaningful results by performing semantic matching that consults the semantic
domain modeled by cuisine ontology shown in Figure 1 for computing the results. Specifically, a user can issue the following query:

```sql
SELECT * FROM served_food
WHERE ONT_RELATED(cuisine, 'IS_A', 'Latin American', 'Cuisine_ontology')=1;
```

Here the ONT_RELATED operator determines if the two input terms are related by the input relationship type argument by consulting the specified ontology. If they are related, then the operator will return 1, otherwise 0.

The query identifies rows containing cuisines that are related to ‘Latin American’ based on ‘IS_A’ relationship. The query will generate restaurants 2 and 14 since ‘Mexican’ and ‘Portuguese’ are related to ‘Latin American’ cuisine. Thus, one can incorporate semantics of the particular knowledge domain in SQL queries by introducing ontology-based semantic matching.

 Optionally, a user may want to get a measure for the rows filtered by ONT_RELATED operator. This can be achieved by using ONT_DISTANCE ancillary operator. The ONT_DISTANCE operator gives a measure of how closely the terms are related by measuring the distance between the two terms. Continuing with the example, one can get the result sorted on distance measure as follows:

```sql
SELECT * FROM served_food
WHERE ONT_RELATED (cuisine, 'IS_A', 'Latin American', 'Cuisine_ontology', 123) = 1
ORDER BY ONT_DISTANCE (123);
```

Similarly, another ancillary operator ONT_PATH would be useful, which computes path information between the two terms.

Providing ontology-based semantic matching capability and querying as part of SQL will greatly facilitate developing semantically rich life science applications by allowing applications to consult ontologies that capture domain semantics. Also, applications that have to work with domain-specific knowledge repositories (such as BioInformatics, and Healthcare Applications) can take advantage of this capability. Thus, the following functionality is needed:

- Use OWL ontologies to model semantic domain for relational data,
- Provide a set of semantic-match SQL operators, namely, ONT_RELATED, ONT_DISTANCE, and ONT_PATH to query columns with respect to a semantic domain, and
- Provide an indexing scheme ONT_INDEXTYPE to speed up such queries.

An earlier paper [5] introduced this functionality and described its implementation in Oracle RDBMS. This paper provides a brief overview of the functionality and focuses on its use via a case study involving National Cancer Institute’s (NCI) Ontology.

2. Functionality Overview

For representing ontologies Web Ontology Language (OWL) [6] is used. Users can create and load OWL Lite and DL ontologies into the database via a system-defined API. For querying purposes the following functionality is provided (see Figure 2):

```
ONT_EXPAND Operator
ONT_RELATED Operator
(with ONT_PATH & ONT_DISTANCE ancillary operators)
```

- An RDBMS schema, consisting of several system-defined tables, is created for storing information extracted from the ontologies.
- Two operators are provided for querying purposes. The ONT_EXPAND operator can be used to query the ontology independently, whereas ONT_RELATED operator can be used to perform queries on a user table holding ontology terms.

---

1 This argument identifies the filtering operator expression (ONT_RELATED) that computes this ancillary value [11].
• Optionally, a user can use two (ancillary) operators, ONT\_DISTANCE and ONT\_PATH, in queries involving the ONT\_RELATED operator to get additional measures (distance and path) for the filtered rows.
• Optionally, user can create index on the column holding ontology terms using a new ONT\_INDEXTYPE indexing scheme to speed up queries involving above-mentioned operators (for details see [5]).

2.1 ONT\_RELATED Operator

This operator models the basic semantic matching operation. It determines if the two input terms are related with respect to the specified RelType relationship argument within an ontology. If they are related it returns 1, otherwise it returns 0.

\[
\text{ONT\_RELATED} \ (\text{Term1, RelType, Term2, OntologyName}) \ \text{RETURNS} \ \text{INTEGER};
\]

The RelType can specify a single ObjectProperty (for example, ‘IS\_A’, ‘EQV’, etc.) or it can specify a mixed path via OR operators (for example, ‘IS\_A OR EQV’). Note that both Term1 and Term2 need to be simple terms. In addition, user can use the SQL INTERSECT, UNION, and MINUS operators to combine results from several queries involving ONT\_RELATED operator.

2.2 ONT\_EXPAND Operator

This operator is introduced to query an ontology directly (that is, without referencing the user tables).

\[
\text{CREATE TYPE ONT\_TermRelType AS OBJECT (}
\text{Term1Name VARCHAR(32), PropertyName VARCHAR(32), Term2Name VARCHAR(32),}
\text{TermDistance NUMBER, TermPath VARCHAR(2000))};
\]

\[
\text{CREATE TYPE ONT\_TermRelTableType AS TABLE OF ONT\_TermRelType;}
\]

\[
\text{ONT\_EXPAND} \ (\text{Term1, RelType, Term2, OntologyName}) \ \text{RETURNS} \ \text{ONT\_TermRelTableType;}
\]

Term1 is typically specified as NULL, which indicates any term, whereas non-NULL values for RelType and Term2 are specified as input. The operator returns all the matching \(<\text{Term1, RelType, Term2}>\) tuples in the closure taking into account the characteristics (transitivity and symmetry) of the specified RelType. In addition, it computes the relationship measures in terms of distance (TermDistance) and path (TermPath). Similar to ONT\_RELATED operator, the RelType can specify either a simple relationship or a mixed path using OR operator. ONT\_EXPAND invocation may specify input values for any one or more of the three parameters. Wildcard for these three parameters can be specified by using NULLs.

2.3 ONT\_DISTANCE and ONT\_PATH Ancillary Operators

These operators compute the distance and path measures respectively for the rows filtered using ONT\_RELATED operator.

\[
\text{ONT\_DISTANCE} \ (\text{NUMBER}) \ \text{RETURNS} \ \text{NUMBER};
\]

\[
\text{ONT\_PATH} \ (\text{NUMBER}) \ \text{RETURNS} \ \text{VARCHAR};
\]

A single resulting row can be related in more than one way with the input term. For such cases, the above operators return the optimal measure, namely smallest distance or shortest path. For computing all the matches, the following two operators are provided:

\[
\text{ONT\_DISTANCE\_ALL} \ (\text{NUMBER}) \ \text{RETURNS} \ \text{TABLE OF NUMBER};
\]

\[
\text{ONT\_PATH\_ALL} \ (\text{NUMBER}) \ \text{RETURNS} \ \text{TABLE OF VARCHAR};
\]

3 NCI Cancer Ontology: A Case Study

3.1 Ontology Overview

NCI Cancer Ontology captures detailed semantic relationship among genes, diseases, drugs and chemicals, anatomy, organisms, and proteins. The ontology has been specified using OWL Lite. Version 4.06i contains taxonomies of approximately 36,000 concepts represented as OWL classes, which are related to one another via subclass relationships. It defines about 90 object properties that can be used to specify binary relationships between individuals belonging to the different classes. The class descriptions use about 35,000 local restrictions defined on those object properties. In addition, the ontology uses about 40 annotation properties to describe class and property
attributes. Altogether, the ontology contains close to 700,000 triples that represent the concepts and their complex relationships in this domain. Figure 3 depicts a portion of the ontology’s class hierarchy.

![Figure 3: Portion of NCI Cancer Ontology Class Heirarchy](image)

### 3.2 Patients and Specialists Tables

Two user tables, Patients and Specialists, are defined as shown below:

Patients(id NUMBER, diagnosis VARCHAR(200), notes TEXT, ...)
Specialists(name VARCHAR(50), specialization VARCHAR(200), ...)

Semantic domain for two columns, the Diagnosis column in PATIENTS table and the Specialization column in SPECIALISTS table, is part of the Cancer Ontology. Thus, data stored in these columns must always come from existing terms in that domain. This is similar to how a referential constraint is used. The ontology, however, is not just a repository of terms that may be used, but also the interrelationships among those terms. Furthermore, the semantic domain for a table column need not be restricted to a single ontology. Multiple ontologies may be appropriate for the same domain in that they capture the same set of terms that are stored in the columns, but define different interrelationships among those terms.

### 3.3 Queries

Typically database queries involve syntactic matching. In this section, we illustrate use of the operators defined in the earlier sections in associating semantics with this matching functionality.

**Simple Single Table Semantic Filtering**

Consider the query: Find all patients whose diagnosis is of the type ‘Immune_System_Disorder’. Clearly, a typical database query involving syntactic match may not return all the relevant rows. However, the query with ONT_RELATED operator as shown below will return all the relevant results:

```sql
SELECT id FROM Patients
WHERE ONT_RELATED(diagnosis, 'rdfs:subClassOf', 'Immune_System_Disorder', 'Cancer_ontology') = 1;
```

One can extend the above query to specify some constraints on the paths between ‘Immune_System_Disorder’ and each qualifying value in the diagnosis column. Constraints may address the length and/or content of the path. This can be implemented using the ancillary operators ONT_DISTANCE and ONT_PATH.

```sql
SELECT id FROM Patients
WHERE ONT_RELATED(diagnosis, 'rdfs:subClassOf', 'Immune_System_Disorder', 'Cancer_ontology', 123) = 1 AND
   pathContains('ONT_PATH_ALL (123), 'Rheumatoid_Arthritis') = 0;
```

Another use of ancillary operators may involve grouping on the basis of path characteristics (length and/or content). Consider the extension to the above query: *group by the subclasses of ‘Immune_System_Disorder’. This can be achieved by using*

```sql
SELECT extractPathTerm(ONT_PATH(123), 2) disease_type, COUNT(*) FROM Patients
WHERE ONT_RELATED(diagnosis, 'rdfs:subClassOf', 'Immune_System_Disorder', 'Cancer_ontology', 123) = 1
```

---

2 pathContains() and extractPathTerm() are system provided utility functions.
GROUP BY extractPathTerm(ONT_PATH(123), 2);

Note that extractPathTerm(path, i) extracts the ith term present in the path argument.

Semantic Join

Consider the equi-join query: For each patient, based upon the diagnosis, find the specialist doctor(s) who would be appropriate for referral. A typical database join query would do a syntactic match on the diagnosis and specialization columns. However, that may not be able to find a specialist for all patients. For example, if no specialist is available for ‘AIDS’, we may want to send the patient to a specialist whose specialization is in ‘T-Cell Immunodeficiency’. This semantic matching requirement for the join condition can be implemented using the ONT_RELATED operator as follows:

```
SELECT P.id, S.name FROM Patients P, Specialists S
WHERE ONT_RELATED(P.diagnosis, 'rdfs:subClassOf', S.specialization,
   'Cancer_ontology') = 1;
```

One can extend the above query using ONT_DISTANCE operator to return the list of specialists sorted according to the distance between the diagnosis and a (semantic) matching specialization. For example, if an ‘AIDS’ specialist is available and also an ‘T-Cell Immunodeficiency’ specialist is available, the ‘AIDS’ specialist will appear first in the list for a patient diagnosed with ‘AIDS’ in the following query:

```
SELECT P.id, S.name FROM Patients P, Specialists S
WHERE ONT_RELATED(P.diagnosis, 'rdfs:subClassOf', S.specialization,
   'Cancer_ontology', 123) = 1
ORDER BY P.id, ONT_DISTANCE(123);
```

Querying Ontologies

The ontology can be directly queried using ONT_EXPAND operator. For example, user can issue the following query to get all diagnoses of the type ‘Immune_System_Disorder’ present in the NCI Cancer Ontology:

```
SELECT * FROM TABLE(ONT_EXPAND(NULL, 'rdfs:subClassOf', 'Immune_System_Disorder',
   'Cancer_ontology'));
```

4 Conclusions and Future Work

In this paper, we addressed the requirement of querying a table column with respect to a semantic domain modeled using OWL ontology. The set of semantic match operators, namely, ONT_RELATED, ONT_DISTANCE, and ONT_PATH, are sufficiently expressive in querying relational data with semantic domains. Also, ONT_EXPAND can be used to query ontologies directly. We illustrated the use of these operators using NCI Cancer Ontology. We expect that this functionality will facilitate building semantically rich life science applications. In future, we plan to support incrementally loading of ontologies, merging of ontologies, and provide further optimizations for queries involving semantic match operators.

References