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 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 concept 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 such columns [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.

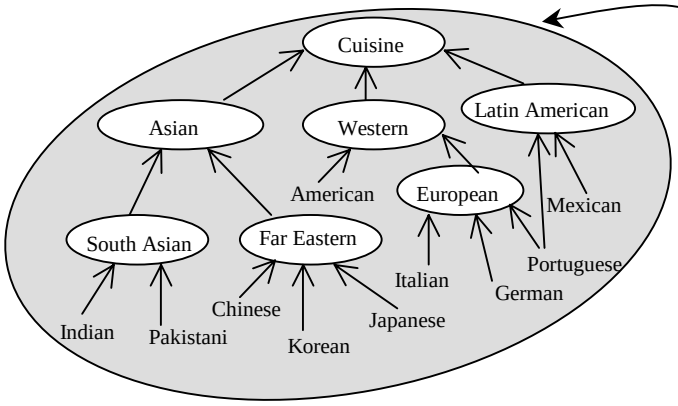

Table: served_food

restaurant_id	cuisine
1	American
2	Mexican
2	American
14	Portuguese

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 does not generate any 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 for computing the results. Specifically, a user can issue the following query:

```
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:

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

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. This requirement is addressed with the following functionality in Oracle RDBMS:

- 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,
- Provide an indexing scheme ONT_INDEXTYPE to speed up such queries, and
- Provide ONT_EXPAND operator for directly querying ontologies.

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 [7].

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):

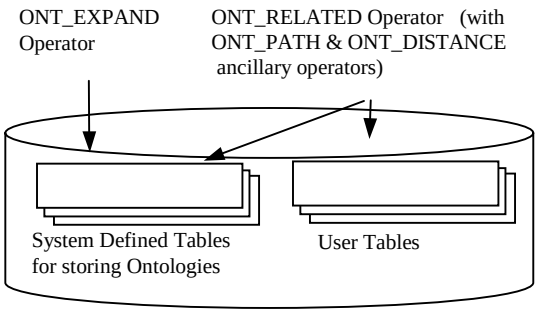

Figure 2: Query Functionality

- 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.
- Optionally, a user can use (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.

[1] This argument identifies the filtering operator expression (ONT_RELATED) that computes this ancillary value [8].

- 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.

```
ONT_RELATED (Term1, RelType, Term2, OntologyName) RETURNS 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).

```
CREATE TYPE ONT_TermRelType AS OBJECT (
 Term1Name VARCHAR(32), PropertyName VARCHAR(32), Term2Name VARCHAR(32),
 TermDistance NUMBER,  TermPath VARCHAR(2000)
);
CREATE TYPE ONT_TermRelTableType AS TABLE OF ONT_TermRelType;
ONT_EXPAND (Term1, RelType, Term2, OntologyName) RETURNS 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 `<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 wildcard by using NULL for any of the three parameters.

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:

```
ONT_DISTANCE (NUMBER) RETURNS NUMBER;
ONT_PATH (NUMBER) RETURNS 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:

```
ONT_DISTANCE_ALL (NUMBER) RETURNS TABLE OF NUMBER;
ONT_PATH_ALL (NUMBER) RETURNS TABLE OF VARCHAR;
```

3 NCI Cancer Ontology: A Case Study

To illustrate the functionality described above, the NCI Cancer Ontology [7] is used.

3.1 Ontology Overview

NCI Cancer Ontology captures detailed semantic relationship among genes, diseases, drugs and chemicals, anatomy, organisms, and proteins. The ontology 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.

3.2 Patients and Specialists Tables

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

```
Patients(id NUMBER, diagnosis VARCHAR(200), notes CLOB, ...);
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.

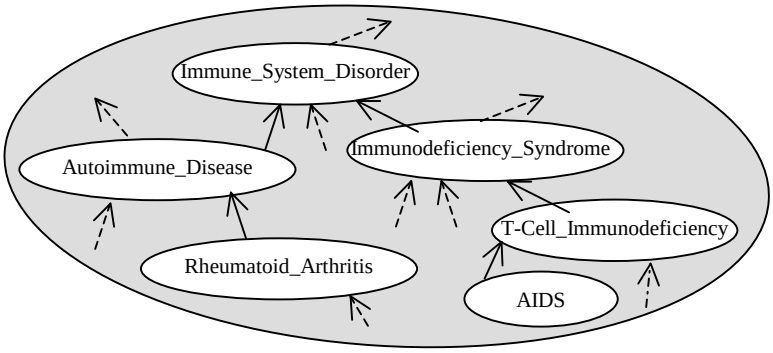

Figure 3: Portions of NCI Cancer Ontology

3.2 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:

```
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.

For example, the following query further restricts the result so that *each qualifying value in the diagnosis is related to 'Immune_System_Disorder' via 'Rheumatoid_Arthritis':*

```
SELECT id FROM Patients
WHERE ONT_RELATED(diagnosis, 'rdfs:subClassOf', 'Immune_System_Disorder',
                   'Cancer_ontology', 123) = 1 AND
      pathContains[2](ONT_PATH_ALL (123), 'Rheumatoid_Arthritis') = 0;
```

Also, multiple ONT_RELATED predicates results can be combined. For example, the following query *finds diagnosis related to 'Diseases_and_Disorder' via 'Immune_System_Disorder' in less than 5 hops:*

```
SELECT id FROM Patients
WHERE ONT_RELATED(diagnosis, 'rdfs:subClassOf',
                  'Immune_System_Disorder', 'Cancer_ontology', 1) = 1 AND
      ONT_RELATED('Immune_System_Disorder','rdfs:subClassOf',
                  'Diseases_and_Disorders', 'Cancer_ontology', 2) = 1 AND
      ONT_DISTANCE(1) + ONT_DISTANCE(2) < 5;
```

Another use of ancillary operators may involve grouping on the basis of path characteristics (length and/or content). For example, the following query *groups by the subclasses of 'Immune_System_Disorder'*:

```
SELECT extractPathTerm(ONT_PATH(123), 2) disease_type, COUNT(*) FROM Patients
```

[2] pathContains() and extractPathTerm() are system provided utility functions.

```
WHERE ONT_RELATED(diagnosis, 'rdfs:subClassOf', 'Immune_System_Disorder',
                  'Cancer_ontology', 123) = 1
GROUP BY extractPathTerm(ONT_PATH(123), 2);
```

Note that `extractPathTerm(path, i)` extracts the `i`th 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 a `'T-Cell_Immunodeficiency'` specialist is available then the `'AIDS'` specialist will appear earlier 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'));
```

In addition, one can query metadata. For example, the following query may be posed to get *Annotation Properties*:

```
SELECT Term1Name FROM TABLE(ONT_EXPAND(NULL, 'rdf:type', 'owl:AnnotationProperty',
                                       '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 incremental loading of ontologies, merging of ontologies, and provide further optimizations for queries involving semantic match operators.

References

[1] Gene Ontology Consortium, http://www.geneontology.org.

[2] BioPAX, http://www.biopax.org.

[3] SNOMED Clinical Terms, http://www.nlm.nih.gov/research/umls/Snomed/snomed_main.html.

[4] T. Topaloglou, et al., "Biological Data Management: Research, Practice, and Opportunities," In *Proceedings of the 30th Int. Conf. on Very Large Data Bases*, pp.1233-1236, Aug. 2004.

[5] S. Das, E. I. Chong, G. Eadon, and J. Srinivasan, "Supporting Ontology-based Semantic Matching in RDBMS," In *Proceedings of the 30th Int. Conf. on Very Large Data Bases*, pp.1054-1065, Aug. 2004.

[6] OWL Web Ontology Language Reference, http://www.w3.org/TR/owl-ref.

[7] National Cancer Institute Thesaurus and Ontology, http://www.mindswap.org/2003/CancerOntology.

[8] R. Murthy, S. Sundara, N. Agarwal, Y. Hu, T. Chorma, J. Srinivasan, "Supporting Ancillary Values from User Defined Functions in Oracle", In *Proceedings of the 19th International Conference on Data Engineering,* pp. 151-162, 2003.