

Copyright © 2006 IEEE. Reprinted from (Proceedings of the 22<sup>nd</sup> ICDE Conference).

This material is posted here with permission of the IEEE. Such permission of the IEEE does not in any way imply IEEE endorsement of any of the Oracle USA, New England Development Center's products or services. Internal or personal use of this material is permitted. However, permission to reprint/republish this material for advertising or promotional purposes or for creating new collective works for resale or redistribution must be obtained from the IEEE by writing to [pubs-permissions@ieee.org](mailto:pubs-permissions@ieee.org).

By choosing to view this document, you agree to all provisions of the copyright laws protecting it.

# Supporting Keyword Columns with Ontology-based Referential Constraints in DBMS

Eugene Inseok Chong

Souripriya Das

George Eadon

Jagannathan Srinivasan

Oracle

One Oracle Drive, Nashua, NH 03062, USA

## Abstract

*Keywords are typically used to qualify rows in a table. However, the fact that a keyword denotes a concept, which belongs to a specific knowledge domain, is not semantically enforced in current database systems. This paper proposes defining ontology based referential constraint for such keyword columns. A query on ontology, specified as part of the referential constraint, is used to identify the domain for the keyword column. Furthermore, since ontology may evolve causing change to the domain of the keyword column, the paper proposes use of ontology based transformation functions to either automatically evolve or to recommend refinements for the values in the keyword column. Also, queries on a keyword column can perform semantic match, that is, match a keyword to related terms based on the associated ontology. Thus, the proposed approach of semantically connecting keyword columns to ontologies 1) enhances semantic data integrity, 2) facilitates evolution of keyword columns with the referenced ontology, and 3) enables semantic match queries on keyword columns.*

## 1. Introduction

Keywords are used to characterize table rows. For example, a table containing medical articles can have a keyword column that holds the set of keywords associated with the article. Column containing keywords (referred to as *keyword columns* hereafter) enables capturing information in a succinct manner and facilitates keyword based searching.

Although keyword columns can easily be supported in a DBMS by using a VARCHAR or CHAR column, the representation fails to capture the semantic link of keyword to the corresponding knowledge domain. Thus, data integrity is compromised leading to storage of possibly erroneous data, which could result in incorrect answers for a subsequent keyword based search.

The paper addresses this problem by introducing ontology-based referential constraints for keyword columns. That is, as part of table creation, or in a subsequent constraint definition statement, user can associate a referential constraint with keyword columns.

A query on ontology, specified as part of the referential constraint, is used to identify the domain for the keyword column. Furthermore, since ontology may evolve causing change to the domain for the keyword column, the paper proposes use of ontology-based transformation functions to automatically evolve the values in the keyword column or to recommend possible replacement values.

The three key benefits of this approach are as follows:

- *Enhances Data Integrity for Keyword Columns:* The constraint ensures that only valid keywords (those belonging to the subset of terms identified by the ontology query specified in the referential constraint) are stored in the column.
- *Facilitates Evolution of Keyword Column with Ontology:* As the ontology changes, automatic transformation can be done to keyword column entries so that they contain most suitable term from the ontology. For cases where the suitable term cannot be unambiguously determined, the possible terms can be suggested as recommendations.
- *Enables Semantic Search on Keyword Columns:* In addition to the typical equality based match of keywords, semantic match can be supported on keyword column. This is possible because the values used in a keyword column, constrained via an ontology-based referential constraint, constitute a subset of terms in the referenced ontology and so matching can be based on knowledge of relationships between the terms in the referenced ontology.

The referenced ontologies could range from simple controlled vocabularies (such as SNOMED Clinical Terms [2] and MeSH Medical Subject Heading Thesaurus [3]) to full-fledged knowledge representation for a domain (such as BioPAX Biological Pathways Ontology [3], and National Cancer Institute's Thesaurus and Ontology [4]).

**An example:** Consider a cancer patient data table with a 'diagnosis' keyword column (Figure 1). It would be desirable to constrain this keyword column to terms from National Cancer Institute's (NCI) Cancer Ontology.

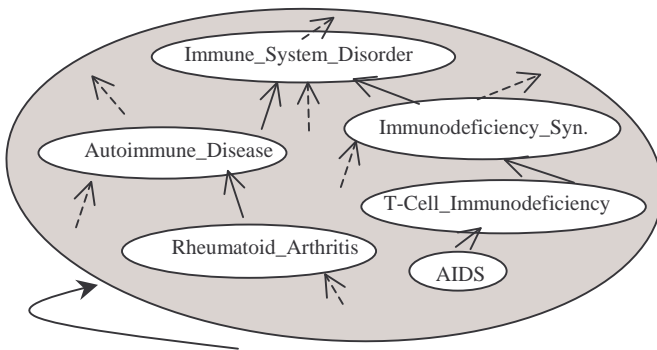
An ontology-based referential constraint can be specified (using ONT\_EXPAND table function introduced in [10] for querying ontologies) as follows:

```

ALTER TABLE patients
ADD CONSTRAINT diag_constraint
FOREIGN KEY (diagnosis)
REFERENCES (SELECT TermName
            FROM TABLE (ONT_EXPAND (
                NULL,
                'rdfs:subClassOf',
                'Immune_System_Disorder',
                'Cancer Ontology')));

```

The table function ONT\_EXPAND returns all terms (indicated by using the NULL value as the first parameter) that are subclasses of the term 'Immune\_System\_Disorder' defined in the Cancer Ontology. That is, the patient data is constrained to keywords denoting subclasses of 'Immune\_System\_Disorder' in the Cancer Ontology.



patient_id	Diagnosis
1721	Rheumatoid_Arthritis
3412	Immunodeficiency_Syndrome
2331	AIDS

Figure 1: Constraining Diagnosis Keyword Column

If transformation functions are specified as part of referential constraint, they could allow evolution of the diagnosis column with the changes to Cancer Ontology. For example, a desired action on deletion of 'Rheumatoid\_Arthritis' from the ontology could be to automatically transform the rows with that keyword (here row corresponding to patient\_id 1721) to its immediate superclass, namely, 'AutoImmune\_Disease'.

Also, the keyword-based search query can take the ontology into consideration to perform semantic match. For example,

```

SELECT p.patient_id FROM patients p
WHERE ONT_RELATED (p.diagnosis,
                  'IS_A',
                  'Immune_System_Disorder',
                  'Cancer Ontology') = 1;

```

Note that the ONT\_RELATED operator introduced in [10] allows matching of the terms based on the specified relationship 'IS\_A' in the example above. That is, the

above query will return all patients who are diagnosed with 'Immune\_System\_Disorder' or any of its subclasses by consulting the referenced ontology.

Ontology-based referential constraint can be supported as an extension to the traditional referential constraint mechanism in a DBMS. Below we identify the key components as well as outline our approach:

- *Ontology Storage and Querying:* This is a new component that needs to be added to the DBMS. We reuse the implementation described in [10], which allows storage of OWL [7] ontologies in DBMS and provides a SQL table function ONT\_EXPAND to query the ontology.
- *Ontology-based Constraint Definition:* The constraint definition is supported in a manner similar to the current referential constraint mechanism that links a child table to a parent table. The key difference is that the possible set of values is identified by the query on the ontology.
- *On Ontology Update Semantics:* Upon deletion of a keyword from the domain, an ON DELETE transform function specified as part of constraint definition, allows for automatic transformation of the referenced keyword to a closest related keyword (whenever possible) for all matching entries in the keyword column table. Upon insertion of a keyword into the domain, an ON INSERT refinement function allows for generating recommendations possible replacement with more suitable values.
- *Enforcing Ontology-based Constraint:* Updates on the keyword column table only need to ensure that the keywords inserted or updated belong to the set of ontology terms returned by the specified ontology query in the referential constraint. We propose use of a materialized view based approach for tracking changes and taking appropriate actions. For updates on ontologies, there are two possibilities i) OLAP style update, and ii) OLTP style. For OLAP style update, we support disabling before the updates and enabling the constraint after the operation. The erroneous keywords are inserted into an EXCEPTIONS table when the constraint is enabled. For OLTP style update, that is, when ontology is incrementally updated, we propose using triggers on the materialized view so that appropriate actions are taken. Details are discussed in Section 4.

In the paper, the solution is described for a column containing a single keyword. However, in practice a column can contain a set of keywords. Such keyword columns can be handled by using a collection type column (see Section 5 for details).

The key contributions of the paper are:

- *Ontology based referential constraints:* To the best

of our knowledge, this is the first proposal that enhances data integrity for keyword columns via ontology-based referential constraint.

- *Automatic Evolution of Keyword Columns with Ontology*: Again, the use of ontology opens up the possibility of evolving keyword column with the changes to the ontology.
- *An efficient Constraint Enforcement Scheme*: In the event of changes to both referenced ontologies and keyword column tables, the paper proposes an efficient materialized view-based scheme for enforcing the constraint.

**Related work:** Our notion of ontology-based referential constraints can be viewed as an extension of traditional referential constraints that enforce foreign key relationship between referencing table and another table in the database [14]. The key difference is the ability to specify an ontology query to determine the domain for the column. Also, the referenced ontology provides the semantics, namely relationships between ontology terms that allow for evolving a keyword column value based upon changes to the referenced ontology.

Our idea of automatically evolving the keyword column has similarity with the techniques used in ontology aligning and merging [12, 13]. For example, the PROMPT [12] ontology merging tool employs a semi-automatic method to assist user in the tedious task of merging ontologies with overlapping domains. However, it uses a general (ontology-neutral) knowledge model, whereas our approach relies on the semantics captured in the ontology. Also, the nature of tasks in the two cases are different, namely, merging/aligning of ontologies vs. evolution of keyword columns with changes to the referenced ontologies.

**Organization of the paper:** Section 2 gives the terminology and describes the problem. Section 3 presents the key concepts of our approach and illustrates them via several examples. Section 4 discusses how this can be implemented in Oracle DBMS. Section 5 discusses some issues related to ontology-based referential integrity constraint for keyword columns. Section 6 concludes with a summary and future directions.

## 2. Problem description: concepts and terminology

This section motivates the key concepts and related terminology needed to illustrate the full scope of the problem of supporting ontology-based referential constraints in database management systems.

### 2.1. Basic terminology

*Keyword Column* refers to a column of database table that holds a keyword. In practice, a row may be qualified by one or more keywords. A column holding multiple keywords is referred to as *Keyword-Set Column*.

*Ontology* is an explicit conceptualization of knowledge belonging to a particular domain. It consists of terms (concepts) and their relationships. A *Mapping Ontology* provides mapping between two or more ontologies which have overlapping domains.

*Resource Description Framework (RDF)*[5] allows one to assert facts such as relationships in <subject, predicate, object> triple format. *RDF Schema (RDFS)* [6] provides vocabulary and a basic set of inference rules. Simple ontologies can be defined using RDFS. For more complex ontologies, *Web Ontology Language (OWL)* [7] can be used, which uses RDF and provides more complex relationships.

An *Entailed Ontology* refers to ontology plus any triples inferred by applying associated (RDFS and/or OWL) rules.

### 2.2. Constraining keyword columns

*Ontology-based referential constraint* for a keyword column constrains values to terms or a subset of terms from the referenced ontology. The relevant concepts are as follows.

- *Constrained Keyword Column (K)*: The keyword column whose domain of possible values is constrained by the referential constraint.
- *Referenced Ontology (O)*: The ontology is referenced by the constraint. This is analogous to the parent table in a traditional referential integrity constraint.
- *Constraint Query (Q)*: This query is posed against the referenced ontology *O* to obtain a set of terms from the referenced ontology. These terms constitute the domain for the constrained keyword column. The ability to specify the constraint query allows the user to control the content of the domain.
- *Constraint Query Result Set (R)*: This refers to the set of terms returned by the constraint query and is also the domain of the constrained keyword column.

**Transformation and refinement functions:** The terms in the domain of a keyword column need not be treated as independent of one another. The knowledge of relationships between these terms in the referenced ontology makes this domain different than the simple enumerated sets seen in traditional referential constraints. This additional knowledge facilitates automatic evolution of keyword column content to maintain the constraint as the referenced ontology changes over time. The relevant concepts here are as follows:

- *Transformation function ( $f_i$ )*: When changes to the

referenced ontology cause deletion of a term from the domain of a keyword column, any use of that term in the keyword column must be eliminated to maintain validity. Replacement with NULL is always an option. However, the availability of knowledge, from the version of ontology prior to the change, about relationships of a deleted term with other terms that are present in the modified domain, allows exploiting this knowledge to compute possible replacement(s). This computation may fail to find any replacements or find one or more possible replacement values. The logic for this computation may be provided by a user-specified function with the following signature and functionality:

$$f_i: (O_{old}, R_{new}, DeletedTerm) \rightarrow TermSet$$

where,  $O_{old}$  is the version of ontology prior to the change,  $R_{new}$  is the new constraint query result set, and DeletedTerm is a term that is currently used in the keyword column and is not present in  $R_{new}$ . The TermSet returned could be NULL, indicating no replacements could be found, or a (possibly singleton) subset of  $R_{new}$ . If a single replacement value is returned by the function, then the actual replacement can take place automatically. Otherwise, the set of replacement values may be stored as a suggestion.

- *Refinement function ( $f_r$ ):* Changes to the referenced ontology may also add new terms from the ontology to the domain of a keyword column. Availability of the knowledge of all relationships between a new term and other terms in the current domain of the keyword column allows exploiting this knowledge to compute possible refinements of one or more of the currently used values with the newly added term (i.e., their replacement with the newly added term which is deemed more suitable by the computation). The logic for this computation may be modeled via a user-specified function with the following signature and basic functionality:

$$f_r: (O_{new}, R_{new}, InsertedTerm) \rightarrow TermSet$$

where,  $O_{new}$  and  $R_{new}$  are the new ontology and the new constraint query result set, and InsertedTerm is a term that has been newly added to the result set  $R_{new}$ . The TermSet returned could be NULL or a subset of  $R_{new}$  and may be used to suggest possible refinements.

**Batch refinement:** The concepts identified so far are sufficient to maintain the validity of a keyword column with an ontology-based referential constraint. That is, they ensure that the values present in a keyword column are always restricted to the constraint query result set obtained using the constraint query against the current content of the ontology. However, the recommended set

of more suitable values is not necessarily the most up-to-date. This is because not every change to the ontology may result in a change in the constraint query result set. Some such changes, that do not cause any change in the constraint query result set, may actually cause a different term or set of terms to become more suitable and hence possible refinements of a value present in the keyword column or suggested earlier as a possible replacement or refinement.

Use of a batch refinement operation would allow generating recommendations based upon the current value of the referenced ontology. The signature and basic functionality of this operation may be modeled as follows:

$$f_{batch\_refine}: (O_{current}, R_{current}, ValueSet) \rightarrow RefineSet$$

where  $O_{current}$  and  $R_{current}$  are the current values of the referenced ontology and the constraint query result set, the ValueSet is the set of current entries in the keyword column and values suggested as refinements or replacements, and RefineSet is set of <value in ValueSet, set of possible refinements>. This batch refinement operation may be used to suggest possible refinements based upon current value of the ontology.

### 2.3. Querying keyword columns

The semantics of querying on keyword columns with referential constraints needs to be extended. In addition to traditional keyword match based on lexical equality, user should be able to perform a semantic match on the keyword column by taking the associated ontology into consideration. For example, a keyword column entry 'S&H' may be matched against 'Shipping & Handling' if the ontology associated with the keyword column says 'S&H' is same as 'Shipping & Handling'.

The ability to perform such semantic match needs to be extended to both single table queries as well as queries that join on keyword columns with ontology-based referential constraints. In general, determining the matching ontology (that is, the ontology to use for matching) in a semantic match operation with various kinds of ontology associations for the operands may be as follows:

- When neither operand has an associated ontology, a user-specified ontology (if any) may be used as the matching ontology.
- When only one of the operands is a keyword column with an associated ontology, that ontology may be used as the matching ontology.
- When both of the operands are keyword columns with associated ontologies, and they are the same ontology, then use that ontology as the matching ontology.
- When both of the operands have associated

ontologies, and they are distinct ontologies, use these ontologies with a mapping ontology. The mapping ontology may be user-specified or may be selected based upon availability.

### 3. Our approach

This section describes our approach. First an overview is presented, which is followed by key concepts of our approach.

#### 3.1. Overview

The basic approach is as follows:

- Extend the traditional referential integrity constraint to allow a query to be specified as part of the constraint definition. However, only a restricted class of queries is allowed. Specifically, the queries can reference a single ontology (further details in Section 4.2).
- Provide support for associating transformation function and refinement function with the constraint.
- A procedural API is provided that allows for batch refinement of keyword column terms.

#### 3.2. Ontology-based referential constraints for keywords columns

Traditional referential integrity constraints that enforce foreign key relationship between referencing table and another table is specified by the syntax of the form '`<column a> REFERENCES <table b> (<column c>)`', where `<column a>` is a foreign key and `<column c>` is a primary key of `<table b>`. The database system ensures that the `<column a>` only references values in `<column c>`.

This is extended to support ontology-based referential integrity constraint mechanism as follows:

```
<column a> REFERENCES (<ontology query>)
```

For example, a user wants to create a journal article table that contains a keyword `<symptoms>` for signs and symptoms of diseases. The keyword refers to the medical subject heading ontology MeSH [3]. The user can specify the referential constraint using `ONT_EXPAND` table function as follows:

```
symptoms REFERENCES (  
  SELECT Term1Name  
  FROM TABLE (ONT_EXPAND (  
    NULL,  
    'rdfs:subClassOf',  
    'Signs and Symptoms',  
    'MeSH')));
```

The table function `ONT_EXPAND` returns all terms that are subclasses of the term 'Signs and Symptoms' defined in the ontology MeSH. Note that `<ontology`

`query>` may contain any ontology vocabularies including OWL.

Now, consider the case where the user wants to restrict the column values to keywords within a distance of 4 (that is, from the term 'Signs and Symptoms') to avoid too specific terms. The user may specify using a filter condition like:

```
symptoms REFERENCES (  
  SELECT Term1Name  
  FROM TABLE (ONT_EXPAND (  
    NULL,  
    'rdfs:subClassOf',  
    'Signs and Symptoms',  
    'MeSH'))  
  WHERE TermDistance < 4);
```

The `TermDistance` and `TermPath` are distance and path measures of the returned term (`Term1Name`) with respect to the term 'Signs and Symptoms'. They are computed as part of `ONT_EXPAND` and are available as additional columns of the table returned from `ONT_EXPAND`.

While `ONT_EXPAND` provides a useful functionality, it is confined to a single triple pattern. For more complex triple patterns, `RDF_MATCH` table function [11] can be used. `RDF_MATCH` can handle any number of triple patterns. For example, let us assume that a user wants to create a journal article table for Macromolecules and its column `<chemical_compound>` contains a keyword for specific chemical compound type that the article is written about such as 'Proteins' and 'Polypeptides'. The Pathway/Genome ontology BioPax [1] contains all kinds of chemical compounds terms. The user may specify like:

```
chemical_compound REFERENCES (  
  SELECT t.r  
  FROM TABLE (RDF_MATCH (  
    '(?r rdfs:subClassOf Macromolecules)',  
    RDFModels ('BioCyc'),  
    RDFRules ('rdfs'), NULL)) t);
```

`RDF_MATCH` table function is used for querying the ontology 'BioCyc' and the search pattern is specified using SPARQL-like syntax [15], where variable starts with '?' character. Here `?r` represents all terms that are subclasses of the term 'Macromolecules'. It also uses RDFS inferencing to determine if a chemical compound term belongs to Macromolecules. The query will generate all terms that belong to Macromolecules compounds. Therefore the `<chemical_compound>` column in the user table will contain only Macromolecules compounds.

If the user wants to further restrict the `<chemical_compounds>` column to compounds, which belong to both classes of 'Proteins' and 'Complexes', the user may issue the query like:

```
chemical_compound REFERENCES (  
  SELECT t.r  
  FROM TABLE (RDF_MATCH (  
    '(?r rdfs:subClassOf Proteins)',  
    '(?r rdfs:subClassOf Complexes)',  
    RDFModels ('BioCyc'),  
    RDFRules ('rdfs'), NULL)) t);
```

```

`(?r rdfs:subClassOf Proteins)
 (?r rdfs:subClassOf Complexes)',
RDFModels ('BioCyc'),
RDFRules ('rdfs'), NULL)) t);

```

Here the same variable `?r` should match to the same term, hence it should be a subclass of both `Proteins` and `Complexes`.

Also, if the user wants to have the compound keyword restrict to small molecules that belong to `Vitamins`, `Ions`, and `Hormones` only, then the user can issue the query using a filter condition like:

```

chemical_compound REFERENCES (
  SELECT t.s
  FROM TABLE (RDF_MATCH (
    `(?r rdfs:subClassOf Small_Molecules)
    (?s rdfs:subClassOf ?r)',
    RDFModels ('BioCyc'),
    RDFRules ('rdfs'), NULL)) t
  WHERE t.r IN
    ('Vitamins', 'Ions', 'Hormones'));

```

Multiple keywords can be handled using a collection type for <chemical\_compound> column. However, the referential constraint is defined on the keyword column of the underlying storage table (Details in Section 5.2).

**On ontology update semantics:** When a term is deleted from the ontology that is referenced by the user table keyword column, there are two options that the user can use to maintain the referential integrity:

- `ON DELETE SET NULL`: This option sets the keyword entries in the keyword columns to NULL that has been deleted from the ontology. This is the default behavior.
- `ON DELETE TRANSFORM USING <function>`: This option replaces the term deleted with the term returned by <function> whenever it is possible. If the deleted term cannot be unambiguously replaced then the matching keyword entries are set to NULL. In addition, the <rowid\_containing\_deleted\_term, deleted\_term, returned\_values, 'Delete'> are stored in a RECOMMENDATIONS table as possible refinement suggestions for the user.

The following example shows how the transformation function can be used when a term is deleted from the MeSH ontology:

```

symptoms REFERENCES (
  SELECT Term1Name
  FROM TABLE (ONT_EXPAND (
    NULL,
    'rdfs:subClassOf',
    'Signs and Symptoms',
    'MeSH'))))
ON DELETE TRANSFORM USING find_broader_term;

```

The function `find_broader_term` can be defined as follows:

```

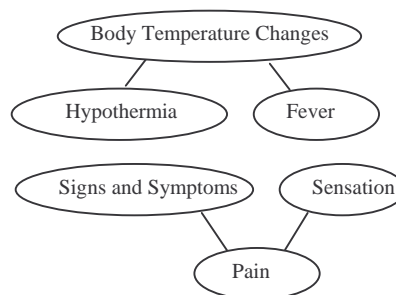
function find_broader_term (Oold, Rnew, dterm)

```

```

{
  Open a cursor for the query that finds
  broader terms for the deleted term like:
  SELECT subject
  FROM Oold
  START WITH object = dterm
  CONNECT BY PRIOR subject = object
    AND (PRIOR subject ≠ Rnew);
  Fetch rows using the cursor and return the
  resulting set
}

```



Rowid	Term	Recommendations	Action
.....	Pain	Sensation Signs and Symptoms	Delete
NULL	Sensation	Hearing Smell Taste	Insert

Figure 2: Sample Terms & RECOMMENDATIONS table

In Figure 2 above, if the term `Hypothermia` is deleted from the ontology, the term will be replaced with the term `Body Temperature Changes`. On the other hand, if the term `Pain` is deleted, the term will be set to NULL and the possible replacements are generated in the RECOMMENDATIONS table as shown in Figure 2. Deletions from the user table have no effect on the constraint validation as is the case with traditional referential constraints.

In addition, deletes (or inserts or updates) to ontology may require refining the constraint query itself. This issue requires further work and is not addressed in this paper.

When a term is inserted into the ontology, the following option is provided to allow automatic generation of possible refinement terms:

- `ON INSERT RECOMMEND USING <function>`. This option generates recommendations for refining keyword entries that may possibly be replaced with the newly inserted term. This information is stored in the RECOMMENDATIONS table in the form of (NULL, inserted\_term, replacement\_terms, 'Insert').

In Figure 2, if the new terms such as `Hearing`, `Smell` and `Taste` are inserted under the term `Sensation`, the refinement function might consider these new terms to be suitable refinements for `Sensation`, in which case the RECOMMENDATIONS table will contain a row as shown in Figure 2.

**DISABLE and ENABLE constraint:** When data is loaded in batch into a table, it would be a good idea to disable the constraints and enable them back when the data has been completely loaded. The exceptions go into exceptions table so that the user can take appropriate action on the exceptions data.

There are two cases where the ontology is bulk-loaded and the user table is bulk-loaded. If the ontology is batch-appended, there are no exceptions on the user table because all referenced terms remain the same. However, recommendations for possible refinements of existing keywords with some of the newly inserted terms will be generated. When a user table is batch-appended, there may be terms that do not exist in the ontology and those exceptions will go into the exceptions table when the constraint is enabled. The exceptions table contains row identifiers so that users can get access to table rows to modify the keyword column.

For batch operations involving updates, the referential constraint query results before the batch operation and after are compared and new terms in the results are treated as inserts and disappeared terms in the results are treated as deletes. Then the inserts/deletes are processed according to the ontology update semantics described above.

### 3.3. Semantic matching on keyword columns

Once the keyword columns are populated users can do semantic matching on those columns by referring to the ontology. Consider a `medical_articles` (`article_id`, `keyword`) table (Figure 3) that maintains the keywords for medical articles which are constrained to MeSH ontology medical terms.

To find articles about the sensation, user can formulate the following query:

```
SELECT m.article_id
FROM medical_articles m
WHERE ONT_RELATED(m.keyword,
    'rdfs:subClassOf',
    'Sensation',
    'MeSH') = 1;
```

Here the `ONT_RELATED` operator matches the values of the keyword column with the terms belonging to the category 'Sensation' by using the 'rdfs:subClassOf' relationship. The terms are matched by finding transitive closure from the term 'Sensation'. The query will return `<A1, A2, A3, ... >`.

Article_Id	Keyword
A1	Sensation
A2	Hearing
A3	Smell
...	...

Figure 3: Medical\_Articles Table

The operator can also be used for querying over a collection of heterogeneous tables (that is, individual tables use separate ontologies for their keyword column) using an integrated ontology [17]. Consider two patient tables `p1` and `p2` with keyword columns referencing 'SNOMED' ontology [1] and 'ICD-9' ontology [18] respectively. User would like to query using canonical terms defined in 'LinkBase' [19], which contains mappings for both 'SNOMED' and 'ICD-9'. A query using `ONT_RELATED` operator can be issued that references 'LinkBase' terms as follows:

```
SELECT p1.patient_id ID FROM p1
WHERE ONT_RELATED(p1.keyword, 'IS_A',
    <canonical_cancer_term>,
    'LinkBase')=1

UNION

SELECT p2.patient_id ID FROM p2
WHERE ONT_RELATED(p2.keyword, 'IS_A',
    <canonical_cancer_term>,
    'LinkBase')=1;
```

Finding a semantic join between two keyword columns from different/same user tables can also be done using `ONT_RELATED` operator with extended semantics. For example, if we have a `researcher_interests` (`researcher_id`, `keyword`) table that maintains the keywords identifying subject areas that are of interest to the researcher, we may want to match researcher's interest with the articles above. Such semantic join can be captured by overloading the `ONT_RELATED` operator. However, the detailed discussion is beyond the scope of this paper.

## 4. Implementation scheme

This section discusses how ontology-based constraints can be supported in DBMS. The implementation scheme is described in the context of Oracle DBMS. However, most of the DBMS features used are generic in nature and hence the scheme could easily be supported in other commercial DBMS.

### 4.1. System architecture

Figure 4 shows the overall architecture of the system. The role of various components is described below.

**Ontology storage and querying:** We assume that the DBMS has been extended to allow storage and querying of OWL ontologies. In particular, we use the implementation described in [10], which provides SQL table function `ONT_EXPAND` to query ontologies. In addition, for supporting transformations on deletion, one should be able to query the old version of ontology. Oracle's Flashback query [16] may be used to provide this functionality. For simplicity, we assume that concurrent modifications to the ontology are not permitted.

**Ontology editor:** Ontologies are typically modified using an ontology editor such as Protégé [8] and the corresponding changes are performed against the underlying database.

**Constraint definition module:** The traditional constraint definition module needs to be extended to handle the new syntax, namely the specification of ontology query.

**Constraint enforcement module:** This module contains bulk of the changes, namely for enforcing the constraint as well as generating recommendations. This is covered in details in sections below.

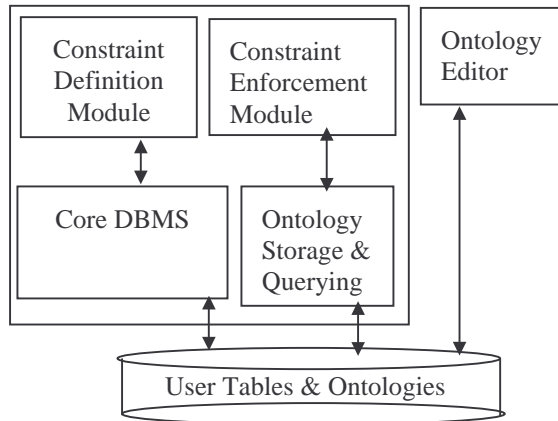


Figure 4: Overall Architecture

## 4.2. Overview

The implementation scheme leverages Oracle’s materialized view mechanism [9] and the traditional referential constraint mechanism. Specifically, a materialized view is created for the referential constraint query result and a conventional referential integrity constraint is defined between the user table keyword column and the terms in the materialized view thereby ensuring that the keyword terms are constrained (Figure 5).

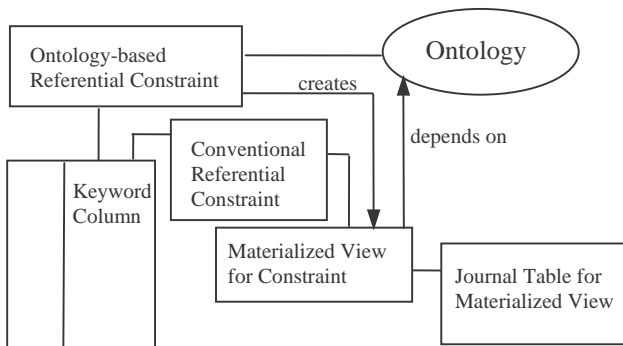


Figure 5: Constraint Enforcement

The materialized view can be maintained (refreshed) incrementally or fully on demand as the underlying

ontology is modified. Therefore, the restrictions imposed upon Oracle’s materialized view mechanism for incremental maintenance (for example, the constraint query does not use SELECT list subqueries or NOT EXISTS predicates in nested queries) still hold in our approach.

In all cases we provide ‘batch enforcement’ of the constraint, which is intended for use with batch ontology modifications where the constraint is disabled before load and enabled afterwards. This batch enforcement could also be used to validate the constraint as the ontology is incrementally modified.

When the materialized view is incrementally maintained we provide ‘incremental enforcement’ of the constraint to efficiently validate the constraint as the ontology is modified.

## 4.3. Batch enforcement

For batch enforcement, additional processing is needed for both DISABLE and ENABLE constraint operations. The details are described below.

**DISABLE constraint:** When the ontology-based constraint is disabled, a copy of current materialized result set is created. This is subsequently used to identify changes in the referential constraint query result. Also, we save the current version information for the ontology.

**ENABLE constraint:** Let  $R_0$  be the constraint query result set when the constraint was last disabled, and  $R_1$  be the current constraint query result set. We determine the set of deleted terms using the following query:

```
SELECT term FROM R0
MINUS
SELECT term FROM R1
```

If the constraint has a transformation function associated with it, we invoke it for these terms (passing the ontology version information saved when the constraint was disabled). If the transformation function yields a single result ( $T_{new}$ ) for a deleted term ( $T_{old}$ ), we update all instances of  $T_{old}$  in the keyword column to  $T_{new}$ . If the transformation function yields either no terms or multiple terms, then the ROWIDs of all rows that reference  $T_{old}$  are recorded in the recommendations table along with the transformation function results before all instances of  $T_{old}$  in the keyword column are updated to NULL. If the constraint does not have a transformation function, then all instances of  $T_{old}$  in the keyword column are updated to NULL.

Similarly, to determine the set of insertions to the constraint query result set, the following query is used.

```
SELECT term FROM R1
MINUS
SELECT term FROM R0
```

If the constraint has a refinement function we invoke

it for these terms. The refinement function's results are stored in the recommendations table.

Finally the underlying conventional constraint is enabled with the appropriate EXCEPTIONS clause (as derived from any EXCEPTIONS clause used when the ontology-based constraint was enabled). If this is successful the RO result set (which was saved when the constraint was disabled) can be dropped.

This scheme could also be used as part of a protocol to enforce the constraint during incremental ontology modifications. In this case the constraint would be disabled before the modification is started and enabled after the modification is completed.

#### 4.4. Incremental enforcement

When the materialized view is incrementally maintained, the conventional referential integrity constraint between the constrained table and the materialized view is altered to a deferred constraint (i.e., a constraint that's enforced only at transaction commit-time) with ON DELETE SET NULL semantics. With a deferred constraint we avoid spurious constraint violations during transient states that may exist during materialized view maintenance.

Row-level triggers on the materialized view journal all changes in the constraint query result set to a temporary journal table. A row-level update trigger on the keyword column is used to write orphaned keywords and their row's rowid to the recommendation table when the ON DELETE SET NULL behavior updates an orphaned keyword. (If a keyword is updated to null by explicit DML this trigger will log the keyword and its rowid to the recommendation table, however since there is no ontology update in this case the update will not be further processed by the constraint's refinement or translation mechanisms.)

After the materialized view is maintained (i.e., after changes to the underlying ontology tables have committed) we process the journal table to remove idempotent operations (e.g., a term may be deleted and reinserted during materialized view maintenance). Then the appropriate calls are made to the constraint's refinement and translation functions based on the remaining operations, and the results are propagated to the keyword column and the constraint's recommendation table.

#### 4.5. Improved incremental enforcement

There are several problems with the above incremental enforcement implementation:

- The ON DELETE SET NULL referential constraint may set keywords to NULL as the materialized view

evolves through transient states, even though the keyword will not be orphaned when the maintenance is complete.

- The journal table is processed as part of a follow-up transaction, and therefore any translations and recommendations we make are not atomic with the changes to the ontology.
- If it is not possible to refresh the materialized view because one of the objects it references has been dropped or altered then the user will see unexpected errors during commit (Note that this can also occur during batch enforcement).

To address these problems, we propose introducing AFTER REFRESH and AFTER REFRESH ERROR triggers on materialized views in Oracle. Such triggers would run as part of the transaction that refreshes the materialized view.

With these triggers, we can improve the incremental enforcement scheme as follows:

- The referential integrity constraint between keyword column and constraint query result set does not need ON DELETE SET NULL semantics, and the row-level update trigger on the keyword column is not needed.
- In an AFTER REFRESH trigger on the materialized view, the journal table can be processed, the appropriate refinement and translation function calls can be made, and the keyword column and recommendation table can be updated.
- In an AFTER REFRESH ERROR trigger on the materialized view we can raise a helpful error, such as "ontology-based referential constraint query requires modification."

### 5. Issues

This section discusses two issues that require further work.

#### 5.1. Tracking dependency between keyword column and referenced ontology

The ontology-based referential constraint creates a dependency from the keyword column to the referenced ontology. The challenge, however, is to capture the precise nature of this dependency. The dependency could be coarsely captured as entire ontology or at a finer level as a portion of the ontology consisting of some of the terms and some of the relationships between those terms addressed by the constraint query. Further work is needed in precisely identifying this dependency and in developing efficient ways of detecting changes to the dependent set of terms and relationships. This is further complicated as an arbitrarily complex query on ontology

can be used to determine the domain of the keyword column. Also, OWL inferencing rules introduce another set of transformations. Namely, a change in ontology causes a change to entailed ontology, which in turn causes a change to the domain of the column as determined by the constraint query.

## 5.2. Supporting constraint on keyword-set columns

The solution presented considered column with single keyword column. This could be extended in a straightforward manner for keyword-set column (containing multiple keywords) by representing it using a collection type (referred to as nested table column in Oracle). Oracle creates a single underlying storage table to hold the nested table column values for all the rows. Thus, by creating the ontology-based referential constraint on the underlying storage table one could enforce the data integrity for keyword-set columns. However, to support semantic match on keyword-set column, new operators are needed that can take multiple keywords into consideration for performing the semantic match. Also, additional work is required to address the case when the keyword-set column references terms from multiple ontologies.

## 6. Conclusions

The paper presented the concept of ontology-based referential constraints for keyword columns. A distinguishing aspect of these constraints is the use of a query on ontology for identifying the domain for the keyword column. Furthermore, to facilitate evolution of keyword column with ontology, the constraint can optionally specify a transformation function to handle deletion of referenced terms and a refinement function to generate recommendation in the event of addition of new terms to keyword column domain.

The DBMS extensions needed to support ontology-based referential constraint were outlined in the context of Oracle. Also, the implementation scheme discussed how they could be supported using Oracle's materialized views, triggers, and flashback queries.

Supporting ontology-based referential constraints in DBMS greatly enhances the data integrity of keyword column as well as opens up the possibility of querying by semantically matching keyword column to ontology terms.

In future, we plan to conduct experiments with synthetic data sets to characterize the overheads incurred in enforcing such referential constraints.

## Acknowledgments

We thank Jay Banerjee for his comments on an earlier version of this paper.

## References

- [1] BioPAX, <http://www.biopax.org>
- [2] SNOMED Clinical Terms, [http://www.nlm.nih.gov/research/umls/Snomed/snomed\\_main.html](http://www.nlm.nih.gov/research/umls/Snomed/snomed_main.html).
- [3] Mesh Medical Subject Headings, <http://www.nlm.nih.gov/mesh/meshhome.html>
- [4] National Cancer Institute Thesaurus and Ontology, <http://www.mindswap.org/2003/CancerOntology>.
- [5] RDF Primer, W3C Recommendation 10 February 2004, <http://www.w3.org/TR/rdf-primer>.
- [6] RDF Vocabulary Description Language 1.0: RDF Schema, W3C Recommendation 10 February 2004, <http://www.w3.org/TR/rdf-schema>.
- [7] OWL Web Ontology Language Reference, <http://www.w3.org/TR/owl-ref>.
- [8] The Protégé Ontology Editor and Knowledge Acquisition System, <http://protege.stanford.edu/>
- [9] R. G. Bello, et al., "Materialized Views in Oracle," *Proceedings of the 24<sup>th</sup> Int. Conf. on Very Large Data Bases*, 1998, pp.659-664.
- [10] S. Das, E. I. Chong, G. Eadon, and J. Srinivasan, "Supporting Ontology-based Semantic Matching in RDBMS," *Proceedings of the 30<sup>th</sup> Int. Conf. on Very Large Data Bases*, pp.1054-1065, Aug. 2004.
- [11] E. I. Chong, S. Das, G. Eadon, and J. Srinivasan, "An Efficient SQL-based RDF Querying Scheme," *Proceedings of the 31<sup>th</sup> Int. Conf. on Very Large Data Bases*, pp.1216-1227, Aug. 2005.
- [12] N. F. Noy, M. A. Musen, "PROMPT: Algorithm and Tool for Automated Ontology Merging and Alignment," AAAI/IAAI 2000, pp.450-455.
- [13] D.L. McGuinness, R. Fikes, J. Rice, and S. Wilder, "An Environment for Merging and Testing Large Ontologies," *Proc. of the 7th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR2000)*.
- [14] C. J. Date, *An Introduction to Database Systems, Eighth Edition*, Addison Wesley, 2003.
- [15] *SPARQL Query Language for RDF*, W3C Working Draft, 12 October 2004, <http://www.w3.org/TR/2004/WD-rdf-sparql-query-20041012/>
- [16] *Oracle® Database Concepts*, 10g Release 2 (10.2), Part Number B14220-01.
- [17] J.-L. Verschelde, M. Casella D.Santos, T. Deray, B. Smith and W. Ceusters, "Ontology Assisted Database Integration to Support Natural Language Processing and Biomedical Data-Mining," *Journal of Integrative Bioinformatics*, January 2004.
- [18] The International Classification of Diseases, <http://www.cdc.gov/nchs/icd9.htm>
- [19] LinkBase, <http://www.landcglobal.com/pages/linkbase.php>