Oracle Database 11g Release 2
Dive into Semantic Technologies: Security, Document Index, and Change Management

Aravind Yalamanchi, Consultant Member of Technical Staff
Bill Beauregard, Senior Principal Product Manager
<table>
<thead>
<tr>
<th>Date/Time</th>
<th>Title</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sunday, Oct. 11</td>
<td>Build Better SQL and Document Queries with Oracle Semantic Technologies, Part 1</td>
<td>Hilton Hotel Continental Parlor 1/2/3</td>
</tr>
<tr>
<td>10:30 a.m.</td>
<td>Build Better SQL and Document Queries with Oracle Semantic Technologies, Part 2</td>
<td>Hilton Hotel Continental Parlor 1/2/3</td>
</tr>
<tr>
<td>11:45 a.m.</td>
<td>Dive into Semantic Technologies: Security, Document Index, and Change Management</td>
<td>Hilton Hotel Golden Gate 2</td>
</tr>
<tr>
<td>2:30 p.m.</td>
<td>Put SPARQL in Your Code: Building Applications with Oracle Semantic Technologies</td>
<td>Hilton Hotel Golden Gate 2</td>
</tr>
<tr>
<td>Tuesday, Oct. 13</td>
<td>Turning the Database Inside-Out with Complex Event Processing</td>
<td>Hilton Hotel Golden Gate 1</td>
</tr>
</tbody>
</table>
# Semantics at OOW 2009 - Sessions

<table>
<thead>
<tr>
<th>Date/Time</th>
<th>Title</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wednesday, Oct. 14</td>
<td>Who’s Who and What’s What with Oracle Database Semantic Technologies</td>
<td>Moscone South Room 270</td>
</tr>
</tbody>
</table>

- DEMOgrounds
  - Oracle Database Semantic Technologies - *Moscone West, W-018*
The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described for Oracle’s products remains at the sole discretion of Oracle.
Agenda

- Overview of Oracle Database Semantic Technologies
- Strong security
- Semantic indexing of documents
- Change management for collaboration
Semantic Data Mgt. Characteristics

• Discovery of data relationships across…
  – Structured data (database, apps, web services)
  – Unstructured data (email, office documents) Multi-data types (graphs, spatial, text, sensors)

• Text Mining & Web Mining infrastructure
  – Terabytes of structured & unstructured data

• Queries are not defined in advance
• Schemas are continuously evolving
• Associate more meaning (context) to enterprise data to enable its (re)use across applications
• Allow sharing and reuse of enterprise and web data.
• Built on open, industry W3C standards:
  – SQL, XML, RDF, OWL, SPARQL
Semantic Database: Store, Infer, Query
Semantic Database: Store, Infer, Query

:California :partOf :USA :partOf :NorthAmerica

:partOf rdf:type owl:TransitiveProperty
Semantic Database: Store, Infer, Query

 Asserted Facts

<table>
<thead>
<tr>
<th>:partOf</th>
<th>rdf:type</th>
<th>owl:TransitiveProperty</th>
</tr>
</thead>
<tbody>
<tr>
<td>:California</td>
<td>:partOf</td>
<td>:USA</td>
</tr>
<tr>
<td>:USA</td>
<td>:partOf</td>
<td>:NorthAmerica</td>
</tr>
</tbody>
</table>
Semantic Database: Store, Infer, Query

Asserted Facts

<table>
<thead>
<tr>
<th>:partOf</th>
<th>rdf:type</th>
<th>owl:TransitiveProperty</th>
</tr>
</thead>
<tbody>
<tr>
<td>:California</td>
<td>:partOf</td>
<td></td>
</tr>
<tr>
<td>:USA</td>
<td>:partOf</td>
<td></td>
</tr>
<tr>
<td>:NorthAmerica</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Semantic Database: Store, Infer, Query

Asserted Facts

<table>
<thead>
<tr>
<th>:partOf</th>
<th>rdf:type</th>
<th>owl:TransitiveProperty</th>
</tr>
</thead>
<tbody>
<tr>
<td>:California</td>
<td>:partOf</td>
<td>:USA</td>
</tr>
<tr>
<td>:USA</td>
<td>:partOf</td>
<td>:NorthAmerica</td>
</tr>
</tbody>
</table>

Derived Facts

| :California | :partOf     | :NorthAmerica           |
Semantic Database: Store, Infer, Query

**Asserted Facts**

<table>
<thead>
<tr>
<th>:partOf</th>
<th>rdf:type</th>
<th>owl:TransitiveProperty</th>
</tr>
</thead>
<tbody>
<tr>
<td>:California</td>
<td>:partOf</td>
<td>:USA</td>
</tr>
<tr>
<td>:USA</td>
<td>:partOf</td>
<td>:NorthAmerica</td>
</tr>
</tbody>
</table>

**Derived Facts**

| :California | :partOf | :NorthAmerica |

**Query:**

```
SELECT ?x ?y
FROM ...
WHERE { ?x :partOf ?y }
```
Semantic Database: Store, Infer, Query

Asserted Facts

<table>
<thead>
<tr>
<th>:partOf</th>
<th>rdf:type</th>
<th>owl:TransitiveProperty</th>
</tr>
</thead>
<tbody>
<tr>
<td>:California</td>
<td>:partOf</td>
<td>:USA</td>
</tr>
<tr>
<td>:USA</td>
<td>:partOf</td>
<td>:NorthAmerica</td>
</tr>
<tr>
<td>:partOf</td>
<td>rdf:type</td>
<td>owl:TransitiveProperty</td>
</tr>
</tbody>
</table>

Derived Facts

| :California | :partOf | :NorthAmerica |

Query: SELECT ?x ?y
FROM ...
WHERE { ?x :partOf ?y }

Result:

<table>
<thead>
<tr>
<th>?x</th>
<th>?y</th>
</tr>
</thead>
<tbody>
<tr>
<td>:California</td>
<td>:USA</td>
</tr>
<tr>
<td>:California</td>
<td>:NorthAmerica</td>
</tr>
<tr>
<td>:USA</td>
<td>:NorthAmerica</td>
</tr>
</tbody>
</table>
Text Mining: National Intelligence

Web Resources

Information Extraction

Categorization, Feature/term Extraction

Processed Document Collection

RDF/OWL

Ontologies

Domain Specific Knowledge Base

Ontology Engineering Modeling Process

SQL/SPARQL Query

Explore

Analyst

Browsing, Presentation, Reporting, Visualization, Query

Content Mgmt. Systems

News, Email, RSS
Data Integration Platform in Health Informatics

Enterprise Information Consumers (EICs)

- Patient Care
- Workforce Management
- Business Intelligence
- Clinical Analytics

Integration Server (Semantic Knowledge base)

Access

Run-Time Metadata

Deploy

Model Virtual

Relate

Model Physical

Access

LIS, CIS, HTB, HIS
Data Integration Platform in Health Informatics

Enterprise Information Consumers (EICs)
- Patient Care
- Workforce Management
- Business Intelligence
- Clinical Analytics

Run-Time Metadata
- Integration Server (Semantic Knowledge base)

Access

Deploy

Model Virtual

Relate

Model Physical

Access

LIS, CIS, HTB, HIS
Oracle database 11g is the leading commercial database with native RDF/OWL data management.

- Scalable & secure platform for wide-range of semantic applications
- Readily scales to ultra-large repositories (+1 billion)
- Choice of SQL or SPARQL query
- Leverages Oracle Partitioning, RAC supported
- Growing ecosystem of 3rd party tools partners

**Key Capabilities:**

**Load / Storage**
- Native RDF graph data store
- Manages billions of triples
- Fast batch, bulk and incremental load

**Query**
- SQL: SEM_Match
- SPARQL: via Jena plug-in
- Ontology assisted query of RDBMS data

**Reasoning**
- Forward chaining model
- RDFS++ OWL, OWL Prime
- User defined rule base
Semantic Data Management Workflow

Edit & Transform
- Entity Extraction & Transform
- Ontology Engineering
- Categorization
- Custom Scripting

Load, Query & Inference
- RDF/OWL Data Management
- SQL & SPARQL Query
- Inferencing
- Semantic Rules
- Scalability & Security

Applications & Analysis
- Graph Visualization
- Link Analysis
- Statistical Analysis
- Faceted Search
- Pattern Discovery
- Text Mining

Transaction Systems
Unstructured Content
RSS, email
Other Data Formats
Data Sources
Partners

ORACLE SPATIAL

Partners
Semantic Technology Partners
Integrated Tools and Solution Providers:
Release 11.1 Semantic Technologies

- W3C standards: RDF, RDFS, OWL, SPARQL
- SQL & DML access to RDF/OWL data and ontologies
  - Querying in SQL w/ SPARQL-like graph patterns
  - Querying w/ SPARQL in Java through Jena adaptor
  - Ontology-assisted querying of relational data
- Native Inferencing: RDFS/OWL & user-defined rules
  - Persistent, scalable, ahead of queries, plugable reasoners
- Model level security
- Bulk and incremental loading of triples
  - Up to 8 exabytes, 1 million graphs w/ multi-graph querying
  - Integral Oracle Partitioning
Release 11.2 Semantic Technologies

- Strong security for Semantic Technologies
  - Security policies and data classification for RDF data
- Semantic indexing of documents
  - Semantic indexing of documents based on popular natural language tools
- Faster, more efficient reasoning to find new relationships
  - Parallel and incremental inference, owl:sameAs optimization
- Change management for collaboration
- Standards & open source support
  - SPARQL query support for Filter, Union in SEM_MATCH table function
  - OWL: union, intersection, OWL 2 property chains, disjoint properties
  - Pellet OWL DL reasoner Integration
  - Jena V2.5
  - Java SDK for SPARQL for 3rd party integration e.g., Sesame
  - W3C Simple Knowledge Organization System (SKOS) & SNOMED ontology
  - Utility to swap, rename, and merge model, rename entailment, remove duplicates
Strong Security
Fine-Grained Access Control for RDF

- Fine-Grained Access Control mechanisms restrict access to data *within* the most critical database objects.
- Intercept and rewrite the user query to restrict the result set using additional predicates.
- Restricted result set only includes the rows the user “needed to know”.
- Offers triple-level security for RDF data for defense, intelligence, and commercial regulatory environments.
Enterprise Security for Semantic Data

• Access control policies on semantic data
  – Uses Virtual Private Database feature of Oracle Database
  – Applies declarative constraints to RDF classes and properties
  – Restricts access to parts of the RDF graph based on certain characteristics of the instance data and application/user context
  – E.g: Only a manager of a contract can access its monetary value.

• Data classification labels for semantic data
  – Uses Oracle Label Security option of Oracle Database
  – Assigns sensitivity labels to users and RDF data.
  – Restricts access to users having compatible access labels.
  – E.g: A triple marked as “Sensitive” is only accessible to the users with clearance for “Sensitive” or “Highly Sensitive” information.
Virtual Private Database for RDF Data

Ability to restrict access to parts of the RDF graph based on the application/user context.

- An individual can only access information about the project he works on.
- Monetary value of a project can only be accessed by the project lead or the VP of the department.
- Only a department VP can access information about the department’s budget.
VPD Constraints for RDF

• The security conditions or the data access constraints are associated with RDF Classes and Properties.
  – An instance of Project class can only be accessed by an individual with hasMember relationship with the instance.
  – The hasValue property for a Project instance can be accessed by an individual recognized as the project lead (hasLead property).

• Declarative constraints restrict access to parts of the RDF graph.
  – E.g: A user only has access to projects that are Active.
    { ?project rdf:type :Project } \rightarrow
      { ?project :hasStatus :Active }

• SPARQL queries on a VPD-enabled graph are automatically rewritten to include appropriate constraints.
Compile-time Analysis of Queries with VPD

Query: Get the list of projects and their values

```
SELECT ?proj ?val
FROM ProjectsGraph
WHERE { ?proj :hasValue ?val }
```
Compile-time Analysis of Queries with VPD

Query: Get the list of projects and their values

```
SELECT ?proj ?val
FROM ProjectsGraph
WHERE { ?proj [hasValue] ?val }
```
Compile-time Analysis of Queries with VPD

Query: Get the list of projects and their values

```
SELECT  ?proj  ?val
FROM    ProjectsGraph
WHERE   { ?proj   :hasValue  ?val }
```
Compile-time Analysis of Queries with VPD

Query: Get the list of projects and their values

```sql
SELECT ?proj ?val
FROM ProjectsGraph
WHERE {
  ?proj :hasValue ?val .
  ?proj :hasStatus :Active
}
```
Compile-time Analysis of Queries with VPD

To facilitate compile-time analysis, queries may not use unbound variables in the predicate position.

Query: Get the list of projects and their values

```
SELECT  ?proj  ?val
FROM    ProjectsGraph
WHERE   { ?proj   :hasValue  ?val .
          ?proj   :hasStatus :Active . }
```
VPD Constraints for RDF

- A VPD Constraint is used to enforce access restrictions on parts of the RDF graph, identified using its metadata.
- One or more constraints are grouped into a VPD policy that is applied to an RDF Graph.
- Each VPD constraint has a *match* pattern and an *apply* pattern, which are expressed as SPARQL graph patterns.
  - The *match* pattern identifies parts of the RDF graph being accessed by a user query.
    ```sparql
    ?project :hasValue ?val
    ```
  - The *apply* pattern defines additional graph patterns to be applied on the sub-graph before it can be used to construct the query results.
    ```sparql
    ?project :ownedBy ?dept .
    ?dept :hasVP "sys_context('sa $env','user')"^^orardf:instruction
    ```
- Context-dependent VPD constraints may be defined using Secure Application context.
VPD enabled RDF

- Create RDF Model
- Create VPD Policy and Context
  - Define RDF Metadata
  - Define VPD Constraints
  - Implement VPD Context Package
- Apply VPD Policy to an RDF Model
- Query with VPD Access Restrictions

Security Administrator Role

Unprivileged user
VPD enabled RDF

Create RDF Model

Create VPD Policy and Context

Define RDF Metadata

Define VPD Constraints

SQL> -- create a context to conditionally enable/disable constraints
SQL> create context contracts_appctx using sec_admin.contracts_ctxpack;
Context Created.

SQL> -- Create a VPD policy with optional Namespace mapping and policy function.
SQL> begin
2    sem_rdfsa.create_vpd_policy(
3      policy_name   => 'contracts_policy',
4      namespace_map => mdsys.rdf_alias(
5                                   mdsys.rdf_alias('', 'http://www.myorg.com/contract/classes/'),
6                                   mdsys.rdf_alias('prjp', 'http://www.myorg.com/contract/properties/')),
7      context_name  => 'contracts_appctx');
8  end;
9  /

Role

Unprivileged user
VPD enabled RDF

- Create RDF Model
- Create VPD Policy and Context
  - Define RDF Metadata
  - Define VPD Constraints
  - Implement VPD Context Package
- Apply VPD Policy to an RDF Model
- Query with VPD Access Restrictions

Security Administrator Role

Unprivileged user
VPD enabled RDF

- Define RDF Metadata
- Create RDF Model
- Define VPD Constraints
- Create VPD Policy and Context
- Query with VPD Access Restrictions
- Apply VPD Policy to an RDF Model

SQL> -- Define RDF Metadata pertaining to a given model. This includes Domain/Range subClassof and subPropertyOf relationships
SQL> begin
  2    sem_rdfs.maint_vpd_metadata(
  3      policy_name => 'contracts_policy',
  4      t_subject   => '<http://www.myorg.com/contract/classes/Contract>',
  5      t_predicate => '<http://www.w3.org/2000/01/rdf-schema#subClassOf>',
  6      t_object    => '<http://www.myorg.com/contract/classes/Project>');
  7  end;
  8  /
PL/SQL procedure successfully completed.

SQL>

SQL> begin
  2    sem_rdfs.maint_vpd_metadata(
  3      policy_name => 'contracts_policy',
  4      t_subject   => '<http://www.myorg.com/contract/properties/hasValue>',
  5      t_predicate => '<http://www.w3.org/2000/01/rdf-schema#domain>',
  6      t_object    => '<http://www.myorg.com/contract/classes/Contract>');
  7  end;
  8  /
PL/SQL procedure successfully completed.
VPD enabled RDF

- Create RDF Model
- Create VPD Policy and Context
  - Define RDF Metadata
  - Define VPD Constraints
  - Implement VPD Context Package
- Apply VPD Policy to an RDF Model
- Query with VPD Access Restrictions
VPD enabled RDF

Create RDF Model

Create VPD Policy and Context

Define RDF Metadata

Define VPD Constraints

Implement VPD Context Package

SQL> -- Constraint on Project class for users with Project lead role --
SQL> begin
2    sem_rdfs.add_vpd_constraint(
3      policy_name    => 'contracts_policy',
4      constr_name    => 'access_to_project_lead',
5      match_pattern  => '{ ?x rdf:type :Project } ',
6      apply_pattern  => '{ ?x prjp:hasLead
7          "sys_context(''sa$userenv'',''app_user'"^^orardf:instruction}'
8      constr_group  => 'PROJECT_LEAD');
9  end;
10  /
PL/SQL procedure successfully completed.
VPD enabled RDF

Create RDF Model

Create VPD Policy and Context

Define RDF Metadata

Define VPD Constraints

Implement VPD Context Package

SQL> -- Constraint on Project class for users with Project lead role --
SQL> begin
2    sem_rdfs.add_vpd_constraint(
3      policy_name    => 'contracts_policy',
4      constr_name    => 'access_to_project_lead',
5      match_pattern  => ' { ?x rdf:type :Project } ',
6      apply_pattern  => ' { ?x prjp:hasLead
7                  "sys_context(''sa$userenv'',''app_user'"^^orardf:instruction} ,
8      constr_group => 'PROJECT_LEAD');
9  end;
10  /
PL/SQL procedure successfully completed.

SQL> -- A department VP's access to project value --
SQL> begin
2    sem_rdfs.add_vpd_constraint(
3      policy_name    => 'contracts_policy',
4      constr_name    => 'vps_access_to_project_value',
5      match_pattern  => ' { ?proj prjp:hasValue ?val } ',
6      apply_pattern  => ' { ?proj prjp:ownedBy ?dept .
7                  ?dept prjp:hasVP
8                  "sys_context(''sa$userenv'',''app_user'"^^orardf:instruction } ,
9      constr_group => 'VP');
10  end;
11  /
PL/SQL procedure successfully completed.
VPD enabled RDF

Create RDF Model

Create VPD Policy and Context

Define RDF Metadata

Define VPD Constraints

Implement VPD Context Package

Apply VPD Policy to an RDF Model

Query with VPD Access Restrictions

Security Administrator Role

Unprivileged user
VPD enabled RDF

Security Administrator Role

Unprivileged user

Define RDF Metadata

Create RDF Model

Apply VPD Policy to an RDF Model

Define VPD Constraints

Query with VPD Access Restrictions

--- Implement the VPD policy context package to activate appropriate constraint groups based on the application context

---

CREATE OR REPLACE PACKAGE BODY contracts_ctxpack

PROCEDURE set_ctx_at_logon

hrdata HrTable%ROWTYPE;

BEGIN

SELECT * into hrdata FROM HrTable WHERE guid = sys_context('userenv','current_user');
dbms_session.set_context('contracts_appctx', sem_rdfsa.VPD_DEF_CONSTR_ENFORCE, '0');
if (hrdata.emprole = 'VP') then
dbms_session.set_context('contracts_appctx', 'VP', '1');
elsif (hrdata.emprole = 'PROJECT_LEAD') then
dbms_session.set_context('contracts_appctx', 'PROJECT_LEAD', '1');
elsif
else
dbms_session.set_context('contracts_appctx', sem_rdfsa.VPD_DEF_CONSTR_ENFORCE, '1');
end if;
END;

Implement VPD Context Package
VPD enabled RDF

1. Implement VPD Context Package
2. Create VPD Policy and Context
3. Define RDF Metadata
4. Define VPD Constraints
5. Implement VPD Context Package
6. Apply VPD Policy to an RDF Model
7. Query with VPD Access Restrictions

Security Administrator Role

Unprivileged user
Implement VPD Context Package

VPD enabled RDF

Create RDF Model

Create VPD Policy and Context

Implement VPD Context Package

Create VPD Policy and Context

Apply VPD Policy to an RDF Model

Query with VPD Access Restrictions

SQL> -- Apply the VPD policy to an existing Model to enforce VPD policies
SQL> -- for queries involving the model --
SQL> begin
2     sem_rdfs_app_vpd_policy(
3       policy_name  =>'contracts_policy',
4       model_name   =>'contracts');
5  end;
6  /
PL/SQL procedure successfully completed.

Security Administrator Role

Unprivileged user
VPD enabled RDF

Create RDF Model

Create VPD Policy and Context

Define RDF Metadata

Define VPD Constraints

Implement VPD Context Package

Apply VPD Policy to an RDF Model

Query with VPD Access Restrictions

Security Administrator Role

Unprivileged user
SQL> -- set some dummy context based on the user logged in --
SQL> connect susan/i_am_a_vp
Connected.
SQL> select prj, status, val
2  from table(sem_match('sparql{ ?prj pred:hasStatus ?status . OPTIONAL { ?prj pred:hasValue ?val } }',
3            sem_models('contracts'),null, ...));

<table>
<thead>
<tr>
<th>PRJ</th>
<th>STATUS</th>
<th>VAL</th>
</tr>
</thead>
</table>

SQL> connect andy/i_am_a_project_lead
Connected.
SQL> select prj, status, val
2  from table(sem_match('sparql{ ?prj pred:hasStatus ?status . OPTIONAL { ?prj pred:hasValue ?val } }',
3            sem_models('contracts'),null, ...));

<table>
<thead>
<tr>
<th>PRJ</th>
<th>STATUS</th>
<th>VAL</th>
</tr>
</thead>
</table>

SQL> connect bob/i_am_a_team_member
Connected.
SQL> select prj, status, val
2  from table(sem_match('sparql{ ?prj pred:hasStatus ?status . OPTIONAL { ?prj pred:hasValue ?val } }',
3            sem_models('contracts'),null, ...));

<table>
<thead>
<tr>
<th>PRJ</th>
<th>STATUS</th>
<th>VAL</th>
</tr>
</thead>
</table>
Data Classification Labels for Access Control

Label Components:

• Levels – Determine the vertical sensitivity of data and the highest classification level a user can access.

• Compartments – Facilitate compartmentalization of data. Users need exclusive membership to a compartment to access its data.

• Groups – Allow hierarchical categorization of data. A user with authorization to a parent group can access data in any of its child groups.

CONF : NAVY, MILITARY : NY, ON → Data Label matches User Access Label ↓

HIGHCONF : MILITARY, NAVY, SPCLOPS : US, UK
Oracle Label Security for RDF Data

- Oracle Label Security enforces security at the “row” level.
  - Attempts to access key or non-key columns of a specific row are validated using row/data sensitivity labels.

<table>
<thead>
<tr>
<th>ContractID</th>
<th>Organization</th>
<th>ContractValue</th>
<th>RowLabel</th>
</tr>
</thead>
<tbody>
<tr>
<td>ProjectHLS</td>
<td>N. America</td>
<td>1000000</td>
<td>SE:HLS:US</td>
</tr>
</tbody>
</table>

- Notion of “row” does not exist for RDF data
  - Conceptually, a relational table row maps to a set of triples and the sensitivity label applies to the complete set.
Securing triples with sensitivity labels

- The user’s read access label must cover (= or >) the triple’s sensitivity label for the user to read the triple.
- Triples describing a single resource may employ different sensitivity labels for greater control.
Securing triples with sensitivity labels

- The user’s read access label must cover (= or >) the triple’s sensitivity label for the user to read the triple.
- Triples describing a single resource may employ different sensitivity labels for greater control.
Securing RDF Data – Write Access

- **Securing Subjects**: Sensitivity label associated with a *Subject* determines the minimum access label for all the triples with the subject.

  Subject: `<http://../contract/projectHLS>` \( \text{SE:HLS:US} \)

- **Securing Predicates**: Sensitivity label associated with a *Predicate* determines the minimum access label for the triples involving the predicate type

  Predicate: `hasContractValue` \( \text{TS:FIN:} \)

- **Securing Objects**: Sensitivity label associated with a resource (URI) determines the minimum access label for the triples using this resource as *Object*.

  Resource: `<http://../contractType/Govt>` \( \text{SE::} \)
  
  Triple: `<.../projectHLS>` `isA`  
  `<.../contractType/Govt>`
Policy enforcement for Insert operations

Attempt to Insert a Triple

Secure Subject Option? yes

Subject’s Sen. Label compatible ? yes

Secure Predicate Option? yes

Predicate’s Sen. Label compatible ? yes

Object a Resource (URI)? yes

Secure Object Option? yes

Object’s Sen. Label compatible ? no

Record Triple into RDF Storage

If the RDF Resource has a sensitivity label, ensure that the user has write access to the label

Error: Unauthorized Operation!!
Policy enforcement for Insert operations

Attempt to Insert a Triple

- Secure Subject Option? yes
  - Subject’s Sen. Label compatible? yes
    - Record Triple into RDF Storage
  - no
- Secure Predicate Option? yes
  - Predicate’s Sen. Label compatible? yes
    - Record Triple into RDF Storage
  - no
- Object a Resource (URI)? yes
  - Secure Object Option? yes
    - Object’s Sen. Label compatible? yes
      - Record Triple into RDF Storage
  - no
- no

If the RDF Resource has a sensitivity label, ensure that the user has write access to the label.

Sensitivity label for the new Triple is set to User’s Initial Row Label or it may be generated using a Label function.

Error: Unauthorized Operation!!
**OLS enabled RDF**

- Create RDF Network
- Create OLS Policy

Apply OLS Policy to RDF Data with appropriate options

Create RDF Model

Add Subjects and/or Predicates with Minimum Sensitivity labels

Add Triples with labels that at least cover all their components’ labels

Optional

SYSDBA Role

Security Administrator Role

Unprivileged user
OLS enabled RDF

Create RDF Network

Create OLS Policy

SQL> EXECUTE SA_SYSDBA.CREATE_POLICY('defense','ctxt1');

SQL> EXECUTE SA_COMPONENTS.CREATE_LEVEL('defense',500,'TS','TOP_SECRET');
SQL> EXECUTE SA_COMPONENTS.CREATE_LEVEL('defense',300,'SE','SECRET');
SQL> EXECUTE SA_COMPONENTS.CREATE_LEVEL('defense',100,'UN','UNCLASSIFIED');

SQL> EXECUTE SA_COMPONENTS.CREATE_COMPARTMENT('defense',100,'US_SPCL','United States Special');
SQL> EXECUTE SA_COMPONENTS.CREATE_COMPARTMENT('defense',200,'US_ONLY','United States Only ');

SQL> EXECUTE SA_COMPONENTS.CREATE_GROUP('defense',10,'COAL','Coalition');
SQL> EXECUTE SA_COMPONENTS.CREATE_GROUP('defense',11,'US','United States','COAL');

SQL> EXECUTE SA_LABEL_ADMIN.CREATE_LABEL('defense',1100,'UN:US_SPCL:');
SQL> EXECUTE SA_LABEL_ADMIN.CREATE_LABEL('defense',2100,'SE:US_SPCL:');

SQL> EXECUTE SA_USER_ADMIN.SET_USER_LABELS('defense','A','SE:US_SPCL:US,CA');
SQL> EXECUTE SA_USER_ADMIN.SET_USER_LABELS('defense','B','TS:US_SPCL');
SQL> EXECUTE SA_USER_ADMIN.SET_USER_LABELS('defense','C','TS:US_SPCL,US_ONLY:');

SYSDBA Role

Security Administrator Role

Unprivileged user
OLS enabled RDF

Create RDF Network

Create OLS Policy

Apply OLS Policy to RDF Data with appropriate options

Create RDF Model

Add Subjects and/or Predicates with Minimum Sensitivity labels

Add Triples with labels that at least cover all their components’ labels

SYSDBA Role

Security Administrator Role

Unprivileged user
OLS enabled RDF

Create RDF Network

Create OLS Policy

Apply OLS Policy to RDF Data with appropriate options

Create RDF Model

SYSDBA Role

SQL>
SQL> -- Apply the OLS policy to the RDF repository with default
SQL> -- table_options = ALL_CONTROL and NULL label_function.
SQL>
SQL> begin
2    sem_rdfsa.apply_ols_policy (  
3        policy_name   => 'defense',
4        rdfsa_options => sem_rdfsa.SECURE_SUBJECT+
5                         sem_rdfsa.SECURE_PREDICATE);
6  end;
7  /

PL/SQL procedure successfully completed.

SQL> -- Additional options for RDF Secure Access include
SQL> -- sem_rdfsa.SECURE_OBJECT to secure Objects
SQL> -- sem_rdfsa.OPT_DEFINE_BEFORE_USE which forces the user to assign
SQL> -- a minimum sensitivity label to Subject and/or Predicate
SQL> -- before they are used in a Triple
OLS enabled RDF

Create RDF Network

Create OLS Policy

Apply OLS Policy to RDF Data with appropriate options

Create RDF Model

Add Subjects and/or Predicates with Minimum Sensitivity labels

Add Triples with labels that at least cover all their components' labels

Optional

SYSDBA Role

Security Administrator Role

Unprivileged user
OLS enabled RDF

Create RDF Network

Create OLS Policy

Apply OLS Policy to RDF Data with appropriate options

Create RDF Model

SQL>
--- Create application table
create table contracts_rdf_data (id NUMBER,
  triple SDO_RDF_TRIPLE_S);

SQL>
--- Create the model
begin
  sem/apis.create_rdf_model (model_name => 'contracts',
    table_name => 'contracts_rdf_data',
    column_name => 'triple');
end;
/
SQL>
OLS enabled RDF

- Create RDF Network
- Create OLS Policy

- Apply OLS Policy to RDF Data with appropriate options

- Create RDF Model
- Add Subjects and/or Predicates with Minimum Sensitivity labels
- Add Triples with labels that at least cover all their components’ labels

Roles:
- SYSDBA Role
- Security Administrator Role
- Unprivileged user
SQL> select sys_context('sa$defense_x', 'initial_row_label') as initial_label from dual;

INITIAL_LABEL
-------------------------
SE:US_SPCL:US,CA
SQL>
SQL> begin
  2   sem_rdfsa.set_resource_label(
  3       model_name => 'contracts',
  4       subject => 'http://www.myorg.com/contract/projectHLS',
  5       label_string => 'SE:US_SPCL:US');
  6   end;
  7 / 
SQL>

Create RDF Network
Create OLS Policy

Apply OLS Policy to RDF Data
with appropriate options

Create RDF Model

Add Subjects and/or Predicates with Minimum Sensitivity labels

Add Triples with labels that at least cover all their components’ labels

SYSDBA Role
Security Administrator Role
Unprivileged user
SQL> select sys_context('sa$defense_x','initial_row_label') as initial_label from dual;

INITIAL_LABEL
-------------------------
SE:US_SPCL:US,CA

SQL> begin
  2    sem_rdfsa.set_resource_label
  3         model_name   => 'contracts',
  4         subject      => 'http://www.myorg.com/contract/projectHLS',
  5         label_string => 'SE:US_SPCL:US'
  6    end;
    end;
  7  /

SQL>

SQL> -- user's MIN Write and MAX Write are enforced
SQL> begin
  2    sem_rdfsa.set_predicate_label
  3         model_name   => 'contracts',
  4         predicate    => 'http://www.myorg.com/pred/hasValue',
  5         label_string => 'TS:US_SPCL')
  6  end;
  7  /

begin *
ERROR at line 1:
ORA-55359: unauthorized operation with policy DEFENSE - 28115

Create RDF Model

Add Subjects and/or Predicates with Minimum Sensitivity labels

Add Triples with labels that at least cover all their components’ labels

SYSDBA Role

Security Administrator Role

Unprivileged user

Optional
Add Subjects and/or Predicates with Minimum Sensitivity labels

Optional

Add Triples with labels that at least cover all their components’ labels

SYSDBA Role

Security Administrator Role

Unprivileged user
OLS enabled RDF

Create RDF Network

Create OLS Policy

Apply OLS Policy to RDF Data with appropriate options

Create RDF Model

Add Subjects and/or Predicates with Minimum Sensitivity labels

Optional

Add Triples with labels that at least cover all their components’ labels

SYSDBA Role

Security Administrator Role

Unprivileged user
SQL> select sys_context('sa$defense_x','initial_row_label') as initial_label ..

INITIAL_LABEL
-------------------------
SE:US_SPCL:US,CA
SQL>
SQL> insert into a.contracts_rdf_data values (1,  
  2  sdo_rdf_triple_s ('contracts','http://www.myorg.com/contract/projectHLS',  
  3    'http://www.myorg.com/pred/drivenBy',  
  4    'http://www.myorg.com/department/Dept1'));
SQL>
SQL> insert into a.contracts_rdf_data values (2,  
  2  sdo_rdf_triple_s ('contracts','http://www.myorg.com/contract/projectHLS',  
  3    'http://www.myorg.com/pred/hasValue',  
  4    "100000"^^xsd:integer));

ERROR at line 2:
ORA-55359: unauthorized operation with policy DEFENSE
Add Subjects and/or Predicates with Minimum Sensitivity labels

Add Triples with labels that at least cover all their components’ labels

SYSDBA Role

Security Administrator Role

Unprivileged user
OLS enabled RDF

Create RDF Network
Create OLS Policy
Apply OLS Policy to RDF Data
with appropriate options
Create RDF Model
Add Subjects and/or Predicates
with Minimum Sensitivity labels
Add Triples with labels that at least
cover all their components’ labels

SQL> select sys_context('sa$defense_x','initial_row_label') as initial_label..
INITIAL_LABEL
-------------------------
TS:US_SPCL:US,CA
SQL>
SQL> insert into a.contracts_rdf_data values (1, 2    sdo_rdf_triple_s ('contracts','http://www.myorg.com/contract/projectHLS', 3      'http://www.myorg.com/pred/drivenBy', 4      'http://www.myorg.com/department/Dept1'));

SQL> select sys_context('sa$defense_x','initial_row_label') as initial_label..
INITIAL_LABEL
-------------------------
TS:US_SPCL:US,CA
SQL>
SQL> insert into a.contracts_rdf_data values (2, 2    sdo_rdf_triple_s ('contracts','http://www.myorg.com/contract/projectHLS', 3      'http://www.myorg.com/pred/hasValue', 4      '"100000"^^xsd:integer'));

SQL> insert into a.contracts_rdf_data values (3, 2    sdo_rdf_triple_s ('contracts','http://www.myorg.com/contract/projectHLS', 3      'http://www.myorg.com/pred/hasValue', 4      '"100000"^^xsd:integer'));

SQL> commit;
RDF and OLS : Query

SQL> select sys_context('sa$defense_x','initial_row_label') as initial_label from dual;

INITIAL_LABEL
-------------------------
SE:US_SPCL:US,CA
SQL>

SQL> select s, p from TABLE(SEM_MATCH('{ ?s ?p ?o }',
  2          SEM_Models('Contracts'), null, null, null));

S                                P
---------------------------------------- ----------------------------------------
SQL>

SQL> connect e/e
Connected.

SQL> select sys_context('sa$defense_x','initial_row_label') as initial_label from dual;

INITIAL_LABEL
-------------------------
TS:US_SPCL,FIN:US,CA
SQL>

SQL> select s, p from TABLE(SEM_MATCH(' { ?s ?p ?o }',
  2          SEM_Models('Contracts'), null, null, null));

S                                P
---------------------------------------- ----------------------------------------
http://www.myorg.com/contract/projectHLS http://www.myorg.com/pred/hasValue
SQL>
Label Generation for Inferred Triple

• Predefined Label Generators:
  – Use Subject Label: Use subject’s label as the label for the triple.
  – Use Predicate Label: Use predicate’s label as the label for the triple.
  – Use Object Label: Use the label associated with the resource in the object position as the label for the inferred triple.
  – Use Rule Label: Use the label associated with the (last) rule inferring the triple as the label for the triple.
  – Use Dominating Label: Use all available component labels to determine the dominating one and assign it as the label for the triple.

• Custom Label Generators using Extensibility:
  – Use Component & Rule Labels: Use custom logic with Subject, Predicate, Object, and Rule labels to generate a label for the triple.
  – Use Antecedents’ Labels: Conflict resolution based on the labels for antecedent triples generating the current triple.
Semantic Indexing for Documents
Semantic Indexing for Documents

- Links people – places – things – events to documents stored in Oracle Database through a semantic index.
- Supports popular natural language processing tools that extract semantic metadata from documents, such as OpenCalais from Thomson Reuters and GATE.
- Provides a SQL interface for SPARQL queries to find documents of interest.
- Ability to combine Domain Ontologies with extracted metadata to facilitate Ontology-Assisted searches.
- Extends the power of Oracle Database to include semantic matching in cross-domain queries.
Semantic Indexing for Documents

- Create a semantic index on documents stored in Oracle Database using third-party entity extractors.
  - Out-of-the-box support for OpenCalais from Thomson Reuters & General Architecture for Text Engineering (GATE).
  - Extensible framework to plug-in any information extractor.
- Embed SPARQL graph pattern queries in SQL to find documents of interest and return relevant information.
  E.g., Find documents that refer to some financial business organizations in the context of Bailout.

```sql
SELECT docId, SEM_CONTAINS_SELECT(1) result FROM Newsfeed
WHERE SEM_CONTAINS (article,
  '{ ?org rdf:type c:Organization .
    ?org pred:categoryName c:BusinessFinance .
    ?event pred:event c:Bailout’ ) = 1
```
Semantic Search - Key Components

- **Extensible Information Extractor**
  - Programmable API to plug-in 3rd party extractors into the database.

- **SEM_CONTAINS Operator**
  - To identify documents of interest based on their extracted metadata, using standard SQL queries.

- **SEM_CONTAINS_SELECT Ancillary Operator**
  - To return relevant information about the documents matched using SEM_CONTAINS operator.

- **SemContext Index type**
  - Index type that interacts with the extractor to manage the metadata extracted from the documents efficiently and facilitates semantic search.
Indiana authorities filed felony charges and a court issued an arrest warrant for a financial manager who apparently tried to fake his death by crashing his airplane in a Florida swamp. Marcus, 38, remained on the lam Tuesday afternoon, two days after authorities say he staged his disappearance and then rode out of a small Alabama town on a red motorcycle under cover of darkness.
Indiana authorities filed felony charges and a court issued an arrest warrant for a financial manager who apparently tried to fake his death by crashing his airplane in a Florida swamp. Marcus, 38, remained on the lam Tuesday afternoon, two days after authorities say he staged his disappearance and then rode out of a small Alabama town on a red motorcycle under cover of darkness.

**RDF/XML Output**

```
<rdf:RDF>
  <rdf:Description rdf:about="http://../Marcus">
    <p:hasName .. >Marcus</c:name>
    <p:hasAge .. >38</c:age>
    <p:hasGender .. >Male</c:gender>
  </rdf:Description>
  <rdf:Description rdf:about="http://../FloridaSwamp">
    <c:hasLocation>Florida</c:location> …
  </rdf:Description>
</rdf:RDF>
```
Matching Documents with SPARQL

Find all articles that match the SPARQL pattern

```
SELECT docId FROM Newsfeed
WHERE SEM_CONTAINS (Articles,
    { ?org  rdf:type           c:Organization .
    ?org  pred:hasName       "Acme Corp"^^xsd:string .
    ?org  pred:categoryName   c:BusinessFinance
    ... }
)
Matching Documents with SPARQL

Find all articles that match the SPARQL pattern

```
SELECT docId FROM Newsfeed
WHERE SEM_CONTAINS (Articles,
{ ?org rdf:type c:Organization .
  ?org pred:hasName "Acme Corp"^^xsd:string .
  ?org pred:categoryName c:BusinessFinance
  ... })
```
Returning bindings for matched documents

• Information returned about each matching document can be used to rank the results.

```
SELECT docId, SEM_CONTAINS_SELECT(1) FROM Newsfeed
WHERE SEM_CONTAINS (Articles,
  'SELECT ?org  ?score
   WHERE { ?org  rdf:type            c:Organization .
           ?org  pred:categoryName   c:BusinessFinance .
           OPTIONAL { ?comp  p:score  ?score } }', 1) = 1
```

• Bindings from the matching documents are returned in SPARQL Query result XML format.

```
<result>
  <binding name="ORG">
    <uri>http://newscorp.com/Organization/AcmeCorp</uri>
  </binding>
  <binding name="SCORE">
    <literal>0.952</literal>
  </binding>
</result>
```
CREATE INDEX ArticleIndex 
ON NewsFeed (Article) 
INDEXTYPE IS SemContext 
PARAMETERS ('gate_nlp')

SELECT docId FROM Newsfeed 
WHERE SEM_CONTAINS (Articles, 
   '{ ?s ?p "Acme.." }') = 1
CREATE INDEX ArticleIndex
ON NewsFeed (Article)
INDEXTYPE IS SemContext
PARAMETERS ('gate_nlp')

<table>
<thead>
<tr>
<th>DocId</th>
<th>Article</th>
<th>Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Indiana authorities filed felony charges and a court issued an arrest warrant for a financial ...</td>
<td>CNN</td>
</tr>
<tr>
<td>2</td>
<td>Major dealers and investors in over-the-counter derivatives agreed to report all credit-default ...</td>
<td>NW</td>
</tr>
</tbody>
</table>

SELECT docId FROM Newsfeed
WHERE SEM_CONTAINS (Articles, '{ ?s ?p "Acme.." }') = 1
    AND Source = 'CNN'

**SemContext Index**

**SemContext index on Article column**

<table>
<thead>
<tr>
<th>Subject</th>
<th>Property</th>
<th>Object</th>
<th>rid</th>
</tr>
</thead>
<tbody>
<tr>
<td>p:Marcus</td>
<td>rdf:type</td>
<td>rc::Person</td>
<td>r1</td>
</tr>
<tr>
<td>p:Marcus</td>
<td>:hasName</td>
<td>&quot;Marcus&quot;^^..</td>
<td>r1</td>
</tr>
<tr>
<td>p:Marcus</td>
<td>:hasAge</td>
<td>&quot;38&quot;^^xsd:..</td>
<td>r1</td>
</tr>
</tbody>
</table>

**Triples table with rowid references**

**Newsfeed table**

SELECT docId FROM Newsfeed
WHERE SEM_CONTAINS (Articles, '{ ?s ?p "Acme.." }') = 1
    AND Source = 'CNN'

**ORACLE**
CREATE INDEX ArticleIndex
ON NewsFeed (Article)
INDEXTYPE IS SemContext
PARAMETERS (‘gate_nlp’)

SELECT docId FROM Newsfeed
WHERE SEM_CONTAINS (Articles,
   '{ ?s ?p "Acme.." }') = 1
   AND Source = ‘CNN’
SemContext Index – Additional Capabilities

- Ability to combine Domain Ontologies with extracted metadata to facilitate Ontology-Assisted document search.

```sql
SELECT docId FROM Newsfeed
WHERE SEM_CONTAINS (Articles,
  `{ ?comp rdf:type c:Organization .
    ?comp pred:categoryName c:BusinessFinance .
    ?city geo:state "NY"^^xsd:string }`) = 1
```

- Ability to edit extracted metadata for improved search results.
  - Allows combining triples extracted by multiple extraction tools.
  - Allows extension of the knowledge base with community feedback.

- Ability to infer additional information from the extracted metadata through Inference.*

* Only Cross-Document Inference supported in 11.2
Choosing an Extractor at Query time

- The SemContext index can be created with multiple Extractor policies so that a specific extractor can be chosen at query time.

```sql
CREATE INDEX ArticleIndex ON Newsfeed (Article)
  INDEXTYPE IS mdsys.SemContext
  PARAMETERS ("gatenlp_policy iKnow_policy Calais_plus_geo")
```

- The exact policy to be used for a query is specified as an optional argument to the SEM_CONTAINS operator.

```sql
SELECT docId FROM Newsfeed
WHERE SEM_CONTAINS (Articles,
  "\{ ?comp rdf:type c:Company .
    ?comp p:categoryName c:BusinessFinance }",
  "iKnow_policy") = 1
```
Change Management for Collaboration
Change Management for Semantic Data

• Manage public and private versions of semantic data in database workspaces

• Collaborate with multi-user workspaces

• Isolate a group of changes in a workspace
  – Data in multiple valid states: e.g., Current – Planned - History
  – Publish all, some or none of the changes made in a workspace

• Create multiple scenarios in different workspaces
  – “What if” analysis
  – Allow multiple application testers to use the same data set

• Provides efficient data storage and querying
  – New versions created only for changed data
  – Queries are workspace-specific
  – Uses the Workspace Manager feature of Oracle Database
Oracle Semantic Technologies Versioning

• An RDF graph may be logically versioned.
  – No duplication of data and all versions of data use a common physical storage.
  – Triples added/deleted in a workspace are private to the workspace until the workspace is merged.
  – Ability to detect conflicts with application logic.

• Physically versioned inferred data
  – Each workspace shares the inferred data with its child workspace until some data modifications are made in the child workspace.
  – Each workspace with some workspace-private changes maintains a private copy of the inferred data.

• SEM_MATCH queries on version-enabled RDF graphs are version aware.
Versioning RDF Data

Go To LIVE Workspace

Optional

Create a Child Workspace W1

SEM_MATCH Query

Go To W1 Workspace

Modify Data in the RDF Model

SEM_MATCH Query

Recreate any Invalid Entailments
Versioning RDF Data

1. Go To LIVE Workspace (Optional)
2. Create a Child Workspace W1
3. SEM_MATCH Query

4. Go To W1 Workspace
5. Modify Data in the RDF Model
6. SEM_MATCH Query
7. Recreate any Invalid Entailments
Versioning RDF Data

Go To LIVE Workspace

Optional

Create a Child Workspace W1

SEM_MATCH Query

SQL>
SQL> -- Go to Live workspace. Also the default for a new
SQL> -- database session --
SQL>
SQL> begin
2    dbms_wm.gotoWorkspace(
3                  workspace    => 'LIVE');
4  end;
5  /

Go To W1 Workspace

Modify Data in the RDF Model

SEM_MATCH Query

Recreate any Invalid Entailments
Versioning RDF Data

Go To LIVE Workspace

Create a Child Workspace W1

SEM_MATCH Query

Go To LIVE Workspace

Go To W1 Workspace

Modify Data in the RDF Model

SEM_MATCH Query

Recreate any Invalid Entailments
Versioning RDF Data

Go To LIVE Workspace

Optional

Create a Child Workspace W1

Go To W1 Workspace

Go To LIVE Workspace

Go To W1 Workspace

Modify Data in the RDF Model

SEM_MATCH Query

Recreate any Invalid Entailments

SQL>
SQL>  -- Create a child workspace to maintain some RDF data
SQL>  -- modifications private to that workspace.
SQL>
SQL> begin
  2    dbms_wm.createWorkspace(
  3       workspace => 'W1',
  4       description => 'Modify projectHLS data');
  4    end;
  5 /

Create a Child Workspace W1
Versioning RDF Data

Go To LIVE Workspace

Create a Child Workspace W1

SEM_MATCH Query

Optional

Go To W1 Workspace

Modify Data in the RDF Model

SEM_MATCH Query

Recreate any Invalid Entailments
Versioning RDF Data

Go To LIVE Workspace

Create a Child Workspace W1

Optional

Go To W1 Workspace

Modify Data in the RDF Model

SEM_MATCH Query

Recreate any Invalid Entailments

SQL>
SQL> -- Go to W1 workspace in a different session.
SQL>
SQL> begin
  2        dbms_wm.gotoWorkspace(
  3            workspace => 'W1');
  4    end;
  5 /

ORACLE
Versioning RDF Data

- Go To LIVE Workspace
- Create a Child Workspace W1
- Optional
- SEM_MATCH Query

Go To W1 Workspace
- Modify Data in the RDF Model
- SEM_MATCH Query
- Recreate any Invalid Entailments
Create a Child Workspace W1
Go To LIVE Workspace
Optional
Go To W1 Workspace
Modify Data in the RDF Model

SEM_MATCH Query

SQL> select proj, dept from TABLE(SEM_MATCH( {?proj ctr:drivenBy ?dept}, ...));

PROJ                                        DEPT
------------------------------------------- ------------------------------------------
...

SQL> -- Update a triple to re-assign the project to a different department
SQL> update contracts_rdf_data set triple = sdo_rdf_triple_s ('contracts',
2 'http://www.myorg.com/contract/projectHLS>',
3 'http://www.myorg.com/contract/properties/drivenBy>',
4 'http://www.myorg.com/department/Dept9')) where id = 1;

Recreate any Invalid Entailments

Modify Data in the RDF Model

SEM_MATCH Query
Versioning RDF Data

Go To LIVE Workspace

Create a Child Workspace W1

SEM_MATCH Query

Optional

Go To W1 Workspace

Modify Data in the RDF Model

SEM_MATCH Query

Recreate any Invalid Entailments
Versioning RDF Data

SQL> -- Live workspace does not show the effects of workspace W1 --
SQL> select proj, dept
2    from TABLE(SEM_MATCH( {?proj ctr:drivenBy ?dept},
3             SDO_RDF_Models('Contracts'), null, null, null));

<table>
<thead>
<tr>
<th>PROJ</th>
<th>DEPT</th>
</tr>
</thead>
</table>

Recreate any Invalid Entailments

SQL> -- Workspace W1 maintains the updated version --
SQL> select proj, dept
2    from TABLE(SEM_MATCH( {?proj ctr:drivenBy ?dept}, ...));

<table>
<thead>
<tr>
<th>PROJ</th>
<th>DEPT</th>
</tr>
</thead>
</table>
Versioning RDF Data

Go To LIVE Workspace

Optional

Create a Child Workspace W1

SEM_MATCH Query

Go To W1 Workspace

Modify Data in the RDF Model

SEM_MATCH Query

Recreate any Invalid Entailments
SQL> -- Data modifications within a workspace leave the index segment invalid/incomplete
SQL> -- while the same index segment in other workspaces may be valid.

SQL>

SQL> select index_name, index_view_name, status from mdsys.RDF_RULES_INDEX_INFO;

<table>
<thead>
<tr>
<th>INDEX_NAME</th>
<th>INDEX_VIEW_NAME</th>
<th>STATUS</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONTRACTS_RDFS</td>
<td>RDFI_CONTRACTS_RDFS</td>
<td>INVALID</td>
</tr>
</tbody>
</table>

SQL> begin
   -- recreate the index with workspace private data --
   sem_apis.create_entailment(
      index_name_in => 'contracts_rdfs',
      models_in     => SDO_RDF_Models('contracts'),
      rulebases_in  => SDO_RDF_Rulebases('RDFS'));
end;
/

SQL> select index_name, index_view_name, status from mdsys.RDF_RULES_INDEX_INFO;

<table>
<thead>
<tr>
<th>INDEX_NAME</th>
<th>INDEX_VIEW_NAME</th>
<th>STATUS</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONTRACTS_RDFS</td>
<td>RDFI_CONTRACTS_RDFS</td>
<td>VALID</td>
</tr>
</tbody>
</table>

Recreate any Invalid Entailments
Workspace Manager Operations for RDF

- **Application Table/Model**: EnableVersioning, DisableVersioning, merge, refresh, rollback
- **Workspace**: create, goto, refresh, merge, rollback, remove, compress, alter, events
- **Savepoints**: create, alter, goto, rollback
- **Privileges**: access, create, delete, rollback, merge
- **Access Modes**: read, 1 writer, WS operations, none
- **Locks**: exclusive and shared on the application table rows
- **Differences**: compares savepoints and workspaces
- **Detect / Resolve Conflicts**: Conflicts detected based on application table primary keys with the ability to choose the version to merge
Summary

- Requirement to search and derive greater business knowledge from enterprise databases and applications
- Oracle database 11g is the leading commercial database with native RDF/OWL data management
- Scalable & secure platform for wide-range of semantic applications
- Readily scales to ultra-large repositories (+10 billion)
- Choice of SQL or SPARQL query
- Leverages Oracle Partitioning, RAC supported
- Growing ecosystem of 3rd party tools partners
For More Information

search.oracle.com

Semantic Technologies

or

oracle.com
ORACLE IS THE INFORMATION COMPANY