experience

OPENWORLD

November 11–15, 2007
Why, When, and How to Use Oracle Database 11g Semantic Technologies

Xavier Lopez, Ph.D, Director, Oracle Server Technologies
Souripriya Das, Ph.D, Consultant Member, Oracle Server Technologies
Semantic Data Management 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
- Overcome isolated systems design
- Built on open, industry standards:
  - SQL, XML, RDF, OWL, SPARQL
Why is this Useful?

- Designed to represent knowledge in a distributed world
- A method to decompose knowledge into small pieces, with rules about the semantics of those pieces
- RDF data is self-describing; it “means” something
- Allows you to model and integrate DBMS schemas
- Allows you to integrate data from different sources without custom programming
- Supports decentralized data management
- Infer implicit relationships across data
Application Integration

User

Query & results

RDF/OWL

Data Ontologies

Structured & Unstructured Data Sources
Areas of Applicability

- Intelligence, Law Enforcement:
  - Threat analysis, asset tracking, integrated justice
- Integrated BioInformatics:
  - Bio-Pathway analysis, protein interaction
- Finance
  - Fraud detection, Compliance Management
- Web and Social Network Solutions
  - Recommender, Social Network Analysis, Activity Analysis
- Enterprise Business Applications
  - Grid resource mgmt, EII, BI, Configuration mgmt.
Common Information Model (CIM)
A Federation of Ontologies
Analytic Intelligence

Context-Specific Access to Knowledge
Analytic Intelligence

Context-Specific Access to Knowledge

Diagram depicting relationships between Event, Person, Group, perpetrator, hasMember, alliedWith, hasAlias, and hasMember.
Analytic Intelligence

Context-Specific Access to Knowledge

W3C RDF & OWL standards
Knowledge Mining Workflows

Web Resources

News, Email, RSS

Content Mgmt. Systems

Information Extraction
Categorization, Feature/term Extraction

Processed Document Collection

Knowledge Mining & Analysis
- Text Indexing using Oracle Text
- Non-Obvious Relationship Discovery
- Pattern Discovery
- Text Mining
- Faceted Search

RDF/OWL

Ontology Engineering Modeling Process

Domain Specific Knowledge Base

Analyst

Browsing, Presentation, Reporting, Visualization, Query

SQL/SPARQL Query

Explore
Semantic Data Management Workflow

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

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

**Applications & Analysis**
- Graph Visualization
- Link Analysis
- Statistical Analysis
- Faceted Search
- Pattern Discovery
- Text Mining
Knowledge Management Platform
Analyzing Patterns, Trends & Relationships

Customer Solutions
- National Security
- Financial Risk Analysis
- Regulatory Compliance
- Life Sciences Drug Discovery
- Manufacturing Configuration Mgmt
- Health Sciences BioSurveillance

PRESENTATION LAYER
Graph, Timeline, Metadata, Spatial
- Entity Matching 
- Categorization 
- Non-obvious Relationship Analysis 
- Search and Text Indexing
- SPARQL/SQL APIs
- Faceted Search & Discovery
- Graph Analysis
- Link Analysis
- Centrality
- Inferencing (RDFS, OWL)

INFORMATION EXTRACTION
Entity Extraction, Categorization, ETL, Ontology Engineering

Oracle 11g
RDF/OWL Knowledge Base

Structured DBMS, Unstructured, Spatial, RSS, email, Documents

Partner Technologies
Oracle Technologies
DB Semantic Technology in Oracle 11g
Technical Overview
Semantic Technology: Building Blocks

• Representation
  • RDF (Resource Description Framework)
  • Vocabularies
    • RDFS (RDF Schema Language)
    • OWL (Web Ontology Language)

• Inference
  • Implicit rules that capture semantics of each vocabulary
  • RDF, RDFS, OWL-Lite, OWL-DL, OWL-Full

• Query
  • Using graph-patterns in languages such as SPARQL
Semantic Technology Stack

User Interface & Applications

Trust

Proof

Unifying Logic

Query: SPARQL

Ontology: OWL

Rule: RIF

RDFS

Data interchange: RDF

Crypto

XML

URI/IRI

Standards based

http://www.w3.org/2007/03/layerCake.svg
RDF

- Originally created to encode metadata such as ‘author’, ‘date’, etc. for web resources.
- Recently, it has become popular to relate things in the real-world such as people, places, concepts etc.
- The basic unit of information (fact) is represented as `<subject, predicate, object>` triple
- Triples together form a graph, connecting pieces of data
Vocabularies (RDFS and OWL)

• RDFS (RDF Schema)
  • Structuring of resources and properties
    • rdfs:class → Class of resources
    • rdfs:subClassOf → hierarchy of classes
    • rdfs:subPropertyOf → hierarchy of properties

• OWL (Web Ontology Language)
  • Builds on RDF(S) …
    • Property Characteristics: transitivity, symmetry, functional, inverse functional, inverse
    • Class construction via set operations and property restrictions
  • Separate layers have been defined balancing expressibility vs. implementability: OWL Lite, OWL DL, OWL Full
RDF(S) Example

**Asserted Facts**  <http://example.org/emp>

<table>
<thead>
<tr>
<th>:OracleHQemployee</th>
<th>rdfs:subClassOf</th>
<th>:SWcompanyEmployee</th>
</tr>
</thead>
<tbody>
<tr>
<td>:OracleHQemployee</td>
<td>:corpOfficeLoc</td>
<td>:California</td>
</tr>
<tr>
<td>:John</td>
<td>rdf:type</td>
<td>:Oracle HQ Employee</td>
</tr>
</tbody>
</table>

**Derived Facts**

<table>
<thead>
<tr>
<th>:John</th>
<th>rdf:type</th>
<th>:SWcompanyEmployee</th>
</tr>
</thead>
</table>
Graph Pattern Based Query (SPARQL)

```
SELECT x, y
FROM <http://example.org/emp>
WHERE { ?x rdf:type ?y }
```

<table>
<thead>
<tr>
<th>x</th>
<th>y</th>
</tr>
</thead>
<tbody>
<tr>
<td>:John</td>
<td>:OracleHQEmployee</td>
</tr>
<tr>
<td>:John</td>
<td>:SWcompanyEmployee</td>
</tr>
<tr>
<td>:California</td>
<td>:OracleHQEmployee</td>
</tr>
<tr>
<td>:SWcompanyEmployee</td>
<td>:SWcompanyEmployee</td>
</tr>
</tbody>
</table>

SELECT x, y
FROM <http://example.org/emp>
WHERE { ?x rdf:type ?y }

<table>
<thead>
<tr>
<th>x</th>
<th>y</th>
</tr>
</thead>
<tbody>
<tr>
<td>:John</td>
<td>:OracleHQEmployee</td>
</tr>
<tr>
<td>:John</td>
<td>:SWcompanyEmployee</td>
</tr>
</tbody>
</table>

```
```
OWL Example

Asserted Facts (referencing OWL) <http://example.org/gmap>

<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           |
Advancing W3C Semantic Standards

• Our implementation entirely based on W3C standards (RDF, RDFS, OWL)
  • Native SPARQL support is planned

• Members of following W3C Web Semantic Activities:
  • W3C Data Access Working Group (DAWG)
  • W3C OWL 1.1 Working group
  • W3C Semantic Web Education & Outreach (SWEO)
  • W3C Health Care & Life Sciences Interest Group (HCLS)
  • W3C Multimedia Semantics Incubator group
Technical Features

- Storage model, loading, and management for data represented in RDF/OWL
- SQL-based query of RDF/OWL data
- Ontology-assisted query of Relational data
- Native inferencing engine to infer new relationships from RDF/OWL data
Functionality: Overview

INFER
- RDF/S
- User-def.

QUERY
- Query RDF/OWL data and ontologies

STORE
- Incr. DML
- Batch-Load

RDF/OWL data and ontologies & rulebases

Enterprise (Relational) data
Functionality: Overview

INFER
- RDF/S
- User-def.

QUERY
- Query RDF/OWL data and ontologies

STORE
- Incremental DML
- Batch Load
- Bulk Load

RDF/OWL data and ontologies & rulebases

Enterprise (Relational) data
Functionality: Overview

Infer:
- OWL subsets
- RDF/S
- User-def.

Query:
- Query RDF/OWL data and ontologies

Store:
- Incremental DML
- Batch Load
- Bulk Load

RDF/OWL data and ontologies & rulebases

Enterprise (Relational) data
Functionality: Overview

**INFER**
- OWL subsets
- RDF/S
- User-def.

**QUERY**
- Query RDF/OWL data and ontologies
- Ontology-Assisted Query of Enterprise Data

**STORE**
- Incr. DML
- Batch-Load
- Bulk-Load

RDF/OWL data and ontologies & rulebases

Enterprise (Relational) data
Storage: Overview

Application Tables with RDF object type columns

RDF/OWL data and ontologies

Vocabularies and Rulebases

OWL subset

RDF / RDFS

Rulebase \( m \)

Inferred Triple Set 1

Inferred Triple Set 2

Inferred Triple Set \( p \)

Oracle DB Semantic Network (inside MDSYS)

Rules Indexes (Derived data)
Major Steps for building semantic app.
Major Steps for building semantic app.

- Create Semantic Network in Oracle
Major Steps for building semantic app.

- Create Semantic Network in Oracle
- Create an RDF/OWL model associating with a table column of type SDO_RDF_TRIPLE_S
Major Steps for building semantic app.

- **Create Semantic Network** in Oracle
- **Create an RDF/OWL model** associating with a table column of type `SDO_RDF_TRIPLE_S`
- **Store new RDF/OWL data** into Oracle via bulk-load, batch-load, or SQL INSERT; Also, perform any DML
Major Steps for building semantic app.

- **Create Semantic Network** in Oracle
- **Create an RDF/OWL model** associating with a table column of type `SDO_RDF_TRIPLE_S`
- **Store new RDF/OWL data** into Oracle via bulk-load, batch-load, or SQL INSERT; Also, perform any **DML**
- Optionally, **create user-defined rulebases**
Major Steps for building semantic app.

- **Create Semantic Network** in Oracle
- **Create an RDF/OWL model** associating with a table column of type `SDO_RDF_TRIPLE_S`
- **Store new RDF/OWL data** into Oracle via `bulk-load`, `batch-load`, or `SQL INSERT`; Also, perform any `DML`
- Optionally, **create user-defined rulebases**
- **Infer new RDF/OWL data** using Oracle’s native inference engine (`OWLPRIME`, `OWLSIF`, `RDFS++`, `RDFS & User-defined Rules`)
Major Steps for building semantic app.

• **Create Semantic Network** in Oracle
• **Create an RDF/OWL model** associating with a table column of type `SDO_RDF_TRIPLE_S`
• **Store new RDF/OWL data** into Oracle via `bulk-load`, `batch-load`, or `SQL INSERT`; Also, perform any `DML`
• Optionally, **create user-defined rulebases**
• **Infer new RDF/OWL data** using Oracle’s native inference engine (**OWLPRIME, OWLSIF, RDFS++**, `RDFS` & User-defined Rules)
• **Query RDF/OWL data** and ontologies via SQL using `SEM_MATCH` table function; optionally **combine with SQL operations** such as `JOIN`, `ORDER BY`, `GROUP BY`, `FILTER conditions`, etc.
Major Steps for building semantic app.

• **Create Semantic Network** in Oracle
• **Create an RDF/OWL model** associating with a table column of type **SDO_RDF_TRIPLE_S**
• **Store new RDF/OWL data** into Oracle via **bulk-load**, batch-load, or **SQL INSERT**; Also, perform any **DML**
• Optionally, **create user-defined rulebases**
• **Infer new RDF/OWL data** using Oracle’s native inference engine (**OWLPRIME**, **OWLSIF**, **RDFS++**, **RDFS & User-defined Rules**)
• **Query RDF/OWL data** and ontologies via SQL using **SEM_MATCH** table function; optionally **combine with SQL operations** such as join, order by, group by, filter conditions, etc.
• **Perform Ontology-assisted Query** against enterprise (relational) data using **SEM_RELATED** and **SEM_DISTANCE** operators
Creating Semantic Network and Semantic Models
Creating Semantic Network and Semantic Models

Creating a semantic network

1. SEM_APISCREATE_SEM_NETWORK(<tablespace>);
Creating Semantic Network and Semantic Models

Creating a semantic network

1. SEM_APIS.CREATE_SEM_NETWORK (<tablespace>);
Creating Semantic Network and Semantic Models

Creating a semantic network

1. `SEM_APIS.CREATE_SEM_NETWORK (<tablespace>);`

Creating a semantic model

2. Create an application table with an `SDO_RDF_TRIPLE_S` type col
   
   ```sql
   CREATE TABLE ATAB (ID int, TRI SDO_RDF_TRIPLE_S) compress;
   ```

3. Create a model associated with the `SDO_RDF_TRIPLE_S` column
   
   ```sql
   SEM_APIS.CREATE_SEM_MODEL (  
       'MODEL1', -- <model_name>  
       'ATAB',  -- <app_table_name>  
       'TRI'    -- <RDF type col name>  
   );
   ```
Access Control
Access Control

• Models
  • A database view owned by MDSYS gets created at model creation
  • The creator gets SELECT privilege with GRANT option
  • DML on a model is done via DML on the associated RDF object type column and requires invoker to have appropriate privileges on the associated application table
Access Control

• Models
  • A database view owned by MDSYS gets created at model creation
  • The creator gets SELECT privilege with GRANT option
  • DML on a model is done via DML on the associated RDF object type column and requires invoker to have appropriate privileges on the associated application table

• Rulebases
  • A database view owned by MDSYS gets created at rulebase creation
  • The creator gets SELECT and DML privilege with GRANT option
Access Control

• Models
  • A database view owned by MDSYS gets created at model creation
  • The creator gets SELECT privilege with GRANT option
  • DML on a model is done via DML on the associated RDF object type column and requires invoker to have appropriate privileges on the associated application table

• Rulebases
  • A database view owned by MDSYS gets created at rulebase creation
  • The creator gets SELECT and DML privilege with GRANT option

• Rules Indexes (Inferred Triple Sets)
  • A database view owned by MDSYS gets created at rules index creation
  • Creator must have SELECT privilege on underlying model and rulebase views
  • The creator gets SELECT privilege with GRANT option
Storage: Highlights
Storage: Highlights

- Stores `<subject, predicate, object>` triples
  - Removes duplicates to ensure RDF/OWL graph is a set
  - Uses RDF-specific compression for tables and indexes
Storage: Highlights

• Stores <subject, predicate, object> triples
  • Removes duplicates to ensure RDF/OWL graph is a set
  • Uses RDF-specific compression for tables and indexes
• No limits on amount of data that can be stored
  • Current users: Swissprot: 800 million triples, UTH: 600 million+
Storage: Highlights

- Stores <subject, predicate, object> triples
  - Removes duplicates to ensure RDF/OWL graph is a set
  - Uses RDF-specific compression for tables and indexes
- No limits on amount of data that can be stored
  - Current users: Swissprot: 800million triples, UTH: 600million+
- Can handle multiple lexical forms of the same value
  - Ex: “00123”^^xsd:decimal and “123”^^xsd:decimal
  - Ex: “2004-12-21T22:00:00-08:00”^^xsd:dateTime and “2004-12-22T01:00:00-05:00”^^xsd:dateTime
Storage: Highlights

• Stores <subject, predicate, object> triples
  • Removes duplicates to ensure RDF/OWL graph is a set
  • Uses RDF-specific compression for tables and indexes
• No limits on amount of data that can be stored
  • Current users: Swissprot: 800million triples, UTH: 600million+
• Can handle multiple lexical forms of the same value
  • Ex: “00123”^^xsd:decimal and “123”^^xsd:decimal
  • Ex: “2004-12-21T22:00:00-08:00”^^xsd:dateTime and “2004-12-22T01:00:00-05:00”^^xsd:dateTime
• Maintains fidelity (user-specified lexical form)
Storage: Highlights

- Stores <subject, predicate, object> triples
  - Removes duplicates to ensure RDF/OWL graph is a set
  - Uses RDF-specific compression for tables and indexes
- No limits on amount of data that can be stored
  - Current users: Swissprot: 800million triples, UTH: 600million+
- Can handle multiple lexical forms of the same value
  - Ex: “00123”^^xsd:decimal and “123”^^xsd:decimal
  - Ex: “2004-12-21T22:00:00-08:00”^^xsd:dateTime and “2004-12-22T01:00:00-05:00”^^xsd:dateTime
- Maintains fidelity (user-specified lexical form)
- Provides access control for models, rulebases, and rules indexes
Storage: Highlights

- Stores <subject, predicate, object> triples
  - Removes duplicates to ensure RDF/OWL graph is a set
  - Uses RDF-specific compression for tables and indexes
- No limits on amount of data that can be stored
  - Current users: Swissprot: 800million triples, UTH: 600million+
- Can handle multiple lexical forms of the same value
  - Ex: “00123”^^xsd:decimal and “123”^^xsd:decimal
  - Ex: “2004-12-21T22:00:00-08:00”^^xsd:dateTime and “2004-12-22T01:00:00-05:00”^^xsd:dateTime
- Maintains fidelity (user-specified lexical form)
- Provides access control for models, rulebases, and rules indexes
- Supports long literal values
Loading RDF/OWL data
Loading RDF/OWL data

Load data using
Loading RDF/OWL data

Load data using

• Bulk-load (very fast)
  • Load data into a staging table (using SQL*Loader from a file or Named Pipe containing N-Triple formatted data)
  • Invoke PL/SQL API to invoke bulk load from the staging table
Loading RDF/OWL data

Load data using

• Bulk-load (very fast)
  • Load data into a staging table (using SQL*Loader from a file or Named Pipe containing N-Triple formatted data)
  • Invoke PL/SQL API to invoke bulk load from the staging table

• Batch-load (fast, can handle long literals as well)
  • Invoke Java-based API to load from file containing N-Triple formatted data
Loading RDF/OWL data

Load data using

• Bulk-load (very fast)
  • Load data into a staging table (using SQL*Loader from a file or Named Pipe containing N-Triple formatted data)
  • Invoke PL/SQL API to invoke bulk load from the staging table

• Batch-load (fast, can handle long literals as well)
  • Invoke Java-based API to load from file containing N-Triple formatted data

• SQL INSERT (for loading small amounts of data)
Loading APIs: Bulk-Load
Loading APIs: Bulk-Load

• Use SQL*Loader to load staging table
  • Control file template is available in 11g companion CD
    • Only the Staging Table name may need to be changed
  • Staging Table definition is shown in documentation
    • Use COMPRESS option, if available
  • Input file must be N-Triple formatted
    • Named Pipe may be used to save disk space
Loading APIs: Bulk-Load

• Use SQL*Loader to load staging table
  • Control file template is available in 11g companion CD
    • Only the Staging Table name may need to be changed
  • Staging Table definition is shown in documentation
    • Use COMPRESS option, if available
  • Input file must be N-Triple formatted
    • Named Pipe may be used to save disk space
• SEM_APIS.BULK_LOAD_FROM_STAGING_TABLE
  • Model_owner
  • Table_owner
  • Table_name
  • Flags (default NULL): ‘ VALUES_TABLE_INDEX_REBUILD ‘
Loading APIs: Batch-Load

- Batch-load uses the `oracle.spatial.rdf.client.BatchLoader` class packaged in `<ORACLE_HOME>/md/jlib/sdordf.jar`

- Example (on Linux)
  ```
  java
  -Ddb.user=scott -Ddb.password=password
  -Ddb.host=127.0.0.1 -Ddb.port=1522 -Ddb.sid=orcl
  -classpath ${ORACLE_HOME}/md/jlib/sdordf.jar:${ORACLE_HOME}/jdbc/lib/ojdbc5.jar
  oracle.spatial.rdf.client.BatchLoader
  <N-TripleFile> <tablename> <tablespaceName> <modelName>
  ```
Query RDF Data

- SPARQL-like graph pattern embedded in SQL query
- Matches RDF/OWL graph patterns with patterns in stored data
- Returns a table of results
- Can use SQL operators/functions to process results
- Avoids staging when combined with queries on relational data

```
SELECT t.x ...
FROM ...,
TABLE (SEM_MATCH invocation)
) t, ...
WHERE ...
```
Query RDF Data

- SPARQL-like graph pattern embedded in SQL query
- Matches RDF/OWL graph patterns with patterns in stored data
- Returns a table of results
- Can use SQL operators/functions to process results
- Avoids staging when combined with queries on relational data

```
SELECT t.x ...
FROM ...
TABLE (SEM_MATCH (SEM_Models('gmap'),
SEM_Rulebases('OWLPRIME'),
SEM_Aliases(...) 
null
) t, ...
WHERE ...
```

-- pattern: all parts of N.A.
-- RDF/OWL data models
-- rulebases
-- aliases
-- no filter condition
Query RDF Data

- SPARQL-like graph pattern embedded in SQL query
- Matches RDF/OWL graph patterns with patterns in stored data
- Returns a table of results
- Can use SQL operators/functions to process results
- Avoids staging when combined with queries on relational data

```sql
SELECT t.x ...
FROM ...
, TABLE (SEM_MATCH invocation) t, ...
WHERE ...
```
Table Columns returned by SEM_MATCH

Each returned row contains one (or more) of the following cols for each variable \(?x\) in the graph-pattern:

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>(x)</td>
<td>varchar2</td>
<td>Value matched with (?x)</td>
</tr>
<tr>
<td>(x$rdfVTYP)</td>
<td>varchar2</td>
<td>Value TYPe: URI, Literal, or Blank Node</td>
</tr>
<tr>
<td>(x$rdfLTYP)</td>
<td>varchar2</td>
<td>Literal TYPe: e.g., xsd:integer</td>
</tr>
<tr>
<td>(x$rdfCLOB)</td>
<td>CLOB</td>
<td>CLOB value matched with (?x)</td>
</tr>
<tr>
<td>(x$rdfLANG)</td>
<td>varchar2</td>
<td>LANGuage tag: e.g., “en-us”</td>
</tr>
</tbody>
</table>

**Projection Optimization**: Only the columns referred to by the containing query are returned.
RDF Query in SQL

```
SELECT e, p FROM TABLE(SEM_MATCH('(?e rdf:type                              ?empCategory)
(?empCategory :corpOfficeLoc ?loc)
(?loc :partOf ?p)'
SEM_Models('emp', 'gmap'), SEM_Rulebases('OWLPRIME'),
SEM_ALIASES(SEM_ALIAS('', 'http://www.example.org/')),NULL));
```

<table>
<thead>
<tr>
<th>E</th>
<th>P</th>
</tr>
</thead>
<tbody>
<tr>
<td>:John</td>
<td>:USA</td>
</tr>
<tr>
<td>:John</td>
<td>:NorthAmerica</td>
</tr>
</tbody>
</table>
Ontology-Assisted Query: Overview

• Motivation
  • Traditionally relationship between two terms is checked only in a **syntactic** manner
  • Need a new operator which can do **semantic** relationship check by consulting an ontology

• Introduces two operators
  • **SEM_RELATED** (<col>,<pred>, <ontologyTerm>, <ontologyName> [,<invoc_id>])
  • **SEM_DISTANCE** (<invoc_id>) ← **Ancillary Oper.**
Example: Query using Semantic Operators

Patients

<table>
<thead>
<tr>
<th>ID</th>
<th>DIAGNOSIS</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Hand_Fracture</td>
</tr>
<tr>
<td>2</td>
<td>Rheumatoid_Arthritis</td>
</tr>
</tbody>
</table>

“Find all entries in diagnosis column that are related to ‘Upper_Extremity_Fracture’”

Syntactic query will not work:
SELECT p_id, diagnosis FROM Patients WHERE diagnosis = ‘Upper_Extremity_Fracture’;
Example: Query using Semantic Operators

```
SELECT p_id, diagnosis FROM Patients
WHERE SEM_RELATED (diagnosis, 'rdfs:subClassOf', 'Upper_Extremity_Fracture', sem_models('NCI'), sem_rulebases('OWLPRIME'), ...) = 1;
```

“Find all entries in diagnosis column that are related to Upper_Extremity_Fracture”
Example: Query using Semantic Operators

```
Patients

<table>
<thead>
<tr>
<th>ID</th>
<th>DIAGNOSIS</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Hand_Fracture</td>
</tr>
<tr>
<td>2</td>
<td>Rheumatoid_Arthritis</td>
</tr>
</tbody>
</table>
```

“Find all entries in diagnosis column that are related to ‘Upper_Extremity_Fracture’”

Syntactic query will not work:

```
SELECT p_id, diagnosis FROM Patients WHERE diagnosis = 'Upper_Extremity_Fracture';
```
Example: Query using Semantic Operators

```
SELECT p_id, diagnosis FROM Patients
WHERE SEM_RELATED (diagnosis, 'rdfs:subClassOf', 'Upper_Extremity_Fracture', sem_models('NCI'), sem_rulebases('OWLPRIME'), ... , 777) = 1 AND SEM_DISTANCE(777) <= 2;
```

“Find all entries in diagnosis column that are related to Upper_Extremity_Fracture.”
Example: Query using Semantic Operators

```
<table>
<thead>
<tr>
<th>ID</th>
<th>DIAGNOSIS</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Hand_Fracture</td>
</tr>
<tr>
<td>2</td>
<td>Rheumatoid_Arthritis</td>
</tr>
</tbody>
</table>
```

“Find all entries in diagnosis column that are related to ‘Upper_Extremity_Fracture’”

Syntactic query will not work:
```
SELECT p_id, diagnosis FROM Patients WHERE diagnosis = ‘Upper_Extremity_Fracture’;
```
Inference: Overview

- Native inferencing for
  - RDF, RDFS
  - OWL subsets
  - User-defined rules
- Rules are stored in rulebases
- RDF/OWL graph is entailed (new triples are inferred) by applying rules in rulebase(s) to model(s)
- Inferencing is based on forward chaining: new triples are inferred and stored ahead of query time
Inferencing

• RDFS Example
  • rdfs:subClassOf is transitive
  • rdfs:subPropertyOf is transitive
  • X rdf:type C, C rdfs:subClassOf SC => X rdf:type SC

• OWL Example
  • :partOf rdf:type owl:TransitiveProperty
  • :friendOf rdf:type owl:SymmetricProperty

• User-defined Rule Example:
  • X :hasParent Y, Y :hasBrother Z => X :hasUncle Z
OWL Subsets Supported

- **RDFS++**
  - RDFS plus owl:sameAs and owl:InverseFunctionalProperty
- **OWLSIF (OWL with IF semantics)**
  - Based on Dr. Horst’s pD* vocabulary¹
- **OWLPrime**
  - rdfs:subClassOf, subPropertyOf, domain, range
  - owl:TransitiveProperty, SymmetricProperty, FunctionalProperty, InverseFunctionalProperty, inverseOf
  - owl:sameAs, differentFrom
  - owl:disjointWith, complementOf,
  - owl:hasValue, allValuesFrom, someValuesFrom
  - owl:equivalentClass, equivalentProperty
- **Jointly determined with domain experts, customers and partners**

¹ Completeness, decidability and complexity of entailment for RDF Schema and a semantic extension involving the OWL vocabulary
11g OWL Inference PL/SQL API

- **SEM_APIS.CREATE_ENTAILMENT**
  - `Index_name`
  - `sem_models('GraphTBox', 'GraphABox', ...)`,
  - `sem_rulebases('OWLPrime')`,
  - `passes`,
  - `Inf_components`,
  - `Options`
  - Use “PROOF=T” to generate inference proof

- **SEM_APIS.VALIDATE_ENTAILMENT**
  - `sem_models(('GraphTBox', 'GraphABox', ...),`
  - `sem_rulebases('OWLPrime'),`
  - `Criteria`,
  - `Max_conflicts`,
  - `Options`
  - Typical Usage:
    - First load RDF/OWL data
    - Call create_entailment to generate inferred graph
    - Query both original graph and inferred data
    - Inferred graph contains only new triples! Saves time & resources

Above APIs can be invoked from Java clients through JDBC
Performance Evaluation using Desktop
Database Setup

- Linux based **commodity** PC (1 CPU, 3GHz, 2GB RAM)
- Database installed on machine “semperf3”

- Two other PCs are just serving storage over network
Oracle 11g Bulk-Loader Performance
Oracle 11g Bulk-Loader Performance

LUBM-50 (6.9 million triples)
Oracle 11g Bulk-Loader Performance

LUBM-50 (6.9 million triples)

- TIME $\rightarrow$ 0 hour 13.5 min (31 mil / hr)
Oracle 11g Bulk-Loader Performance

LUBM-50 (6.9 million triples)

- **TIME** → 0 hour 13.5 min (31 mil / hr)
  - SQL loader: 4 min 56 sec
Oracle 11g Bulk-Loader Performance

LUBM-50 (6.9 million triples)

- **TIME → 0 hour 13.5 min (31 mil / hr)**
  - SQL loader: 4 min 56 sec
  - bulk_load API: 8 min 33 sec
Oracle 11g Bulk-Loader Performance

LUBM-50 (6.9 million triples)
- **TIME** → 0 hour 13.5 min (**31 mil / hr**)
  - SQL loader: 4 min 56 sec
  - bulk_load API: 8 min 33 sec

LUBM-500 (69 million triples)
Oracle 11g Bulk-Loader Performance

LUBM-50 (6.9 million triples)
- **TIME** → 0 hour 13.5 min (31 mil / hr)
  - SQL loader: 4 min 56 sec
  - bulk_load API: 8 min 33 sec

LUBM-500 (69 million triples)
- **TIME** → 3 hour 20 min (21 mil / hr)
Oracle 11g Bulk-Loader Performance

LUBM-50 (6.9 million triples)
- **TIME** → **0 hour 13.5 min** *(31 mil / hr)*
  - SQL loader: 4 min 56 sec
  - bulk_load API: 8 min 33 sec

LUBM-500 (69 million triples)
- **TIME** → **3 hour 20 min** *(21 mil / hr)*
  - SQL loader: 0 hour 46 min
Oracle 11g Bulk-Loader Performance

LUBM-50 (6.9 million triples)
• **TIME** → 0 hour 13.5 min *(31 mil / hr)*
  • SQL loader: 4 min 56 sec
  • bulk_load API: 8 min 33 sec

LUBM-500 (69 million triples)
• **TIME** → 3 hour 20 min *(21 mil / hr)*
  • SQL loader: 0 hour 46 min
  • bulk_load API: 2 hour 34 min
Oracle 11g Bulk-Loader Performance

LUBM-50 (6.9 million triples)
• **TIME → 0 hour 13.5 min (31 mil / hr)**
  • SQL loader: 4 min 56 sec
  • bulk_load API: 8 min 33 sec

LUBM-500 (69 million triples)
• **TIME → 3 hour 20 min (21 mil / hr)**
  • SQL loader: 0 hour 46 min
  • bulk_load API: 2 hour 34 min

LUBM-1000 (138 million triples)
Oracle 11g Bulk-Loader Performance

LUBM-50 (6.9 million triples)
- **TIME** → 0 hour 13.5 min (31 mil / hr)
  - SQL loader: 4 min 56 sec
  - bulk_load API: 8 min 33 sec

LUBM-500 (69 million triples)
- **TIME** → 3 hour 20 min (21 mil / hr)
  - SQL loader: 0 hour 46 min
  - bulk_load API: 2 hour 34 min

LUBM-1000 (138 million triples)
- **TIME** → 6 hour 23 min (21 mil / hr)
Oracle 11g Bulk-Loader Performance

LUBM-50 (6.9 million triples)
- **TIME** → 0 hour 13.5 min (**31 mil / hr**)
  - SQL loader: 4 min 56 sec
  - bulk_load API: 8 min 33 sec

LUBM-500 (69 million triples)
- **TIME** → 3 hour 20 min (**21 mil / hr**)
  - SQL loader: 0 hour 46 min
  - bulk_load API: 2 hour 34 min

LUBM-1000 (138 million triples)
- **TIME** → 6 hour 23 min (**21 mil / hr**)
  - SQL loader: 1 hour 34 min
Oracle 11g Bulk-Loader Performance

LUBM-50 (6.9 million triples)
• **TIME** → 0 hour 13.5 min (**31 mil / hr**)
  • SQL loader: 4 min 56 sec
  • bulk_load API: 8 min 33 sec

LUBM-500 (69 million triples)
• **TIME** → 3 hour 20 min (**21 mil / hr**)
  • SQL loader: 0 hour 46 min
  • bulk_load API: 2 hour 34 min

LUBM-1000 (138 million triples)
• **TIME** → 6 hour 23 min (**21 mil / hr**)
  • SQL loader: 1 hour 34 min
  • bulk_load API: 4 hour 49 min
Oracle 11g Bulk-Loader Performance

LUBM-50 (6.9 million triples)
• **TIME** → 0 hour 13.5 min (31 mil / hr)
  • SQL loader: 4 min 56 sec
  • bulk_load API: 8 min 33 sec

LUBM-500 (69 million triples)
• **TIME** → 3 hour 20 min (21 mil / hr)
  • SQL loader: 0 hour 46 min
  • bulk_load API: 2 hour 34 min

LUBM-1000 (138 million triples)
• **TIME** → 6 hour 23 min (21 mil / hr)
  • SQL loader: 1 hour 34 min
  • bulk_load API: 4 hour 49 min

• **DISK SPACE** → 19 GB (7.3 mil / GB)
Oracle 11g Bulk-Loader Performance

LUBM-50 (6.9 million triples)
- **TIME** → 0 hour 13.5 min *(31 mil / hr)*
  - SQL loader: 4 min 56 sec
  - bulk_load API: 8 min 33 sec

LUBM-500 (69 million triples)
- **TIME** → 3 hour 20 min *(21 mil / hr)*
  - SQL loader: 0 hour 46 min
  - bulk_load API: 2 hour 34 min

LUBM-1000 (138 million triples)
- **TIME** → 6 hour 23 min *(21 mil / hr)*
  - SQL loader: 1 hour 34 min
  - bulk_load API: 4 hour 49 min
- **DISK SPACE** → 19 GB *(7.3 mil / GB)*
  - Triples and Values: 5 GB
Oracle 11g Bulk-Loader Performance

LUBM-50 (6.9 million triples)
- **TIME** → 0 hour 13.5 min (31 mil / hr)
  - SQL loader: 4 min 56 sec
  - bulk_load API: 8 min 33 sec

LUBM-500 (69 million triples)
- **TIME** → 3 hour 20 min (21 mil / hr)
  - SQL loader: 0 hour 46 min
  - bulk_load API: 2 hour 34 min

LUBM-1000 (138 million triples)
- **TIME** → 6 hour 23 min (21 mil / hr)
  - SQL loader: 1 hour 34 min
  - bulk_load API: 4 hour 49 min
- **DISK SPACE** → 19 GB (7.3 mil / GB)
  - Triples and Values: 5 GB
  - Indexes: 11 GB
Oracle 11g Bulk-Loader Performance

LUBM-50 (6.9 million triples)
- **TIME** → 0 hour 13.5 min (31 mil / hr)
  - SQL loader: 4 min 56 sec
  - bulk_load API: 8 min 33 sec

LUBM-500 (69 million triples)
- **TIME** → 3 hour 20 min (21 mil / hr)
  - SQL loader: 0 hour 46 min
  - bulk_load API: 2 hour 34 min

LUBM-1000 (138 million triples)
- **TIME** → 6 hour 23 min (21 mil / hr)
  - SQL loader: 1 hour 34 min
  - bulk_load API: 4 hour 49 min
- **DISK SPACE** → 19 GB (7.3 mil / GB)
  - Triples and Values: 5 GB
  - Indexes: 11 GB
  - App Table (compressed): 3 GB
Oracle 11g Bulk-Loader Performance

LUBM-50 (6.9 million triples)
- **TIME** → **0 hour 13.5 min (31 mil / hr)**
  - SQL loader: 4 min 56 sec
  - bulk_load API: 8 min 33 sec

LUBM-500 (69 million triples)
- **TIME** → **3 hour 20 min (21 mil / hr)**
  - SQL loader: 0 hour 46 min
  - bulk_load API: 2 hour 34 min

LUBM-1000 (138 million triples)
- **TIME** → **6 hour 23 min (21 mil / hr)**
  - SQL loader: 1 hour 34 min
  - bulk_load API: 4 hour 49 min

- **DISK SPACE** → **19 GB (7.3 mil / GB)**
  - Triples and Values: 5 GB
  - Indexes: 11 GB
  - App Table (compressed): 3 GB
## Query Performance

<table>
<thead>
<tr>
<th>Ontology LUBM50</th>
<th>LUBM Benchmark Queries</th>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
<th>Q4</th>
<th>Q5</th>
<th>Q6</th>
<th>Q7</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.8 million &amp; 3+ million inferred</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>OWLPrime</strong></td>
<td># answers</td>
<td>4</td>
<td>130</td>
<td>6</td>
<td>34</td>
<td>719</td>
<td>393730</td>
<td>59</td>
</tr>
<tr>
<td></td>
<td>Time (sec)</td>
<td>0.09</td>
<td>0.80</td>
<td>0.28</td>
<td>6.1</td>
<td>0.4</td>
<td>36.82</td>
<td>1.05</td>
</tr>
<tr>
<td><strong>OWLPrime + Pellet on TBox</strong></td>
<td># answers</td>
<td>4</td>
<td>130</td>
<td>6</td>
<td>34</td>
<td>719</td>
<td>519842</td>
<td>67</td>
</tr>
<tr>
<td></td>
<td>Time (sec)</td>
<td>0.09</td>
<td>0.79</td>
<td>0.28</td>
<td>9.5</td>
<td>0.4</td>
<td>43.65</td>
<td>1.13</td>
</tr>
</tbody>
</table>
### Query Performance (2)

<table>
<thead>
<tr>
<th>Ontology LUBM50</th>
<th>LUBM Benchmark Queries</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.8 million &amp; 3+ million inferred</td>
<td>Q8</td>
</tr>
<tr>
<td>OWLPrime</td>
<td># answers</td>
</tr>
<tr>
<td></td>
<td>Time (sec)</td>
</tr>
</tbody>
</table>
## Inference Performance

<table>
<thead>
<tr>
<th>Ontology (size) (after duplicate elimination)</th>
<th>RDFS</th>
<th>OWLPrime</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>#Triples inferred (millions)</td>
<td>Time (speed)</td>
</tr>
<tr>
<td>LUBM-50</td>
<td>2.75</td>
<td>12.25 min (13.5 mil / hr)</td>
</tr>
<tr>
<td>6.6 million</td>
<td></td>
<td></td>
</tr>
<tr>
<td>LUBM-1000</td>
<td>55.09</td>
<td>7 hr 19 min (7.5 mil / hr)</td>
</tr>
<tr>
<td>133.6 million</td>
<td></td>
<td></td>
</tr>
<tr>
<td>UniProt</td>
<td>3.4</td>
<td>24.1 min (8.5 mil / hr)</td>
</tr>
<tr>
<td>20 million</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Results collected on a single CPU PC (3GHz), 2GB RAM (1.4G dedicate to DB), Multiple Disks over NFS
Oracle 11g RDF/OWL

- **New features**
  - Bulk loader
  - Native OWL inference support (with optional proof generation)
  - Semantic operators
- **Performance improvement**
  - Much faster compared to 10gR2
    - Loading
    - Query (table function re-write, query hints)
    - Inference
- **Shipped in 2007**
- **Java API support**
  - Oracle Jena adaptor (released on OTN)
  - Sesame (forthcoming)
For More Information

search.oracle.com

semantic technologies

or

oracle.com