The Semantic Web for Application Developers

Oracle New England Development Center
Zhe Wu, Ph.D.
alan.wu@oracle.com
Agenda

• Background
  • 10gR2 RDF
  • 11g RDF/OWL

• New 11g features
  • Bulk loader
  • Semantic operators
  • Native OWL inferencing

• Oracle Jena adaptor

• Example usage flow

• Performance

• Summary
Semantic Technology Stack

User Interface & Applications

Trust

Proof

Unifying Logic

Query: SPARQL
Ontology: OWL
Rule: RIF
RDFS

Data interchange: RDF
XML

URI/IRI

Standards based

http://www.w3.org/2007/03/layerCake.svg
Advancing W3C Semantic Standards

• Our implementation entirely based on W3C standards (RDF, RDFS, OWL)
  • SPARQL support is planned
• Members of following W3C Web Semantic Activities:
  • W3C Data Access Working Group (DAWG)
  • W3C Semantic Web Education & Outreach (SWEO)
  • W3C Health Care & Life Sciences Interest Group (HCLS)
  • W3C Multimedia Semantics Incubator group
  • W3C OWL 1.1 Working group
Oracle 10gR2 RDF

- **Storage and Load**
  - Stored after normalization in Oracle Tables
  - Incremental Load using DMLs
    - `insert into rdf_data values (…, sdo_rdf_triple_s(1, ‘<subject>’, ‘<predicate>’, ‘<object>’));`
  - Java API based Batch Loader Utility
- **Inference** (forward chaining based)
  - Supports RDFS inference
  - Supports User-Defined rules
  - PL/SQL API `create_rules_index`
- **Query** using SDO_RDF_MATCH Table function
  - `Select x, y from table(sdo_rdf_match( ‘(?x rdf:type :Protein) (?x :name ?y)’ ....));`
  - Seamless SQL integration
- **Model level security and access control**
- **Shipped in 2005**
Oracle 10gR2 RDF

• **Storage and Load**
  - Stored after normalization in Oracle Tables
  - Incremental Load using DMLs
    - `insert into rdf_data values (... , sdo_rdf_triple_s(1, '<subject>', '<predicate>', '<object>'));`
  - Java API based Batch Loader Utility

• **Inference** (forward chaining based)
  - Supports RDFS inference
  - Supports User-Defined rules
  - PL/SQL API `create_rules_index`

• **Query** using SDO_RDF_MATCH Table function
  - `Select x, y from table(sdo_rdf_match( (?x rdf:type :Protein) (?x :name ?y) … ));`
  - Seamless SQL integration

• **Model level security and access control**
• **Shipped in 2005**
Oracle 10gR2 RDF

- **Storage and Load**
  - Stored after normalization in Oracle Tables
  - Incremental Load using DMLs
    - `insert into rdf_data values (…, sdo_rdf_triple_s(1, ‘<subject>’, ‘<predicate>’, ‘<object>’));`
  - Java API based Batch Loader Utility

- **Inference** (forward chaining based)
  - Supports RDFS inference
  - Supports User-Defined rules
  - PL/SQL API `create_rules_index`

- **Query** using SDO_RDF_MATCH Table function
  - Select x, y from table(sdo_rdf_match( ‘(?x rdf:type :Protein) (?x :name ?y)’ ....));
  - Seamless SQL integration

- **Model level security and access control**
- **Shipped in 2005**
Oracle 10gR2 RDF

- **Storage and Load**
  - Stored after normalization in Oracle Tables
  - Incremental Load using DMLs
    - `insert into rdf_data values (…, sdo_rdf_triple_s(1, ‘<subject>’, ‘<predicate>’, ‘<object>’));`
  - Java API based Batch Loader Utility

- **Inference** (forward chaining based)
  - Supports RDFS inference
  - Supports User-Defined rules
  - PL/SQL API `create_rules_index`

- **Query** using SDO_RDF_MATCH Table function
  - `Select x, y from table(sdo_rdf_match(`
    - `(?x rdf:type :Protein) (?x :name ?y)` ...
  - Seamless SQL integration

- **Model level security and access control**
- Shipped in 2005
Oracle 10gR2 RDF

- **Storage and Load**
  - Stored after normalization in Oracle Tables
  - Incremental Load using DMLs
    - `insert into rdf_data values (…, sdo_rdf_triple_s(1, '<subject>', '<predicate>', '<object>'));`
  - Java API based Batch Loader Utility
- **Inference** (forward chaining based)
  - Supports RDFS inference
  - Supports User-Defined rules
  - PL/SQL API `create_rules_index`
- **Query** using SDO_RDF_MATCH Table function
  - `Select x, y from table(sdo_rdf_match('(?x rdf:type :Protein) (?x :name ?y)’ ....));`
  - Seamless SQL integration
- **Model level security and access control**
- **Shipped in 2005**
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)
Example RDF Graph and Query

SEM_MATCH returns the name of Tom’s grandfather in the form of a relational table

```
select x, y, name from TABLE(SEM_MATCH(
    '(:Tom :hasParent ?x)
    (?x     :hasFather ?y)
    (?y     :name         ?name)',
    SEM_Models('family'),       .., .., ..));
```

<table>
<thead>
<tr>
<th>X</th>
<th>Y</th>
<th>NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>Matt</td>
<td>John</td>
<td>“John D”</td>
</tr>
</tbody>
</table>
Oracle 11g Feature: Bulk Loader

- Scalable, efficient solution for loading hundreds of millions of triples and beyond
  - Re-architected, re-designed in 11gR1

- Usage flow
  - Using SQL Loader to load an N-TRIPLE file into a staging table
  - Invoke `sem/apis.bulk_load_from_staging_table` PL/SQL API

- To load RDF graphs in RDF/XML or N3 format
  - Use named pipe together with a third-party tool that converts source file into N-TRIPLE file, **or**
  - Use Oracle Jena adaptor
    - `oracleBulkUpdateHandler.addInBulk(GraphUtil.findAll(sourceGraph), tbs);`

N-TRIPLE format

````
<:Mary> <:motherOf> <:John>
<:Mary> <:motherOf> <:Sue>
<:Sue> <:sisterOf> <:John>
<:Tom> <:hasParent> <:Matt>
```
Oracle 11g Feature: Semantic Operators

- Scalable, efficient SQL operators to perform ontology-assisted query against enterprise relational data

<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>

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

Syntactic query against relational table will not work!

```sql
SELECT p_id, diagnosis
FROM Patients
WHERE diagnosis = 'Upper_Extremity_Fracture';
```

→ Zero Matches!

New Semantic query against relational data (while consulting ontology)

```sql
SELECT p_id, diagnosis
FROM Patients
WHERE SEM_RELATED (diagnosis, ‘rdfs:subClassOf’, ‘Upper_Extremity_Fracture’, ‘Medical_ontology’) = 1;
```
Oracle 11g Feature: Semantic Operators

- Scalable, efficient SQL operators to perform ontology-assisted query against enterprise relational data

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

**Syntactic query against relational table will not work!**

```
SELECT p_id, diagnosis
FROM Patients
WHERE diagnosis = 'Upper_Extremity_Fracture';
```

**New Semantic query against relational data (while consulting ontology)**

```
SELECT p_id, diagnosis
FROM Patients
WHERE SEM_RELATED (diagnosis, 'rdfs:subClassOf', 'Upper_Extremity_Fracture', 'Medical_ontology') = 1
AND SEM_DISTANCE() <= 2;
```
Oracle 11g Feature: Semantic Operators

- Scalable, efficient SQL operators to perform ontology-assisted query against enterprise relational data

### Traditional Syntactic query against relational data

**Query:**
```
SELECT p_id, diagnosis
FROM Patients
WHERE diagnosis = 'Upper_Extremity_Fracture';
```

- **Result:** Zero Matches!

### New Semantic query against relational data (while consulting ontology)

**Query:**
```
SELECT p_id, diagnosis
FROM Patients
WHERE SEM_RELATED (diagnosis, 'rdfs:subClassOf', 'Upper_Extremity_Fracture', 'Medical_ontology') = 1;
```

---

<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>

- **Recommended use:**
  - Use for queries requiring ontology-assisted data exploration.
  - Enhances query flexibility and reduces logical inference errors.

---

**Data Sample:**

- **Patients diagnosis table**
  - Hand_Fracture
  - Rheumatoid_Arthritis

**Related Classes:**
- Upper_Extremity_Fracture
- Arm_Fracture
- Forearm_Fracture
- Elbow_Fracture
- Finger_Fracture
- Hand_Fracture
- Elbow_Fracture

**RDF Classes:**
- rdfs:subClassOf
Oracle 11g Feature: OWL Inference

• Scalable, efficient, forward-chaining based reasoner that supports an expressive subset of OWL-DL

• Why OWL-DL subset?
  • Have to support large ontologies (with large ABox)
    • Hundreds of millions of triples and beyond
    • No existing reasoner handles complete DL semantics at this scale
    • Neither Pellet nor KAON2 can handle LUBM10 or ST ontologies on a setup of 64 Bit machine, 4GB Heap¹

• Why forward chaining?
  • Efficient query support
  • Can accommodate any graph query patterns

¹ The summary Abox: Cutting Ontologies Down to Size. ISWC 2006
OWL Subsets Supported

• **Three subsets for different applications**
  - **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,
    - owl:inverseOf, sameAs, differentFrom
    - owl:disjointWith, complementOf,
    - owl:hasValue, allValuesFrom, someValuesFrom

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
)
  - **Inferred graph contains only new triples! Saves time & resources**

**Typical Usage:**
- First load RDF/OWL data
- Call create_entailment to generate inferred graph
- Query both original graph and inferred data

**Typical Usage:**
- First load RDF/OWL data
- Call create_entailment to generate inferred graph
- Call validate_entailment to find inconsistencies

- **Above APIs can be invoked from Java clients through JDBC**
Applications of Partial DL Semantics

• “One very heavily used space is that where RDFS plus some minimal OWL is used to enhance data mapping or to develop simple schemas.”
  -James Hendler

• Complexity distribution of existing ontologies
  • Out of 1,200+ real-world OWL ontologies
    • Collected using Swoogle, Google, Protégé OWL Library, DAML ontology library …
  • 43.7% (or 556) ontologies are RDFS
  • 30.7% (or 391) ontologies are OWL Lite
  • 20.7% (or 264) ontologies are OWL DL.
  • Remaining OWL FULL

2 A Survey of the web ontology landscape. ISWC 2006
Support Semantics beyond OWLPrime (1)

- Option 1: add user-defined rules
  - Both 10gR2 RDF and 11g RDF/OWL supports user-defined rules in this form (filter is supported)

```
<table>
<thead>
<tr>
<th>Antecedents</th>
<th>Consequents</th>
</tr>
</thead>
<tbody>
<tr>
<td>?z :brotherOf ?x .</td>
<td></td>
</tr>
</tbody>
</table>
```

- E.g. to support core semantics of owl:intersectionOf

```xml
<owl:Class rdf:ID="FemaleAstronaut">
  <rdfs:label>female astronaut</rdfs:label>
  <owl:intersectionOf rdf:parseType="Collection">
    <owl:Class rdf:about="#Female" />
    <owl:Class rdf:about="#Astronaut" />
  </owl:intersectionOf>
</owl:Class>
```

1. ➔ :FemaleAstronaut rdfs:subClassOf :Female
2. ➔ :FemaleAstronaut rdfs:subClassOf :Astronaut
3. ➔ ?x rdf:type :Female .
   ➔ ?x rdf:type :Astronaut . ➔
   ➔ x rdf:type :FemaleAstronaut
Support Semantics beyond OWLPrime (2)

- **Option2: Separation in TBox and ABox reasoning through Oracle Jena Adaptor**
  - TBox tends to be small in size
    - Generate a class subsumption tree using complete DL reasoners (like Pellet, KAON2, Fact++, Racer, etc)
  - ABox can be arbitrarily large
    - Use Oracle OWL to infer new knowledge based on the class subsumption tree from TBox

![Diagram showing separation of TBox and ABox reasoning through Oracle Jena Adaptor](image-url)
Oracle Jena Adaptor

• Implementation of Jena’s Graph/Model/BulkUpdateHandler API
  • Started based on UTHSC’s request
  • Joint efforts by UTHSC, Top Quadrant, HP Lab, and Oracle

• What is achieved
  • SPARQL support
  • Fast data loading through Java API
  • Easy integration with external OWL-DL reasoners
  • Better 10gR2 performance
  • On-demand statistics collection

• Example
  Oracle oracle = new Oracle(jdbcUrl, user, password);
  GraphOracleSem graph = new GraphOracleSem(oracle, modelName);
  Node sub = Node.createURI("http://test/s");
  Node pred = Node.createURI("http://test/p");
  Node obj = Node.createURI("http://test/o");

  Triple triple1 = Triple.create(sub, pred, obj);
Example 11g RDF/OWL Usage Flow

- Create an application table
  - create table app_table(triple sdo_rdf_triple_s);

- Create a semantic model
  - exec sem_apis.create_sem_model('family', 'app_table', 'triple');

- Load data
  - Use DML, Bulk loader, or Batch loader
    - insert into app_table (triple) values(1, sdo_rdf_triple_s('family', '<http://www.example.org/family/Matt>', '<http://www.example.org/family/fatherOf>', '<http://www.example.org/family/Cindy>'));
  - ... 

- Run inference
  - exec sem_apis.create_entailment('family_idx', sem_models('family'), sem_rulebases('owlprime'));

- Query both original model and inferred data
  select p, o

After inference is done, what will happen if
- New assertions are added to the graph
  - Inferred data becomes incomplete. Existing inferred data will be reused if create_entailment API invoked again. Faster than rebuild.

- Existing assertions are removed from the graph
  - Inferred data becomes invalid. Existing inferred data will not be reused if the create_entailment API is invoked again.
Performance Evaluation using desktop PC
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

[Diagram showing network setup with semperf3, semperf1, semperf2, and Giga-bit Network connections.]
Data Loading Performance

- Oracle 11g bulk loader
  - LUBM50 (6.9 million triples)
    - TOTAL → 13 min 29 sec
      - SQL loader: 4 min 56 sec; sem_api.bulk_load_from_staging_table: 8 min 33 sec
  - LUBM500 (69 million triples)
    - TOTAL → 3 hour 20 min
      - SQL loader: 46 min; sem_api.bulk_load_from_staging_table: 2 hour 34 min
  - LUBM1000 (138 million triples)
    - TOTAL → 6 hour 23 min
      - SQL loader 1 hour 34 min; sem_api.bulk_load_from_staging_table: 4 hour 49 min
# Query Performance

<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>Q1</td>
</tr>
<tr>
<td><strong>OWLPrime</strong></td>
<td># answers</td>
</tr>
<tr>
<td>Time (sec)</td>
<td>0.09</td>
</tr>
<tr>
<td><strong>OWLPrime + Pellet on TBox</strong></td>
<td># answers</td>
</tr>
<tr>
<td>Time (sec)</td>
<td>0.09</td>
</tr>
<tr>
<td>Ontology LUBM50</td>
<td>LUBM Benchmark Queries</td>
</tr>
<tr>
<td>-----------------</td>
<td>------------------------</td>
</tr>
<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>
<tr>
<td>OWLPrime + Pellet on TBox</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>
<th>OWLPrime + Pellet on TBox</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>#Triples inferred (millions)</td>
<td>Time</td>
<td>#Triples inferred (millions)</td>
</tr>
<tr>
<td>LUBM50 6.6 million</td>
<td>2.75</td>
<td>12min 14s</td>
<td>3.05</td>
</tr>
<tr>
<td>LUBM1000 133.6 million</td>
<td>55.09</td>
<td>7h 19min</td>
<td>61.25</td>
</tr>
<tr>
<td>UniProt 20 million</td>
<td>3.4</td>
<td>24min 06s</td>
<td>50.8</td>
</tr>
</tbody>
</table>

### As a reference (not a comparison)

BigOWLIM *loads, inferences, and stores* (2GB RAM, P4 3.0GHz,)
- LUBM50 in 11 minutes (JAVA 6, -Xmx192)
- LUBM1000 in 11h 20min (JAVA 5, -Xmx1600)

Note: Our inference time *does not* include loading time! Also, set of rules is different.

• Results collected on a single CPU PC (3GHz), 2GB RAM (1.4G dedicate to DB), Multiple Disks over NFS

---

Summary

- 11g provides a robust, efficient, and scalable semantic data management
  - Data loading
  - Query (graph query and ontology-assisted query)
  - Inference

- Java APIs make application building easier

- Future direction
  - Even better performance
  - Richer semantics
  - More application/tools integrations
For More Information

http://search.oracle.com

Please visit our demo booth and session (S291507)
Thursday 11.30-.12.30

“Why, When, and How to Use Oracle Database 11g Semantic Technologies”
Acknowledgment

- Parsa Mirhaji, Narendra Kunapareddy (UTHSC)
- Holger Knublauch, Gokhan Soydan (TopQuadrant)
- Andy Seaborne (Hewlett-Packard Limited)
- Oracle DB Semantic Technologies development team & product management team