Using OWL/RDFS for Building Semantic Web Applications

Xavier Lopez, Ph.D., Director, Oracle Server Technologies
Zhe Wu, Ph.D., Consultant Member, Oracle Server Technologies
Agenda

• Introduction to Semantic Web and Oracle 11g Semantic Technologies
  • What is semantic web?
  • Business use cases
• Capabilities overview
  • Architecture/Query/Store/Inference/Java APIs
  • Scalability and performance
• Web ontology languages overview
  • RDFS/OWL
• Using RDFS/OWL in your semantic applications
• Summary
Introduction to Semantic Web and Business Use Cases
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
- 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
Case Study: National Intelligence

Information Extraction
Categorization, Feature/term Extraction

Web Resources
Web Resources
News, Email, RSS
Content Mgmt. Systems

RDF/OWL
Processed Document Collection

Ontology Engineering Modeling Process
OWL Ontologies
Domain Specific Knowledge Base

SQL/SPARQL Query
Explore

Analyst

Browsing, Presentation, Reporting, Visualization, Query

Ontology Engineering Modeling Process
OWL Ontologies
Domain Specific Knowledge Base

SQL/SPARQL Query
Explore

Analyst

Browsing, Presentation, Reporting, Visualization, Query
Data Integration Platform in Health Informatics

Enterprise Information Consumers (EICs)

Access

Run-Time Metadata

Integration Server (Semantic Knowledge base)

Deploy

Model Virtual

Relate

Model Physical

Access

LIS  CIS  HTB  HIS

Patient Care

Workforce Management

Business Intelligence

Clinical Analytics

ORACLE
Data Integration Platform in Health Informatics

Enterprise Information Consumers (EICs)

Access

Run-Time Metadata

Integration Server (Semantic Knowledge base)

Deploy

Design-Time Metadata

Access

Model Virtual

Relate

Model Physical

Patient Care

Workforce Management

Business Intelligence

Clinical Analytics

LIS

CIS

HTB

HIS

ORACLE
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

Other Data Formats
- Unstructured Content
- RSS, email
- Other Data Formats

Data Sources

Partners
Oracle 11g RDF/OWL Graph Data Management

- Oracle 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 and Advanced Compression. 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 Technology Partners
Integrated Tools and Solution Providers:
Commitment to W3C Semantic Standards

• Our implementation entirely based on W3C standards (RDF, RDFS, OWL)
  • SPARQL support through Jena
• Members of following W3C Web Semantic Activities:
  • W3C Data Access Working Group (DAWG)
  • W3C OWL Working group
  • W3C Semantic Web Education & Outreach (SWEO)
  • W3C Health Care & Life Sciences Interest Group (HCLS)
  • W3C Multimedia Semantics Incubator group
  • W3C Semantic Web Rules Language (SWRL)
Oracle 11g Semantic Technologies
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.
Capabilities Overview

INFER
- RDF/S
- OWL
- User defined rules

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

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

- RDF/OWL data
- Ontologies & rule bases
- Relational data
Store Semantic Data

- Native graph data store in Oracle database
  - Implemented using relational tables/views
  - Optimized for semantic data
- Scales to very large datasets
  - No limits to amount of data that can be stored
- Stored along with other relational data
  - Leverages decades of experience
  - Can be combined with other relational data
    - Business Data
    - XML
    - Location
    - Images, Video
Store Semantic Data: APIs

- **Incremental DMLs (small number of changes)**
  - Insert
  - Delete
  - GraphOracleSem.add, delete

- **Batch loader**
  - BatchImport
  - OracleBulkUpdateHandler.addInBatch(…)

- **Bulk loader (large number of changes)**
  - sem_apis.bulk_load_from_staging_table(…)
  - OracleBulkUpdateHandler.addInBulk(…)

Recommended loading method for very small number of triples

Recommended loading method for very large number of triples
Infer Semantic Data

• Native inferencing in the database for
  • RDF, RDFS, and a rich subset of OWL semantics (OWLSIF, OWLPRIME, RDFS++)
  • User-defined rules

• Forward chaining.
  • New relationships/triples are inferred and stored ahead of query time
  • Removes on-the-fly reasoning and results in fast query times

• Proof generation
  • Show one deduction path
OWL Subsets Supported

- **RDFS++**
  - RDFS plus owl:sameAs and owl:InverseFunctionalProperty
- **OWLSIF (OWL with IF semantics)**
  - Based on Dr. Horst’s pD* vocabulary
- **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
Infer Semantic Data: APIs

- **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
  )
  - Java API: GraphOracleSem.performInference()

**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**

**Recommended API for inference**

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

• Choice of SQL or SPARQL

• SPARQL-like graph queries can be embedded in SQL
  • Key advantages
    • Graph queries can be integrated with enterprise relational data
    • Graph queries can be enhanced with relational operators.
      • E.g. replace, substr, concatenation, to_number, …

• Jena Adaptor for Oracle can be used, includes a full SPARQL API

• Oracle plans to natively support SPARQL
Query Semantic Data: APIs

• Graph query using SEM_MATCH

```sql
select g.s, t.frequency from table(sem_match (       -- query graph + relational
    '?s <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
    <http://www.w3.org/2002/07/owl#Class>'),
    sem_models('nci1'), null, null, null)) g,  terms t
where substr(g.s, instr(g.s,'#',-1)+1)= t.subject;
```

```sql
select o from table(sem_match (                       -- query original graph + inferred
    '<http://www.mindswap.org/2003/nciOncology.owl#Finger_Fracture>
    <http://www.w3.org/2000/01/rdf-schema#subClassOf> ?o>',
    sem_models('nci'), sem_rulebases('owlprime'), null, null));
```

```sql
select o from table(sem_match (                        -- query multi-graphs + inferred
    '<http://www.mindswap.org/2003/nciOncology.owl#Finger_Fracture>
    <http://www.w3.org/2000/01/rdf-schema#subClassOf> ?o>',
    sem_models('nci', 'gene'), sem_rulebases('owlprime'), null, null, null, 'ALLOW_DUP=T'));
```

• Graph query using Jena Adaptor

http://www.oracle.com/technology/obe/11gr1_db/datamgmt/nci_semantic_network/nci_Semantics_les01.htm
Query Semantic Data: 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:**

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

**Query Semantic Data:**

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

**Traditional Syntactic** query against relational data

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

```
SELECT p_id, diagnosis
FROM Patients
WHERE diagnosis = '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!**
Query Semantic Data: Semantic Operators

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

Query:

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

**Patients diagnosis table**

<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 Semantic Data**

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

**Traditional Syntactic** query against relational data 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;
```
Java APIs: Jena Adaptor

- Implements Jena’s Graph/Model/BulkUpdateHandler/… APIs
- “Proxy” like design
  - Data not cached in memory for scalability
  - SPARQL query converted into SQL and executed inside DB
    - A SPARQL with just conjunctive patterns is converted into a single SEM_MATCH query
- Usage example
  
  ```java
  String queryString =
      " PREFIX foaf: <http://xmlns.com/foaf/0.1/> " +
      "SELECT ?person WHERE { ?person foaf:firstName "Julie" . } ";

  QueryExecution qexec = QueryExecutionFactory.create(
      QueryFactory.create(queryString), modelOracleSem);
  ```
Java APIs: Jena Adaptor (2)

- Allows various data loading
  - Bulk/Batch/Incremental load RDF or OWL (in N3, RDF/XML, N-TRIPLE etc.) with strict syntax verification and long literal support

- Integrates Oracle 11gR1 RDF/OWL with Top Braid Composer

- Makes integration with external reasoners easier
  - E.g. PelletInfGraph can be created on top of GraphOracleSem

- 2nd version released on OTN (Aug 2008)
11gR1 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>'));`

- Collect statistics using `exec sem_apis.analyze_model('family');`

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

- Collect statistics using `exec sem_apis.analyze_rules_index('family_idx');`

- Query both original model and inferred data
  ```
  select p, o 
  from table(sem_match('(<http://www.example.org/family/Matt> ?p ?o)', sem_models('family'), sem_rulebases('owlprime'), null, null));
  ```

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.

Important for performance
Scalability and Performance
# Bulk Loader Performance on Desktop PC

<table>
<thead>
<tr>
<th>Ontology size</th>
<th>Time</th>
<th>Space (in GB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>LUBM50 6.9 million</td>
<td>8 min</td>
<td>1 min</td>
</tr>
<tr>
<td>LUBM1000 138 million</td>
<td>3 hr 25 min</td>
<td>19 min</td>
</tr>
<tr>
<td>LUBM8000 1,106 million</td>
<td>30 hr 43 min</td>
<td>2 h 35 m</td>
</tr>
<tr>
<td>UniProt (old) 207 million</td>
<td>4 hr 40 min</td>
<td>30 m</td>
</tr>
</tbody>
</table>

- Results collected on a single CPU PC (3GHz), 4GB RAM, 7200rpm SATA 3.0Gbps, 32 bit Linux. RDBMS 11.1.0.6
- Empty network is assumed

[1] Uses flags=>' VALUES_TABLE_INDEX_REBUILD '
[2] Less time for minimal syntax check.
[3] More time is needed when RDF values used in N-Triple file are checked for correctness.
[5] Staging table has table compression enabled.
## Inference Performance on Desktop PC

<table>
<thead>
<tr>
<th>Ontology (size)</th>
<th>RDFS</th>
<th>OWLPrime</th>
<th>OWLPrime + Pellet on TBox</th>
</tr>
</thead>
<tbody>
<tr>
<td>(after duplicate elimination)</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>

### OWLPrime Inference (with Pellet on TBox)

![Graph showing OWLPrime Inference Time (minutes) vs. Number of universities]

- **2.52k triples/s**
- **6.49k triples/s**

### As a reference (not a comparison)

BigOWLIM *loads, inferences, and stores* (2GB RAM, P4 3.0GHz,)
- LUBM50 in 11 minutes (JAVA 6, -Xmx192), \(^1\)
- LUBM1000 in 11h 20min (JAVA 5, -Xmx1600), \(^1\)

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

---

# Inference Performance on Desktop PC

## 1 Billion Triples

### OWLPrime + Pellet on TBox

<table>
<thead>
<tr>
<th>Ontology (size)</th>
<th>2GB Physical Memory</th>
<th>4GB Physical Memory</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>#Triples inferred (millions)</td>
<td>Time</td>
</tr>
<tr>
<td>LUBM 8000</td>
<td>521.7</td>
<td>88.5hr</td>
</tr>
<tr>
<td>1.068 billion+ triples</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### OWLPrime Inference Time-Memory Tradeoff

![Graph showing time-memory tradeoff](chart.png)

- Results collected on a single CPU PC (3GHz), 2~4GB RAM, 2 Local 7200 RPM SATA Disks
- Check inference best practice document on OTN for more details [1]

---

Inference Performance on Desktop PC
1 Billion Triples

- Using better I/O by adding two commodity disks

<table>
<thead>
<tr>
<th>Ontology (size)</th>
<th>OWLPrime + Pellet on TBox</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>2 SATA 7200RPM DISKS, 4GB RAM</td>
</tr>
<tr>
<td></td>
<td>#Triples inferred (millions)</td>
</tr>
<tr>
<td>LUBM 8000</td>
<td>1.068 billion+ triples</td>
</tr>
</tbody>
</table>

- Using the same setup, UniProt (956.9 million triples) OWLPrime inference takes 2hr 29 minutes to generate 23.3 million new triples
Inference Performance on Server

**Using server-class machine with much better I/O (flash-based storage), CPU and memory**

<table>
<thead>
<tr>
<th>Ontology (size)</th>
<th>OWLPrime + Pellet on TBox</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Single CPU, 2GB RAM</td>
</tr>
<tr>
<td></td>
<td>#Triples inferred</td>
</tr>
<tr>
<td>LUBM1000</td>
<td>(millions)</td>
</tr>
<tr>
<td>133.6 million triples</td>
<td>65.25</td>
</tr>
</tbody>
</table>

OWLPrime Inference Comparison

![Graph showing inference time comparison between Commodity Linux Box and Server Box (with NAND hard disc).](image-url)
Inference Performance on Server Going Parallel

- Server class machine: 16 Cores, NAND based flash storage, 32GB RAM, 64 bit Linux
- LUBM1000 benchmark. One execution of owl:someValuesFrom rule generates 12.99 million triples.

Parallel inference is planned post 11gR1
## Query Performance on Desktop PC

<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>OWLPrime</td>
<td># answers</td>
</tr>
<tr>
<td>Complete?</td>
<td>Y</td>
</tr>
<tr>
<td>Time (sec)</td>
<td>0.09</td>
</tr>
<tr>
<td>OWLPrime + Pellet on TBox</td>
<td># answers</td>
</tr>
<tr>
<td>Complete?</td>
<td>Y</td>
</tr>
<tr>
<td>Time (sec)</td>
<td>0.09</td>
</tr>
</tbody>
</table>

- Results collected on a single CPU PC (3GHz), 2GB RAM (1.4G dedicate to DB), Multiple Disks over NFS
Query Performance on Desktop PC (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>Complete?</td>
<td>N</td>
</tr>
<tr>
<td>Time (sec)</td>
<td>2.76</td>
</tr>
<tr>
<td>OWLPrime + Pellet on TBox</td>
<td># answers</td>
</tr>
<tr>
<td>Complete?</td>
<td>Y</td>
</tr>
<tr>
<td>Time (sec)</td>
<td>3.9</td>
</tr>
</tbody>
</table>

- Results collected on a single CPU PC (3GHz), 2GB RAM (1.4G dedicate to DB), Multiple Disks over NFS
Query Performance on Server Going Parallel

- Server class machine: 16 Cores, NAND based flash storage, 32GB RAM, 64 bit Linux
- LUBM query benchmark (LUBM1000).
Ontology Language Overview
Basic Elements of RDF

• Instances
  E.g. :John, :MovieXYZ, :PurchaseOrder432

• Classes
  • Class represents a group/category/categorization of instances
  E.g. :John rdf:type :Student

• Properties
  • Linking data together
  E.g. :John :brother :Mary,
    :John :hasAge “33”^^xsd:integer.
RDF Schema (RDFS)

- Core language constructs
  - rdfs:subClassOf
    - :A rdfs:subClassOf :B → instance of A is also instance of B
  - rdfs:subPropertyOf (property transfer)
    - :firstAuthor rdfs:subPropertyOf :Author
    - skos:prefLabel rdfs:subPropertyOf rdfs:label
  - rdfs:domain and rdfs:range (specify how a property can be used)
    - E.g. :performSurgeryOn rdfs:domain :Surgeon
          :performSurgeryOn rdfs:range :Patient
  - rdfs:label, seeAlso, isDefinedBy, ...
    - :Jack rdfs:seeAlso http://.../Jack_Blog
Web Ontology Language (OWL)

• More expressive compared to RDFS

• Property related constructs
  • owl:inverseOf
    • E.g. :write owl:inverseOf :authoredBy
  • owl:SymmetricProperty
    • :relatedTo rdf:type owl:SymmetricProperty
    • foaf:knows is not defined as a symmetric property!
  • owl:TransitiveProperty
    • :partOf rdf:type owl:TransitiveProperty.
    • skos:broader rdf:type owl:TransitiveProperty
  • owl:equivalentProperty
  • owl:FunctionalProperty
    • :hasBirthMother rdf:type owl:FunctionalProperty
  • owl:InverseFunctionalProperty
    • foaf:mbox rdf:type owl:InverseFunctionalProperty

• Instances (owl:sameAs, owl:differentFrom)
OWL

• Class related constructs
  • owl:equivalentClass
  • owl:disjointWith
    • :Boys  owl:disjointWith  :Girls
  • owl:complementOf
    • :Boys  owl:complementOf  :Non_Boys
  • owl:unionOf, owl:intersectionOf, owl:oneOf
  • owl:Restriction is used to define a class whose members have certain restrictions w.r.t a property
    • owl:someValuesFrom
    • owl:allValuesFrom
    • owl:hasValue
OWL

- Class related constructs
  - owl:equivalentClass
  - owl:disjointWith
    - :Boys owl:disjointWith :Girls
  - owl:complementOf
    - :Boys owl:complementOf :Non-Boys
  - owl:unionOf, owl:intersectionOf
  - owl:Restriction is used to define a class whose members have certain restrictions w.r.t a property
    - owl:someValuesFrom
    - owl:allValuesFrom
    - owl:hasValue

```owl
:ApprovedPurchaseOrder owl:equivalentClass
[ a owl:Restriction ;
  owl:onProperty :approvedBy ;
  owl:someValuesFrom :Manager ]

:PO1 :approvedBy :managerXyz
:managerXyz rdf:type :Manager
⇒ :PO1 rdf:type :ApprovedPurchaseOrder
```
OWL

- Class related constructs
  - owl:equivValClass
  - owl:disjointWith
    - :Boys owl:disjointWith :Girls
  - owl:complementOf
    - :Boys owl:complementOf :Non_Boys
  - owl:unionOf, owl:intersectionOf, owl:oneOf
    - owl:Restriction is used to define a class whose members have certain restrictions
      - owl:someValuesFrom
      - owl:allValuesFrom
      - owl:hasValue

```owl
owl:allValuesFrom

:Vegetarian rdfs:subClassOf
  [ a owl:Restriction ;
  owl:onProperty :eats; 
  owl:allValuesFrom :VegetarianFood ]

:Jen rdf:type :Vegetarian .
:Jen :eats :Marzipan .

⇒ :Marzipan rdf:type :VegetarianFood .
```

We SHOULD not use :eats rdfs:range :VegetarianFood
OWL

• Class related constructs
  • owl:equivalentClass
  • owl:disjointWith
    • :Boys owl:disjointWith :Girls
  • owl:complementOf
    • :Boys owl:complementOf :Non_Boys
  • owl:unionOf, owl:intersectionOf, owl:oneOf
  • owl:Restriction is used to define a class whose members have certain restrictions w.r.t a property
    • owl:someValuesFrom
    • owl:allValuesFrom
    • owl:hasValue

owl:hasValue

• :HighPriorityItem owl:equivalentClass [ a owl:Restriction ;
  owl:onProperty :hasPriority ;
  owl:hasValue :High ]

:Item1 rdf:type :HighPriorityItem .
leftrightarrow :Item1 :hasPriority :High
OWL

• Class related constructs
  • Cardinality restrictions constrain the number of distinct individuals that can associate with a class instance via a particular property
    • owl:minCardinality
    • owl:maxCardinality
    • owl:cardinality
  E.g. To express that a basketball game has at least 2 players:
    :BasketBallGame rdfs:subClassOf
    [ a owl:Restriction;
      owl:onProperty  :hasPlayer;
      owl:minCardinality  2 ]

• Others
  • DatatypeProperty, AnnotationProperty, OntologyProperty,…
Using RDFS/OWL in your semantic applications
Using RDFS

- Approximate set intersection with $C \subseteq A \cap B$
  
  $C \text{ rdfs:subClassOf } A$
  
  $C \text{ rdfs:subClassOf } B$

- Approximate property intersection with $p \subseteq r \cap s$
  
  $p \text{ rdfs:subPropertyOf } r$
  
  $p \text{ rdfs:subPropertyOf } s$

- Approximate set union with $A \cup B \subseteq C$
  
  $A \text{ rdfs:subClassOf } C$
  
  $B \text{ rdfs:subClassOf } C$

- Approximate property union with $r \cup s \subseteq p$
  
  $r \text{ rdfs:subPropertyOf } p$
  
  $s \text{ rdfs:subPropertyOf } p$

- Term reconciliation
  
  - Property transfer using :author rdfs:subPropertyOf dc:creator.
  
    :Researcher rdfs:subClassOf :Analyst .
Using RDFS

- Approximate set intersection with \( C \subseteq A \cap B \)
  \[
  C \text{ rdfs:subClassOf } A \\
  C \text{ rdfs:subClassOf } B
  \]

- Approximate property intersection with \( p \subseteq r \cap s \)
  \[
  p \text{ rdfs:subPropertyOf } r \\
  p \text{ rdfs:subPropertyOf } s
  \]

- Approximate set union with \( A \cup B \subseteq C \)
  \[
  A \text{ rdfs:subClassOf } C \\
  B \text{ rdfs:subClassOf } C
  \]

- Approximate property union with \( r \cup s \subseteq p \)
  \[
  r \text{ rdfs:subPropertyOf } p \\
  s \text{ rdfs:subPropertyOf } p
  \]

- Term reconciliation
  - Different property names used
    - One library uses term \( \text{library1:borrows} \)
    - Another library uses term \( \text{library2:checkOut} \)
  
  Solution:
  - \( \text{library1:borrows} \text{ rdfs:subPropertyOf } \text{library2:checkOut} \)
  - \( \text{library2:checkOut} \text{ rdfs:subPropertyOf } \text{library1:borrows} \)
Using RDFS

• Approximate set intersection with \( C \subseteq A \cap B \)
  \[
  C \text{ rdfs:subClassOf } A \\
  C \text{ rdfs:subClassOf } B 
  \]

• Approximate property intersection with \( p \subseteq r \cap s \)
  \[
  p \text{ rdfs:subPropertyOf } r \\
  p \text{ rdfs:subPropertyOf } s 
  \]

• Approximate set union with \( A \cup B \subseteq C \)
  \[
  A \text{ rdfs:subClassOf } C \\
  B \text{ rdfs:subClassOf } C 
  \]

• Approximate property union with \( r \cup s \subseteq p \)
  \[
  r \text{ rdfs:subPropertyOf } p \\
  s \text{ rdfs:subPropertyOf } p 
  \]

• Term reconciliation
  • Property transfer using: \( \text{author} \text{ rdfs:subPropertyOf } \text{dc:creator} \)
  • Equivalent classes using:
    - \( :\text{Researcher} \text{ rdfs:subClassOf } :\text{Analyst} \)
    - \( :\text{Analyst} \text{ rdfs:subClassOf } :\text{Researcher} \)
Using OWL

• **Merge Information using simple constructs**

  • **Using functional property**
    
    ![Use functional property example](image)

    :hasBirthMother rdf:type owl:FunctionalProperty
    :John :hasBirthMother :Mary
    :John :hasBirthMother :Jill
    :Mary owl:sameAs :Jill
    all information about :Mary and :Jill will be merged

  • **Using inverse functional property**
    
    ![Use inverse functional property example](image)

    :directManagerOf rdf:type owl:InverseFunctionalProperty
    :Jack :directManagerOf :Eddie
    :CoolJack :directManagerOf :Eddie
    :Jack owl:sameAs :CoolJack
    all information about :Jack and :CoolJack will be merged

  • **Using owl:sameAs directly**
    
    ![Use owl:sameAs directly example](image)

    <http://purl.uniprot.org/citations/2450746> owl:sameAs
    <http://purl.uniprot.org/medline/88166744> .
Using OWL

• **Use Classes as property values**
  
  :BookXyz dc:subject :TigerSubject .
  :TigerSubject rdf:type :Subject .
  :TigerSubject rdfs:seeAlso :Tiger .
  :Tiger rdfs:subClassOf :Animal .

• **Representing is_part_of relationship**

  is_part_of
    a owl:TransitiveProperty, owl:ObjectProperty ;
    rdfs:domain Item;
    rdfs:range Item;
    owl:inverseOf has_part ;

• **minCardinality 0 can be used to denote “may” or “optional” relationship.**

• **No specific semantics though.**

2. [http://www.w3.org/TR/swbp-classes-as-values/](http://www.w3.org/TR/swbp-classes-as-values/)
Common Pitfalls

• Domain/range have intersection semantics
  :eats :range :VegetarianFood .
  :eats :range :Meat
  ➔ :eats :range (:VegetarianFood ∩ :Meat)

• owl:complementOf is strong
  Is :Male owl:complementOf :Female?

• Transitive properties should not be functional or inverse functional. It is legal but does not make sense
  :relatedTo rdf:type owl:TransitiveProperty .
  :relatedTo rdf:type owl:FunctionalProperty .
  :John :relatedTo :Jack .
  :Jack :relatedTo :Mary
  ➔ :Jack owl:sameAs :Mary
Common Pitfalls

• owl:allValuesFrom **does not imply** existence
  :Carnivore owl:equivalentClass [ a owl:Restriction;
      owl:onProperty :eats;
      owl:allValuesFrom :Meat]

  :John rdf:type Carnivore **does not imply**
  :John :eats [ a :Meat]

• owl:allValuesFrom and Open World Assumption (OWA)
  :John :eats :Steak
  :Steak rdf:type :Meat **does not imply** :John rdf:type :Carnivore

• owl:min/maxCardinality and OWA and absence of
  Unique Name Assumption (UNA)
Summary

• 11g provides a solid, efficient, and scalable semantic data management
  • Store
  • Query (graph query and ontology-assisted query)
  • Inference
• Java APIs make application building easier
• RDFS/OWL are powerful knowledge modeling languages.
  • Be careful with consequence of OWA and absence of UNA
• Future direction
  • Even better performance
  • Richer semantics
  • More application/tools integrations
For More Information

http://search.oracle.com

Please visit our demo booth
## Semantics at OOW 2008 - Sessions

<table>
<thead>
<tr>
<th>Date/Time</th>
<th>Title</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sunday, Sept 21</td>
<td><em>[Get the Most from Oracle Database 11g Semantic Technology: Best Practices]</em></td>
<td>Marriott Salon 12/13</td>
</tr>
<tr>
<td>1:15-2:15 p.m.</td>
<td><em>[Using RDFS/OWL for Building Semantic Web Applications]</em></td>
<td>Marriott Salon 12/13</td>
</tr>
<tr>
<td>Monday, Sept 22</td>
<td><em>[Oracle Database 11g Production Case Study: Teranode and Pfizer Semantic/Relational Approach to Science Collaboration]</em></td>
<td>Marriott Salon 12/13</td>
</tr>
<tr>
<td>4:00-5:00 p.m.</td>
<td><em>[Oracle Database 11g Production Case Study: Eli Lilly and Company—Semantic Technologies from a DBA's Point of View]</em></td>
<td>Marriott Salon 01</td>
</tr>
</tbody>
</table>

- DEMOgrounds - Database – Moscone West

Oracle Semantic Database Technologies - *Pod L14*
Appendix
<table>
<thead>
<tr>
<th>Rulebase Name</th>
<th>RDFS/OWL constructs included</th>
</tr>
</thead>
<tbody>
<tr>
<td>RDFS++</td>
<td>all RDFS vocabulary constructs, owl:InverseFunctionalProperty, owl:sameAs</td>
</tr>
<tr>
<td>OWLSTIF</td>
<td>all RDFS vocabulary constructs, owl:FunctionalProperty, owl:InverseFunctionalProperty, owl:SymmetricProperty, owl:TransitiveProperty, owl:sameAs, owl:inverseOf, owl:equivalentClass, owl:equivalentProperty, owl:hasValue, owl:someValuesFrom, owl:allValuesFrom</td>
</tr>
</tbody>
</table>
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)

### Antecedents vs. Consequences

<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)

- Option 2: Separation of schema (TBox) and instance (ABox) reasoning through 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
What is Semantic Web?

- **Basic Technologies**
  - **URI**
    - Uniform resource identifier
  - **RDF**
    - Resource description framework
  - **RDFS**
    - RDF Schema
  - **OWL**
    - Web ontology language

---

- **User Interface & Applications**
- **Trust**
- **Proof**
- **Unifying Logic**
- **Query: SPARQL**
- **Ontology:**
  - **OWL**
- **Rule:**
  - **RIF**
- **Data interchange:**
  - **RDF**
  - **XML**
  - **URI/IRI**
- **Standards based**