11gR1 OWLPrime

Oracle New England Development Center
Zhe Wu
alan.wu@oracle.com, Ph.D.
Consultant Member of Technical Staff
Dec 2007
Agenda

- Background
  - 10gR2 RDF
  - 11gR1 RDF/OWL
- 11gR1 OWL support
  - RDFS++, OWLSIF, OWLPrime
- Inference design & implementation in RDBMS
- Performance
- Completeness evaluation through queries
Oracle 10gR2 RDF

- **Storage**
  - Use DMLs to insert triples incrementally
    - `insert into rdf_data values (... sdo_rdf_triple_s(1, '<subject>', '<predicate>', '<object>'))`;
  - Use Fast Batch Loader with a Java interface

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

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

- **Shipped in 2005**
Oracle 11gR1 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
    • Inference

• **Shipped** (Linux/Windows platform) in 2007

• **Java API support**
  • Oracle Jena Adaptor (released on OTN) implemented HP Jena APIs.
  • Sesame (forthcoming)
Oracle 11gR1 OWL is a scalable, efficient, forward-chaining based reasoner that supports an expressive subset of OWL-DL
Why?

• Why inside RDBMS?
  • Size of semantic data grows really fast.
  • RDBMS has transaction, recovery, replication, security, …
  • RDBMS is **efficient** in processing queries.

• Why OWL-DL subset?
  • Have to **scale** and 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
    • 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
    • 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
Semantics Characterized by Entailment Rules

- RDFS has 14 entailment rules defined in the SPEC.
  - E.g. rule: `{aaa rdfs:domain XXX .
    uuu aaa yyy .} \ {uuu rdf:type XXX .}

- OWLPrime has 50+ entailment rules.
  - E.g. rule: `{aaa owl:inverseOf bbb .
    bbb rdfs:subPropertyOf ccc .
    ccc owl:inverseOf ddd .} \ {aaa rdfs:subPropertyOf ddd .}

  `{xxx owl:disjointWith yyy .
  a rdf:type xxx .
  b rdf:type yyy .} \ {a owl:differentFrom b .}

- These rules have efficient implementations in RDBMS
Applications of Partial DL Semantics

- 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

1 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>?x :parentOf ?y .</td>
<td>\</td>
</tr>
</tbody>
</table>

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

  <owl:Class rdf:ID="FemaleAstronaut">
  <rdfs:label>chair</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 in TBox and ABox reasoning**
  - 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
11g OWL Inference PL/SQL API

- **SEM_API.CREATES_ENTAILMENT**
  - Index_name
  - sem_models(‘GraphTBox’, ‘GraphABox’, …),
  - sem_rulebases(‘OWLPrime’),
  - passes,
  - Inf_components,
  - Options

  Use “PROOF=T” to generate inference proof

- **SEM_API.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

Above APIs can be invoked from Java clients through JDBC

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

• Give users more control over inference process
  • Selective inference (component based)
    • Allows more focused inference.
    • E.g. give me only the subClassOf hierarchy.

• Set number of passes
  • Normally, inference continue till no further new triples found
  • Users can set the number of inference passes to see if what they are interested has already been inferred
  • E.g. I want to know whether this person has more than 10 friends

• Set tablespaces used, parallel index build

• Change statistics collection scheme
11gR1 OWL Usage Example

- 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
  - …
- 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
  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.
Separate TBox and ABox Reasoning

- Utilize Pellet and Oracle’s implementation of Jena Graph API
  - Create a Jena Graph with Oracle backend
  - Create a PelletInfGraph on top of it
  - PelletInfGraph.getDeductionsGraph
- Issues encountered: no subsumption for anonymous classes from Pellet inference.

```xml
<owl:Class rdf:ID="Employee">
  <owl:union rdf:parseType="Collection">
    <owl:Restriction>
      <owl:onProperty rdf:resource="#reportsTo" />
      <owl:someValuesFrom>
        <owl:Class rdf:about="#Manager" />
      </owl:someValuesFrom>
    </owl:Restriction>
  </owl:union>
  <owl:Class rdf:about="#CEO" />
</owl:Class>
```

**Solution:** create intermediate named classes

- Similar approach applies to Racer Pro, KAON2, Fact, etc. through DIG
Soundness

- Soundness of 11g OWL verified through
  - Comparison with other well-tested reasoners
  - Proof generation
    - A proof of an assertion consists of a rule (name), and a set of assertions which together deduce that assertion.
    - Option “PROOF=T” instructs 11g OWL to generate proof

```
TripleID1 :emailAddress rdf:type owl:InverseFunctionaProperty .
TripleID2 :John :emailAddress :John_at_yahoo_dot_com .
TripleID3 :Johnny :emailAddress :John_at_yahoo_dot_com .
:John owl:sameAs :Johnny (proof := TripleID1, TripleID2, TripleID3, “IFP”)
```
Design & Implementation
Design Flow

• Extract rules
• Each rule implemented individually using SQL
• Optimization
  • SQL Tuning
  • Rule dependency analysis
  • Dynamic statistics collection
• Benchmarking
  • LUBM
  • UniProt
  • Randomly generated test cases

TIP
• Avoid incremental index maintenance
• Partition data to cut cost
• Maintain up-to-date statistics
Execution Flow

Background- Storage scheme
- Two major tables for storing graph data
- VALUES table stores mapping from URI (etc) to integers
- IdTriplesTable stores basically SID, PID, OID

<table>
<thead>
<tr>
<th>VALUE</th>
<th>ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>http://…/John</td>
<td>123</td>
</tr>
</tbody>
</table>

"Implementing an Inference Engine for RDFS/OWL Constructs and User-Defined Rules in Oracle" ICDE 2008
Entailment Rule Implementation In SQL

Example Rule

```
aaa owl:inverseOf bbb .
bbb rdfs:subPropertyOf ccc .
ccc owl:inverseOf ddd .
```

SQL Implementation

```
select distinct T1.SID sid,
       ID(rdfs:subPropertyOf) pid,
       T3.OID oid
from <IVIEW> T1,
     <IVIEW> T2,
     <IVIEW> T3
where T1.PID=ID(owl:inverseOf)
  and T2.PID=ID(rdfs:subPropertyOf)
  and T3.PID=ID(owl:inverseOf)
  and T1.OID=T2.SID
  and T2.OID=T3.SID
  and NOT EXISTS (select 1 from <IVIEW> m
                   where m.SID=T1.SID
                   and m.PID=ID(rdfs:subPropertyOf)
                   and m.OID=T3.OID)
```
Performance Evaluation
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
Machine/Database Configuration

- NFS configuration
  - `rw,noatime,bg,intr,hard,timeo=600,wsizer=32768,rsize=32768,tcp`
- Hard disks: 320GB SATA 7200RPM (much slower than RAID). Two on each PC
- Database (11g release on Linux 32bit platform)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>db_block_size</td>
<td>8192</td>
<td>size of a database block</td>
</tr>
<tr>
<td>memory_target</td>
<td>1408M</td>
<td>memory area for a server process + memory area for storing data and control information for the database instance</td>
</tr>
<tr>
<td>workarea_size_policy</td>
<td>auto</td>
<td>enables automatic sizing of areas used by memory intensive processes</td>
</tr>
<tr>
<td>statistics_level</td>
<td>TYPICAL</td>
<td>enables collection of statistics for database self management</td>
</tr>
</tbody>
</table>
Tablespace Configuration

- Created bigfile (temporary) tablespaces
- LOG files located on semperf3 diskA

<table>
<thead>
<tr>
<th>Tablespace</th>
<th>Machine</th>
<th>Disk</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>USER_TBS</td>
<td>semperf2</td>
<td>diskA</td>
<td>for storing user’s application table. It is only used during data loading. Not relevant for inference.</td>
</tr>
<tr>
<td>Temporary Tablespace</td>
<td>semperf1</td>
<td>diskB</td>
<td>Oracle’s temporary tablespace is for intermediate stages of SQL execution.</td>
</tr>
<tr>
<td>UNDO</td>
<td>semperf2</td>
<td>diskB</td>
<td>for undo segment storage</td>
</tr>
<tr>
<td>SEM_TS</td>
<td>semperf3</td>
<td>diskB</td>
<td>for storing graph triples</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

## Query Answering After Inference

<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></td>
<td>Complete?</td>
</tr>
<tr>
<td>OWLPrime + Pellet on TBox</td>
<td># answers</td>
</tr>
<tr>
<td></td>
<td>Complete?</td>
</tr>
</tbody>
</table>
### Query Answering After Inference (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></td>
</tr>
<tr>
<td><strong>OWLPrime</strong></td>
<td># answers</td>
</tr>
<tr>
<td>Complete?</td>
<td>N</td>
</tr>
<tr>
<td><strong>OWLPrime + Pellet on TBox</strong></td>
<td># answers</td>
</tr>
<tr>
<td>Complete?</td>
<td>Y</td>
</tr>
</tbody>
</table>
## Query Answering After Inference (3)

<table>
<thead>
<tr>
<th>Ontology LUBM1000</th>
<th>LUBM Benchmark Queries</th>
</tr>
</thead>
<tbody>
<tr>
<td>133 million &amp; 60+ 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>OWLPrime + Pellet on TBox</td>
<td># answers</td>
</tr>
<tr>
<td>Complete?</td>
<td>Y</td>
</tr>
</tbody>
</table>

---

**OWLPrime**

- **# answers**
  - Q1: 4
  - Q2: 2528
  - Q3: 6
  - Q4: 34
  - Q5: 719
  - Q6: 7924765
  - Q7: 59

- **Complete?**
  - Y (Q1)
  - Unknown (Q2)
  - Y (Q3)
  - Y (Q4)
  - Y (Q5)
  - N (Q6)
  - N (Q7)

**OWLPrime + Pellet on TBox**

- **# answers**
  - Q1: 4
  - Q2: 2528
  - Q3: 6
  - Q4: 34
  - Q5: 719
  - Q6: 10447381
  - Q7: 67

- **Complete?**
  - Y (Q1)
  - Unknown (Q2)
  - Y (Q3)
  - Y (Q4)
  - Y (Q5)
  - Unknown (Q6)
  - Y (Q7)
### Query Answering After Inference (4)

<table>
<thead>
<tr>
<th>Ontology</th>
<th>LUBM Benchmark Queries</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>LUBM1000</strong></td>
<td>Q8</td>
</tr>
<tr>
<td>133 million &amp; 60+ million inferred</td>
<td>5916</td>
</tr>
<tr>
<td><strong>Complete?</strong></td>
<td>N</td>
</tr>
<tr>
<td><strong>OWLPrime</strong></td>
<td>7790</td>
</tr>
<tr>
<td><strong>Complete?</strong></td>
<td>Y</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Ontology</th>
<th>LUBM Benchmark Queries</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>OWLPrime + Pellet on TBox</strong></td>
<td>7790</td>
</tr>
<tr>
<td><strong>Complete?</strong></td>
<td>Y</td>
</tr>
</tbody>
</table>
For More Information

http://search.oracle.com

semantic technologies

or

http://www.oracle.com/