Building Enterprise Applications With Oracle Database 11g Semantic Technologies

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

• Introduction to Semantic Web and Oracle Database 11g Semantic Technologies
  • What is Semantic Web?
  • Business use cases

• Overview of release 11.1 Capabilities
  • Architecture/Query/Store/Inference/Java APIs
  • Tips and best practices

• Overview of planned features for the upcoming release
  • Query/Inference/Utility/Enterprise features

• Performance and scalability evaluation
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.
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

Processed Document Collection
- RDF/OWL

Ontology Engineering Modeling Process
- OWL Ontologies

Domain Specific Knowledge Base

SQL/SPARQL Query

Explore

Analyst

Browsing, Presentation, Reporting, Visualization, Query

Web Resources
- News, Email, RSS

Content Mgmt. Systems
Data Integration Platform in Health Informatics

Enterprise Information Consumers (EICs)

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

Design-Time Metadata

Access

Run-Time Metadata

Integration Server (Semantic Knowledge base)

Deploy

Access

LIS
CIS
HTB
HIS

Model Virtual

Relate

Model Physical

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

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

Partners

Partners
Oracle Database 11g RDF/OWL Graph Data Management

• Oracle database 11g is the leading commercial database with native RDF/OWL data management

• Scalable & secure platform for wide-range of semantic applications

• Readily scales to ultra-large repositories (+1 billion)

• Choice of SQL or SPARQL query

• Leverages Oracle Partitioning 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:
Planned New Features

- Strong security for Oracle semantic database
  - Security policies and data classification for RDF data
- Semantic indexing of documents
  - Semantic indexing of documents based on popular natural language tools
- Faster, more efficient reasoning to find new relationships
  - Parallel and incremental inference, owl:sameAs optimization
- Change management for collaboration
- Standards & open source support
  - SPARQL query support for Filter, Union
  - OWL: union, intersection, OWL 2 property chains, disjoint properties,…
  - Pellet OWL DL reasoner Integration
  - Jena V2.5
  - Java SDK for SPARQL for 3rd party integration e.g., Sesame
  - W3C Simple Knowledge Organization System (SKOS) & SNOMED ontology
Oracle Database 11g Semantic Technologies
Overview of Release 11.1 Capabilities

**INFER**
- RDF/S
- OWL
- User defined rules

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

**STORE**
- Incremental 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
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

**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 (  
  '(?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;  
-- using a predicate to tie
```

```sql
select o from table(sem_match (  
  '<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 (  
  '<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

**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:** “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
AND SEM_DISTANCE() <= 2;
```

**Traditional Syntactic query against relational data**
Java APIs: Jena Adaptor (v2.0)

• 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

• 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 Database release 11.1 RDF/OWL with tools including
  • TopBraid Composer
  • External complete DL reasoners (e.g. Pellet)

Release 11.1 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
  ```sql
  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!
Release 11.1 RDF/OWL Usage Flow in Java

- Create an Oracle object
  - `oracle = new Oracle(oracleConnection);`
- Create a GraphOracleSem Object
  - `graph = new GraphOracleSem(oracle, model_name, attachment);`
- Load data
  - `graph.add(Triple.create(...)); // for incremental triple additions`
- Collect statistics
  - `graph.analyze();`
- Run inference
  - `graph.performInference();`
- Collect statistics
  - `graph.analyzeInferredGraph();`
- Query
  - `QueryFactory.create(...);`
  - `queryExec = QueryExecutionFactory.create(query, model);`
  - `resultSet = queryExec.execSelect();`
Tips and Best Practices
Setup for Performance (1)

- Use a balanced hardware system for database
  - A single, huge physical disk for everything is not recommended.
  - Multiple hard disks tied together through ASM is a good practice
  - Make sure throughput of hardware components match up

<table>
<thead>
<tr>
<th>Component</th>
<th>Hardware spec</th>
<th>Sustained throughput</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU core</td>
<td>-</td>
<td>100 - 200 MB/s</td>
</tr>
<tr>
<td>1/2 Gbit HBA</td>
<td>1/2 Gbit/s</td>
<td>100/200 MB/s</td>
</tr>
<tr>
<td>16 port switch</td>
<td>8 * 2 Gbit/s</td>
<td>1,200 MB/s</td>
</tr>
<tr>
<td>Fiber channel</td>
<td>2 Gbit/s</td>
<td>200 MB/s</td>
</tr>
<tr>
<td>Disk controller</td>
<td>2 Gbit/s</td>
<td>200 MB/s</td>
</tr>
<tr>
<td>GigE NIC (interconnect)</td>
<td>2 Gbit/s</td>
<td>80 MB/s*</td>
</tr>
<tr>
<td>Disk (spindle)</td>
<td>2 Gbit/s</td>
<td>30 - 50 MB/s</td>
</tr>
<tr>
<td>MEM</td>
<td></td>
<td>2k-7k MB/s</td>
</tr>
</tbody>
</table>

Some numbers are from Data Warehousing with 11g and RAC presentation
Setup for Performance (2)

- Database parameters¹
  - SGA, PGA, filesystemio_options, db_cache_size, …

- Linux OS Kernel parameters
  - shmmmax, shmall, aio-max-nr, sem, …

- For Java clients using JDBC (Jena Adaptor)
  - Network MTU, Oracle SQL*Net parameters including SDU, TDU, SEND_BUF_SIZE, RECV_BUF_SIZE,
  - Linux Kernel parameters: net.core.rmem_max, wmem_max, net.ipv4.tcp_rmem, tcp_wmem, …

- No single size fits all. Need to benchmark and tune!

Common Problems and Solutions (1)

- Running out of space…
  - Solution: use BIGFILE tablespace to begin with
    e.g. create bigfile temporary tablespace tmpts
      tempfile '+DATA'
      size 512M reuse
      autoextend on next 512M maxsize <MAX_SIZE>
      EXTENT MANAGEMENT LOCAL
      ;
      ALTER DATABASE DEFAULT TEMPORARY TABLESPACE tmpts;

- Need to convert data from relational, RDF/XML, N3, …
  - Solution: use D2RQ, Jena, or other third party tools
  - Exercise care when inserting directly into staging table!

- Not seeing the best query plan?
  - Run sem_perf.gather_stats, sem_apis.analyze_model, GraphOracleSem.analyze, …
  - Multiple-model query: try ALLOW_DUP=T, virtual model
  - Tweak hints and/or indexes (add_sem_index, alter_sem_index_on_model/entailment).
Common Problems and Solutions (2)

- Only want to query inferred data?
  - Solution: use INF_ONLY=T in SEM_MATCH

- Are additional indexes on application table useful for query performance?
  - No (unless your query involves app table). They are **not** used by SEM_MATCH. They will slow down inserts.

- On using Jena Adaptor 2.0
  - Some FAQ based on users’ questions from UTH, Revelytix, Ebay, Brainstage, Metatomix, Hisoft, TopQuadrant, …

- What is “Could not set namespace prefix” error message?
  - Solution: drop the model and restart.
    - There is no need to create model explicitly in Jena Adaptor!

- “NoClassDefFoundError: com/hp/hpl/jena/sparql/engine/main/StageBasic”
  - Solution: use Jena 2.5.6 or wait for the next version of Jena Adaptor
Common Problems and Solutions (3)

- More on using Jena Adaptor 2.0
  - “java.lang.VerifyError: …”
    - Solution: check to see if there is an early version of pellet.jar in your classpath

- Is it a good idea to create a brand-new Oracle object to serve each request?
  - Depends on which constructor you use.
  - In a J2EE setup, create Oracle using OracleConnection is recommended.

- How to query “incomplete” inferred data from Jena
  - Solution: use QueryOptions
  - E.g. Attachment attachment = Attachment.createInstance(…, QueryOptions.ALLOW_QUERY_INCOMPLETE);
  - QueryOptions.ALLOW_QUERY_VALID_AND_DUP, and ALLOW_QUERY_INCOMPLETE_AND_DUP are useful for muti-graph queries.
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.
Overview of Planned Features (1)

• Query
  • More native SPARQL syntax support in SEM_MATCH
    • FILTER, UNION support on top of existing capabilities

• Inference
  • More semantics support including
    • owl:intersectionOf, unionOf, oneOf, W3C SKOS, SNOMED, OWL2’s
      owl:propertyChainAxiom, owl:NegativePropertyAssertion,
      owl:hasKey, owl:propertyDisjointWith, more OWL 2 RL/RDF rules,

• Performance enhancement
  • Large scale owl:sameAs handling, parallel inference, incremental
    inference

• Utilities
  • Swap, rename (model, entailment), merge models, remove duplicates
Overview of Planned Features (2)

• Enterprise features
  • Fine-grained security for semantic data
    • Security policies and data classification for RDF data

• Semantic indexing for documents
  • Entity extraction based on popular natural language tools

• Change management for collaboration
More SPARQL Support in SEM_MATCH

- **FILTER** specifies a filter expression in the graph pattern to restrict the solutions to a query
e.g., returns grandchildren info for only grandfathers in either NY or CA

```sql
SELECT x, y
FROM TABLE(SEM_MATCH(
FILTER (?z = "NY" || ?z = "CA")}}',})
```

- **UNION** matches one of alternative graph patterns
e.g., grandfathers are returned only if they are residents of NY or CA or own property in NY or CA, or if both conditions are true

```sql
SELECT x, y
FROM TABLE(SEM_MATCH(
'{{?x :grandParentOf ?y . ?x rdf:type :Male
{{?x :residentOf ?z} UNION {?x :ownsPropertyIn ?z}}
FILTER (?z = "NY" || ?z = "CA")}}',})
```

- **OPTIONAL** has already been supported in a patch on top of 11.1.0.7
Enabling New Inference Capabilities

- Enabling Parallel inference option
  ```java
  EXECUTE sem_apis.create_entailment('M_IDX',sem_models('M'),
  sem_rulebases('OWLPRIME'), sem_apis.REACH_CLOSURE, null, 'DOP=x');
  • Where ‘x’ is the degree of parallelism (DOP)
  ```

- Enabling Incremental inference option
  ```java
  EXECUTE sem_apis.create_entailment ('M_IDX',sem_models('M'),
  sem_rulebases('OWLPRIME'),null,null, 'INC=T');
  ```

- Enabling owl:sameAs option to limit duplicates
  ```java
  EXECUTE Sem_apis.create _entailment('M_IDX',sem_models('M'),
  sem_rulebases('OWLPRIME'),null,null,'OPT_SAMEAS=T');
  ```

- Enabling compact data structures
  ```java
  EXECUTE Sem_apis.create _entailment('M_IDX',sem_models('M'),
  sem_rulebases('OWLPRIME'),null,null, 'RAW8=T');
  ```

- Enabling SKOS inference
  ```java
  EXECUTE Sem_apis.create_entailment('M_IDX',sem_models('M'),
  sem_rulebases('SKOSCORE'),null,null...);
  ```
New Inference Components

- **UNION**: (OWL 1) owl:unionOf
- **INTERSECT & INTERSECTSCOH**: (OWL 1) owl:intersectionOf
- **SNOMED**: (OWL 2) Systematized Nomenclature of Medicine
- **PROPDISJH**: (OWL 2) interaction between owl:propertyDisjointWith and rdfs:subPropertyOf.
- **CHAIN**: (OWL 2) Supports chains of length 2
- **SKOSAXIOMS**: most of the axioms defined in SKOS reference
- **MBRLST**: for any resource, every item in the list given as the value of the skos:memberList property is also a value of the skos:member property.
- **SVFH**: capturing interaction between owl:someValuesFrom and rdfs:subClassOf
- **THINGH & THINGSAM**: any defined OWL class is a subclass of owl:Thing & instances of owl:Thing are equal to themselves
Utility APIs

- **SEM_APIS.remove_duplicates**
  - e.g. exec sem_apis.remove_duplicates('graph_model');

- **SEM_APIS.merge_models**
  - Can be used to clone model as well.
  - e.g. exec sem_apis.merge_models('model1','model2');

- **SEM_APIS.swap_names**
  - e.g. exec
    sem_apis.swap_names('production_model','prototype_model');

- **SEM_APIS.alter_model (entailment)**
  - e.g. sem_apis.alter_model('m1', 'MOVE', 'TBS_SLOWER');

- **SEM_APIS.rename_model (_entailment)**
Enterprise Security for Semantic Data

- RDF data security for defense and intelligence, and the commercial regulatory environment
  - Intercept and rewrite the user query to restrict the result set using additional predicates and return only “need to know” data

- Access control policies on semantic data
  - Uses Virtual Private Database feature of Oracle Database
  - Applies constraints to classes and properties
  - Restricts access to parts of the RDF graph based on the application/user context

- Data classification labels for semantic data
  - Uses Oracle Label Security option of Oracle Database
  - Assigns sensitivity labels to users and RDF data.
  - Restricts access to users having compatible access labels.
Semantic Indexing for Documents

- Links people – places – things – events to documents stored in Oracle Database though a semantic index
- Extends the power of Oracle Database to include semantic search in cross-domain queries.

- Key Components
  - Programmable API to plug-in 3rd party entity extractors
    - E.g. OpenCalais from Thomson Reuters
  - SEM_CONTAINS Operator
  - SEM_CONTAINS_SELECT Ancillary Operator
  - SemContext Index type
Semantic Indexing and Query Flow

- Extracting RDF from documents

<table>
<thead>
<tr>
<th>DocId</th>
<th>Article</th>
</tr>
</thead>
<tbody>
<tr>
<td>r1</td>
<td>1</td>
</tr>
<tr>
<td>r2</td>
<td>2</td>
</tr>
</tbody>
</table>

**Newsfeed table**

<table>
<thead>
<tr>
<th>NG</th>
<th>Subject</th>
<th>Property</th>
<th>Object</th>
</tr>
</thead>
<tbody>
<tr>
<td>r1</td>
<td>p:Marcus</td>
<td>rdf:type</td>
<td>rc:Person</td>
</tr>
<tr>
<td>r1</td>
<td>p:Marcus</td>
<td>pred:hasName</td>
<td>&quot;Marcus&quot;^^xsd:string</td>
</tr>
<tr>
<td>r1</td>
<td>p:Marcus</td>
<td>pred:hasAge</td>
<td>“38”^^xsd:integer</td>
</tr>
<tr>
<td>..</td>
<td>..</td>
<td>..</td>
<td>..</td>
</tr>
<tr>
<td>r2</td>
<td>c:AcmeCorp</td>
<td>rdf:type</td>
<td>rc:Organization</td>
</tr>
</tbody>
</table>

**RDF/XML for each document**

**Triples table**

- Semantic query through SEM_CONTAINS

```sql
SELECT docId, SEM_CONTAINS_SELECT(1) binding FROM Newsfeed
WHERE SEM_CONTAINS (article,
  '{ ?org pred:categoryName c:BusinessFinance .
    ?org pred:score   ?score .
    FILTER (?score > 0.5)}', 1 ) = 1
```
Change Mgmt./Versioning for Semantic Data

- Manage public and private versions of semantic data in database workspaces (*Workspace Manager*)

- An RDF Model is version-enabled by version-enabling its application table.

- Application table data modified within a workspace is private to the workspace until it is merged.

- SEM_MATCH queries on version-enabled models are version aware and only return relevant data.
  - New versions created only for changed data

- Versioning is provisioned for inference
Performance and Scalability Evaluation
### 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></td>
<td>bulk-load API[1]</td>
<td>Sql*loader time range</td>
</tr>
<tr>
<td>LUBM50 6.9 million</td>
<td>8 min</td>
<td>1min 4.3min</td>
</tr>
<tr>
<td>LUBM1000 138 million</td>
<td>3hr 25min</td>
<td>19min 1h 26m</td>
</tr>
<tr>
<td>LUBM8000 1,106 million</td>
<td>30hr 43min</td>
<td>2h 35m 11h 11m 32m</td>
</tr>
<tr>
<td>UniProt (old) 207 million</td>
<td>4hr 40min</td>
<td>30m 1h 55m</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.
## Query Performance

<table>
<thead>
<tr>
<th>Query</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># 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>Complete?</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Time (sec)</td>
<td>0.05</td>
<td>0.75</td>
<td>0.20</td>
<td>0.5</td>
<td>0.22</td>
<td>1.86</td>
<td>1.71</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Query</th>
<th>Q8</th>
<th>Q9</th>
<th>Q10</th>
<th>Q11</th>
<th>Q12</th>
<th>Q13</th>
<th>Q14</th>
</tr>
</thead>
<tbody>
<tr>
<td># answers</td>
<td>7790</td>
<td>13639</td>
<td>4</td>
<td>224</td>
<td>15</td>
<td>228</td>
<td>393730</td>
</tr>
<tr>
<td>Complete?</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Time (sec)</td>
<td>1.07</td>
<td>1.65</td>
<td>0.01</td>
<td>0.02</td>
<td>0.03</td>
<td>0.01</td>
<td>1.47</td>
</tr>
</tbody>
</table>

- **Setup:** Intel Q6600 quad-core, 3 7200RPM SATA disks, 8GB DDR2 PC6400 RAM, No RAID.
  64-bit Linux 2.6.18. Average of 3 warm runs
Query Performance on Server
Going Parallel

LUBM1000 Query Performance

- Setup: Server class machine with 16 cores, NAND based flash storage, 32GB RAM, Linux 64 bit, Average of 3 warm runs
• OWLPrime (11.1.0.7) inference performance scales really well with hardware. It is *not* a parallel inference engine though.
# Inference Performance

| Parallel Inference               | • Time to finish inference: 12 hrs.  
|                                  | • **3.3x faster compared to serial inference in release 11.1** |
| (LUBM8000)                      |                                                                                  |
| 1.06 billion triples            |                                                                                  |
| + 860M inferred                 |                                                                                  |
| **Parallel Inference**          | • Time to finish inference: 40 hrs.  
| (LUBM25000)                     | • **30% faster than nearest competitor**  
| 3.3 billion triples             | • **1/5 cost of other hardware configurations**  
| + 2.7 billion inferred          |                                                                                  |
| **Incremental Inference**       | • Time to update inference: less than 30 seconds after adding 100 triples.  
| (LUBM8000)                      | • **At least 15x to 50x faster than a complete inference done with release 11.1**  
| 1.06 billion triples            |                                                                                  |
| + 860M inferred                 |                                                                                  |
| **Large scale owl:sameAs Inference** | • 60% less disk space required  
| (UniProt 1 Million sample)      | • **10x faster inference compared to release 11.1**  
|                                                                                  |

- Setup: Intel Q6600 quad-core, 3 7200RPM SATA disks, 8GB DDR2 PC6400 RAM, No RAID.  
  64-bit Linux 2.6.18. **Assembly cost: less than USD 1,000**
For More Information

http://search.oracle.com

semantic technologies
ORACLE IS THE INFORMATION COMPANY