Oracle Database Semantic Technologies: Understanding How to Install, Load, Query and Inference

Zhe Wu, Ph.D., Oracle
Matthew Perry, Ph.D., Oracle
Vladimir Kolovski, Ph.D., Oracle

June 2010
Outline

• Introduction to Semantic Web
• Overview of Oracle Database Semantic Technologies
  – Quick introduction to RDF, SPARQL, RDFS, and OWL
• Best practices in installation and configuration
  – A balanced hardware system is a key factor for performance
• Best practices in loading (live demo)
• Best practices in querying (live demo)
• Best practices in inferencing (live demo)
• Performance and scalability of Oracle Database 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 DECISION. 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 Technology Stack

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

http://www.w3.org/2007/03/layerCake.svg
Business Needs

• 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

• Enable data reuse by associating more meaning (context) with the data

• Allow schemas to continuously and dynamically evolve

• Support queries that are not defined in advance
Canonical Use Case: Text Mining

National Intelligence

Web Resources

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

Content Mgmt. Systems

News, Email, RSS

Text Mining

Categorized
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
Canonical Use Case: Data Integration

Health Informatics

Enterprise Information Consumers (EICs)

Access

Run-Time Metadata

Integration Server (Semantic Knowledge base)

Access

Design-Time Metadata

Deploy

Model

Virtual

Relate

Model

Physical

Patient Care

Workforce Management

Business Intelligence

Clinical Analytics

LIS

CIS

HTB

HIS

ORACLE
Semantic Application Workflow

Transform & Edit Tools

- Entity Extraction & Transform
  - OpenCalais
  - Linguamatics
  - GATE
  - D2RQ
- Ontology Eng.
  - TopQuadrant
  - Mondeca
  - Ontoprise
  - Protege
- Categorization
  - Cyc
- Custom Scripting

Load, Query & Inference

- RDF/OWL Data Management
- SQL & SPARQL
  - Sesame Adapter
  - Jena Adapter
- Native Inferencing
- Semantic Rules
- Scalability & Security
- Semantic Indexing

Applications & Analysis Tools

- BI, Analytics
  - Teranode
  - Metatomix
  - MedTrust
- Graph Visualization
  - Cytoscape
- Social Network Analysis
- Metadata Registry
- Faceted Search

Partner Tools

Transaction Systems

Unstructured Content

RSS, email

Other Data Formats

Data Sources

Partner Tools
### Oracle’s Partners for Semantic Technologies

#### Integrated Tools and Solution Providers:

<table>
<thead>
<tr>
<th><strong>Ontology Engineering</strong></th>
<th><strong>Reasoners</strong></th>
<th><strong>Applications</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><a href="#">TopQuadrant</a></td>
<td><a href="#">clarkparsia, llc</a></td>
<td><a href="#">TERANODE</a></td>
</tr>
<tr>
<td><a href="#">ontoprise</a></td>
<td><a href="#">ontoprise</a></td>
<td></td>
</tr>
<tr>
<td><a href="#">protégé</a></td>
<td><a href="#">protégé</a></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Query Tool Interfaces</strong></th>
<th><strong>Standards</strong></th>
<th><strong>SI / Consulting</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><a href="#">Jena</a></td>
<td><a href="#">W3C</a></td>
<td><a href="#">NORTHROP GRUMMAN</a></td>
</tr>
<tr>
<td><a href="#">openRDF.org</a></td>
<td></td>
<td><a href="#">Raytheon</a></td>
</tr>
<tr>
<td><a href="#">Sesame</a></td>
<td></td>
<td><a href="#">orbis Technologies, Inc.</a></td>
</tr>
<tr>
<td><a href="#">Joseki</a></td>
<td></td>
<td><a href="#">BOEING</a></td>
</tr>
<tr>
<td><a href="#">cYcorp</a></td>
<td></td>
<td><a href="#">McDonald Bradley</a></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>NLP Entity Extractors</strong></th>
<th><strong>SI / Consulting</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><a href="#">CALAIS</a></td>
<td><a href="#">Cognia</a></td>
</tr>
<tr>
<td><a href="#">Linguamatics</a></td>
<td><a href="#">accenture</a></td>
</tr>
<tr>
<td><a href="#">GATE</a></td>
<td><a href="#">NBII</a></td>
</tr>
</tbody>
</table>

**General Architecture for Text Engineering**
Some Oracle Database Semantics Customers

- Life Sciences
  - Lilly
  - Pfizer
  - Swiss Institute of Bioinformatics

- Defense/Intelligence
  - Hutchinson 3G
    - Austria

- Clinical Medicine & Research
  - The University of Texas Health Science Center at Houston
  - Cleveland Clinic

- Education
  - Thomson Reuters

- Telecomm & Networking
  - CISCO

- Publishing
  - Thomson Reuters
Oracle Database Provides

**Reasoning and Discovery**
- RDFS / OWL inferencing
- User-defined rules for inferencing
- Plug-in architecture for inference engines such as PelletDB
- Inferencing proofs and explanations
- SPARQL & mixed SQL DB queries

**Data Integration**
- Distributed SPARQL queries through Service in Jena
- Ontologically-assisted SQL queries
- Integration with 3rd party NLP entity extraction engines: e.g., OpenCalais
- Semantic Indexing for documents
Oracle Database Provides

**Scalability**
- Efficient RDBMS storage and loading of RDF data
- Support RAC, Exadata platform, partitioning, compression, versioning
- Incremental & parallel inferencing
- Supports concurrent users, distributed applications

**Security**
- Graph level security
- Virtual Private Database declarative constraints based on RDF data char. & app. / user context
- Oracle Label Security restricts RDF data access to users having compatible access labels
Quick Introduction to RDF
Resource Description Framework (RDF)

- A data model for web resources and their relationships
- The model itself is a graph
- The graph can be serialized into multiple formats:
  - RDF/XML, N3, N-TRIPLE, ...
- Construction unit: assertion or triple (or fact)

<http://foobar> <:produces> <:mp3>

<table>
<thead>
<tr>
<th>Subject</th>
<th>Predicate</th>
<th>Object</th>
</tr>
</thead>
</table>

http://www.foobar.com

http://www.oracle.com
Quick Introduction to SPARQL
SPARQL Query Language for RDF

Querying RDF data

- Find pairs of siblings (same parents)
- SELECT ?x ?y
  FROM <rdf_graph> WHERE {
    ?x hasFather ?f . ?x hasMother ?m .
    ?y hasFather ?f . ?y hasMother ?m .
    FILTER( ?x != ?y )
  }

Result (bindings)

<table>
<thead>
<tr>
<th>?x</th>
<th>?y</th>
</tr>
</thead>
<tbody>
<tr>
<td>Suzie</td>
<td>Matt</td>
</tr>
<tr>
<td>Matt</td>
<td>Suzie</td>
</tr>
<tr>
<td>Cathy</td>
<td>Jack</td>
</tr>
<tr>
<td>Jack</td>
<td>Cathy</td>
</tr>
<tr>
<td>Tom</td>
<td>Cindy</td>
</tr>
<tr>
<td>Cindy</td>
<td>Tom</td>
</tr>
</tbody>
</table>
Quick Introduction to RDFS and OWL
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

Derives implicit relationships using inference
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)

Derives implicit relationships using inference
OWL

• Class related constructs
  • owl:equivlentClass
  • 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:someValuesFrom

: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: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: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

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

:Item1 rdf:type :HighPriorityItem .
⇔ :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, …
Best practices in installation and configuration
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 **matches** 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 parameters
  - SGA, PGA, filesystemio_options, db_cache_size, …

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

- For Java clients using JDBC (Jena Adapter)
  - 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!

Examples of Unbalanced Hardware Setup (1)

- I/O bound system (a real world system)
  - Hardware
    - Dell quad core CPU
    - 16GB RAM
    - Dual IDE and only 1 disk is used for the database
  - Software
    - Linux 64 bit
    - Oracle 11g Release 2
    - No ASM configuration
Examples of Unbalanced Hardware Setup (2)

• CPU bound system (a real world system)
  – Hardware
    • 8 Intel® Xeon® CPU 2.83GHz
    • 32GB RAM
    • Sun Storage F5100 Flash Array
      – Over 1 million I/O operations per second
  – Software
    • Oracle 11g Release 2
    • ASM configuration
    • Linux 64 bit
Examples of **Unbalanced** Hardware Setup (3)

- **I/O bound system (a real world system)**
  - **Hardware**
    - 4 Intel(R) Xeon(TM) CPU 3.20GHz
    - 16GB RAM
    - **RAID5**
    - Multiple 10000RPM SCSI III disks
  - **Software**
    - Linux 2.6.18-128.0.0.0.2.el5 x86_64
    - Oracle 11g Release 2
    - No ASM
Installation and Configuration of Oracle Database
Semantic Technologies
Installation and Configuration (1)

• Load the PL/SQL packages and jar file
  – cd $ORACLE_HOME/md/admin
  – As sysdba
  – SQL> @catsem

• Create a tablespace for semantic network
  create bigfile tablespace semts
  datafile '?/dbs/semts01.dat' size 512M reuse
  autoextend on next 512M maxsize unlimited
  extent management local
  segment space management auto;
Installation and Configuration (2)

• Create a temporary tablespace
  create bigfile temporary tablespace semtmpts  
   tempfile '/dbs/semtmpts.dat'
   size 512M reuse
   autoextend on next 512M maxsize unlimited
   EXTENT MANAGEMENT LOCAL
   ;
   ALTER DATABASE DEFAULT TEMPPORARY TABLESPACE semtmpts;

• Create an undo tablespace
  CREATE bigfile UNDO TABLESPACE semundots
   DATAFILE '/dbs/semundots.dat' SIZE 512M REUSE
   AUTOEXTEND ON next 512M maxsize unlimited
   EXTENT MANAGEMENT LOCAL
   ;
   ALTER SYSTEM SET UNDO_TABLESPACE=semundots;
Installation and Configuration (3)

- Create a semantic network
  - As sysdba
    - SQL> exec sem_apis.create_sem_network('semts');

- Verification
  - As scott (or other)
    - SQL> create table test_tpl(triple sdo_rdf_triple_s) compress;
    - SQL> exec sem_apis.create_sem_model('test','test_tpl','triple');

Live demo of creating a semantic model
Core Entities Relevant to Semantic Network

Semantic network: models, rule bases, entailments, …

- **SDO_RDF_TRIPLE_S**: A new object type for RDF.
- **Application Table**: Contains col of object type sdo_rdf_triple_s to allow loading and accessing RDF triples, and storing ancillary values.
- **Model**: A model holds an RDF graph and is associated with an sdo_rdf_triple_s column in an application table.
- **Rulebase**: A rulebase contains a set of rules used for inferencing.
- **Entailments**: An entailment stores triples derived via inferencing.
### Mapping Core Entities to DB objects

<table>
<thead>
<tr>
<th>Sem. Store entity type</th>
<th>Database object</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model $m$</td>
<td>View mdsys.RDFM_m</td>
</tr>
<tr>
<td>Rulebase $rb$</td>
<td>View mdsys.RDFR_rb</td>
</tr>
<tr>
<td>Rules Index (entailment) $x$</td>
<td>View mdsys.RDFI_x</td>
</tr>
<tr>
<td>Virtual Model $vm$</td>
<td>View mdsys.SEMV_vm (\text{(duplicate)})</td>
</tr>
<tr>
<td></td>
<td>View mdsys.SEMU_vm (\text{(unique)})</td>
</tr>
</tbody>
</table>

- View access control capabilities in database is leveraged to provide access control for the core entities.
- *Instead-of triggers* are used to allow incremental DML on models and rulebases.
Best Practices in Loading
Loading Semantic Data: APIs

- **Incremental DMLs (small number of changes)**
  - SQL: Insert
  - SQL: Delete
  - Java API (Jena): GraphOracleSem.add, delete
  - Java API (Sesame): OracleSailConnection.addStatement, removeStatements

- **Batch loader**
  - BatchImport
  - Java API: OracleBulkUpdateHandler.addInBatch(…)

- **Bulk loader (large number of changes)**
  - PL/SQL: sem_apis.bulk_load_from_staging_table(…)
  - Java API (Jena): OracleBulkUpdateHandler.addInBulk(…)
  - Java API (Sesame): OracleBulkUpdateHandler.addInBulk, prepareBulk…
Load Data into Staging Table using SQL*Loader

- Create a staging table
  ```sql
  CREATE TABLE STAGE_TABLE (
    RDF$STC_sub varchar2(4000) not null,
    RDF$STC_pred varchar2(4000) not null,
    RDF$STC_obj varchar2(4000) not null
  )
  compress pctfree 0 nologging tablespace <TS>;
  ```

- Unzip input data file on the fly
  - mkfifo /tmp/input1
  - gunzip -c data_part1.nt.gz > /tmp/input1 &
  - Repeat for part2, part3, ...

- Use multiple SQL*Loader processes
  ```bash
  sqlldr userid=scott/tiger control=simple.ctl data=/tmp/input1 parallel=true direct=true skip=0 load=1990000000 discardmax=190000000 log=lb1.log bad=lb1.bad discard=lb1.rej errors=100000000 &
  ```
  Same thing to data_part2, data_part3, ...
  Same thing to input2, input3, ...
  ...
Load Data into Staging Table using prepareBulk

- When you have many RDF/XML, N3, TriX or TriG files

```java
OracleSailConnection osc = oracleSailStore.getConnection();

store.disableAllAppTabIndexes();
for (int idx = 0; idx < szAllFiles.length; idx++) {
    ...
    osc.getBulkUpdateHandler().prepareBulk(
        fis,         "http://abc",            // baseURI
        RDFFormat.NTRIPLES,                   // dataFormat
        "SEMTS",                              // tablespaceName
        null,                                 // flags
        null,                                 // register a
        null,                                 // StatusListener
        "STAGE_TABLE",                        // table name
        (Resource[]) null                     // Resource... for contexts
    );
    osc.commit(); fis.close();
}
```

Can start multiple threads and load files in parallel
Complete Data Loading

- Create a semantic model and run bulk load from staging table API

```sql
create table myrdf_tpl (triple sdo_rdf_triple_s) compress
  nologging tablespace semts;  -- remove nologging if
  -- needed

exec
  sem_apis.create_sem_model('myrdf','myrdfTpl','triple');

grant select on stage_table to mdsys;
grant insert on myrdf_tpl to mdsys;

exec sem_apis.bulk_load_from_staging_table(myrdf,'scott',
  'stage_table', flags=>'PARALLEL_CREATE_INDEX
  PARALLEL=4');
```
After Data Is Loaded

• Check number of triples in the model and application table
  - `select count(1) from mdsys.rdfm_<ModelName>;
  - `select count(1) from <AppTable>;

• Analyze the semantic model if there is enough change to the model
  - `exec sem_apis.analyze_model(`'<ModelName>'`);

• Analyze the semantic network if there is enough change to the whole network
  - `exec sem_perf.gather_stats(true, 4);` -- just on value$
    -- table
  - `exec sem_perf.gather_stats(false, 4);` -- whole network

• Start inference and query

  Live demo of data loading
More Data Loading Choices (1)

• Use External Table to load data into Staging Table

```sql
CREATE TABLE stable_ext(
    RDF$STC_sub varchar2(4000),
    RDF$STC_pred varchar2(4000),
    RDF$STC_obj varchar2(4000))
ORGANIZATION EXTERNAL (
    TYPE ORACLE_LOADER
    DEFAULT DIRECTORY tmp_dir
    ACCESS PARAMETERS(
        RECORDS DELIMITED by NEWLINE
        PREPROCESSOR bin_dir:'uncompress.sh'
        FIELDS TERMINATED BY ' ' )
LOCATION ('data1.nt.gz', 'data2.nt.gz', ..., 'data_4.nt.gz')
)
REJECT LIMIT UNLIMITED
;
```

Multiple files is critical to performance
More Data Loading Choices (2)

• Load directly using Jena Adapter

    Oracle oracle = new Oracle(szJdbcURL, szUser, szPasswd);
    Model model = ModelOracleSem.createOracleSemModel(
        oracle, szModelName);
    InputStream in = FileManager.get().open("./univ.owl");
    model.read(in, null);

• More loading examples using Jena Adapter

    – Examples 7-2, 7-3, and 7-12 (SPARUL) [1]

[1]: Oracle® Database Semantic Technologies Developer’s Guide
http://download.oracle.com/docs/cd/E11882_01/appdev.112/e11828/toc.htm
More Data Loading Choices (3)

• Load directly using Sesame Adapter

```java
OraclePool op = new OraclePool(
    OraclePool.getOracleDataSource(jdbcUrl, user, password));
OracleSailStore store = new OracleSailStore(op, model);
SailRepository sr = new SailRepository(store);
RepositoryConnection repConn = sr.getConnection();
repConn.setAutoCommit(false);
repConn.add(new File(trigFile), "http://my.com/", RDFFormat.TRIG);
repConn.commit();
```

• More loading examples using Sesame Adapter

  – Examples 8–5, 8–7, 8–8, 8–9, and 8–10 [1]

[1]: Oracle® Database Semantic Technologies Developer's Guide
http://download.oracle.com/docs/cd/E11882_01/appdev.112/e11828/toc.htm
Best Practices in Querying Semantic Data
SPARQL Query Architecture

HTTP

Java

SQL

Standard SPARQL Endpoint

Jena API
Jena Adapter

Sesame API
Sesame Adapter

SEM_MATCH
Outline

• Description of SEM_MATCH capabilities

• Live demo of SEM_MATCH

• Performance best practices for SEM_MATCH
SEM_MATCH: Adding SPARQL to SQL

- Extends SQL with SPARQL constructs
  - Graph Patterns
  - OPTIONAL, UNION
  - FILTER – including SPARQL built-ins
- Allows native “SPARQL” queries against RDF graphs stored in Oracle database
- Benefits:
  - Allows powerful SQL constructs:
    - aggregates, subqueries, analytical functions, expressions
    - JOINs with other object-relational data
    - Spatial, Image, etc.
SEM_MATCH: Adding SPARQL to SQL

**SPARQL**

```sparql
PREFIX foaf: <http://...>
SELECT ?n1 ?n2
FROM <http://g1>
WHERE
  {?p foaf:name ?n1
    OPTIONAL {?p foaf:knows ?f .
    ?f foaf:name ?n2 }
  FILTER (REGEX(?n1, "^A")) }
```

**SQL**

```sql
SELECT n1, n2
FROM TABLE(SEM_MATCH(
'{'
?p foaf:name ?n1
  OPTIONAL {?p foaf:knows ?f .
  ?f foaf:name ?n2 }
  FILTER (REGEX(?n1, "^A")) ','
SEM_MODELS('g1'),...,
SEM_ALIASES(
  SEM_ALIAS('foaf', 'http://...')) , ...))
```
SELECT n1, n2
FROM TABLE(
  n1 | n2
  ---|---
  Alice | John
  Alice | Jill
  Alice | Bill
  Alex  | Tom
  Alex  | Jerry
)
SELECT n1, n2
FROM
( SELECT v1.value AS n1, v2.value AS n2
  FROM VALUES v1, VALUES v2
  TRIPLES t1, TRIPLES t2, ...
  WHERE t1.obj_id = v1.value_id
  AND t1.pred_id = 1234
  AND ...
)

Get 1 declarative SQL query
- Query optimizer sees 1 query
- Get all the performance of Oracle SQL Engine
  - compression, indexes, parallelism, etc.
SEM_MATCH Table Function Arguments

SEM_MATCH(
    query,
    models,
    rulebases,
    aliases,
    filter,
    index_status,
    options
);

'{ ?a foaf:name ?b }'

Container(s) for asserted triples

Built-in (e.g. OWL2RL) and user-defined rulebases

Entailed triples

Fine-grained access control

'{INCOMPLETE}'

('foaf', 'http://www...')

'(to_date(var,'YYYY-MM-DD') ... )'

'ALLOW_DUP=T'
SEM_MATCH Demo
GovTrack RDF Data

RDF/OWL data about activities of US Congress
- Political Party Membership
- Voting Records
- Bill Sponsorship
- Committee Membership
- Offices and Terms

GovTrack in Oracle

GovAssertVM
- Asserted data only
  (2.8M triples)

GovAllVM
- Asserted + Inferred
  (3.1M triples)

Virtual Models
- Semantic Models
- Rulebases
- Entailments

GOV_TBOX

GOV_PEOPLE

GOV_BILLS_110

GOV_BILLS_111

GOV_VOTES_07

GOV_VOTES_08

GOV_VOTES_09

GOV_TRACK_OWL

OWL2RL

Inference

http://www.govtrack.us/developers/rdf.xpd
Virtual Models

• A virtual model is a logical RDF graph that can be used in a SEM_MATCH query.
  - Result of UNION or UNION ALL of one or more models and optionally the corresponding entailment

• create_virtual_model (vm_name, models, rulebases)
• drop_virtual_model (vm_name)
• SEM_MATCH query accepts a single virtual model
  - No other models or rulebases need to be specified
• DMLs on virtual models are not supported
Virtual Model Example

Creation

```
begin
    sem_apis.create_virtual_model('gov_assert_vm',
        sem_models('gov_tbox', 'gov_people', 'gov_votes_07',
                   'gov_votes_08', 'gov_votes_09', 'gov_bills_110',
                   'gov_bills_111'));

    sem_apis.create_virtual_model('gov_all_vm',
        sem_models('gov_tbox', 'gov_people', 'gov_votes_07',
                   'gov_votes_08', 'gov_votes_09', 'gov_bills_110',
                   'gov_bills_111'),
        sem_rulebases('OWL2RL'));
end;
```

Access Control

```
grant select on mdsys.semv_gov_assert_vm to scott;
grant select on mdsys.semv_gov_all_vm to scott;
```
Live Example 1: Access Control

- **govtrack** is owner of all govtrack data
- **scott** cannot access any govtrack data by default
- **govtrack** can grant select privileges on relevant database objects (e.g. `mdsys.semv_gov_all_vm`) to **scott**
- **scott** can then query govtrack data (e.g. `gov_all_vm` Virtual Model)
Live Example 2: Basic Query

Find information about all Kennedys

```sql
select fn, bday, g, t, hp, r
from table(sem_match(' 
  { ?s vcard:N ?n .
    ?n vcard:Family "Kennedy" .
    ?s foaf:name ?fn .
    ?s foaf:gender ?g .
    ?s foaf:homepage ?hp .
    ?s foaf:religion ?r
  }' ,
  ,sem_models('gov_all_vm'), null
  ,sem_aliases(...) ,null, null
  ,' ALLOW_DUP=T ' ));
```
Live Example 3: OPTIONAL Query

Find information about all Kennedys, with title, homepage and religion optional

```sql
select fn, bday, g, t, hp, r
from table(sem_match(
  '{ ?s vcard:N ?n .
     ?n vcard:Family "Kennedy" .
     ?s foaf:name ?fn .
     ?s foaf:gender ?g .
     OPTIONAL { ?s foaf:title ?t .
                   ?s foaf:homepage ?hp .
                   ?s foaf:religion ?r }
  }'
,sem_models('gov_all_vm'), null
,sem_aliases(...) 
,null, null
,' ALLOW_DUP=T ' ));
```
Live Example 4: Simple FILTER

Find all people with a last name that starts with “A”

```sql
select fname, lname
from table(sem_match(
'{ ?s rdf:type foaf:Person .
 ?vcard vcard:Given ?fname .
 ?vcard vcard:Family ?lname
 FILTER (STR(?lname) < "B") }
,sem_models('gov_all_vm'), null
,sem_aliases(...)
,null, null
,' ALLOW_DUP=T ')
));
```
Find all Lincolns without a homepage

```sql
select fn, bday, hp
from table(sem_match(
'{ ?s vcard:N ?n .
   ?n vcard:Family "Lincoln" .
   ?s foaf:name ?fn .
   FILTER (!BOUND(?hp))
   OPTIONAL {
       ?s foaf:homepage ?hp
   }
}

),sem_models('gov_all_vm'), null ,sem_aliases(...)
,null, null
,' ALLOW_DUP=T '));
```
Live Example 6: UNION

Find all Legislative Documents introduced in February 2007 that were sponsored or cosponsored by Barack Obama

```sql
select title, dt
from table(sem_match(
'\{ ?s foaf:name "Barack Obama" .
  \{ ?b bill:sponsor ?s } 
  \n  UNION
  \{ ?b bill:cosponsor ?s } 
  ?b bill:introduced ?dt
  FILTER("2007-02-01"^^xsd:date <= ?dt &&
          ?dt < "2007-03-01"^^xsd:date ) \}'
,sem_models('gov_all_vm'), null
,sem_aliases(...)
,null, null
,' ALLOW_DUP=T ')
));
```
Live Example 7: Inference

GovTrack Bill Types
Live Example 7: Inference

Find all Legislative Documents (and their types) sponsored by Barack Obama

```
select title, dt, btype
from table(sem_match(
  '{ ?s foaf:name "Barack Obama" .
    ?b bill:sponsor ?s .
    ?b bill:introduced ?dt
    FILTER("2007-03-28"^^xsd:date <= ?dt &&
             ?dt < "2007-04-01"^^xsd:date ) }
),sem_models('gov_assert_vm'), null
,sem_aliases(...)
,null, null
,' ALLOW_DUP=T ' ));
```
Live Example 8: SQL Constructs \textit{(temporal interval computations)}

Find the youngest person to take office since 2000

```
select * from (
select fn, bday, tsfrom,
    (to_date(tsfrom,'YYYY-MM-DD') –
    to_date(bday,'YYYY-MM-DD')) YEAR(3) TO MONTH
from table(sem_match('{
    ?role time:from ?tfrom . ?tfrom time:at ?tsfrom
    FILTER (?tsfrom >= "2000-01-01"^^xsd:date)
}
,sem_models('gov_all_vm'), null, sem_aliases(...) ,null, null,' ALLOW_DUP=T ' ))
order by (to_date(tsfrom,'YYYY-MM-DD') –
    to_date(bday,'YYYY-MM-DD')) asc
where rownum <= 1;
```
SEM_MATCH Best Practices for Query Performance
## Basic RDF Data Storage Tables

### RDF_VALUE$ Table

<table>
<thead>
<tr>
<th>vname_prefix</th>
<th>vname_suffix</th>
<th>Value_id</th>
<th>canon_id</th>
<th>literal_type</th>
<th>language_type</th>
</tr>
</thead>
<tbody>
<tr>
<td><a href="http://xyz.com/">http://xyz.com/</a></td>
<td>John</td>
<td>100</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><a href="http://xyz.com/">http://xyz.com/</a></td>
<td>Mary</td>
<td>200</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><a href="http://xyz.com/">http://xyz.com/</a></td>
<td>managerOf</td>
<td>300</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### RDF_LINK$ Table

<table>
<thead>
<tr>
<th>start_node_id</th>
<th>p_value_id</th>
<th>canon_end_node_id</th>
<th>model_id</th>
<th>end_node_id</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>300</td>
<td>200</td>
<td>1</td>
<td>200</td>
</tr>
</tbody>
</table>

- **Unique key:** start_node_id, p_value_id, model_id
- **pkey:** Unique key
- **Partition:** M1, M2
Basic Performance Tips
Tip 1: Always Gather Statistics

- **SEM_APIS Procedures (local)**
  - ANALYZE_MODEL
  - ANALYZE_ENTAILMENT

- **SEM_PERF Procedures (global)**
  - GATHER_STATS

```sql
SQL> exec sem_apis.analyze_model('GOV_PEOPLE',degree=>4);
PL/SQL procedure successfully completed.

SQL> exec sem_perf.gather_stats(degree=>4);
PL/SQL procedure successfully completed.
```
Tip 2: MultiColumn Statistics

- Usually we access the RDF_LINK$ table with more than one column
  - e.g. give me all the canon_ids for a given (p_value_id, start_node_id) combination
- If we gather statistics for columns as a group (*column group*) then the optimizer will have much better selectivity estimations for the group (e.g. (p_value_id, start_node_id)).

```sql
SQL> exec sem_apis.analyze_model('gov_people',degree=>4,
     METHOD_OPT =>'FOR COLUMNS (P_VALUE_ID,CANON_END_NODE_ID)
                 SIZE AUTO');
PL/SQL procedure successfully completed.

SQL> exec sem_apis.analyze_model('gov_people',degree=>4,
     METHOD_OPT =>'FOR COLUMNS (P_VALUE_ID,START_NODE_ID)
                 SIZE AUTO');
PL/SQL procedure successfully completed.
```

Recommended Groups: PC and PS
Tip 3: Use Virtual Models

• When querying over multiple models and/or entailments use a Virtual Model
  – Allows some query simplifications that may increase performance
  
  – Fast switching of new/updated graphs
    • Just redefine virtual model to point to new graph … queries remain unchanged

  – Simplify access control
    • Instead of giving permission to each model and entailment … just create 1 VM and give access to it
Tip 4: Reducing RDF_VALUE$ JOINs

- Only select a query variable if you truly need to
  - If a query variable is not selected then we don’t need to join with RDF_VALUE$
- Use the VAR$RDFVID column
  - SEM_MATCH returns multiple columns for each query variable
  - One of those columns is an internal identifier (VAR$RDFVID)
  - We can eliminate a VALUE$ join for variable v if only v$RDFVID is selected
Example: Using the VID Column to Reduce JOINs

Get a count for each Class

```sql
select v.value_name group_name, t.member_count from ( 
    SELECT y$rdfvid, count(*) member_count 
    FROM table(sem_match( 
        '{?x rdf:type ?y}', 
        SEM_Models('rdf_model_uniprot'),null,null,null 
    )) 
    GROUP BY y$rdfvid 
) t, mdsys.rdf_value$ v 
where t.y$rdfvid = v.value_id;
```
Tip 5: Query Options

• OPTIONS argument in SEM_MATCH
  – ‘ALLOW_DUP=T’
  • By default, Oracle follows set semantics for storage and query
    – To maintain set semantics for a multi-model query, we have to do a SQL UNION on triple data
    – UNION significantly impacts performance
  • This hint bypasses UNION to gain performance at the cost of possible duplicate rows
Tip 5: Query Options

- OPTIONS argument in SEM_MATCH
  - ‘ FAST_DATE_FILTER=T ’
  - Consider FILTER (?x < “2010-06-21T12:00:00Z”^^xsd:dateTime)
  - Comparison of values with and without timezones is complex
    - “2000-06-21T12:00:00Z” < “2000-06-21T13:00:00” – ??
    - “2000-06-21T12:00:00Z” < “2000-07-21T13:00:00” – true
  - Extra logic adds some overhead
  - This hint bypasses the extra logic
  - Note: answers are always correct if all values either have or do not have timezones
Tip 6: FILTER Optimizations

• Try to use `sameTerm()`
  - `sameTerm(?a, ?b)` is more efficient than `(?a = ?b)`
    • `sameTerm` uses id-based comparison
    • `=` involves more complex lexical-to-value mapping

• Try to avoid negation
  - `(?a < “abc”^^xsd:string)` is better than `!(?a >= “abc”^^xsd:string)`
  - Negation logic is complex for typed literals
Creating Indexes to Speed up Query Execution
Tip 7: Semantic Network Indexes

• Allows custom B-Tree indexing for RDF models and entailments

• `add_sem_index` (index_code)
  – Adds a new nonunique index for every RDF model
  – Index_code is some combination of one or more of the column abbreviations: M,S,P,C. (Example: ‘CPS’)

• `alter_sem_index_on_model`(model_name,index_code,command)

• `alter_sem_index_on_entailment`(entailment_name,index_code,command)
  – Rebuilds the index on the specified RDF model or entailment

• `drop_sem_index` (index_code)
### Index Code Columns: S  P  C  M

<table>
<thead>
<tr>
<th>start_node_id</th>
<th>p_value_id</th>
<th>canon_end_node_id</th>
<th>model_id</th>
<th>end_node_id</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>300</td>
<td>200</td>
<td>1</td>
<td>200</td>
</tr>
</tbody>
</table>

---

**RDF_LINK$ Table**

**Partition**

- M1
- M2
Tip 7: Semantic Network Indexes (cont.)

Multi-column indexes speed up query execution significantly

Example

```
{  ?a foaf:knows  ?b . #t0
   ?b foaf:name  ?n . #t1
}
```

Consider join of t0 and t1

- We have bindings for ?a and ?b and we want all bindings for ?n
- Internally, this becomes: given P and S, get all C
- A ‘PSCM’ index can do this very efficiently with a single index scan
Tip 7: Semantic Network Indexes (cont.)

• Recommendations:
  – Always include $M$ column in last position
    • Especially important for Virtual Model queries
  – For a basic setup, use these 2 indexes
    • $PCS\!\!M$ – always there … enforces a uniqueness constraint on RDF\_LINK$
    • $PSC\!\!M$
    • These indexes perform well in the common case (constant URI in predicate position)
Semantic Network Indexes Example

Find all Information about Ron Paul

```sql
select p, o
from table(sem_match(
'{ ?s foaf:name "Ronald Paul" .
  ?s ?p ?o }
,sem_models('gov_all_vm'), null
,sem_aliases(sem_alias(...)
,null, null
,' ALLOW_DUP=T ')
));
```

Join Operation: given S, get all PC

We need an **SPCM** index for the most efficient evaluation
## Execution Plan Without SPCM Index

<table>
<thead>
<tr>
<th>Id</th>
<th>Operation</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>SELECT STATEMENT</td>
<td></td>
</tr>
<tr>
<td>* 1</td>
<td>FILTER</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>NESTED LOOPS</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>NESTED LOOPS</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>NESTED LOOPS</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>VIEW</td>
<td></td>
</tr>
<tr>
<td>* 6</td>
<td>HASH JOIN</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>PARTITION LIST INLIST</td>
<td></td>
</tr>
<tr>
<td>* 8</td>
<td>INDEX RANGE SCAN</td>
<td>RDF_LNK_PVIDCENSNMID_IDX</td>
</tr>
<tr>
<td>9</td>
<td>PARTITION LIST INLIST</td>
<td></td>
</tr>
<tr>
<td>* 10</td>
<td>TABLE ACCESS FULL</td>
<td>RDF_LINK$</td>
</tr>
<tr>
<td>11</td>
<td>TABLE ACCESS BY INDEX ROWID</td>
<td>RDF_VALUE$</td>
</tr>
<tr>
<td>* 12</td>
<td>INDEX UNIQUE SCAN</td>
<td>C_PK_VID</td>
</tr>
<tr>
<td>* 13</td>
<td>INDEX UNIQUE SCAN</td>
<td>C_PK_VID</td>
</tr>
<tr>
<td>14</td>
<td>TABLE ACCESS BY INDEX ROWID</td>
<td>RDF_VALUE$</td>
</tr>
<tr>
<td>* 15</td>
<td>TABLE ACCESS FULL</td>
<td>RDF_RI_SHAD_8$</td>
</tr>
</tbody>
</table>
Execution Plan With SPCM Index

<table>
<thead>
<tr>
<th>Id</th>
<th>Operation</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>SELECT STATEMENT</td>
<td></td>
</tr>
<tr>
<td>* 1</td>
<td>FILTER</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>NESTED LOOPS</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>NESTED LOOPS</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>NESTED LOOPS</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>VIEW</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>NESTED LOOPS</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>PARTITION LIST INLIST</td>
<td>RDF_LNK_PVIDCENSNMID_IDX</td>
</tr>
<tr>
<td>* 8</td>
<td>INDEX RANGE SCAN</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>PARTITION LIST INLIST</td>
<td></td>
</tr>
<tr>
<td>* 10</td>
<td>INDEX RANGE SCAN</td>
<td>RDF_LNK_SPCM_IDX</td>
</tr>
<tr>
<td>11</td>
<td>TABLE ACCESS BY INDEX ROWID</td>
<td>RDF_VALUE$</td>
</tr>
<tr>
<td>* 12</td>
<td>INDEX UNIQUE SCAN</td>
<td>C_PK_VID</td>
</tr>
<tr>
<td>* 13</td>
<td>INDEX UNIQUE SCAN</td>
<td>C_PK_VID</td>
</tr>
<tr>
<td>14</td>
<td>TABLE ACCESS BY INDEX ROWID</td>
<td>RDF_VALUE$</td>
</tr>
<tr>
<td>* 15</td>
<td>TABLE ACCESS FULL</td>
<td>RDF_RI_SHAD_8$</td>
</tr>
</tbody>
</table>
Tip 8: Value-based Indexes for FILTERs

```
FILTER (?lname = "Kennedy" || ?lname = "Lincoln")
```

Find all Kennedys and Lincolns

```
select fname, lname 
from table(sem_match(
  ?vcard vcard:Given ?fname .
  ?vcard vcard:Family ?lname
  FILTER (?lname = "Kennedy" || ?lname = "Lincoln") }'
,sem_models('gov_all_vm'), null 
,sem_aliases(...) 
,null, null 
,' ALLOW_DUP=T ' 
));
```

The evaluation of this FILTER uses a function on RDF_VALUE$

We can create a function-based index that will speed up such FILTERs

Applies when we have

FILTER(var <comp> xsd:literal)
Tip 8: Value-based indexes for FILTERs

- SEM_APIS.getV$NumericVal(…) → NUMBER
  - xsd numeric types
- SEM_APIS.getV$StringVal(…) → VARCHAR2
  - xsd:string and plain literals
- SEM_APIS.getV$DateTimeTZVal(…) → TIMESTAMP
  - xsd:dateTime
- SEM_APIS.getV$DateTZVal(…) → TIMESTAMP
  - xsd:date
- SEM_APIS.getV$TimeTZVal(…) → TIMESTAMP
  - xsd:time

Native Oracle Types
Value-based Index Example

Find all Kennedys and Lincolns

```sql
select fname, lname
from table(sem_match('{/ s vcard:N ?vcard .
   ?vcard vcard:Given ?fname .
   ?vcard vcard:Family ?lname
   FILTER (?lname = "Kennedy" ||
   ?lname = "Lincoln") }
,sem_models('gov_all_vm'), null
,sem_aliases(...)
,null, null,' ALLOW_DUP=T '));
```

Index Creation

```sql
create index mdsys.v$stringIdx on mdsys.rdf_value$(
   sem_apis.getV$StringVal(value_type,vname_prefix,
   vname_suffix, literal_type, language_type));
```
### Index-based Execution Plan

<table>
<thead>
<tr>
<th>Id</th>
<th>Operation</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>SELECT STATEMENT</td>
<td></td>
</tr>
<tr>
<td>* 1</td>
<td>FILTER</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>NESTED LOOPS</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>NESTED LOOPS</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>VIEW</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>NESTED LOOPS</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>NESTED LOOPS</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>NESTED LOOPS</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>INLIST ITERATOR</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>TABLE ACCESS BY INDEX ROWID</td>
<td>RDF_VALUE$</td>
</tr>
<tr>
<td>* 10</td>
<td>INDEX RANGE SCAN</td>
<td>V$STRINGIDX</td>
</tr>
<tr>
<td>11</td>
<td>PARTITION LIST INLIST</td>
<td></td>
</tr>
<tr>
<td>* 12</td>
<td>INDEX RANGE SCAN</td>
<td>RDF_LNK_PVIDCENSNMD_IDX</td>
</tr>
<tr>
<td>13</td>
<td>PARTITION LIST INLIST</td>
<td></td>
</tr>
<tr>
<td>* 14</td>
<td>INDEX RANGE SCAN</td>
<td>RDF_LNK_PSCM_IDX</td>
</tr>
<tr>
<td>15</td>
<td>PARTITION LIST INLIST</td>
<td></td>
</tr>
<tr>
<td>* 16</td>
<td>INDEX RANGE SCAN</td>
<td>RDF_LNK_PVIDCENSNMD_IDX</td>
</tr>
<tr>
<td>* 17</td>
<td>INDEX UNIQUE SCAN</td>
<td>C_PK_VID</td>
</tr>
<tr>
<td>18</td>
<td>TABLE ACCESS BY INDEX ROWID</td>
<td>RDF_VALUE$</td>
</tr>
<tr>
<td>* 19</td>
<td>TABLE ACCESS FULL</td>
<td>RDF_RI_SHAD_8$</td>
</tr>
</tbody>
</table>
# Table / Index Size for GovTrack Data

<table>
<thead>
<tr>
<th>Table / Index</th>
<th>Size (MB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>RDF_VALUE$</td>
<td>32</td>
</tr>
<tr>
<td>v$StringIDX</td>
<td>20</td>
</tr>
<tr>
<td>RDF_LINK$</td>
<td>77</td>
</tr>
<tr>
<td>PCSM_IDX</td>
<td>108</td>
</tr>
<tr>
<td>PSCM_IDX</td>
<td>136</td>
</tr>
<tr>
<td>SPCM_IDX</td>
<td>144</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Number of Triples</th>
</tr>
</thead>
<tbody>
<tr>
<td>GOV_ASSERT_VM</td>
<td>2.8 Million</td>
</tr>
<tr>
<td>GOV_ALL_VM</td>
<td>3.1 Million</td>
</tr>
</tbody>
</table>
Tuning Query Execution Plans
Tip 9: Query Tuning with HINTs

- Allows tuning SQL execution plan
- Gives fine-grained control of:
  - Triple pattern evaluation order
  - Type of join used
  - Which indexes to use
SELECT e, p FROM
TABLE(SEM_MATCH('(?e rdf:type ?empCategory)
(?empCategory :HQLoc ?loc)
(?loc :partOf ?p)
'),
SEM_Models('emp', 'gmap'), SEM_Rulebases('OWLPRIME'),
SEM_ALIASES(SEM_ALIAS('', 'http://example.org/')),NULL));

- Aliases for the tables involved in the Join (in translated SQL query)
  - For the 3 triple patterns: t0, t1, t2 ➔ aliases for RDF Model
  - For the 4 distinct variables: ?e, ?empCategory, ?loc, ?p ➔ aliases for RDF_VALUE$
HINT0 Specification

• HINTS in SEM_MATCH query
  – Similar to SQL hints
    • Join order: LEADING(…), ORDERED
    • Join type: USE_HASH(…), USE_NL(…), ...
    • Access path: INDEX(…), ...
  – Aliases are a bit different
    • Variable names: ?x, ?y, ?f, ?m, ...
    • Pattern ordinal based: t0, t1, t2, ...
HINT0: Example

Find everyone with a last name that starts with “A”

```
select fname, lname
from table(sem_match(
  '{ ?s vcard:N ?vcard . #t0
    ?vcard vcard:Given ?fname . #t1
    ?vcard vcard:Family ?lname #t2
    FILTER (?lname < "B") }
  ,sem_models('gov_all_vm'), null
  ,sem_aliases(...)  
  ,null, null
  ,' ALLOW_DUP=T
  HINT0={ leading(?lname t2 t0 t1)
    use_nl(?lname t2 t0 t1)
    index(?lname v$stringidx) } ' 
));
```

Goal: start with VALUE$ index and drive the query from there using nested loop join.
Enterprise Security for Semantic Data
Enterprise Security for Semantic Data

- Oracle RDF provides security through both Oracle Virtual Private Database and Label Security

- Virtual Private Database intercepts and rewrites the user query to restrict the result set using additional predicates.
  - A policy function associated with a table determines the exact security condition for each user-query on the table

- Oracle Label Security - mandatory access control
  - Labels assigned to both users and data
  - Data labels determine the sensitivity of the rows or the rights a person must possess in order to read or write the data.
  - User labels indicate their access rights to the data records.
Virtual Private Database for RDF Data

[VPD] Access control policies on semantic data

Policy ➞ user can access value of projects s/he leads
Match pattern ➞  \{ ?x :hasValue ?v \}
Apply pattern ➞  \{ ?x :hasLead “sys_context(…)” \}

Query : Get the list of projects and their values

SELECT  ?proj  ?val
WHERE   { ?proj   :hasValue  ?val .
            ?proj   :hasLead “sys_context(…)” }
Oracle Label Security for RDF data

- Oracle Label Security enforces security at the “row” level.
  - Attempts to access specific rows are validated using row/data sensitivity *labels*.
  - If user’s read label > row’s label, user can access row

<table>
<thead>
<tr>
<th>ContractID</th>
<th>Organization</th>
<th>ContractValue</th>
<th>RowLabel</th>
</tr>
</thead>
<tbody>
<tr>
<td>ProjectHLS</td>
<td>N. America</td>
<td>1000000</td>
<td>SE:HLS:US</td>
</tr>
</tbody>
</table>

- Two different levels of granularity for RDF OLS
  - Resource-level security
    - Each subject, predicate, object could be associated with a label
    - Upon inserting a triple, the user’s label has to dominate all of the resource labels
  - Triple-level security
    - Thin layer on top of OLS
## Resource vs Triple Level Security Modes

<table>
<thead>
<tr>
<th>Feature</th>
<th>Resource-Level</th>
<th>Triple-Level</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Resource constraints</strong></td>
<td>Yes - provide a minimum bound for triple labels</td>
<td>No</td>
</tr>
<tr>
<td><strong>Insert Model</strong></td>
<td><strong>Strict</strong> - cannot insert a triple unless all resource labels are dominated by user’s label</td>
<td><strong>Open</strong> - anybody can say anything about anything; the triple is always tagged with the user’s default label</td>
</tr>
<tr>
<td><strong>Inference/Bulk Load</strong></td>
<td>Has to be run as security admin/ various options on generating labels for inferred triples</td>
<td>Any user can do it; you can only perform inference on the data you can see; triples inserted with user’s default label</td>
</tr>
<tr>
<td><strong>Changing triple label after insert</strong></td>
<td>Non-trivial</td>
<td>Easy to do</td>
</tr>
</tbody>
</table>
Inference
Oracle’s Focus: Lightweight OWL Inference

• “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 (2006)
  - Out of 1,200+ real-world OWL ontologies:
    - 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

• New OWL 2 Standard profiles:
  - OWL 2 RL, OWL 2 EL, OWL 2 QL – all polynomial time complexity

2 A Survey of the web ontology landscape. ISWC 2006
Core Inference Features

- Inference done using forward chaining
  - Triples inferred and stored ahead of query time
  - Removes on-the-fly reasoning and results in fast query times
- Various native rulebases provided
  - E.g., RDFS, OWL 2 RL, SNOMED (EL+), SKOS
- Validation of inferred data
- User-defined rules
- Proof generation
  - Shows one deduction path
OWL Subsets Supported

- **OWL subsets for different applications**
  - RDFS++
    - RDFS plus owl:sameAs and owl:InverseFunctionalProperty
  - OWLSIF (OWL with IF semantics)
    - Based on Dr. Horst’s pD* vocabulary\(^1\)
  - OWLPrime
    - Includes RDFS++, OWLSIF with additional rules
    - Jointly determined with domain experts, customers and partners
  - OWL 2 RL
    - W3C Standard
    - Adds rules about keys, property chains, unions and intersections to OWLPrime
  - SNOMED

---

\(^1\) 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: \( p \text{ rdfs:domain } x \).
    \[ u \quad p \quad y \quad \Rightarrow \quad u \text{ rdf:type } x \].

- OWL 2 RL has 70+ entailment rules.
  - E.g. rule: \( p \quad \text{rdf:type } \text{owl:FunctionalProperty} \).
    \[ x \quad p \quad y_1 \quad \Rightarrow \quad y_1 \text{ owl:sameAs } y_2 \].
    \[ x \quad \text{owl:disjointWith } y \quad \Rightarrow \quad x \text{ owl:differentFrom } y \quad \Rightarrow \quad a \text{ owl:differentFrom } b \].

- These rules have efficient implementations in RDBMS.
Inference APIs

- **SEM_APIS.CREATE_ENTAILMENT**(
  - `index_name`
  - `sem_models('GraphTBox', 'GraphABox', ...)`,
  - `sem_rulebases('OWL2RL'),`
  - `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`
)

- *Jena Adapter 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
Extending Semantics Supported by 11.2 OWL Inference

- **Option 1: add user-defined rules**
  - Both 10g and 11g RDF/OWL support user-defined rules in this form:

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

- Filter expressions are allowed
  - ?x :hasAge ?age.
Option 2: Separation in TBox and ABox reasoning through PelletDb (using Oracle Jena Adapter)

- TBox (schema related) tends to be small in size
  - Generate a class subsumption tree using a complete DL reasoners like Pellet
- ABox (instance related) can be arbitrarily large
  - Use the native inference engine in Oracle to infer new knowledge based on class subsumption tree from TBox
11g Release 2 Inference Features

- Richer semantics support
  - OWL 2 RL, SKOS, SNOMED (subset of OWL 2 EL)
- Performance enhancements
  - Large scale owl:sameAs handling
    - Compact materialization of owl:sameAs closure
  - Parallel inference
    - Leverage native Oracle parallel query and parallel DML
  - Incremental inference
    - Efficient updates of inferred graph through additions
  - Compact Data Structures
Enabling Advanced Inference Capabilities

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

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

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

- Compact data structures
  ```sql
  EXECUTE Sem_apis.create_entailment ('M_IDX',sem_models('M'),
  sem_rulebases('OWLPRIME'),null,null, 'RAW8=T');
  ```

- OWL2RL/SKOS inference
  ```sql
  EXECUTE Sem_apis.create_entailment('M_IDX',sem_models('M'),
  sem_rulebases(x),null,null...);
  ```
  x in (‘OWL2RL’,‘SKOSCORE’)
Tuning Tips for Best Inference Performance

- Analyze models before running inference
  - execute immediate semApis.analyze_model(...);
- Need a **balanced** hardware setup to use parallel inference
  - E.g., a server with multi-core/multi-cpu processors and ample I/O throughput
  - Use Oracle Automatic Storage Management (ASM) to manage the disks
- Use RAW8=T option for compact data structures
  - Smaller data structures imply less I/O
- Dynamic incremental inference
  - Selectively applies semi-naïve rule evaluation while generating the entailment
  - Off by default, could be turned on with DYN_INC_INF=T option
Inference Demo
Jena Adapter for Oracle Database 11g Release 2
Jena Adapter for Oracle Database 11g Release 2

• Implements Jena Semantic Web Framework APIs
  • Popular Java APIs for semantic web based applications
  • Adapter adds Oracle-specific extensions

• Jena Adapter provides three core features:
  • Java API for Oracle RDF Store
  • SPARQL Endpoint for Oracle with SPARQL 1.1. support
  • Oracle-specific extensions for query execution control and management
Jena Adapter as a Java API for Oracle RDF

- “Proxy” like design
  - Data not cached in memory for scalability
  - SPARQL query converted into SQL and executed inside DB
    - Various optimizations to minimize the number of Oracle queries generated given a SPARQL 1.1. query

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

- Allows integration of Oracle Database 11g RDF/OWL with various tools
  - TopBraid Composer
  - External OWL DL reasoners (e.g., Pellet)

Programming Semantic Applications in Java

- Create a connection 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();
Jena Adapter Feature: SPARQL Endpoint

- SPARQL service endpoint supporting full SPARQL Protocol
  - Integrated with Jena/Joseki 3.4.0 (deployed in WLS 10.3 or Tomcat 6)
  - Uses J2EE data source for DB connection specification
  - SPARQL 1.1. and Update (SPARUL) supported

- Oracle-specific declarative configuration options in Joseki
  - Each URI endpoint is mapped to a Joseki service:

    ```
    <$service>
    rdf:type       joseki:Service ;
    rdfs:label     "SPARQL with Oracle Semantic Data Management" ;
    joseki:serviceRef "GOV_ALL_VM" ;#web.xml must route this name to Joseki
    joseki:dataset <#oracle> ; # dataset part
    joseki:processor joseki:ProcessorSPARQL_FixedDS;
    ```
Example Joseki Dataset configuration:

```xml
<#oracle> rdf:type oracle:Dataset;
    joseki:poolSize 1;  # Number of concurrent connections
    oracle:connection [ a oracle:OracleConnection ; ];

oracle:defaultModel [  
    oracle:useVM "TRUE" ];

oracle:namedModel [  
    oracle:namedModelURI <http://oracle.com/govtrack#GOV_VOTES_07>;  
    oracle:modelName "GOV_VOTES_07" ].
```
Jena Adapter Query Improvements: Performance

• Tight integration with Jena 2.6.2 and ARQ 2.8.1 for faster query performance

• Previously: Relying on Jena’s ARQ engine
  – ARQ responsible for generating query plan and performing joins
    • Calls SEM_MATCH for each BGP

• New Approach: hybrid ARQ/Oracle query answering
  – Translate SPARQL 1.0 queries into a single SEM_MATCH/SQL call
    • Allows use of RESULT_CACHE
    • If not possible, try again on next largest sub query
  – Fall back to Jena query engine for SPARQL 1.1 query constructs
    • E.g., nested subqueries, federated SPARQL queries, etc.
Query Answering Example

SPARQL Query

```
SELECT ?person ?name ?phone
WHERE
{ { ?person foaf:name ?name. }
UNION
{ { SELECT *
    WHERE
    { ?person taxinfo:name ?name
      OPTIONAL
      { ?person taxinfo:phone ?phone}
    }
    LIMIT 1
  }
}

123
```

Jena ARQ Algebra

```
(project (?person ?name ?phone)
(union
(bgp (triple ?x foaf:name ?name))
(slice _ 1
(conditional
(bgp (triple
 ?person tax:name ?name)))
(bgp (triple
 ?person tax:phone ?phone)))))
```

ORACLE® 123
Query Answering Example

```
PROJECT
?person ?name

UNION

BGP
?person foaf:name
?name

SLICE _ 1

CONDITIONAL

BGP
?person taxinfo:name
?name

BGP
?person taxinfo:phone
?phone
```
Property Path Queries

- Part of SPARQL 1.1.
  - Regular expressions for properties: \( ? + * ^ / | \)
- Translated to hierarchical SQL queries
  - Using Oracle CONNECT BY clause
- Examples:
  - Find all reachable friends of John
    - SELECT * WHERE { :John foaf:friendOf+ ?friend. }
  - Find reachable friends through two different paths
    - SELECT * WHERE {
      :John (foaf:friendOf|urn:friend)+ ?friend. }
  - Get names of people John knows one step away:
    - SELECT * WHERE { :John foaf:knows/foaf:name ?person }.
  - Find all people that can be reached from John by foaf:knows
    - SELECT * WHERE {
      ?x foaf:mbox <mailto:john@example> .
      ?x foaf:knows+/foaf:name ?name .
    }
Query Extensions in Jena Adapter

- Query management and execution control
  - Timeout
  - Query abort framework
    - Including monitoring threads and a management servlet
    - Designed for a J2EE cluster environment
  - Hints allowed in SPARQL query syntax
  - Parallel execution

- Support ARQ functions for projected variables
  - fn:lower-case, upper-case, substring, ...

- Native, system provided functions can be used in SPARQL
  - oext:lower-literal, oext:upper-literal, oext:build-uri-for-id, ...
Query Extensions in Jena Adapter

- Extensible user-defined functions in SPARQL
  
  Example
  
  ```sparql
  PREFIX ouext: <http://oracle.com/semtech/jena-adapter/ext/user-def-function#>
  SELECT ?subject ?object (ouext:my_strlen(?object) as ?obj1)
  WHERE { ?subject dc:title ?object }
  ```

  User can implement the `my_strlen` functions in Oracle Database

- Connection Pooling through OraclePool
  
  ```java
  java.util.Properties prop = new java.util.Properties();
  prop.setProperty("InitialLimit", "2"); // create 2 connections
  prop.setProperty("InactivityTimeout", "1800"); // seconds
  ...
  OraclePool op = new OraclePool(szJdbcURL, szUser, szPasswd, prop, "OracleSemConnPool");
  Oracle oracle = op.getOracle();
  ```
Best Practices using Jena Adapter

• Query options can be specified by overloading SPARQL PREFIX with Oracle-specific NS

• Use timeout and qid to control long-running queries:
  
  ```sparql
  PREFIX ORACLE_SEM_FS_NS: <http://oracle.com/semtech#timeout=3,qid=1234>
  ```

  Will time out the query if not finished after 3 seconds

• Use hints to influence optimizer plan:
  
  ```sparql
  PREFIX ORACLE_SEM_HT_NS: <http://oracle.com/semtech#leading(t0,t1), use_nl(t0,t1)>
  ```
Best Practices using Jena Adapter

• Various options to improve query performance:

```PREFIX ORACLE_SEM_FS_NS:
<http://oracle.com/semtech#DOP=4, INF_ONLY, ALLOW_DUP=T>
SELECT * WHERE {?subject ?property ?object }
```

• Performance options available:
  – ALLOW_DUP=T – allow duplicates with multi-model queries
  – DOP=n – degree of parallelism
  – INF_ONLY causes only the inferred model to be queried.
  – ORDERED
  – PLAIN_SQL_OPT=F disables the native compilation of queries directly to SQL.
  – RESULT_CACHE uses Oracle result caching.
Jena Adapter/Joseki Demo

- Joseki Setup
- SPARQL 1.1. Features
  - SPARUL
  - Property Paths
  - Federated SPARQL
- Query Management and Control
  - Timeout/Abort
  - Hints and Options (result_cache, parallel)
- User-Defined Functions
  - uppercase system provided fcn
  - select rdfvid, use vid2uri to get URI
Java APIs: Sesame Adapter
Sesame Adapter for Oracle 11g Release 2

• Provides a Java-based interface to Oracle semantic data management through Sesame APIs

• Why Sesame Adapter?
  – Sesame supports contexts
    • A Sesame statement is a quad (s, p, o, context)
    • Jena statements deal with triples
  – Customer requests
  – Integration with Sesame API-supporting tools

• Sesame Adapter implements the SAIL API
  – OracleSailConnection implements SailConnection
  – OracleSailStore implements Sail
An Example Usage Flow

SailRepository sr = new SailRepository(
    new OracleSailStore(oracle, modelName, ...attachmentForInference));
RepositoryConnection repConn = sr.getConnection();
    // data loading can happen here
repConn.add(alice, friendOf, bob, context1);

String query =
" PREFIX foaf: <http://xmlns.com/foaf/0.1/> " +
" PREFIX dc: <http://purl.org/dc/elements/1.1/> " +

TupleQuery tq = repConn.prepareTupleQuery(QueryLanguage.SPARQL, query);
TupleQueryResult tqr = tq.evaluate();
while(tqr.hasNext()) {
    psOut.print((tqr.next().toString()));
};
tqr.close();
Graph Queries in Sesame Adapter

PREFIX ORACLE_SEM_HT_NS: <http://oracle.com/semtech#leading(t0),dop=4>

- For each semantic model, a context column is added to the application table
  - <http://G1> will be matched against the context column
- An Oracle semantic model can contain multiple named graphs
  - In Jena Adapter, the correspondence was 1 : 1
Best Practices With Sesame Adapter Queries

- Same query hints/options available as in Jena Adapter
  - DOP, ORDERED, QID, TIMEOUT, etc.
- Context information is stored in application tables
  - Have to join application table with internal LINK$ for each triple pattern in GRAPH clause
  - For named graph query answering, minimize the number of patterns in the GRAPH clause
- A B-Tree index is created on the application table that contains CTXT, P, S, O
  - OracleSailStore supports user-specified indexes
Integration with Sesame Based Tools

• Sesame Server
  – SPARQL HTTP endpoint

• Sesame Console
  – command line based tool to create/manage repositories
  – We provide an Oracle specific console template for creating a model
  – console options:

  **connect** Connects to a (local or remote) set of repositories
  **disconnect** Disconnects from the current set of repositories
  **create** Creates a new repository
  **drop** Drops a repository
  **open** Opens a repository to work on, takes a repository ID as argument
  **close** Closes the current repository
  **show** Displays an overview of various resources
  **load** Loads a data file into a repository, takes a file path or URL as argument
  **verify** Verifies the syntax of an RDF data file, takes a file path or URL as argument
  **clear** Removes data from a repository
  **serql** Evaluates the SeRQL query, takes a query as argument
  **sparql** Evaluates the SPARQL query, takes a query as argument
  **set** Allows various console parameters to be set
  **exit, quit** Exit the console
Sesame Based Tools (2)

- **Sesame Workbench**
  - New/delete repository (w/o inference).
  - Add/Remove/Clear data.
  - Get namespaces, contexts; run queries ...

![Sesame Workbench](image)
Sesame Adapter Dependencies

- Oracle Database 11.2
- JDK 1.6
- WebLogic Server 10.3/ Tomcat 6/ OC4J
- Sesame 2.3.1
  - Sesame Console
  - Sesame Server (deployed in WebLogic Server, Tomcat)
  - Sesame Workbench (same deployment)
- Sesame Adapter contains converters from Jena to Sesame adapter-compatible Oracle models and back
Demo

• Sesame Console

• Sesame Server/Workbench
Semantic Technologies Utilities for Oracle Database
11g Release 2
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/rename_entailment**
Performance and Scalability Evaluation
Performance on Desktop PC
Bulk Loader Performance on Desktop PC: 11.2 Latest

<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(^2) Time (incl. Parse)</td>
<td>Sql*loader time (^3)</td>
</tr>
<tr>
<td>LUBM50 6.9 million</td>
<td>2.6min</td>
<td>0.4min</td>
</tr>
<tr>
<td>LUBM1000 138.3 million</td>
<td>1hr 10min</td>
<td>8 min</td>
</tr>
<tr>
<td>LUBM8000 1,106 million</td>
<td>9hr 15min</td>
<td>1hr 5min</td>
</tr>
</tbody>
</table>

- Used Core 2 Duo PC (3GHz), 8GB RAM, ASM, 3 SATA Disks (7200rpm), 64 bit Linux. Planned for an upcoming patchset.
- Empty network is assumed

\(^1\) This is an internal version of latest Oracle RDBMS 11.2
\(^2\) Uses flags=>' parse parallel=4 parallel_create_index ' plus a new as-yet-unnamed option for value processing
\(^3\) Uses parallel=true option and 8 to 10 gzipped N-Triple files as data files and a no-parse control file. \(^4\) Application table has table compression enabled.
\(^5\) Staging table has table compression enabled.
## 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;</td>
<td>Query</td>
</tr>
<tr>
<td>5.4 million inferred</td>
<td>Q1</td>
</tr>
<tr>
<td>Query</td>
<td># answers</td>
</tr>
<tr>
<td>Q1</td>
<td>4</td>
</tr>
<tr>
<td>Q2</td>
<td>130</td>
</tr>
<tr>
<td>Q3</td>
<td>6</td>
</tr>
<tr>
<td>Q4</td>
<td>34</td>
</tr>
<tr>
<td>Q5</td>
<td>719</td>
</tr>
<tr>
<td>Q6</td>
<td>519842</td>
</tr>
<tr>
<td>Q7</td>
<td>67</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
11.1.0.7 Inference Performance on Desktop PC

- OWLPrime (11.1.0.7) inference performance scales really well with hardware. It is *not* a parallel inference engine though.
### 11.2.0.1 Inference Performance on Desktop PC

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

• Setup: Intel Q6600 quad-core, 3 7200RPM SATA disks, 8GB DDR2 PC64000 RAM, No RAID. 64-bit Linux 2.6.18. **Assembly cost: less than USD 1,000**
Performance on Server
Load Performance on Server

- **LUBM1000 (138M triples)**
  - 8.3 minutes to load data into staging table
  - 78.8 minutes to load data from staging table (DOP=8)

- **LUBM8000 (1B+)**
  - 25 minutes to load data into staging table
  - 10hr 36 minutes to load data from staging table (DOP=8)

- **Setup:** Dual quad-core, Sun Storage F5100 Flash Array, 32 GB RAM
Inference Performance on Server

- Inference performance for LUBM1000 (138M)
  - 24.6 minutes to infer 108M+ new triples (DOP=8)

- Inference performance for LUBM8000 (1B+)
  - 226 minutes to infer 860M+ new triples (DOP=8)

- Setup: Dual quad-core, Sun Storage F5100 Flash Array, 32 GB RAM
Query Performance on Server

- Parallel query execution

[Bar chart: LUBM1000 Query Performance]

- Setup: Server class machine with 16 cores, NAND based flash storage, 32GB RAM, Linux 64 bit, Average of 3 warm runs
Performance on Sun Oracle Database Machine (Exadata V2)
Load Performance on Exadata V2

- **LUBM 25K benchmark ontology (3.3 Billion triples)**
  - *(Note: These are preliminary numbers and will be updated.)*
  - 105 minutes to load the data into staging table
  - 730 minutes for the bulk-load API, but with values pre-loaded

- **Setup:** Sun Oracle Data Machine and Exadata Storage Server (8 node cluster, Full Rack)

Inference Performance on Exadata V2

- **LUBM 25K benchmark ontology**
  (3.3 Billion triples)
  - OWLPrime inference with new inference components took 247 minutes (4 hours 7 minutes)
  - More than 2.7 billion new triples inferred
  - DOP = 32

- **Preliminary result on LUBM 100K benchmark ontology**
  (13 Billion+ triples)
  - One round of OWLPrime inference (limited to OWL Horst semantics) finished in 1.97 hours
  - 5 billion+ new triples inferred
  - DOP = 32

- **Setup:** Full Rack Sun Oracle Data Machine and Exadata Storage Server (8 node cluster)
### Query Performance on Exadata V2

<table>
<thead>
<tr>
<th>Ontology LUBM25K</th>
<th>LUBM Benchmark Queries</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.3 billion &amp; 2.7 billion inferred</td>
<td>Query</td>
</tr>
<tr>
<td>Query</td>
<td># answers</td>
</tr>
<tr>
<td>Complete?</td>
<td>Y</td>
</tr>
<tr>
<td>Time (sec)</td>
<td>0.01</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>6.8M</td>
<td>4</td>
<td>224</td>
<td>15</td>
<td>0.11M</td>
<td>197M</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>0.48</td>
<td>203.06</td>
<td>0.01</td>
<td>0.02</td>
<td>0.02</td>
<td>2.40</td>
<td>19.45</td>
</tr>
</tbody>
</table>

- Setup: Full Rack Sun Oracle Data Machine and Exadata Storage Server (8 node cluster)
- Auto DOP is used. Total # of answers 465,849,803 in less than 5 minutes
Summary: Oracle Database 11g Release 2

The only W3C standards compliant RDF/OWL data store w/ comprehensive capabilities for ....

**Reasoning and Discovery** supporting std. ontologies persistent, native & 3rd party inference, and user-defined rules

**Scalability** to evolve schemas dynamically and grow to 100’s billions of triples, incremental & parallel inference

**Data Integration** to link structured & unstructured content, Loosely couple business silos

**Security** to protect data on a “need to know” basis

**Integrated querying & manageability** SPARQL & SQL for RDF/OWL, relational, XML, text, location, & multimedia data
For More Information

search.oracle.com

Semantic Technologies

or

oracle.com
SOFTWARE. HARDWARE. COMPLETE.
Extended Example: Geo-Semantic Querying
Concrete Example

GovTrack (Semantic Data)
- Political Party Membership
- Voting Records
- Bill Sponsorship
- Committee Membership
- Offices and Terms

US Census (Spatial Data)
- Cartographic Boundary Files

Who is the representative for the district that contains Nashua, NH?

What Bills are sponsored by politicians that represent districts within 20 miles of Nashua, NH?
Oracle Spatial Query Capabilities

Filtering based on distance and topological spatial relationships

Select state and cd for the congressional district that contains a specific point

```sql
SELECT state, cd
FROM c_districts
WHERE SDO_RELATE(geom,
    SDO_GEOMETRY('POINT(-71.4644 42.7575)'),
    'MASK=CONTAINS') = 'TRUE';
```

---

```
SQL> desc c_districts;
Name                Type
------------------- -----------
STATE               VARCHAR2(2)
CD                 VARCHAR2(2)
LSAD               VARCHAR2(2)
NAME               VARCHAR2(90)
LSAD_TRANS         VARCHAR2(50)
GEOM               MDSYS.SDO_GEOMETRY
```

---

**MASK VALUES**

- TOUCH
- OVERLAPBDYDISJOINT
- OVERLAPBDYINTERSECT
- EQUAL
- INSIDE
- COVEREDBY
- CONTAINS
- COVERS
- ANYINTERACT
- ON
GeoSpatial Querying in an RDBMS

SQL-Based Approach

Oracle 11g

- Object-Relational Data (Spatial Objects)
- Triple Store (Semantic Data)
- Geo-Semantic SQL-Based Querying: SEM_MATCH + SDO_RELATE
Step 2: Combining the Data

C_DISTRICTS (Spatial Data Table)

<table>
<thead>
<tr>
<th>STATE</th>
<th>CD</th>
<th>LSAD</th>
<th>NAME</th>
<th>LSAD_TRANS</th>
<th>GEOM</th>
</tr>
</thead>
<tbody>
<tr>
<td>01</td>
<td>02</td>
<td>C2</td>
<td>2</td>
<td></td>
<td>Congressional District</td>
</tr>
</tbody>
</table>

GOV_TRACK (Semantic Model)

<table>
<thead>
<tr>
<th>PREFIX pol:</th>
<th>SUBJ</th>
<th>PRED</th>
<th>OBJ</th>
</tr>
</thead>
</table>

Strategy: Generate a URI for each row in C_DISTRICTS that matches the URI for the congressional district used in GovTrack
Step 2: Combining the Data

-- generate uris for c_districts table --
alter table c_districts add (geom_uri varchar2(4000));

update c_districts set geom_uri =
'http://www.rdfabout.com/rdf/usgov/geo/us/' ||
(case when state = '01' then 'al'
    when state = '02' then 'ak'
    ... ... ... ... ...
    when state = '56' then 'wy' end) ||
(case when cd = '00'
    then ''
    when regexp_instr(cd, '^0[1-9]') > 0
    then '/cd/111/' || substr(cd, 2)
    else '/cd/111/' || cd end);
GovTrack Semantic Model

Combined Data

Person

role: hasRole

Role

name: “Barney Frank”

Office

name: forOffice

name: represents

Congressional District

name: represents

name: forOffice

C_Districts Geometry Data

<table>
<thead>
<tr>
<th>STATE</th>
<th>CD</th>
<th>GEOM</th>
<th>GEOM_URI</th>
</tr>
</thead>
<tbody>
<tr>
<td>25</td>
<td>4</td>
<td>SDO_GEOMETRY(...)</td>
<td><a href="http://%E2%80%A6/us/ma/cd/110/4">http://…/us/ma/cd/110/4</a></td>
</tr>
<tr>
<td>…</td>
<td>…</td>
<td>…</td>
<td>…</td>
</tr>
</tbody>
</table>
Step 3: Geo-Semantic Querying

-- Basic Idea --
SELECT
   sem.a, ..., sem.z, geo.a, ..., geo.z
FROM
   ( SEM_MATCH_SUBQUERY ) sem,
   ( SPATIAL_SUBQUERY ) geo
WHERE
   sem.cdist = geo.geom_uri

Semantic Subquery
Geospatial Subquery
Join Condition based on Congressional District Polygons
Step 3: Geo-Semantic Querying

Select congressman and district for Nashua, NH

```
select sem.name, sem.cdist
from (select name, cdist from table (sem_match(?person usgovt:name ?name . ?person pol:hasRole ?role . ?role pol:forOffice ?office . ?office pol:represents ?cdist)' sem_models('GOV_TRACK'), null, sem_aliases(sem_alias('usgovt', 'http://…/usgovt/'), sem_alias('pol', 'http://…/politico/')), null, null))) sem,
( select geom_uri from c_districts where sdo_relate(geom, sdo_geometry('POINT(-71.46444 42.7575)', 524288), 'mask=CONTAINS') = 'TRUE' ) geo
where sem.cdist = geo.geom_uri;
```
Step 3: Geo-Semantic Querying

**Select congressman and district for Nashua, NH**

```sql
select sem.name, sem.cdist
from
    (select name, cdist
        from table(sem_match(
            '{?person usgovt:name ?name .
            ?person pol:hasRole ?role .
            ?office pol:represents ?cdist}',
            sem_models('GOV_TRACK'),null,
            sem_aliases(sem_alias('usgovt', 'http://…/usgovt/'),
                        sem_alias('pol', 'http://…/politico/')),
            null,null))) sem,

Spatial Subquery

```
Step 3: Geo-Semantic Querying

Select congressman and district for Nashua, NH

```
select sem.name, sem.cdist
from
  (select name, cdist
   from table(sem_match(''
      '?person usgovt:name ?name .
      ?person pol:hasRole ?role .
      ?office pol:represents ?cdist''
    ),
    sem_models('GOV_TRACK'),null,
    sem_aliases(sem_alias('usgovt', 'http://.../usgovt/'),
               sem_alias('pol', 'http://.../politico/'),
               null,null))) sem,
  (select geom_uri
   from c_districts
   where sdo_relate(geom,
      sdo_geometry('POINT(-71.46444 42.7575)', 524288),
      'mask=CONTAINS') = 'TRUE') geo
```

Join Condition
Step 3: Geo-Semantic Querying

```
SELECT congressman and district for Nashua, NH

SELECT sem.name, sem.cdist
FROM
(SELECT name, cdist
FROM table(sem_match('{
  ?person usgovt:name ?name .
  ?person pol:hasRole ?role .
  ?office pol:represents ?cdist}
',
sem_models('GOV_TRACK'),null,
sem_aliases(sem_alias('usgovt', 'http://.../usgovt/'),
            sem_alias('pol', 'http://.../politico/'),
            null,null))) sem,
(SELECT geom_uri
FROM c_districts
WHERE sdo_relate(geom,
    sdo_geometry('POINT(-71.46444 42.7575)', 524288),
    'mask=CONTAINS') = 'TRUE') geo
WHERE sem.cdist = geo.geom_uri;
```
Using the Sesame Console

- Provide an Oracle specific console template for creating a model
  - Analogous to Joseki’s declarative config files

- Example options:
  - Model_name, rulebases, use virtual model
  - Application table indexes,…
Query Answering

• Previously: Relying on Jena’s ARQ engine
  – ARQ responsible for generating query plan and performing joins
  • Calls SEM_MATCH for each BGP query

• New Approach: hybrid ARQ/Oracle query answering
  – Try to convert the whole query to a single SEM_MATCH call
  • If not possible, try again on next largest sub query
  • Rest of the query is handled by Jena
SEM_MATCH and Plain SQL Queries

- SEM_MATCH captures the following cases:
  - Plain BGP, UNION, FILTER, OPTIONAL
  - LIMIT/OFFSET, ORDER BY and projection vars not allowed in sub queries

- Another class of queries is translated directly to SQL:
  - Plain BGP and parallel OPTIONAL
  - LIMIT/OFFSET, ORDER BY and projection vars not allowed in subqueries
  - Conjunctive property paths
  - Why SQL?
    - Efficient property path queries (translated to CONNECT BY)
    - Can use RESULT_CACHE in query
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>

Patients
diagnosis
table

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’;

Zero Matches!

Traditional Syntactic query against relational data

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;
Semantic Indexing of Documents (SemContext Index)

CREATE INDEX ArticleIndex
ON NewsFeed (Article)
INDEXTYPE IS SemContext
PARAMETERS ('gate_nlp')

<table>
<thead>
<tr>
<th>DocId</th>
<th>Article</th>
<th>Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Indiana authorities filed felony charges and a court issued an arrest warrant for a financial …</td>
<td>CNN</td>
</tr>
<tr>
<td>2</td>
<td>Major dealers and investors in over-the-counter derivatives agreed to report all credit-default …</td>
<td>NW</td>
</tr>
<tr>
<td>..</td>
<td>..</td>
<td>..</td>
</tr>
</tbody>
</table>

CREATE INDEX ArticleIndex
ON NewsFeed (Article)
INDEXTYPE IS SemContext
PARAMETERS ('gate_nlp')

SELECT docId FROM Newsfeed
WHERE SEM_CONTAINS (Articles,
  '{?x rdf:type rc:Person .
   ?x pred:hasAge ?age .
   FILTER (?age >= 35)}) = 1
AND Source = 'CNN'

Newsfeed table

SemContext index on Article column

Triples table with rowid references

Analytical Queries On Graph Data

Analytical Queries On Graph Data
Virtual Private Database for RDF Data

[VPD] Access control policies on semantic data

Policy ➔ user can access value of projects s/he leads
Match pattern ➔ { ?x :hasValue ?v }
Apply pattern ➔ { ?x :hasLead "sys_context(...)" }

Query: Get the list of projects and their values

```
SELECT ?proj ?val 
FROM ProjectsGraph 
WHERE { ?proj :hasValue ?val .
            ?proj :hasLead "sys_context(...)" }
```
Oracle Label Security for RDF data

- Oracle Label Security enforces security at the “row” level.
  - Attempts to access key or non-key columns of a specific row are validated using row/data sensitivity labels.

<table>
<thead>
<tr>
<th>ContractID</th>
<th>Organization</th>
<th>ContractValue</th>
<th>RowLabel</th>
</tr>
</thead>
<tbody>
<tr>
<td>ProjectHLS</td>
<td>N. America</td>
<td>1000000</td>
<td>SE:HLS:US</td>
</tr>
</tbody>
</table>

- We provide two different levels of granularity for RDF OLS
  - Resource-level security
    - Each subject, predicate, object could be associated with a label
    - Upon inserting a triple, a user’s label has to dominate all of the component labels
  - Triple level security
    - Thin layer on top of OLS, no additional restrictions
Resource vs Triple Level Security

<table>
<thead>
<tr>
<th>Feature</th>
<th>Resource-Level</th>
<th>Triple-Level</th>
</tr>
</thead>
<tbody>
<tr>
<td>Resource constraints</td>
<td>Yes- provide a minimum bound for triples</td>
<td>No</td>
</tr>
<tr>
<td>Insert Model</td>
<td><strong>Closed</strong> - cannot insert a triple unless all resource labels are covered</td>
<td><strong>Open</strong> - anybody can say anything about anything, but the information is always tagged with the user’s label</td>
</tr>
<tr>
<td>Inference/Bulk Load</td>
<td>Has to be run as security admin/ various options on labels for inferred triples</td>
<td>Any user can do it; you can only perform inference on the data you can see; triples inserted with user’s default label</td>
</tr>
<tr>
<td>Setting triple label after insertion</td>
<td>Not available</td>
<td>Available</td>
</tr>
</tbody>
</table>
Virtual Private Database for RDF Data

Ability to restrict access to parts of the RDF graph based on the application/user context.

- **An individual can only access information about the project he or she works on.**

- **Monetary value of a project can only be accessed by the project lead or the VP of the department.**

- **Only a department VP can access information about the department’s budget.**
Change Mgmt./Versioning for Sem. Data

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

- An RDF Model is version-enabled by version-enabling the corresponding application table.
  - `exec DBMS_WM.enableVersioning (table_name => 'contracts_rdf_data');`
- RDF 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