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

Bill Beauregard, Senior Principal Product Mgr.
Souri Das, Ph.D., Oracle
Matthew Perry, Ph.D., Oracle
Karl Rieb, Oracle
Seema Sundara, Oracle

June 2011
Outline

• Overview
• Installation and configuration
• Loading
• Querying in SQL & SPARQL
• Inferencing
• Security
• Semantic Indexing of Unstructured Content
• Demo
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.
The Problem: Integration & Discovery

• Overcoming IT data silos
  – Application integration
  – Single view of enterprise
  – Risk mgt.
  • Master data mgt (MDM)
  • Data warehousing
  • Content mgt.

• Real time data access across silos

• Discovery of data relationships across...
  – Structured data (database, apps, web services)
  – Unstructured data (email, office documents)
  – Multiple data types (graphs, spatial, text, sensors)

• Enabling data reuse by associating more meaning (context) with the data
Benefits of RDF to Support Data Integration

• Model complex real-world relationships beyond tables and joins in the data as a graph

  Allow schemas to continuously and dynamically evolve

• Discover new information by inferencing among relationships with rules, standard concepts and terms

  Enable machine-driven discovery of relationships without restructuring the data model

• Obtain more semantically complete information for decision-making using graph pattern queries

  Support discovery workflows, navigate through the data based on relationships
Data Integration
Relational to Vocabulary Mapping

Compartments: Soma, Dendrite, Unspecified
Cells: CA1 pyramidal neuron, Olfactory bulb mitral cell
Receptors: 
- I_A
- I_L high threshold
Channels: 
- beta Amyloid
- NMDA

Pathological Change: involves
Agent: inhibits
Neuronal Property: has
Compartment: is_located_in
Drug: is_located_in

<table>
<thead>
<tr>
<th>Compartments</th>
<th>Cell: NeuronDB</th>
<th>Receptor</th>
<th>Channel</th>
<th>Pathological Agent (PA)</th>
<th>PA Action</th>
<th>Drug</th>
<th>Drug Action</th>
<th>Stage</th>
<th>Note</th>
<th>Detail</th>
</tr>
</thead>
<tbody>
<tr>
<td>Soma</td>
<td>CA1 pyramidal neuron</td>
<td>I_A</td>
<td>beta Amyloid</td>
<td>Inhibits</td>
<td></td>
<td></td>
<td></td>
<td>Early</td>
<td>View</td>
<td>66249</td>
</tr>
<tr>
<td></td>
<td>Olfactory bulb mitral cell</td>
<td>Gabaa</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Early</td>
<td>View</td>
<td>66750</td>
</tr>
<tr>
<td>Dendrite</td>
<td>CA1 pyramidal neuron</td>
<td>I_A</td>
<td>beta Amyloid</td>
<td>Inhibits</td>
<td></td>
<td></td>
<td></td>
<td>Early</td>
<td>View</td>
<td>66249</td>
</tr>
<tr>
<td></td>
<td>Olfactory bulb mitral cell</td>
<td>Gabaa</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Early</td>
<td>View</td>
<td>66750</td>
</tr>
<tr>
<td>Unspecified</td>
<td>Oocyte</td>
<td>I_L high threshold</td>
<td>beta Amyloid</td>
<td>Inhibits</td>
<td></td>
<td></td>
<td></td>
<td>Early</td>
<td>View</td>
<td>66252</td>
</tr>
<tr>
<td></td>
<td>CA1 pyramidal neuron</td>
<td>I_Calcium</td>
<td>beta Amyloid</td>
<td>Inhibits</td>
<td></td>
<td></td>
<td></td>
<td>Early</td>
<td>View</td>
<td>66750</td>
</tr>
<tr>
<td></td>
<td>CA1 pyramidal neuron</td>
<td>NMDA</td>
<td>beta Amyloid</td>
<td>Inhibits</td>
<td></td>
<td></td>
<td></td>
<td>View</td>
<td></td>
<td>66250</td>
</tr>
</tbody>
</table>
Integrated Bioinformatics Data

Source: Siderean Software
Use Cases That Benefit from Semantic Analysis

• Metadata management
• Integration
• Richer and extensible querying
• Graph analytics
• Content management
• Knowledge base
### Extract, Model, Reason & Discover Workflow

<table>
<thead>
<tr>
<th>Transform &amp; Edit Tools</th>
<th>Database Management</th>
<th>Applications &amp; Analysis Tools</th>
</tr>
</thead>
<tbody>
<tr>
<td>Entity Extraction &amp; Transform Ontology Eng. Categorization Custom Scripting</td>
<td>• RDF Data Loading • SPARQL / SQL • Native Inferencing • Semantic Rules • Open source integration • Scalability, Security, Versioning • Semantic Indexing</td>
<td>BI, Analytics Graph Visualization Social Network Analysis Metadata Registry Faceted Search</td>
</tr>
</tbody>
</table>

**Partner and Open Source Tools**

**ORACLE SPATIAL**

**Partner and Open Source Tools**

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

**Partner and Open Source Tools**
Importance of an RDF/OWL Database

- Scalable & secure platform scales to billions of triples
- RAC & Exadata scalability
- Compression & partitioning
- SQL*Loader direct path load
- Parallel load, inference, query
- Oracle DataGuard availability
- Triple-level DoD-strength security
- Choice of SPARQL or SQL
- Native inference engine
- W3C standards compliance
- Growing ecosystem of 3rd party tools partners

Key Capabilities:

**Load / Storage**
- Native RDF graph data store
- Manages billions of triples
- Optimized storage architecture

**Query**
- SPARQL-Jena/Joseki, Sesame
- SQL/graph query, b-tree indexing
- Ontology assisted SQL query

**Reasoning**
- RDFS, OWL2 RL, EL+, SKOS
- User-defined SWRL-like rules
- Incremental, parallel reasoning
- Plug-in architecture
Importance of W3C & OGC Semantic Standards

- Key W3C Web Semantic Activities:
  - W3C RDF Working Group
  - W3C SPARQL Working Group
  - W3C RDB2RDF Working Group
  - W3C OWL Working group
  - W3C Semantic Web Education & Outreach (SWEO)
  - W3C Health Care & Life Sciences Interest Group (HCLS)
  - W3C Multimedia Semantics Incubator group
  - W3C Semantic Web Rules Language (SWRL)

- OGC GeoSPARQL Standard Working Group
Industries Deploying Semantic Technologies

- Life Sciences
  - Lilly
  - Pfizer
- Defense/Intelligence
  - Geospatial Intelligence National Security Agency
  - Switzerland Institute of Bioinformatics
- Clinical Medicine & Research
  - Cleveland Clinic
- Education
  - The University of Michigan
- Telecomm & Networking
  - Cisco
  - Hutchinson 3G Austria
- Publishing
  - Westlaw, Thomson Reuters
Cisco Enterprise Collaboration Platform - QUAD

Chose RDF as the data model for sharing ideas and following people, communities, and information across the enterprise.

Chose database based on its scalability, fine-grained security, incremental inferencing, and support for standards.

- Billions of relationships
- Unifying RDF metadata model for
  - blogs, wikis, calendar, IM, WebEX, voice, and video
- Transactional workload requires incremental load & inference
- SPARQL graph queries
Text Mining: National Intelligence

Web Resources → Information Extraction → Categorization, Feature/term Extraction → RDF/OWL Processed Document Collection

RDF/OWL

Domain Specific Knowledge Base

Ontology Engineering Modeling Process

OWL Ontologies

Content Mgmt. Systems

SQL/SPARQL Query

Explore

Analyst

Browsing, Presentation, Reporting, Visualization, Query

News, Email, RSS

Analyst
Dreamworks Entertainment Repository

- Requires database to scale for millions of movie shot files, thousands of artists
- RDF graph describing a movie shot allows sharing and reuse
- UI uses SPARQL graph pattern query to find movie shots
Pfizer Re-use of Legacy Data

- Internal Compound Re-purposing
  - Save time and money by re-using internal compounds and associated research
  - Identify compounds across different databases and tools

- Why RDF
  - Store and represent any type of data
  - Ontology model changes easily as data & science change
  - Rapid response to changing customer needs

- Why the database
  - Combine SQL and graph queries
  - key facts in RDF, primary data relational and XML
  - Use in-house expertise of DBAs and database developers
<table>
<thead>
<tr>
<th>Semantic Technologies Partners</th>
<th>Integrated Tools and Solution Providers:</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Ontology Engineering</strong></td>
<td><strong>Reasoners</strong></td>
</tr>
<tr>
<td>TopQuadrant</td>
<td>clark-parsia, llc</td>
</tr>
<tr>
<td>protégé</td>
<td></td>
</tr>
<tr>
<td>ontoprise</td>
<td></td>
</tr>
<tr>
<td>know how to use Know-how</td>
<td></td>
</tr>
<tr>
<td><strong>Open Source Frameworks</strong></td>
<td><strong>Standards</strong></td>
</tr>
<tr>
<td>jena</td>
<td>W3C Semantic Web</td>
</tr>
<tr>
<td>open RDF.org</td>
<td></td>
</tr>
<tr>
<td>Sesame</td>
<td></td>
</tr>
<tr>
<td>Joseki</td>
<td></td>
</tr>
<tr>
<td><strong>Applications</strong></td>
<td><strong>SI / Consulting</strong></td>
</tr>
<tr>
<td>PolarLake</td>
<td></td>
</tr>
<tr>
<td>Mondeca</td>
<td></td>
</tr>
<tr>
<td>Teranode</td>
<td></td>
</tr>
<tr>
<td>MedTrust</td>
<td></td>
</tr>
<tr>
<td>infoMENTUM</td>
<td></td>
</tr>
<tr>
<td>Raytheon</td>
<td></td>
</tr>
<tr>
<td>Northrop Grumman</td>
<td></td>
</tr>
<tr>
<td>McDonald Bradley</td>
<td></td>
</tr>
<tr>
<td>accenture</td>
<td></td>
</tr>
<tr>
<td>Boeing</td>
<td></td>
</tr>
<tr>
<td>ORACLE</td>
<td></td>
</tr>
</tbody>
</table>
Oracle Database: Enterprise Database for RDF

- Scalability & performance for largest RDF applications
- Growing list of 3rd partner tooling
  - W3C & OGC standards support
  - Open source frameworks integration
- Native, persistent, extensible inferencing for discovery
- SPARQL/SQL integration
- Ontology-assisted SQL queries for more complete results
- DoD-strength triple-level label security
- Indexing of concepts and entity relationships found in unstructured content
Core Entities in Oracle Database Semantic Store

- **Sem. Network** ➔ Dictionary and data tables for storage and management of asserted and inferred RDF triples. OWL and RDFS rule bases are preloaded.

- **Model** ➔ A model holds an RDF graph (set of S-P-O triples).

- **Rulebase** ➔ A rulebase is a set of rules used for inferencing.

- **Entailments** ➔ An entailment stores triples derived via inferencing.

- **Application Table** ➔ Contains a column of type `sdo_rdf_triple_s`, associated with an RDF model, to allow DML and access to RDF triples, and storing ancillary values.
Core Functionality: Load / Query / Inference

- **Load** →
  - Bulk load
  - Incremental load

- **Query and DML** →
  - SPARQL (from Java/endpoint)

- **Inference** →
  - Native support for OWL 2 RL, SNOMED (OWL 2 EL subset), OWLprime, OWLSIF, RDFS+
  - Named Graph Local Inference
  - User-defined rules

Oracle Database

- Rulebases & Vocabularies
  - OWL subset
  - RDF / RDFS

- Rule base m

Application Tables

- Models
  - M1
  - M2
  - Mn

- Entailments
  - X1
  - X2
  - Xp

Semantic Network (MDSYS)

- Triples

- Values
Enterprise Functionality: SQL / Sem. Indexing / Security

- **SPARQL (embedding) in SQL**
  - Allows joining SPARQL results with relational data
  - Allows use of rich SQL operators (such as aggregates)

- **Semantic indexing**
  - Stores RDF triples as index info for documents stored in a table
  - Index content created using 3rd party information extractors

- **Security: Fine-Grained Access Control (for each triple)**
  - Uses Oracle Label Security (OLS)
  - Each RDF triple has an associated label

- **Querying Text and Spatial data using SPARQL**
Interfaces

- SQL-based (SQL and PL/SQL)

- Java-based
  - Jena (using Jena Adapter from Oracle)
  - Sesame (using Sesame Adapter from Oracle)

- SPARQL Endpoints
  - Joseki
  - OpenRDF Workbench
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$_vm$ (duplicate)</td>
</tr>
<tr>
<td></td>
<td>View mdsys.SEMU$_vm$ (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.
Architectural Overview

SPARQL Endpoints: Joseki / Sesame

3rd Party Tools: Topbraid Composer

Programming Interface:
- SPARQL: Jena / Sesame
- Java Programs
- JDBC

SQL Interface:
- SQLplus
- PL/SQL
- SQLdev.

Tools (cytoscope):
- Visualizer
- Reasoners: Pellet

3rd-Party Callouts:
- NLP Info. Extractor: Calais, GATE
- Reasoners: Pellet

Core functionality:
- LOAD
  - Bulk-Load
  - Incr. DML
- INFER
  - OWL subsets
  - RDF/S
  - User-def.
- QUERY (SPARQL in SQL)
  - Query RDF/OWL data and ontologies
  - Ontology-assisted Query of Enterprise Data

Oracle DB:
- RDF/OWL data and ontologies
- Rulebases: OWL, RDF/S, user-defined
- Inferred RDF/OWL data
- Semantic Indexes
- Enterprise (Relational) data
- Security: Oracle Label Security
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 TEMPORARY 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');
Loading RDF triples
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

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

• Create a staging table
  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
  sqlldr userid=scott/tiger control=simple.ctl data=/tmp/input1
  parallel=true direct=true skip=0 load=1990000000
discardmax=1900000000 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", RDFFormat.NTRIPLES, "SEMTS", null, null,
        "STAGE_TABLE", (Resource[]) null);  
    osc.commit(); fis.close();
}

- The latest Jena Adapter has prepareBulk and completeBulk APIs
```
Complete Data Loading

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

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

exec
    sem_apis.create_sem_model('myrdf', 'myrdf_tpl', '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

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]

• Loading RDFa
  - graphOracleSem.getBulkUpdateHandler().prepareBulk(rdfaUrl, ... )

[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
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**
Best Practices in Querying Semantic Data
Semantic Operators Expand Terms for SQL SELECT

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

<table>
<thead>
<tr>
<th>ID</th>
<th>DIAGNOSIS</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Hand_Fracture</td>
</tr>
<tr>
<td>2</td>
<td>Rheumatoid_Arthritis</td>
</tr>
</tbody>
</table>

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

Syntactic query against relational table will not work!

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

Zero Matches!

**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;
```
Outline

• Description of SEM_MATCH capabilities

• Live demo

• Indexes for SEM_MATCH

• Performance best practices
SEM_MATCH: Adding SPARQL to SQL

• Extends SQL with SPARQL constructs
  – Graph Patterns, OPTIONAL, UNION
  – Dataset Constructs
  – FILTER – including SPARQL built-ins
  – Prologue
  – Solution Modifiers

• Benefits:
  – Allows SQL constructs/functions:
  – JOINs with other object-relational data
  – DDL Statements: create tables/views
SELECT n1, n2
FROM
TABLE(
    SEM_MATCH(
        '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")) }
        ORDER BY ?n1 ?n2',
    SEM_MODELS('M1'),...));
**SEM_MATCH: Adding SPARQL to SQL**

**SQL Table Function**

```sql
SELECT n1, n2
FROM

TABLE(

<table>
<thead>
<tr>
<th>n1</th>
<th>n2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alex</td>
<td>Jerry</td>
</tr>
<tr>
<td>Alex</td>
<td>Tom</td>
</tr>
<tr>
<td>Alice</td>
<td>Bill</td>
</tr>
<tr>
<td>Alice</td>
<td>Jill</td>
</tr>
<tr>
<td>Alice</td>
<td>John</td>
</tr>
</tbody>
</table>

) SEM_MODELS('M1'),…
```
SEM_MATCH: Adding SPARQL to SQL

Rewritable SQL Table Function

```sql
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,
    options
);

'SELECT ?a
WHERE { ?a foaf:name ?b }'

Container(s) for asserted quads

Basic unit of access control

Entailed triples

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

'ALLOW_DUP=T STRICT_TERM_COMP=F'
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

GOV_ASSERT_VM
- Asserted data only
  (2.8M triples)

GOV_ALL_VM
- Asserted + Inferred
  (3.1M triples)

GOV_TBOX

GOV_PEOPLE

GOV_BILLS_110

GOV_BILLS_111

GOV_VOTES_07

GOV_VOTES_08

GOV_VOTES_09

GOV_DISTRICTS (US Census)

GOV_TRACK_OWL

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

U.S. Census Bureau
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

```sql
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', 'gov_districts'));

    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', 'gov_districts'),
        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: Basic Query

Find information about all Kennedys

```sql
select fn, bday, g, t, hp, r
from table(sem_match('SELECT ?fn ?bday ?g ?t ?hp ?r
WHERE
{ ?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, null, null,null,' ALLOW_DUP=T '));
```
Live Example 2: OPTIONAL Query

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

```
select fn, bday, g, t, hp, r
from table(sem_match('SELECT ?fn ?bday ?g ?t ?hp ?r
WHERE
  { ?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, null, null, null, ' ALLOW_DUP=T ' ));
```
Live Example 3: Simple FILTER

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

```sql
select fname, lname
from table(sem_match(
  'SELECT ?fname ?lname
  WHERE
  { ?s rdf:type foaf:Person .
    ?vcard vcard:Given ?fname .
    ?vcard vcard:Family ?lname
    FILTER (STR(?lname) < "B") }',
  sem_models('gov_all_vm'), null, null, null
  ,null, ' ALLOW_DUP=T ' ));
```
Live Example 4: Negation as Failure

Find all Lincolns without a homepage

```sql
select fn, bday, hp
from table(sem_match('SELECT ?fn ?bday ?hp
    WHERE
    { ?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, null, null ,null, ' ALLOW_DUP=T '));
```
Live Example 5: 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('SELECT ?title ?dt
WHERE
{ ?s foaf:name "Barack Obama" .
  { ?b bill:sponsor ?s } 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, null, null ,null, ' ALLOW_DUP=T ' ));
```
Live Example 6: Inference

GovTrack Bill Types

Diagram showing the hierarchy and relationships between different types of legislative documents and bills.
Live Example 6: Inference

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

```sql
select title, dt, btype
from table(sem_match(
  'SELECT ?title ?dt ?btype
  WHERE
  { ?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, null, null
  ,null, 'ALLOW_DUP=T'
));
```
Live Example 7: SQL Constructs (temporal interval computations)

Find the youngest person to take office since 2000

```sql
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('SELECT ?fn ?bay ?tsfrom
  WHERE
    ?role time:from ?tfrom . ?tfrom time:at ?tsfrom
    FILTER (?tsfrom >= "2000-01-01"^^xsd:date)
  }
  ,sem_models('gov_all_vm'), null, null, null))
order by (to_date(tsfrom,'YYYY-MM-DD') -  
    to_date(bday,'YYYY-MM-DD'))) asc)
where rownum <= 1;
```
Indexes for SEM_MATCH
## Basic RDF Data Storage Tables

### RDF_VALUE$ Table

<table>
<thead>
<tr>
<th>vname_prefix</th>
<th>vname_suffix</th>
<th>Value_id</th>
<th>literal_type</th>
<th>language_type</th>
<th>long_value</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>g_id</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>300</td>
<td>200</td>
<td>1</td>
<td>400</td>
</tr>
</tbody>
</table>

---

**Partition**

- **Partition M1**
- **Partition M2**
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,G. (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)`
Semantic Network Indexes (cont.)

Multi-column indexes speed up query execution significantly

Example

\[
\{ \ ?a \text{ foaf:knows} \ ?b \ . \ #t0 \\
\ ?b \text{ foaf:name} \ ?n \ . \ #t1 \\
\}
\]

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 ‘PSCGM’ index can do this very efficiently with a single index scan
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
    - $PCSGM$ – always there … enforces a uniqueness constraint on RDF_LINK$
    - $PSCGM$
    - These indexes perform well in the common case (constant URI in predicate position)
Datatype Indexes for FILTERs

Find all Persons with last names between “Pa” and “Pb”

select fname, lname
from table(sem_match(
'SELECT ?fname ?lname
WHERE
{ ?s vcard:N ?vcard .
  ?vcard vcard:Given ?fname .
  ?vcard vcard:Family ?lname
FILTER (?lname >= "Pa" &&
     ?lname < "Pb") }
,sem_models('gov_all_vm'), null
,null, 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> string literal)
Datatype indexes for FILTERs

• Convenient API
  - `sem_apis.add_datatype_index(<URI>)`
  - `sem_apis.drop_datatype_index(<URI>)`
  - `sem_apis.alter_datatype_index(<URI>, command)`

• Supported Datatypes:
  - xsd numeric types
  - xsd:string and plain literal
  - xsd:dateTime

• Oracle Extensions
  - spatial (http://xmlns.oracle.com/rdf/geo/WKTLiteral)
  - text (http://xmlns.oracle.com/rdf/text)

```sql
SQL> exec sem_apis.add_datatype_index(
            'http://www.w3.org/2001/XMLSchema#string');
```
Oracle Extensions for Text and Spatial
Full Text Indexing with Oracle Text

- Filters graph patterns based on text search string
- Indexes all RDF Terms
  - URIs, Literals, Language Tags, etc.
- Provide SPARQL extension function
  - `orardf:.RegularExpressions(?var, "Oracle text search string")`
  - Search String
    - Group Operators: AND, OR, NOT, NEAR, ...
    - Term Operators: stem($), soundex(!), wildcard(%)
Find all bills about Children and Taxes

```sql
select s, title, dt
from table(sem_match('SELECT ?s ?title ?dt
WHERE
{ ?b bill:sponsor ?s .
  ?s foaf:name ?n .
  ?b bill:introduced ?dt
FILTER (orardf:textContains(?title,

  "$children AND $taxes")
})
,sem_models('gov_all_vm'), null, null, null
,null, ' ALLOW_DUP=T ')
);```

Spatial Support with Oracle Spatial

- Support geometries encoded as orageo:WKTLiterals

:semTech2011 orageo:hasPointGeometry "POINT(-122.4192 37.7793)"^^orageo:WKTLiteral.

- Provide library of spatial query functions

SELECT ?s
WHERE { ?s orageo:hasPointGeometry ?geom
FILTER(orageo:withinDistance(?geom, "POINT(-122.4192 37.7793)"^^orageo:WKTLiteral, "distance=10 unit=KM"))}
orageo:WKTLiteral Datatype

- Optional leading Spatial Reference System URI followed by OGC WKT geometry string.
  <http://xmlns.oracle.com/rdf/geo/srid/{srid}>
- WGS 84 Longitude, Latitude is the default SRS (assumed if SRS URI is absent)

SRS: WGS84 Longitude, Latitude
"POINT(-122.4192 37.7793)"^^orageo:WKTLiteral

SRS: NAD27 Longitude, Latitude
"<http://xmlns.oracle.com/rdf/geo/srid/8260>
  POINT(-122.4181 37.7793)"^^orageo:WKTLiteral

- Prepare for spatial querying by creating a spatial index for the orageo:WKTLiteral datatype

SQL> exec sem_api.add_datatype_index(
    'http://xmlns.oracle.com/rdf/geo/WKTLiteral',
    options=>'TOLERANCE=1.0 SRID=8307
      DIMENSIONS=((LONGITUDE,-180,180)(LATITUDE,-90,90))');
What Types of Spatial Data are Supported?

• Spatial Reference Systems
  – Built-in support for 1000’s of SRS
  – Plus you can define your own
  – Coordinate system transformations applied transparently during indexing and query

• Geometry Types
  – Support OGC Simple Features geometry types
    • Point, Line, Polygon
    • Multi-Point, Multi-Line, Multi-Polygon
    • Geometry Collection
  – Up to 500,000 vertices per Geometry
Spatial Function Library

• Topological Relations
  - `orageo:relate`

• Distance-based Operations
  - `orageo:distance`, `orageo:withinDistance`,
  - `orageo:buffer`, `orageo:nearestNeighbor`

• Geometry Operations
  - `orageo:area`, `orageo:length`
  - `orageo:centroid`, `orageo:mbr`,
  - `orageo:convexHull`

• Geometry-Geometry Operations
  - `orageo:intersection`, `orageo:union`,
  - `orageo:difference`, `orageo:xor`
GovTrack Spatial Demo

- Congressional District Polygons (435)
  - Complex Geometries
  - Average over 1000 vertices per geometry

Load .shp file from US Census into Oracle Spatial

Generate triples using sdo_util.toWKTGeometry()

Load into Oracle semantic model
Which congressional district contains Nahsua, NH

```
select name, cdist
from table(sem_match('SELECT ?name ?cdist
  WHERE
  { ?person usgovt:name ?name .
    ?person pol:hasRole ?role .
    ?cdist orageo:hasWKTGeometry ?cgeom
    FILTER (orageo:relate(?cgeom,
    "POINT(-71.46444 42.7575)"^^orageo:WKTLiteral,
    "mask=contains") ) } ',sem_models('gov_all_vm'), null, null, null ,null, ' ALLOW_DUP=T ' ));
```
Who are my nearest 10 representatives ordered by centerpoint

```
select name, cdist
from table(sem_match(
  'SELECT ?name ?cdist
WHERE
  { ?person usgovt:name ?name .
  ?person pol:hasRole ?role .
  ?cdist orageo:hasWKTTGeometry ?cgeom
FILTER (orageo:nearestNeighbor(?cgeom,
  "POINT(-71.46444 42.7575)"^^orageo:WKTLiteral,
  "sdo_num_res=10") ) }
ORDER BY ASC(orageo:distance(orageo:centroid(?cgeom),
  "POINT(-71.46444 42.7575)"^^orageo:WKTLiteral,
  "unit=KM") )
,sem_models('gov_all_vm'), null, null, null
,null, ' ALLOW_DUP=T '));
```
Best Practices for Query Performance
Basic Performance Tips

• Database Initialization Parameters
  – sga_target, pga_aggregate_target, db_cache_size, etc.

• Reduce VALUE$ Joins
  – Only select a query variable if you truly need to
  – Use the VAR$RDFVID column
  – Use sameTerm(A, B) instead of A = B

• Query Options
  – ‘ALLOW_DUP=T’ – relax set semantics for multi-model queries

• Use Virtual Models
  – Internal query simplifications
  – Convenience: fast switching of new/updated graphs, simplified access control
Getting Good Query Execution Plans
Tip 1: Always Gather Statistics

- **SEM_APIs Procedures (local)**
  - `ANALYZE_MODEL`
  - `ANALYZE_ENTAILMENT`

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

```sql
SQL> exec semApis.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: Estimating Selectivity with Dynamic Sampling

- Triples table is not well suited for traditional optimizer statistics
  - Usually access table with multiple constraints
    - \( (p\text{-}value\_id = 123 \text{ AND } start\_node\_id = 456) \)
  - Too many possible combinations of values
  - No meaningful ordering, so histograms don’t work well

- One Solution: *Dynamic Sampling*
  - Dynamically compute selectivity of each triple pattern based on a small sample
  - Determine triple pattern execution order with dynamic estimates
  - Configured with integer parameter (0 – 10), 2 is default

```sql
SQL> alter session set optimizer_dynamic_sampling = 3;
SQL> select /*+ dynamic_sampling(6) */
  from table (sem_match(…));
```

Recommended Range: 3 - 6
Tip 3: Query Optimizer Hints: HINT0 Framework

Find all Persons with last names between “Pa” and “Pb”

```
select fname, lname
from table(sem_match(
  'SELECT ?fname ?lname
  WHERE
  { # HINT0={ LEADING(?lname)
    # INDEX(?lname rdf_v$str_idx)
    # USE_NL(t0 t1 t2 ?fname ?lname) }
  ?s vcard:N ?vcard .       # t0
  ?vcard vcard:Given ?fname .  # t1
  ?vcard vcard:Family ?lname  # t2
  FILTER (?lname >= "Pa" &&
    ?lname < "Pb") }
,sem_models('gov_all_vm'), null ,null, null, null,' ALLOW_DUP=T '));
```
Tip 4: Use Parallel Query Execution

- Oracle parallel SQL engine is highly optimized
- Critical for hash join on very large datasets
  - Available memory is proportional to number of parallel threads

```
SQL> alter session enable parallel query;
SQL> alter session force parallel query;
SQL> alter session force parallel query parallel 4;
SQL> alter session set parallel_degree_policy = AUTO;

SQL> select /*+ PARALLEL */ ... from table (sem_match(...));

SQL> select /*+ PARALLEL(4) */ ... from table (sem_match(...));
```

Experiment with your data, queries and hardware
SPARQL Querying
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)

• 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:

```xml
#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>;
  joseki:processor: Joseki:ProcessorSPARQL_FixedDS;
```
SPARQL Endpoint: Example

- Example Joseki Dataset configuration:

```sparql
<#oracle> rdf:type oracle:Dataset;
    joseki:poolSize 4;  # Number of concurrent connections allowed to this dataset.
    oracle:connection [ a oracle:OracleConnection ; ];

oracle:defaultModel [
    oracle:firstModel "GOV_PEOPLE";
    oracle:modelName "GOV_TBOX";
    oracle:modelName "GOV_VOTES_07";
    oracle:rulebaseName "OWLPRIME";
    oracle:useVM "TRUE" ];

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

- Tight integration with Jena 2.6.3 and ARQ 2.8.5 for faster query performance
- Previously: Relying on Jena’s ARQ engine
  - ARQ responsible for generating query plan and performing joins
    - Single SQL query for each BGP
- New Approach: hybrid ARQ/Oracle query answering
  - Translate SPARQL 1.0 queries into a single SQL query
    - 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
}
}
}
```

**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))))))
```
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
    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.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();
Federated Query and Bind Variables

• Choose the right join option to improve performance of federated queries

  PREFIX ORACLE_SEM_FS_NS: <http://oracle.com/semtech#join=n>
  SELECT ?s ?s1 ?o
  WHERE {
    ?s1 ?p1 ?s . {
      SERVICE <http://sparql.org/books> { ?s ?p ?o }
    }
  }

• Using bind variables in SPARQL queries

  PREFIX ORACLE_SEM_FS_NS: <http://oracle.com/semtech#s2s>
  PREFIX ORACLE_SEM_UEAP_NS: <http://oracle.com/semtech#x$RDFVID%20in(?,?,?)>
  PREFIX ORACLE_SEM_UEPJ_NS: <http://oracle.com/semtech#x$RDFVID>
  PREFIX ORACLE_SEM_UEBV_NS: <http://oracle.com/semtech#1,2,3>
  SELECT ?subject ?x
  WHERE { ?subject <urn:related> ?x }
Best Practices for 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 for 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.
  - s2s (sparql2sql), midtier_cache, BEST_EFFORT_QUERY=t
Mid-Tier Caching

- Stores the mapping from internal resource IDs to lexical forms in memory
- Adopts a very **compact** data structure for the mapping
- Simplifies queries to be executed in the database
- Improves query performance (less joins)

- Cache is populated
  - by invoking `graphOracleSempopulateCache()`, or
  - on demand when a mapping is not found in memory

- **Usage:**
  
  ```sparql
  PREFIX ORACLE_SEM_FS_NS: <http://oracle.com/semtech#midtier_cache>
  SELECT ...
  ... <SPARQL QUERY> ...
  ```
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
Inference
Core Inference Features in Oracle Database

- 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 (subset of OWL 2 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* vocabulary¹
  – OWLPrime
    • Includes RDFS++, a substantial subset of OWL
    • 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 (subset of OWL 2 EL)

• **Choice of rulebases**
  • If ontology is in EL, choose SNOMED component
  • If OWL 2 features (chains, keys) are not used, choose OWLPrime
  • Choose OWL2RL otherwise.

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:
    \[ p \text{ rdfs:domain } x . \]
    \[ u \text{ p } y . \]
    \[ \Rightarrow u \text{ rdf:type } x . \]

• OWL 2 RL has 70+ entailment rules.
  – E.g. rule:
    \[ p \text{ rdf:type owl:FunctionalProperty} . \]
    \[ x \text{ p } y1 . \]
    \[ x \text{ p } y2 . \]
    \[ \Rightarrow y1 \text{ owl:sameAs } y2 . \]
    \[ x \text{ owl:disjointWith } y . \]
    \[ a \text{ rdf:type } x . \]
    \[ b \text{ rdf:type } y . \]
    \[ \Rightarrow a \text{ owl:differentFrom } b . \]

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

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

**Jena Adapter API:** GraphOracleSem.performInference()
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.
Extending Semantics Supported by 11.2 OWL Inference

• 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
  ```
  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
  ```
  EXECUTE sem_apis.create_entailment ('M_IDX',sem_models('M'),
  sem_rulebases('OWLPRIME'),null,null, 'INC=T');
  ```

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

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

- OWL2RL/SKOS inference
  ```
  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 sem_api.s_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
Named Graph Based Global/Local Inference

- Named Graph Based Global Inference (NGGI)
  - Perform inference on just a subset of the triples
  - Some usage examples
    - Run NGGI on just the TBox
    - Run NGGI on just a single named graph
    - Run NGGI on just a single named graph and a TBox

- Named Graph Based Local Inference (NGLI)
  - Perform local inference for each named graph (optionally with a common TBox)
    - Triples from different named graphs will not be mixed together.
  - NGGI and NGLI together can achieve efficient named graph based inference maintenance
Inference Demo
Enterprise Security for Semantic Data
Enterprise Security for Semantic Data

• Model-level access control
  • Each semantic model accessible through a view (RDFM\_modelName)
    • Grant/revoke privileges on the view
    • Discretionary access control on application table for model

• Finer granularity possible through Oracle Label Security
  • Triple level security
  • Mandatory Access Control
Oracle Label Security

• Oracle Label Security – Mandatory Access Control
  • Data records and users tagged with security labels
  • Labels determine the sensitivity of the data or the rights a person must possess in order to read or write the data.

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

• User labels indicate their access rights to the data records.
  • For reads/deletes/updates: user’s label must dominate row’s label
  • For inserts: user’s label applied to inserted row
• A Security Administrator assigns labels to users
OLS Data Classification

Label Components:

- **Levels** – Determine the vertical sensitivity of data and the highest classification level a user can access.
- **Compartments** – Facilitate compartmentalization of data. Users need exclusive membership to a compartment to access its data.
- **Groups** – Allow hierarchical categorization of data. A user with authorization to a parent group can access data in any of its child groups.

CONF : NAVY, MILITARY : NY, DC \(\rightarrow\) Row Label matches User Access Label

\[\downarrow\]

HIGHCONF : MILITARY, NAVY, SPCLOPS : US, UK
• Sensitivity labels associated with individual triples control read access to the triples.

• Triples describing a single resource may employ different sensitivity labels for greater control.
Securing RDF Data using OLS: Example (1)

- Create an OLS policy
  - Policy is the container for all the labels and user authorizations
  - Can create multiple policies containing different labels

- Create label components
  - Levels:
    - UN (unclassified) < SE (secret) < TS (top secret)
  - Compartments:
    - HLS (Homeland Security), CIA, FBI
  - Groups:
    - NY, DC → EASTUS → US
    - SD, SF → WESTUS

- Create labels
  - “EASTSE” = SE:CIA,HLS:EASTUS
  - “USUN” = UN:FBI,HLS:US
Securing RDF Data using OLS: Example (2)

- Assign labels to users
  - John
    “EASTSE” (SE:CIA,HLS:EASTUS)
    - John can read SE and UN triples
    - John can read triples for CIA and HLS
    - John can read triples for NY, DC, and EASTUS
    - When inserting a row, the default write label is “EASTSE”
  
  - Mary
    “USUN” (UN:FBI,HLS:US)
    - Mary can only read UN triples
    - Mary can read triples for FBI and HLS
    - Mary can read all group triples (e.g. SF, NY, WESTUS, etc)
    - When inserting a row, the default write label is “USUN”
Securing RDF Data using OLS: Example (3)

- Apply the OLS policy to RDF store
  - Triple inserts, deletes, updates, and reads will use the policy

- John inserts triple:
  `<http://John> <rdf:type> <http://Person>`

- Mary inserts triple:
  `<http://Mary> <rdf:type> <http://Person>`

- Both these triples inserted in model but tagged with different label values ("EASTSE", "USUN")

- Users can have multiple labels
  - Only one label active at any time (user can switch labels)
  - Only active label applied to operations (e.g. queries, deletes, inferred triples)
Securing RDF Data using OLS: Example (4)

- Example labels and read access

<table>
<thead>
<tr>
<th>“EASTSE”</th>
<th>John Read</th>
<th>Triple Label</th>
<th>Mary Read</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>No</td>
<td><em>TS</em>: HLS: DC</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>No</td>
<td><em>SE</em>: HLS, FBI: DC</td>
<td>No</td>
</tr>
<tr>
<td><strong>Yes</strong></td>
<td></td>
<td><em>UN</em>: HLS: DC</td>
<td><strong>Yes</strong></td>
</tr>
<tr>
<td><strong>Yes</strong></td>
<td></td>
<td><em>UN</em>: HLS, CIA: NY</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>No</td>
<td><em>SE</em>: CIA: SF</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>No</td>
<td><em>UN</em>: HLS, FBI: NY</td>
<td><strong>Yes</strong></td>
</tr>
<tr>
<td></td>
<td>No</td>
<td><em>UN</em>: HLS: SF</td>
<td><strong>Yes</strong></td>
</tr>
</tbody>
</table>
Securing RDF Data using OLS: Example (5)

- Same triple may exist with different labels:
  
  `

- When Mary queries, only 1 triple returned (UN triple)
- When John queries, both UN and SE triples are returned
  - No way to distinguish since we don’t return label information!
  - Solution: use MIN_LABEL option in SEM_MATCH
  - This query will filter out triples that are dominated by SE:
    
    ```sql
    SELECT s,p,y
    FROM table(sem_match('{?s ?p ?y}' , sem_models(TEST'),
    null, null, null, null,
    'MIN_LABEL=SE POLICY_NAME=DEFENSE'))
    ```
  - MIN_LABEL can be used to filter out untrustworthy data
Semantic Indexing for Unstructured Content
Overview: Creating and Using a Semantic Index

CREATE INDEX ArticleIndex
ON Newsfeed (Article)
INDEXTYPE IS SemContext
PARAMETERS ('my_policy')
LOCAL PARALLEL 4

Newsfeed table

<table>
<thead>
<tr>
<th>Rowid</th>
<th>docId</th>
<th>Article</th>
<th>p_date</th>
</tr>
</thead>
<tbody>
<tr>
<td>r1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>r2</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Triples table with rowid references

<table>
<thead>
<tr>
<th>Subject</th>
<th>Property</th>
<th>Object</th>
<th>Graph</th>
</tr>
</thead>
</table>

SemContext index on Article column

Local 1 parallel 4 extractor

SELECT Sem_Contains_Select(1)
FROM Newsfeed
WHERE Sem_Contains (Article,
‘{?x rdf:type rc:Person .
  ?x :hasAge ?age .
  FILTER(?age >= 35)}’,1) = 1
AND p_date > to_date('01-Jan-11')

1 LOCAL index support for semantic indexing is restricted to range-partitioned base tables only.
Semantic Indexing - Key Components

- **Extensible Information Extractor**
  - Programmable API to plug-in 3rd party extractors into the database.

- **SemContext Indextype**
  - A custom indexing scheme that interacts with the extractor to manage the metadata extracted from the documents efficiently and facilitates semantic search via SQL queries.

- **SEM_CONTAINS Operator**
  - To identify documents of interest based on their extracted metadata, using standard SQL queries.

- **SEM_CONTAINS_SELECT Ancillary Operator**
  - To return additional information (SPARQL Query Results XML) about the documents identified using SEM_CONTAINS operator.
Semantic Indexing - Key Concepts

• Policy
  – Base Policy: <policy_name, extractor_type>
  – Dependent Policy: <policy_name, base_policy_name, knowledge_bases, entailments>

• Association between indexes and policies
  – Multiple policies may be associated with an index
  – Triples extracted using each base policy is stored separately

• Policy for use with a Sem_Contains invocation
  – can optionally be specified by user

• Inference
  – Document-centric: uses named graph local inference (NGLI)
  – Corpus-centric
Inference: document-centric

**Ontology:** schema triples (for *extracted* data)

```
<John> rdf:type <Parent> <.../r1>
<John> <grewUpIn> <NYC> <.../r1>
<John> rdf:type <Man> <.../r2>
... ... ...
```

**RDF model:** set of *extracted* triples

```
<John> rdf:type <Adult> <.../r1>
<John> <familiarWith> <NYC> <.../r1>
<John> rdf:type <Male> <.../r2>
... ... ...
```

**Entailment:** set of *inferred* triples

```
<John> rdf:type <Parent> <.../r1>
<John> <grewUpIn> <NYC> <.../r1>
<John> rdf:type <Man> <.../r2>
... ... ...
```
Combining Ontologies with extracted triples

• The triples extracted from documents can be combined with global domain ontologies for added value.
• User-defined models with triples that apply to all the documents and corresponding entailment can be associated with the Extractor policy.

```
begin
    sem_rdfctx.create_policy (policy_name => 'my_policy_plus_geo', base_policy => 'my_policy', user_models => SEM_MODELS('USGeography'), user_entailments => SEM_MODELS('Doc_inferred', 'USGeography_inferred'));
end;
```

SELECT docId FROM Newsfeed
WHERE SEM_CONTAINS (Articles, '{ ?comp  rdf:type c:Company .
    ?city  geo:state        "NY"^^xsd:string}', 'my_policy_plus_geo') = 1

Will result in a multi-model query involving an my_policy Index Model, the USGeography model and the entailments.
Improved Semantic Search with Feedback

- The triples extracted from a document can be edited for improved search results.
  - Allows combining triples extracted from multiple extraction tools.
  - Allows extension of the knowledge base with community feedback.

```ruby
begin
  sem_rdfctx.maintain_triples (
    index_name => 'ArticleIndex',
    policy_name => 'my_policy',
    where_clause => 'docId in (18,36,198)',
    rdf_content =>
      sys.xmlType('<rdf:RDF>
        <rdf:Description rdf:about="..">
          <rdf:type rdf:resource=".."/>
          <p:location rdf:resource=".."/>
          ...
        </rdf:Description>
      </rdf:RDF>'),
    action => 'ADD');
end;
```
Abstract Extractor Type

• An abstract extractor type (in PL/SQL) defines the common interfaces for all extractor implementations.

• Specific implementations for the abstract type interact with individual third-party extractors and produce RDF/XML documents for the input document.

• Network based extractors may extend the Web Service extractor type, which encapsulates web service callouts.
A sample extractor type -- interface

create or replace type rdfctxu.info_extractor under rdfctx_extractor (  
  overriding member function getDescription return VARCHAR2,  
  overriding member function rdfReturnType return VARCHAR2,  
  overriding member function extractRDF(  
    document CLOB, docId VARCHAR2, params VARCHAR2 ...)  
  return CLOB,  
  overriding member function getContext(attribute VARCHAR2)  
  return VARCHAR2,  
  overriding member function batchExtractRdf(  
    docCursor SYS_REFCURSOR,  
    extracted_info_table VARCHAR2,  
    params VARCHAR2,  
    partition_name VARCHAR2 default NULL ...)  
  return CLOB)
Ontology-assisted Querying of Relational Data
Semantic Operators Expand Terms for SQL SELECT

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

<table>
<thead>
<tr>
<th>ID</th>
<th>DIAGNOSIS</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Hand_Fracture</td>
</tr>
<tr>
<td>2</td>
<td>Rheumatoid_Arthritis</td>
</tr>
</tbody>
</table>

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

Syntactic query against relational table will not work!

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

Zero Matches!

Traditional Syntactic query against relational data

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;
```
For More Information

Google:
Oracle RDF

or
oracle.com

(william.beauregard@oracle.com)
Hardware and Software

Engineered to Work Together