ORACLE®

Not only SPARQL, Not just RDF/OWL:

Special Enterprise-focused Semantic Capabilities for Effective Utilization of Semantic Technologies in Oracle Database

Souri Das, Ph.D., Architect, Oracle
Seema Sundara, Consultant Member , Oracle

June 2011

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.

Outline

- Core functionality[1]
  - RDF/OWL loading, SPARQL query and DML, inference
- Enterprise functionality[1]
  - SPARQL in SQL
  - Semantic Indexing of Unstructured Content
  - Security
  - Querying complex data types in SPARQL/SQL
- Summary

[1]Some of the functionality may require installation of patch 9825019: SEMANTIC TECHNOLOGIES 11G R2 FIX BUNDLE 3

Extract, Model, Reason & Discover Workflow

Transaction Systems

Unstructured Content

RSS, email

Other Data Formats

Data Sources

Transform & Edit Tools

Entity Extraction & Transform
Ontology Eng.
Categorization
Custom Scripting

Partner and Open Source Tools

Database Management

- RDF Data Loading
- SPARQL / SQL
- Native Inferencing
- Semantic Rules
- Open source integration
- Security
- Semantic Indexing

ORACLE SPATIAL

Applications & Analysis Tools

BI, Analytics
Graph Visualization
Social Network Analysis

Metadata Registry

Faceted Search

Partner and Open Source Tools

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

Core Entities in Oracle Database Semantic Store

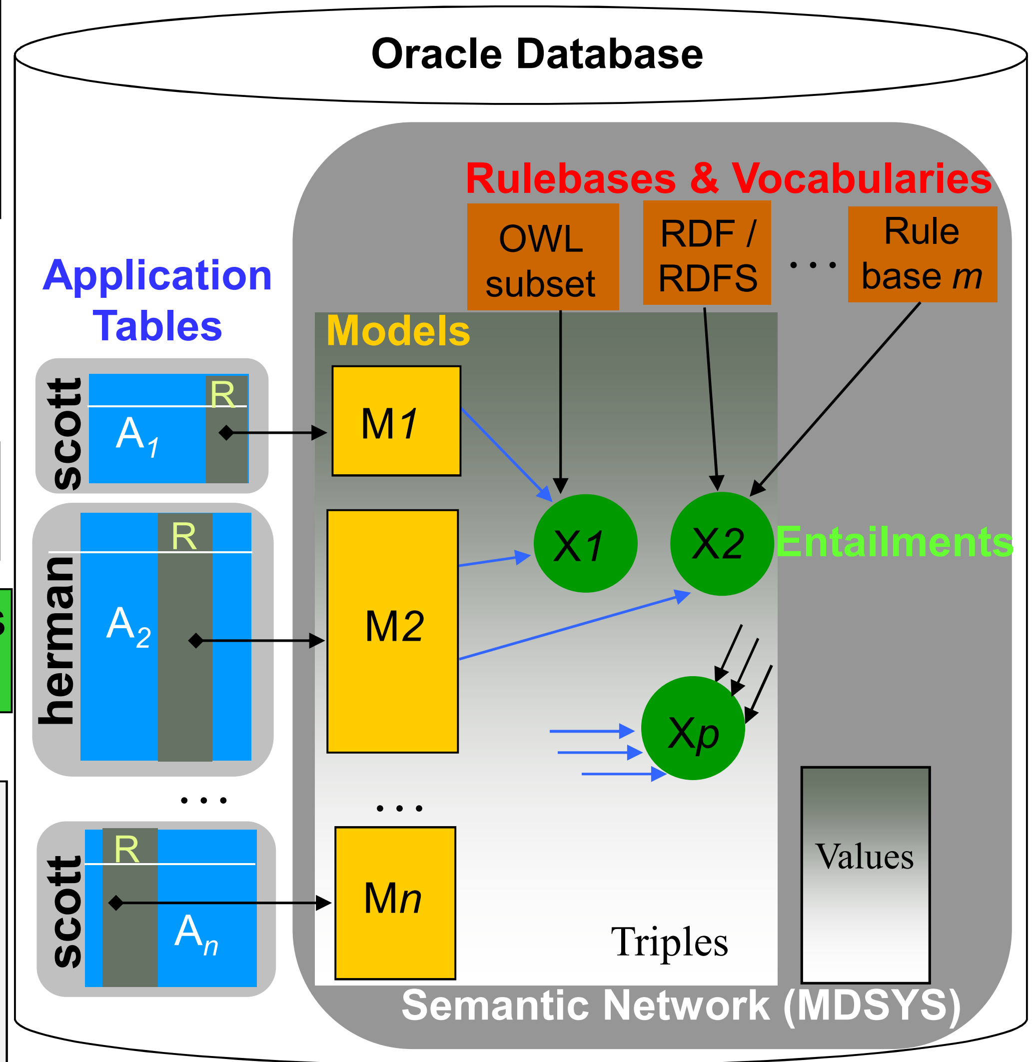
- **Sem. Network ➔** Dictionary and data tables for storage and management of asserted and inferred RDF triples. OWL and RDFS rule bases are preloaded.

- **Model ➔** A model holds RDF graphs (each a 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.

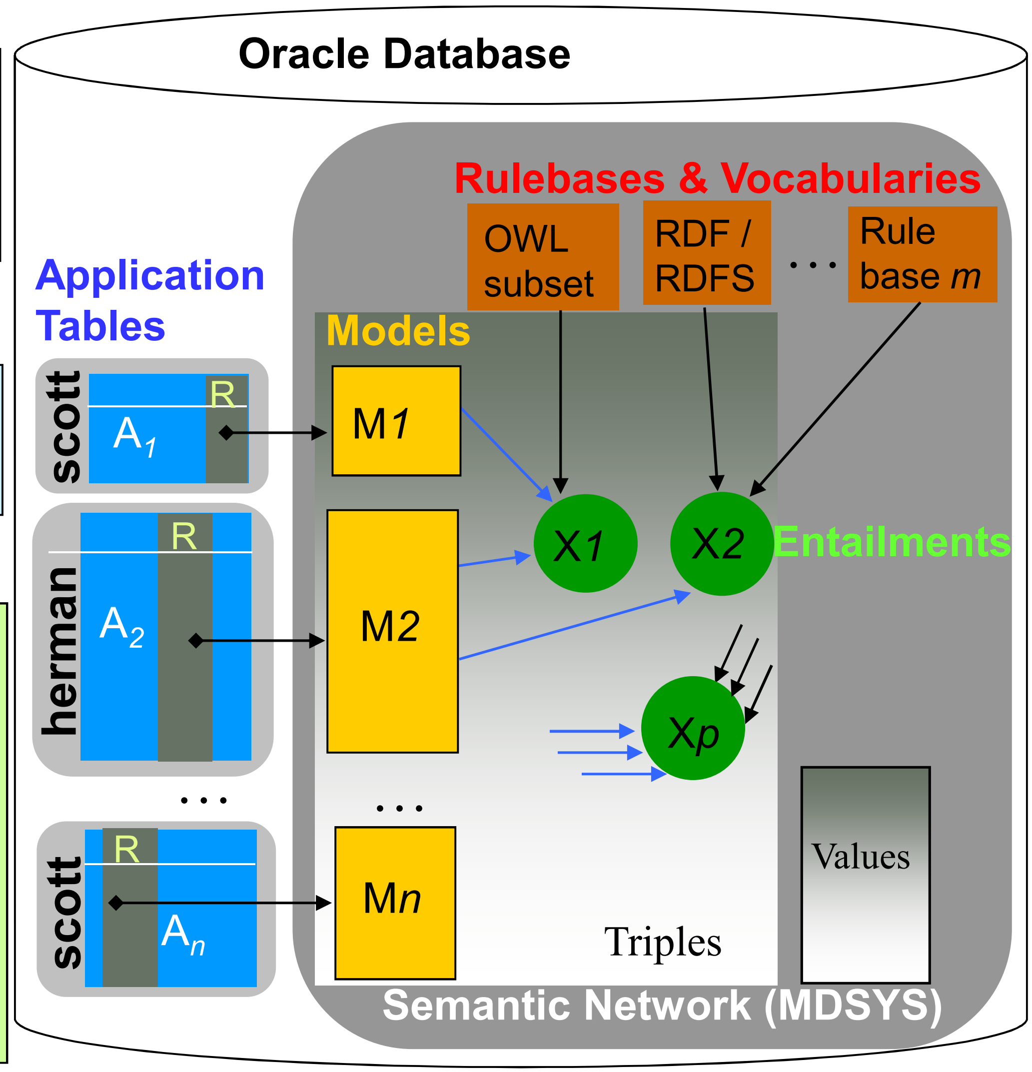


Core Functionality: Load / Query / Inference

- **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, SKOSCORE, etc.
  - Named Graph (Local/Global) Inference
  - User-defined rules



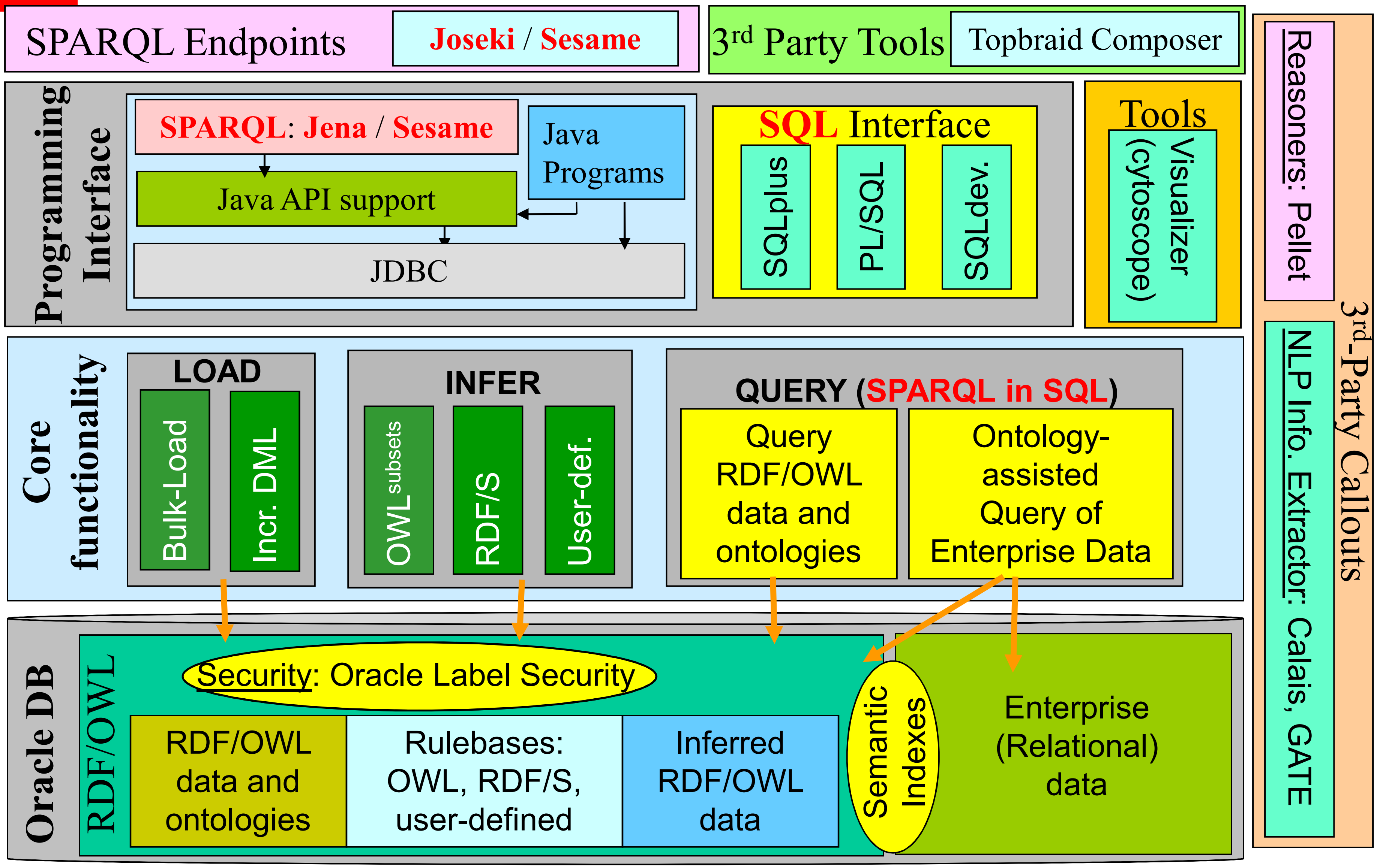
Enterprise Functionality: SQL / Sem. Indexing / Security

- SPARQL (embedding) in SQL
  - Allows joining SPARQL results with relational data
  - Allows use of rich SQL operators (such as aggregates)
- Semantic indexing
  - Index consists of RDF triples extracted from documents stored (directly or indirectly) in a table column
  - Extraction done by one or more 3rd party information extractors
- Security: Fine-Grained Access Control (for each triple)
  - Uses Oracle Label Security (OLS)
  - Each RDF triple has an associated sensitivity label
- Querying Text and Spatial data using SPARQL

Interfaces

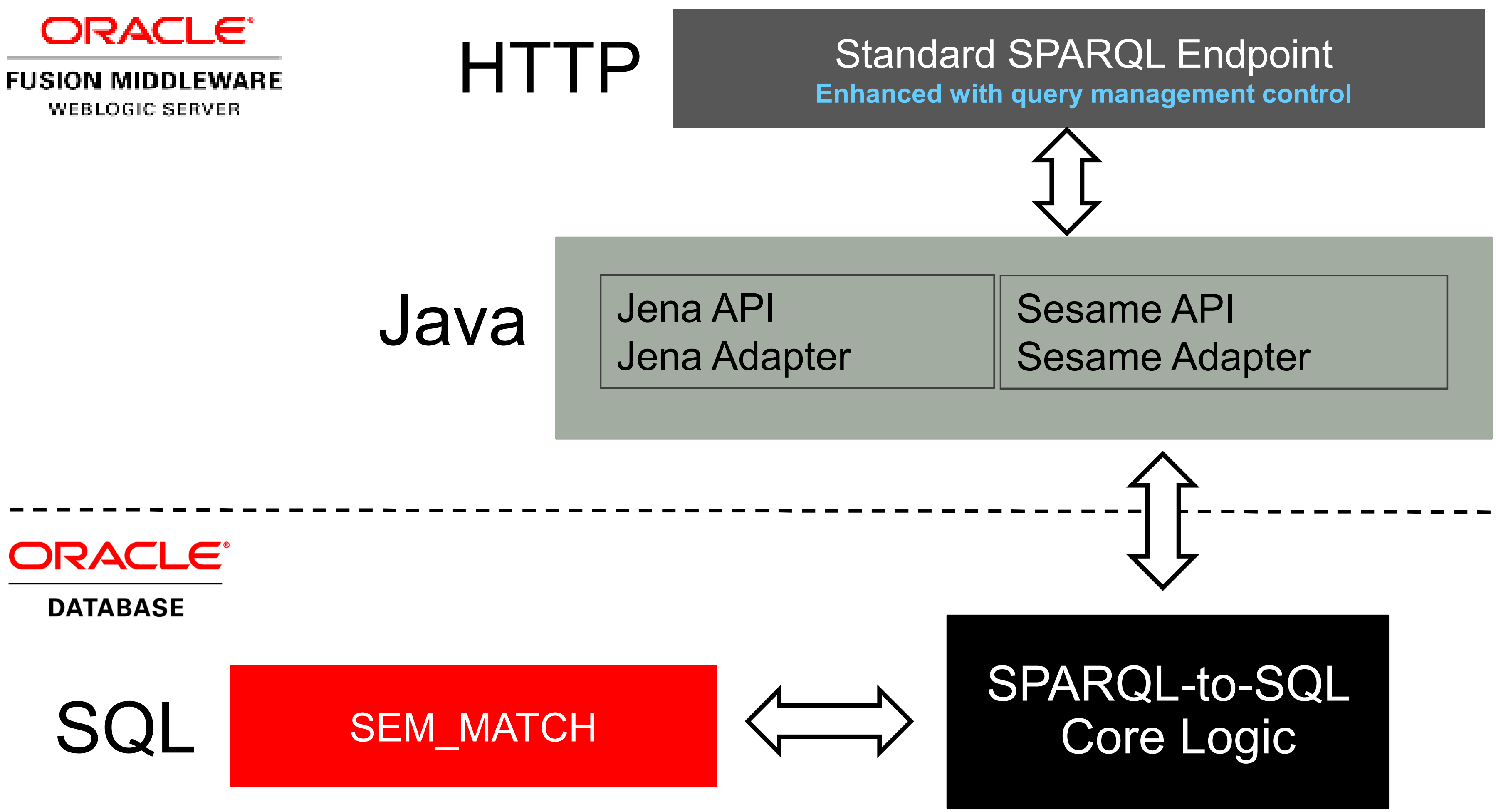
- SQL-based (SQL and PL/SQL)
- Java-based
  - Jena (using Jena Adapter from Oracle)
  - Sesame (using Sesame Adapter from Oracle)
- SPARQL Endpoints
  - Joseki
  - OpenRDF Workbench

Architectural Overview



SPARQL in SQL

SPARQL and "SPARQL in SQL" Query Architecture



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

SEM_MATCH: Adding SPARQL to SQL

```
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

```
SELECT n1, n2
FROM
 TABLE(
```

n1	n2
Alex	Jerry
Alex	Tom
Alice	Bill
Alice	Jill
Alice	John

SEM_MATCH: Adding SPARQL to SQL

Rewritable **SQL Table Function**

```
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 a single declarative SQL query

- Query optimizer sees a single query
- Get all the performance of Oracle SQL Engine
  - compression, indexes, parallelism, statistics, etc.

Query Optimizer Hints

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

```
select /*+ parallel(4) */ 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 '));
```

Goal: start with VALUE$ index and drive the query from there using nested loop join

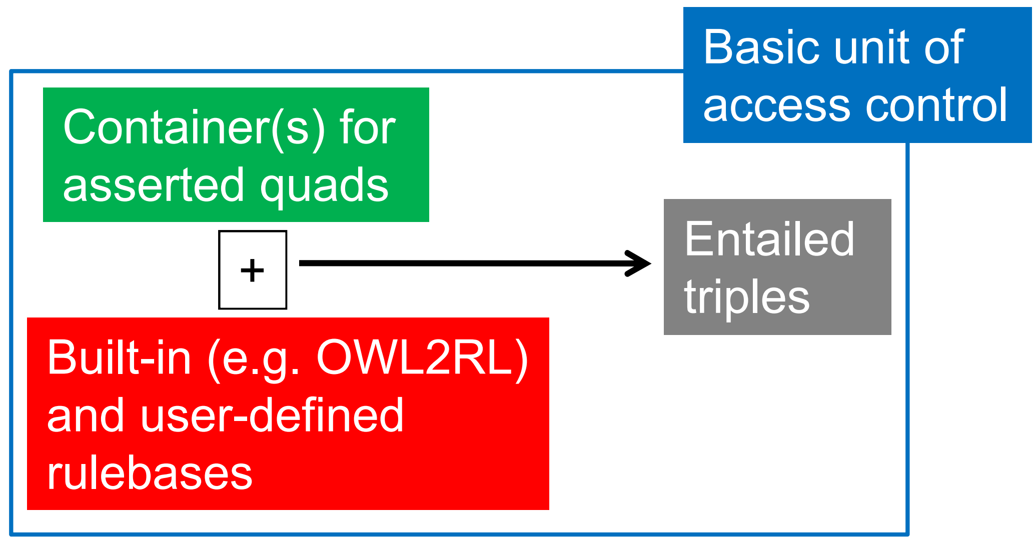
Can Influence:

- **Join Order**
- **Join Type**
- **Access Path**

SEM_MATCH Table Function Arguments

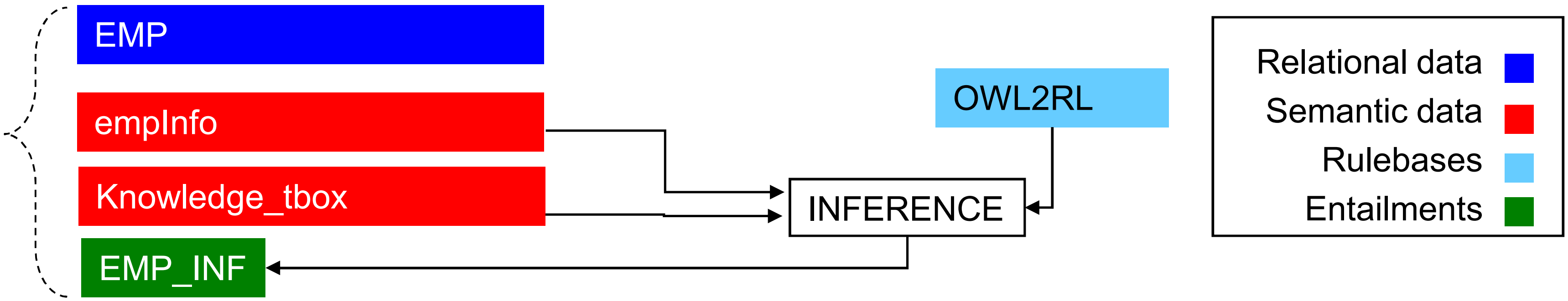
```
SEM_MATCH(
   query,
   models,
   rulebases,
   options
);
```

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




```
'ALLOW_DUP=T STRICT_TERM_COMP=F'
```

SPARQL in SQL: against Relational + RDF data



EMP table

EMPNO	ENAME	JOB	MGR
7566	JONES	MANAGER	7839
7698	BLAKE	MANAGER	7839
7782	CLARK	MANAGER	7839

Knowledge_Tbox

<understands_beginner>
rdfs:subPropertyOf <understands> .
<understands_expert>
rdfs:subPropertyOf <understands> .

EMP_INF

<empno/7566>
<understands> <SemanticTech> .
<empno/7782>
<understands> <SemanticTech> .

empInfo

<empno/7566>
<empno> "7566"^^xsd:integer ;
<understands_beginner> <SemanticTech> .

<empno/7698>
<empno> "7698"^^xsd:integer .

<empno/7782>
<empno> "7782"^^xsd:integer ;
<understands_expert> <SemanticTech> ;
<presentedAt> <SemTechConf2011> .

<SemTechConf2011>
<focus> <SemanticTech> .

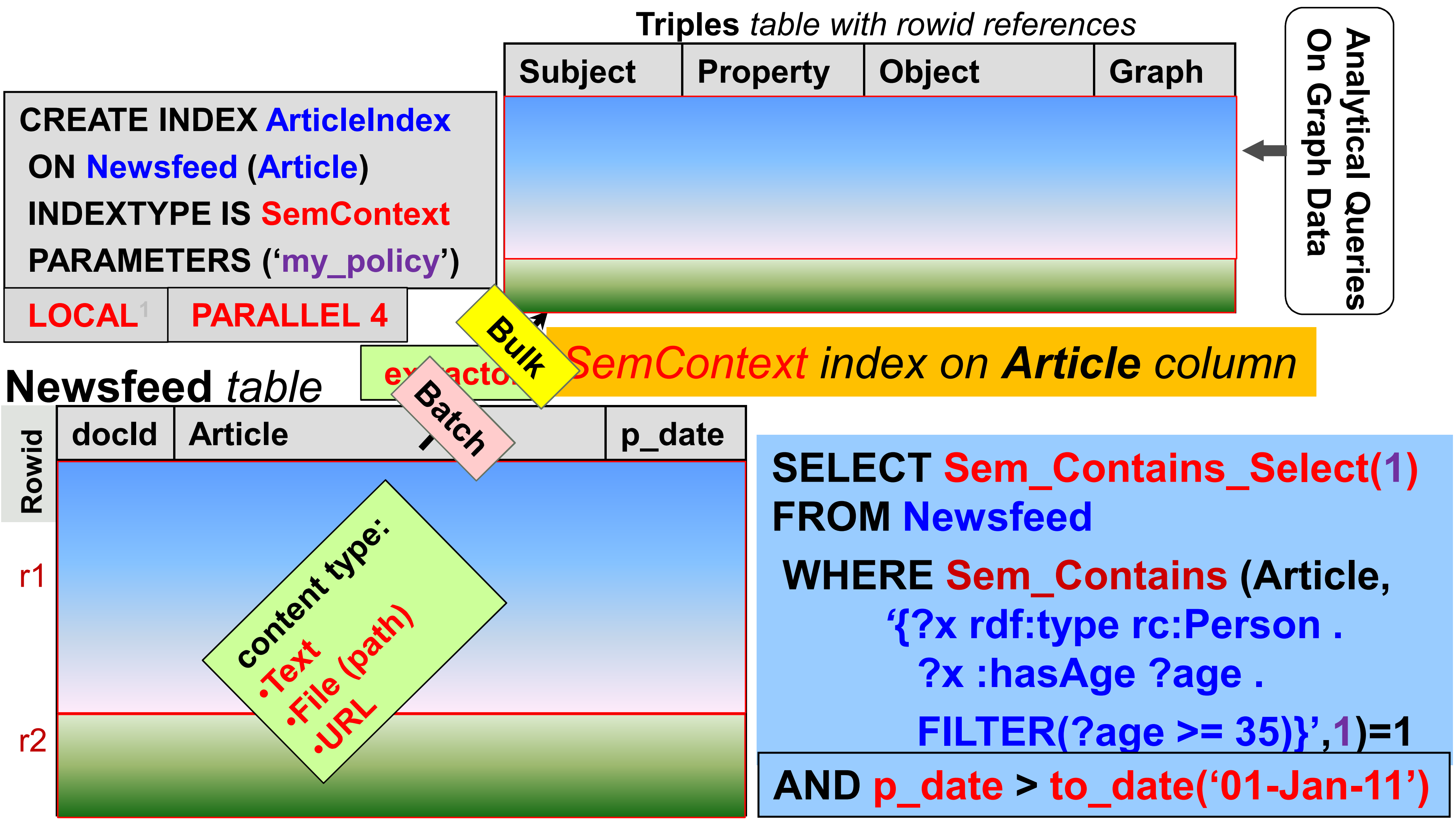
Example: SPARQL in SQL: against Relational + RDF

Find employees who understand Semantic Tech.

```
select ename, mgr, forum, paper
from emp,
     table(sem_match(
'SELECT ?eno ?forum ?paper
 WHERE
 { ?s <empno> ?eno .
   ?s <understands> <SemanticTech> .
   OPTIONAL {?s <presentedAt> ?forum .
             ?forum  <focus>  <SemanticTech>}
   OPTIONAL {?s <published> ?paper .
             ?paper  <hasKeyword> "Semantics"}
 }'
, sem_models('empInfo','knowledge_tbox')
, sem_rulebases('OWL2RL'), null, null, null
))
where eno=empno;
```

Semantic Indexing for Unstructured Content

Overview: Creating and Using a Semantic Index



[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

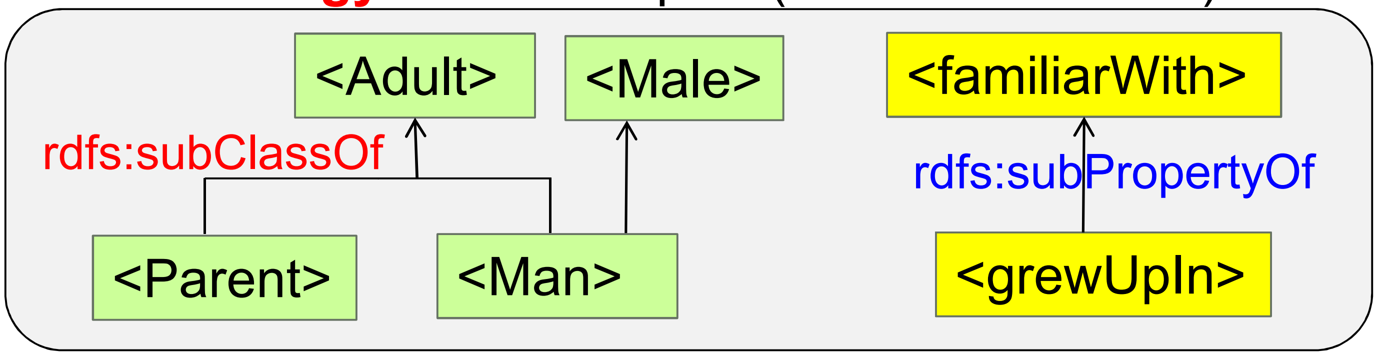
- Policy
  - Base Policy: <policy_name, extractor_type>
  - Dependent Policy: <policy_name, base_policy_name, knowledge_bases, entailments>
- Association between indexes and policies
  - Multiple policies may be associated with an index
  - Triples extracted using each base policy is stored separately
- 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

Base table

id	document
1	John is a parent. He grew up in NYC.
2	John is a man.

Ontology: schema triples (for *extracted* data)



Index (RDF model): *extracted* triples

Subject	Property	Object	Graph
<John>	rdf:type	<Parent>	<…/r1>
<John>	<grewUpIn>	<NYC>	<…/r1>
<John>	rdf:type	<Man>	<…/r2>
…	…	…	…

Entailment: set of *inferred* triples

Subject	Property	Object	Graph
<John>	rdf:type	<Adult>	<…/r1>
<John>	<familiarWith>	<NYC>	<…/r1>
<John>	rdf:type	<Adult>	<…/r2>
<John>	rdf:type	<Male>	<…/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 .
                ?comp  p:categoryName   c:BusinessFinance .
                ?comp  p:location       ?city .
                ?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.

```
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

Patients diagnosis table

ID	DIAGNOSIS
1	Hand_Fracture
2	Rheumatoid_Arthritis

Query: "Find all entries in diagnosis column that are related to 'Upper_Extremity_Fracture'"

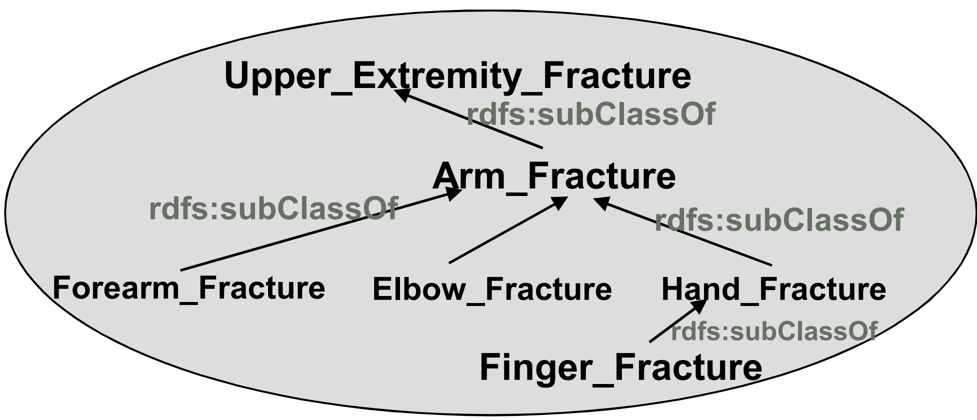
Syntactic query against relational table will not work!

```
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;
```

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 access control is possible through Oracle Label Security (OLS)
  - A thin-layer of RDF-specific capabilities on top of OLS
  - Triple level security
  - Mandatory Access Control

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 posses in order to read or write the data.

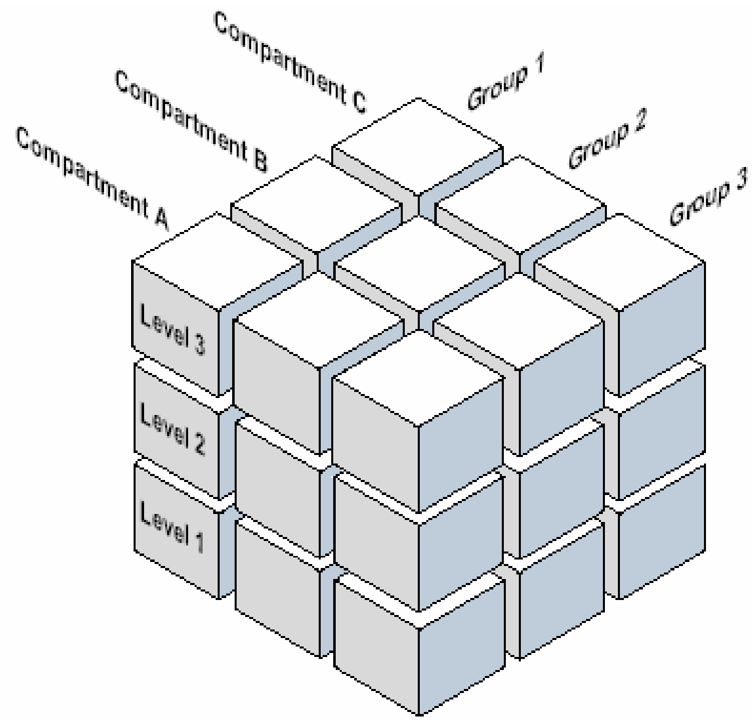
ContractID	Organization	ContractValue	Label
ProjectHLS	N. America	1000000	SE:HLS:US

  - 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:

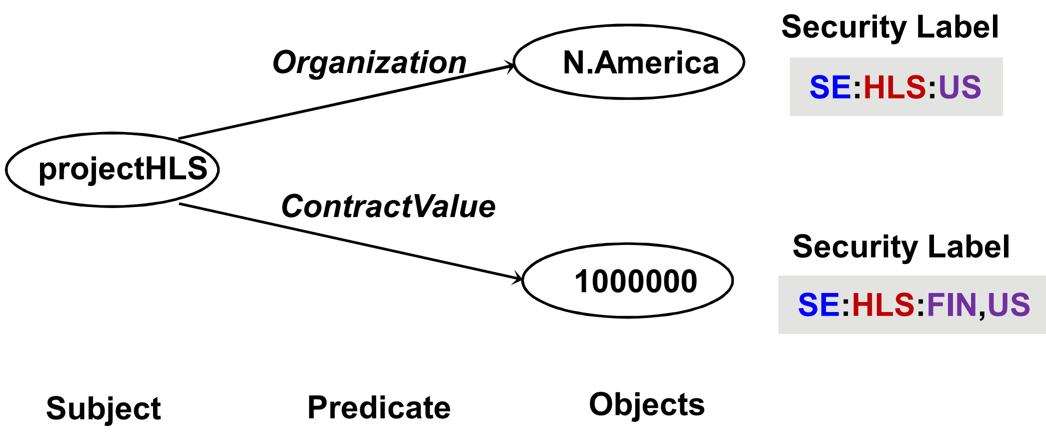
- **Levels** – Determine the vertical sensitivity of data and the highest classification level a user can access.
- **Compartments** – Facilitate compartmentalization of data. Users need membership to a compartment to access its data.
- **Groups** – Allow hierarchical categorization of data. A user with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



Row Label →	`CONF`	:	`NAVY,MILITARY`	:	`NY,DC`
User Access Label →	`HIGHCONF`	:	`MILITARY,NAVY,SPCLOPS`	:	`US,UK`

***User Access Label* dominates the row label**

RDF Triple-level Security with OLS



Triples table

Subject	Predicate	Object	RowLabel
projectHLS	Organization	N.America	SE:HLS:US
projectHLS	ContractValue	1000000	SE:HLS:FIN,US

- 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 → US
- 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

- Insert triples
  - John inserts triple:

    ```
    <http://John> <rdf:type> <http://Person>
    ```

  - Mary inserts triple:

    ```
    <http://Mary> <rdf:type> <http://Person>
    ```

  - Both these triples are stored in the RDF 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

"EASTSE"
(SE:CIA,HLS:EASTUS)

"USUN"
(UN:FBI,HLS:US)

John Read	Triple Label	Mary Read
No	TS:HLS:DC	No
No	SE:HLS,FBI:DC	No
Yes	UN:HLS:DC	**Yes**
Yes	UN:HLS,CIA:NY	No
No	SE:CIA:SF	No
No	UN:HLS,FBI:NY	**Yes**
No	UN:HLS:SF	**Yes**

Securing RDF Data using OLS: Example (5)

- Same triple may exist with different labels:

```
<http://John> <rdf:type> <http://Person> 'UN:HLS:DC'
<http://John> <rdf:type> <http://Person> 'SE:HLS:DC'
```

- When Mary queries, only one 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 strictly dominated by SE:

```
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

Oracle Extensions for Text and Spatial

Datatype Indexes for FILTERs

```
FILTER (?lname >= "Pa" &&
        ?lname < "Pb")
```

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> exec sem_apis.add_datatype_index(
               'http://xmlns.oracle.com/rdf/text');
```

Text Filter Query

Find all bills about Children and Taxes

```
select s, title, dt
from table(sem_match(
'SELECT ?s ?title ?dt
 WHERE
 { ?b bill:sponsor ?s .
   ?s foaf:name ?n .
   ?b dc:title ?title .
   ?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 (SRS) 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_apis.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-Polyon
    - 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 Data

U.S. Census Bureau

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

Spatial Filter Query 1

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 .
   ?role pol:forOffice ?office .
   ?office pol:represents ?cdist .
   ?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 '
));
```

Spatial Filter Query 2

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 .
   ?role pol:forOffice ?office .
   ?office pol:represents ?cdist .
   ?cdist orageo:hasWKTGeometry ?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 '));
```

Summary: Sem. Tech. in Oracle DB: beyond core func.

Support for semantic technologies in Oracle 11g extends beyond the core of

- W3C standards compliant RDF/OWL load, inference, SPARQL query

to include the following enterprise functionality

- **SPARQL in SQL** allows efficient retrieval of information from mixed store consisting of relational tables and RDF graphs utilizing the rich table-friendly constructs of SQL and the graph-friendly constructs of SPARQL
- **Semantic Indexing** to allow efficient semantic search of documents based on their semantic content extracted by 3rd party NLP information extractors
- **Security at RDF triple-level** to allow mandatory access control of semantic data
- **Efficient querying of complex data types** using SPARQL (or SPARQL in SQL) leveraging native indexing supported in the Oracle database such as Oracle Spatial (R-tree) index and Oracle Text index

For More Information

Oracle RDF

or

oracle.com

Hardware and Software

ORACLE

Engineered to Work Together

ORACLE®