ORACLE

Oracle Database Semantic Technologies Tutorial

Oracle Semantic Technologies Agenda

- Introduction
- Tutorial

Oracle Database Semantic Data Store

- A feature of Oracle Spatial 11g Option for Oracle Database 11g Enterprise Edition
  - Requires Partitioning and Advanced Compression Options
- An open and persisted RDF data model and analysis platform for semantic applications
- An RDF Data Model with inferencing (RDFS, OWL and user-defined rules)
- Performs SQL-based access to triples and inferred data
- Combines SQL query of relational data with RDF graphs and ontologies
- Supports large graphs (billion+ triples)
- Easily extensible by 3rd party tools/apps

Only Oracle Database 11g…

Has an open, persistent, analytic semantic data management platform

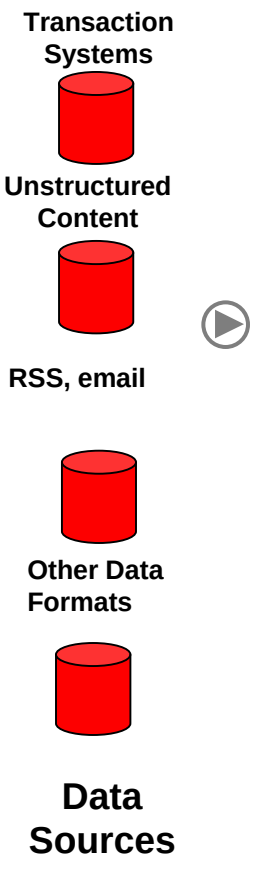
- Scalability – Trillions of triples
- Availability – tens of thousands of users
- Security – protect sensitive business data
- Performance – timely load, query & inference
- Accessibility – to enterprise applications
- Manageability – leverage IT resources

= Oracle Database Strengths

Oracle Customer Examples

- Enterprise Information Integration
  - Hutchinson 3G Austria
- Large Public Dataset for Data Integration
  - Uniprot dataset at the Swiss Institute of Bioinformatics
- Data Integration
  - Yale University
  - Stanford University
  - University of Cincinnati
- Bio-surveillance
  - University of Texas at Houston
- Re-use of Legacy Data
  - Pharmaceutical companies

Semantic Data Management Workflow



Edit & Transform

- Entity Extraction & Transform
- Ontology Engineering
- Categorization
- Custom Scripting

Load, Query & Inference

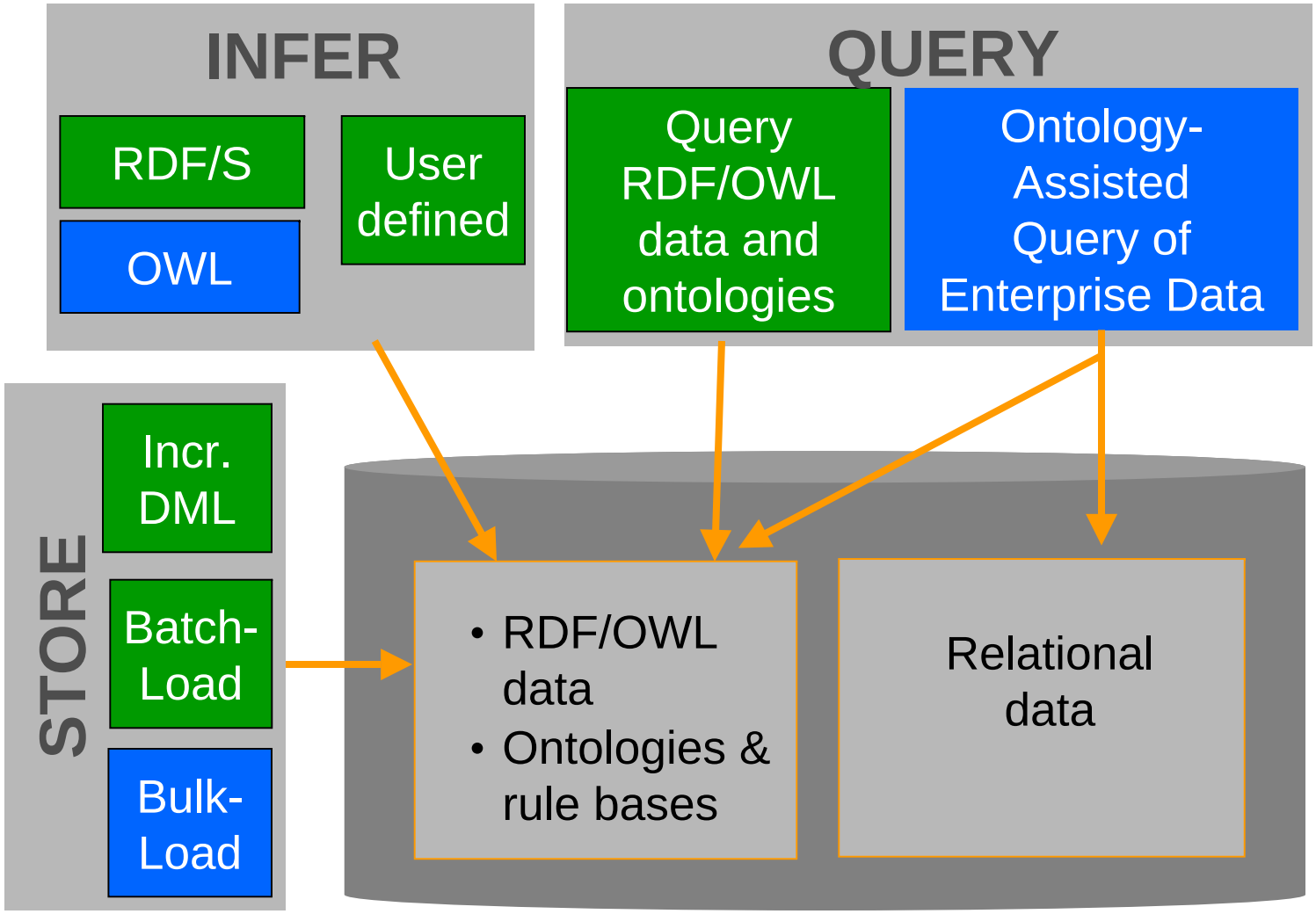
- RDF/OWL Data Management
- SQL & SPARQL Query
- Inferencing
- Semantic Rules
- Scalability & Security

ORACLE SPATIAL

Applications & Analysis

- Graph Visualization
- Link Analysis
- Statistical Analysis
- Faceted Search
- Pattern Discovery
- Text Mining

Oracle Spatial 11g Semantic Capabilities



Semantic Data Management Tasks

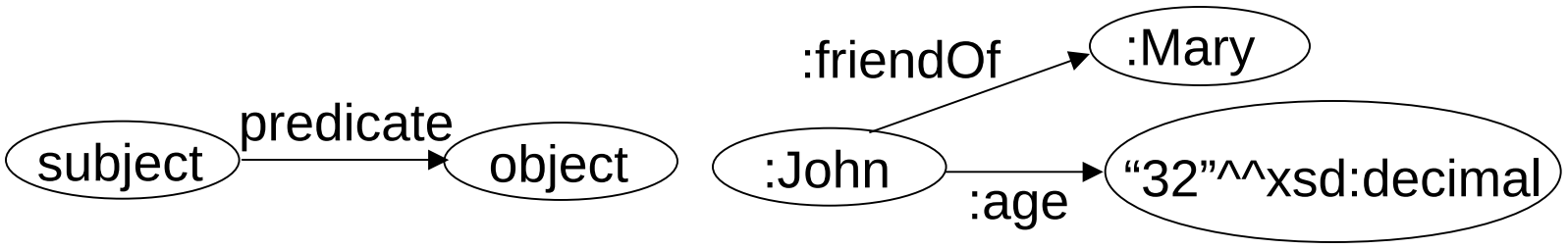
- Create a new Semantic Network
- Create a RDF/OWL model
- Load RDF/OWL data
- Optionally create user-defined rulebases
- Infer new RDF/OWL data
- Query RDF/OWL data
- Perform Ontology-assisted Query

Tutorial Agenda

- RDF Concepts
- Using Oracle Database 11g as a Semantic Data Store
- Jena Adaptor
- System Metadata and Views
- National Cancer Institute Ontology Case Study
- Best Practices

RDF Concepts

- Originally created to encode metadata such as ‘author’, ‘date’, etc. for web resources.
- It has become popular to relate things in the real-world such as people, places, concepts, schemas
- The basic unit of information (fact) is represented as <subject, property, object> triple
- Triples together form a graph, connecting pieces of data



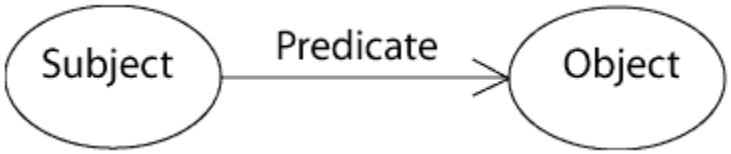
Triples

- Each triple represents a statement of a relationship between the things denoted by the nodes that it links. Each triple has three parts:

1. A subject,
2. An object, and
3. A predicate (also called a property) that denotes a relationship.

The direction of the arc is significant: it always points toward the object.

The nodes of an RDF graph are its subjects and objects.



URIs

A Uniform Resource Identifier (URI), is a compact string of characters used to identify or name a resource. The main purpose of this identification is to enable interaction with representations of the resource over a network, typically the World Wide Web, using specific protocols. URIs are defined in schemes defining a specific syntax and associated protocols.

RDF Triples

- RDF triple represents a *directed labeled* edge

<http://purl.uniprot.org/uniprot/Q4U9M9>

<http://www.w3.org/1999/02/22-rdf-syntax-ns#type>

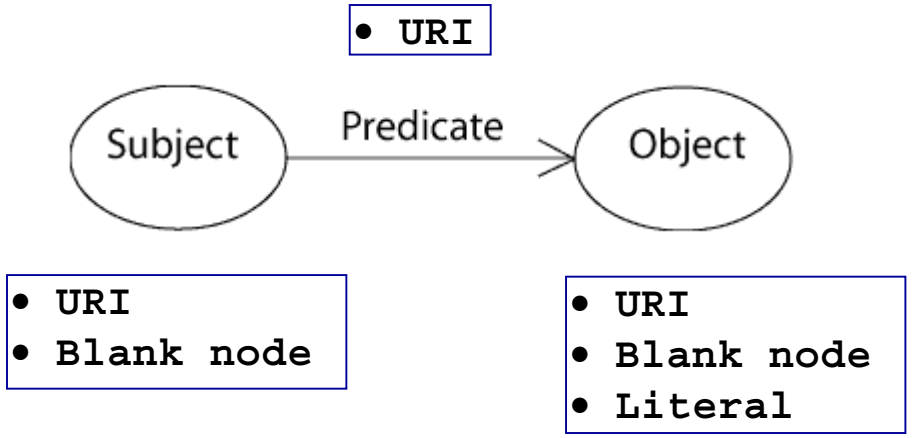
<http://purl.uniprot.org/core/Protein>

- RDF triple *set* represents a directed labeled graph

```
# uniprot:... rdf:type :Protein
<http://purl.uniprot.org/uniprot/Q4U9M9> <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <http://purl.uniprot.org/core/Protein> .
<http://purl.uniprot.org/uniprot/P15711> <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <http://purl.uniprot.org/core/Protein> .
<http://purl.uniprot.org/uniprot/Q43495> <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <http://purl.uniprot.org/core/Protein> .
<http://purl.uniprot.org/uniprot/P18646> <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <http://purl.uniprot.org/core/Protein> .
<http://purl.uniprot.org/uniprot/P13813> <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <http://purl.uniprot.org/core/Protein> .
# other attributes
<http://purl.uniprot.org/uniprot/Q4U9M9> <http://www.w3.org/2000/01/rdf-schema#seeAlso> <http://purl.uniprot.org/interpro/IPR007480> .
```

RDF Data Model

- RDF terms (values) can be of three types
  - URIs ➔ <…>
  - Blank Nodes ➔ _:x
  - Literals ➔ “John”, “2004-12-21T11:22:33-05:00”^^xsd:dateTime, “0010”^^xsd:integer
- Positional constraints
  - Predicate ➔ must be a URI
  - Subject ➔ must be a URI or a blank node
  - Object ➔ must be a URI, blank node, or literal
- Set property
- Value-point based equivalence



Ontology

- A systematic arrangement of all of the important categories of objects or concepts which exist in some field of discourse, showing the relations between them
  - The Gene Ontology project provides a controlled vocabulary to describe gene and gene product attributes in any organism
  - The Protein Ontology (PO) provides a unified vocabulary for capturing declarative knowledge about protein domain and to classify that knowledge to allow reasoning

Inferencing: Rules and Rulebases

- **Inferencing** is the ability to make logical deductions based on **rules**
- A **rule** is an object that can be applied to draw inferences from semantic data.
- A rule consists of:
  - An IF side pattern for the **antecedents**
  - An optional filter condition that further restricts the subgraphs
  - A THEN side pattern for the **consequents**
- For example, the rule that a chairperson of a conference is also a reviewer of the conference could be represented as follows:

```
('chairpersonRule', -- rule name
 '(?r :ChairPersonOf ?c)', -- IF side pattern
 NULL,  -- filter condition
 '(?r :ReviewerOf ?c)', -- THEN side pattern
 )
```

Rules Index

- Contains precomputed triples
- Inferred from model(s) and rulebase(s)
- Mandatory when using SEM_MATCH (for each rulebase-model combination)
- Use SEM_APIS.CREATE_RULES_INDEX to create a rules index
- View under MDSYS is created (SEMI_rules-index-name)

```
BEGIN
  SEM_APIS.CREATE_RULES_INDEX(
    'rdfs_rix_family',
    SEM_Models('family'),
    SEM_Rulebases('RDFS','family_rb'));
END;
/
```

Tutorial Agenda

- RDF Concepts
- Using Oracle Database 11g as a Semantic Data Store
- Jena Adaptor
- System Metadata and Views
- National Cancer Institute Ontology Case Study
- Best Practices

Introducing the Family Tree Use Case

- **John and Janice** have two children:
  Suzie and Matt.

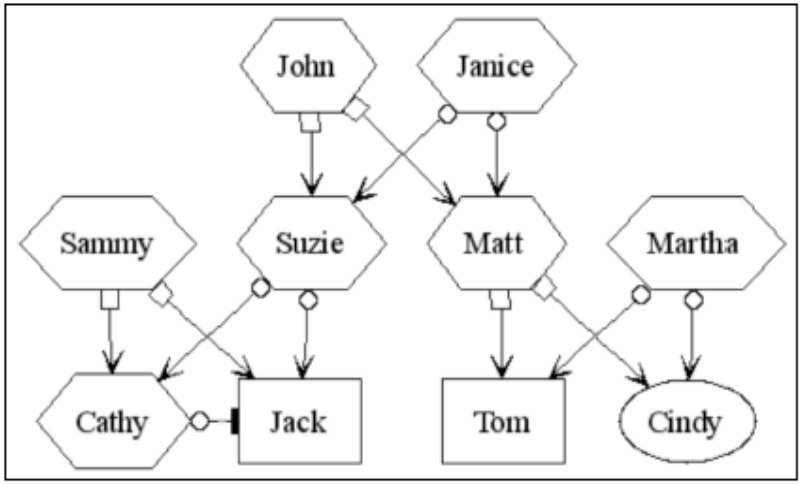
- **Matt married Martha**, and they have two children:

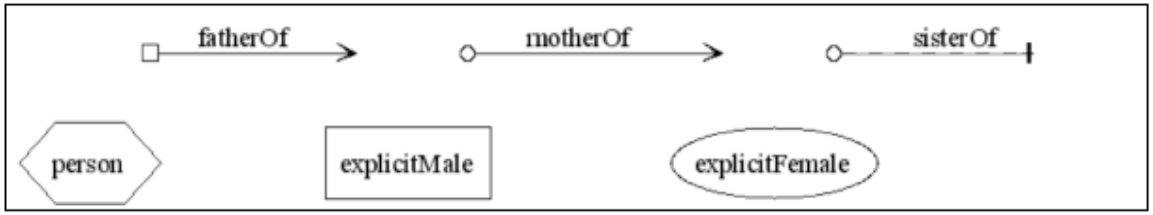
  Tom (male, height 5.75) and
  Cindy (female, height 06.00)

- **Suzie married Sammy**,

they have two children:

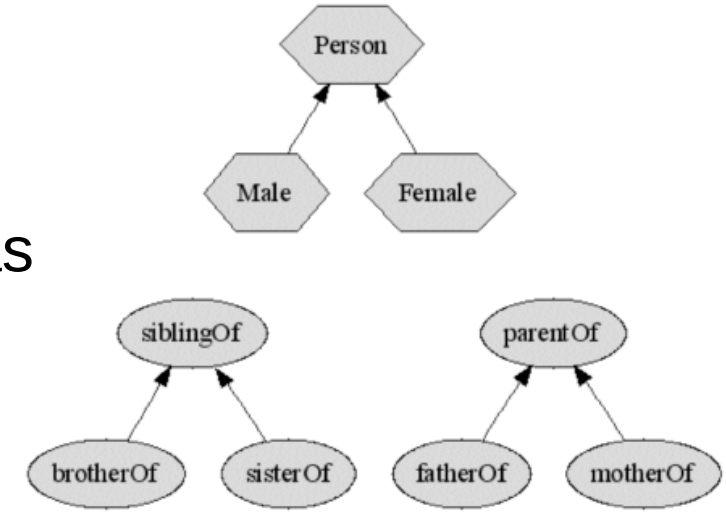
  Cathy (height 5.8) and
  Jack (male, height 6).





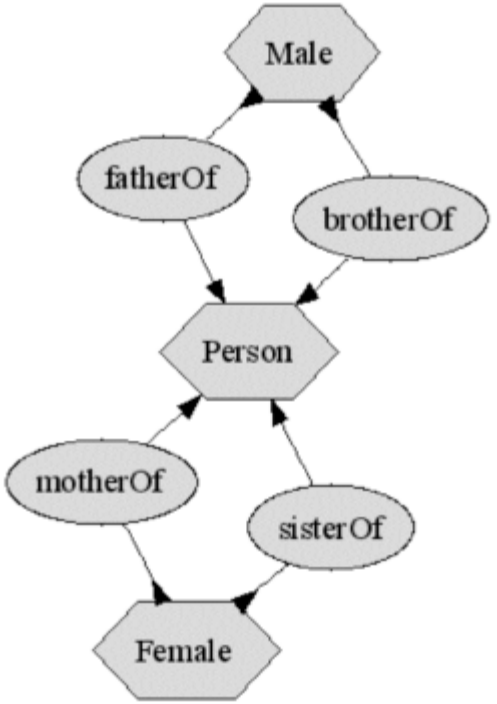
Family Tree Classes and Properties

- **Person** is a class that has two subclasses:
  Male and **Female**
- **siblingOf** is a property that has two subproperties:
  brotherOf and **sisterOf**
- **parentOf** is a property that has two subproperties:
  fatherOf and **motherOf**



Family Tree Domains

- The domain of the **fatherOf** and **brotherOf** properties is **Male**
- The domain of the **motherOf** and **sisterOf** properties is **Female**



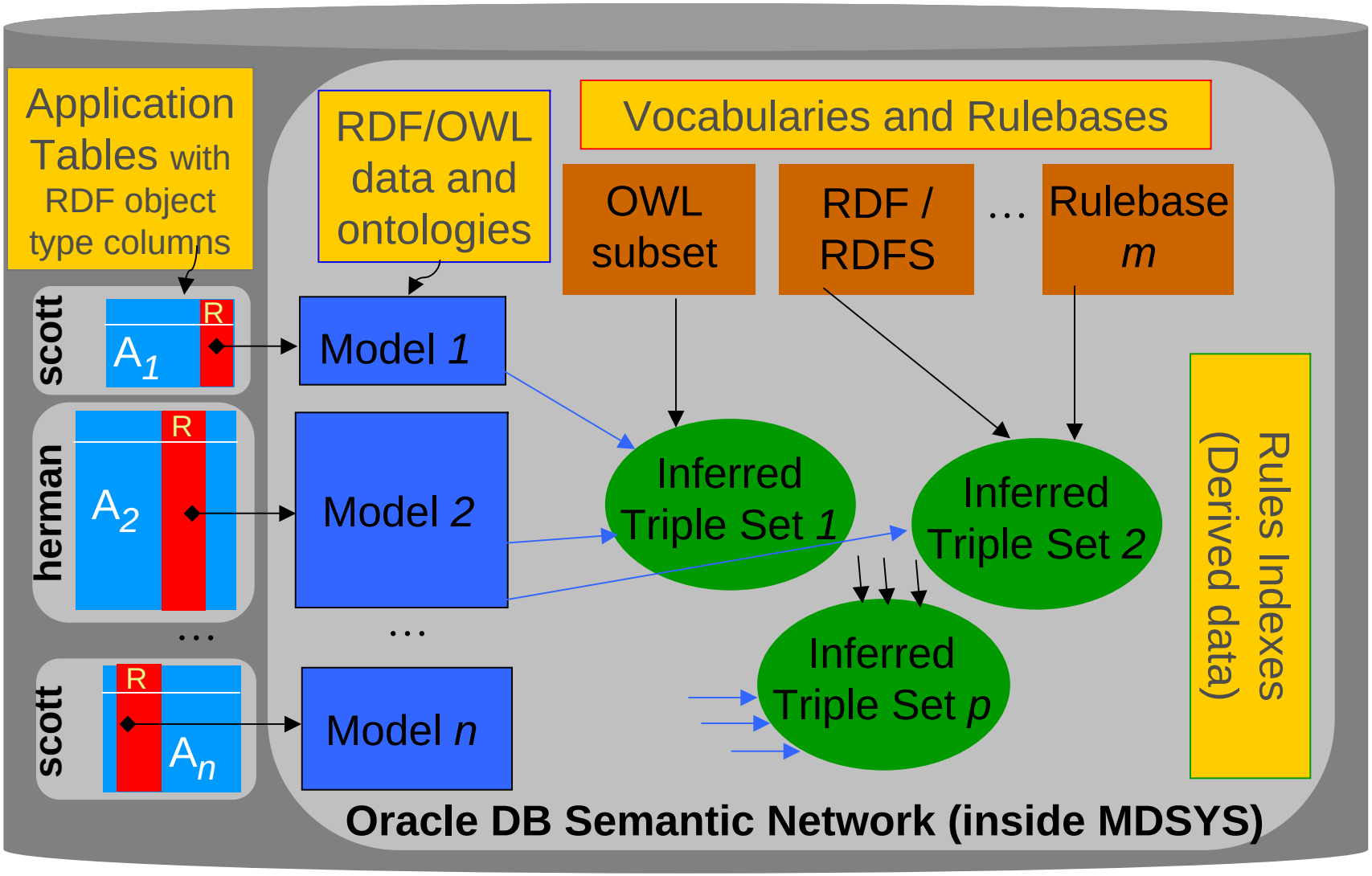
Using Oracle Database 11g as a Semantic Data Store

- Create a Semantic Network
- Create an RDF/OWL model
- Load RDF/OWL data
- Infer RDF/OWL data
- Query RDF/OWL data and ontologies
- Perform Ontology-assisted Query of relational data

Creating a Semantic Network and Model

- Creating a Semantic Network
  - Generates all RDF% tables and views under user MDSYS
- Creating a RDF/OWL model
  - Create a table with a column of type SDO_RDF_TRIPLE_S
  - Optionally choose a tablespace for the model

Entities in a Semantic Network



Security & Access Control

- Models
  - A database view owned by MDSYS created at model creation
  - Creator gets SELECT privilege with GRANT option
  - DML on a model is done via DML on the associated RDF object type column and requires invoker to have appropriate privileges on the associated application table
- Rulebases
  - Database view owned by MDSYS created at rulebase creation
  - Creator gets SELECT and DML privilege with GRANT option
- Rules Indexes (Inferred Triple Sets)
  - Database view owned by MDSYS created at rules index creation
  - Creator must have SELECT privilege on underlying model and rulebase views
  - Creator gets SELECT privilege with GRANT option

Create a Semantic Network

- Connected as sys as sysdba
  - Install the Oracle Database 11.1.0.6 patch set
    - http://www.oracle.com/technology/software/tech/semantic_technologies/index.html
  - Run ?/md/admin/catsem11i.sql
  - See MetaLink Note 472876.1 What Are The Tasks For a DBA To Setup Semantic Models In The Database
- Create a tablespace for the system tables

```
CREATE TABLESPACE rdf_tblspace DATAFILE
  '/oradata/orcl/rdf_tblspace.dat' SIZE 1024M
  AUTOEXTEND ON NEXT 256M MAXSIZE 16384M;
```

- Create a semantic data network

```
EXECUTE SEM_APIS.CREATE_SEM_NETWORK('rdf_tblspace');
```

```
* This generates all RDF% tables and views under user
  MDSYS
```

Create an RDF Model of Family Relationships

- Create a table to store references to the semantic data, include a column of type SDO_RDF_TRIPLE_S

```
CREATE TABLE family_rdf_data
(id NUMBER, triple SDO_RDF_TRIPLE_S);
```

- Create the model on the triple column

```
execute SEM_APIS.create_sem_model
('family', 'family_rdf_data', 'triple');
```

Model Info (RDF_MODEL$/RDF_LINK$)

- **Under MDSYS:**

sql tracing

```
INSERT INTO mdsys.rdf_model$
  (owner, model_id, model_name, table_name, column_name)
VALUES
  (:owner, :m_id2, :model, :rdf_table, :rdf_column)
```

```
CREATE VIEW RDFM_FAMILY
  as select * from rdf_link$ partition (model_1)

CREATE VIEW SEMM_FAMILY
  as select * from rdf_link$ partition (model_1)
```

Based on RDF_LINK$

```
select * from mdsys.rdf_model$;

OWNER      MODEL_ID MODEL_NAME    TABLE_NAME         COLUMN_NAME
------- ---------- ------------ ----------------- ------------
RDF             1 FAMILY       FAMILY_RDF_DATA   TRIPLE
```

Using Oracle Database 11g as a Semantic Data Store

- Create a Semantic Network
- Create an RDF/OWL model
- Load RDF/OWL data
- Infer RDF/OWL data
- Query RDF/OWL data and ontologies
- Perform Ontology-assisted Query of relational data

Loading RDF/OWL Data

- Oracle Database is a scalable native graph data store
  - Oracle Database 11g stores up to 8 exabytes
- Semantic data stored optimally in relational tables
- Load Options: Bulk, Batch, and DML INSERT
- Single management environment for all your data

Alternatives for Loading RDF/OWL Data

- Bulk-load
  - Fastest, but skips triples with literals > 4k bytes
  - Load data into a staging table (using SQL*Loader from a file or Named Pipe containing N-Triple formatted data)
  - Invoke PL/SQL API to do bulk load from the staging table
- Batch-load
  - Fast and can handle long literals
  - Invoke Java-based API to load from file containing N-Triple formatted data
- DML INSERT on RDF/OWL data
  - INSERT on app table (for loading small amounts of data)
  - UPDATE of app table's SDO_RDF_TRIPLE_S col
  - DELETE of row(s) from app table

Bulk Loading Using SQL*Loader

- Use SQL*Loader to load staging table
  - Control file template is available in 11g companion CD
    - Only the Staging Table name may need to be changed
  - Staging Table definition is shown in documentation
    - Use COMPRESS (to reduce storage cost)
  - Input file must be N-Triple formatted
    - *Named Pipe* may be used to save disk space
- SEM_APIS.BULK_LOAD_FROM_STAGING_TABLE
  - Model_owner
  - Table_owner
  - Table_name
  - Flags (default NULL)

Bulk Loading Using Sql*loader

```
create table staging (
                    RDF$STC_sub varchar2(4000) not null,
                    RDF$STC_pred varchar2(4000) not null,
                    RDF$STC_obj varchar2(4000) not null,
                    RDF$STC_sub_ext varchar2(64),
                    RDF$STC_pred_ext varchar2(64),
                    RDF$STC_obj_ext varchar2(64),
                    RDF$STC_canon_ext varchar2(64))
tablespace uniprot_staging
storage (initial 64M next 64M);
```

- You must grant SELECT privilege and UPDATE privilege on the last four required columns (with names ending with _ext) in the staging table to user MDSYS. You must also grant INSERT privilege on the application table to user MDSYS .

```
grant SELECT,
UPDATE(RDF$STC_sub_ext,RDF$STC_pred_ext,RDF$STC_obj_ext,RDF$STC_canon_ext)
on staging to MDSYS;
```

```
$ORACLE_HOME/bin/sqlldr userid=rdf \
control=$ORACLE_HOME/md/demo/network/rdf_demos/bulkload.ctl data=uniprot_100m.nt \
direct=true skip=0 discardmax=1000000 bad=uniprot.bad discard=uniprot.rej   \
log=uniprot.log errors=1000000

Load completed - logical record count 100000000.
```

API: Bulk_load_from_staging_table

```
create table uniprot_rdf_data(id number, triple sdo_rdf_triple_s)
tablespace uniprot;

exec sem_apis.create_sem_model('uniprot', 'uniprot_rdf_data', 'triple');
```

```
exec sem_apis.bulk_load_from_staging_table('uniprot', 'rdf', 'staging');
*
ERROR at line 1:
ORA-55302: insufficient privileges FAILED: INSERT INTO
  "RDF"."UNIPROT_RDF_DATA"
("TRIPLE")
==> SQLERRM=ORA-01031: insufficient privileges
```

```
grant insert on uniprot_rdf_data to mdsys;

exec sem_apis.bulk_load_from_staging_table('uniprot', 'rdf', 'staging');
select count(*) from uniprot_rdf_data;
 COUNT(*)
----------
  99999420
```

Batch Loading Using the Java API

- Batch-load uses the oracle.spatial.rdf.client.BatchLoader class packaged in <ORACLE_HOME>/md/jlib/sdordf.jar
- Example (on Linux)
  - java
    -Ddb.user=scott -Ddb.password=password
    -Ddb.host=127.0.0.1 -Ddb.port=1522 -Ddb.sid=orcl
    -classpath ${ORACLE_HOME}/md/jlib/sdordf.jar:${ORACLE_HOME}/jdbc/lib/ojdbc5.jar
    oracle.spatial.rdf.client.BatchLoader
    <N-TripleFile> <tablename> <tablespaceName>
    <modelName>

DML Inserts to Load RDF data

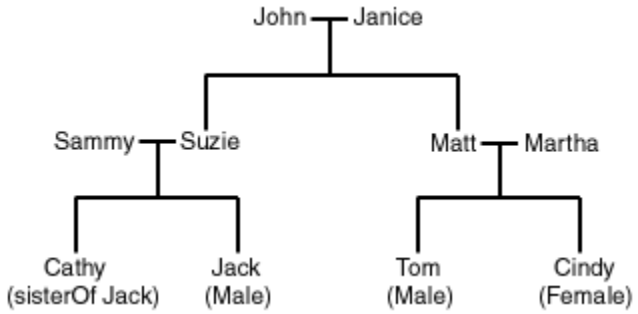
- INSERT INTO <app_table> VALUES (…, SDO_RDF_TRIPLE_S(<model>,<sub>,<pred>,<obj>) …)
- UPDATE <app_table> a SET a.<rdf_triple_col> = SDO_RDF_TRIPLE_S(<model>,<sub>,<pred>,<obj>) WHERE …
- DELETE FROM <app_table> a WHERE …
- Where clause:
  WHERE a.<rdf_triple_col>.rdf_p_id =
  (SELECT value_id FROM mdsys.RDF_VALUE$
  WHERE vname_prefix=… AND vname_suffix=…
  AND value_type=…);

Semantic Data (Insert)

- **John** is the father of **Suzie**

```
INSERT INTO family_rdf_data VALUES
(1, SDO_RDF_TRIPLE_S('family',
  'http://www.example.org/family/John',
  'http://www.example.org/family/fatherOf',
  'http://www.example.org/family/Suzie'));
```

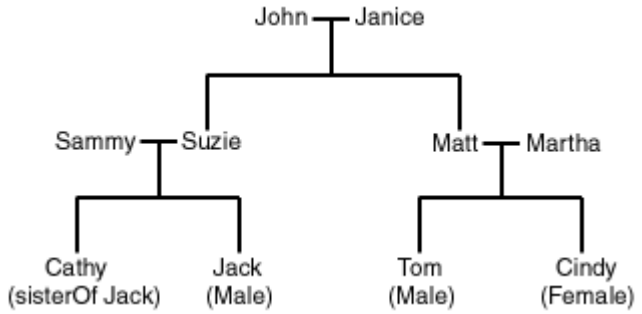
```
select value_name
from mdsys.rdf_value$

http://www.example.org/family/John
http://www.example.org/family/Suzie
http://www.example.org/family/fatherOf
```




```
select P_VALUE_ID, START_NODE_ID, CANON_END_NODE_ID, END_NODE_ID, MODEL_ID, COST
from mdsys.semm_family;

 P_VALUE_ID          START_NODE_ID      CANON_END_NODE_ID
--------------------- --------------------- --------------------
          END_NODE_ID MODEL_ID        COST
--------------------- -------- ----------
  8384342129400492900   4603382517724666236   6779450493542747620
  6779450493542747620         1           1
```

Semantic Data (Insert)

- **John** is the father of **Suzie**

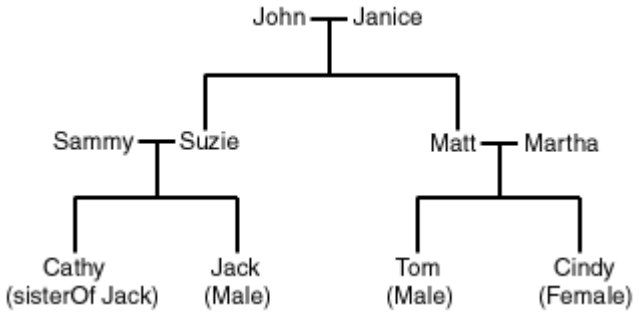



```
INSERT INTO family_rdf_data VALUES
(1, SDO_RDF_TRIPLE_S('family',
'http://www.example.org/family/John',
  'http://www.example.org/family/fatherOf',
  'http://www.example.org/family/Suzie'));
```

```
select to_char(f.triple.rdf_s_id, '99999999999999999999') as Subject,
       to_char(f.triple.rdf_p_id, '99999999999999999999') as Property,
       to_char(f.triple.rdf_o_id, '99999999999999999999') as Object
from family_rdf_data f

SUBJECT              PROPERTY             OBJECT
-------------------- -------------------- --------------------
  4603382517724666236   8384342129400492900   67794504935427476 20
```

Semantic Data (Insert)

- **Martha** is the mother of **Cindy**

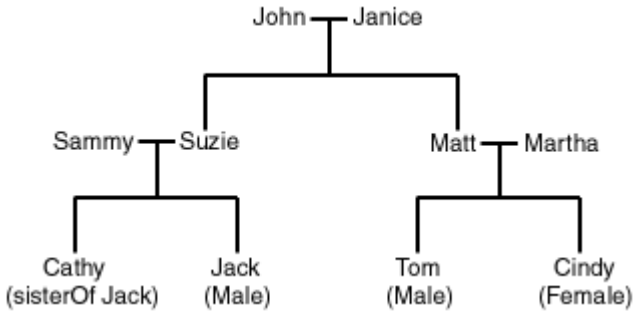



```
INSERT INTO family_rdf_data VALUES (12,
SDO_RDF_TRIPLE_S('family',
'http://www.example.org/family/Martha',
'http://www.example.org/family/motherOf',
'http://www.example.org/family/Cindy'));
```

- **Cathy** is the sister of **Jack**

```
INSERT INTO family_rdf_data VALUES (13,
SDO_RDF_TRIPLE_S('family',
'http://www.example.org/family/Cathy',
'http://www.example.org/family/sisterOf',
'http://www.example.org/family/Jack'));
```

Semantic Data (Insert)



- A **sister** is **female**

```
INSERT INTO family_rdf_data VALUES (25,
SDO_RDF_TRIPLE_S('family',
'http://www.example.org/family/sisterOf',
'http://www.w3.org/2000/01/rdf-schema#domain',
'http://www.example.org/family/Female'));
```

- A **mother** is **female**

```
INSERT INTO family_rdf_data VALUES (29,
SDO_RDF_TRIPLE_S('family',
'http://www.example.org/family/motherOf',
'http://www.w3.org/2000/01/rdf-schema#domain',
'http://www.example.org/family/Female'));
```

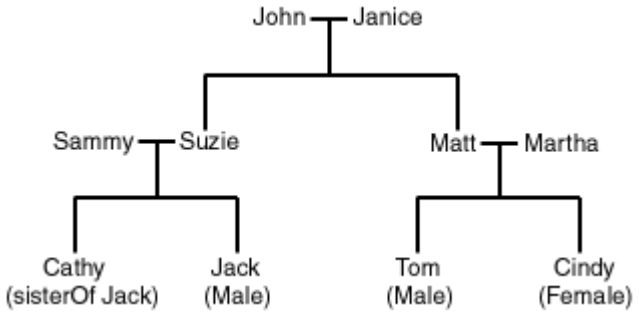
Semantic Data (Insert)

- **Cathy**'s height is 5.8 (**decimal**)

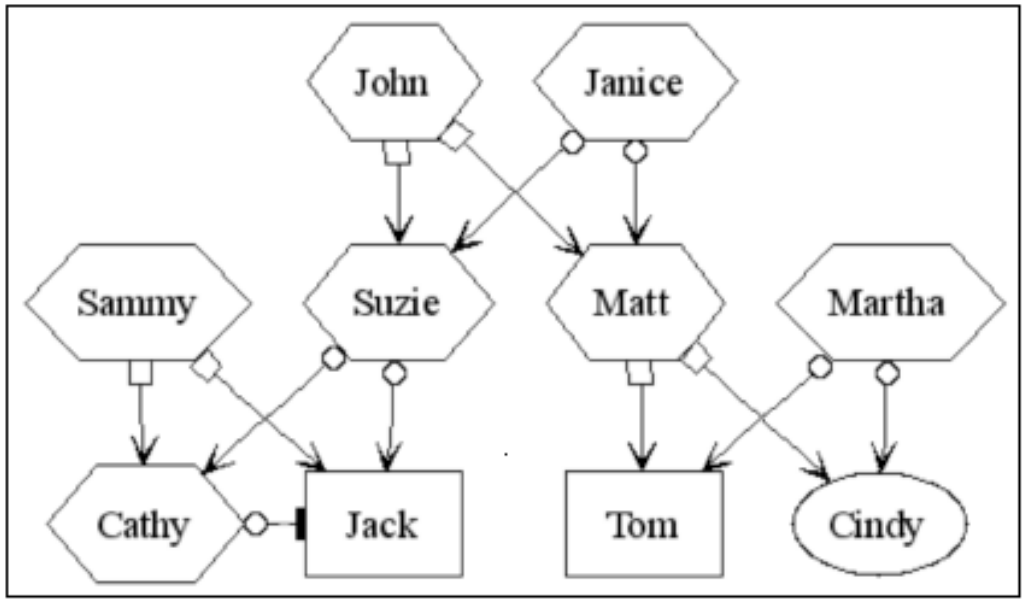
```
INSERT INTO family_rdf_data VALUES (30,
SDO_RDF_TRIPLE_S('family',
'http://www.example.org/family/Cathy',
'http://www.example.org/family/height',
'"5.8"^^xsd:decimal'));
```

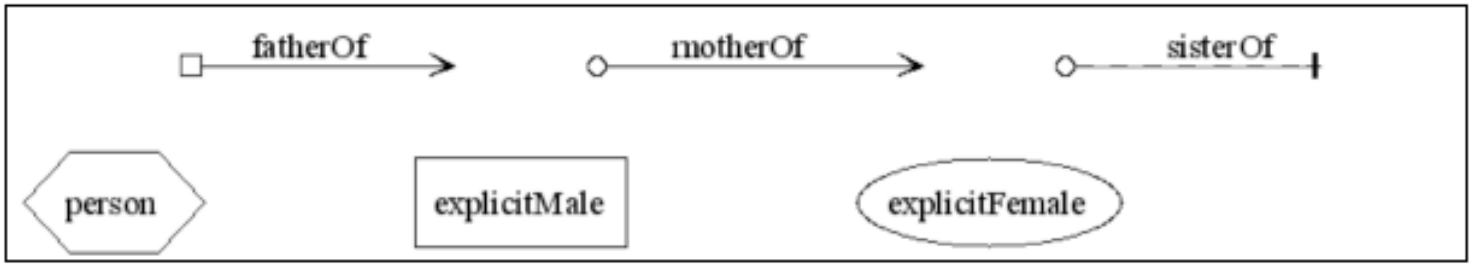
- **Jack**'s height is 6 (**integer**)

```
INSERT INTO family_rdf_data VALUES (31,
SDO_RDF_TRIPLE_S('family',
'http://www.example.org/family/Jack',
'http://www.example.org/family/height',
'"6"^^xsd:integer'));
```



Family Model Data





Using Oracle Database 11g as a Semantic Data Store

- Create a Semantic Network
- Create an RDF/OWL model
- Load RDF/OWL data
- Infer RDF/OWL data
- Query RDF/OWL data and ontologies
- Perform Ontology-assisted Query of relational data

Inference

1. The act or process of deriving logical conclusions from premises known or assumed to be true.
2. The act of reasoning from factual knowledge or evidence.

Inferencing RDF Data

- Native inference engine in Oracle Database for
  - RDF, RDFS, OWL subset, and user-defined rules
  - Rules are stored in rulebases in Oracle Database
- New relationships (triples) are entailed (inferred) from an RDF/OWL graph by applying rules to a model
  - Uses Forward Chaining
  - Minimizes on-the-fly computation for faster queries
- Automatic identification of new relationships (triples)

Ex: hand_fracture :subClassOf arm_fracture,
arm_fracture :subClassOf upper_extremity_fracture
=> hand_fracture :subClassOf upper_extremity_fracture

Vocabularies: RDFS and OWL

- RDFS (RDF Schema)
  - Structuring of resources and properties
    - rdfs:class → Class of resources
    - rdfs:subClassOf → hierarchy of classes
    - rdfs:subPropertyOf → hierarchy of properties
- OWL (Web Ontology Language)
  - Builds on RDFS …
    - Property Characteristics: transitivity, symmetry, functional, inverse functional, inverse
    - Class construction via set operations and property restrictions
  - Separate layers have been defined balancing *expressibility vs. implementability*: OWL Lite, OWL DL, OWL Full

Entailment Rules

- RDFS has 14 entailment rules
  - E.g. rule :

```
aaa   rdfs:domain XXX .
uuu    aaa          yyy .        ➔ uuu rdf:type XXX .
```

- OWLPrime has 50+ entailment rules
  - E.g. rule :

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

```
 xxx   owl:disjointWith yyy .
  a      rdf:type          xxx .
  b      rdf:type          yyy .      ➔ a owl:differentFrom b .
```

- These rules have efficient implementations in Oracle

Entailment Examples

- RDFS Example
  - rdfs:subClassOf is transitive
  - rdfs:subPropertyOf is transitive
  - X rdf:type C, C rdfs:subClassOf SC => X rdf:type SC
- OWL Example
  - :partOf rdf:type owl:TransitiveProperty
  - :friendOf rdf:type owl:SymmetricProperty
- User-defined Rule Example:
  - X :hasParent Y, Y :hasBrother Z => X :hasUncle Z

SEM_MATCH namespaces

The following default namespaces (namespace_id and namespace_val attributes) are used by the SEM_MATCH table function:

('rdf', 'http://www.w3.org/1999/02/22-rdf-syntax-ns#')

('rdfs', 'http://www.w3.org/2000/01/rdf-schema#')

('xsd', 'http://www.w3.org/2001/XMLSchema#')

You can override any of these defaults by specifying the namespace_id value and a different namespace_val value in the aliases attribute.

Creating a RDFS Rules Index

- RDFS inferencing in the family model

```
BEGIN
  SEM_APIS.CREATE_RULES_INDEX(
    'rdfs_rix_family',
    SEM_Models('family'),
    SEM_Rulebases('RDFS'));
END;
/
```

```
select * from mdsys.rdf_rulebase$
```

```
OWNER        ID NAME            STATUS
---------- --- --------------- ----------
SYS           1 RDF             VALID
SYS           2 RDFS            VALID
SYS           3 RDFS++          VALID
SYS           4 OWLSIF          VALID
SYS           5 OWLPRIME        VALID
```

RDFS Rules Index (Cont)

```
Select P_VALUE_ID, START_NODE_ID, CANON_END_NODE_ID,
  END_NODE_ID, MODEL_ID
from MDSYS.SEMI_RDFS_RIX_FAMILY;

 P_VALUE_ID          START_NODE_ID     CANON_END_NODE_ID
--------------------- --------------------- --------------------
          END_NODE_ID MODEL_ID
--------------------- --------
  8440289324123914894   4603382517724666236   2635066808621017356
  2635066808621017356         1

  1459100526467182037   5745257785176001272   2635066808621017356
  2635066808621017356         1
.
.
.

211 rows selected.
```

RDFS Rules Index (Cont)

```
select owner, index_name, index_view_name
From MDSYS.SEM_RULES_INDEX_INFO;
```

SEMI_RDFS_RIX_FAMILY "alias" to RDFI_ RDFS_RIX_FAMILY

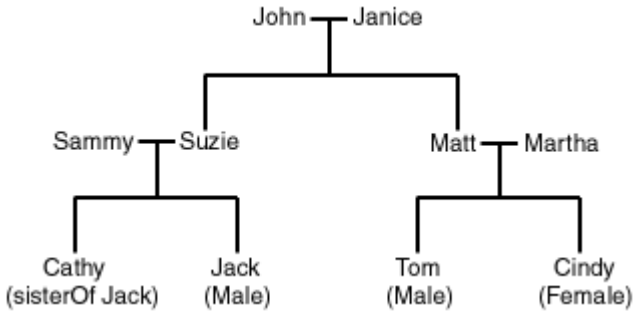
```
OWNER      INDEX_NAME                INDEX_VIEW_NAME
---------- ------------------------- --------------------
RDF        RDFS_RIX_FAMILY           RDFI_RDFS_RIX_FAMILY
```

```
select * from MDSYS.SEM_RULES_INDEX_DATASETS;
```

```
INDEX_NAME                DATA_TYP DATA_NAME
------------------------- -------- -------------------------
RDFS_RIX_FAMILY           MODEL    FAMILY
RDFS_RIX_FAMILY           RULEBASE RDFS
```

SEM_MATCH without inference

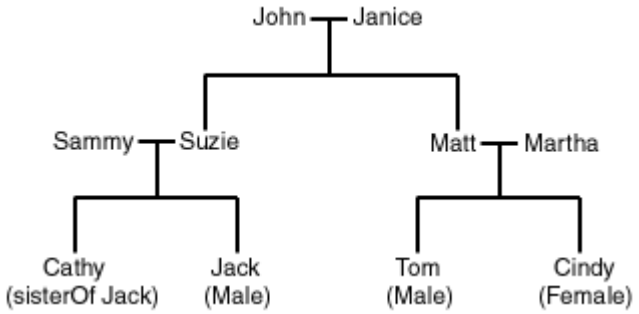
```
SELECT m
  FROM TABLE(SEM_MATCH(
    '(?m rdf:type :Male)',
    SDO_RDF_Models('family'),
    null,
  SDO_RDF_Aliases(SDO_RDF_Alias('','http://www.example.org/family/')),
  null));
```




```
M
------------------------------------
http://www.example.org/family/Jack
http://www.example.org/family/Tom
```

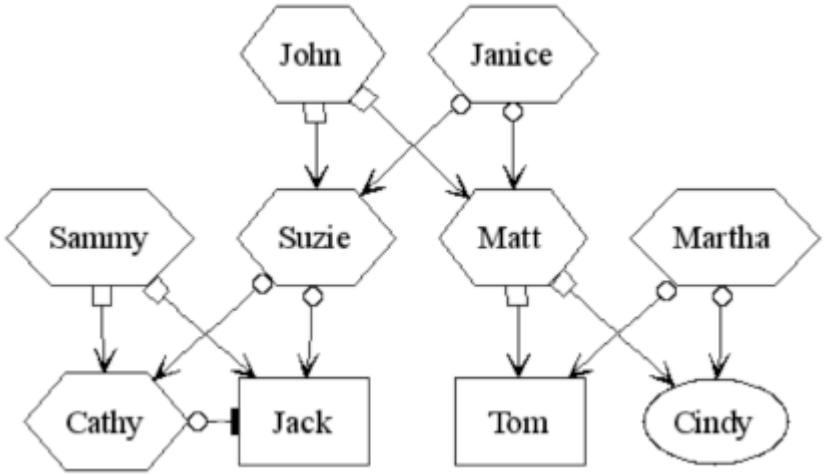
```
SEM_MATCH(
  query          VARCHAR2,
  models         SEM_MODELS,
  rulebases      SEM_RULEBASES,
  aliases        SEM_ALIASES,
  filter         VARCHAR2,
  index_status   VARCHAR2
 ) RETURN ANYDATASET;
```

RDFS inference




```
SELECT m
  FROM TABLE(SEM_MATCH(
    '(?m rdf:type :Male)',
    SDO_RDF_Models('family'),
    SDO_RDF_Rulebases('RDFS'),
  SDO_RDF_Aliases(SDO_RDF_Alias('','http://www.example.org/family/')),
  null));
```

```
M
----------------------------------------
http://www.example.org/family/Jack
http://www.example.org/family/Tom
http://www.example.org/family/John
http://www.example.org/family/Sammy
http://www.example.org/family/Matt
```



If RDFS Rules Index Isn't Defined…

```
.
.
.
*
ERROR at line 1:
ORA-20000: We do not have a rules index
 for this Model-Rulebase combination
ORA-06512: at "MDSYS.RDF_MATCH_IMPL_T",
 line 176
ORA-06512: at line 1
```

OWL Subsets Supported

Three subsets to meet most needs*

- RDFS++
  - RDFS plus owl:sameAs and owl:InverseFunctionalProperty
- OWLSIF
  - Based on Dr. Horst's pD* vocabulary[1]
- OWLPrime
  - owl:TransitiveProperty, SymmetricProperty, FunctionalProperty, InverseFunctionalProperty
  - owl:inverseOf
  - sameAs, differentFrom
  - owl:disjointWith, complementOf
  - owl:hasValue, allValuesFrom, someValuesFrom
  - owl:equivalentClass, equivalentProperty

* Jointly determined with domain experts, customers and partners

1 Completeness, decidability and complexity of entailment for RDF Schema and a semantic extension involving the OWL vocabulary

RDFS/OWL vocabulary constructs included in each rulebase

Rulebase Name	RDFS/OWL constructs included
RDFS++	all RDFS vocabulary constructs, owl:InverseFunctionalProperty, owl:sameAs
OWLSIF	all RDFS vocabulary constructs, owl:FunctionalProperty, owl:InverseFunctionalProperty, owl:SymmetricProperty, owl:TransitiveProperty, owl:sameAs, owl:inverseOf, owl:equivalentClass, owl:equivalentProperty, owl:hasValue, owl:someValuesFrom, owl:allValuesFrom
OWLPRIME	rdfs:subClassOf, rdfs:subPropertyOf, rdfs:domain, rdfs:range, owl:FunctionalProperty, owl:InverseFunctionalProperty, owl:SymmetricProperty, owl:TransitiveProperty, owl:sameAs, owl:inverseOf, owl:equivalentClass, owl:equivalentProperty, owl:hasValue, owl:someValuesFrom, owl:allValuesFrom, owl:differentFrom, owl:disjointWith, owl:complementOf

11g OWL Inference PL/SQL API

- **SEM_APIS.CREATE_ENTAILMENT**(
  - Index_name
  - sem_models('GraphTBox', 'GraphABox', …),
  - sem_rulebases('OWLPrime'),
  - passes,
  - Inf_components,
  - Options

)
  - Use "PROOF=T" to generate inference proof

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

- **SEM_APIS.VALIDATE_ENTAILMENT**(
  - sem_models(('GraphTBox', 'GraphABox', …),
  - sem_rulebases('OWLPrime'),
  - Criteria,
  - Max_conflicts,
  - Options

)

Typical Usage:

- First load RDF/OWL data
- Call create_entailment to generate inferred graph
- Call validate_entailment to find inconsistencies

OWL Inferencing (Native)

```
EXECUTE sem_apis.create_entailment('owltst_idx',
sem_models('owltst'), sem_rulebases('OWLPRIME'));
```

View **mdsys.semi_owltst_idx**

```
SEM_APIS.CREATE_ENTAILMENT(
index_name_in      IN  VARCHAR2,
models_in          IN  SEM_MODELS,
rulebases_in       IN  SEM_RULEBASES,
passes             IN  NUMBER DEFAULT
SEM_APIS.REACH_CLOSURE,
inf_components_in IN  VARCHAR2 DEFAULT NULL,
options            IN  VARCHAR2 DEFAULT NULL);
```

Adding User-Defined Rules

- Oracle Database 10gR2 RDF and 11g RDF/OWL supports user-defined rules in this form (filter is supported)

Antecedents		Consequents
?x :parentOf ?y . ?z :brotherOf ?x .	➔	?z :uncleOf ?y

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

```
<owl:Class rdf:ID="FemaleAstronaut">
  <rdfs:label>chair</rdfs:label>
  <owl:intersectionOf rdf:parseType="Collection">
    <owl:Class rdf:about="#Female" />
    <owl:Class rdf:about="#Astronaut" />
  </owl:intersectionOf>
</owl:Class>
```

➔

1. ➔ :FemaleAstronaut rdfs:subClassOf :Female
2. ➔ :FemaleAstronaut rdfs:subClassOf :Astronaut
3. ?x rdf:type :Female .
   ?x rdf:type :Astronaut . ➔
   x rdf:type :FemaleAstronaut

Adding a User-Defined RDFS Rule Base

```
EXECUTE SEM_APIS.CREATE_RULEBASE('family_rb');

INSERT INTO mdsys.semr_family_rb VALUES(
  'grandparent_rule',
  '(?x :parentOf ?y) (?y :parentOf ?z)',
  NULL,
  '(?x :grandParentOf ?z)',
  SEM_ALIASES(SDO_SEM_ALIAS('','http://www.example.org/family/')));
```

```
RULE_NAME        ANTECEDENTS                          CONSEQUENTS
---------------- ------------------------------------ ------------------------------
GRANDPARENT_RULE (?X <http://www.example.org/fa (?X <http://www.example.org/fa
                 mily/parentOf> ?Y)(?Y <http:// mily/grandParentOf> ?Z)
                 www.example.org/family/parentO
                 f> ?Z)
```

User-defined RDFS Rules for family_rb

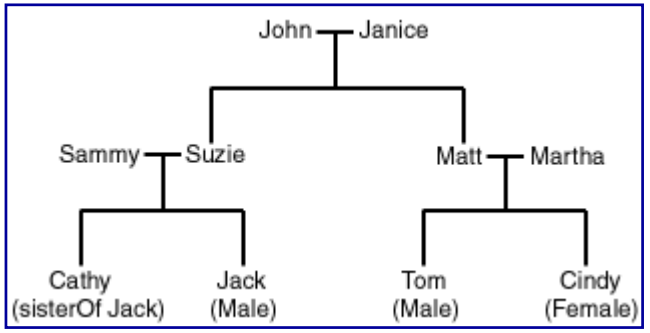
```
SELECT x grandfather, y grandchild
  FROM TABLE(SEM_MATCH(
    '(?x :grandParentOf ?y) (?x rdf:type :Male)',
    SDO_RDF_Models('family'),
    SDO_RDF_Rulebases('RDFS'),
    SDO_RDF_Aliases(SDO_RDF_Alias('','http://www.example.org/family/')),
    null));
```

No rows selected

```
EXECUTE SDO_RDF_INFERENCE.DROP_RULES_INDEX ('rdfs_rix_family');

BEGIN
  SDO_RDF_INFERENCE.CREATE_RULES_INDEX(
    'rdfs_rix_family',
    SDO_RDF_Models('family'),
    SDO_RDF_Rulebases('RDFS','family_rb'));
END;
/
```

User-defined RDFS Rules for family_rb

- Select all grandfathers and their grandchildren from the family model
- Use inferencing from both the RDFS and family_rb rulebases

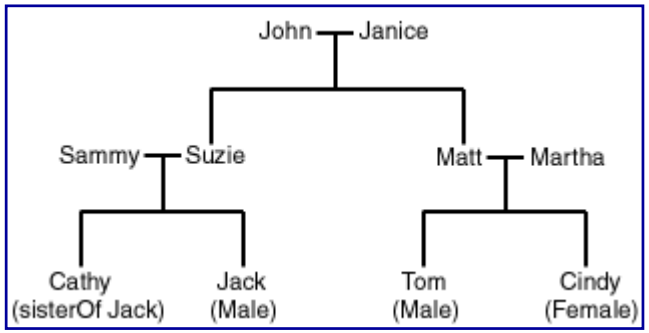



```
SELECT x grandfather, y grandchild
  FROM TABLE(SEM_MATCH(
    '(?x :grandParentOf ?y) (?x rdf:type :Male)',
    SDO_RDF_Models('family'),
    SDO_RDF_Rulebases('RDFS','family_rb'),
    SDO_RDF_Aliases(SDO_RDF_Alias('','http://www.example.org/family/')),
    null));
```

GRANDFATHER	GRANDCHILD
http://www.example.org/family/John	http://www.example.org/family/Tom
http://www.example.org/family/John	http://www.example.org/family/Cathy
http://www.example.org/family/John	http://www.example.org/family/Jack
http://www.example.org/family/John	http://www.example.org/family/Cindy

Changing User-defined Rules

```
UPDATE mdsys.rdfr_family_rb SET
  antecedents = '(?x :parentOf ?y) (?y :parentOf ?z) (?z :height ?h)',
  filter = '(h >= 6)',
  aliases = SDO_RDF_Aliases(SDO_RDF_Alias('','http://www.example.org/family/'))
WHERE rule_name = 'GRANDPARENT_RULE';

** Re-create rules index **

SELECT x grandfather, y grandchild
  FROM TABLE(SEM_MATCH(
    '(?x :grandParentOf ?y) (?x rdf:type :Male)',
    SDO_RDF_Models('family'),
    SDO_RDF_RuleBases('RDFS','family_rb'),
    SDO_RDF_Aliases(SDO_RDF_Alias('','http://www.example.org/family/')),
    null));
```




```
GRANDFATHER                          GRANDCHILD
------------------------------------ ------------------------------------
http://www.example.org/family/John   http://www.example.org/family/Jack
http://www.example.org/family/John   http://www.example.org/family/Cindy
```

User-Defined OWL Inferencing

```
INSERT INTO owltst VALUES (1,
sdo_rdf_triple_s('owltst',
'http://foo.com/name/John', 'http://foo.com/rel/fatherOf',
'http://foo.com/name/Mary'));

INSERT INTO owltst VALUES (1,
sdo_rdf_triple_s('owltst',
'http://foo.com/name/Jack', 'http://foo.com/rel/brotherOf',
'http://foo.com/name/John'));

-- Create a user-defined rulebase.
EXECUTE sem_apis.create_rulebase('user_rulebase');

-- Insert a simple "uncle" rule.
INSERT INTO mdsys.semr_user_rulebase
VALUES ('uncle_rule',
'(?x <http://foo.com/rel/brotherOf> ?y)(?y
<http://foo.com/rel/fatherOf> ?z)',
NULL, '(?x <http://foo.com/rel/uncleOf> ?z)', null);

-- In the following statement, 'USER_RULES=T' is required,
to
-- include the original graph plus the inferred triples.

EXECUTE sem_apis.create_entailment('owltst2_idx',
sem_models('owltst'),
sem_rulebases('OWLPRIME','USER_RULEBASE'),
SEM_APIS.REACH_CLOSURE, null, 'USER_RULES=T');

-- In the result of the following query, :Jack :uncleOf :Mary
is inferred.

SELECT s,p,o FROM table(SEM_MATCH('(?s ?p ?o)',
SEM_MODELS('OWLTST'),
SEM_RULEBASES('OWLPRIME','USER_RULEBASE'), null,
null ));
```

Testing the Soundness of OWL Inference

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

TripleID1 :emailAddress rdf:type owl:InverseFunctionaProperty .
TripleID2 :John :emailAddress :John_at_yahoo_dot_com .
TripleID3 :Johnny :emailAddress :John_at_yahoo_dot_com .

:John owl:sameAs :Johnny **(proof := TripleID1, TripleID2, TripleID3, “IFP”)**

OWL Inferencing Proofs

```
EXECUTE sem_apis.create_entailment('owltst_idx', sem_models('owltst'),
sem_rulebases('owlprime'), SEM_APIS.REACH_CLOSURE, 'SAM', 'PROOF=T');
```

```
SELECT link_id || ' generated by ' || explain as triple_and_its_proof
FROM mdsys.semi_owltst_idx;

TRIPLE_AND_ITS_PROOF
--------------------------------------------------------------------
8_5_5_4 generated by 4_D_5_5 : SYMM_SAMH_SYMM
8_4_5_4 generated by 8_5_5_4 4_D_5_5 : SAM_SAMH
. . .
. . .
```

OWL Inferencing Proofs (Cont)

```
SELECT to_char(x.triple.rdf_m_id, 'FMXXXXXXXXXXXXXXXX') ||'_'|| to_char(x.triple.rdf_s_id, 'FMXXXXXXXXXXXXXXXX') ||'_'||
to_char(x.triple.rdf_p_id, 'FMXXXXXXXXXXXXXXXX') ||'_'||  to_char(x.triple.rdf_c_id, 'FMXXXXXXXXXXXXXXXX'), x.triple.get_triple()
FROM ( SELECT sdo_rdf_triple_s(
t.canon_end_node_id, t.model_id, t.start_node_id, t.p_value_id, t.end_node_id) triple
FROM (select * from mdsys.semm_owltst union all select * from mdsys.semi_owltst_idx)  t
WHERE t.link_id IN ('4_D_5_5','8_5_5_4'))  x;
```

```
LINK_ID X.TRIPLE.GET_TRIPLE()(SUBJECT, PROPERTY, OBJECT)
---------- ----------------------------------------------------------------
4_D_5_5 SDO_RDF_TRIPLE('<http://foo.com/name/John>', '<http://www.w3.org/2002/07/owl#sameAs>', '<http://foo.com/name/JohnQ>')
8_5_5_4 SDO_RDF_TRIPLE('<http://foo.com/name/JohnQ>', '<http://www.w3.org/2002/07/owl#sameAs>', '<http://foo.com/name/John>')
```

For the proof entry 8_5_5_4 generated by 4_D_5_5 : SYMM_SAMH_SYMM

For the triple with LINK_ID = 8_5_5_4, it is inferred from the triple with 4_D_5_5 using the symmetricity of owl:sameAs.

Validating an OWL Triple

```
-- Insert an offending triple.

insert into owltst values (1, sdo_rdf_triple_s('owltst',

'urn:C1',

'http://www.w3.org/2000/01/rdf-schema#subClassOf',

'http://www.w3.org/2002/07/owl#Nothing'));


. . .

lva := sem_apis.validate_entailment(sem_models('OWLTST'),
sem_rulebases('OWLPRIME')) ;

. . .
```

```
Offending entry := 1   10001   (4_2_4_8 2 4 8)   Unsatisfiable class.
```

Advanced Inference Options

Give users more control over inference process

- Selective inference (component based)
  - Allows more focused inference.
  - E.g. give me only the subClassOf hierarchy.
- Set number of passes
  - Normally, inference continue till no further new triples found
  - Users can set the number of inference passes to see if what they are interested has already been inferred
  - E.g. I want to know whether this person has more than 10 friends
- Set tablespaces used, parallel index build
- Change statistics collection scheme

Using Oracle Database 11g as a Semantic Data Store

- Create a Semantic Network
- Create an RDF/OWL model
- Load RDF/OWL data
- Infer RDF/OWL data
- Query RDF/OWL data and ontologies
- Perform Ontology-assisted Query of relational data

Querying RDF/OWL Data

- Matches RDF/OWL graph patterns with stored data
- Returns a table of results
- Supports SQL operators/functions to process results
- Supports Table Function Rewrite
- No staging when combined with relational queries

```
SELECT ...
FROM ...,
TABLE (SEM_MATCH invocation
 ) t, ...
 WHERE ...
```

```
SEM_MATCH (
    '(?x :partOf :NorthAmerica)',          -- pattern: all persons
    SEM_Models('gmap'),                    --  RDF/OWL models
    SEM_Rulebases('OWLPRIME'),             -- rulebases
    SEM_Aliases(...)                       -- aliases
    null                                   -- no filter condition
)
```

Table Columns returned by SEM_MATCH

Each returned row contains one (or more) of the following cols for each variable ?x in the graph-pattern:

Column Name	Type	Description
x	varchar2	Value matched with ?x
x$rdfVTYP	varchar2	Value TYPe: URI, Literal, or Blank Node
x$rdfLTYP	varchar2	Literal TYPe: e.g., xsd:integer
x$rdfCLOB	CLOB	CLOB value matched with ?x
x$rdfLANG	varchar2	LANGuage tag: e.g., “en-us”

Projection Optimization: Only the columns referred to by the containing query are returned.

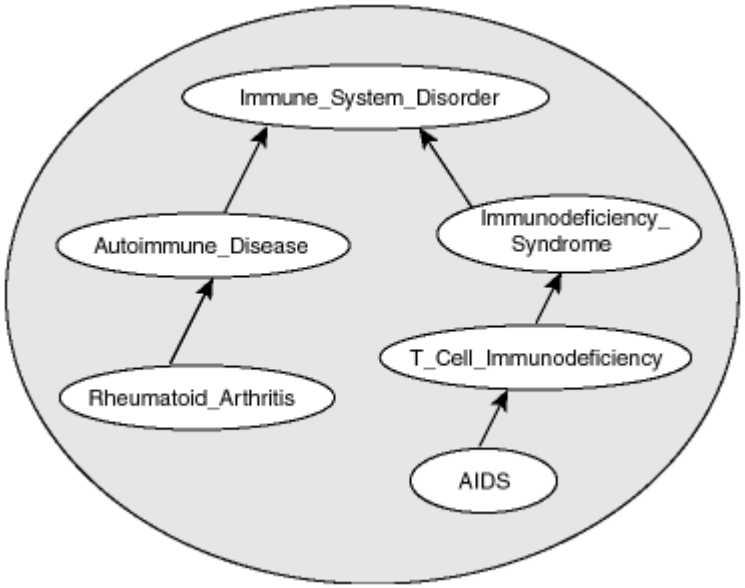
Using Oracle Database 11g as a Semantic Data Store

- Create a Semantic Network
- Create an RDF/OWL model
- Load RDF/OWL data
- Infer RDF/OWL data
- Query RDF/OWL data and ontologies
- Perform Ontology-assisted Query of relational data

Ontology-Assisted SQL Query

- Traditionally the relationship between two terms is only checked in a syntactic manner
- Need a way to check semantic relationships by consulting an ontology
- Introduces two operators
  - SEM_RELATED (<col>,<pred>, <ontologyTerm>, <ontologyName> [,<invoc_id>])
  - SEM_DISTANCE (<invoc_id>) ← Ancillary Oper.

Operators SEM_RELATED SEM_DISTANCE

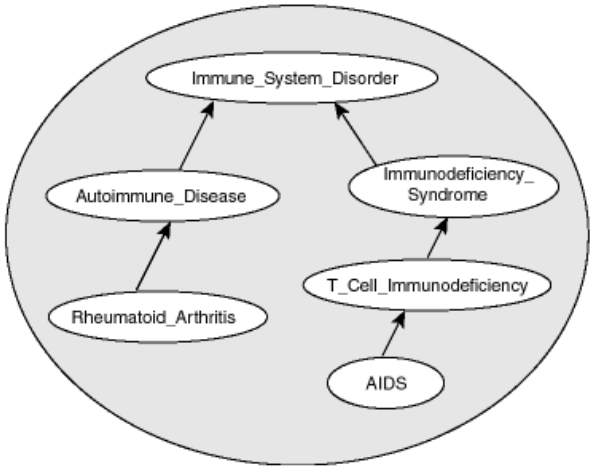



```
CREATE TABLE CANCER_RDF_DATA (ID NUMBER,
TRIPLE SDO_RDF_TRIPLE_S);

EXEC SEM_APIS.CREATE_SEM_MODEL('cancer',
'CANCER_RDF_DATA', 'TRIPLE');
```

```
SELECT diagnosis

FROM patients

WHERE diagnosis =
'Immune_System_Disorder';

No rows selected
```

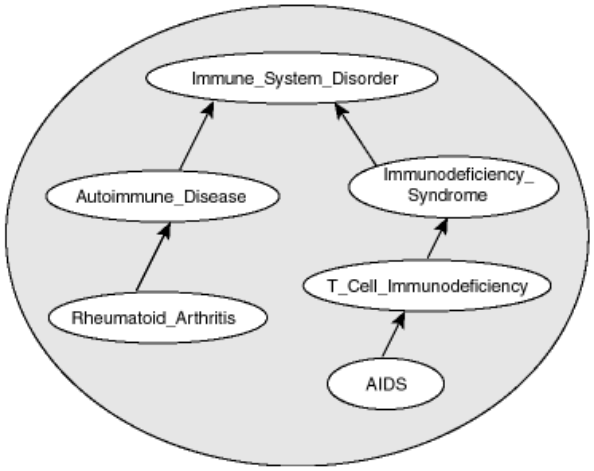
```
PATIENT_ID     DIAGNOSIS
----------      -------------------------
1234            Rheumatoid_Arthritis
2345            Immunodeficiency_Syndrome
3456            AIDS
```

Operators SEM_RELATED SEM_DISTANCE (cont)



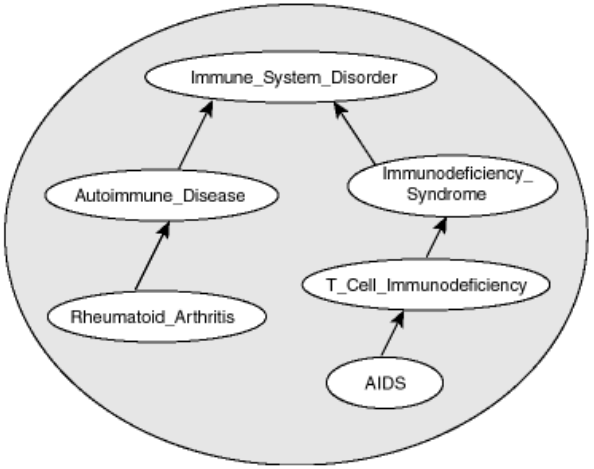

```
col diagnosis format a60
SELECT PATIENT_ID, DIAGNOSIS
FROM PATIENTS
WHERE SEM_RELATED(DIAGNOSIS,'rdfs:subClassOf',
'<http://www.example.org/cancer/Immune_System_Disorder>',
sem_models('cancer'), NULL) = 1;

-- PATIENT_ID DIAGNOSIS
-- ---------- ------------------------------------------------------------
--       2345 <http://www.example.org/cancer/Immunodeficiency_Syndrome>
```

Operators SEM_RELATED SEM_DISTANCE (Use model and rulebase)




```
EXEC SEM_APIS.CREATE_ENTAILMENT
('diagnosis_ent_sem_idx', sem_models('cancer')
,sem_rulebases('owlprime'));
SELECT PATIENT_ID, DIAGNOSIS
FROM PATIENTS
WHERE SEM_RELATED(DIAGNOSIS,
                  'rdfs:subClassOf',
                  '<http://www.example.org/cancer/Immune_System_Disorder>',
                  sem_models('cancer'),
                  sem_rulebases('OWLPRIME')) = 1;
PATIENT_ID DIAGNOSIS
---------- ------------------------------------------------------------
      1234 <http://www.example.org/cancer/Rheumatoid_Arthritis>
      2345 <http://www.example.org/cancer/Immunodeficiency_Syndrome>
      3456 <http://www.example.org/cancer/AIDS>
```

Operators SEM_RELATED
SEM_DISTANCE (cont)




```
SELECT diagnosis, SEM_DISTANCE(123) FROM patients
WHERE SEM_RELATED (diagnosis,
'<http://www.w3.org/2000/01/rdf-schema#subClassOf>'
'<http://www.example.org/cancer/Immune_System_Disorder>',
sem_models('cancer'), sem_rulebases('OWLPRIME'), 123) = 1;

DIAGNOSIS                                                  SEM_DISTANCE(123)
---------------------------------------------------------- -----------------
<http://www.example.org/cancer/Rheumatoid_Arthritis>                       2
<http://www.example.org/cancer/Immunodeficiency_Syndrome>                  1
<http://www.example.org/cancer/AIDS>                                       3
```

Creating an Index for SEM_RELATED

The SEM_RELATED operator requires an index on the application table

```
CREATE INDEX diagnosis_sem_idx ON patients (diagnosis)
INDEXTYPE IS MDSYS.SEM_INDEXTYPE;
```

PATIENTS TABLE

PATIENT_ID	DIAGNOSIS
1234	Rheumatoid_Arthritis
2345	Immunodeficiency_Syndrome
3456	AIDS

Tutorial Agenda

- RDF Concepts
- Using Oracle Database 11g as a Semantic Data Store
- Jena Adaptor
- System Metadata and Views
- National Cancer Institute Ontology Case Study
- Best Practices

Jena Adaptor for Oracle Database

- Provides SPARQL access to Oracle Database
- Integrates 3rd party tools w/ Oracle Database
- Facilitates integration with external reasoners
  - e.g. PelletInfGraph can be created on top of GraphOracleSem
- Supports Bulk, Batch, DML data load with long literals
- Implements Jena Graph/Model/BulkUpdateHandler APIs
- Scaleable and integrated
  - Data not cached in memory
  - SPARQL queries processed as SQL in Oracle Database
  - A SPARQL query with only conjunctive patterns is a single SEM_MATCH query
- Download from OTN, supported by Oracle

Oracle Jena Adaptor Example

```
String queryString =
      " PREFIX foaf: <http://xmlns.com/foaf/0.1/> " +
      "SELECT ?person WHERE { ?person foaf:firstName \"Julie\" .
  } ";

QueryExecution qexec = QueryExecutionFactory.create(
      QueryFactory.create(queryString), modelOracleSem);
```

Tutorial Agenda

- RDF Concepts
- Using Oracle Database 11g as a Semantic Data Store
- Jena Adaptor
- Performance
- System Metadata and Views
- National Cancer Institute Ontology Case Study
- Best Practices

Metadata for Models

- Per Model:
  - Row in Table MDSYS.SEM_MODEL$
  - View SEMM_model_name

```
COL owner       FORMAT a15
COL model_id    FORMAT 9999
COL model_name  FORMAT a14
COL table_name  FORMAT a21
COL column_name FORMAT a11
SELECT owner, model_id, model_name, table_name,
column_name
FROM MDSYS.SEM_MODEL$;
```

Metadata for Models (cont)

```
OWNER            MODEL_ID MODEL_NAME     TABLE_NAME            COLUMN_NAME
-------------- -------- -------------- --------------------- -----------
SCOTT                   1 NATURE         NATURE_RDF_DATA       TRIPLE
SCOTT                   2 PIM            PERSONAL_INFO         RDF_DATA
SCOTT                   3 BOOKSTORE      BOOK_REVIEWS          RDF_DATA
SCOTT                   4 NSU            NSU                   TRIPLE
SCOTT                   5 NATURE2        NATURE                TRIPLE
SCOTT                   6 JOBS           JOBS                  TRIPLE
SCOTT                   7 FAMILY         FAMILY_RDF_DATA       TRIPLE

7 rows selected.
```

- $ORACLE_HOME/md/demo/network/rdf_demos
  - Install_demo.sql

Metadata for Models (cont)

```
conn mdsys/mdsys
Connected.
select object_name, object_type
from user_objects
where object_name like '%BOOKSTORE%';

OBJECT_NAME                    OBJECT_TYPE
------------------------------ -------------------
SEMM_BOOKSTORE                 VIEW
RDFM_BOOKSTORE                 VIEW
BOOKSTORE_INS                  TRIGGER
BOOKSTORE_DML                  TRIGGER
BOOKSTORE_DEL                  TRIGGER
```

Metadata for Models (cont)

```
set long 9000000
select text
from user_views where view_name = 'SEMM_BOOKSTORE';


TEXT
--------------------------------------------------------------------------------
select "P_VALUE_ID","START_NODE_ID","CANON_END_NODE_ID","END_NODE_ID","MODEL_ID"
,"COST","CTXT1","CTXT2","DISTANCE","EXPLAIN","PATH","LINK_ID" from rdf_link$ par
tition (model_3)
```

Rules Indexes

- When a rules index is created
  - Two views are created:
    - RDFI_rules_index_name
    - SEMI_rules_index_name
- The views have the same columns as the model view SEMM_model_name
- Both, model and rules index views are based on RDF_LINK$ table

RDFS and RDF vocabulary constructs

```
desc mdsys.semr_rdfs
```

Name	Null?	Type
RULE_NAME	NOT NULL	VARCHAR2(30)
ANTECEDENTS		VARCHAR2(4000)
FILTER		VARCHAR2(4000)
CONSEQUENTS	NOT NULL	VARCHAR2(4000)
ALIASES		RDF_ALIASES

RDFS and RDF vocabulary constructs

```
RULE_NAME
------------------------------
RDF-AXIOMS
RDF1
RDFS-AXIOMS-DOMAIN
RDFS-AXIOMS-MISC
RDFS-AXIOMS-RANGE
RDFS10
RDFS11
RDFS12
RDFS2
RDFS3
RDFS4A
RDFS4B
RDFS5
RDFS6
RDFS7
RDFS8
RDFS9
```

RDFS

RDF

```
RULE_NAME
------------------------------
RDF-AXIOMS
RDF1
```

Tutorial Agenda

- RDF Concepts
- Using Oracle Database 11g as a Semantic Data Store
- Jena Adaptor
- System Metadata and Views
- National Cancer Institute Ontology Case Study
- Best Practices

NCI Demo

- Scenario
  - Schema “nciuser”
  - Tables:
    - NCI_RDF_DATA (Model table)
    - STABLE (Staging table)
    - PATIENTS_DATA (Application table, used to simulate patients information)
- Model: NCI
- Load data into the NCI Model -> Populate Application Table PATIENTS_DATA -> Queries

NCI Demo

First Level Classes

Anatomy_Kind

Biological_Process_Kind

Chemicals_and_Drugs_Kind

Chemotherapy_Regimen_Kind

Clinical_or_Research_Activity_Kind

Diagnostic_and_Prognostic_Factors_Kind

EO_Anatomy_Kind

EO_Findings_and_Disorders_Kind

Equipment_Kind

Findings_and_Disorders_Kind

Gene_Kind

NCI_Kind

Organism_Kind

Properties_or_Attributes_Kind

Protein_Kind

Retired_Kind

Technique_Kind

Create User, Table and Model

```
create user nciuser identified by nciuser
default tablespace rdf_users
temporary tablespace temp;

grant create session, resource to nciuser;

connect nciuser/nciuser

create table nci_rdf_data (id number, triple sdo_rdf_triple_s);
execute sem_apis.create_sem_model('nci', 'nci_rdf_data', 'triple');
```

Load Data

```
CREATE TABLE stable (
                    RDF$STC_sub varchar2(4000) not null,
                    RDF$STC_pred varchar2(4000) not null,
                    RDF$STC_obj varchar2(4000) not null,
                    RDF$STC_sub_ext varchar2(64),
                    RDF$STC_pred_ext varchar2(64),
                    RDF$STC_obj_ext varchar2(64),
                    RDF$STC_canon_ext varchar2(64)
);

grant insert on nci_rdf_data to mdsys;

grant SELECT,
UPDATE(RDF$STC_sub_ext,RDF$STC_pred_ext,RDF$STC_obj_ext,RDF$STC_canon_ext) on
stable to MDSYS;

!$ORACLE_HOME/bin/sqlldr userid=nciuser/nciuser
control=$ORACLE_HOME/md/demo/network/rdf_demos/bulkload.ctl data=nci_z.nt
direct=true skip=0 load=1000000 discardmax=0 bad=d0.bad discard=d0.rej
log=d0.log errors=100000000
```

From Staging Table to the Model

The syntax for the bulk_load_from_staging_table procedure is:

```
exec sem_apis.bulk_load_from_staging_table('<model_name>', '<owner of the
staging table>', '<staging table name>');
```

```
exec sem_apis.bulk_load_from_staging_table('nci','nciuser','stable');
```

Create the rules index

```
execute sem_apis.drop_entailment('nci_idx');

execute sem_apis.create_entailment('nci_idx',sem_models('NCI'),
sem_rulebases('owlprime'),0,null);
```

Patients_Data Table

```
desc PATIENTS_DATA
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 ID                                        NOT NULL NUMBER
 NAME                                               VARCHAR2(35)
 AGE                                                NUMBER
 DIAGNOSIS                                          VARCHAR2(200)
 DIAG_LENGTH                                        NUMBER
```

Build Semantic Index

```
create index nciIndex on patients_data(diagnosis)
indextype is mdsys.sem_indextype
parameters ('ONTOLOGY_MODEL(nci), RULEBASE(owlprime)');

execute
dbms_stats.gather_index_stats('nciuser','nciIndex');
```

Querying the Model

```
col diagnosis format a65
select distinct diagnosis, SEM_DISTANCE(123)from patients_data
where sem_related (diagnosis,
'<http://www.w3.org/2000/01/rdf-schema#subClassOf>',
'<http://www.mindswap.org/2003/nciOncology.owl#Upper_Extremity_Fracture>',
    sdo_rdf_models('nci'),
    sdo_rdf_rulebases('owlprime'), 123) = 1
ORDER BY SEM_DISTANCE(123) asc;
```

```
.   .   .
DIAGNOSIS
-----------------------------------------------------------------
SEM_DISTANCE(123)
-----------------
<http://www.mindswap.org/2003/nciOncology.owl#Humerus_Fracture>
                1
<http://www.mindswap.org/2003/nciOncology.owl#Finger_Fracture>
                2
8 rows selected.
```

Querying the Model (cont)

```
select distinct diagnosis, SEM_DISTANCE(123) from patients_data
where sem_related (diagnosis,
'<http://www.w3.org/2000/01/rdf-schema#subClassOf>',
'<http://www.mindswap.org/2003/nciOncology.owl#Connective_Tissue_Disorder>',
    sdo_rdf_models('nci'),
    sdo_rdf_rulebases('owlprime'), 123) = 1
ORDER BY SEM_DISTANCE(123) asc;
```

```
 .   .   .
DIAGNOSIS
--------------------------------------------------------------
SEM_DISTANCE(123)
-----------------
<http://www.mindswap.org/2003/nciOncology.owl#Telangiectatic_Osteosarcoma>
                6
<http://www.mindswap.org/2003/nciOncology.owl#Childhood_Intracortical_Osteosarcoma>
                7
<http://www.mindswap.org/2003/nciOncology.owl#Childhood_Parosteal_Osteosarcoma>
                7
156 rows selected.
```

NCI Demo Summary

- Introduction to the NCI Semantic Network
- Overview
- Scenario
- Create the Infrastructure and Model to Hold the Data
- Load the Data into the Model
- The Patients Table
- Build Semantic Indexes
- Querying the Model

Tutorial Agenda

- RDF Concepts
- Using Oracle Database 11g as a Semantic Data Store
- Jena Adaptor
- System Metadata and Views
- National Cancer Institute Ontology Case Study
- Best Practices

Loading via bulk-load API

- if you know there are no syntax errors, consider using a simpler sqlldr control file (with less strict checks)
- if doing a large load (in terms estimated count of new values vs. pre-existing values), consider using VALUES_TABLE_INDEX_REBUILD option
- if doing a large append (in terms of estimated count of new triples vs. pre-existing triples) into an RDF model, consider using RDF_MODEL_INDEX_REBUILD option

Summary

Semantic Technology support in the database

- Store RDF/OWL data and ontologies
- Inference new RDF/OWL triples via native inferencing
- Query RDF/OWL data and ontologies
- Ontology-Assisted Query of relational data

Whitepapers, documentation, sample code, downloads:

oracle.com/technology/tech/semantic_technologies

For More Information

search.oracle.com

or

oracle.com