RDF Support in Oracle RDBMS

Souripriya Das, Ph.D.
Consultant Member of Technical Staff
Oracle New England Development Center

ORACLE

Overview

Three types of database objects

- Model → RDF graph consisting of a set of triples
- Rulebase → Set of (user-defined) rules
- Rule Index → Entailed RDF graph

We discuss following aspects for each type of object

- DDL
- DML
- Views
- Security

RDF Query (with Inference)

RDF Models

Model: Overview

- Each RDF Model (graph) consists of a set of triples
- A triple (statement) consists of three components
  - Subject → URI or blank node
  - Predicate → URI
  - Object → URI or literal or blank node
- A statement itself can be a resource (allowing nested graphs)

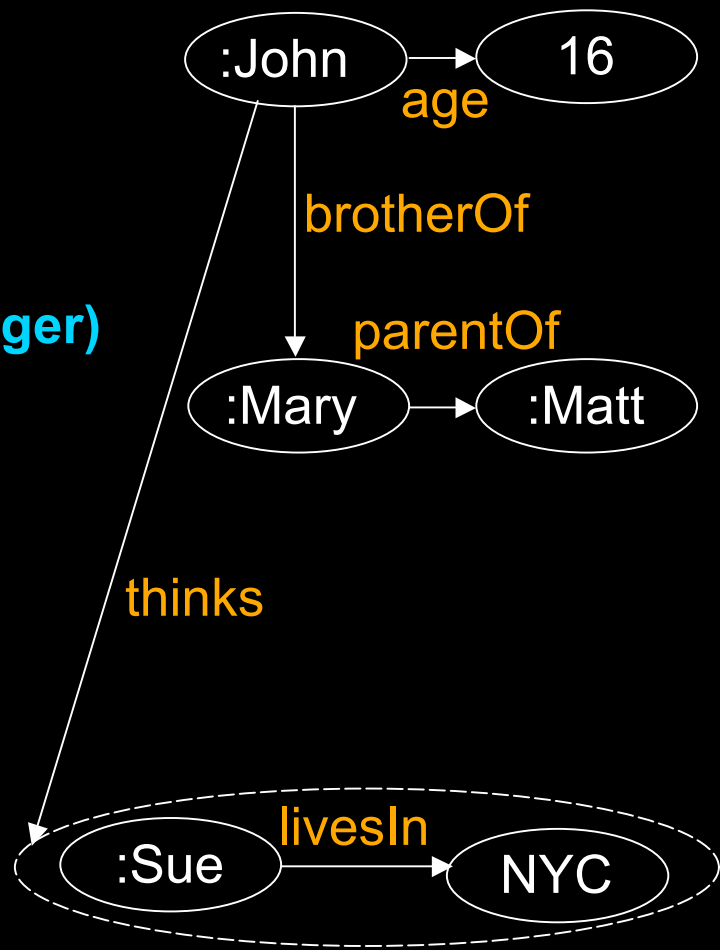
Model: Example

Family:

(:John :brotherOf :Mary)
(:John :age "16"^^xsd:Integer)
(:Mary :parentOf :Matt)
(:John :name "John")
(:Mary :name "Mary")

Reification:

(:John :thinks _:S1)
(_:S1 rdf:subject :Sue)
(_:S1 rdf:predicate :livesIn)
(_:S1 rdf:object "NYC")



RDF Query

SDO_RDF_MATCH Table Func

- Arguments
  - Graph pattern
    - A sequence of triple patterns
    - Triple patterns typically use variables
  - RDF Data set → a set of models
  - Filter
  - Aliases
- …

```
FROM TABLE(SDO_RDF_MATCH(
            '(?x :brotherOf ?y) (?y :parentOf ?z)',
            SDO_RDF_Models('family'),
            …
        )) t
…
```

ORACLE

SDO_RDF_MATCH: return

Columns (of type VARCHAR2) in each returned row:

- For each variable ?x in Graph Pattern
  - x
  - x$rdfVTYP
    - URI, Literal, Blank node
  - x$rdfLTYP
    - Specific literal type (e.g., xsd:integer)
  - x$rdfCLOB
    - Contains actual value, if ?x matches a CLOB value
  - x$rdfLANG
    - Language tag, if any (e.g., “en-us”)
- If no variable in Graph Pattern
  - A dummy column

SDO_RDF_MATCH: matching

Matching multiple representations

- The same point in value space may have multiple representations
  - "10"^^xsd:Integer
  - "10"^^xsd:PositiveInteger
  - "010"^^xsd:Integer
  - "000010"^^xsd:Integer
- SDO_RDF_MATCH automatically resolves these

RDF Query: Example

- Find salary and hiredate of all the uncles
- SELECT emp.name, emp.salary, emp.hiredate

```
FROM emp,
        TABLE(SDO_RDF_MATCH(
                ‘(?x :brotherOf ?y)
                 (?y :parentOf  ?z)
                 (?x :name      ?name)’,
                SDO_RDF_Models(‘family'),
        …)) t
WHERE emp.name=t.name;
```

- Use of SDO_RDF_MATCH allows embedding a graph query in a SQL query

RDF Query: Example 2

- Find pairs of persons residing at the same address where the first person rents a truck and the second person buys a fertilizer
-
```
SELECT t3.x name1, t3.y name2
FROM AddrTable t1, AddrTable t2,
        TABLE(SDO_RDF_MATCH(
                ‘(?x :rents ?a)  (?a rdf:type :Truck)
                 (?y :buys  ?b) (?b rdf:type :Fertilizer)’,
                SDO_RDF_Models(‘Activities'),
        …)) t3
WHERE t1.name=t3.x and t2.name=t3.y and
         t1.addr=t2.addr;
```

RDF Rulebases

Rulebase: Overview

- Each RDF rulebase consists of a set of rules
- Each rule consists of
  - antecedent: graph-pattern
  - filter condition (optional)
  - Consequent: graph-pattern
- One or more rulebases may be used with relevant RDF models (graphs) to obtain entailed graphs

Rulebase: Example

Rules in a rulebase family_rb:

Antecedent: **'(?x :brotherOf ?y) (?y :parentOf ?z)'**
Filter: **NULL**
Consequent: **'(?x :uncleOf ?z)'**

Antecedent: **'(?x :age ?a)'**
Filter: **'a >= 65'**
Consequent: **'(?x :ageGroup "Senior")'**

Antecedent: **'(?x :parentOf ?y) (?y :parentOf ?z)'**
Filter: **NULL**
Consequent: **'(?x :grandParentOf ?z)'**

RDF Rule Indexes

Rule Index: Overview

- A rule index represents an entailed graph
- A rule index is created on an RDF dataset (consisting of a set of RDF models and a set of RDF rulebases)

Rule Index: Example

- A rule index may be created on a dataset consisting of
  - family RDF data, and
  - family_rb rulebase (shown earlier)
- The rule index will contain inferred triples showing uncleOf and ageGroup information

RDF Query with Inference

SDO_RDF_MATCH with Rulebases

- Arguments
  - Graph pattern
    - A sequence of triples (with variables)
  - RDF Data set
    - a set of models
    - a set of rulebases
  - Filter
  - Aliases
- ...

```
FROM TABLE(SDO_RDF_MATCH(
            '(?x :uncleOf ?y)',
            SDO_RDF_Models('family'),
            SDO_RDF_Rulebases ('rdfs', 'family_rb')
            ...
        )) t
...
```

RDF Query w/ Inference: Example

- Find salary and hiredate of all the uncles
-

```
SELECT emp.name, emp.salary, emp.hiredate
FROM emp,
        TABLE(SDO_RDF_MATCH(
                ‘(?x :uncleOf ?y) (?x :name ?name)’,
                SDO_RDF_Models(‘family'),
                SDO_RDF_Rulebases(‘rdfs’, ‘family_rb'),
        …)) t
WHERE emp.name=t.name;
```

RDF Query w/ Inference: Example 2

- Find pairs of persons residing at the same address where the first person rents a truck and the second person buys a fertilizer
-
```
SELECT t3.x name1, t3.y name2
FROM AddrTable t1, AddrTable t2,
        TABLE(SDO_RDF_MATCH(
                ‘(?x :rents ?a)  (?a rdf:type :Truck)
                 (?y :buys  ?b) (?b rdf:type :Fertilizer)’,
                SDO_RDF_Models(‘Activities'),
                SDO_RDF_Rulebases(‘rdfs’),
        …)) t3
WHERE t1.name=t3.x and t2.name=t3.y and
         t1.addr=t2.addr;
```

RDF Models

Model: DDL

- Procedures provided as part of the API may be used to
  - Create a model
  - Drop a model
- When a user creates a model, a database view gets created automatically
  - **rdfm_family**
- A model corresponds to a column of type **SDO_RDF_TRIPLE_S** in a base table
- Each model has exactly one base table associated with it

Model: DDL → Creating a Model

- Create an Application Table

```
CREATE TABLE family_table (
   id NUMBER, family_triple SDO_RDF_TRIPLE_S);
```

- Create a Model

```
EXEC SDO_RDF.CREATE_RDF_MODEL(
   'family', 'family_table','family_triple');
```

- Automatically creates the following database view

rdfm_family (…)

Loading RDF Data into Oracle

- Java API provided to load NTriple into NDM
- Sample XSLs provided
  - To convert RDF to NTriple
  - To convert RDF to INSERT statements

Model: DML

- SQL DML commands may be used to do DML operations on a base table to effect DML (i.e., triple insert, delete, and update) on the corresponding model

- Insert Triples

```
INSERT INTO family_table VALUES (1,
                SDO_RDF_TRIPLE_S(‘family',
                        '<http://example.org/family/John>',
                        '<http://example.org/family/brotherOf>',
                        ‘<http://example.org/family/Mary>'));
```

Model: Security

- The creator of the base table corresponding to a model can grant privileges to other users
- To perform DML to a model, a user must have DML privileges for the corresponding base table
- The creator of a model can grant QUERY privileges on the corresponding database view to other users
- A user can query only those models for which s/he has QUERY privileges to the corr. database views
- Only the creator of a model can drop the model

Model: Views

- Database views corresponding to the models

RDF Rulebases

Rulebase: DDL

- Procedures provided as part of the API may be used to
  - Create a rulebase
    create_rulebase('family_rb');
  - Drop a rulebase
  - **drop_rulebase('family_rb');**
- When a user creates a rulebase, a database view gets created automatically
  - **rdfr_family_rb (rule_name, antecedent, filter, consequent, aliases)**

Rulebase: DML

- SQL DML commands may be used on the database view corresponding to a target rulebase to insert, delete, and update rules
- **insert into mdsys.rdfr_family_rb values(**
  ‘uncle_rule',
  ‘(?x :brotherOf ?y) (?y :parentOf ?z)’,
  NULL,
  '(?x :uncleOf ?z)',
  SDO_RDF_Aliases(…));

Rulebase: Security

- Creator of a rulebase can grant privileges to the corresponding database view to other users
- Performing DML operations requires invoker to have appropriate privileges on the database view
- Only the creator of a rulebase can drop the rulebase

Rulebase: Views

- **RDF_RULEBASE_INFO**
  - Contains the list of rulebases
  - For each rulebase, contains additional information (such as, creator, view name, etc)
- Content of each rulebase is available from the corresponding database view

RDF Rule Indexes

Rule Index: DDL

- Procedures provided as part of the API may be used to
  - Create a rule index
    create_rules_index ('family_rb_rix_family‘,
    SDO_RDF_Models('family'),
    SDO_RDF_Rulebases(‘rdfs','family_rb'));
  - Drop a rule index
    drop_rules_index ('family_rb_rix_family');
- When a user creates a rule index, a database view gets created automatically
  - **rdfi_family_rb_rix_family (…)**

Rule Index: Security

- To create a rule index on an RDF dataset (models and rulebases), user needs to have QUERY privileges on those models and rulebases
- Creator of a rule index holds QUERY privilege on the rule index and may grant this privilege to other users
- Only the creator of a rule index can drop it

Rule Index: Views

- **RDF_RULEINDEX_INFO**
  - Contains the list of rule indexes
  - For each rule index, contains additional information (such as, creator, status, etc)
- **RDF_RULEINDEX_DATASETS**
  - For every rule index, stores the names of its models and rulebases

Rule Index: Dependencies

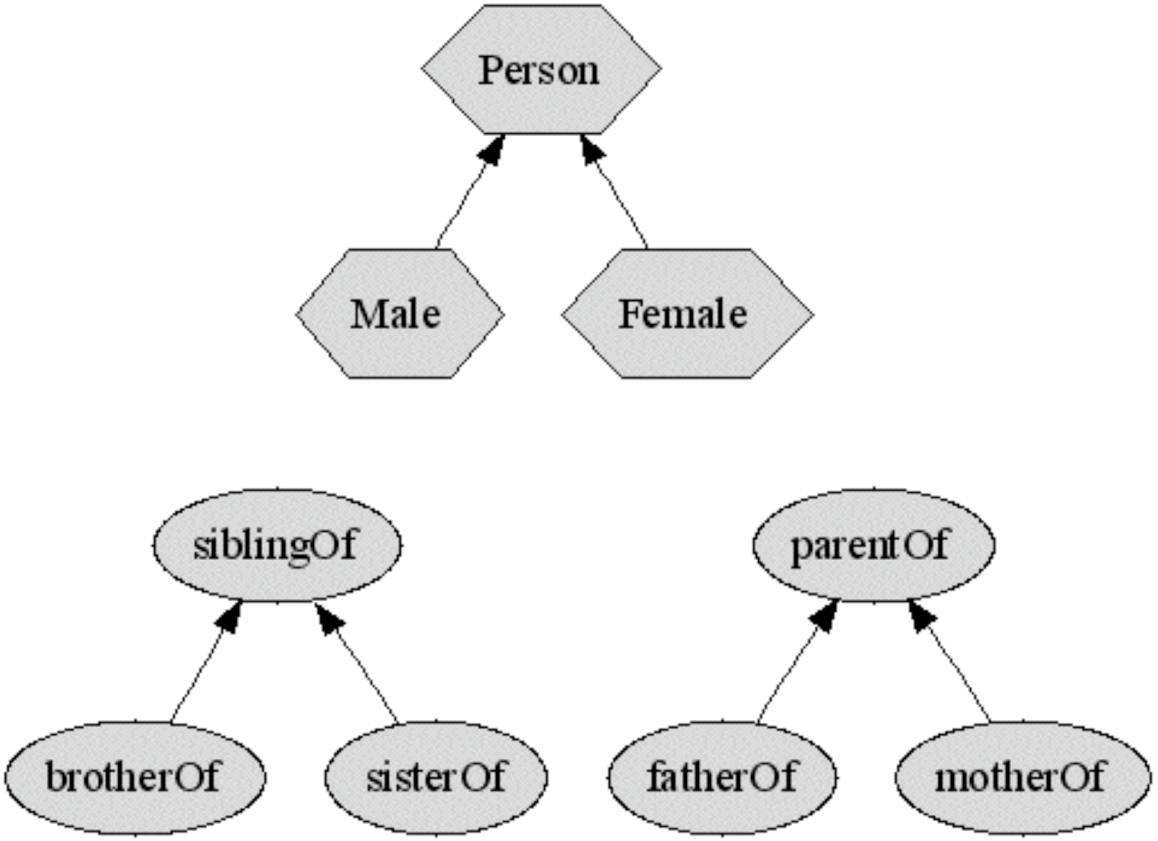
- Content of a rule index depends upon the content of each element of its dataset
  - Any modification to the models or rulebases in its dataset invalidates the rule index
  - Dropping a model or rulebase will drop dependent rule indexes automatically.

Summary

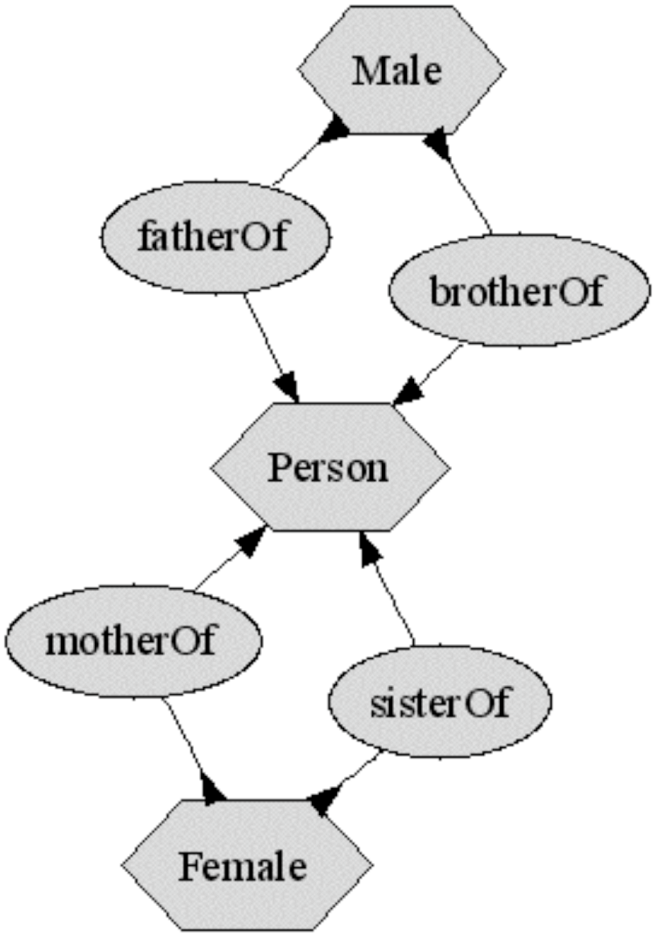
- RDF Data Model
  - Models (Graphs)
  - RDF Query using SDO_RDF_MATCH Table Function
- RDF Data Model with (user-defined) Rules
  - Models (Graphs)
  - Rulebases
  - Rule Indexes
  - RDF Query on entailed RDF graphs
- Management (DDL, DML, Security, …)
  - Models, Rulebases, and Rule Indexes

RDF Data Model Demo

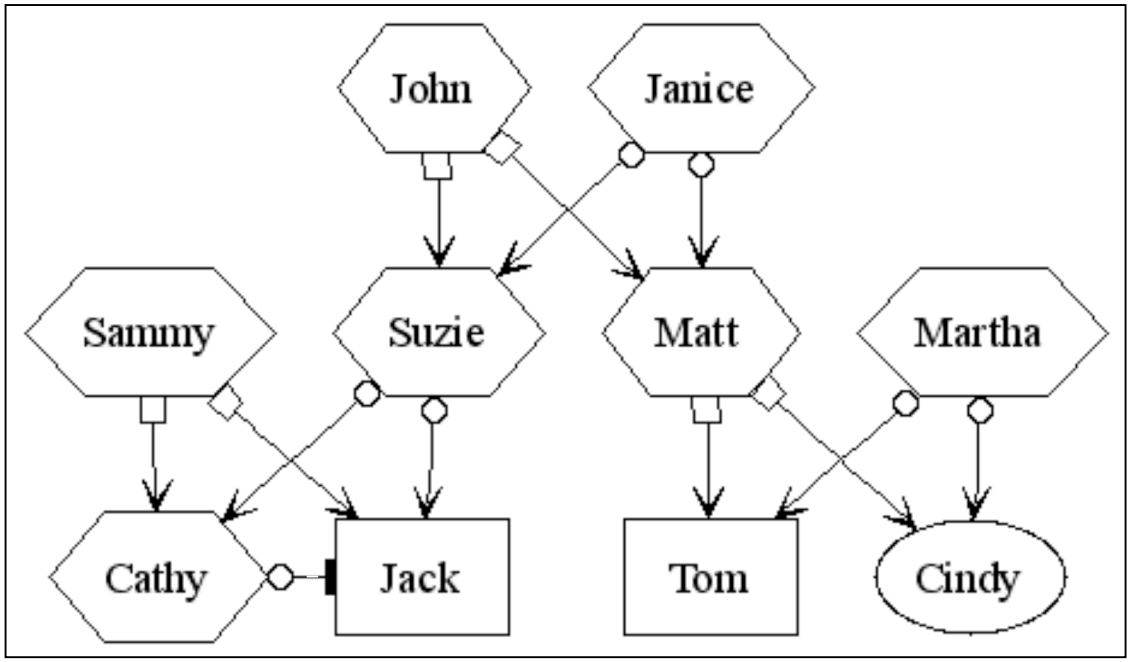
Demo: Family Schema

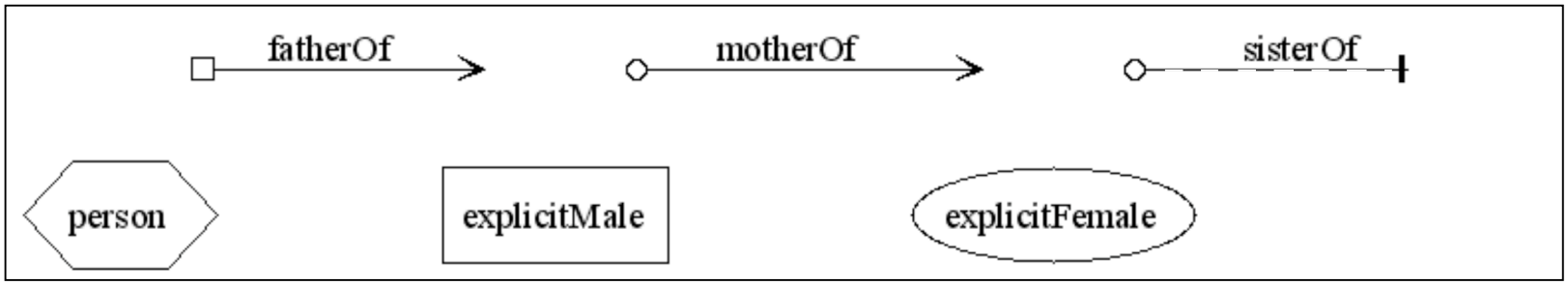


Demo: Family Schema 2

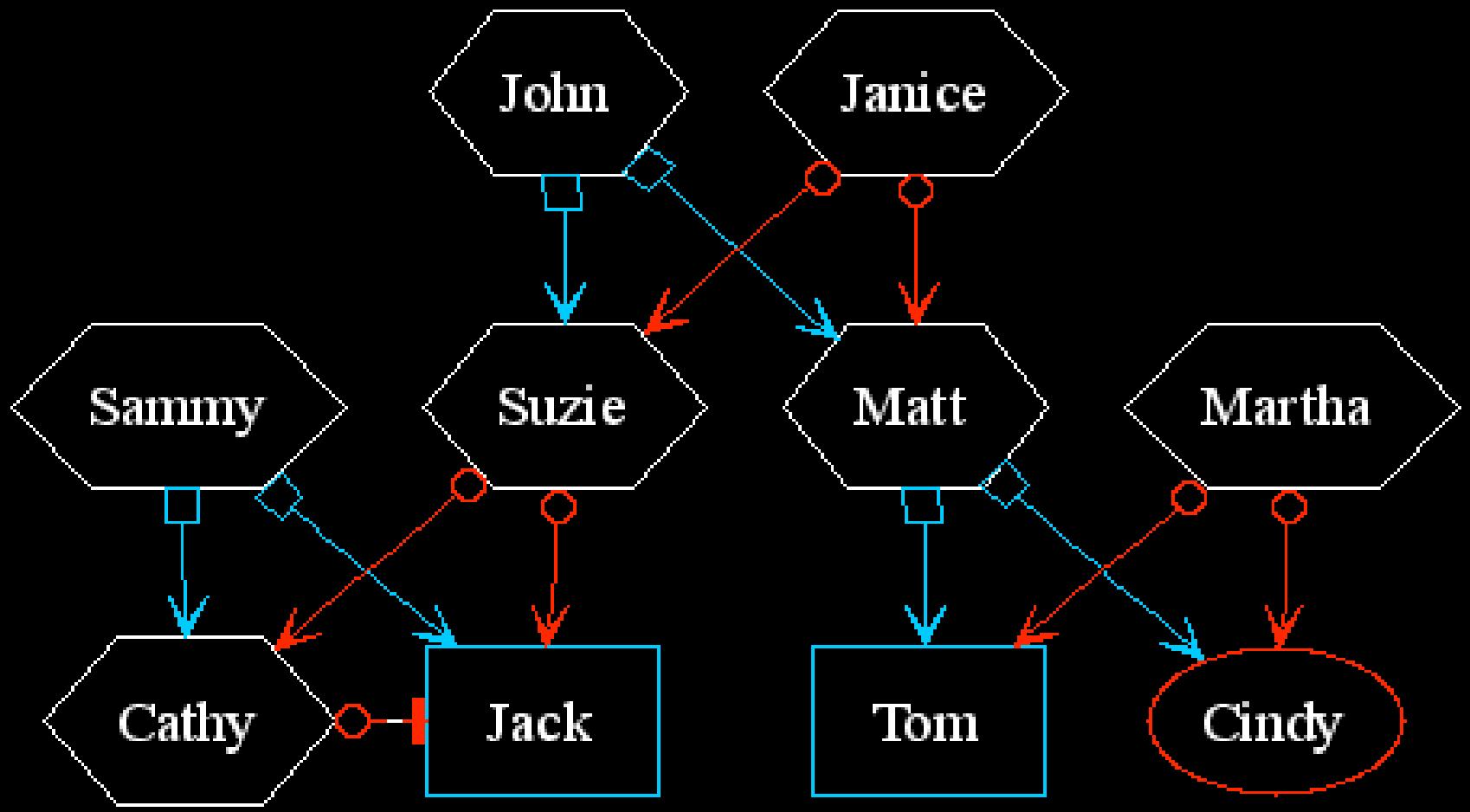


Demo: Family Model Data





Demo: Family Model Data (Alt)



Demo: Query without Inference

```
select m from TABLE(SDO_RDF_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
```

Demo: Query w/ RDFS Inference

```
select m from TABLE(SDO_RDF_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/Matt
http://www.example.org/family/Sammy
```

Demo: Family Rulebase

Antecedent: **‘(?x :parentOf ?y) (?y :parentOf ?z)’**

Filter: **NULL**

Consequent: **‘(?x :grandParentOf ?z)’**

Demo: Query w/ Family and RDFS Inference

```
select x, y from TABLE(SDO_RDF_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));
```

X	Y
http://www.example.org/family/John	http://www.example.org/family/Cindy
http://www.example.org/family/John	http://www.example.org/family/Tom
http://www.example.org/family/John	http://www.example.org/family/Jack
http://www.example.org/family/John	http://www.example.org/family/Cathy