Introduction to RDF Graph for Oracle Database 19c

Architecture and Overview

Matthew Perry
CMTS
Oracle Graph Technologies
October 2019
Safe harbor statement

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 decisions.

The development, release, timing, and pricing of any features or functionality described for Oracle's products may change and remains at the sole discretion of Oracle Corporation.
Agenda

1. Introduction to RDF Graph
2. Core Database Features
3. Mid-tier Components and Tools
4. Performing Analytics with RDF Graph Data
5. New Features in Recent Releases
Agenda

1. Introduction to RDF Graph
2. Core Database Features
3. Mid-tier Components and Tools
4. Performing Analytics with RDF Knowledge Graph Data
5. New Features in Recent Releases
A Little About our Team

- First release of RDF Knowledge Graph was Oracle 10g R2 (2005)
- Active in standards development – team members have been …
  - Member of W3C SPARQL 1.1 WG
  - Member of W3C RDF 1.1 WG
  - Co-editor of W3C OWL 2 Web Ontology Language Profiles spec.
  - Co-editor of W3C R2RML: RDB to RDF Mapping Language spec.
  - Co-editor of OGC GeoSPARQL spec.
- Several research papers in this area
  - VLDB, ICDE, EDBT, ISWC, etc.
Oracle Supports Two Graph Data Models

**Linked Data Knowledge Graph**
- Data Federation
- Knowledge Representation
- Metadata Management

**Property Graph**
- Path Analytics
- Social Network Analysis
- Entity Analytics

**Use Case**
- Life Sciences
- Health Care
- Publishing
- Finance

**Industry Domain**
- Financial
- Retail, Marketing
- Social Media
- Smart Manufacturing

**Graph Analytics**
Database Version for RDF Knowledge Graph

- **On-premise**
  - Oracle Database Enterprise and Standard Edition
  - Oracle Database Cloud Services
  - DBCS (Bare Metal & VM)
  - Exadata Cloud Service
  - Exadata Cloud at Customer
  - ATP-Dedicated (planned)

https://blogs.oracle.com/oraclespatial has a 3-part series on getting started with RDF on DBCS

“Using RDF Knowledge Graphs in the Oracle Public Cloud”
RDF Graph for Oracle Database 19c Features

**Load / Storage**
- Fast bulk-load and indexing
- RDF view of Relational data
- Manages over a trillion quads

**Query**
- SPARQL-Jena/Fuseki
- SPARQL-in-SQL query & update
- Federated query
- GeoSPARQL

**Reasoning**
- RDFS, OWL2 RL, EL, SKOS
- Ladder-based inference
- Incremental, parallel reasoning
- User-defined rules
- Plug-in architecture

**Analytics**
- OBIEE
- Oracle Advanced Analytics
- PGX

RDF Knowledge Graph
Leverages Oracle capabilities:
- RAC & Exadata scalability
- Compression & partitioning
- In-Memory Column Store
- SQL*Loader direct path load
- Parallel DDL, DML, and query
- RMAN Backup and Recovery
- High Availability
- Oracle Label security
- Enterprise Manager
- Logical Standby
RDF Graph for Oracle Database 19c Architecture

- Protégé Plugin
- Fuseki Endpoint
- Cytoscape Plugin
- SQL Developer RDF Support
- Enterprise Manager and Other DB Tools

Support for Apache Jena (Java API)

SQL and PL/SQL API
- RDF Bulk Loader
- Forward-chaining OWL Reasoner
- SPARQL-to-SQL Query Translator
- SPARQL Update Processor

Generic Relational Schema for Storing RDF Data

RDF Views of Relational Data
Agenda

1. Introduction to RDF Graph
2. Core Database Features
3. Mid-tier Components and Tools
4. Performing Analytics with RDF Graph Data
5. New Features in Recent Releases
RDF Bulk Loader

Client

- Oracle SQL*Loader
- Oracle Protégé Plugin
- Jena Adapter Java API
- orardfldr

DB Server

Step 1
- Staging Table
- External Table

Step 2
- Parallel RDF Bulk Load
  - Value Checking
  - Canonicalization
  - Duplicate Elimination
  - Internal ID Generation
  - Index Creation

Parallel RDF Relational Schema

Export Formats
- N-Triple
- N-Quad
- Trig
- Turtle
- RDF/XML
- JSON-LD
- Others

External Table
- Oracle
- Protégé Plugin
- Jena Adapter Java API
- orardfldr

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.
Core Inference Features in Oracle Database

- Oracle provides native inference in the database for
  - RDFS, RDFS++
  - OWLPRIME, OWL2RL, OWL2EL, SKOSCORE
  - User-defined rules
  - User-defined inferencing

- Inference done using forward chaining
  - Triples inferred and stored ahead of query time
  - Removes on-the-fly reasoning and results in fast query times

- Proof generation
  - Shows one deduction path
Native Inference Engine in Oracle: APIs

**SEM_APIS.CREATE_ENTAILMENT**

```
entailment_name
sem_models(‘GraphTBox’, ‘GraphABox’, …),
sem_rulebases(‘OWL2RL’),
passes,
inf_components,
Options,…
```

Use “PROOF=T” to generate inference proof

**SEM_APIS.VALIDATE_ENTAILMENT**

```
sem_models(‘GraphTBox’, ‘GraphABox’, …),
sem_rulebases(‘OWL2RL’),
Criteria,
Max_conflicts,
Options
```

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

Java API: `performInference`, `deleteInference`, `setInferenceOption`, `analyze` methods in
- `GraphOracleSem`, `DatasetGraphOracleSem` (Jena Adapter)

Typical Usage:
- First load RDF/OWL data
- Call `create_entailment` to generate inferred graph
- Call `validate_entailment` to find inconsistencies
Enabling Advanced Inference Capabilities

- **Parallel inference option**
  - EXECUTE sem_apis.create_entailment('M_IDX', sem_models('M'),
    sem_rulebases('OWL2RL'), null, null, 'DOP=x');
  - Where ‘x’ is the degree of parallelism (DOP)

- **Incremental inference option**
  - EXECUTE sem_apis.create_entailment('M_IDX', sem_models('M'),
    sem_rulebases('OWL2RL'), null, null, 'INC=T');

- **Enabling owl:sameAs option to limit duplicates**
  - EXECUTE sem_apis.create_entailment('M_IDX', sem_models('M'),
    sem_rulebases('OWL2RL'), null, null, 'OPT_SAMEAS=T');

- **Control of individual inference rules used**
  - EXECUTE sem_apis.create_entailment('M_IDX', sem_models('M'),
    sem_rulebases('OWL2RL'), null, 'SAM-, SCOH-', null, null);
  - For example, disable sameAs generation for individuals and disable subClassOf hierarchy
SPARQL Query and Update
What is SPARQL?

- SPARQL Protocol and RDF Query Language
  - W3C standard for querying and manipulating RDF content
  - Queries/updates and corresponding results are communicated via HTTP with a SPARQL endpoint
  - A SPARQL endpoint implements the SPARQL protocol and serves RDF data from a RDF triplestore or RDF view
SPARQL Graph Pattern

- Basic unit of SPARQL queries

Result 1: {\(t=\text{univ:Student}, \ p=\text{univ:student123}, \ n=\text{"John Green"}, \ g=\text{"male"}, \ b=\text{"1999-06-15"^^xsd:date}\)}

Result 2: {\(t=\text{univ:Student}, \ p=\text{univ:student456}, \ n=\text{"Susan Blue"}, \ g=\text{"female"}, \ b=\text{"2000-02-10"^^xsd:date}\)}
SPARQL Graph Pattern

- Basic unit of SPARQL queries

How do we express this with SPARQL?

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX vcard: <http://www.w3.org/2001/vcard-rdf/3.0#>

SELECT ?t ?n ?b ?g
WHERE
  ?p foaf:gender ?g }
```

Basic Graph Pattern (BGP)
SPARQL and SPARQL in SQL Architecture

HTTP

- Standard SPARQL Endpoint
  - Enhanced with query management control

Java

- Support for Apache Jena

Apache Jena

SQL

- SEM_MATCH
- SEM_APIS.UPDATE_MODEL
- SEM_APIS.SPARQL_TO_SQL

SPARQL-to-SQL Core Logic
SEM_MATCH: Adding SPARQL to SQL

- Extends SQL with full SPARQL 1.1 query constructs
- Benefits:
  - Integrates graph data with existing enterprise data
  - JOINs with other relational (and object-relational) data
  - Allows SQL constructs/functions
  - DDL Statements: create tables/views
  - Allows use of enterprise SQL development tools
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**

**SELECT** n1, n2  
**FROM**  
**TABLE (**

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

**SQL Table Function**
SPARQL 1.1 Update

- SPARQL 1.1 Update is intended to be a standard language for specifying and executing updates to RDF graphs in a Graph Store.

<table>
<thead>
<tr>
<th>Graph Update operations</th>
<th>Graph Management operations</th>
</tr>
</thead>
<tbody>
<tr>
<td>INSERT DATA Adds some triples, given inline in the request, into a graph.</td>
<td>CREATE Creates a new graph in stores that support empty graphs.</td>
</tr>
<tr>
<td>DELETE DATA Removes some triples, given inline in the request, if the respective graph contains those.</td>
<td>DROP Removes a graph and all of its contents.</td>
</tr>
<tr>
<td>DELETE/INSERT Perform pattern-based actions for graph updates</td>
<td>COPY Modifies a graph to contain a copy of another.</td>
</tr>
<tr>
<td>LOAD Reads the contents of a document representing a graph into a graph in the Graph Store.</td>
<td>MOVE Moves all of the data from one graph into another.</td>
</tr>
<tr>
<td>CLEAR Removes all the triples in (one or more) graphs.</td>
<td>ADD Reproduces all data from one graph into another.</td>
</tr>
</tbody>
</table>
Top-level Procedure:
SEM_APIs.UPDATE_MODEL

begin
    sem_apis.update_model(
        apply_model=>'M1',
        update_stmt=>'INSERT {?s :mbox ?n}
                      WHERE {?s :email ?n}'
    );
end;
/
W3C RDB2RDF – RDF Views

- Use virtual RDF data
- Benefits
  - Existing relational data stays in place and corresponding applications do not need to change
  - Use of virtual mapping eliminates synchronization issues
  - Common vocabulary helps with data integration issues
Relational Data to RDF (W3C RDB2RDF)

- Two types of mapping: Direct and R2RML

- Direct Mapping
  - Automatically generates a mapping based on an input relational schema

- R2RML (RDB to RDF Mapping Language)
  - Language for expressing customized mappings
Tools Make Things Easier

SQL Developer RDF Plugin

Jena Fuseki Web Application
Agenda

1. Introduction to RDF Graph
2. Core Database Features
3. Mid-tier Components and Tools
4. Performing Analytics with RDF Graph Data
5. New Features in Recent Releases
Support for Apache Jena
Support for Apache Jena

- Implements Apache Jena APIs
  - Popular Java APIs for semantic web based applications
  - Adds Oracle-specific extensions

- Jena support provides three core features:
  - Java APIs for RDF Semantic Graph
    - Data loading, querying, inferencing
  - SPARQL Endpoint for Oracle with SPARQL 1.1 support
  - Oracle-specific extensions for query execution control and management
Provide Convenient Java API for RDF Knowledge Graph

- “Proxy” like design
  - Data not cached in memory for scalability
  - SPARQL query converted into SQL and executed inside DB
  - Inference pushed down to the database for execution

- Various data loading methods
  - Bulk/Incremental load RDF or OWL (stored in FS/HDFS/Website) in a variety of formats with strict syntax verification and long literal support
    - N3
    - Turtle
    - RDF/XML
    - N-TRIPLE
    - N-Quads
    - TriG
Support for Protégé 5
RDF Graph Support for Protégé 5

- Two key goals
  - Integrate all the powerful ontology editing capabilities of Protégé
  - Use the same GUI as a unified interface to manage semantic data stored in Oracle Database
Key Features for Protégé Support

- Incremental Edit Saving
- Advanced Bulk Loading
- XML Catalog – Ontology Libraries
  - Support `owl:import` of ontologies stored in Oracle Database
- Models Manager
- Entailments Manager
- SPARQL Query Worksheet
- SQL Query Worksheet
Oracle SQL Developer RDF Support
Oracle SQL Developer RDF Support

- Since SQL Dev. Version 18.1
- Primary Goals
  - Serve as the primary GUI for RDF Knowledge Graph SQL and PL/SQL APIs
  - Single GUI for both Knowledge Graph and general database operations
  - Provide a familiar interface to Oracle Database users
  - Provide a friendly interface for Sem Web experts
Oracle SQL Developer RDF Support

- Expand RDF Semantic Graph to see various menu items
  - Models, Rulebases, Entailments, …
- Right-Click for operations
- Forms to build up argument lists for PL/SQL APIs
Oracle SQL Developer RDF Support

- **SPARQL Editor**
  - Opens after selecting a model, virtual model, or RDF view
  - Provides a pure-SPARQL query interface
  - Several query templates are available (e.g., distinct predicates)
  - Syntax highlighting and auto-completion
Oracle SQL Developer RDF Support

- R2RML Mapping Editor
  - Shows a tree view of an R2RML mapping
  - Allows drill down from Triples Map -> Subject Map -> Predicate-Object Map
- You can edit or delete existing elements or add new ones
- “Commit Mapping” refreshes the associated RDF view
Agenda

1. Introduction to RDF Graph
2. Core Database Features
3. Mid-tier Components and Tools
4. Performing Analytics with RDF Graph Data
5. New Features in Recent Releases
Oracle RDF KG as an Integration / Analytics Platform

Oracle Big Data Connectors
Oracle Database Gateways

Property Graph View of RDF
External Table / DB Link

CREATE VIEW SEM_MATCH
RDF View

Relational Table/View
Oracle Advanced Analytics

PGX

Property Graph View
OBIEE

My SQL
DB2
SQL Server
public static void createPropertyGraphViewOnRDF(Connection conn,
String pgGraphName,
String rdfModelName,
boolean virtualModel,
PGUtils.RDFPredicate[] predListForVertexAttrs,
PGUtils.RDFPredicate[] predListForEdges)

This method creates the property graph views (vertex and edge views) for the given RDF model.

Parameters:
- conn - a connection instance to Oracle database
- pgGraphName - the name of the property graph to be created
- rdfModelName - the name of the RDF model
- virtualModel - a flag represents if the RDF model is virtual model or not
- predListForVertexAttrs - an array of RDFPredicate objects specifying how to create vertex view using these predicates
- predListForEdges - an array of RDFPredicate specifying how to create edge view using these predicates
Agenda

1. Introduction to RDF Graph
2. Core Database Features
3. Mid-tier Components and Tools
4. Performing Analytics with RDF Graph Data
5. New Features in Recent Releases
12c Release 2 New Features

- Usability
  - SQL Developer RDF support
- Standards support
  - In-database SPARQL Update support
- Improved performance
  - Schema changes for more efficient number, timestamp, spatial geometry processing
  - More efficient RDF View query execution
  - Support for in-memory columnar representation of RDF data
- Integration
  - RDF views of Property Graph data / Property Graph views of RDF data
18c New Features

- **Usability**
  - Enhanced CLOB support in bulk load
    - CLOB loading allowed in `sem_apis.bulk_load_from_staging_table`
    - External Table based bulk-load now supports VCONLY, CLOBONLY, and VIEW options
  - Support for Turtle and Trig formats in PL/SQL APIs

- **Improved performance**
  - Composite partitioning of semantic networks
  - Support for in-memory virtual columns to eliminate value table joins
  - Hybrid strategy for SPARQL OPTIONAL evaluation
  - Better optimizer statistics with column groups and larger histograms
  - Compile-time value substitution for BIND
19c New Features

- Usability, Security, and Availability
  - Support for schema-private semantic networks
    - Stores RDF data in user schema instead of MDSYS
    - Allows multiple semantic networks in the same database
    - Enables compatibility with Oracle Database Vault (a requirement for Autonomous DB)
  - Logical standby for RDF

- Tool Support
  - Bring open-source based tools up to latest version
  - ORARDFFLDR command-line utility