Developing GeoSPARQL Applications with Oracle Spatial and Graph

Matthew Perry, Ana Estrada, Souripriya Das, Jayanta Banerjee

Oracle, One Oracle Drive, Nashua, NH 03062, USA

Abstract. Oracle Spatial and Graph – RDF Semantic Graph, an option for Oracle Database 12c, is the only mainstream commercial RDF triplestore that supports the OGC GeoSPARQL standard for spatial data on the Semantic Web. This demonstration will give an overview of the GeoSPARQL implementation in Oracle Spatial and Graph and will show how to load, index and query spatial RDF data. In addition, this demonstration will discuss complimentary tools such as Oracle Map Viewer.

1 Introduction

The Open Geospatial Consortium (OGC) published the GeoSPARQL standard in 2012 [1]. This standard defines, among other things, a vocabulary for representing geospatial data in RDF and an extension of the W3C SPARQL query language [4] that allows spatial processing.

Oracle Database has provided support for RDF data with the Spatial and Graph option since 2005. GeoSPARQL has been supported since 2013, beginning with version 12c Release 1 [2]. Oracle Spatial and Graph is the only mainstream commercial RDF triplestore that supports significant components of the OGC GeoSPARQL standard. A demonstration of Oracle's technology will give an overview of the current state-of-the-art for managing linked geospatial data.

The remainder of this paper describes major components of OGC GeoSPARQL and then discusses the architecture of Oracle Spatial and Graph. This is followed by a description of key features of Oracle Spatial and Graph that will be demonstrated.

2 The OGC GeoSPARQL Standard

The OGC GeoSPARQL standard defines a number of components ranging from top-level OWL classes to a set of rules for query transformations. Two of the most fundamental components of the standard are (1) RDFS datatypes for serializing spatial geometries and (2) a set of SPARQL extension functions for spatial computations.

OGC GeoSPARQL defines two RDFS datatypes for serializing spatial geometries: `ogc:wktLiteral`[1] and `ogc:gmlLiteral`. These serializations are based on existing

[1] The namespace prefix ogc expands to <http://www.opengis.net/ont/geosparql#>

adfa, p. 1, 2011.

and widely supported OGC standards, which allow use of existing GIS tools for processing. A sample `ogc:wktLiteral` is shown below.

```
"<http://www.opengis.net/def/crs/OGC/1.3/CRS84>
 POINT(-83.38 33.95)"^^ogc:wktLiteral
```

The `ogc:wktLiteral` serialization uses an OGC WKT string describing a geometry value. The WKT string is optionally prefixed with a coordinate reference system URI that identifies the CRS used for the geometry. If an explicit CRS URI is not provided, then `<http://www.opengis.net/def/crs/OGC/1.3/CRS84>` (WGS84 long-lat) is assumed as the CRS.

OGC GeoSPARQL includes several SPARQL extension functions. These include both non-topological functions (distance, buffer, intersection, etc.) and topological functions to test spatial relations based on the Dimensionally Extended Nine Intersection Model (DE-9IM). A general purpose `ogcf:relate`[2] function is defined for testing DE-9IM intersection patterns consisting of true/false values, and convenience functions for testing common named topological relations (e.g., `ogcf:sfWithin`) are available. An example GeoSPARQL query is shown below. This query finds `lgd:Monuments` that are within a query window.

```
PREFIX geovocab: <http://geovocab.org/geometry#>
PREFIX lgd:  <http://linkedgeodata.org/ontology/>
PREFIX ogc:  <http://www.opengis.net/ont/geosparql#>
PREFIX ogcf: <http://www.opengis.net/def/function/geosparql/>
SELECT ?s ?l ?wkt
WHERE { ?s rdf:type lgd:Monument . ?s rdfs:label ?l .
        ?s geovocab:geometry ?geom . ?geom ogc:asWKT ?wkt .
        FILTER(ogcf:sfWithin(?wkt,
           "POLYGON((-71.44 42.50, -71.42 42.40, -71.08 42.39,
                     -71.03 42.56, -71.44 42.50))"^^ogc:wktLiteral))}
```

3 Support for GeoSPARQL in Oracle Spatial and Graph

Oracle Spatial and Graph supports the following conformance classes for the OGC GeoSPARQL standard using well-known text serialization and the Simple Features relation family.

- Core
- Topology Vocabulary Extension (Simple Features)
- Geometry Extension (WKT, 1.2.0)
- Geometry Topology Extension (Simple Features, WKT, 1.2.0)
- RDFS Entailment Extension (Simple Features, WKT, 1.2.0)

[2] The namespace prefix ogcf expands to <http://www.opengis.net/def/function/geosparql/>

Oracle Spatial and Graph uses a relational schema to store RDF data and processes SPARQL queries by translating them to equivalent SQL queries that are executed by the parallel SQL engine in Oracle Database. SPARQL queries can be executed through SQL, Java and HTTP interfaces. Core SPARQL-to-SQL translation logic runs on the database server, and this logic can be invoked from the SEM_MATCH SQL table function, which provides a SQL interface for SPARQL query execution, or from an adapter for Apache Jena[3], which provides a Java programming framework and a standard-compliant HTTP SPARQL endpoint. Apache Jena typically runs on a mid-tier server.

A simplified version of Oracle's relational schema for RDF data consists of an `RDF_LINK$` table and an `RDF_VALUE$` table. The `RDF_VALUE$` table stores an `ID` to lexical value mapping for RDF terms, and the `RDF_LINK$` table stores 4-tuples of `ID`s representing `subject(S)`, `predicate(P)`, `object(O)`, and `graph(G)`. This basic schema is shown in Fig. 1.

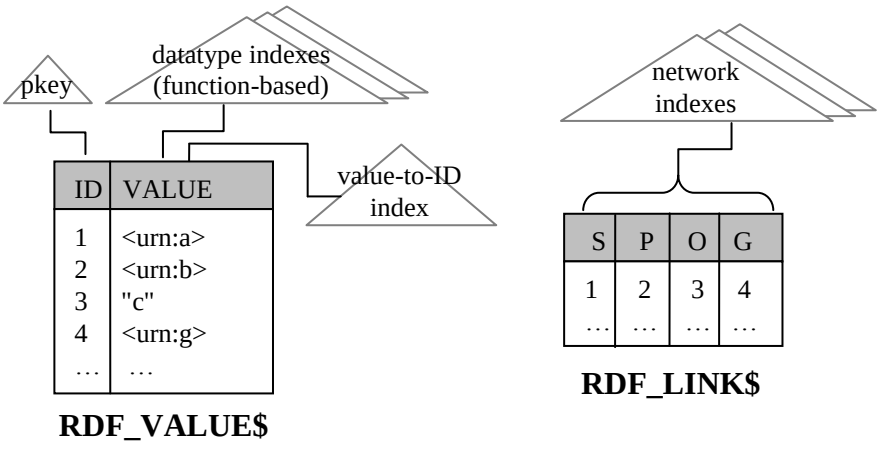

Fig. 1. Relational Schema for RDF Data

The SQL translation of a SPARQL query consists of various operations against the `RDF_LINK$` table to evaluate structural aspects of the SPARQL query pattern (`INNER JOIN` for conjunction, `UNION ALL` for disjunction, `OUTER JOIN` for SPARQL `OPTIONAL`, etc.), and joins with the `RDF_VALUE$` table are used to retrieve lexical values that need to be returned from the query or are needed to evaluate a SPARQL `FILTER` expression. Constant values in SPARQL triple patterns are resolved to `ID`s at query compilation time to minimize joins with the `RDF_VALUE$` table in the final SQL translation.

Several indexes are used to speed up query processing. These indexes can be customized through a management API. Multi-column indexes for various permutations of `SPOG` (referred to as network indexes) can be created on `RDF_LINK$`. Three to four

[3] https://jena.apache.org/

such indexes will typically cover most access patterns and allow evaluation of operations on RDF_LINK$ with only index scans. The RDF_VALUE$ table includes a primary key index on the ID column for fast lookups of lexical values. Function-based indexes on the VALUE column of RDF_VALUE$ (referred to as datatype indexes) are used for fast evaluation of SPARQL FILTER expressions. The SQL translation of a SPARQL FILTER expression uses PL/SQL functions to convert VARCHAR lexical values into native Oracle types. For example, the SPARQL expression FILTER(?x < 10) would translate to getNumberValue(VALUE) < 10. In this case, getNumberValue returns an Oracle NUMBER for all numeric literals and NULL for all other values. A function-based index on getNumberValue will allow an index-based evaluation of FILTER(?x < 10). Analogous functions exist for each datatype supported by SPARQL, and a convenient API is provided for managing datatype indexes, including full-text and spatial indexes.

Oracle Spatial and Graph provides an SDO_GEOMETRY SQL object type, an R-Tree index type, and several SQL functions that perform various spatial computations [3]. These SQL capabilities are used to provide GeoSPARQL query support. ogc:wktLiteral values are stored as VARCHARs (CLOBs in some cases) in RDF_VALUE$, and a getGeometryValue(value, target_crs_id) function is used to convert ogc:wktLiterals to SDO_GEOMETRY objects. As an example, consider the following GeoSPARQL FILTER.

```
FILTER(ogcf:sfWithin(?wkt,
   "POLYGON((-71.44 42.50, -71.42 42.40, -71.08 42.39,
             -71.03 42.56, -71.44 42.50))"^^ogc:wktLiteral))
```

This FILTER would translate to the following SQL expression.

```
SDO_RELATE(
  getGeometryValue(V1.VALUE, 8307),
  getGeometryValue("POLYGON((-71.44 42.50, -71.42 42.40, -71.08 42.39,
                             -71.03 42.56, -71.44 42.50))", 8307),
  'MASK=INSIDE+COVEREDBY+ON')='TRUE'
```

A function-based R-Tree index on RDF_VALUE$ for getGeometryValue will allow an index-based evaluation of spatial FILTERs. The CRS used for computation is set at index time. Any number of different CRS values could be used within ogc:wktLiteral values stored in RDF_VALUE$, but these are normalized to a common CRS, which is specified during index creation. That is, getGeometryValue does any necessary transformations and returns SDO_GEOMETRY objects in the target CRS, which is indicated by the target_crs_id parameter.

As concrete example, consider the SQL execution plan shown in Fig. 2. This is the execution plan for the GeoSPARQL query shown in Section 2. The execution starts with an index operation on the function-based R-Tree index to find IDs from RDF_VALUE$ for ogc:wktLiterals that satisfy the FILTER condition. These IDs are then used to drive a series of index-based nested loop joins to evaluate the triple pattern conjunctions. Finally, lexical values for the remaining projected variables (?s

and ?l) are retrieved from RDF_VALUE$. The lexical value for ?wkt was retrieved in the first step.

```
-------------------------------------------------------------------------
| Id  | Operation                                | Name                 |
-------------------------------------------------------------------------
|   0 | SELECT STATEMENT                         |                      |
|   1 |  NESTED LOOPS                            |                      |
|   2 |   NESTED LOOPS                           |                      |
|   3 |    VIEW                                  |                      |
|   4 |     NESTED LOOPS                         |                      |
|   5 |      NESTED LOOPS                        |                      |
|   6 |       NESTED LOOPS                       |                      |
|   7 |        NESTED LOOPS                      |                      |
|   8 |         TABLE ACCESS BY INDEX ROWID      | RDF_VALUE$           |
|*  9 |          DOMAIN INDEX (SEL: 0.100000 %)| RDF_V$GEO_IDX        |
|  10 |         PARTITION LIST SINGLE            |                      |
|* 11 |          INDEX RANGE SCAN                | RDF_LNK_CPSGM_IDX    |
|  12 |        PARTITION LIST SINGLE             |                      |
|* 13 |         INDEX RANGE SCAN                 | RDF_LNK_CPSGM_IDX    |
|  14 |       PARTITION LIST SINGLE              |                      |
|* 15 |        INDEX RANGE SCAN                  | RDF_LNK_CPSGM_IDX    |
|  16 |      PARTITION LIST SINGLE               |                      |
|* 17 |       INDEX RANGE SCAN                   | RDF_LNK_PSCGM_IDX    |
|  18 |    TABLE ACCESS BY INDEX ROWID           | RDF_VALUE$           |
|* 19 |     INDEX UNIQUE SCAN                    | C_PK_VID             |
|  20 |   TABLE ACCESS BY INDEX ROWID            | RDF_VALUE$           |
|* 21 |    INDEX UNIQUE SCAN                     | C_PK_VID             |
-------------------------------------------------------------------------
```

Fig. 2. GeoSPARQL Query Execution Plan

4 Planned Demonstration

The demonstration will show how to go from a downloaded RDF file containing spatial data serialized with the OGC GeoSPARQL vocabulary to a fully functioning GeoSPARQL-enabled SPARQL endpoint running against Oracle Database 12c with the Spatial and Graph option. The major steps will include (1) efficiently bulk loading spatial RDF data, (2) creating an R-Tree index for efficient GeoSPARQL queries, and (3) configuring a SPARQL-endpoint to execute GeoSPARQL queries against loaded data. In addition, supporting applications like Oracle Map Viewer and Enterprise Manager will be discussed.

References

1. OGC GeoSPARQL - A Geographic Query Language for RDF Data, OGC Standard, document number 11-052r4, 2012.
2. Oracle Spatial and Graph RDF Semantic Graph Developer's Guide, Retrieved June 26, 2015, from https://docs.oracle.com/database/121/RDFRM/toc.htm.
3. Oracle Spatial and Graph Developer's Guide, Retrieved June 26, 2015, from https://docs.oracle.com/database/121/SPATL/toc.htm.
4. SPARQL 1.1 Query Language, W3C Recommendation, Retrieved June 26, 2015, from http://www.w3.org/TR/sparql11-query/.