Realizing the Benefits of Linked Geospatial Data with R2RML and GeoSPARQL

Souripriya Das
Matthew Perry

Oracle Spatial and Graph
January 30, 2017
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, and timing of any features or functionality described for Oracle’s products remains at the sole discretion of Oracle.
Agenda

1. Why Linked Geospatial Data?
2. Review of RDF/OWL/SPARQL
3. RDB2RDF Standard
4. GeoSPARQL Standard
5. Adding GeoSPARQL Support to RDB2RDF
6. Demo
Agenda

1. Why Linked Geospatial Data
2. Review of RDF/OWL/SPARQL
3. RDB2RDF Standard
4. GeoSPARQL Standard
5. Adding GeoSPARQL Support to RDB2RDF
6. Demo
Why Linked Geospatial Data?

• Linked Data is beneficial
  – Web of data rather than documents
  – Based on URIs, links
  – Helps discoverability, accessibility, interoperability, reusability

• Spatial Data is important
  – Almost everything is related to location somehow (especially with ubiquity of GPS-enabled mobile devices).
  – Lots of GIS data is available but hard for non-experts to find and use

• Goal: to make spatial data more discoverable, accessible and interoperable
Traditional Spatial Data Infrastructure (SDI)

- Intended for expert users
- Map/dataset oriented
- Data not accessible by general Web searches
- Depends on metadata catalog, which is often sparsely populated
- Cannot just follow links, usually complex APIs or search interface
- Difficult to understand and use the data (multiple CRS issues, proprietary formats)
Some issues with SDIs could be alleviated with Linked Data principles

- Globally unique, persistent HTTP URIs for Spatial Things
  - Resource-centric instead of map-centric
  - Facilitates linking, discoverability, interoperability
  - Can help make data crawlable

- Linked Data friendly formats like RDF, JSON-LD
  - Supports web linking
  - More processable by Web infrastructure

- Web-friendly Data access
  - SPARQL endpoint
  - REST APIs that return RDF, JSON-LD
One way to get there

- Most traditional SDIs are underpinned by a Relational Database (Oracle Spatial, PostGIS, etc.)
- W3C RDB2RDF standard (enhanced with spatial capabilities) can be particularly useful to expose this data as Linked Data
  - Provides virtual RDF while allowing data to stay in existing relational form (no ETL)
  - R2RML mappings can be used to provide meaningful HTTP URIs
  - SPARQL endpoint can be used to drive many Web APIs
Agenda

1. Introduction and Motivation
2. Review of RDF/OWL/SPARQL
3. RDB2RDF Standard
4. GeoSPARQL Standard
5. Adding GeoSPARQL Support to RDB2RDF
6. Demo
What is RDF?

- A graph data model for (web) resources and their relationships

- The graph can be serialized into
  - RDF/XML, N3, N-TRIPLE, N-Quads ...

- Construction unit: **Triple**
  (or assertion, or fact)

  `<http://foobar> <.../produce> <.../mp3>`

  Subject  Predicate  Object

- **Quads** (named graphs) add context, provenance, identification, etc. to assertions

  `<http://foobar> <:produces> <:mp3> <:ProductGraph>`
Basic Elements of RDF

- **Instances**
  - Examples → :John, :MovieXYZ, :PurchaseOrder432

- **Classes**
  - Class represents a group/category/categorization of instances
  - Examples → :John rdf:type :Student

- **Properties**
  - Linking data together
  - Examples → :Mary :hasBrother :John.
    :Mary :wasBornOn "1980-06-08"^^xsd:date.
    :John :graduatedFrom :CalTech.
Triples Are Easy. But Why?

- Graph modeling is **flexible**
  - Adding and removing edges is simple
- **Standard** based graph representation
  - RDF was defined by W3C. Allows interoperability.
- Computers can **understand the semantics** RDF graphs (triples)
  - Same URI means same resource

```
"CA"

:locatedIn

http://www.foobar.com

http://www.foobar.com/products/mp3

:produce

http://www.oracle.com

:customerOf

http://www.oracle.com/products/RDF

:produce

http://www.oracle.com
```
Triples Are Easy. But Why? (2)

- Graph modeling is flexible
  - Adding and removing edges is simple
- Standard based graph representation
  - RDF was defined by W3C. Allows interoperability.
- Computers can understand the semantics RDF graphs (triples)
  - Same URI means same resource
Triples Are Easy. But Why? (3)

- Graph modeling is flexible
  - Adding and removing edges is simple
- Standard based graph representation
  - RDF was defined by W3C. Allows interoperability.
- Computers can understand the semantics RDF graphs (triples)
  - Discover hidden relationships, or detect inconsistency via logical inference
Triples Are Easy. But Why? (4)

- Graph modeling is flexible
  - Adding and removing edges is simple
- Standard based graph representation
  - RDF was defined by W3C. Allows interoperability.
- Computers can understand the semantics RDF graphs (triples)
  - Discover hidden relationships, or detect inconsistency via logical inference
SPARQL Graph Pattern
Basic unit of SPARQL queries


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)
Agenda

1. Introduction and Motivation
2. Review of RDF/OWL/SPARQL
3. RDB2RDF Standard
4. GeoSPARQL Standard
5. Adding GeoSPARQL Support to RDB2RDF
6. Demo
A Look at Some System Architectures

**Typical Traditional Application Architecture**

- Mid-tier server and database server
- Applications communicate via SQL/JDBC with RDBMS backend
- Multiple traditional relational schemas
- **Issues**
  - Inflexible schema
  - Limited semantics
  - Limited interoperability
A Look at Some System Architectures

Typical **Semantic** Application Architecture

- Common ontologies used to integrate datasets
- Applications communicate via SPARQL / HTTP with native triplestore backend
- Issues
  - Radical change for customer
  - Need for ETL to RDF
  - RDF/OWL may not be necessary for the entire data
A Look at Some System Architectures

How about RDB2RDF?

- 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
Using Direct Mapping: Overall Flow

- **Query Writer**
- **SPARQL QUERY**
- **Schema:** Classes and Predicates
- **Map:** R2RML mapping
- **R2RML Processor**
- **SPARQL to SQL Translator**
- **Source Relational Database**
- **Tables and views**
- **Direct Map Author**
Using R2RML: Overall Flow

- **Query Writer**
  - **Schema:** Classes and Predicates
  - **Map:** R2RML mapping

- **R2RML Processor**
  - **Source Relational Database**
  - **R2RML Document**
  - **R2RML Map Author**
  - **SPARQL QUERY**
  - **SPARQL to SQL Translator**
R2RML Basics: Mapping

- **TriplesMap**: Row of a Logical Table (Table / View / SQL query) $\rightarrow$ triples
  - **SubjectMap**: Primary key value of a Row $\rightarrow$ subject (+ static class + GraphMap)
  - **PredicateMap**: Names of columns and constraints $\rightarrow$ predicates (incl. rdf:type)
  - **ObjectMap**: Values in columns or foreign keys $\rightarrow$ objects (+ class)
  - **GraphMap**: Put this triple in a named graph
Vocabulary: R2RML Classes and Relations

Source: (annotated with relation names)
R2RML: RDB to RDF Mapping Language
W3C Recommendation 27 September 2012
http://www.w3.org/TR/r2rml/
Ex: Mapping EMP and DEPT Tables to RDF

```
<http://x.com/Dept/D100>
  rdf:type x:Department ;
  <"../Dept/Deptno"> 100 ;
  <"../Dept/DeptName"> "Sales" ;
  <"../Dept/Location"> "NYC"
.
```

**EMP**

<table>
<thead>
<tr>
<th>pkey</th>
<th>ENO</th>
<th>ENAME</th>
<th>EXPERTISE</th>
<th>DNO</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>John</td>
<td>DB</td>
<td></td>
<td>100</td>
</tr>
</tbody>
</table>

**DEPT**

<table>
<thead>
<tr>
<th>pkey</th>
<th>DNO</th>
<th>DNAME</th>
<th>LOC</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>Sales</td>
<td>NYC</td>
<td></td>
</tr>
</tbody>
</table>
Ex: Mapping EMP and DEPT Tables to RDF

Subject: http://x.com/Emp/E{ENO}

Predicate-object pairs:
- pkey: ENO, ENAME, EXPERTISE, DNO
- ref.: DNO, DNAME, LOC

Class: x:Employee

<http://x.com/Emp/E1> rdf:type x:Employee ;
<../Emp/Empno> 1 ;
<../Emp/EmpName> “John” ;
<../Emp/Expertise> “DB” ;
<../Emp/DeptNum> 100 ;
<../Emp/Department> <http://x.com/Dept/D100> .

Table:

<table>
<thead>
<tr>
<th>EMP</th>
<th>DEPT</th>
</tr>
</thead>
<tbody>
<tr>
<td>ENO</td>
<td>DNO</td>
</tr>
<tr>
<td>1</td>
<td>100</td>
</tr>
<tr>
<td>ENAME</td>
<td>DNAME</td>
</tr>
<tr>
<td>John</td>
<td>Sales</td>
</tr>
<tr>
<td>EXPERTISE</td>
<td>LOC</td>
</tr>
<tr>
<td>DB</td>
<td>NYC</td>
</tr>
</tbody>
</table>
## Schema for Generated RDF data

### Classes
- x:Department
- x:Employee

### Properties
- `<../Dept/Deptno>`
- `<../Dept/DeptName>`
- `<../Dept/Location>`
- `<../Emp/Empno>`
- `<../Emp/EmpName>`
- `<../Emp/Expertise>`
- `<../Emp/DeptNum>`
- `<../Emp/Department>`
## R2RML mapping

**set of TriplesMaps**

<table>
<thead>
<tr>
<th>TriplesMap: ...</th>
<th>PredicateObjectMap</th>
<th>PredicateObjectMap</th>
<th>PredicateObjectMap</th>
</tr>
</thead>
<tbody>
<tr>
<td>rr:logicalTable: ...</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SubjectMap:</td>
<td>PredicateMap:</td>
<td>PredicateMap:</td>
<td>PredicateMap:</td>
</tr>
<tr>
<td>[rr:class ...]</td>
<td>GraphMap:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>[rr:graphMap ...]</td>
<td>ObjectMap:</td>
<td>ObjectMap:</td>
<td></td>
</tr>
<tr>
<td>TriplesMap: ...</td>
<td>PredicateObjectMap</td>
<td></td>
<td></td>
</tr>
<tr>
<td>rr:logicalTable: ...</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SubjectMap:</td>
<td>PredicateMap:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>[rr:class ...]</td>
<td>ObjectMap:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>[rr:graphMap ...]</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.
### R2RML Mapping for EMP (w/ multi-ObjectMap)

<table>
<thead>
<tr>
<th>Empno</th>
<th>Ename</th>
<th>W_phone</th>
<th>C_phone</th>
<th>H_phone</th>
<th>W_addr</th>
<th>H_addr</th>
<th>DeptNo</th>
</tr>
</thead>
<tbody>
<tr>
<td>NUMBER</td>
<td>Varchar</td>
<td>NUMBER</td>
<td>Varchar</td>
<td>Varchar</td>
<td>Varchar</td>
<td>Varchar</td>
<td>NUMBER</td>
</tr>
<tr>
<td>1</td>
<td>JOHN</td>
<td>1112223333</td>
<td>1113334444</td>
<td>2225556666</td>
<td>NYC</td>
<td>NJ</td>
<td>100</td>
</tr>
</tbody>
</table>

**Tmap ➔ <#EmpTM>**

```rml
rr:logicalTable [ 
  rr:tableName "EMP"
].
```

**Smap ➔ []**

```rml
rr:template "http://ex.org/E/{EMPNO}" ; 
rr:class ex:Employee .
```

**POmap ➔ []**

```rml
rr:predicate em:phone ; 
rr:objectMap [
  rr:column "W_PHONE"
,  rr:column "C_PHONE"
,  rr:column "H_PHONE"
].
```

**POmap ➔ []**

```rml
rr:predicate em:dept ; 
rr:objectMap [ rr:parentTriplesMap <#DeptTM> ; 
  rr:joinCondition [ rr:child "DEPTNO" ; rr:parent "DEPTNO" ]].
```

---

*Oracle*
R2RML Mapping: using “R2RML Views”

```r2rml
<#DeptTableView>
  rr:sqIQuery "select DEPTNO, DNAME, LOC
  , (select COUNT(*) from EMP where EMP.DEPTNO=DEPT.DEPTNO) as STAFF
  from FROM DEPT "
</#DeptTableView>

<#TriplesMap1>
  rr:logicalTable <#DeptTableView>;

  rr:subjectMap [ rr:template "http://data.example.com/department/{DEPTNO}";
    rr:class ex:Department; ];

  rr:predicateObjectMap [ rr:predicate ex:name; rr:objectMap [ rr:column "DNAME" ]; ];
  rr:predicateObjectMap [ rr:predicate ex:location; rr:objectMap [ rr:column "LOC" ]; ];
  rr:predicateObjectMap [ rr:predicate ex:staff; rr:objectMap [ rr:column "STAFF" ]; ].
</#TriplesMap1>
```
R2RML Mapping: Translating type codes to IRIs

```xml
<#TriplesMap1>
  rr:logicalTable [ rr:sqlQuery ""
  SELECT *, (CASE JOB WHEN 'CLERK' THEN 'general-office'
  WHEN 'NIGHTGUARD' THEN 'security'
  WHEN 'ENGINEER' THEN 'engineering'
  END) ROLE
  FROM EMP
  
  rr:subjectMap [ rr:template "http://data.example.com/employee/{EMPNO}" ]; ];
  rr:predicateObjectMap [ rr:predicate ex:role;
  rr:objectMap [ rr:template "http://data.example.com/roles/{ROLE}" ]; ];
</#TriplesMap1>

```
Agenda

1. Introduction and Motivation
2. Review of RDF/OWL/SPARQL
3. RDB2RDF Standard
4. GeoSPARQL Standard
5. Adding GeoSPARQL Support to RDB2RDF
6. Demo
OGC GeoSPARQL Standard

- Open Geospatial Consortium Standard
- Published in 2012
- Modular standard that defines several components:
  - RDF Vocabulary
  - Datatypes for geometry literals
  - SPARQL extension functions
  - Entailment rules
Why GeoSPARQL?
Linked Geospatial Data

- Many Linked Open Data (LOD) datasets have geospatial components
- Some GIS providers want to publish LOD
- Barriers to integration
  - Vendor-specific geometry support
  - Different vocabularies
    - W3C Basic Geo, GML XMLLiteral, Vendor-specific
  - Different spatial reference systems
    - WGS84 Lat-Long, British National Grid
Semantic GIS

• GIS applications with semantically complex thematic aspects
  – Logical reasoning to classify features
    • Land cover type, suitable farm land, etc.
  – Complex Geometries
    • Polygons and Multi-Polygons with 1000’s of points
  – Complex Spatial Operations
    • Union, Intersection, Buffers, etc.

Find parcels with an area of at least 3 sq. miles that touch a local feeder road and are inside an area of suitable farm land.
Key Qualities of the GeoSPARQL Standard

- Provides a common target for implementers & users
  - Representation and query
- Works within SPARQL’s extensibility framework
- Simple enough for general users
  - Keep the common case simple (WGS 84 point data)
- Capable enough for GIS professionals
  - Multiple SRS’s, complex geometries, complex operators
- Leverages the wealth of existing spatial standards

ISO 19107 – Spatial Schema
ISO 13249 – SQL/MM

Simple Features
Well Known Text (WKT)
GML
From SPARQL to GeoSPARQL
**SPARQL Query**

### RDF Data

<table>
<thead>
<tr>
<th>:res1</th>
<th>rdf:type</th>
<th>:House</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>:baths</td>
<td>&quot;2.5&quot;^^xsd:decimal</td>
</tr>
<tr>
<td></td>
<td>:bedrooms</td>
<td>&quot;3&quot;^^xsd:decimal</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>:res2</th>
<th>rdf:type</th>
<th>:Condo</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>:baths</td>
<td>&quot;2&quot;^^xsd:decimal</td>
</tr>
<tr>
<td></td>
<td>:bedrooms</td>
<td>&quot;2&quot;^^xsd:decimal</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>:res3</th>
<th>rdf:type</th>
<th>:House</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>:baths</td>
<td>&quot;1.5&quot;^^xsd:decimal</td>
</tr>
<tr>
<td></td>
<td>:bedrooms</td>
<td>&quot;3&quot;^^xsd:decimal</td>
</tr>
</tbody>
</table>

**SPARQL Query**

```
SELECT ?r ?ba ?br
WHERE {
  ?r :bedrooms ?br
}
```

**Result Bindings**

<table>
<thead>
<tr>
<th>:res1</th>
<th>:baths</th>
<th>:bedrooms</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>&quot;2.5&quot;</td>
<td>&quot;3&quot;</td>
</tr>
<tr>
<td>:res3</td>
<td>&quot;1.5&quot;</td>
<td>&quot;3&quot;</td>
</tr>
</tbody>
</table>

---

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.
SPARQL Query

**RDF Data**

```
:res1 rdf:type :House .
:res1 :baths "2.5"^^xsd:decimal .
:res1 :bedrooms "3"^^xsd:decimal .
:res2 rdf:type :Condo .
:res2 :baths "2"^^xsd:decimal .
:res2 :bedrooms "2"^^xsd:decimal .
:res3 rdf:type :House
:res3 :baths "1.5"^^xsd:decimal .
```

**SPARQL Query**

```
SELECT ?r ?ba ?br
  ?r :bedrooms ?br
  FILTER (?ba > 2) }
```

**Result Bindings**

```
<table>
<thead>
<tr>
<th>?r</th>
<th>?ba</th>
<th>?br</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>&quot;1.5&quot;</td>
<td>&quot;2&quot;</td>
</tr>
<tr>
<td></td>
<td>&quot;2.5&quot;</td>
<td>&quot;3&quot;</td>
</tr>
</tbody>
</table>
```
```
Spatial SPARQL QUERY

:res1 rdf:type :House .
:res1 :baths "2.5"^^xsd:decimal .
:res1 :bedrooms "3"^^xsd:decimal .
:res1 ogc:hasGeometry :geom1 .
:geom1 ogc:asWKT "POINT(-122.25 37.46)"^^ogc:wktLiteral .

:res3 :baths "1.5"^^xsd:decimal .
:res3 ogc:hasGeometry :geom3 .
:geom3 ogc:asWKT "POINT(-122.24 37.47)"^^ogc:wktLiteral .

GeoSPARQL Query

SELECT ?r ?ba ?br
  ?r ogc:hasGeometry ?g . ?g ogc:asWKT ?wkt
  FILTER(ogcf:sfWithin(?wkt, "POLYGON(...)"^^ogc:wktLiteral))
}
```

This is what GeoSPARQL standardizes

Vocabulary & Datatypes

Extension Functions

Find houses within a search polygon
GeoSPARQL Vocabulary
Details of ogc:wktLiteral

All RDF Literals of type ogc:wktLiteral shall consist of an optional IRI identifying the spatial reference system followed by Simple Features Well Known Text (WKT) describing a geometric value [ISO 19125-1].

"<http://www.opengis.net/def/crs/OGC/1.3/CRS84>
POINT(-122.4192 37.7793)"^^ogc:wktLiteral

WGS84 longitude – latitude is the default CRS

"POINT(-122.4192 37.7793)"^^ogc:wktLiteral

European Petroleum Survey Group (EPSG) maintains a set of CRS identifiers.
More About Coordinate Reference Systems

A CRS is based on a specific datum.

Because Earth has a very irregular shape, different datums and corresponding CRSs provide more accurate fits for different locations.

Local CRSs are used for high precision applications.

The same location will have different coordinates in each CRS.

**Geoid**

Close approximation of Earth’s shape (mean sea level)

**Reference Ellipsoid / Datum**

Mathematically defined approximate surface

---

**GeoSPARQL Default**

- **WGS84** – Global (GIS, most web data, etc.)
- **ETRS89** – Europe
- **NAD83** – North America
- **GDA94** – Australia
- **OSGB36** – UK
Topological Relations between `ogc:SpatialObject`

- `ogc:sfEquals`
- `ogc:sfTouches`
- `ogc:sfOverlaps`
- `ogc:sfContains`
- `ogc:sfWithin`
- `ogc:sfDisjoint`
- `ogc:sfIntersects`
- `ogc:sfCrosses`

- `rdfs:domain/range ogc:SpatialObject`
- Assumes Simple Features Relation Family
- Also support Egenhofer and RCC8
### Example Data

<table>
<thead>
<tr>
<th>City</th>
<th>rdfs:subClassOf</th>
<th>ogc:Feature</th>
</tr>
</thead>
<tbody>
<tr>
<td>Park</td>
<td>rdfs:subClassOf</td>
<td>ogc:Feature</td>
</tr>
<tr>
<td>exactGeometry</td>
<td>rdfs:subPropertyOf</td>
<td>ogc:hasGeometry</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SanFrancisco</th>
<th>rdf:type</th>
<th>City</th>
</tr>
</thead>
<tbody>
<tr>
<td>UnionSquarePark</td>
<td>rdf:type</td>
<td>Park</td>
</tr>
<tr>
<td>UnionSquarePark</td>
<td>:commissioned</td>
<td>&quot;1847-01-01&quot;^^xsd:date</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>UnionSquarePark</th>
<th>:exactGeometry</th>
<th>geo1</th>
</tr>
</thead>
<tbody>
<tr>
<td>geo1</td>
<td>ogc:asWKT</td>
<td>&quot;Polygon(...)&quot;^^ogc:wktLiteral</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SanFrancisco</th>
<th>:exactGeometry</th>
<th>geo2</th>
</tr>
</thead>
<tbody>
<tr>
<td>geo2</td>
<td>ogc:asWKT</td>
<td>&quot;Polygon(...)&quot;^^ogc:wktLiteral</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>UnionSquarePark</th>
<th>ogc:sfWithin</th>
<th>SanFrancisco</th>
</tr>
</thead>
</table>
GeoSPARQL Query Functions
GeoSPARQL Query Functions


GeoSPARQL Query Functions

- ogcf:intersection(geom1: ogc:wktLiteral,
                   geom2: ogc:wktLiteral): ogc:wktLiteral

- ogcf:union(geom1: ogc:wktLiteral,
             geom2: ogc:wktLiteral): ogc:wktLiteral
GeoSPARQL Query Functions

- `ogcf:difference(geom1: ogc:wktLiteral,
  geom2: ogc:wktLiteral): ogc:wktLiteral`

- `ogcf:symDifference(geom1: ogc:wktLiteral,
  geom2: ogc:wktLiteral): ogc:wktLiteral`
GeoSPARQL Query Functions


- `ogcf:getSRID(geom: ogc:wktLiteral): xsd:anyURI`
GeoSPARQL Topological Query Functions


Assumes Simple Features Relation Family
GeoSPARQL Topological Query Functions

- `ogcf:sfTouches(geom1: ogc:wktLiteral,
   geom2: ogc:wktLiteral): xsd:boolean`

- `ogcf:sfCrosses(geom1: ogc:wktLiteral,
   geom2: ogc:wktLiteral): xsd:boolean`

- `ogcf:sfWithin(geom1: ogc:wktLiteral,
  geom2: ogc:wktLiteral): xsd:boolean`

Assumes Simple Features Relation Family
GeoSPARQL Topological Query Functions

- `ogcf:sfContains(geom1: ogc:wktLiteral,
  geom2: ogc:wktLiteral): xsd:boolean`

- `ogcf:sfOverlaps(geom1: ogc:wktLiteral,
  geom2: ogc:wktLiteral): xsd:boolean`
GeoSPARQL Topological Query Functions


**DE-9IM Intersection Matrix**

<table>
<thead>
<tr>
<th></th>
<th>geom1</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Interior</td>
</tr>
<tr>
<td>geom1</td>
<td>T</td>
</tr>
<tr>
<td>Boundary</td>
<td>F</td>
</tr>
<tr>
<td>Exterior</td>
<td>F</td>
</tr>
</tbody>
</table>

`ogc:sfContains`

`patternMatrix: TTTFFTTFFFT`
Example Query 1

Find historic places within 10 KM of San Francisco, CA city center

```
PREFIX geovocab: <http://geovocab.org/geometry#>
PREFIX uom: <http://xmlns.oracle.com/rdf/geo/uom/>
SELECT ?s ?l ?wkt
WHERE {
  ?s rdfs:label ?l .
  ?s geovocab:geometry ?geom .
  ?geom ogc:asWKT ?wkt .
  FILTER(
    ogcf:distance(?wkt,
      "POINT(-122.416667 37.783333)"^^ogc:wktLiteral,
      uom:KM) <= 10)
}
```
Example Query 2

Find Monuments within a query window

```
PREFIX geovocab: <http://geovocab.org/geometry#>
PREFIX lgd: <http://linkedgeoedata.org/ontology/>
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((-122.313056 37.554167,
                  -122.026389 36.971944,
                  -119.766667 36.75,
                  -120.994444 37.661389,
                  -122.313056 37.554167))"^^ogc:wktLiteral))
  )
}
```
Example Query 3

Find all land parcels that are within the intersection of :City1 and :District1

```
PREFIX : <http://my.com/appSchema#>
PREFIX ogc: <http://www.opengis.net/ont/geosparql#>
PREFIX ogcf: <http://www.opengis.net/def/geosparql/function/>
PREFIX epsg: <http://www.opengis.net/def/crs/EPSG/0/>

SELECT ?parcel
WHERE {
  ?parcel rdf:type :Residential .
  :District1 :exactGeometry ?dGeo .
  ?dGeo ogc:asWKT ?dWKT .
  :City1 :extent ?cGeo .
  ?cGeo ogc:asWKT ?cWKT .
  FILTER(ogcf:sfWithin(?pWKT, ogcf:intersection(?dWKT, ?cWKT)))
}
```
Summary

- **GeoSPARQL Defines:**
  - Basic vocabulary, Query functions, Entailment component
- **Based on existing OGC/ISO standards**
  - WKT, GML, Simple Features, ISO 19107
- **Uses SPARQL’s built-in extensibility framework**
- **Modular specification**
  - Allows flexibility in implementations
  - Easy to extend
Agenda

1. Introduction and Motivation
2. Review of RDF/OWL/SPARQL
3. RDB2RDF Standard
4. GeoSPARQL Standard
5. Adding GeoSPARQL Support to RDB2RDF
6. Demo
R2RML mapping of Relational Table with Spatial Data
Relational Data
for table with a single GeoSpatial column

<table>
<thead>
<tr>
<th>LOC</th>
<th>GEOM_MAIN</th>
</tr>
</thead>
<tbody>
<tr>
<td>NEW YORK</td>
<td>SDO_GEOMETRY('POINT(-74.005897521973 40.712699890137)',8307)</td>
</tr>
<tr>
<td>DALLAS</td>
<td>SDO_GEOMETRY('POINT(-96.796669006348 32.775833129883)',8307)</td>
</tr>
<tr>
<td>CHICAGO</td>
<td>SDO_GEOMETRY('POINT(-87.684722900391 41.836944580078)',8307)</td>
</tr>
<tr>
<td>BOSTON</td>
<td>SDO_GEOMETRY('POINT(-71.063613891602 42.358055114746)',8307)</td>
</tr>
</tbody>
</table>
## R2RML mapping for table with a single GeoSpatial column

<table>
<thead>
<tr>
<th>TriplesMap: LocGeom_IRI</th>
<th>PredicateObjectMap</th>
<th>PredicateObjectMap</th>
</tr>
</thead>
<tbody>
<tr>
<td>rr:logicalTable: RDFUSER.LOCGEOM</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SubjectMap: &quot;<a href="http://ex/LOC/%7BLOC%7D">http://ex/LOC/{LOC}</a>&quot;</th>
<th>PredicateMap:</th>
<th>PredicateMap:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ex:location</td>
<td>ex:hasGeomMain_IRI</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ObjectMap: rr:column &quot;LOC&quot;</th>
<th>ObjectMap:</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>rr:parentTriplesMap: LocGeom_MAIN</td>
<td></td>
<td></td>
</tr>
<tr>
<td>rr:joinCondition: child.LOC = parent.LOC</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>TriplesMap: LocGeom_MAIN</th>
<th>PredicateObjectMap</th>
</tr>
</thead>
<tbody>
<tr>
<td>rr:logicalTable: RDFUSER.LOCGEOM</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SubjectMap: &quot;<a href="http://ex/GEOM_MAIN/%7BLOC%7D">http://ex/GEOM_MAIN/{LOC}</a>&quot;</th>
<th>PredicateMap:</th>
<th>ObjectMap:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ogc:asWKT</td>
<td>rr:column</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&quot;GEOM_MAIN&quot;</td>
</tr>
</tbody>
</table>
### Content of RDF View using R2RML mapping

for table with a single GeoSpatial column

<table>
<thead>
<tr>
<th>loc:NEW%20YORK</th>
<th>ex:location</th>
<th>&quot;NEW YORK&quot; ; geomMain:NEW%20YORK .</th>
</tr>
</thead>
<tbody>
<tr>
<td>geomMain:NEW%20YORK</td>
<td>ogc:asWKT</td>
<td>&quot;POINT(...)&quot;^^ogc:wktLiteral .</td>
</tr>
<tr>
<td>loc:DALLAS</td>
<td>ex:location</td>
<td>&quot;DALLAS&quot; ; geomMain:DALLAS .</td>
</tr>
<tr>
<td>geomMain:DALLAS</td>
<td>ogc:asWKT</td>
<td>&quot;POINT(...)&quot;^^ogc:wktLiteral .</td>
</tr>
<tr>
<td>loc:CHICAGO</td>
<td>ex:location</td>
<td>&quot;CHICAGO&quot; ; geomMain:CHICAGO .</td>
</tr>
<tr>
<td>geomMain:CHICAGO</td>
<td>ogc:asWKT</td>
<td>&quot;POINT(...)&quot;^^ogc:wktLiteral .</td>
</tr>
<tr>
<td>loc:BOSTON</td>
<td>ex:location</td>
<td>&quot;BOSTON&quot; ; geomMain:BOSTON .</td>
</tr>
<tr>
<td>geomMain:BOSTON</td>
<td>ogc:asWKT</td>
<td>&quot;POINT(...)&quot;^^ogc:wktLiteral .</td>
</tr>
</tbody>
</table>

@prefix loc: <http://ex/LOC/> . @ prefix geomMain <http://ex/GEOM_MAIN/>
GeoSPARQL query against RDF View
for table with a single GeoSpatial column

PREFIX ex: <http://ex/>
PREFIX uom: <http://xmlns.oracle.com/rdf/geo/uom/>
select ?loc1 ?loc2 (ogcf:distance(?geom1,?geom2, uom:KM) as ?dist_km)
where {
    ?l1 ex:location ?loc1
       ; ex:hasGeomMain_IRI ?giri1 .
    ?giri1 ogc:asWKT ?geom1 .
    ?l2 ex:location ?loc2
       ; ex:hasGeomMain_IRI ?giri2 .
    FILTER(?loc1 > ?loc2)
}

Find distance (in KM) between each pair of locations
GeoSPARQL query against RDF View
for table with a single GeoSpatial column

PREFIX ex: <http://ex/>
select ?loc ?geom_main
where {
  ?giri_main ogc:asWKT ?geom_main .
  FILTER(ogcf:sfWithin(?geom_main, "POLYGON((-75.0 43.0, -75.0 40.0, -70.0 40.0, -70.0 43.0, -75.0 43.0))"^^ogc:wktLiteral))
}
Relational Data
For table with multiple GeoSpatial columns

<table>
<thead>
<tr>
<th>LOC</th>
<th>GEOM_MAIN</th>
<th>GEOM_SOUTH</th>
</tr>
</thead>
<tbody>
<tr>
<td>NEW YORK</td>
<td>SDO_GEOMETRY('POINT(-74.005897521973 40.712699890137)',8307)</td>
<td>SDO_GEOMETRY('POINT(-74.247863 40.496377)',8307)</td>
</tr>
<tr>
<td>DALLAS</td>
<td>SDO_GEOMETRY('POINT(-96.796669006348 32.775833129883)',8307)</td>
<td>SDO_GEOMETRY('POINT(-96.949015 32.621286)',8307)</td>
</tr>
<tr>
<td>CHICAGO</td>
<td>SDO_GEOMETRY('POINT(-87.684722900391 41.836944580078)',8307)</td>
<td>SDO_GEOMETRY('POINT(-87.615434 41.645612)',8307)</td>
</tr>
<tr>
<td>BOSTON</td>
<td>SDO_GEOMETRY('POINT(-71.063613891602 42.358055114746)',8307)</td>
<td>SDO_GEOMETRY('POINT(-71.131317 42.229262)',8307)</td>
</tr>
</tbody>
</table>
### R2RML mapping
for table with multiple GeoSpatial columns

<table>
<thead>
<tr>
<th>TriplesMap: LocGeom_IRI</th>
<th>PredObjMap</th>
<th>PredObjMap</th>
<th>PredicateObjectMap</th>
</tr>
</thead>
<tbody>
<tr>
<td>rr:logicalTable: RDFUSER.LOCGEOM</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**SubjectMap:**
"http://ex/LOC/{LOC}"

<table>
<thead>
<tr>
<th>TriplesMap: LocGeom_MAIN</th>
<th>PredicateObjectMap</th>
</tr>
</thead>
<tbody>
<tr>
<td>rr:logicalTable: RDFUSER.LOCGEOM</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
</tbody>
</table>

**TriplesMap: LocGeom_SOUTH  | PredicateObjectMap |
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>rr:logicalTable: RDFUSER.LOCGEOM</td>
<td></td>
</tr>
</tbody>
</table>

**SubjectMap:**
"http://ex/GEOM_SOUTH/{LOC}"  

<table>
<thead>
<tr>
<th>PredicateMap: ex:location</th>
<th>PredicateMap: ex:geomMain_IRI</th>
<th>PredicateMap: ex:hasGeomSouth_IRI</th>
</tr>
</thead>
<tbody>
<tr>
<td>ObjectMap: ...</td>
<td>ObjectMap: ...</td>
<td>ObjectMap: rr:parentTriplesMap LocGeom_SOUTH</td>
</tr>
<tr>
<td></td>
<td></td>
<td>rr:joinCondition: child.LOC = parent.LOC</td>
</tr>
</tbody>
</table>

**New TriplesMap**

**New PredicateObjectMap**
Content of RDF View using R2RML mapping
for table with multiple GeoSpatial columns

```
loc:NEW%20YORK ...
ex:hasGeomSouth_IRI geomSouth:NEW%20YORK .
geomSouth:NEW%20YORK ogc:asWKT "POINT(...)"^^ogc:wktLiteral .
loc:DALLAS ...
ex:hasGeomSouth_IRI geomSouth:DALLAS .
geomSouth:DALLAS ogc:asWKT "POINT(...)"^^ogc:wktLiteral .
loc:CHICAGO ...
ex:hasGeomSouth_IRI geomSouth:CHICAGO .
geomSouth:CHICAGO ogc:asWKT "POINT(...)"^^ogc:wktLiteral .
loc:BOSTON ...
ex:hasGeomSouth_IRI geomMain:BOSTON .
geomSouth:BOSTON ogc:asWKT "POINT(...)"^^ogc:wktLiteral .
```

@prefix loc: <http://ex/LOC/> . @ prefix geomSouth <http://ex/GEOM_SOUTH/>
GeoSPARQL query against RDF View
for table with multiple GeoSpatial columns

PREFIX ex: <http://ex/>
PREFIX uom: <http://xmlns.oracle.com/rdf/geo/uom/>
select ?loc (ogcf:distance(?geom_main,?geom_south, uom:KM) as ?dist_km)
where {
  ?l ex:location ?loc
  ; ex:hasGeomMain_IRI ?giri_main
  ; ex:hasGeomSouth_IRI ?giri_south .
  ?giri_main ogc:asWKT ?geom_main .

  Find distance (In KM) between main and southernmost points of each city
Agenda

1. Introduction and Motivation
2. Review of RDF/OWL/SPARQL
3. RDB2RDF Standard
4. GeoSPARQL Standard
5. Adding GeoSPARQL Support to RDB2RDF
6. Demo
Exposing GIS Data as Linked Data

Starting Point:
County geometries with demographic and economic data downloaded from the US Census Bureau in ESRI shapefile format

Goal:
Create a SPARQL endpoint that exposes this data as Linked Data.
Exposing GIS Data as Linked Data

• Software Used
  – Big Data Lite 4.6 Virtual Machine Image
    • Oracle Database 12.1.0.2.0 Enterprise Edition with Spatial and Graph option (Plus patch 25114114)
  – SQL Developer 4.2 Early Adopter
  – Oracle Map Builder 12.2.1.2
  – Oracle Support for Apache Jena 2.11.2 and Fuseki 1.0.1
Integrated Cloud
Applications & Platform Services