Enhancing Statistical Discovery with Oracle RDF on Oracle Cloud

Shoki Nishimura, National Statistics Center of Japan
Yusuke Takeyoshi, Senior Principal Consultant, Oracle Japan
Agenda

✓ Background
  • Introduction of e-Stat System
  • Why We Developed LOD
  • How We Configured e-Stat LOD
  • Integrating GeoSpatial RDF Data in e-Stat LOD
  • Sample Application (Demo)

✓ Technical Details of e-Stat LOD
  • LOD System Architecture
  • Size and Scale of e-Stat LOD
  • SPARQL Performance Concerns
  • Database Design to Improve Performance
Agenda

✓ Background
  • Introduction of e-Stat System
  • Why We Developed LOD
  • How We Configured e-Stat LOD
  • Integrating GeoSpatial RDF Data in e-Stat LOD
  • Sample Application (Demo)

✓ Technical Details of e-Stat LOD
  • LOD System Architecture
  • Size and Scale of e-Stat LOD
  • SPARQL Performance Concerns
  • Database Design to Improve Performance
Introduction of e-Stat System
- Portal Site for Official Statistics of Japan

- In 2008, e-Stat started to publish statistical data of government agencies (Format: Excel)
- In 2014, API service started (Format: XML, JSON, CSV)
- In 2016, LOD (Linked Open Data) service started (Format: RDF)

e-Stat (https://www.e-stat.go.jp/)
What is RDF and LOD?

RDF (Resource Description Framework)
- RDF is a standard model for data interchange on the web.
- RDF uses URIs to name the relationship as well as the two ends of the link (this is usually referred to as a Triple).

Structure of Triple
- Subject — Predicate — Object

Structure of RDF Graph
- Alice is a friend of Bob.
- Leonardo Da Vinci created The Mona Lisa.
- Bob is interested in The Mona Lisa.

LOD (Linked Open Data)
- LOD is structured open data interlinked with other data.
- LOD builds on RDF technologies.

TIM BERNERS-LEE’S FIVE-STAR SCHEME
- http://5star.data.info

KEY:
- OL: open licence
- RE: machine readable
- OF: open format
- URI: uniform resource identifier
- LD: linked data
Why We Developed LOD
From “Link to File” To “Link to Data”

Clarify semantics and origins for data

<table>
<thead>
<tr>
<th>Sex</th>
<th>Total (Sex)</th>
<th>Male</th>
</tr>
</thead>
<tbody>
<tr>
<td>Age</td>
<td>44 years [Person]</td>
<td>45 years [Person]</td>
</tr>
<tr>
<td>Standard area code</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>Saitama-city</td>
<td>16,130</td>
<td>19,245</td>
</tr>
<tr>
<td>Kawaguchi-city</td>
<td>6,582</td>
<td>8,022</td>
</tr>
</tbody>
</table>

Assign URI to page (http://www.e-stat.go.jp/pages.html)
Assign URI to file (http://www.e-stat.go.jp/xls0001.xls)
Assign URI to each data (http://data.e-stat.go.jp/.../obs00001)
Assign URI to each data (http://data.e-stat.go.jp/.../C11201)
Why We Developed LOD

Metadata for statistical data in Japan is not standardized, which makes it hard to process data. Define standardized metadata as **RDF** to make it **machine-readable**.
How We Configured e-Stat LOD

Example: The population of 44-year-old men in Kawaguchi City in 2010

e.g. Population of 2010 Population census

<table>
<thead>
<tr>
<th>Standard area code</th>
<th>Sex Age</th>
<th>Total (Sex)</th>
<th>Male</th>
<th>Female</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>44 years [Unit Of Person]</td>
<td>45 years [Unit Of Person]</td>
<td>44 years [Unit Of Person]</td>
<td>45 years [Unit Of Person]</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Male</td>
<td>Female</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Saitama-city</td>
<td></td>
<td></td>
<td>2</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>Kawaguchi-city</td>
<td></td>
<td></td>
<td>4</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>5</td>
<td></td>
</tr>
</tbody>
</table>

Original Excel format

Converted RDF format (R2RML)
Integrating GeoSpatial RDF Data in e-Stat LOD

Major statistics such as Population Census are integrated with more fine-grained location data, such as 250m x 250m square grids, which are defined as Polygons.

✓ Statistical data can be obtained with GeoSPARQL.

Example of GeoSPARQL FILTER expression

```
FILTER ( ogc:sfWithin(?wkt, "POLYGON(...)"^^geo:wktLiteral)
```

Query population info WITHIN a search polygon
Sample Application


Linked Open Data

LOD sample applications

Search LOD on map

Easy LOD search

Analytics and Data Summit 2020
Sample Application

Click
Region name
Polygon

PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX geo: <http://www.opengis.net/ont/geosparql#>
PREFIX ogcf: <http://www.opengis.net/def/function/geosparql/>
PREFIX dcterms: <http://purl.org/dc/terms/>

SELECT DISTINCT ?area ?areaName ?areaType ?areaid ?areaWKT
WHERE {
?area geo:hasGeometry / geo:asWKT ?areaWKT ;
a ?areaType ;
dcterms:identifier ?areaid ;
rdfs:label ?areaName .
FILTER (ogcf:sfContains(?areaWKT, 'Point(139.78247469054514 35.734430056624056)^^geos:wtLiteral'))
}
LIMIT 10

GeoSPARQL
Sample Application

Select dataset

- Population by sex - 5th level grid squares
- Population by age and sex - 5th level grid squares
- Contraception by sex - 5th level grid squares
- Households by type of household - 5th level grid squares
- Private households by one of households - 5th level grid squares
- Private households by family type of household - 5th level grid squares
- Private households by division of household member's age - 5th level grid squares
- One-person private households by presence of household member 20-29 years of age - 5th level grid squares
- Private households by type of household by presence of adult household members - 5th level grid squares

Select measure and dimension

- Dataset: Population by sex - 5th level grid squares
- Measure: Population
- Dimension: All (Sex), Male, Female

View observation value

- Search result:
  - All (Sex): 2178
  - Male: 1046
  - Female: 1132

SPARQL

```sparql
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX db: <http://purl.org/linked-data/cube#>
PREFIX dbp: <http://purl.org/linked-data/cube#>
PREFIX dmx: <http://data.e-stat.go.jp/lcd/ontology/dimension/>
PREFIX dme: <http://data.e-stat.go.jp/lcd/ontology/measure/>
PREFIX data: <http://data.e-stat.go.jp/lcd/sax/>
PREFIX dcd: <http://purl.org/dc/terms/>

SELECT DISTINCT
  SELECTMeasure
  ?selectedMeasure
  ?selectedDimension

WHERE {
  SELECTMeasure
  ?selectedMeasure
  ?selectedDimension

  FILTER (dbp:All(Sex) = "All (Sex)"
             OR dbp:Male = "Male"
             OR dbp:Female = "Female")

  FILTER (
    ?selectedMeasure = data:Population
    OR ?selectedMeasure = data:Male
    OR ?selectedMeasure = data:Female)

  FILTER (?selectedDimension = dme:Population)
}
```

Download
Agenda

✓ Background
- Introduction of e-Stat System
- Why We Developed LOD
- How We Configured e-Stat LOD
- Integrating GeoSpatial RDF Data in e-Stat LOD
- Sample Application (Demo)

✓ Technical Details of e-Stat LOD
- LOD System Architecture
- Size and Scale of e-Stat LOD
- SPARQL Performance Concerns
- Database Design to Improve Performance
LOD System Architecture

Oracle Gen2 Cloud
- Production env. for e-Stat “LOD” system
- Publishes RDF data generated on the on-premise Exadata

On-Premise (Exadata)
- Production env. for basic e-Stat systems
- Publishes statistical data in relational tables
- Generates RDF data from relational tables

Fuseki SPARQL Endpoint
- e-Stat SPARQL query interface

Compute Cloud
- Service monitoring

Management Cloud
- Database Cloud Extreme Performance (BareMetal instance)

e-Stat Databases
- Transform tables to RDF (R2RML)

Transfer RDF files

Oracle Database
- RDF

Analytics and Data Summit 2020
SPARQL Performance Concerns

- In 2018, triples increased to 1,300 million (including GeoSpatial data).
  - At that time, the e-Stat LOD was running on on-premise Exadata 12cR2.

- SPARQL performance became no longer acceptable…
  - Tested 138 different SPARQL queries (including several GeoSPARQL)

<table>
<thead>
<tr>
<th></th>
<th>Exadata 12cR2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Avg. SPARQL response time</td>
<td>65.74 secs</td>
</tr>
<tr>
<td>Number of queries running over 10 sec.</td>
<td>108</td>
</tr>
<tr>
<td>Number of queries running over 300 sec.</td>
<td>13</td>
</tr>
</tbody>
</table>

- We needed a drastic measures to improve SPARQL performance!
Database Design to Improve Performance

The following tuning drastically improved query performance.

1. Using Database In-Memory features
2. Partitioning RDF table by triple “predicate”
3. Optimizing Optimizer Statistics based on actual SPARQLs

*Tuning 1, 2 are only available from Oracle 18c.*
1. Using Database In-Memory Features

DB In-Memory (DBIM) improved SPARQL performance by 10 to 60 times.

- Enabling DBIM for RDF is very simple:
  ```java
  exec SEM_APIs.ENABLE_INMEMORY(TRUE);
  ```

- To reduce the amount of data accessed:
  - The populated data is automatically compressed in memory
  - In-Memory Indexes automatically prunes the data accessed
Database Design to Improve SPARQL Performance

2. Partitioning RDF Table by Triple “predicate”

In many cases, graph patterns in SPARQL queries specify “predicate” URI and query “object” values.

```
select ?year ?population
where {
  ?s estat-measure:population ?population;
  sdmx-dimension:refArea / rdfs:label 'Kyoto-shi'@en;
  cd-dimension:timePeriod ?year;
  cd-dimension:sex cd-code:sex-all;
  cd-dimension:nationality cd-code:nationality-japan;
  g00200521-dimension-2010:area g00200521-code-2010:area-all;
  cd-dimension:age cd-code:age-all .
}
```
Database Design to Improve SPARQL Performance

2. Partitioning RDF Table by Triple “predicate”

To reduce the amount of data accessed, we partitioned the RDF_LINK$ table using hash values of the RDF predicate IDs.

**EXAMPLE:** Each partition stores triples with the following predicates:
- rdfs:subClassOf
- estat-measure:population
- rdfs:label
- cd-dimension:age
- rdf:type

**SPARQL**

```
SELECT ?s ?o
WHERE {
  ?s rdfs:label ?o
}
LIMIT 10
```

Access only a partition containing “rdfs:label” predicate.
3. Optimizing Optimizer Statistics

In Oracle Database, RDF triples are stored in relational tables (RDF_LINK$, RDF_VALUE$, ...), so SPARQLs are translated and executed as semantically the same SQLs.

➤ Optimizer Statistics are very important to generate optimal execution plans.

Examples of Optimizer Statistics

- **Number of rows**
- **Avg. row length**
- **Distinct num of values in each column**

Optimizer makes optimal execution plans considering statistics

- Full Table Scan or Index Scan ?
- JOIN order
- JOIN method (LOOP or HASH ?)

e.t.c.
Database Design to Improve SPARQL Performance

3. Optimizing Optimizer Statistics

For the optimizer to make a good execution plan against complex SPARQL queries, we gathered **column group statistics**, which enables optimizer to consider a correlation relationship between different columns. Which column group statistics are useful was determined using **SPARQLs actually executed so far in the e-Stat LOD**.

**STEP1:** Tell the database to monitor column group usage for the specified seconds.
```
exec DBMS_STATS.SEED_COL_USAGE(NULL, NULL, 600);
```

**STEP2:** Execute as many SPARQLs as possible, which are executed so far, within the specified time

**STEP3:** Mark the useful column groups detected during the monitoring
```
SELECT DBMS_STATS.CREATE_EXTENDED_STATS('MDSYS', 'RDF_LINKS') FROM DUAL;
```

**STEP4:** Gather statistics by a pre-built procedure SEM_PERF.GATHER_STATS. The marked column group statistics are automatically gathered.
```
exec SEM_PERF.GATHER_STATS(...);
```
Database Design to Improve SPARQL Performance

How Much SPARQL Performance Improved?

<table>
<thead>
<tr>
<th>Tested 138 different SPARQL queries against 1.3 billion triples (including several GeoSPARQL)</th>
<th>On-prem Exadata Oracle 12cR2 (former platform)</th>
<th>Oracle Gen2 Cloud Oracle 18c with tunings</th>
</tr>
</thead>
<tbody>
<tr>
<td>Avg. SPARQL response time</td>
<td>65.74 sec.</td>
<td><strong>1.27 sec.</strong></td>
</tr>
</tbody>
</table>

Former platform ➞ Current platform

![Graph showing performance improvement over time](image)

- Avg. SPARQL response time:
  - 300M of triples in 2015:
  - 500M of triples in 2017:
  - 1,300M of triples in 2018:
  - 2,100M of triples in 2019:
  - The performance remained constant, even after triples increased to 2.1 billion!

Analytics and Data Summit 2020
✓ Publication of the 1\textsuperscript{st} statistical LOD in Japan
   - 9 major statistics are published as LOD with Oracle Cloud
   - RDF triples are generated by use of R2RML from relational tables
   - GeoSpatial triples are integrated and published as LOD

✓ Performance improvement for SPARQL queries
   - We achieved 50 times faster performance applying the following changes:
     - Migrating the entire LOD platform to Oracle Gen2 Cloud
     - Utilizing DBIM features
     - Partitioning a RDF table by triple predicate
     - Gathering column group statistics
Questions & Answers
How We Configured e-Stat LOD

Data in e-Stat LOD is defined using RDF Data Cube Vocabulary (W3C)

- The RDF Data Cube Vocabulary provides a way to publish multi-dimensional statistics in such a way that it can be linked to related data sets and concepts, [https://www.w3.org/TR/vocab-data-cube/](https://www.w3.org/TR/vocab-data-cube/).
- Each observation, or data in each cell, is described by dimensions, measures, and attributes.

<table>
<thead>
<tr>
<th>Dimension</th>
<th>Measure</th>
<th>Observation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total (Sex)</td>
<td>e.g. Population of 2010 Population census</td>
<td></td>
</tr>
<tr>
<td>44 years</td>
<td>45 years</td>
<td>Male</td>
</tr>
<tr>
<td>[Unit Of Person]</td>
<td>[Unit Of Person]</td>
<td>[Unit Of Person]</td>
</tr>
<tr>
<td>Saitama-city</td>
<td>16,130</td>
<td>19,245</td>
</tr>
<tr>
<td>Kawaguchi-city</td>
<td>6,582</td>
<td>8,022</td>
</tr>
</tbody>
</table>
How We Configured e-Stat LOD

RDF data was generated from statistics tables in our database with **R2RML** (R2RML = RDB to RDF Mapping Language).

Statistics Tables (Relational tables)

Transform tables to RDF format (R2RML)

Example of R2RML mapping file

```r
@prefix rr: <http://www.w3.org/ns/r2rml#>.
@prefix ex: <http://example.com/ns#>.

<#TriplesMap1>
  rr:logicalTable [ rr:tableName "EMP" ];
  rr:subjectMap [  
    rr:template "http://data.example.com/employee/{EMPNO}";
    rr:class ex:Employee;
  ];
  rr:predicateObjectMap [  
    rr:predicate ex:name;
    rr:objectMap [ rr:column "ENAME" ];
  ].
```

R2RML defines:
- Base table(s) / view(s)
- Rules that map each row in the base table to RDF triples
Integrating GeoSpatial RDF Data in e-Stat LOD

Administrative Divisions

Japan → prefectural divisions (e.g., Tokyo) → municipal divisions (e.g., Shinjuku-ward)

standard area codes

Statistics

world grid square codes

Grid Squares

3rd grid square (1km)
4th grid square (500m)
5th grid square (250m)

Polygon

statistics-specific codes

Small Areas

Subdivided administrative units of a municipal divisions

Regional Map

Analytics and Data Summit 2020
Size and Scale of e-Stat LOD (as of today)

**AP Tier**
- Fuseki on Jetty (x2 VM instances)
  - CPU: 8 cores
  - RAM: 120 GB

**DB Tier**
- Oracle Database Extreme Performance
  - CPU: 12 cores
  - RAM: 754 GB
  - BareMetal Instance
  - Number of Triples: 2.1 billion
1. Using Database In-Memory Features

DB In-Memory Settings for e-Stat LOD

- 380GB In-Memory Area (SGA = 600GB)
- Set the RDF semantic network indexes to INVISIBLE for the optimizer

Two semantic network indexes are created on RDF_LINK$ table by default:
- Index 1: Predicate - Object – Subject
- Index 2: Predicate - Subject - Object

- Minimize the area for METADATA to maximize the In-Memory area size for DATA
  - Set "_inmemory_64k_percent"=1 to reduce the metadata area to 1%

In-Memory Area (380GB)

- 1 MB Pool: for populated data
- 64 kB Pool: for metadata about populated data

<table>
<thead>
<tr>
<th>POOL</th>
<th>ALLOC GB</th>
</tr>
</thead>
<tbody>
<tr>
<td>1MB POOL</td>
<td>356.8</td>
</tr>
<tr>
<td>64KB POOL</td>
<td>22.9</td>
</tr>
</tbody>
</table>

99% for DATA
1% for METADATA
Database Design to Improve SPARQL Performance

1. Using Database In-Memory Features

Example: Tested SPARQL Query
Queries population census data in Kyoto City

```sparql
SELECT ?year ?population
WHERE {
  ?s estat-measure:population ?population;
  sdmx-dimension:refArea / rdfs:label 'Kyoto-shi'@en;
  cd-dimension:timePeriod ?year;
  cd-dimension:sex cd-code:sex-all;
  cd-dimension:nationality cd-code:nationality-japan;
  g00200521-dimension-2010:area g00200521-code-2010:area-all;
  cd-dimension:age cd-code:age-all.
}
```

<table>
<thead>
<tr>
<th>Year</th>
<th>Population</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;2015&quot;^^xsd:gYear</td>
<td>&quot;1412924&quot;^^xsd:decimal</td>
</tr>
<tr>
<td>&quot;2010&quot;^^xsd:gYear</td>
<td>&quot;1408039&quot;^^xsd:decimal</td>
</tr>
</tbody>
</table>

**WITHOUT DB In-Memory**
7.1 sec

**WITH DB In-Memory**
0.58 sec

12x faster
2. Partitioning RDF Table by Triple “predicate”

Partitioning Settings for e-Stat LOD

✓ Hash-Partitioning RDF_LINK$ can be done when creating a semantic network.

```
BEGIN
  SEM_APIs.CREATE_SEM_NETWORK(
    '< tablespace name for semantic network >',
    options=>' MODEL_PARTITIONING=BY_HASH_P MODEL_PARTITIONS=64 '
  );
END;
```

✓ The number of partitions = 64
- Partitioning by Hash should be done by a power of 2 (2, 4, 8, 16, 32, 64, 128, ...) to equally distribute the number of triples in each partition.
- In e-Stat LOD, the distinct number of predicates is 144.