Spatial and Graph Summit @

ANALYTICS AND DATA SUMMIT 2020

All Analytics. All Data.
No Nonsense.

February 25-27, 2020

Enhancing Statistical Discovery with Oracle RDF on Oracle Cloud

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

National Statistics Center
NSTAC

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

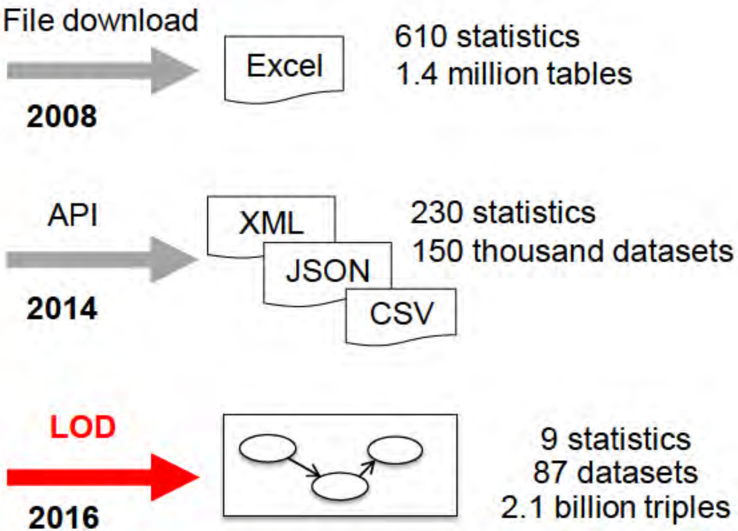
Agenda

- ✓ **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?

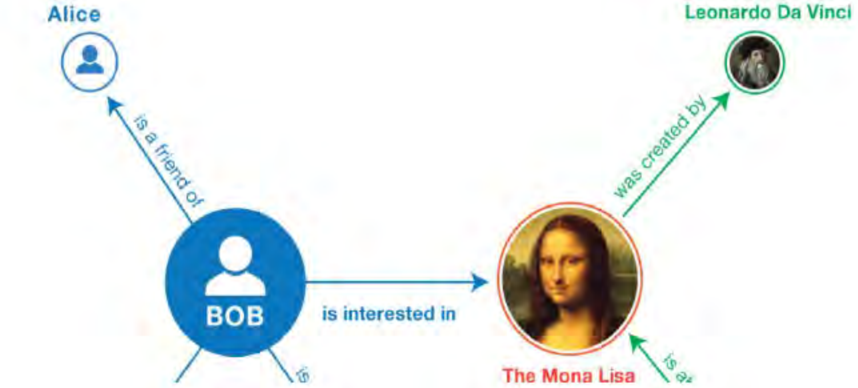
RDF (Resource Description Framework)

- 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

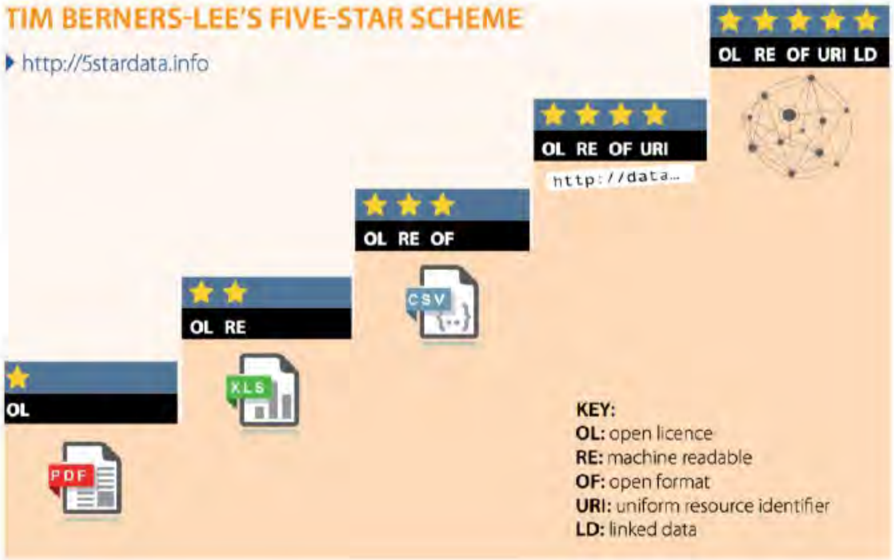
Subject — Predicate → Object

Structure of RDF Graph



LOD (Linked Open Data)

- LOD is structured open data interlinked with other data.
- LOD builds on RDF technologies.



Why We Developed LOD

NSTAC

From *"Link to File"* To *"Link to Data"*

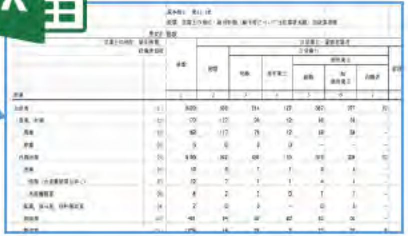
Clarify semantics and origins for data

Link to File

Assign URI to page
(http://www.e-stat.go.jp/pages.html)

HTML

Assign URI to file
(http://www.e-stat.go.jp/xls0001.xls)



Link to Data

Sex / Age / Standard area code	Total (Sex)			Male		
	・・・	44 years [Person]	45 years [Person]	・・・	44 years [Person]	45 years [Person]
・・・	・・・	・・・	・・・	・・・	・・・	・・・
Saitama-city	・・・	16,130	19,245	・・・	8,293	9,938
Kawaguchi-city	・・・	6,582	8,022	・・・	3,526	4,289
・・・	・・・	・・・	・・・	・・・	・・・	・・・

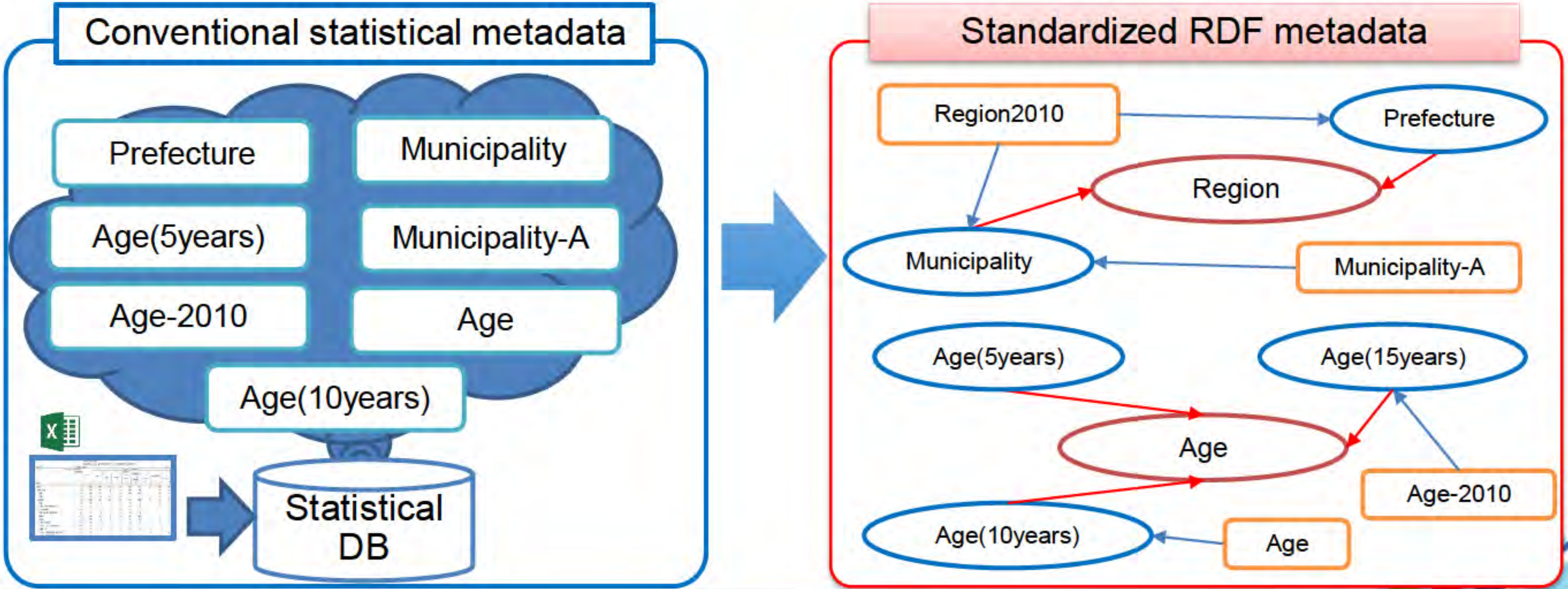
Assign URI to each data
(http://data.e-stat.go.jp/lod/…/obs00001)

Assign URI to each data
(http://data.e-stat.go.jp/lod/…/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

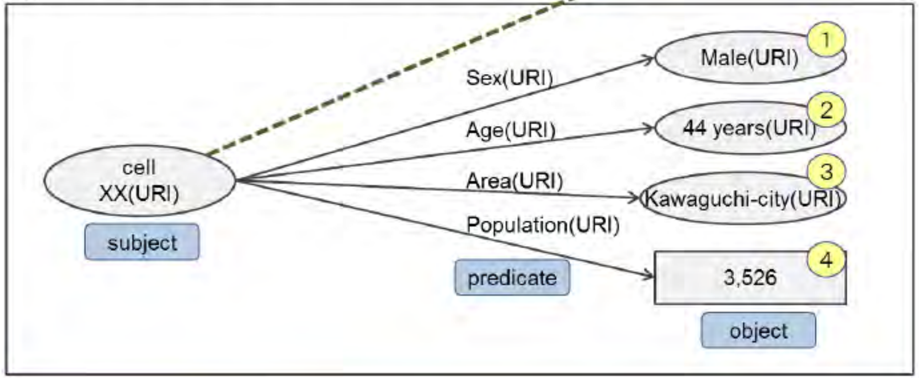
e.g. Population of 2010 Population census

Original Excel format

Sex / Age / Standard area code	Total (Sex)				Male (1)				Female
	...	44 years [Unit Of Person]	45 years [Unit Of Person]	...	...	44 years (2) [Unit Of Person]	45 years [Unit Of Person]	...	...
...	...	...	...	...	...	...	...	...	...
Saitama-city	...	16,130	19,245	...	...	8,293	9,938	...	...
Kawaguchi-city (3)	...	6,582	8,022	...	...	(4) 3,526	4,289	...	...
...	...	...	...	...	...	...	...	...	...

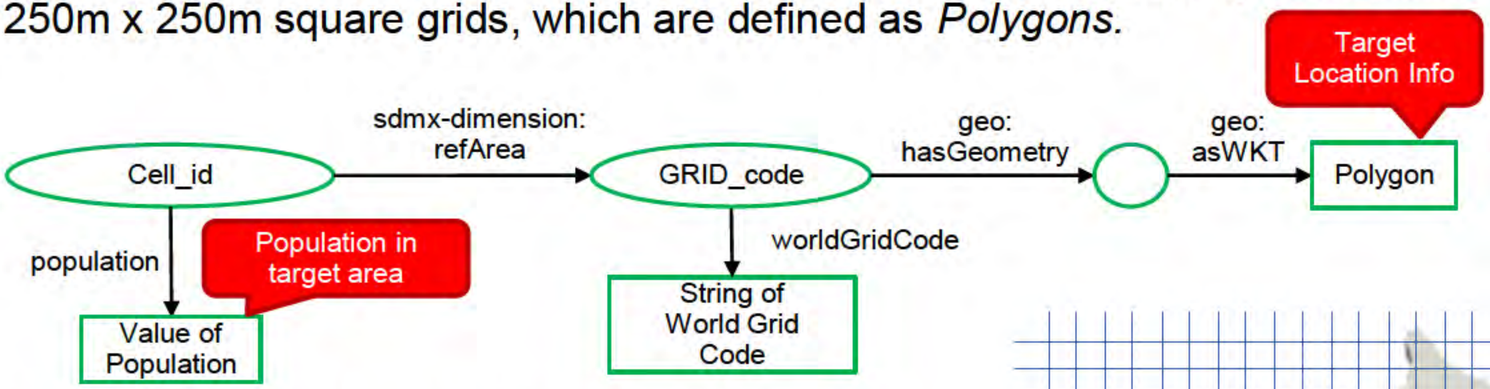
convert each observation values (cells) into RDF data

Converted RDF format (R2RML)



Integrating GeoSpatial RDF Data in e-Stat LOD

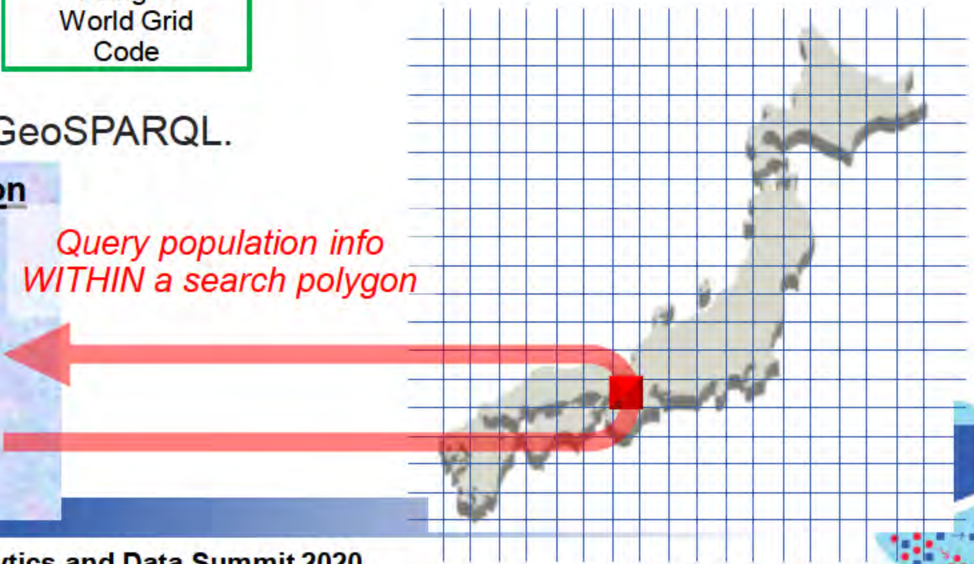
Major statistics such as Population Census are integrated with more fine-grained location data, such as 250m x 250m square grids, which are defined as *Polygons*.



✓ Statistical data can be obtained with GeoSPARQL.

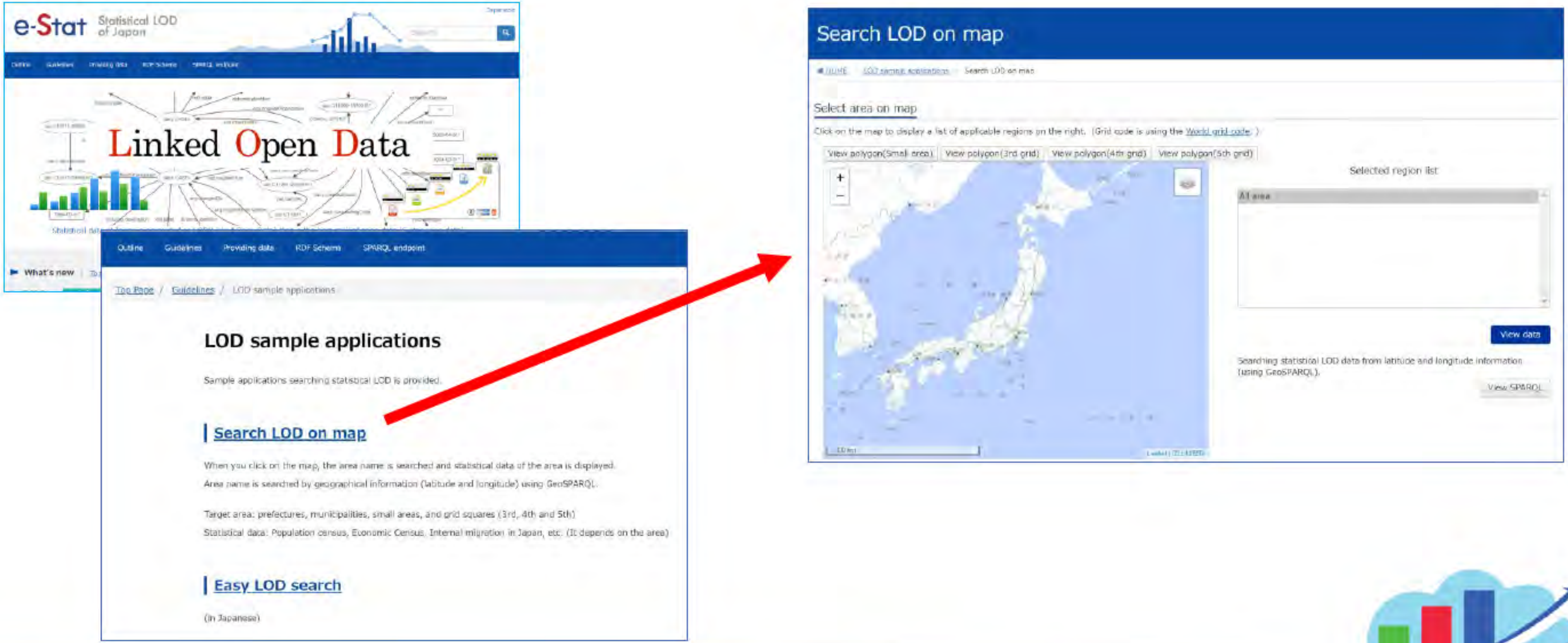
Example of GeoSPARQL *FILTER* expression

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

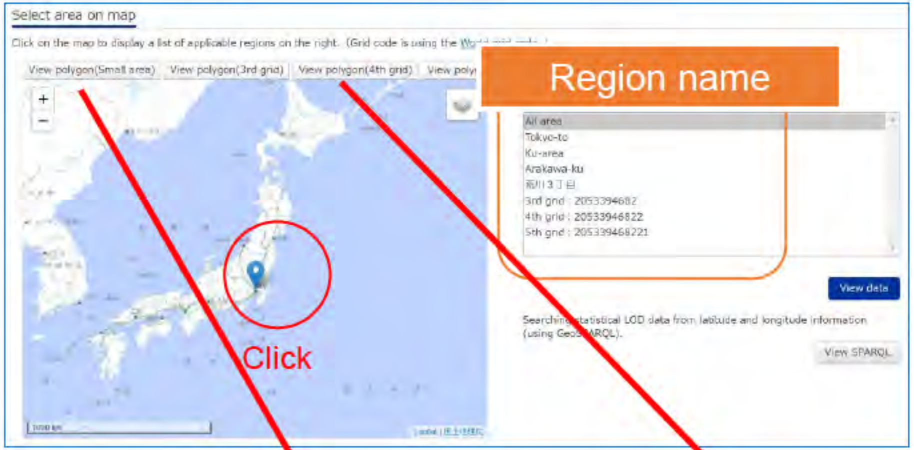


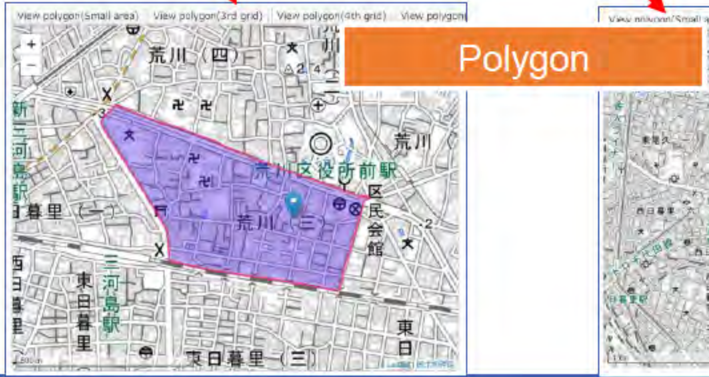
Sample Application

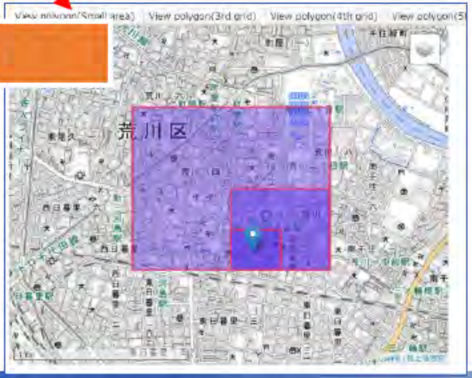
https://data.e-stat.go.jp/lodw/en/



Sample Application








```
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)'^^geo:wktLiteral))
} LIMIT 10
```

GeoSPARQL

Sample Application

select dataset

- Population and Households
  - Population
    - Population by sex - 5th level grid squares
    - Population by age and sex - 5th level grid squares
    - Foreigners by sex - 5th level grid squares
    - households by type of household - 5th level grid squares
    - Private households by size of household - 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 aged household members - 5th level grid squares

Select measure and dimension

- Dataset
  Population by sex - 5th level grid squares
- Measure
  Population
- Dimension

Sex	All (Sex)
	Male
	Female

View count 100 View Data

select measure and dimension

View observation value

- Search result

Sex	Time period	Area	Population(Unit of person)
All (Sex)	2015	5th grid : 205339468221	2178
Male	2015	5th grid : 205339468221	1046
Female	2015	5th grid : 205339468221	1132

- SPARQL

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX qb: <http://purl.org/linked-data/cube#>
PREFIX qb: <http://purl.org/linked-data/cube#>
PREFIX cd-dimension: <http://data.e-stat.go.jp/lod/ontology/crossDomain/dimension/>
PREFIX sdmx-dimension: <http://purl.org/linked-data/sdmx/2009/dimension#>
PREFIX estat-attribute: <http://data.e-stat.go.jp/lod/ontology/attribute/>
PREFIX sac: <http://data.e-stat.go.jp/lod/sac/>
PREFIX dcterms: <http://purl.org/dc/terms/>

SELECT DISTINCT
    ?selectedMeasure1
    ?selectedMeasureUnit1
    ?selectedMultUnit1
```

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**

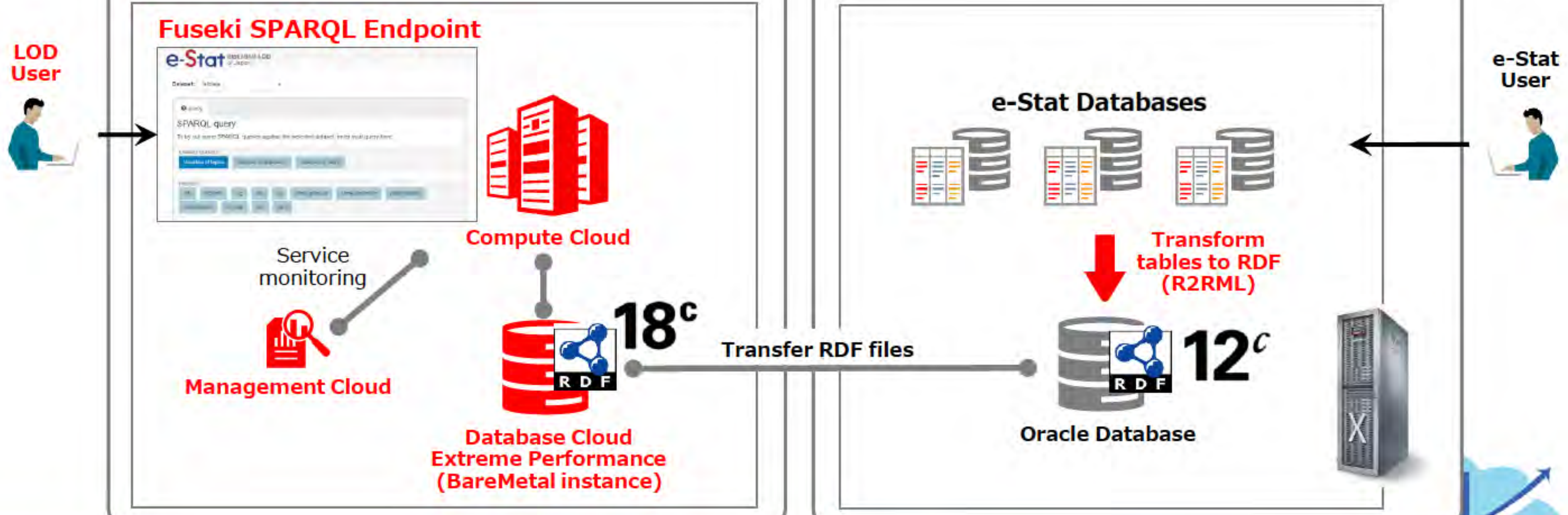
LOD System Architecture

Oracle Gen2 Cloud

- Production env. for **e-Stat "LOD" system**
- **Publishes RDF data** generated on the on-premise Exadata

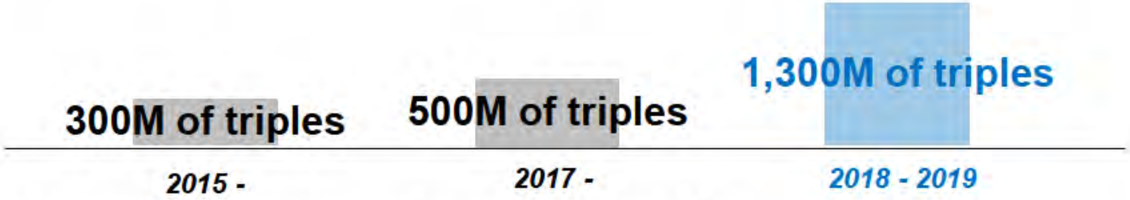
On-Premise (Exadata)

- Production env. for basic **e-Stat systems**
- Publishes statistical data in relational tables
- **Generates RDF data** from relational tables



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)

	Exadata 12cR2
Avg. SPARQL response time	65.74 secs
Number of queries running over 10 sec.	108
Number of queries running over 300 sec.	13

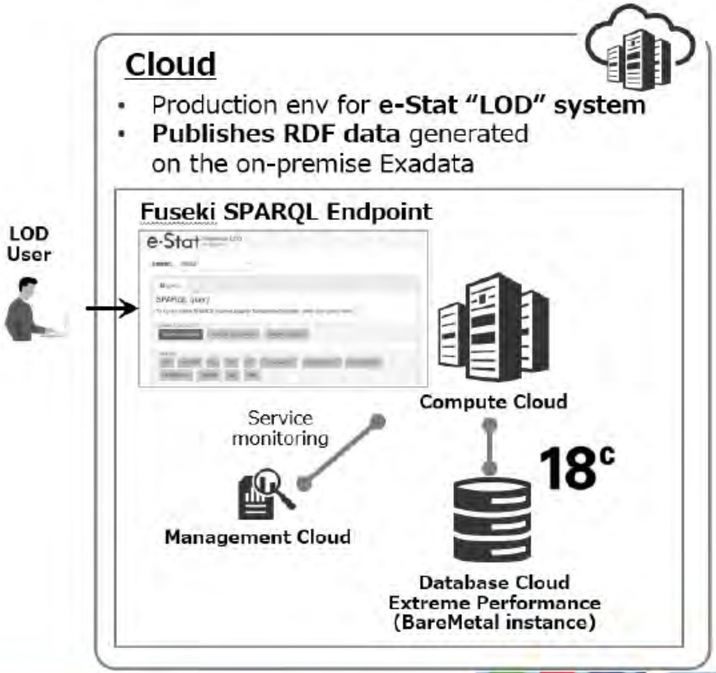
► **We needed a drastic measures to improve SPARQL performance!**

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.



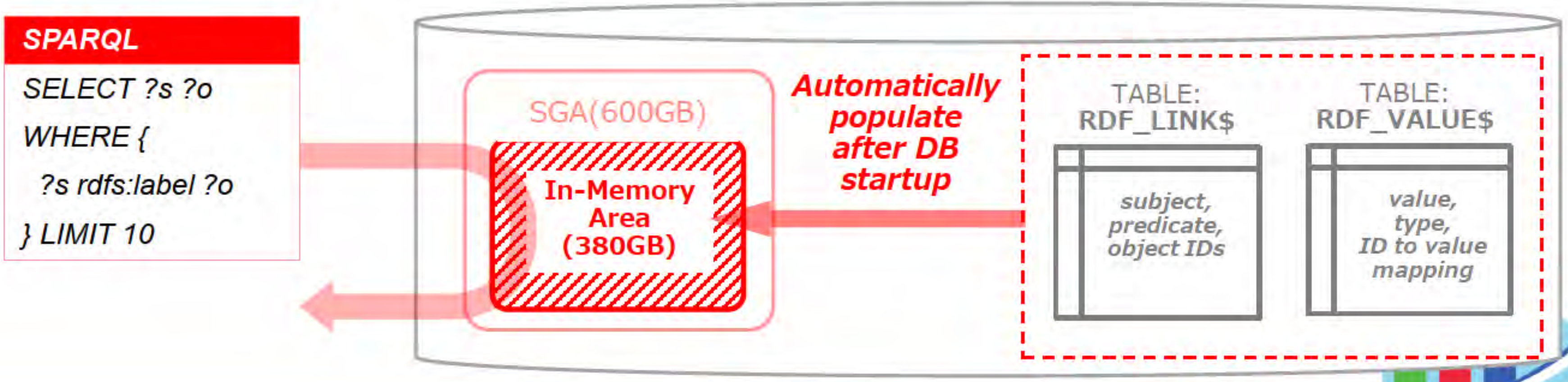
1. Using Database In-Memory Features

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

✓ Enabling DBIM for RDF is very simple.

```
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



2. Partitioning RDF Table by Triple “predicate”

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

subject (variable)

predicate (URI specified)

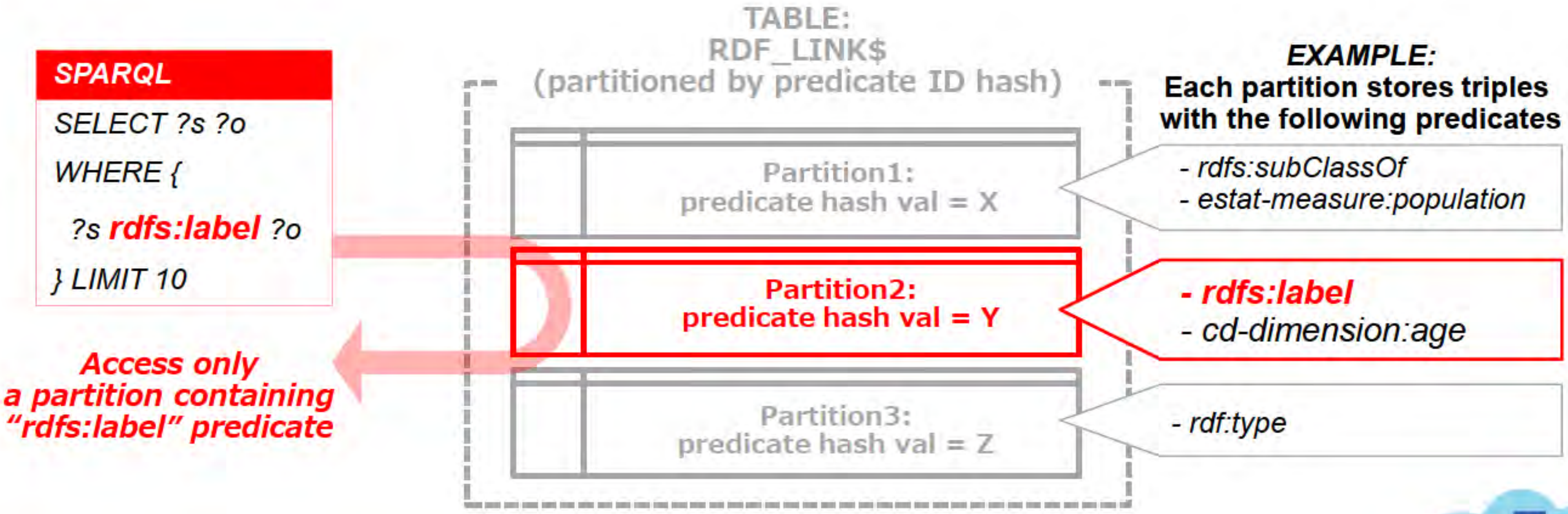
object (variable, literal, URI)

```
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.



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

RDF_LINK$

subject, predicate, object IDs

RDF_VALUE$

value, type, ID to value mapping

Number of rows

Avg. row length

Distinct num of values in each column

e.t.c.

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 correlationship 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_LINK$') FROM DUAL;
```

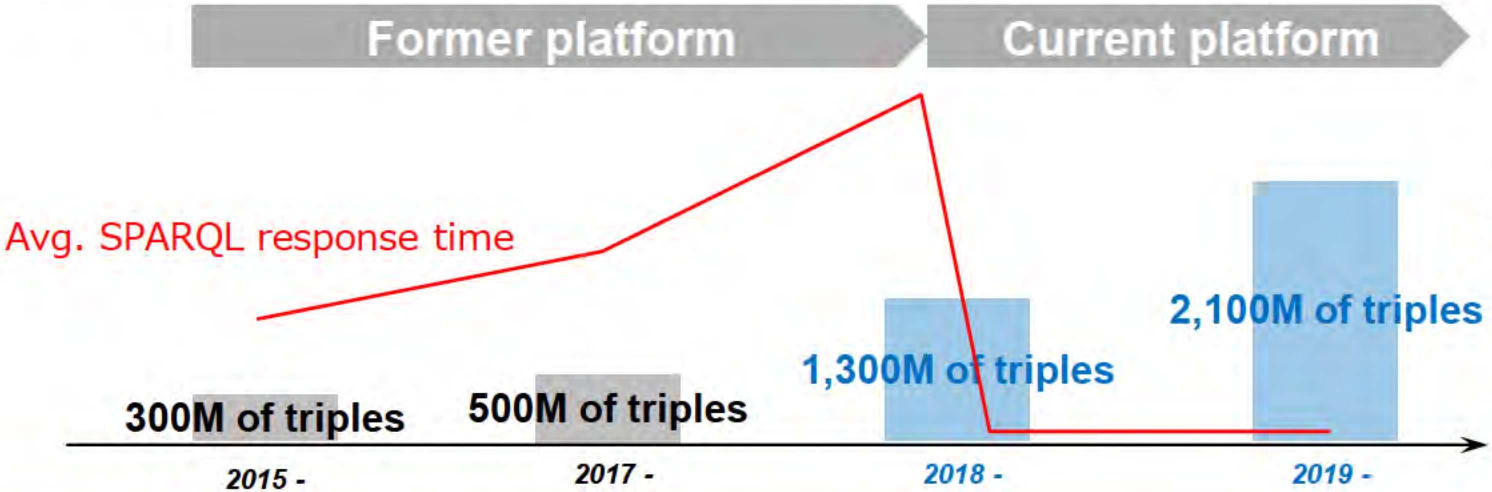
STEP4: Gather statistics by a pre-built procedure SEM_PERF.GATHER_STATS.
The marked column group statistics are automatically gathered.

```
exec SEM_PERF.GATHER_STATS(…);
```

Database Design to Improve SPARQL Performance

How Much SPARQL Performance Improved?

Tested 138 different SPARQL queries against 1.3 billion triples (including several GeoSPARQL)	On-prem Exadata Oracle 12cR2 (former platform)	**Oracle Gen2 Cloud Oracle 18c with tunings**
Avg. SPARQL response time	65.74 sec.	**1.27 sec.**



The performance remained constant, even after triples increased to 2.1 billion!

Summary / Key Takeaways

- ✓ Publication of the 1st 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

ANALYTICS AND DATA SUMMIT 2020

All Analytics. All Data.
No Nonsense.

February 25-27, 2020

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 concept, (https://www.w3.org/TR/vocab-data-cube/).
- Each observation, or data in each cell, is described by dimensions, measures, and attributes.

Measure

Dimension

e.g. Population of 2010 Population census

	Total (Sex)				Male				Female
	...	44 years	45 years	...	...	44 years	45 years	...	...
		[Unit Of Person]	[Unit Of Person]			[Unit Of Person]	[Unit Of Person]		
...	..	...	...	...	...	...	...	...	...
Saitama-city	..	16,130	19,245	...	...	8,293	9,938	...	...
Kawaguchi-city	..	6,582	8,022	...	...	3,526	4,289	...	...
...	..	...	...	...	...	...	...	...	...

Attribute

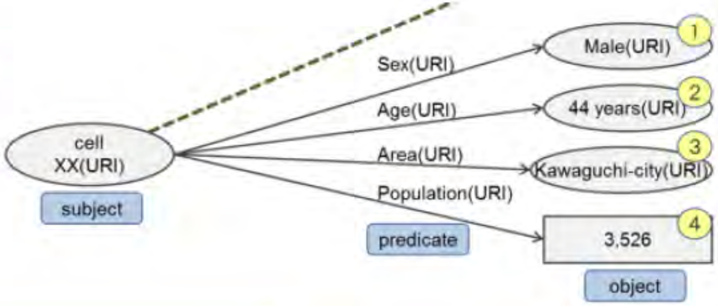
Observation

How We Configured e-Stat LOD

RDF data was generated from statistics tables in our database with **R2RML** (R2RML = **R**DB to **R**DF **M**apping **L**anguage).

Statistics Tables
(Relational tables)

Transform tables to RDF format (R2RML)



Example of R2RML mapping file

```
@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

NSTAC

Administrative Divisions

Japan

prefectural divisions (e.g., Tokyo)

municipal divisions (e.g., Shinjuku-ward)

standard area codes

Statistics

world grid square codes

statistics-specific codes

Grid Squares

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

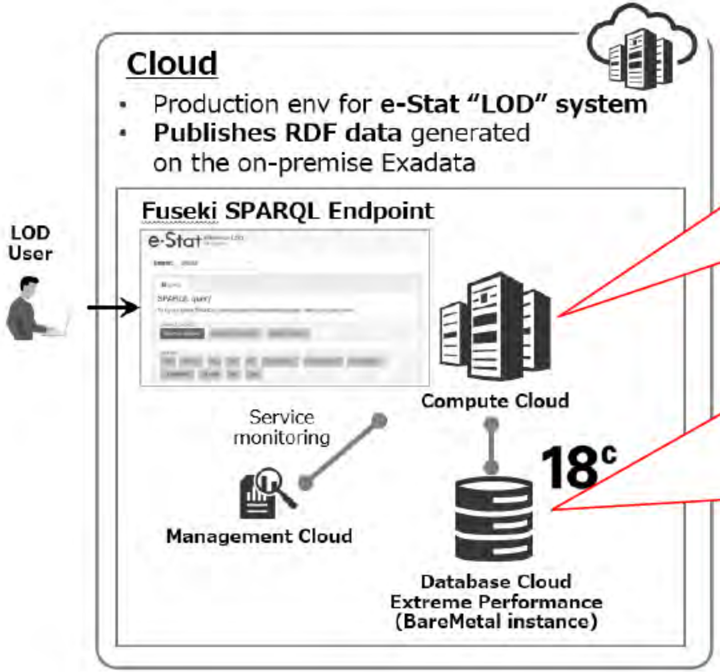
Polygon

Small Areas

Subdivided administrative units of a municipal divisions

Polygon

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

Database Design to Improve SPARQL Performance

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

RDF_LINK$

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

POOL	ALLOC_GB	
1MB POOL	356.8	99% for DATA
64KB POOL	22.9	1% for METADATA

1. Using Database In-Memory Features

<u>Example: Tested SPARQL Query</u>

Queries population census data in Kyoto City

```
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 .
}
```

year	population
"2015"^^xsd:gYear	"1412924"^^xsd:decimal
"2010"^^xsd:gYear	"1408039"^^xsd:decimal

WITHOUT DB In-Memory

7.1 sec

WITH DB In-Memory

0.58 sec

12x faster

Database Design to Improve SPARQL Performance

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.