

An Approach to Handling Spatio-Temporal Contexts on Linked Open Data for Points of Interest

Nurul Firdaus 1*, Naoki Fukuta 2**

* Department of Informatics, Vocational School, Universitas Sebelas Maret

** College of informatics, Academic Institute, Shizuoka University

*nurul.firdaus@staff.uns.ac.id

Info Artikel

Keywords :

spatio-temporal data, ontology,
linked open data

Date of Article

Submitted : 15 November 2022

Revised : 20 November 2022

Accepted : 30 November 2022

Abstract

Many people and organizations, including governments, for example, publicly disseminate data in various ways and for various reasons. For publishing data on the Semantic Web, the Linked Data principles have become the norm. However, numerous difficult jobs and problems must be resolved while linking the vast amount of data by utilizing Linked Open Data (LOD) sources. This thesis outlines our contributions to understanding spatiotemporal contexts to address those issues. Creating various housing options and opportunities is one of the core ideas behind smart growth. As a result, geographical elements like Points of Interest (POIs) affected the real estate market and buyers' choices. Using data from diverse sources, we provide a domain-specific strategy in this work. When estimating instance value for POIs, this work attempts to handle spatiotemporal contexts. We refer to the POI evaluation events that include spatiotemporal notions of the target real estate site as spatio-temporal contexts. We gather the data and format it into a common format to forecast the instance value of an object. Then, we construct a domain-specific ontology to evaluate the application of control in city planning. The instances are then filled using Federated SPARQL Query using data from Endpoints. The design of our prototype system to manage spatio-temporal contexts on LOD for POIs has now been prepared and put into practice.

1. INTRODUCTION

In the next decade, the Smart Cities project will transform citizens' lives, work, and play. In Smart Cities, development considers master planning, urban planning, and utilization of geospatial information for real estate development (i.e., Smart Growth). One of the Smart Growth principles¹ is to create a range of housing opportunities and choices. Therefore, geospatial factors such as Points of Interest (POIs) influence the real estate market and buyer's decision.

Points of Interest (POI) is the most important indicator [1] in real estate decision-making. Hence, the real estate domain is dynamic and complex, where decision-makers usually rely on fluctuating hands changing yearly [2]. It affects users as decision-makers in considering the fluctuating hands to set up the investment. In this study, we believe geospatial indicators as the factors to determine real estate values, especially for POIs' assessment.

To collect the observation data about the POIs' assessment factors of geospatial indicators, integrated data from various sources are required. In [3], it discussed that the Semantic Web has implications for decision-making support, since the filled and unfulfilled promises, are derived from the earlier vision of the Semantic Web and research opportunities. Therefore, Linked Data technologies support intelligent data integration on the Web [4]. Open Data, especially Open Government Data (OGD) [5] [6], refers to a movement that was initiated by "The Memorandum on Transparency and Open Government"². OGD is the major knowledge-sharing movement toward adopting Linked Data for decision-making. The Japanese Government also promotes the Open Data initiative to improve people's lives and stimulate corporate activities, thereby contributing to the social and economic development of Japan³. For the transparency of the real estate market, the Japanese Ministry of Land, Infrastructure, Transport, and Tourism (MLIT) published real estate transaction-price information via Land General Information System⁴. Since they provided to promote reliability and transparency of the market, we can use their data with the support of Linked Open Data (LOD) sources (i.e., GeoNames) to design an ontology approach that presents a short-term and high-resolution analysis.

In this section, we list the contributions of our research studies as follows: we propose a way to design a prototype system with a practical LOD schema that could properly handle spatio-temporal contexts. We build a domain-specific ontology to handle the spatio-temporal representation on LOD

2. METHOD

The research method in building an approach to handling spatio-temporal describes as follows:

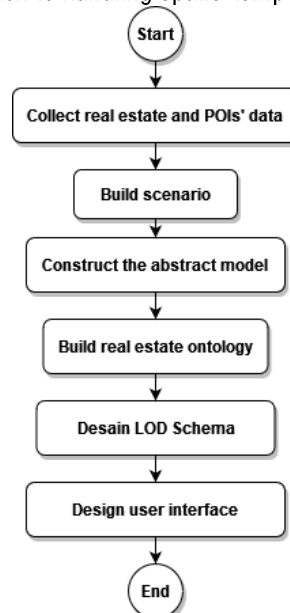


Figure 2.1 The research method in building an approach

¹Smart Growth Principles.

<http://smartgrowth.org/smart-growth-principles/>

²The Official Home of UK Legislation: <http://www.legislation.gov.uk>

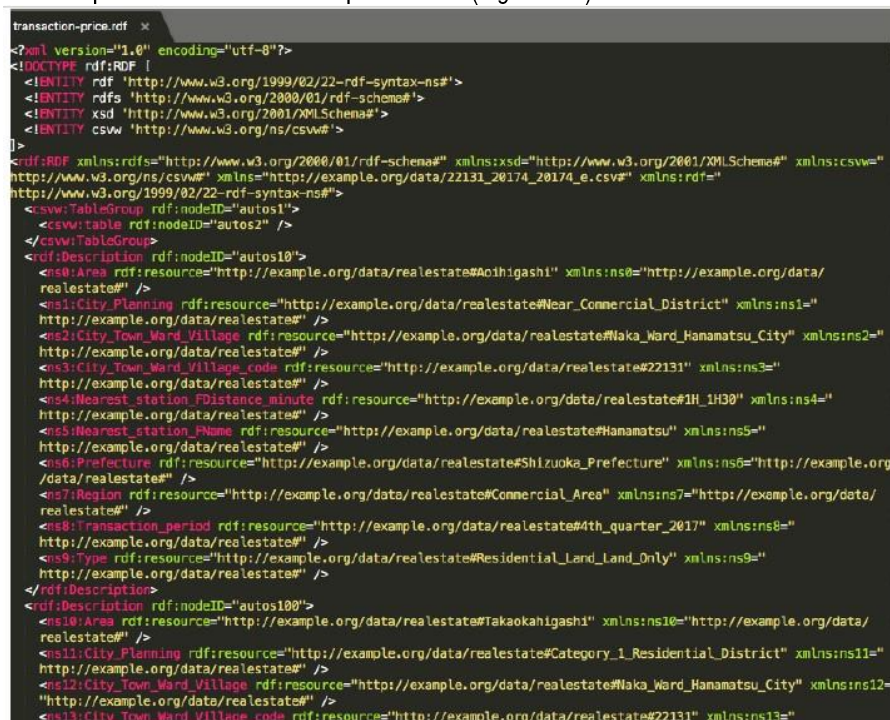
³Open Data Japan: <http://www.data.go.jp/?lang=english>

⁴http://www.land.mlit.go.jp/webland_english/servlet/MainServlet

3. OPEN DATA AND LOD SOURCES

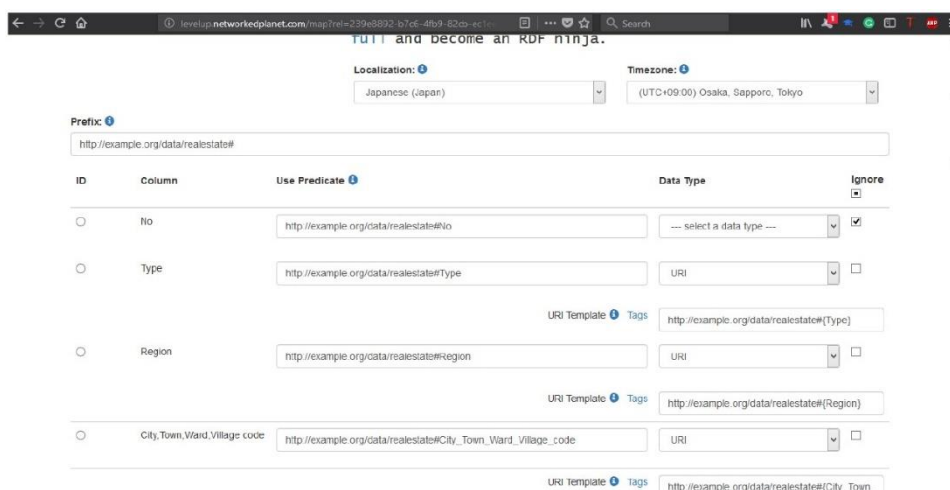
In this section, we briefly introduce Open Data and LOD sources on the geospatial domain that we consider and use in this paper. To fully benefit from Open Data, it is crucial to put information and data into contexts that create new knowledge and enables effective services and applications. We collected observation instances of real estate and POIs from GeoNames API and real estate transaction-price data. Then, we convert the CSV file of real estate transaction-price data into Resource Description Framework (RDF)⁵ using online RDF converter⁶. The step in converting from CSV file to RDF file (Figure 3.1) are below:

- Download real estate transaction-price data in CSV format from Land General Information System Website
- Drag and drop or click CSV file to the online CSV to RDF Converter website
- Configure how we map the values into the outputted RDF (Figure 3.2)



```
transaction-price.rdf
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE rdf:RDF [
  <RDF:RDF rdf:base="http://www.w3.org/1999/02/22-rdf-syntax-ns#" />
  <RDF:RDF rdf:base="http://www.w3.org/2000/01/rdf-schema#" />
  <RDF:RDF xsd="http://www.w3.org/2001/XMLSchema#" />
  <RDF:RDF csvw="http://www.w3.org/ns/csvw#" />
]
<rdf:RDF xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#" xmlns:xsd="http://www.w3.org/2001/XMLSchema#" xmlns:csvw="http://www.w3.org/ns/csvw#" xmlns="http://example.org/data/22131_20174_e.csv#" xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#" />
  <csvw:TableGroup rdf:nodeID="autos1">
    <csvw:table rdf:nodeID="autos2">
      </csvw:table>
    </csvw:TableGroup>
    <rdf:Description rdf:nodeID="autos10">
      <ns0:Area rdf:resource="http://example.org/data/realstate#Aochigashi" xmlns:ns0="http://example.org/data/realstate#" />
      <ns1:City_Planning rdf:resource="http://example.org/data/realstate#Near_Commercial_District" xmlns:ns1="http://example.org/data/realstate#" />
      <ns2:City_Town_Village rdf:resource="http://example.org/data/realstate#Naka_Ward_Hanamatsu_City" xmlns:ns2="http://example.org/data/realstate#" />
      <ns3:City_Town_Village_code rdf:resource="http://example.org/data/realstate#22131" xmlns:ns3="http://example.org/data/realstate#" />
      <ns4:Nearest_station_FDistance_minute rdf:resource="http://example.org/data/realstate#1H_1H30" xmlns:ns4="http://example.org/data/realstate#" />
      <ns5:Nearest_station_PName rdf:resource="http://example.org/data/realstate#Hanamatsu" xmlns:ns5="http://example.org/data/realstate#" />
      <ns6:Prefecture rdf:resource="http://example.org/data/realstate#Shizuoka_Prefecture" xmlns:ns6="http://example.org/data/realstate#" />
      <ns7:Region rdf:resource="http://example.org/data/realstate#Commercial_Area" xmlns:ns7="http://example.org/data/realstate#" />
      <ns8:Transaction_period rdf:resource="http://example.org/data/realstate#4th_quarter_2017" xmlns:ns8="http://example.org/data/realstate#" />
      <ns9:Type rdf:resource="http://example.org/data/realstate#Residential_Land_Land_Only" xmlns:ns9="http://example.org/data/realstate#" />
    </rdf:Description>
    <rdf:Description rdf:nodeID="autos100">
      <ns10:Area rdf:resource="http://example.org/data/realstate#Takaokahigashi" xmlns:ns10="http://example.org/data/realstate#" />
      <ns11:City_Planning rdf:resource="http://example.org/data/realstate#Category_1_Residential_District" xmlns:ns11="http://example.org/data/realstate#" />
      <ns12:City_Town_Village rdf:resource="http://example.org/data/realstate#Naka_Ward_Hanamatsu_City" xmlns:ns12="http://example.org/data/realstate#" />
      <ns13:City_Town_Village_code rdf:resource="http://example.org/data/realstate#22131" xmlns:ns13="http://example.org/data/realstate#" />
    </rdf:Description>
  </rdf:RDF>
```

Figure 3.1 The Result of RDF Converter in RDF/XML Format



levelup.networkedplanet.com/map?lat=23.968932&lon=138.469482&zoom=12

Full and become an RDF ninja.

Localization: Japanese (Japan) Timezone: (UTC+09:00) Osaka, Sapporo, Tokyo

Prefix: http://example.org/data/realstate#

ID	Column	Use Predicate	Data Type	Ignore
<input type="radio"/>	No	http://example.org/data/realstate#No	--- select a data type ---	<input checked="" type="checkbox"/>
<input type="radio"/>	Type	http://example.org/data/realstate#Type	URI	<input type="checkbox"/>
		URI Template: Tags	http://example.org/data/realstate#{Type}	
<input type="radio"/>	Region	http://example.org/data/realstate#Region	URI	<input type="checkbox"/>
		URI Template: Tags	http://example.org/data/realstate#{Region}	
<input type="radio"/>	City, Town, Village code	http://example.org/data/realstate#City_Town_Village_code	URI	<input type="checkbox"/>
		URI Template: Tags	http://example.org/data/realstate#{City_Town_Village_code}	

Figure 3.2 Online RDF Converter

⁵<https://www.w3.org/TR/rdf-schema/>

⁶<http://levelup.networkedplanet.com/>

4. PROTOTYPE SYSTEM

The prototype system requires Human-in-the-loop, which means it requires human interaction in the contexts. First, the user, as the decision maker, inputs the location query, which is the assessment point that the user wants to assess. Then, the system gets the location's geo-coordinate (long, lat). After that, the system sort and select the match data containing the location information. Furthermore, the system presents the assessment status of the location.

4.1. Prototype Overview

The mashup system develops to have a web-based user interface with data collected using REST web services from Geonames API⁷. We implemented a mechanism that uses various frameworks. TDB⁸ is Apache JENA's component storing RDF Triple observation data. We can also use Google Geocoding API to get the geo-coordinate of POIs' addresses and Google Map Geolocation API to pin the map. Ontology development is outside the prototype system (dash square, see Figure 4.1) using knowledge engineering tools in Section 4.6.2. The prototype system used Protégé⁹ [7] [8] [9] and CHRONOS Ed¹⁰ [80], which is a tool for handling temporal ontologies in Protégé. The basic structure of our prototype system shows in Figure 4.1. Otherwise, in retrieving the data, we employed SPARQL Query [10] [11] for querying diverse data sources, and we employed the Representational State Transfer (REST), an architectural style for retrieving resources over HTTP protocol, to retrieve the data from GeoNames Web service. Thus, the prototype system uses a client-server architecture model, as described in the following subsections.

On the server side, the server develops in Java 7 EE Web, and we employ Java framework for building SemanticWeb and Linked Data applications (i.e., JENA API) and Web API for handling connections to the database (i.e., RDF Triplestore) as well as connections to the browser client for generating web pages. Also, Java Virtual Machine (JVM) to run the programs. We employ Apache JENA Fuseki Server to create a local SPARQL Endpoint to store the RDF and OWL data. Further, the prototype system employs JENA RDF API to create and read RDF graphs. Then, serialize the triples using RDF/XML format and JENA ARQ (SPARQL) that support remote Federated SPARQL Query to interlink between local SPARQL Endpoints and External SPARQL Endpoints.

On the client side, we used GoogleMAPJavaScript API and GoogleMAPGeocoding API to build the UI that handles an event and displays geospatial information on the map. We also employ AJAX, a set of Web development techniques using many Web technologies on the client side to create asynchronous Web applications. With AJAX, Web applications can send and retrieve data from a server asynchronously (in the background) without interfering with the display and behavior of the existing page. By decoupling the data interchange layer from the presentation layer, AJAX allows Web pages and, by extension, Web applications, to change the content dynamically without the need to reload the whole page.

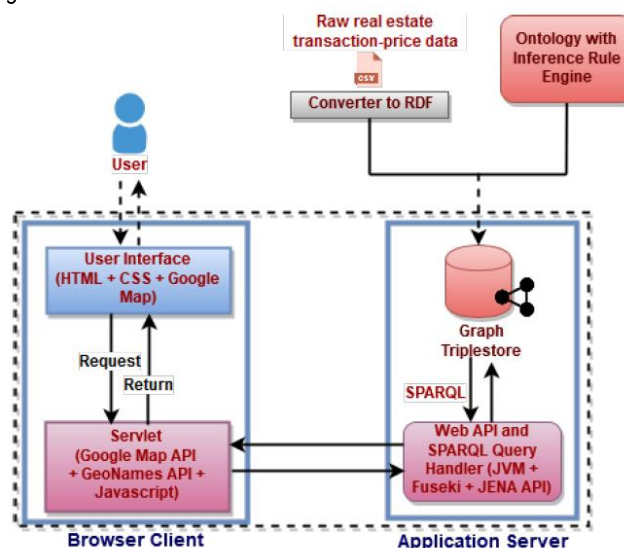


Figure 4.1 The Basic Structure of Prototype System [12]

⁷<http://api.geonames.org/>

⁸<http://jena.apache.org/documentation/tdb/>

⁹<https://protege.stanford.edu>

¹⁰<http://www.intelligence.tuc.gr/prototypes.php>

In order to retrieve the desired data from Endpoint or database. While implementing the prototype system, we consider two types of data retrieval methods (red squares) implemented in our prototype system based on the data sources, the data source's endpoints, and the data format. Figure 4.2 shows the methods for retrieving the data as follows:

Data sources	GeoNames + Google MAP Data	Real Estate Ontology + Transaction-price Data	LinkedGeoData (Amenity Data)
Endpoint	WEB API/WEB Service	Local SPARQL Endpoint / Local RDF Triplestore	External SPARQL Endpoints
Method	GET Request	SPARQL Query / GeoSPARQL Query	
Data Format	JSON	RDF/XML	

Figure 4.2 Data Retrieval Methods

- Retrieving the Data by GET Request from Web API
We have retrieved GeoNames and Google Map data through AJAX's GET request. Since GeoNames and Google MAP have their services and libraries packages for connecting to their data, thus we have utilized their service to get the data.
- Retrieving the Data by SPARQL Query from SPARQL Endpoints
Many Web infrastructures are accustomed to representing information in HTML or, more commonly, in XML. For this reason, the W3C has recommended using an XML serialization of RDF called RDF/XML. Thus, we have retrieved the OWL file that we stored in RDF/XML syntax of real estate ontology that stored the local SPARQL Endpoint by SPARQL Query. Since we employed Apache JENA Fuseki Server that provides IRI for-SPARQL Endpoint, we call the IRI through SPARQL SERVICE. And we have retrieved the instances of Amenity's class in LinkedGeoData from their SPARQL Endpoint to populate POIs' class in real estate ontology using SPARQL CONSTRUCT AND SERVICE through queries.

When consuming web services such as GeoNames, we have to consider the limitation on the number of possible requests or obtained results for most services.

4.2. The Implementation of Prototype System

In this section, we briefly explain the stages of implementing our prototype system and what kind of type data retrieval, and the method we implemented in the prototype system. Figure 4.3 depicts our prototype system implemented using JavaScript for query interface and user interface event and Java JENA framework [13] for building semantic web and linked data application. We have prepared a Local SPARQL Endpoint using Fuseki Server to remote the data and to communicate with real estate ontology and real estate transaction-price data over HTTP protocol. The users input the location queries through the query interface then the queries would process in the query executor and parser through SPARQL and HTTP protocol. In the query executor, our system executes the query to populate and remote our ontology stored in the Local SPARQL Endpoint with instances through the SERVICE keyword of external SPARQL endpoints. Then, the output and the results in JSON format would process in the results visualizer through the HTTP protocol on the map. The SERVICE keyword extends SPARQL 1.1 to support queries that merge data distributed across the web [14].

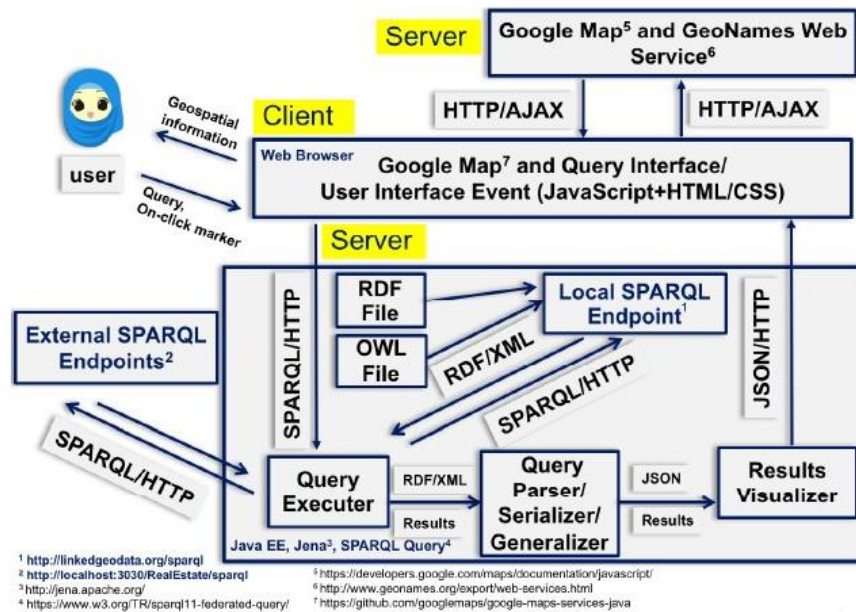


Figure 4.3 The Implementation of Prototype System

4.2.1. Select and Convert the Data

We manually selected the sources used in building the prototype system in the first stage. Then, we converted the open data to standardized in the RDF format. We have prepared local SPARQL Endpoints for our domain-specific ontology using Apache JENA Fuseki.

Listing 1: An Example of SPARQL SELECT to query transaction-price data stored in local SPARQL Endpoint

```
SELECT *
WHERE {
  GRAPH<http://localhost:3030/RealEstate/data/
TransactionPriceData>
  {?s ?p ?o }}
LIMIT 10;
```

Subject	Predicate	Object
_:b95	xmlns:City_Planning	xmlns:Commercial_District
_:b95	xmlns:City_Town_Ward_Village	xmlns:Naka_Ward_Hamanatsu_City
_:b95	xmlns:City_Town_Ward_Village_code	xmlns:22131
_:b95	xmlns:Nearest_station_FDdistance_minute	xmlns:4
_:b95	xmlns:Nearest_station_FName	xmlns:Hamanatsu
_:b95	xmlns:Prefecture	xmlns:Shizuoka_Prefecture
_:b95	xmlns:Region	xmlns:Commercial_Area
_:b95	xmlns:Transaction_period	xmlns:4th_quarter_2017
_:b95	xmlns:Type	xmlns:Residential_Land_Land_and_Building
_:b95	xmlns:Use	"House, Shop"
_:b8	rdf:type	<http://www.w3.org/ns/csvw#Row>
_:b8	<http://www.w3.org/ns/csvw#describes>	_:b96
_:b8	<http://www.w3.org/ns/csvw#rownum>	55
_:b8	<http://www.w3.org/ns/csvw#url>	<http://example.org/data/22131_20174_20174_e.csv#row=56>
_:b96	xmlns:Area	xmlns:Chuo
_:b96	xmlns:City_Planning	xmlns:Commercial_District
_:b96	xmlns:City_Town_Ward_Village	xmlns:Naka_Ward_Hamanatsu_City
_:b96	xmlns:City_Town_Ward_Village_code	xmlns:22131
_:b96	xmlns:Nearest_station_FDdistance_minute	xmlns:10
_:b96	xmlns:Nearest_station_FName	xmlns:Hamanatsu
_:b96	xmlns:Prefecture	xmlns:Shizuoka_Prefecture
_:b96	xmlns:Transaction_period	xmlns:4th_quarter_2017
_:b96	xmlns:Type	xmlns:Pre_owned_Condominiums_etc

Figure 4.4 The Result of SPARQL SELECT

Listing 1 depicts SPARQL Query SELECT to select all the variables of transaction-price data stored in the local SPARQL Endpoint using SERVICE keywords and GRAPH clause to fetch a remote graph that is available or active in the local SPARQL Endpoint¹¹. In our local SPARQL Endpoint, we have two active graphs, which are <http://localhost:3030/RealEstate/data/TransactionPriceData> and

¹¹ <http://localhost:3030/RealEstate/sparql>

<http://localhost:3030/RealEstate/data/RealEstateontology>.

The SELECT query form returns variables and their bindings directly. In this SPARQL SELECT (*) return all variables ?s ?p ?o (?s = subject, ?p = predicate, ?o = object) as triples from Transaction-PriceData graph. In this bindings subject is 'blank node' which is mean the subject of several RDF triples. While LIMIT keyword is used to limit the return of triples. The result of the queries is triples in Figure 4.4.

In the Table 4.1 is data size from the local SPARQL Endpoint. We have 2509 triples from real estate ontology and 1333 triples from transaction price data.

Table 4.1: The Data Size of Local SPARQL Endpoints

Graph Name	Triples
<http://localhost:3030/RealEstate/data/RealEstateontology>	2509
<http://localhost:3030/RealEstate/data/TransactionPriceData>	1333

4.2.2. Populate and Integrate the Instances into the Ontologies:

Since manually entering instances to ontology is time-consuming, we try to automatically populate our domain-specific ontology with instances from external SPARQL Endpoints. Listing 4 depicts SPARQL Query CONSTRUCT to populate POIs' instances from the Amenities class. Listings 5 depicts SPARQL Query to populate Hospital instances from the Hospital class of external SPARQL Endpoints [1] using SERVICE keywords into our domain-specific ontology. The SPARQL CONSTRUCT keyword introduces a graph pattern to be used as a template (i.e., ?POIs rdf:type re:POIs) and (i.e., ?POIs rdf:type re:Hospital) in constructing a new graph (i.e., ?POIs rdf:type lgdo:Amenity) and (i.e., ?POIs rdf:type lgdo:Hospital), based on results from the old data graph. This SPARQL CONSTRUCT specifies rdf:type triples. The result of the queries is in RDF/XML format in Figure 4.5 and Figure 4.6. The type of Linked-GeoData's data on the OpenStreetMap (OSM)[1] data. Nodes represent points on earth and have longitude and latitude values [15].

Listing 2: An Example of SPARQL CONSTRUCT to populate POIs' instances from external SPARQL Endpoints

```
CONSTRUCT {
  ?POIs rdf:type re:POIs .
}
WHERE { ?POIs rdf:type lgdo:Amenity . } ;
```

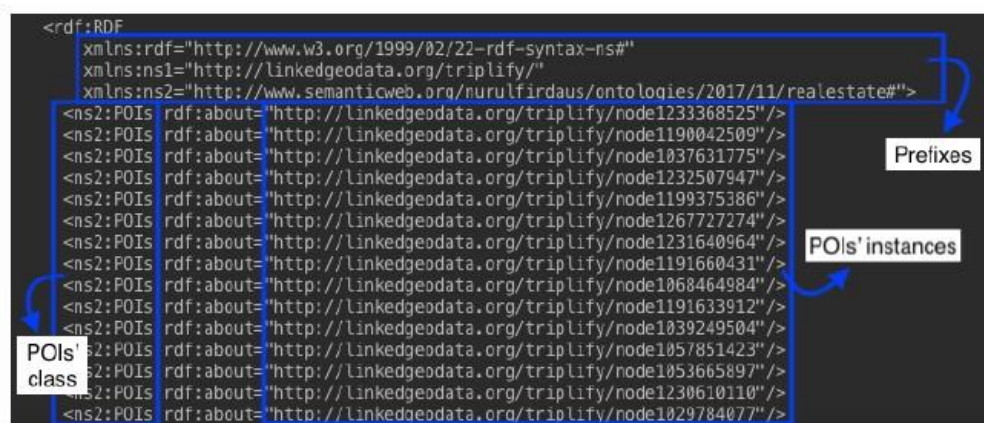


Figure 4.5: The Result of SPARQL CONSTRUCT in RDF/XML Format

Listing 3: An Example of SPARQL CONSTRUCT to populate Hospital instances from external SPARQL Endpoints

```
CONSTRUCT {
  ?hospital rdf:type re:Hospital .
}
SERVICE <http://linkedgeo.org/sparql>
{
  ?hospital rdf:type lgdo:Hospital .
};
```

Table 4.2 shows the performance level our program could reach 100 times of execution. Our program could obtain 50001 POIs' instances of Amenity and Hospital with building time 146384.78/ms for amenity (details in Figure 4.7) and 37566.32/ms for Hospital (details in Figure 4.8).

The datasets do publish as Linked Data (e.g., GeoNames, Event, Time, WGS84). Since LOD sources is a Linked Data application, SPARQL Endpoint (<http://linkedgeodata.org/sparql>) handles geographical ontologies. Via this endpoint, it is possible to execute GeoSPARQL queries and retrieve RDF or JSON content directly. Our system can query an external dataset to retrieve the data via Federated SPARQL query, as depicted in Figure 4.5. Figure 4.9 is an example GeoSPARQL query to retrieve POIs' data in a radius of 100 meters from the target points (i.e., long, lat). Otherwise, Figure 4.10 is the results or output in JSON format or directly in HTML format, as shown in Figure 4.11.

A prototype system needs to have access to both the local and the external endpoints in order to perform an interlinking.

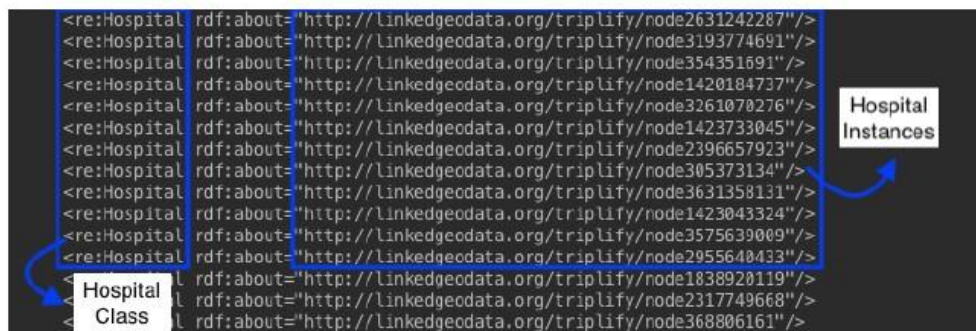


Figure 4.6: The Instances of Hospital

Table 4.2: Performance Level of SPARQL CONSTRUCT

Description	POIs' Instances	Building Time /ms (*)
Type: SPARQL Query CONSTRUCT SERVICE: http://linkedgeodata.org/sparql Task: Obtained the instances from amenity's class	50001	146384.78
Type: SPARQL Query CONSTRUCT SERVICE: http://linkedgeodata.org/sparql Task: Obtained the hospital instances from hospital class	50001	37566.32
(*) The average values on 100 times of execution		

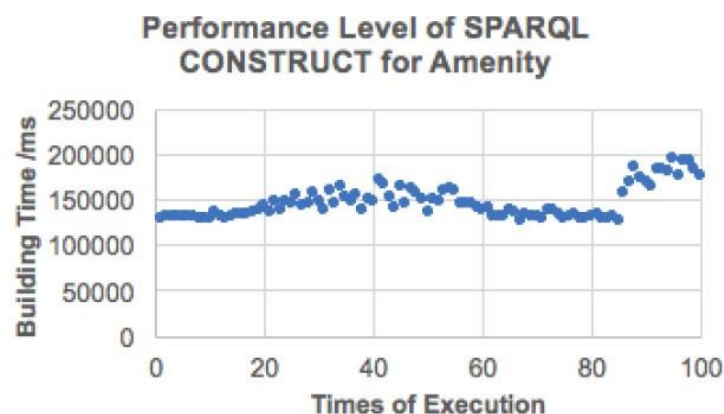


Figure 4.7: Chart of Query Execution for the Instances of Amenity

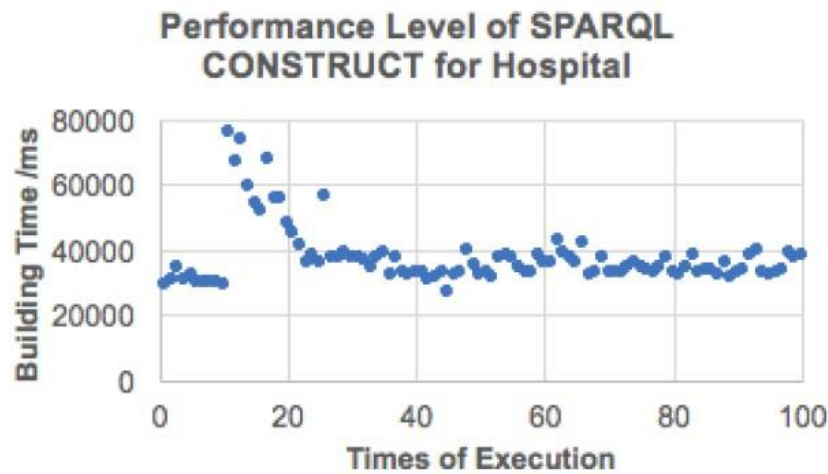


Figure 4.8: Chart of Query Execution for the Instances of Hospital

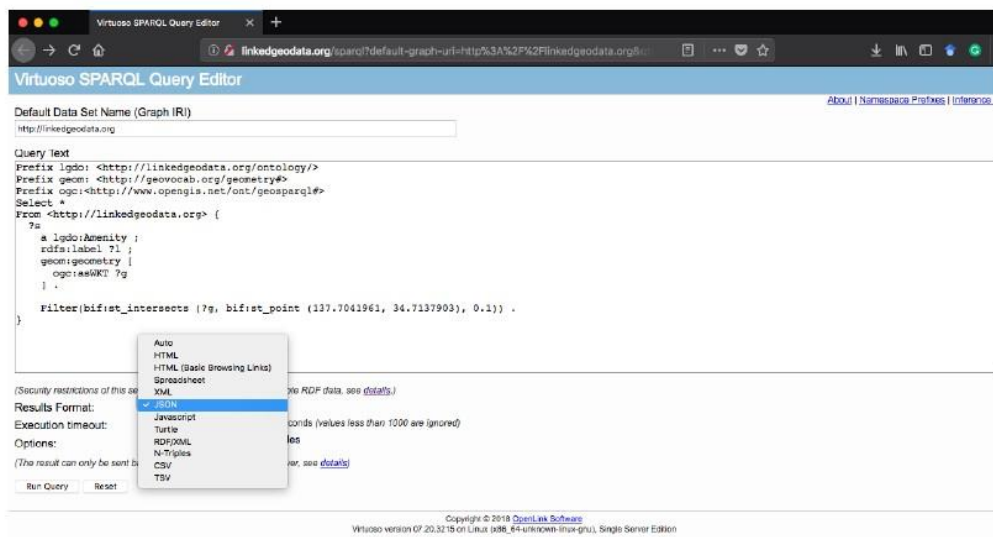


Figure 4.9: An Example of GeoSPARQL query in External SPARQL Endpoint

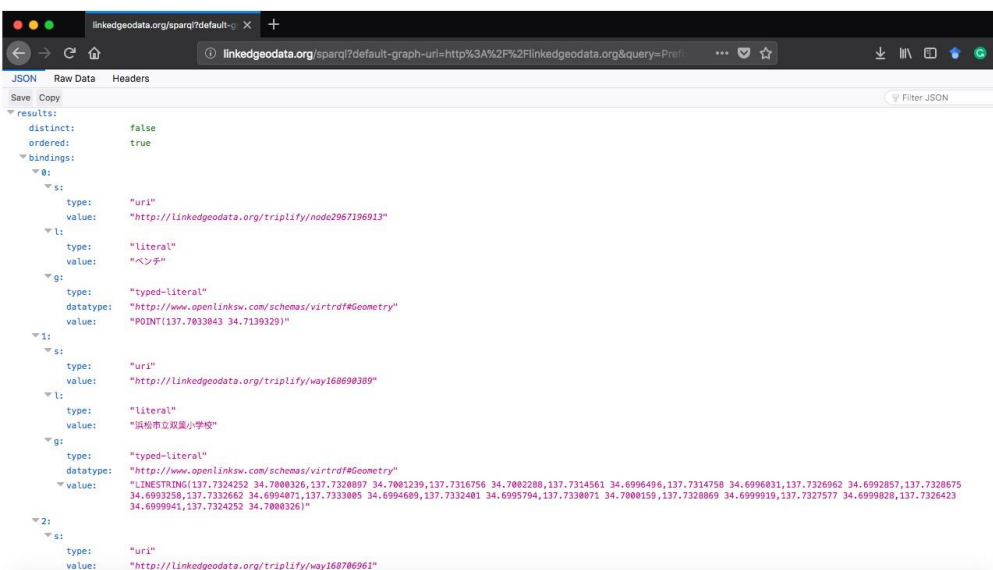


Figure 4.10: The Output of GeoSPARQL query in JSON format


4.2.3. Transform, Retrieve, and Visualize the Data into Map-friendly Format

In this stage, we try to transform data from machine-readable (i.e., JSON) into human-readable meta-data (i.e., HTML/web page) to visualize it on the map.

For that purpose, we try to retrieve the data through WEB API using REST architecture that is available in the GeoNames (i.e., POIs' data shown in Figure 4.12 and Figure 4.13) and Google MAP (i.e., Place library 10) services to perform geocoding and reverse geocoding. Geocoding converts a physical address or location into geo-coordinates (latitude/longitude), and reverse geocoding converts a lat/long to a physical address or location.

In [16] demonstrates how web 2.0 data sources like Amazon, Google, and Yahoo can integrate into the Semantic Web. Our idea is to retrieve spatial data from google through Web API. API is implementing by writing function calls in the model, which provides the linkage to the required subroutine for execution. More and more APIs appear as web services (e.g., GeoNames web service, Google Maps). When used in web development, an API is typically a defined set of Hypertext Transfer Protocol (HTTP) request messages, along with a definition of the structure of response messages, usually in XML or JavaScript Object Notation (JSON) format. While "Web API" is virtually a synonym for web service, the recent trend (so-called Web 2.0) has been moving away from Simple Object Access Protocol (SOAP) based services toward more direct Representational State Transfer (REST) style communications [17].

Web APIs combines multiple services into new applications, known as mashups. It becomes possible to query external SPARQL endpoints and use this information. We use GeoNames API and Google Maps to retrieve the geographical attributes and instances, especially POIs, and display them on the map. The use of a Web API is the Google Map API to search for locations in Google Map Place libraries, using a free text search engine. In addition, the use of GeoNames API to find nearby POIs around the target real estate locations. The Google Map API is used to search for locations in Google Place dataset using a free text search engine. In this way, it becomes possible to automatically find different locations in the Google Place dataset (e.g., Shijimizuka, Hamamatsu) and create connections between these locations in a dataset on the one hand and Google Maps on the other hand.



s	l	g
http://linkedgeoata.org/triiplyf/node2967196913	"ベンチ"	"POINT(137.7033043 34.7139329)^^<http://www.openlinksw.com/schemas/virtrdf#Geometry>
http://linkedgeoata.org/triiplyf/way168690389	"浜松市立双葉小学校"	"LINESTRING(137.7324252 34.7000326,137.7320897 34.7001239,137.7316756 34.7002288,137.7314561 34.6996496,137.7314758 34.6996031,137.7326962 34.6992857,137.7328675 34.6993258,137.7332662 34.6994071,137.7333005 34.6994609,137.7332401 34.6995794,137.7330071 34.7000159,137.7328869 34.6999919,137.7327577 34.6999828,137.7326423 34.6999941,137.7324252 34.7000326)^^<http://www.openlinksw.com/schemas/virtrdf#Geometry>
http://linkedgeoata.org/triiplyf/way168706961	"静岡大学"	"LINESTRING(137.7154753 34.723458,137.7153975 34.7235155,137.7154286 34.7236369,137.7157362 34.725065,137.7162056 34.7270336,137.7196189 34.7265089,137.7196888 34.7264035,137.7194464 34.7252565,137.7193377 34.7252093,137.7193214 34.7251245,137.7193105 34.7250282,137.71938 34.7249438,137.7189037 34.722934,137.7188298 34.7229021,137.7154753 34.723458)^^<http://www.openlinksw.com/schemas/virtrdf#Geometry>
http://linkedgeoata.org/triiplyf/way177924751	"八幡神社"	"LINESTRING(137.7362239 34.7151121,137.7362382 34.7150956,137.7365707 34.7150651,137.7372041 34.7149152,137.7372408 34.7149086,137.7372404 34.7148928,137.7373185 34.7148762,137.7373301 34.7163014,137.7363573 34.7162976,137.7363364 34.7162912,137.736307 34.7162694,137.73629 34.716225,137.7362573 34.7159839,137.7362239 34.7151121)^^<http://www.openlinksw.com/schemas/virtrdf#Geometry>
http://linkedgeoata.org/triiplyf/way192831119	"浜松市大原浄水場"	"LINESTRING(137.7376932 34.795136,137.7375645 34.793277,137.7391202 34.7932065,137.7416956 34.7931083,137.7419522 34.7930692,137.7420898 34.7930274,137.7423772 34.7928553,137.7427256 34.7931964,137.742908 34.7934167,137.7431226 34.7939805,137.743004 34.7940083,137.7430367 34.7949849,137.7410514 34.7950127,137.7376932 34.795136)^^<http://www.openlinksw.com/schemas/virtrdf#Geometry>
http://linkedgeoata.org/triiplyf/way194580587	"三方原学園"	"LINESTRING(137.7329831 34.7563187,137.7330797 34.7559794,137.7331921 34.75568,137.7333101 34.7553098,137.7333852 34.7550586,137.7334123 34.7548334,137.7334228 34.7547016,137.7336108 34.7549789,137.7337395 34.755023,137.7339056 34.7551423,137.7340185 34.7553535,137.7341684 34.755583,137.7343618 34.7557193,137.7343672 34.7557943,137.7342599 34.7558868,137.7341094 34.7559797,137.7340021 34.7561163,137.7339163 34.7562618,137.73382 34.7564069,137.7337929 34.7564777,137.7329831 34.7563187)^^<http://www.openlinksw.com/schemas/virtrdf#Geometry>
http://linkedgeoata.org/triiplyf/way28068053	"市営新川南駐車場"	"LINESTRING(137.7327571 34.701808,137.732763 34.7017971,137.7328118 34.7015488,137.732988 34.7007001,137.7327961 34.7006792,137.7327911 34.7007002,137.7327775 34.7007143,137.7326045 34.7016555,137.732583 34.701769,137.7327571 34.701808)^^<http://www.openlinksw.com/schemas/virtrdf#Geometry>
http://linkedgeoata.org/triiplyf/way176281436	"浜松市立中郡中学校"	"LINESTRING(137.7736026 34.7685549,137.7737635 34.7695155,137.775625 34.7692688,137.7753246 34.7677881,137.7741176 34.7680701,137.7742034 34.7685064,137.7736026 34.7685549)^^<http://www.openlinksw.com/schemas/virtrdf#Geometry>

Figure 4.11. The Output of GeoSPARQL query in HTML format

4.2.4. Modeling Rules

In this stage, we present an example of a mechanism for predicting instance-value pairs to produce assessment status. We have three instances of the status of location and three instances of the status of POIs. These instances would be an assessment status as a result of an assessment event that will occur. We expect to produce an assessment status based on the building use control in Figure 4.14.

The following rules are the basic rules to construct an example of the use of rules with an algorithm (see Algorithm 1) to predict instance-value pairs of the object for producing assessment status.

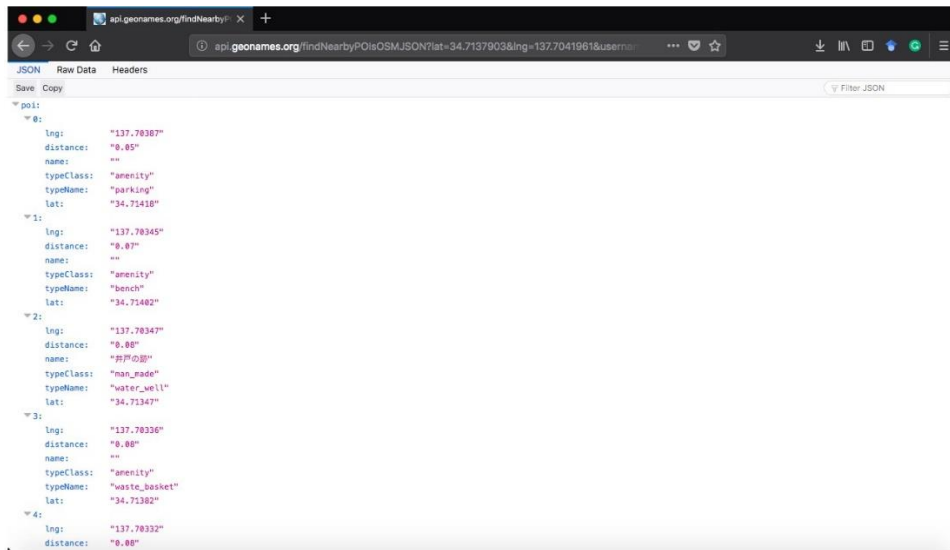


Figure 4.12: An Example GeoNames Web Services to Retrieve Nearby POIs' data

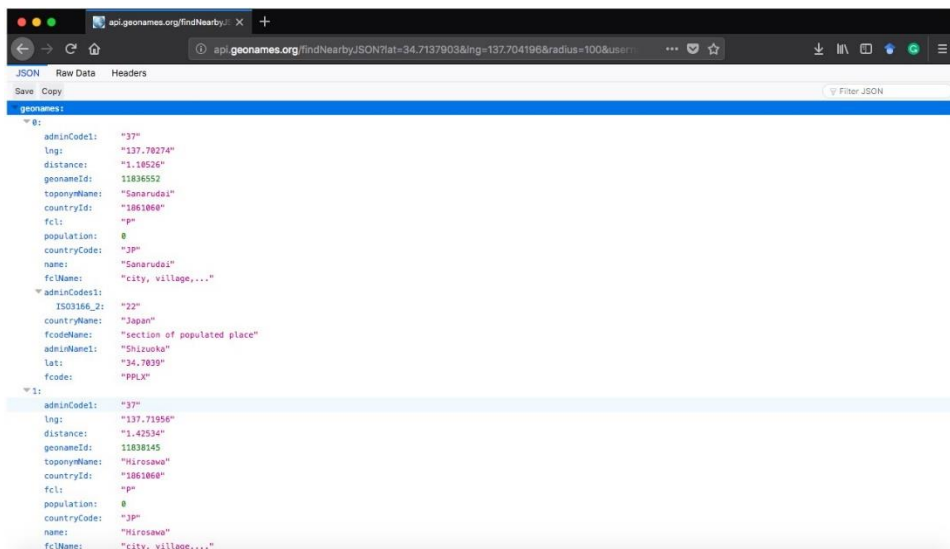


Figure 4.13: An Example GeoNames Web Services to Retrieve Nearby Place

$\forall x x \in \text{RealEstate} \Rightarrow [\forall y \text{ hasCityPlanningAct}(x, y) \Rightarrow y \in \text{LandUseZones}]$

(For all x , x is an instance of real estate implies for all y , x has city planning act y while y is an instance of land use zones)

$\forall x x \in \text{RealEstate} \Rightarrow [\forall s \text{ StatusOfLocation}(x, s) \Rightarrow s \in \text{Assessment_Status}]$

(For all x , x is an instance of real estate implies for all s , x status of locations while s is an instance of assessment status)

$\forall b b \in \text{POIs} \Rightarrow [\forall s \text{ StatusOfPOIs}(b, s) \Rightarrow s \in \text{Assessment_Status}]$

(For all b , b is an instance of POIs implies for all s , b status of POIs s while s is an instance of assessment status)

Examples of buildings	can be built												usually cannot be built	
	Category I exclusively low-rise resi- dential zone	Category II exclusively low-rise resi- dential zone	Category I mid-high-rise oriented resi- dential zone	Category II mid-high-rise oriented resi- dential zone	Category I residential zone	Category II residential zone	Quasi- residential zone	Neigh- bor- hood com- mercial zone	Commer- cial zone	Quasi- industrial zone	Industrial zone	Exclu- sively industrial zone	Areas with no land use zone desig- nation (Utilization Control Areas are excluded)	
Houses, Houses with other small scale function (store, office, etc.)														
Kindergartens, Schools (Elementary, Junior High, Senior High)														
Shrines, Temples, Churches, Clinics														
Hospitals, Universities														
Stores (mainly selling dairy commodities)/Restaurants with floor space of 150m ² max. on the first or second floor (excluding※)													D	
Stores/Restaurants with floor space of 500m ² max. on the first or second floor (excluding※)													D	
Stores/Restaurants not specified above (excluding※)				A	B									
Offices, etc. not specified above				A	B									
Hotels, Inns					B									
Karaoke boxes (excluding※)														
Theaters, Movie theaters (excluding※)							C							
※Theaters, Movie theaters, Stores, Restaurants, Amusement facilities and so on, with more than 10,000m ² of floor area														
Bathhouses with private rooms														
Independent garage with floor space of 300m ² max. on the first or second floor														
Warehouse of warehousing company, independent garage of other types than specified above														
Auto repair shop					E	E	F	G	G					
Factory with some possibility of danger or environmental degradation														
Factory with strong possibility of danger or environmental degradation														

Note A : Must not be built on the third floor or higher. Must not exceed a floor area of 1,500m².
B : Must not exceed a floor area of 3,000m².
C : Audience seating floor area must not exceed 200m².
D : Stores and restaurants must not be built
E : Floor area must not exceed 50m².
F : Floor area must not exceed 150m².
G : Floor area must not exceed 300m².

Figure 4.14: The Building Use Control [18]

$\forall x \in \text{RealEstate} \Rightarrow [\forall y \text{ participateIn } (x, (\text{during } (y, (t_1, t_2))) \Rightarrow y \in \text{POIs' Assessment} \wedge (t_1, t_2) \in \text{Time Interval}]$
(For all x , x is an instance of real estate implies for all y , x participate in y during t_1 until t_2 while y is a POIs' assessment event and t_1 and t_2 are the instances of time interval)

Here we define the variables as shown in Table 4.3:

TABLE 4.3: Explanation of variables

Variables	Explanation
$y = (y_1, y_2, \dots, y_{13})$	Instances of LandUseZones/City-PlanningAct
$b = (b_1, b_2, \dots, b_9)$	Type of POIs
x	Instances of the Target Real Estate Location
$s = (s_1, s_2, \dots, s_p, s_l)$	Instances of Assessment_Status

Figure 4.15 shows the output of SPARQL Query SELECT in Listings 6, which contains the value of real estate area and city planning. Thus, we expect to use this query to perform the scenarios and produce assessment status. Therefore, we apply Algorithm 1, which describes the event when the user types the target real estate location. For example, Aoinishi and the user doing 'click' as an action to the marker on the map, the system would search and sort the data of real estate area that match the user's ask to the system and analyze it. For instance, Aoinishi has city planning act Category_1_Residential_District which means some POIs, such as hospitals and universities, can be built in that area, then the system will produce assessment status Good Location and Hospital/University can be built-in info window's content.

Algorithm 1: Predicting Instance-Value Pairs of An Object for Assessment_Status

Data: Ask the user for the target real estate location (x)

Result: Instance-value pairs(s_p, l): (Can Be Built and GoodLocation) (Usually Cannot Be Built and Bad Location) (No Status of POIs and No Status of Location)

```

1 initialization;
2 foreach  $x \in \text{RealEstate}$  and  $b \in \text{POIs}$  do
3   if  $b$  equals  $y$  and  $b$  is a member of can be built then
4     return  $s$  = Can Be Built and Good Location;
5   else if  $b$  equals  $y$  and  $b$  is a member of usually cannot be built then
6     return  $s$  = Usually Cannot Be Built and Bad Location;
7   else
8     return  $s$  = No Status of POIs and No Status of Location;
9   end
10 end

```


Listing 4: An Example of SPARQL Query to Select Real Estate Area and City Planning Act from Real Estate Transaction-Price Data

```
SELECT ?RealEstateArea ?CityPlanningAct
WHERE {
  ?s <http://example.org/data/realestate#City_Planning >
  ?CityPlanningAct .
  ?s <http://example.org/data/realestate#Area >
  ?RealEstateArea .
};
```

5. CONCLUSION

In our approach, we have divided three stages in handling data integration together with spatio-temporal contexts as follows:

- 1) *Selected and Converted the Data*: In this stage, we assume that the sources used in building the prototype system have already been converted into a standardized RDF format using RDF Converter (e.g., from the CSV file in each row becoming resources and each column becoming properties in the RDF). We implemented the local SPARQL Endpoints using Apache JENA Fuseki. The Endpoint stores triple and provides query-based access to the SPARQL query engine.
- 2) *Populated and Integrated the Instances into the Ontologies*: This stage is required to accumulate the instances, to access and interlink both the local and the external SPARQL Endpoints. It is enabled by integrating the data, providing high-quality information, and collecting the data, which are our data mixed with third-party information. In this stage, we implemented a program that consists of FederatedSPARQL Query to automatically populate the instances of our real estate ontology from external SPARQL Endpoints.
- 3) *Transformed, Retrieved, and Visualized the Data into Map-friendly Format*: In this stage, we assume that the data transformed from machine-readable (i.e., JSON, RDF/XML) into human-readable meta-data (i.e., XML/HTML/web page) and map-friendly format. We have utilized a method to perform geocoding and reverse geocoding, which are available in Google MAP services.

REFERENCES

- [1] B. Balk, H. J. d, and W. Diewert, "Handbook on residential property price indices (rppis)", World Bank Group, 2013.
- [2] C. Dillon, *Landed: The Guide to Buying Property in Japan*. Dillon Communications Ltd, 2010.
- [3] F. Antunes, M. Freire, and J. P. Costa, "Semantic web tools and decisionmaking", in *Joint International Conference on Group Decision and Negotiation*, 2014, pp. 270–277.
- [4] T. Berners-Lee, J. Hendler, and O. Lassila, "The semantic web", *Scientific american*, vol. 284, no. 5, pp. 34–43, 2001.
- [5] J. Hendler, J. Holm, C. Musialek, and G. Thomas, "Us government linked open data: Semantic. data. gov", *IEEE Intelligent Systems*, vol. 27, no. 3, pp. 25–31, 2012.
- [6] F. Bauer and M. Kaltenböck, "Linked open data: The essentials", *Edition mono/monochrome*, Vienna, 2011.
- [7] N. F. Noy, M. Crubézy, R.W. Fergerson, H. Knublauch, S.W. Tu, J. Vendetti, and M. A. Musen, "Protégé-2000: An open-source ontology-development and knowledge-acquisition environment.", in *AMIA Annual Symposium proceedings. AMIA Symposium*, vol. 2003, 2003, pp. 953–953.
- [8] N. F. Noy, M. Sintek, S. Decker, M. Crubézy, R. W. Fergerson, and M. A. Musen, "Creating semantic web contents with protege-2000", *IEEE intelligent systems*, vol. 16, no. 2, pp. 60–71, 2001.
- [9] M. A. Musen, "The protégé project: A look back and a look forward", *AI Matters*, vol. 1, no. 4, pp. 4–12, 2015, ISSN: 2372-3483. DOI: 10.1145/2757001.2757003. [Online]. Available: <http://doi.acm.org/10.1145/2757001.2757003>.
- [10] B. DuCharme, *Learning SPARQL: querying and updating with SPARQL 1.1*. O'Reilly Media, Inc., 2013.

- [11] O. Curé and G. Blin, *RDF database systems: triples storage and SPARQL query processing*. Morgan Kaufmann, 2014.
- [12] N. Firdaus, T. Adachi, and N. Fukuta, 2018, July. Towards Handling Spatio-Temporal Contexts on Linked Open Data for Points of Interest. In *2018 7th International Congress on Advanced Applied Informatics (IIAI-AAI)* (pp. 570-575). IEEE.
- [13] B. McBride, "Jena: Implementing the rdf model and syntax specification", in *Proceedings of the Second International Conference on Semantic Web*, CEUR-WS.org, vol. 40, 2001, pp. 23–28.
- [14] E. Prud'hommeaux and C. B. Aranda, "SPARQL 1.1 federated query", W3C, W3CRecommendation, Mar. 2013, <http://www.w3.org/TR/2013-/REC-sparql11-federated-query-20130321/>.
- [15] S. Auer, J. Lehmann, and S. Hellmann, "Linkedgeodata: Adding a spatial dimension to the web of data", in *International Semantic Web Conference*, Springer, 2009, pp. 731–746.
- [16] C. Bizer, R. Cyganiak, and T. Gauß, "The rdf book mashup: From web APIs to a web of data", in *Proceedings*, vol. 1, 2007.
- [17] B. Schoenmakers, "Designing linked data applications", Master's thesis, Delft University of Technology, 2010.
- [18] M. of Land, Infrastructure, Transport & Tourism, *Urban land use planning system in japan*. [Online]. Available: <http://www.mlit.go.jp/common/001050453.pdf>