SMART TOURS

Ramesh Gorantla
Software Engineering
Arizona State University
Tempe, AZ, USA
rgorantl@asu.edu

Nagarjuna Kalluri
Software Engineering
Arizona State University
Tempe, AZ, USA
nkalluri@asu.edu

Debarati Bhattacharyya
Software Engineering
Arizona State University
Tempe, AZ, USA
dbhatt14@asu.edu

Soham Dasgupta
Information Technology
Arizona State University
Tempe, AZ, USA
sdasgu11@asu.edu

***Abstract*—In this era of digitization, we prefer to have sufficient and relevant information at our fingertips. In this project we are building a Smart Tours application which will guide people visiting museums and other places of interest. A typical problem that people across the globe encounter while visiting museums is the lack of information about the artifacts displayed. This leads to difficulty in understanding the significance of the artifacts they are looking at. Although, some information is present alongside the item on display, most of the times its either incomplete or insufficient for the viewer to realize the artifacts historical importance. In addition to this, a person may wish to look at or know more about artifacts belonging to some specific genre like art, sculpture etc. instead of viewing a plethora of items. This project aims at solving these issues by creating an application which uses Linked data and Ontology techniques to provide viewers an improved and interesting tour of the museums worldwide that hold a glimpse of the heritage of mankind.**

***Index Terms*—Ontology techniques, Semantic, Museum, SPARQL, Resource Description Format**

I. Introduction

Museums and similar places of art usually have limited information about different artifacts displayed there. Also, providing more information and details, physically, often require a lot of investment in time and money. To overcome this, we can have the information conveyed to the viewer directly through mobile or other electronic gadgets. This will result in better experience for the viewers and thus will save space and money for the museum. Recent advancements in Semantic web technologies have made it easy to achieve this goal. One such step towards this goal is to enable museum goers to get the relevant information about the artifact they are currently watching. This is made possible by taking in the current coordinates of his/her location and providing specific details pertaining to that artifacts location. We plan to develop an application which takes in the current coordinates, maps the coordinates to the linked data and subsequently retrieve the relevant information about the artifacts such as history, artist information etc. and then display them back to the user. The main aspects of this project are,

- Data Collection
- Mapping data to RDF
- Linking data to coordinates
- Application to enable users to view the data

We will be using SPARQL as the RDF query language to manipulate and retrieve the data stored in Resource Description Framework format. SPARQL provides us with analytic queries like SORT, JOIN to execute against RDF data sets, consisting of RDF graphs. The outputs of the SPARQL queries can be rendered in different formats like RDF, HTML, JSON, or XML. We will also be using Friend of a Friend(FOAF), which is the standard vocabulary for describing relationships among the different subject types.

We will incorporate multiple triple definitions which will permit joins to different subject types and thus provide us with valuable information efficiently. For example, we can use this technique to join the two triples containing the sculpture description data and sculpture position to return the list of sculptures near a particular person's location by a particular artist.

II. A Glimpse into Similar Works

There has been many works of publishing linked data to the internet. One of such works was connecting the Smithsonian Art Museum data to Linked data cloud. The SAAM ontology is extended from the EDM (European Data Model) which is a large data set of artworks all over Europe and their work showed how to reuse existing ontology terminologies. Another famous work was the American Art Collaborative data set where also Karma data integration tool was been used for cleaning, mapping and publishing data [6].

But most of these works did not have location data mapped with a particular artwork and that was a deficit. This means the user has to search for an artifact through some catalog to get more information regarding the same. Our solution involves building a web app which gets the user location precise to 5 meters and displays the artifact details near the user.

III. Project Flow

The work flow of this project has been designed in a methodical manner. The first iteration comprised of the data set collection and data preparation phase. The artists and artworks data set was obtained from the Museum of Modern Art data set which started data collection approximately around 1929. The data set is an excellent collection of diverse artworks of around 26000 artists all around the world. But location data was randomly generated using python scripts. Hence this phase incorporated both obtaining data and preparing data for further processing.

The next iteration in the project work flow consists of Owl File Creation in which the data sets were converted into .owl format using tools like Protege. After that, the owl files were uploaded into Cellfie tool and were mapped to RDF by connecting the columns to classes. After converting the data to RDF, further validation of the mapping was performed. Karma and OpenRefine tools are also used for the triples generation.

After the step of RDF generation, the files were uploaded into local Fuseki server for Sparql querying. Having done that, the linked owl files were distributed over physical spaces and correspondingly queried using Apache Jena from the local Fuseki instance to return the results pertaining to specific pairs of latitude and longitude values. Finally, the results are displayed in a clean, simple and highly-aesthetic UI on the screen.

IV. Data Collection

A. Data sources

The three data sets we are using in this project are Artists, Artworks and Artworks location.

The Artworks and Artists data has been collected from the Museum of Modern Art (MoMA) collection [1]. This collection contains artifacts from all over the globe for the last 150 years. The data contains a hugely expanding collection of visual items, including sculpture, painting, drawing, architecture, film, photography etc. Artworks location data is generated using a python script.

1) Artworks data set: The Artworks dataset has details about the artworks like title, artist and other info. The data set has around 20174 records. This dataset has only one entity named Artworks. The attributes of the dataset include Title, Artist, ConstituentID, Date, Medium, Dimensions, CreditLine, Classification, Department, ObjectID, URL, ThumbnailURL. ObjectID identifies the artifact uniquely and is unique in the entire Artworks dataset. ConstituentID is the unique id of the artist who created the artifact. ThumbnailURL is the URL of the picture of the artifact. The cardinality relation between Artists and the Artworks is one-to-many as one artist can create multiple artifacts. The cardinality relation between Artworks and Artworks location is one-to-one as one artifact can only be present in one location.

2) Artists data set: The Artists dataset has the information about artists name, gender, nationality, birth year and death year. This dataset has around 15652 records. The attributes of the Artists dataset include ConstituentID, DisplayName, Nationality, Gender, BirthYear and DeathYear. The ConstituentID attribute is the primary key of the Artists entity which uniquely identifies the artist. The cardinality relation between Artists and the Artworks is one-to-many as one artist can create multiple artifacts.

3) Artworks location data set: As location data is not available for artworks, we wrote the below python script to generate the data set. For the 20174 thousand records present in the Artworks data set, we generated unique latitude and longitude data combinations. This dataset has three attributes, ObjectID, Latitude and Longitude. The ObjectID here is the uniqueId of the artifact, Latitude, and Longitude gives us the corresponding location information.

Listing 1. Location data generator

```
f = open("latlon.csv", "a")
for i in range(33001, 33360, 3):
    for j in range(94001, 94991, 3):
            f.write(str(i/1000.0) +
                    "," + str(j/1000.0) +
                            "\n")
```

We considered 33.001 as the starting latitude and 94.001 as the starting longitude. The Lat Long data has a 3 decimal precision, and each latitude or longitude value is 3 degrees apart. For example if one latitude is 33.001 the other one will be at 33.004, same goes for longitude.

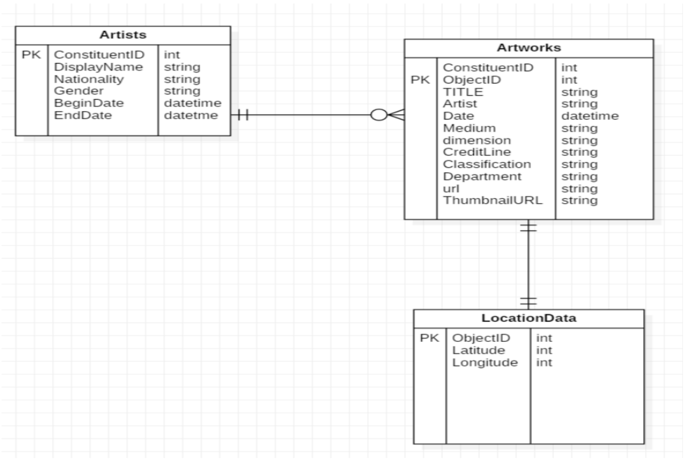


Figure 1. ER Diagram representing database modelling

Artists and Artworks data has been preprocessed to meet our requirements.

B. Data pre-processing steps

- Artists data set
  - Removed 3 columns (ULAN, Wiki QID, ArtistBio) which are not used in the project
  - Treat Empty cells of Nationality, Gender with "N/A"
- Artworks data set
  - Columns ArtistBio, Nationality, BeginDate, EndDate, Gender, AccessionNumber, DateAcquired, Cataloged, Circumference (cm), Depth (cm), Diameter (cm), Height (cm), Length (cm), Weight (kg), Width (cm), Seat Height (cm), Duration (sec.) etc. have been removed.
  - Removed duplicate records and redundant data.

We used Microsoft Excel for the above mentioned preprocessing steps.

V. Ontology Creation

In this phase, three different ontology files are created to represent the three datasources. Protege is used as the tool to create the ontologies. Separate OWL files specifying all the classes, data properties and restrictions are created for each dataset. Once all the details are included in the ontology, it is saved in RDF/XML file format. These are the three files created:

- artists.owl for Artists datasource
- artworks.owl for Artworks datasource
- latlon.owl for Location datasource

VI. Data Conversion

In this phase, data that is collected in earlier phases needs to be added to the ontologies as individuals. Three different tools are used to do the data conversion, Protege cellfie, Google refine, and Karma. Detailed steps for each tool are specified below.

1) Protege Cellfie Plugin [2]: The Cellfie plugin in Protege is used to convert the data sources into triples (subject - object - predicate). User needs to write a query and cellfie then processes the files based on the query and generated triples. Cellfie accepts excel files as the input for processing, but as in this case they are CSV files, the files are converted to excel first and then used for processing. To load the data into the tool, Tools - Create is used and the axioms from excel workbook are loaded. Now, the excel file that needs to be uploaded is selected. This opens a new screen with the data in the excel on the top part, on the bottom part the transformation rules can be written. The Add button is used to write a new rule. This opens up a popup, asking for the row or column to start the processing and at which row or column to stop the processing. After specifying these details, a rule needs to be written in the textbox below. This rule maps the input file to the classification defined. Once the rule is written, Generate Axioms is used to generate the individuals from the file based on the query. Next, there will be an option to add these to the current ontology. Using that all the individuals will be added to the current owl file.

A couple of issues were encountered while doing the processing. Firstly, as the excel files were converted to CSV files, there was some character encoding issue and all those records had to be fixed manually. The other issue was with the size of files. Due to the large size of the data, Protege threw heap memory not sufficient error. To fix this, all the files were splitted into small chunks and each chunk was uploaded at a time.

2) Open Refine [3]: Another tool that has been used to convert the data sources into triples (subject - object - predicate) format is the Google OpenRefine tool. Through usage of this particular tool, it has been found that this tool is very useful in converting large amount of data at one go. Data to the tune of around 21 thousand rows of data has been loaded to the tool in one attempt unlike the other tools where the input data had to be split to be able to load and convert the data. Also, the usage of this particular tool is very simple when compared to other tools owing to its user friendly UI.

Firstly, data present in the csv format for each of the datasources has to be loaded into the tool. The tool also supports other file formats. Once the data is properly loaded, the data is displayed on the UI. If the uploaded data is satisfactory, a separate project needs to be created. The different file formats that tool can change the current data to is displayed on the right upper corner under Extensions tag. Now, Edit RDF Skeleton under the RDF extension tag is selected. Then, the column data needs to be mapped to the RDF property tags present in the ontology. Since the ontology is not explicitly present on the Semantic web, it needs to be added as a prefix here. Since, adding prefix through an URI is throwing error, the local version of the ontology file has been added to generate the corresponding prefix. Once the prefix is added, the mapping from the csv column data to that of RDF/XML property tags is done and the RDF schema is saved. Now, the data is exported to an output file using the Export button on the top right corner of the tool.

The generated output files pose some errors. The namespace provided in the ontology is not reflected once the instances are added by the OpenRefine tool. Also, some discrepancies in the converted data have been found.

3) KARMA [4]: Karma is a very efficient tool that has been used to convert data sources into triples. The Karma source file was downloaded. After that the zip file was extracted in a folder. Next, the directory where Karma was placed was accessed from command prompt. The command mvn clean install was run in the shell which compiled and installed Karma. After installing Karma, the servlet was uncommented inside the web.xml file for Windows machine. Once Karma was installed, the following command was executed: mvn jetty:run was executed from the karma-web folder. After that the following URL was opened in the browser: http://localhost:8080. Once the Jetty server is running and the Karma is opened in the browser, the ontologies like artists.owl were imported. Then the .owl file was imported into the Karma as an ontology. After the ontology was successfully imported, the corresponding csv file was added into the Karma. Once all the rows of the CSV are loaded in Karma, the semantic types were set for each column header. First, the uri was set and it was directly detected by the owl ontology file. That particular uri was selected, and then the literal type, language were selected and finally the semantic type was saved. Correspondingly, the rest of the semantic type properties were selected and the literal type and language were set for them. Once it is done a tree model of the attributes was displayed on the top of properties. Finally, the csv is published into a rdf file. The saved RDF file is downloaded as .ttl format and then it is converted to .owl format.

VII. Data Linking

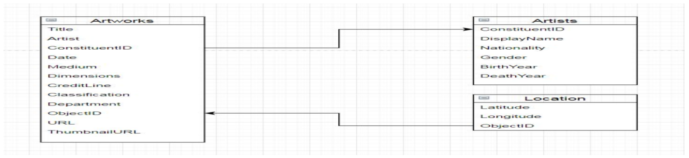
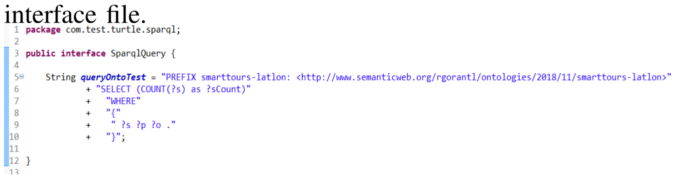


Figure 2. Data Linking

In this phase, all the OWL files that are generated in the above phase need to be linked. To link them Protege is used. Protege has the feature to directly import existing OWL files. After opening Protege, the direct import option in the bottom part of the screen is used to do this. This opens a popup, giving options to select an existing file from local file system. Once selected, Protege links the contents of the uploaded file into current file, it shows the count of individuals imported from the OWL file. The above steps are repeated for all the OWL files. Once all the OWL files are imported into the new file, it is saved. The new file shows the count of individuals and the classes present. Also the data properties can be linked by saying that they are equivalent, by using the equivalence property in the Protege tool.

VIII. Ontology Validation

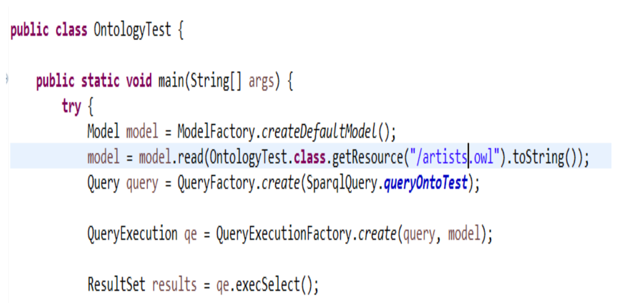
The procedure mentioned below was followed to validate the final ontologies. A Java Maven project was created and two files OntologyTest.java and SparqlQuery.java(as interface) were added to it. Next, all the required dependencies such as apache.jena etc. were included as part of the Maven pom.xml file. Thereafter, the below SPARQL query was put in the interface file.



```
package com.test.turtle.sparql;

public interface SparqlQuery {

    String queryOntoTest = "PREFIX smarttours-latlon: <http://www.semanticweb.org/rgorantl/ontologies/2018/11/smarttours-latlon>"
            + "SELECT (COUNT(?s) as ?sCount)"
            +   "WHERE"
            +   "{"
            +    " ?s ?p ?o ."
            +   "}";

}
```

Figure 3. SPARQL query for getting the axioms count

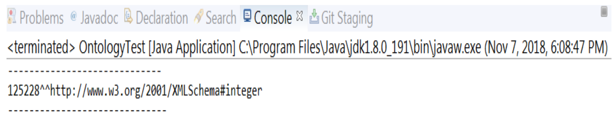
Lastly, the jena QueryFactory was used to run this Sparql query and validate the ontology.



```
public class OntologyTest {

    public static void main(String[] args) {
        try {
            Model model = ModelFactory.createDefaultModel();
            model = model.read(OntologyTest.class.getResource("/artists.owl").toString());
            Query query = QueryFactory.create(SparqlQuery.queryOntoTest);

            QueryExecution qe = QueryExecutionFactory.create(query, model);

            ResultSet results = qe.execSelect();
```

Figure 4. Code snippet showing the query execution

The output for the validation shows the total number of axioms along with the datatype of the count.



```
<terminated> OntologyTest [Java Application] C:\Program Files\Java\jdk1.8.0_191\bin\javaw.exe (Nov 7, 2018, 6:08:47 PM)
-----------------------------
125228^^http://www.w3.org/2001/XMLSchema#integer
-----------------------------
```

Figure 5. Results of query execution

IX. User Interface

User interface has one screen with 2 sections namely -

- Location details input
- Display results

Location details input section is where input location (latitude and longitude) details are given and output results are displayed in a separate section called Display results section in the home screen itself. The Location details input section in the home screen has a text bar to input the location data. Comma separated latitude and longitude values are taken as input for the location data. Validations for input data have also been added to make sure only valid data is queried. On clicking the Find button below the text bar, SPARQL query is run on the data sets using Apache Jena and the results are fetched and displayed as shown in the second UI screen.

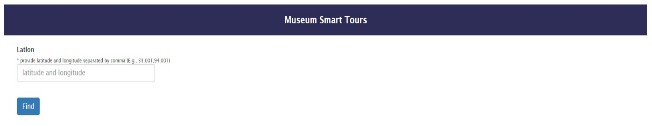
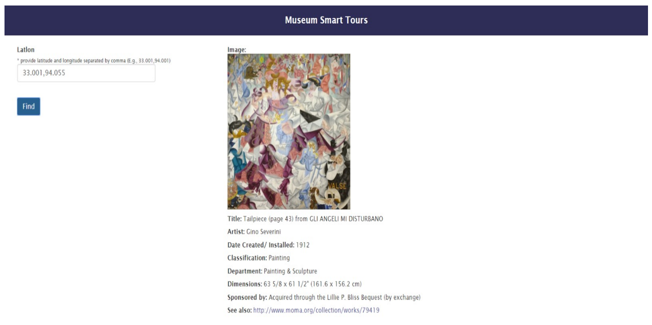

Museum Smart Tours
Latlon
* provide latitude and longitude separated by comma (E.g., 33.001,94.001)
latitude and longitude
Find

Figure 6. User Interface before Query


Museum Smart Tours
Latlon
* provide latitude and longitude separated by comma (E.g., 33.001,94.001)
Find
Image:

33.001,94.055

Title: Tailpiece (page 43) from GLI ANGELI MI DISTURBANO
Artist: Gino Severini
Date Created/ Installed: 1912
Classification: Painting
Department: Painting & Sculpture
Dimensions: 63 5/8 x 61 1/2" (161.6 x 156.2 cm)
Sponsored by: Acquired through the Lillie P. Bliss Bequest (by exchange)
See also: http://www.moma.org/collection/works/79419

Figure 7. User Interface after Query

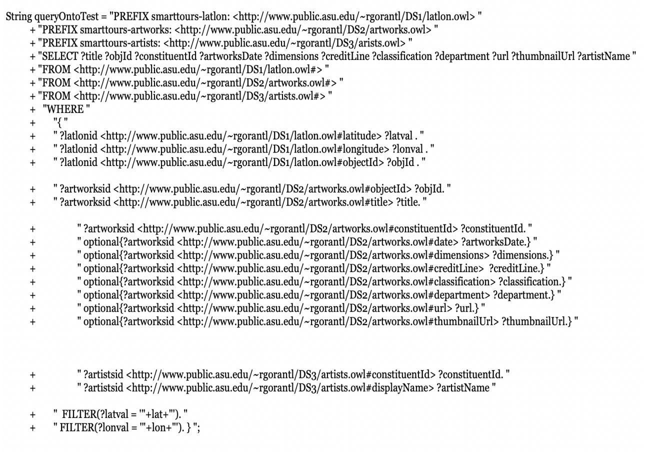
X. Sparql Query Server

One of the most crucial part of any semantic web application is the querying of Linked Data. In this case, it has been achieved using the popular and reliable Apache Jena Fuseki. Apache Jena Fuseki is a SPARQL server which has the capability to run as an operating system service, as a Java web application (WAR file), or as a standalone server. It implements a secure communication using Apache Shiro and also provides an user interface for server monitoring and administration.[5]

It uses SPARQL 1.1 protocols for query and update alongside the SPARQL Graph Store Protocol. Fuseki is tightly integrated with TDB. Thus it stands as a robust, transactional persistent storage layer, and incorporates Jena text query and Jena spatial query. It serves as an excellent protocol engine for other RDF query and storage systems.[5]

For this application, Fuseki was installed and configured on localhost as well as remote server. An ontology was created which directly imported all the main ontologies ie. artists, artworks and latlon. The three main ontologies were hosted on three different physical spaces on the cloud. The import ontology was uploaded to the local Fuseki instance and the SPARQL query was executed on this. Thus, with these imports it could connect to the linked ontologies hosted on the cloud and fetch the results.

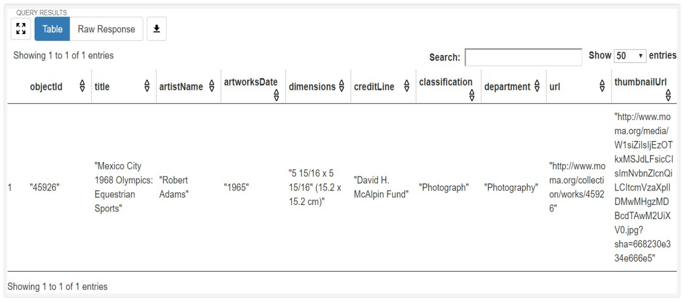
XI. Sparql Query on Linked Data



```
String queryOntoTest = "PREFIX smarttours-latlon: <http://www.public.asu.edu/~rgorantl/DS1/latlon.owl> "
        + "PREFIX smarttours-artworks: <http://www.public.asu.edu/~rgorantl/DS2/artworks.owl> "
        + "PREFIX smarttours-artists: <http://www.public.asu.edu/~rgorantl/DS3/arists.owl> "
        + "SELECT ?title ?objId ?constituentId ?artworksDate ?dimensions ?creditLine ?classification ?department ?url ?thumbnailUrl ?artistName "
        + "FROM <http://www.public.asu.edu/~rgorantl/DS1/latlon.owl#> "
        + "FROM <http://www.public.asu.edu/~rgorantl/DS2/artworks.owl#> "
        + "FROM <http://www.public.asu.edu/~rgorantl/DS3/artists.owl#> "
        +  "WHERE "
        +     "{ "
        +     " ?latlonid <http://www.public.asu.edu/~rgorantl/DS1/latlon.owl#latitude> ?latval . "
        +     " ?latlonid <http://www.public.asu.edu/~rgorantl/DS1/latlon.owl#longitude> ?lonval . "
        +     " ?latlonid <http://www.public.asu.edu/~rgorantl/DS1/latlon.owl#objectId> ?objId . "

        +     " ?artworksid <http://www.public.asu.edu/~rgorantl/DS2/artworks.owl#objectId> ?objId. "
        +     " ?artworksid <http://www.public.asu.edu/~rgorantl/DS2/artworks.owl#title> ?title. "

        +           " ?artworksid <http://www.public.asu.edu/~rgorantl/DS2/artworks.owl#constituentId> ?constituentId. "
        +           " optional{?artworksid <http://www.public.asu.edu/~rgorantl/DS2/artworks.owl#date> ?artworksDate.} "
        +           " optional{?artworksid <http://www.public.asu.edu/~rgorantl/DS2/artworks.owl#dimensions> ?dimensions.} "
        +           " optional{?artworksid <http://www.public.asu.edu/~rgorantl/DS2/artworks.owl#creditLine>  ?creditLine.} "
        +           " optional{?artworksid <http://www.public.asu.edu/~rgorantl/DS2/artworks.owl#classification> ?classification.} "
        +           " optional{?artworksid <http://www.public.asu.edu/~rgorantl/DS2/artworks.owl#department> ?department.} "
        +           " optional{?artworksid <http://www.public.asu.edu/~rgorantl/DS2/artworks.owl#url> ?url.} "
        +           " optional{?artworksid <http://www.public.asu.edu/~rgorantl/DS2/artworks.owl#thumbnailUrl> ?thumbnailUrl.} "


        +           " ?artistsid <http://www.public.asu.edu/~rgorantl/DS3/artists.owl#constituentId> ?constituentId. "
        +           " ?artistsid <http://www.public.asu.edu/~rgorantl/DS3/artists.owl#displayName> ?artistName "

        +     "  FILTER(?latval = '"+lat+"'). "
        +     " FILTER(?lonval = '"+lon+"'). } ";
```

Figure 8. SPARQL query

In this SPARQL query, first the latlon data set is queried with the provided latitude and longitude and then the objectId associated with the input latlon is fetched. Next, the artworks data set is queried and the artwork details for that objectId is obtained. Finally, the constituentId in the artworks data set is taken and the artists data set is queried to extract the artist details. In this application, the same rdf:id is used across the rdf tuples, so querying the data sets has become easy.



Showing 1 to 1 of 1 entries

	objectId	title	artistName	artworksDate	dimensions	creditLine	classification	department	url	thumbnailUrl
1	"45926"	"Mexico City 1968 Olympics: Equestrian Sports"	"Robert Adams"	"1965"	"5 15/16 x 5 15/16" (15.2 x 15.2 cm)"	"David H. McAlpin Fund"	"Photograph"	"Photography"	"http://www.moma.org/collection/works/45926"	"http://www.moma.org/media/W1siZiIsIjEzOTkxMSJdLFsicClslmNvbnZlcnQiLCItcmVzaXplIDMwMHgzMDBcdTAwM2UiXV0.jpg?sha=668230e334e666e5"

Showing 1 to 1 of 1 entries

Figure 9. SPARQL query result

XII. Project Review : Retrospective

Throughout the course of this project quite a few issues were faced but, there were two which were notably the most challenging ones. Firstly, the data collection and data scraping. Though, good amount of data could be gathered for the artists and artworks but hardly any data could be found for the locations. Hence, the latitude and longitude data were generated randomly using a Python script. This data was then mapped to the artists and artifacts manually. As of now, this location data acts good as a prototype but it would be accurate if this data can be fetched real-time using GPS. Nonetheless, this location data can serve as a great groundwork for such implementation.

Another hindrance faced was the federated SPARQL query. It was needed to have the ontologies hosted on three different physical spaces. On one hand, where the local Fuseki on a single instance seemed to work fine, it became somewhat difficult to make the query work through the imports. Lack of documentation available over the internet made this even more challenging.

XIII. Future Scope and Evaluation

As mentioned earlier, with the advancement of Semantic Web Technologies, museums and several similar organizations worldwide have shared their collections meta data as machine-readable linked data over the internet. Coupled with the use of W3C open data standard, RDF and SPARQL, linked open data provides developers and researchers the ability to access and re-use such data for creating alternative applications, devices, interfaces etc. The applications of Linked Data is an emerging trend and experts are constantly putting in a lot of efforts into investigating various ways of leveraging these concepts of Semantic Web so that different organizations can publish their databases on the cloud to make it readily available to the world. But the experience so far demonstrates that publishing such data to the linked data cloud is difficult : the databases are large and complex, the information is richly structured and varies from museum to museum, and it is difficult to link the data to other data sets [7].

This project is one such example of the usage of Linked Open Data that enables the artworks to be found and accessed by population across the globe. Whereas most of the museums till date provide the viewers with static content like booklets, some others have adopted the technological advancements. They often hand out smart devices like tablets etc. to make the experience for the viewers informative yet enjoyable.

The research put forth while developing this application lead to two derivations. Firstly, in both the above approaches, the viewer himself will have to search for artifacts in order to explore them. This can prove quite cumbersome especially if a person is new to a particular place. Secondly, even though a lot of art institutions and museums have successfully uploaded their data on cloud, they still could not map the artworks on display at their respective physical locations to the Global Positioning System (GPS) or latitude-longitude locations. Thus, finding a piece of art real-time can still pose an impediment.

There are various applications over the web similar to the Smart Tours but very few of them provide this facility for museums and art galleries. The ones that do, still do not have the locations mapped to their applications. The application that has been created as part of this project is an attempt to resolve this and provide the viewers an unique and improved tour of a place that holds our heritage.

People tend to compare any new application with the standards set by those of the existing ones. This application is expected to be no different. To be accepted into this dynamic digital market, it must prove itself better or at par with the standards of similar applications out there.

Currently, this application works best on a desktop but in order to make this robust, lightweight and easy to use, going forward this will be implemented as a mobile application. Also, the planned mobile application will be compatible with a wide range of operating system platforms such as Android, iOS etc.

Presently, this Smart Tours uses static or randomly generated location data manually mapped to the artifacts at a particular location. While this may be good for a specific institution but, in order make this application useful and cater to the needs of the people it is imperative to use the real time locations obtained from the GPS. As the viewer moves from one physical location to another, the application should be able to fetch the coordinates of the viewer's position and instantaneously update the viewer with the details of that particular artifact he is looking at.

One of the problems commonly encountered by the above approach is the viewer location accuracy. In case, artifacts at a place lie within very close proximity of one another the GPS may not be able to pinpoint the location of a specific one. As an additional feature and in order to provide a workaround for this, the application will show a list of all the artifacts placed within a particular range of that location.

XIV. Exploring new Tools and Techniques

The larger goal of this application is not just to confine it to artifacts of the Museum of Modern Art, the one that is being used currently but, to develop it in a way that will enable any museum or other organization to map their data to linked data themselves. This project has used the Karma integration tool, which greatly simplifies the problem of mapping structured data into RDF and a high-accuracy approach to linking data sets. Beyond these techniques and tools, the ontologies must be continued to be refined and extended to support a wide range of data. New ways to use the linked data must be explored to create compelling applications for museums. New tools such as a relationship finder application that allows a museum to develop curated experiences, linking artworks and other concepts to present a guided story can be brought into the picture. Museums could offer pre-built curated experiences or the application could be used by students, teachers, and others to create their own self-curated experiences [7].

XV. Conclusion

This paper describes our work on mapping the data of the Museum of Modern Art to Linked Open Data. It depicts the end-to-end process of mapping this data, which includes the selection of the domain ontologies, the mapping of the database tables into RDF, the linking of the data to other related sources, the query of Linked Data and displaying the results on the user interface. This project provided us with a much deeper understanding of the real-world challenges in creating high-quality link data.

XVI. Acknowledgement

This project was created and completed under the guidance of professor Mrs. Srividya Bansal and teaching assistant Mr. Jaydeep Chakraborty. We are thankful to them for their continued support and guidance throughout the course of this project.

References

[1] Artworks and Artists data sourced from "Museum of Modern Art (MoMA)" https://doi.org/10.5281/zenodo.1439879

[2] Protege Cellfie Plugin: A Protege Desktop plugin for importing spreadsheet data into OWL ontologies. Available online at: https://github.com/protegeproject/cellfie-plugin

[3] OpenRefine: A free, open source, powerful tool for working with messy data. Available online at: http://openrefine.org/

[4] Karma: An information integration tool that enables users to quickly and easily integrate data from a variety of data sources. Available online at : http://usc-isi-i2.github.io/karma/

[5] Apache Jena Fuseki : A SPARQL server which can run as an operating system service, as a Java web application (WAR file), and as a standalone server. Available online at : http://jena.apache.org/documentation/fuseki2/

[6] Linked Open Data at SAAM : https://americanart.si.edu/about/lod

[7] Connecting the smithsonian american art museum to the linked data cloud by P Szekely, CA Knoblock, et al at the Extended Semantic Web Conference (2013). Available online at: https://www.isi.edu/integration/papers/szekely13-eswc.pdf