**A close up of a sign

Description automatically generated**

**Hypermarket Ontology**

**Semantic Application Report**

**Semantic Data Technologies**

**2020 MOD004979 TRI1 F01CAM**

**Muhammad Umar (mu283)**

**MS.c AI & Big Data**

**2020-2021**

1. **INTRODUCTION**

Semantic web can be considered as an enhanced version of World wide web. The purpose of Semantic web is to create a “readable” version of web for both humans and for machines. This readable format should be in such a way so that software can easily interact and extract information from web data like humans do.

Currently world wide web has become world largest repository of information. There is now a growing demand to process and visualize this information. For this very purpose, in our assessment, we are going to use semantic web modeling to create a unique semantic model for a specific problem and extract some information from it related to our daily life queries.

The entire process will include the construction of an ontology using RDF data model. For creating our ontology, we will be using Protégé, an open source ontology editor, developed by Standford university, USA. Once ontology is created, data will be added, taken from real time environment so that real time queries can be addressed. SPARQL will be used to query and extract information from our ontology. A web interface will be created for users to interact with our ontology and to perform queries on our information. The queries results will be published, and an analysis will be performed on their working. Lastly, future possibilities and limitations will be highlighted.

1. **CONCEPT & AIM**

Hypermarkets are gaining a tremendous popularity worldwide. More and more people are giving preferences to going directly to hypermarkets instead of small-scale stores. The hypermarts provide massive options to buy products from various vendors and their deals provide good opportunities for low income customers to buy goods that would otherwise be very expensive to buy from local market.

**2.1 – Why this ontology?**

While shopping for good, a lot of questions come to usual minds. Which product is the best among all of these? Which product is the cheapest? Or which product is being sold more in our neighborhood, or which brands are being preferred by people of a specific area? Which items I purchased frequently people bought this year? Let us analyze some of these questions.

Consider two simple queries performed on Google search engine.

1. Most purchased items near me this year?

When it should list items purchased near me, it is showing some result of a survey for most popular items in 2019. The question doesn’t want most popular items. It wants to know the most purchased item.

**Graphical user interface, text, application, email

Description automatically generated**

**Graphical user interface, text, application, email

Description automatically generatedGraphical user interface, text, application, email

Description automatically generated**

Fig 1.

1. What boys buy most in Lahore?

The question is self-explanatory. Although it is grammatically wrong. However nowhere it asks for a “dating website” in Pakistan. The listing should include at-least items bought by males in Lahore city. The other results include visiting places in Lahore which is also not related.

The above experiment begs for a semantic web version where small and simple queries like above can be answered with some relevant accuracy at-least.

**2.2 – Previous work**

1. [**http://www.productontology.org**](http://www.productontology.org)

This service provides ca. 300,000 precise definitions for types of product or services that extend the schema.org and GoodRelations standards for e-commerce markup [1].

The above ontology is a better than all other ontologies present so far. I will be using few details from the ontology, however the ontology is missing a lot of necessary details for hypermarkets e.g. events data, organizational data, reporting, statistics, product categories, reviews, ratings etc.

1. **https://www.bbc.co.uk/ontologies/fo**

BBC introduced its own ontology for food related products. However, the ontology is limited to few functionalities including ontology for food, recipes, its reviews etc. The ontology is based on google rich snippets and schema.org.

**3 –** **ONTOLOGY ENGINEERING**

**3.1 – Architecture**

As mentioned earlier Protégé is being used for Ontology development.

Although most of the classes are named in a fashion as to remain self-explanatory. However, few classes need further explanation.

It is always better to reuse components of previously made ontologies hence, I imported *Agent* entity from *foaf*[2] and *warranty* and *delivery method* entities were imported from [3]. *Foaf:Agent* entity contains *Organization* and *Person* sub classes. In the current ontology we are assuming, there exists a base Organization e.g. Tesco, Union Coop, and these organizations can have multiple stores all around the places. The organization however can also be individual entities referred as shops.

*Person* entity can either be a customer or employee of *organization*. That includes managers or delivery guys. Later on, we can also add more personal classes like security, marketer, however currently it is out of our scope.

*Product* entity is responsible for handling all kinds of objects sold by a specific store. Each product can have multiple categories based on its description. That might include a brand, an organization, an order and its customer buying that specific product.

*Order* entity is the main class to handle all kind of sales. Multiple products and their categories can be attached to the entity along with customer, a delivery location, a delivery contact, the shop etc. *Location* class has sub classes which define a location in depth. A country, its city, its geo location as *LatLng*, possible a google/apple identified place and an *address*. As per standards, these addresses can have multiple lines. Each product is also associated with *Review* which should not be mixed with *Feedback. Feedback* itself is a separate entity, hence added as a disjoint with *Review*. *Feedback* will be associated with *Orders*.

Each product is associated with its own *category.* A product can have multiple categories e.g if there exists a product Milk, its category will be within grocery as Dairy product and it can be Organic.

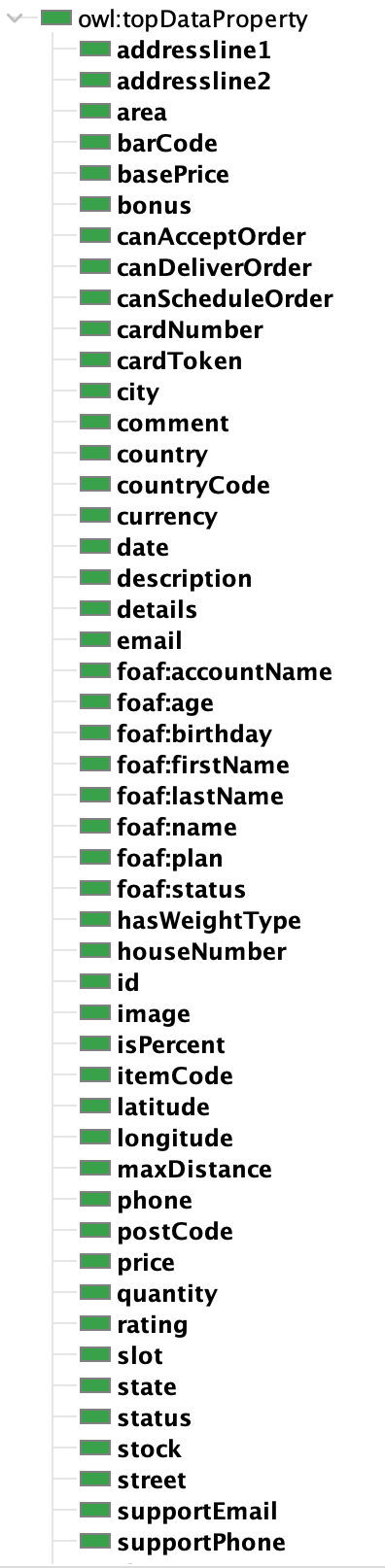
*PaymentMethods* currently include Cash on Delivery, Card on Delivery, Wallet Payment, Credit/Debit Card payment via online portal etc. Each country has its own payment methods and it can be extended to include more methods. *Price* entity deserves special attention. It should be noted that the price can be different based on each store within same organization. Hence price class will be different per product per store per organization.

*Date* seems a simple date and time however a new data property is attached to it namely slots. Consider the fact that each order associated can have a delivery time frame e.g from 8am – 11pm, or each shop can have opening /closing timings.

*Reports & Statistics* is something which will be added in future updates. Each store or product require in-depth report. Suppose we need to include additional reports for each product, or we need to present “smart reports” for customers.

Protégé ontology construction includes 52 classes in total. Axiom count is 8271, Logical axiom count is 6879. Ontology metrics and classes list are shown in Fig 2.

**Diagram

Description automatically generatedTable

Description automatically generatedGraphical user interface, application

Description automatically generated**

Fig 2.

**3.2 – Data Entry**

To create instances of each entity, I used multiple sources. First, all of the data was converted into the form of CSV, modified for conversion and then converted to excel sheet. Cellfie plugin was used to import data into Protégé which is highly dependent on MappingMaster Language. MappingMaster is a domain-specific language (DSL) that defines mappings from spreadsheet content to OWL ontologies. This language is based on the Manchester OWL Syntax, which is itself a DSL for describing OWL ontologies [4].

A sample rule builder used for importing stores data is shown in Fig 3.

For *Brands* an open source brand entry code was used [5]. Around 200 random brands were chosen from the list and was fed to Cellfie. Custom users were added from a well-known supermarket database. For NDA purposes, only names were added as part of data. Three different locations namely, UAE, UK and Pakistan were considered as location for stores and where these customers belonged too. Using excel Random function, all customers were assigned different geo locations as well as corresponding stores.

Same process was repeated for *Products.* Around 200 different products were chosen, each of them was assigned a random store and same base price was as added for each store for testing purposes. Since it was quite difficult to link the categories manually, randomly different categories were assigned.

Rules are added as part of the main source code along with excel sheets with all data [6].

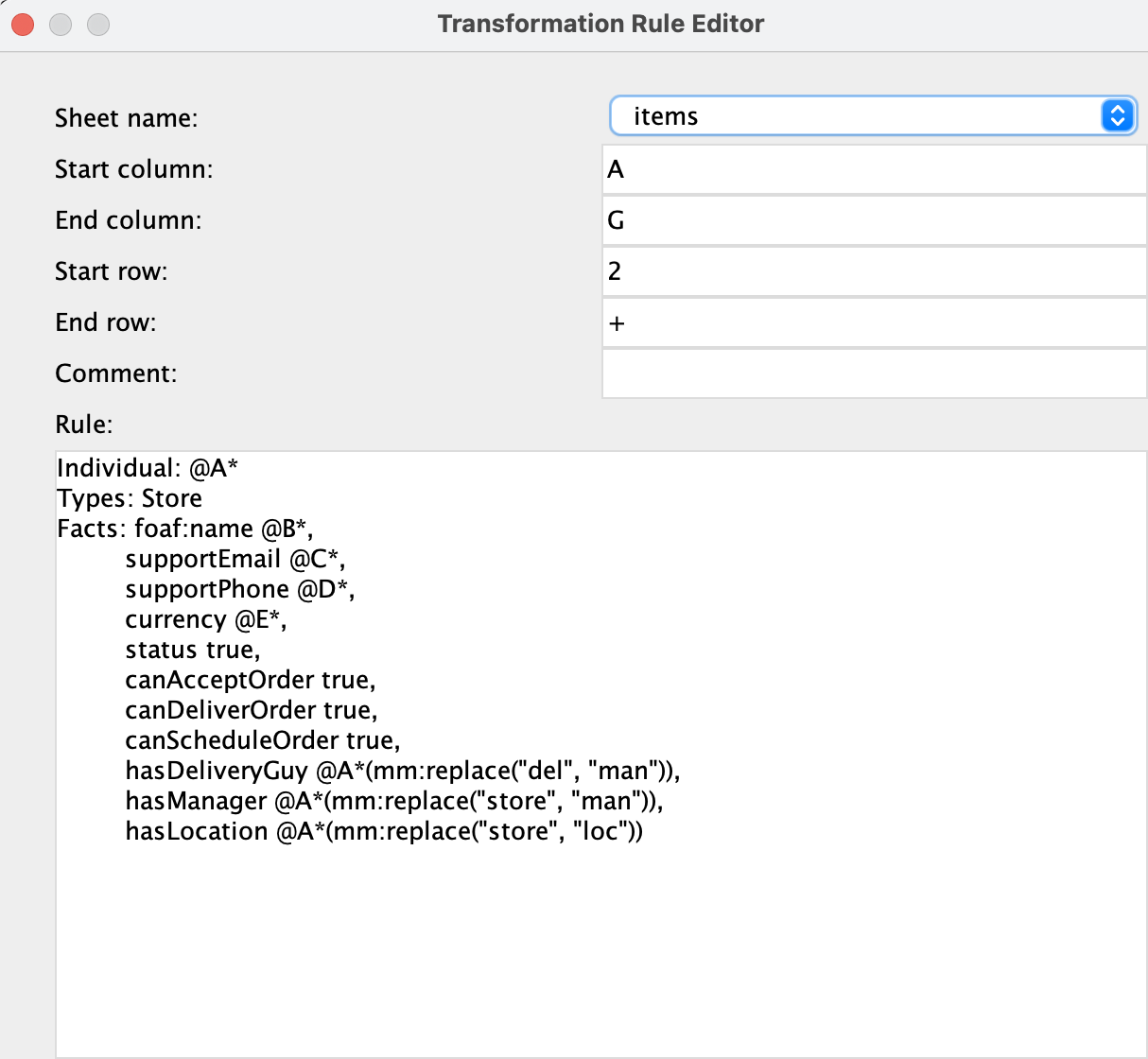


Fig 3.

**3.3 – Analysis**

Let us consider an example an order placed. The order id is defined as order\_9. The OntoGraf representation is shown in Fig 4. As can be seen, order has multiple products attached to it. Order has a loc\_9 as a delivery location whereas the store that is providing these products and delivery is store\_9. For each product there exists a price. For single store buying, It should be noted that prod\_27 has a different store. This is because data entered was based on randomness. Ideally the order products should be consistent with one store only. But for now, we can consider the scenario that customer bought products from different stores and got them delivered via xyz delivery guy. Product 27 is further expanded to list its category and that category products and its brand.

Finally, order is associated with a customer. There are few missing entries like payment mode, charges. These can be added as part of order in further development.

**Diagram

Description automatically generated**

**Diagram

Description automatically generated**

Fig 4

Since our queries mentioned at start, require fetching any items close to a specific location, let us analyze a location named as loc\_5. It can be seen that the location is referred by an order and that order is further attached to customer, products, stores and brands. Hence one can easily make queries based on locality. The OntoGraf representation is shown in Fig 5.

**Diagram

Description automatically generated**

Fig 5.

**4 – IMPLEMENTATION**

The implementation of a small app was done in Kotlin-Android + Jena for Java. The app consists of 4 sections. One for each customers, orders, products and stores. Each section consists of predefined queries. The starting queries are for testing purposes whereas later on the difficulty increases.

The code is divided into multiple files. There is a file containing code for each graphical screen. We start with a splash screen.

* 1. **Splash**

In android a simple splash screen is used to display logo etc however in our case, I am downloading the ontology directly from a backend so it can be directly used anywhere online. The ontology is fetched and updated as an OntModel so Jena can process queries directly. Fig 6 displays the code used to parse the ontology.

* 1. **Home**

The home screen presents list of customer related queries to be performed and display them in an adapter. It should be noted that Android Jena is obsolete and for newest Jena version the libraries need to be re-compiled. However, for demonstration purposes some simple queries will be performed, and rest will be performed directly on SparQL. The first query simple fetches the list of all customers. The Fig 7 display the queries and the design figure.

**Graphical user interface, text, application, chat or text message

Description automatically generated Graphical user interface, text, application

Description automatically generated Graphical user interface, text, application, chat or text message

Description automatically generated**

Text

Description automatically generated

Text

Description automatically generated

Fig 7.

* 1. **Stores**

For stores screens we will start with basic queries and then increase the complexity. The first query displays all stores and shops along with its corresponding city and country name. Fig 8 displays the result and code for selecting all stores. It should be noted that the *BIND* query has been disabled as android Jena currently doesn’t support the command.

**Graphical user interface, text, application

Description automatically generated**

Fig 6.

**Text

Description automatically generatedGraphical user interface, application

Description automatically generated**

Fig 8.

**Graphical user interface, text, application

Description automatically generated Graphical user interface, text, application

Description automatically generated**

Fig 9.

* 1. **Products**

To display products bought in a city, we simply queried for all the orders that were made with a store positioned in a specific city. Fig 9 displays the result. The orders showed 9 items with their prices, store currency and ids of the product.

**5 – ADVANCED QUERIES & EVALUATION**

Once we have setup an interface, we are ready to work on advanced queries to display. The most important questions

To display products bought in a city, we simply queried for all the orders that were made with a store

**6 –** **CRITICAL REFLECTION & FUTURE WORK**

The ontology requires MORE additions some of whom are listed as below.

Suppose someone wants to query, which stores are closer to me, or if specific product is being sold near me based on geo location data. The current system does record the locations however it doesn’t support distance-based queries. For such queries we need to consider two important factors. Firstly, the mathematical calculations and secondly geo spatial based data. Luckily such ontologies do exist which can enhance current ontology by simply adding their headers. For mathematics graph DB provides its own mathematical functions which can be found here [4].

Jena supports spatial based queries however it require the 2 types of RDF representation of geo data, Latitude/Longitude Format (in gonames, DBPedia and Linked Geo Data) (PREFIX geo: <http://www.w3.org/2003/01/geo/wgs84_pos#> and Well Known Text (WKT) Literal (in DBPedia and Linked Geo Data) (PREFIX ogc: <http://www.opengis.net/ont/geosparql#>) [5].

Every holiday in country is associated with events all around the country. Stores announce special events like Black Friday etc. For eastern and middle eastern countries EID festival provides a lot of opportunities. Although the class exists in current ontology however currently it is not so versatile. It requires complete analysis of different possible events and the class properties need to be updated accordingly.

Importantly each product can be associated with different languages. One product can be available in multiple countries and the names, description, details need to be translated into appropriate language of that specific store in which its being sold currently. The Ontology currently doesn’t support this feature.

In the end, for the ontology to be listed in top tier, it must include classes for smart reporting and statistics. E.g what if someone wants to perform a detailed analysis on, all the products sold and estimate the requirements of next month for a store. Surely this is possible, but it requires more data and further distribution of classes in the ontology.

**8 – REFRENCES**

4. https://github.com/MatthiasWinkelmann/english-words-names-brands-places

<https://github.com/protegeproject/mapping-master/wiki/MappingMasterDSL>

https://jena.apache.org/documentation/query/spatial-query.html