Federated Platform for Healthcare

Student Name: Raghav Rathi
Roll Number: 2017083

BTP report submitted in partial fulfillment of the requirements
for the Degree of B.Tech. in Computer Science & Engineering

on 28th May 2021

BTP Track: Engineering Track

BTP Advisors
Dr. Raghava Mutharaju
Dr. Tavpritesh Sethi

Indraprastha Institute of Information Technology
New Delhi

Student's Declaration

I hereby declare that the work presented in the report entitled **" Federated Platform for Healthcare "** submitted by me for the partial fulfillment of the requirements for the degree of *Bachelor of Technology* in *Computer Science & Engineering* at Indraprastha Institute of Information Technology, Delhi, is an authentic record of my work carried out under guidance of **Dr. Raghava Mutharaju** & **Dr. Tavpritesh Sethi**. Due acknowledgements have been given in the report to all material used. This work has not been submitted anywhere else for the reward of any other degree.

Raghav Rathi **IIIT Delhi, 28th May 2021**

Certificate

This is to certify that the above statement made by the candidate is correct to the best of my knowledge.

Dr. Raghava Mutharaju **Place & Date: IIIT Delhi, 28th May 2021**
Dr. Tavpritesh Sethi

Abstract

In clinical research, there is a growing need to process and integrate data from various sources semantically. This report discusses the path to building a federated system for healthcare data that can facilitate standardized data exchange between such stakeholders. We try to achieve our goals by using the semantic interoperability of the Electronic Health Records(EHRs). To materialize this concept, we have tried to use semantic web concepts like OWL and RDF graphs and build an ontology that combines multiple ontologies of different types of data. The ontology is designed by analyzing existing COVID-19 datasets and literature. While building the ontology, we also practice the re-use of existing vocabularies such as SNOMED-CT as well as re-engineering existing ontologies in accordance with the FAIR principles. Further, we discuss an architecture and all the layers and steps involved in integrating and querying on heterogeneous data sources.

Keywords: Clinical Data, Domain Ontology, Knowledge Representation, Semantic Interoperability, COVID-19 Ontology, Ontology re-use, SPARQL.

Acknowledgments

I would like to express my sincere gratitute to my advisors Dr. Raghava Mutharaju and Dr. Tavpritesh Sethi for providing me the opportunity to work on the project and for giving their invaluable guidance, and suggestions. I would also like to thank Rintu Kutum, a Ph.D scholar at CSIR-Institute of Genomics Integrative Biology for his support to help me build an understanding of the healthcare domain.

Work Distribution

This report describes the work that has been carried out by me for the B.tech Project in the semester of Winter 2021.

Contents

Chapter 1

Introduction

Healthcare institutions store their data in some standard that is unknown to other healthcare institutions. This process is a bottleneck in the data exchange between different institutions. Adopting a single standard can, to some extent, resolve this issue but, it will lead to the logistical problem for the institutions to acclimatize to that common standard which may not be feasible for the institutions. Therefore, its suitable to assimilate the various heterogeneous data standards into a single standard at the application level. We try to address the notion of semantic interoperability in the healthcare information systems. There can be either structure, syntactical or semantic interoperability. As compared to structural or syntactical interoperability, semantic interoperability is harder to achieve as meaning of an entity can subjectively vary even in the same information domain [1].

The current approaches for representing healthcare data in electronic health records use information models like HL7, openEHR ISO 13606. These clinical information models are suitable for storing information for use-case-specific problems and data in a systematic, structured format. A significant issue with such clinical models is that they cannot be interoperable with clinical models of different standards or formats. Using such clinical information models guarantees syntactic correctness. However, due to the lack of semantic level granularity, they do not support semantic interoperability and reasoning based on rules [2]. Therefore, we require an approach based on techniques that incorporate semantic web ideas to facilitate the exchange of clinical information.

This report aims to address the idea of semantic interoperability of heterogeneous electronic health records (EHRs) from different sources. As a result, data from various sources can be seamlessly integrated because variations in language and standards, information representation, and storage systems do not create any confusion. To achieve this interoperability level, we will use an ontology-based approach where OWL and RDF ontologies are representation models. We will discuss the reasons for choosing RDF based knowledge graphs over other graph databases as well. Ontologies define relationships between the various entities representing the data in the datasets and provide a meaningful schema for the data. After studying the COVID-19 related domain, we build a domain ontology based on the related entities. The aim is to map the data

from the clinical source to a set of representations based on the ontology that we have created.

Chapter 2

Overview of Problem Statement

Due to the current pandemic, many data related to COVID-19 is involved in various transactions each day. This data is often not easily consumable for low-level analysis unless proper pre-processing is done. It is also the case where many institutions involved in these data transactions do not use a common standard to store or process them. This gives rise to data heterogeneity, which simply means non-standardisation of data across the stakeholders. This becomes a limitation if for example, some other stakeholder who wants to perform data analysis is not adept or experienced with the institute's terminologies or standards that have stored the data. Therefore, the stakeholders of such a platform are not just limited to the data providers but also the data consumers [3]. They may require harmonized and coherent data for application purposes.

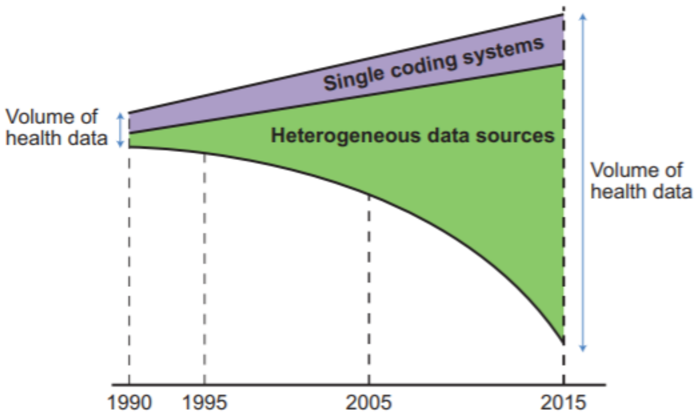


Figure 2.1: Heterogenous Data sources trend [1]

There is a need for a platform that facilitates the exchange of data amongst various stakeholders. Hence, semantic interoperability in the healthcare sector is becoming an increasingly active field of study. To facilitate the interchange of healthcare-related information, we need a robust architecture at the semantic level that would incorporate various types of data coming from diverse and heterogeneous sources in the form of Electronic Health Records.

The idea is to built using ontologies based on semantic web technologies like RDF(Resource Description Framework) and OWL(Web Ontology Language). The main principle behind using the semantic web for interoperability is FAIR principles. In FAIR, “I” stands for interoperability required for data exchange amongst multiple stakeholders, and “R” stands for re-usability of existing domain vocabularies and terminologies [4].

Since SARS-NCov2 is a new virus, the data is scarce. Most of the data is not publicly available as institutions can’t provide patient level data without authentication. At the time of writing this report we have data which is fragmented and scattered which cannot be immediately used. Therefore, we have tried to analyse the data, create attributes and limit the scope of the architecture to incorporate the data, to create a framework that standardizes the medical data for COVID-19 analysis.

Chapter 3

Related Work

Researchers are constantly trying to make the healthcare domain more interoperable due to the vast heterogeneous data volume. Mappings between different standards and domain ontologies are constructed to facilitate the interoperability of EHRs represented with multiple standards. Costa et. [5] built archetype-based models by developing source ontologies for the openEHR and ISO 13606 standards. Archetypes define clinical concepts such as heart rate, some laboratory tests, and blood pressure measurements. An archetype could be specified as a subset of another, contain a fragment of another archetype, be used in conjunction with others via templates. Further, they develop a core ontology that incorporates the domain semantics of the two standards. This kind of data transformation is done by syntactically mapping the archetypes defined by openEHR and ISO 13606 and archetypes defined by the domain ontology.

Gonçalves et al. [6] create an ECG domain ontology and translate the clinical terminologies from three ECG standards to the ontology directly. It was also noted that they faced difficulties developing the domain ontology to which the different ECG standards were mapped.

Berges et al. [7] uses data in relational database format and first obtains the ontological representation of the relevant databases. With the domain ontologies now created, they try to map these domain ontologies to the canonical ontology. This canonical ontology reuses SNOMED-CT and LOINC terminologies.

Most of the work done to map the source data to their domain semantic representation defined by the ontologies uses a similar architecture defined by multiple layers. Sun et al. [3] describe multiple data layers and propose a method for conversion to OWL-based ontology. They also discuss RESTful services to extract the loaded data for application purposes.

Chapter 4

Architecture and Methodology

In the previous semester, we had developed the domain ontology after studying the domain terminologies and entities. We also keep in mind the FAIR principles [4] while developing the ontology.

The advantages of using an Ontology based approach for semantic interoperability are as follows [7]:

1. *The use of ontologies makes sure to focus on the relevant entities rather than the languages or formats of information used to describe the EHRs.*
2. *The schema defined using ontologies is flexible and can easily accomodate any EHRs in a new format.*
3. *The framework that the ontology is built upon is defined by OWL2, using which we can define various reasoning mechanisms based on axioms and properties. These mechanisms will help in inferring and discovering new relations between the various heterogeneous healthcare data sources.*

In the subsequent sections, we describe the architecture that we propose, along with the it's various layers and their implementation.

4.1 Understanding Knowledge Graphs and Ontology

An ontology is a description of the concepts and relationships that helps in realizing the knowledge representation of the domain for which the platform is being built. Ontology modeling helps in sharing a common understanding of the structure of information among the various agents and stakeholders involved, making domain assumptions explicit, reusing domain knowledge, separating domain knowledge from operational knowledge, and analyzing the domain knowledge [8].

Ontology -

- International Resource Identifier (IRI)
- Resource Description Framework (RDF)
- Resource Description Framework Scheme (RDFS)
- Web Ontology Language (OWL)
- SPARQL Protocol and RDF Query Language (SPARQL)

An IRI looks similar to an URL. The main difference is that the URL points to a resource to be displayed in the web browser. IRIs are more general and point to the resources defined under the semantic web standards. We can define any resource such as class, property, and individuals using IRI. Technically, IRIs are a superset of URLs.

RDF is a language for describing IRIs as knowledge graphs. Data in RDF graphs are written as triples of the form subject-predicate-object. There are many different formats of representing RDF graphs like, XML and Turtle.

RDFS is layered on top of RDF and provides functionalities to define classes, subclasses, properties, subproperties for the resources. In general the schema of the graph.

OWL is layered on top of RDF and provides semantics for the schema of the knowledge graph. OWL is an implementation of Description Logic, which is a decidable subset of First Order Logic. OWL supports reasoning services, and provides automated theorem provers. These reasoning services check for consistency in the schema and highlight the inconsistencies. For example, let us define a property *isAncestorOf* which is defined as transitive using *owl:TransitiveProperty*. Let Person be a class with individuals *A, B, C*. Let us have triples defined as *A isAncestorOf B* and *B isAncestorOf C*, then reasoners can infer that *A isAncestorOf C*. If a contradiction in data is observed for example, *C isAncestorOf A*, then reasoners will highlight this inconsistency.

4.1.1 Implementing the Ontology

The implementation methodology to build the ontology is loosely inspired by the "METHONTOLOGY" framework [9], which describes an ontology development process. The "methontology" framework emphasises building the ontology either from scratch, reusing existing ontologies or re-engineering existing ontology.

After defining the problem statement and the purpose of the ontology, we identified and limited the domain's scope, which is part of knowledge acquisition. This was a necessary and crucial step since the healthcare domain is vast, so we needed to use only the necessary concepts. As discussed earlier, only a handful of data is available on the web, and even the data available was either incomplete or fragmented. Despite these limitations, we studied the attributes, the variables to understand what parameters should be used within the ontology. Based on the metadata of the datasets and the types of information that EHRs carry, we get an idea about the clinical terminologies and define our domain semantics, the core ontology.

We have also re-used some concepts defined in CODO (Covid Ontology for cases and patient information) [10] ontology released for the public usage by ISI Bangalore. It extensively defines the classes and properties to deal with the incoming clinical data. Although the scope of that ontology is very vast, we have tried to re-use only the portion that is in accordance with our use-case.

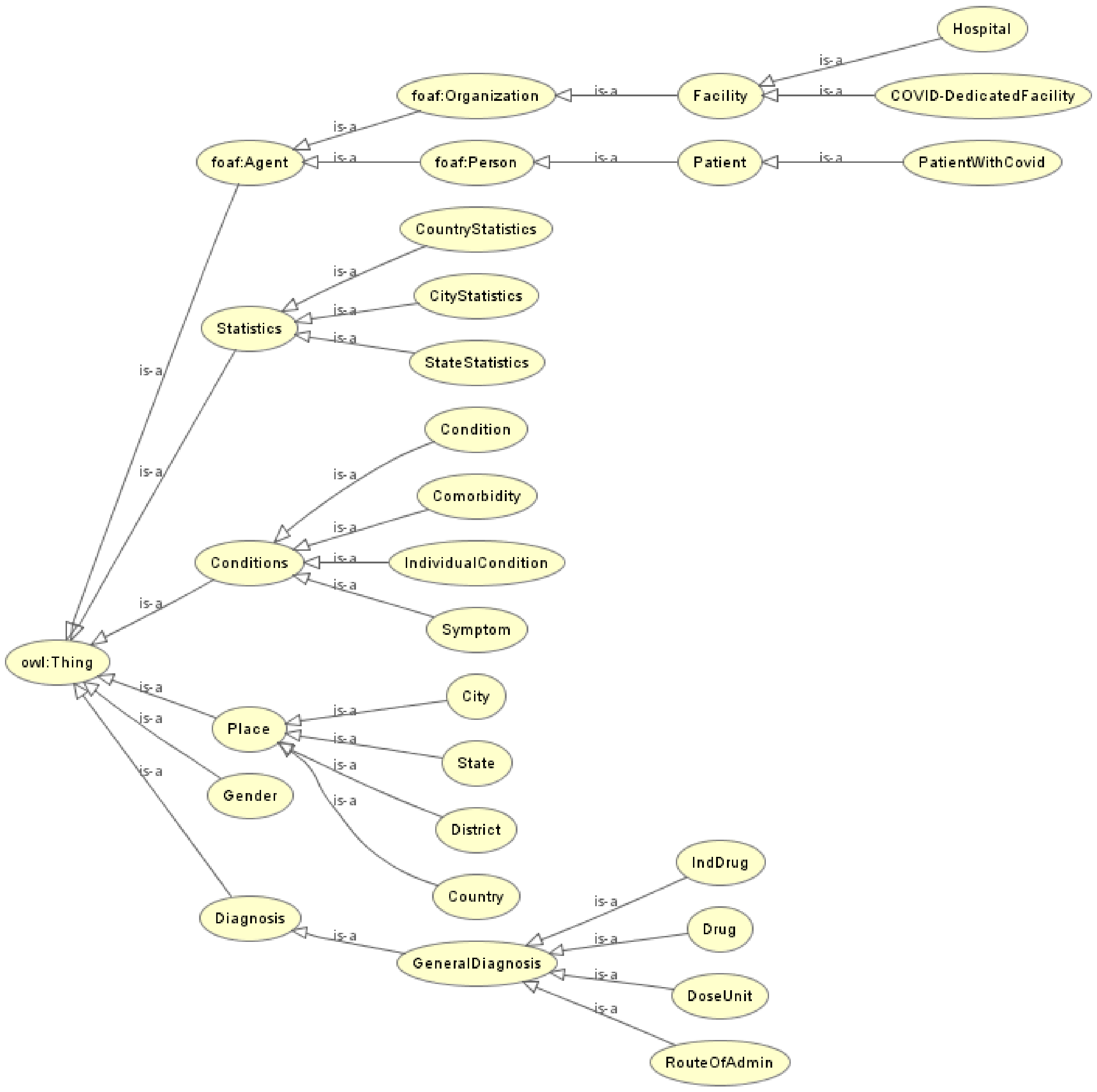
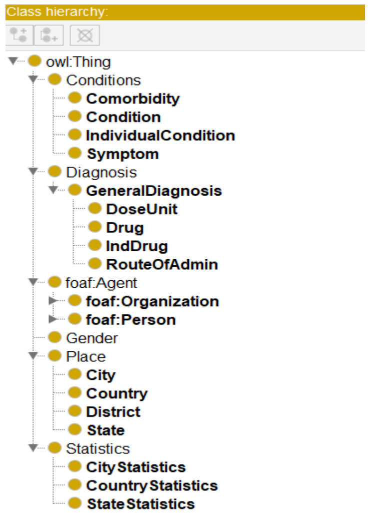
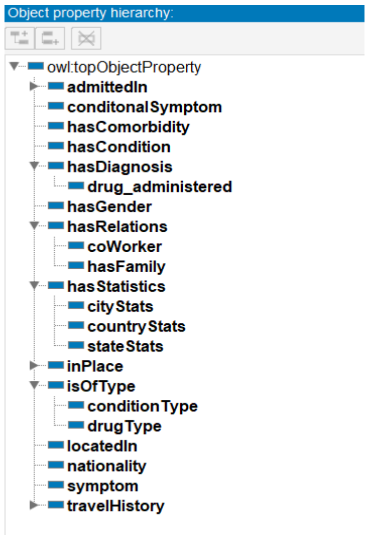


Figure 4.1: Concept Hierarchy in Ontology



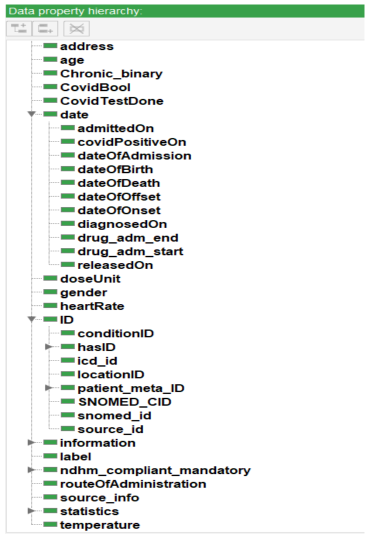
```
Class hierarchy:
▼ owl:Thing
    ▼ Conditions
        Comorbidity
        Condition
        IndividualCondition
        Symptom
    ▼ Diagnosis
        ▼ GeneralDiagnosis
            DoseUnit
            Drug
            IndDrug
            RouteOfAdmin
    ▼ foaf:Agent
        ► foaf:Organization
        ► foaf:Person
    Gender
    ▼ Place
        City
        Country
        District
        State
    ▼ Statistics
        CityStatistics
        CountryStatistics
        StateStatistics
```

(a) Classes for Clinical Ontology



```
Object property hierarchy:
▼ owl:topObjectProperty
    ► admittedIn
    conditonalSymptom
    hasComorbidity
    hasCondition
    ▼ hasDiagnosis
        drug_administered
    hasGender
    ▼ hasRelations
        coWorker
        hasFamily
    ▼ hasStatistics
        cityStats
        countryStats
        stateStats
    ► inPlace
    ▼ isOfType
        conditionType
        drugType
    locatedIn
    nationality
    symptom
    ► travelHistory
```

(b) Object Properties for Clinical Ontology



```
Data property hierarchy:
    address
    age
    Chronic_binary
    CovidBool
    CovidTestDone
  ▼ date
        admittedOn
        covidPositiveOn
        dateOfAdmission
        dateOfBirth
        dateOfDeath
        dateOfOffset
        dateOfOnset
        diagnosedOn
        drug_adm_end
        drug_adm_start
        releasedOn
    doseUnit
    gender
    heartRate
  ▼ ID
        conditionID
      ► hasID
        icd_id
        locationID
      ► patient_meta_ID
        SNOMED_CID
        snomed_id
        source_id
  ► information
    label
  ► ndhm_compliant_mandatory
    routeOfAdministration
    source_info
  ► statistics
    temperature
```

(c) Data Properties for Clinical Ontology

Figure 4.2: Clinical ontology Concepts and Properties

4.2 Overview of the Architecture

In the previous section we described the advantages of ontology, and we build an ontology in Protege. Now, we describe the architecture to semantically process the heterogeneous data to the domain semantics described by the ontology. The architecture has been divided into primarily three layers and three steps [3]. In the subsequent sections we will describe each layer and step of the architecture.

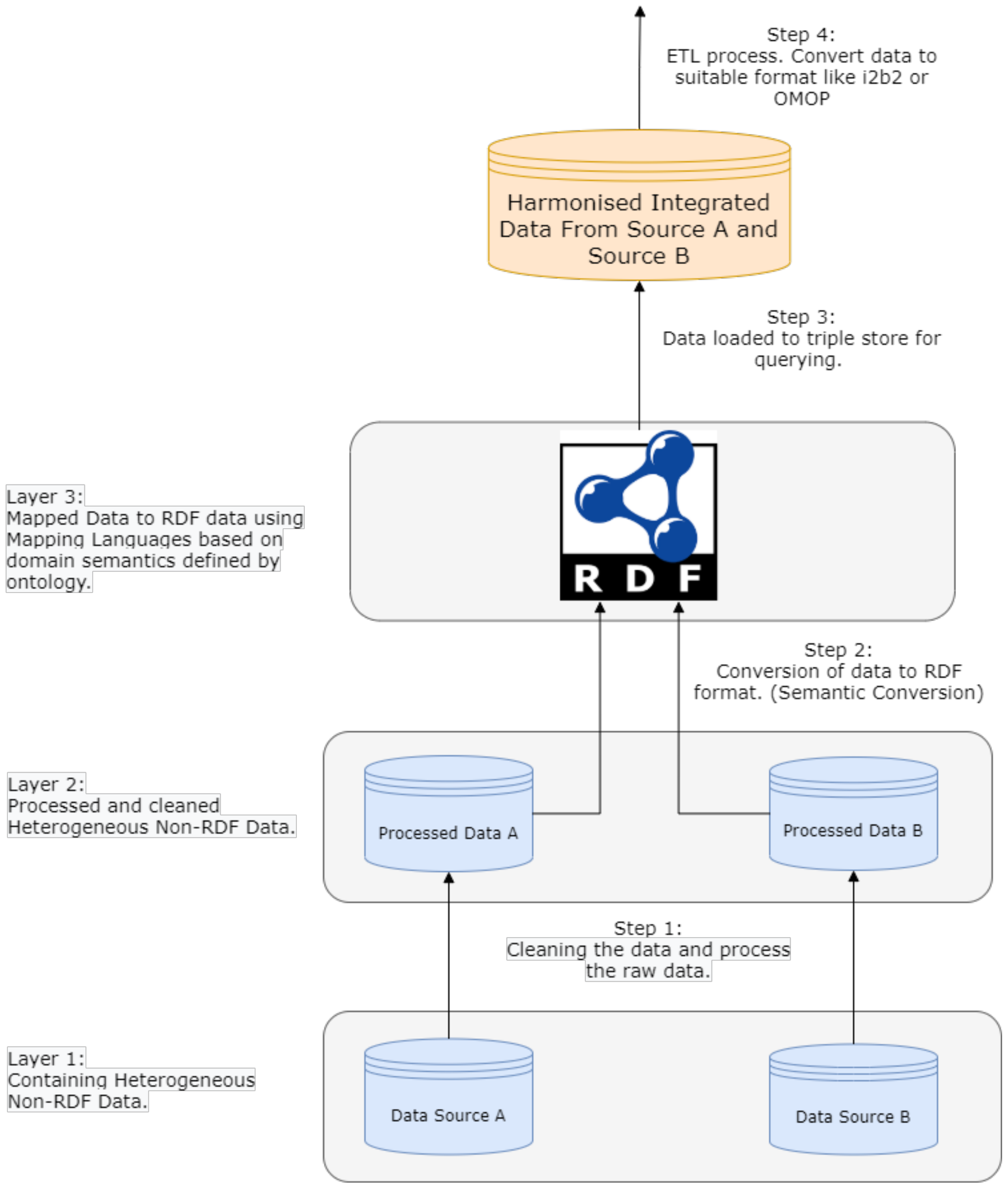

Step 4:
ETL process. Convert data to suitable format like i2b2 or OMOP
Harmonised Integrated Data From Source A and Source B
Step 3:
Data loaded to triple store for querying.
Layer 3:
Mapped Data to RDF data using Mapping Languages based on domain semantics defined by ontology.
R D F
Step 2:
Conversion of data to RDF format. (Semantic Conversion)
Layer 2:
Processed and cleaned Heterogeneous Non-RDF Data.
Processed Data A
Processed Data B
Step 1:
Cleaning the data and process the raw data.
Layer 1:
Containing Heterogeneous Non-RDF Data.
Data Source A
Data Source B

Figure 4.3: Application Architecture

4.2.1 Layer 1

As seen in Figure 4.3 In layer 1, data sets from multiple data sources (in the form of EHRs) exist in their raw formats or standards. These data sets are heterogeneous and uploaded by healthcare institutions. These datasets are loaded from the data warehouses of the institutions. They contain data in unstructured formats, and therefore it is necessary to process them before converting the data to their domain semantics. This raw data is cleaned in step 1. We look for specific data types such as DateTime, floats, strings and clean the external noise.

4.2.2 Layer 2

This layer 2 represents the data processed from the previous layer in step 1. The data in this layer is now structured, and the data types have been corrected. The primary goal for this data cleaning was to convert EHR to a format that can be used by the mapping language like RML or APIs for conversion to RDF data. In addition to this, the data can also use data to train models since the data has been cleaned. [5]

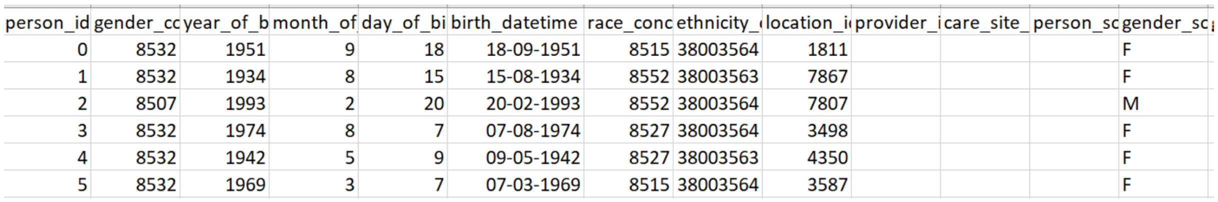
4.2.3 Layer 3

The processed data from layer two is mapped to RDF triples dataset. RDF data as triples is a knowledge graph built using the domain ontology. This data is now in a standardized structure defined by the ontology, also called the domain semantics. This mapping can be done either by using an API or a mapping language like RML [11], which stands for RDF Mapping Language. Later while discussing the implementation, we will explain why choosing a mapping language-based approach instead of an API-based approach and the difficulties we faced. Since each data source has its own RDF graph, the data sets are still heterogeneous but harmonized as they are converted to the domain semantics.
Each RDF graph is loaded to a triple store. Every RDF graph is named uniquely in the triple store. Despite uniquely named, the data is now integrated and no more heterogeneous. We can now query on this integrated RDF data using SPARQL. SPARQL is a query language using which we can retrieve and manipulate RDF data. An interface for facilitating an ETL (Extract, Transform, Load) based platform can be built if the data needs to be converted into an application-friendly format like OMOP or i2b2.

4.3 Implementation

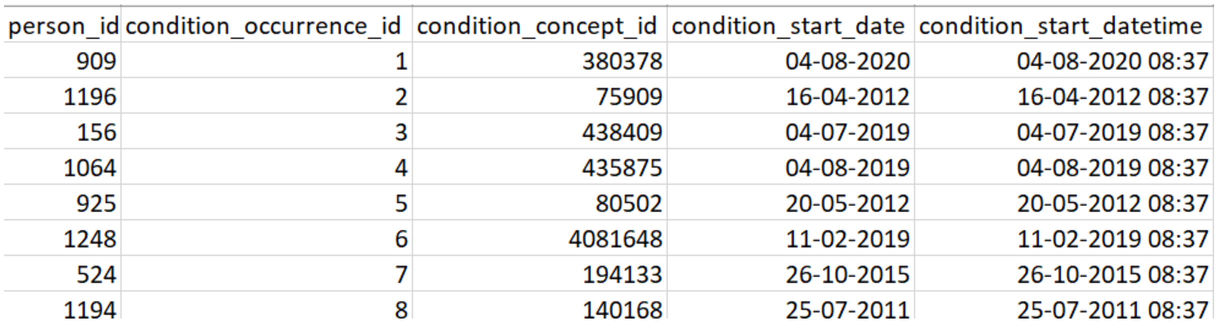
In this section, we will discuss the implementation of the architecture defined previously with the help of an example. For our example, we will be using publicly available datasets based on real use-cases. These datasets are synthetically curated data in the form of OMOP tables for the EHR COVID-19 Dream Challenge. The datasets that we will be using are treated as independent data sources in layer one of the architecture. All the datasets are in CSV format, and each row signifies an EHR. The descriptions of the corresponding datasets are:

1. Person Table: The person tables stores demographic information about all the patients in the dataset. The Person table contains records that uniquely identify each patient in the data. Refer 4.4
2. Condition Occurrence Table: This table describes records of a Person suggesting the presence of a disease or medical condition stated as a diagnosis, a sign, or a symptom, which is either observed by a Provider or reported. Refer 4.5
3. Drug Exposure Table: Rows of this table describe the drug exposure history of the patients. The 'Drug' domain captures records about the utilization of a Drug when ingested or otherwise introduced into the body of a person. Refer 4.6



person_id	gender_cc	year_of_b	month_of	day_of_bi	birth_datetime	race_conc	ethnicity_	location_i	provider_i	care_site_	person_sc	gender_sc
0	8532	1951	9	18	18-09-1951	8515	38003564	1811				F
1	8532	1934	8	15	15-08-1934	8552	38003563	7867				F
2	8507	1993	2	20	20-02-1993	8552	38003564	7807				M
3	8532	1974	8	7	07-08-1974	8527	38003564	3498				F
4	8532	1942	5	9	09-05-1942	8527	38003563	4350				F
5	8532	1969	3	7	07-03-1969	8515	38003564	3587				F

Figure 4.4: Person Table

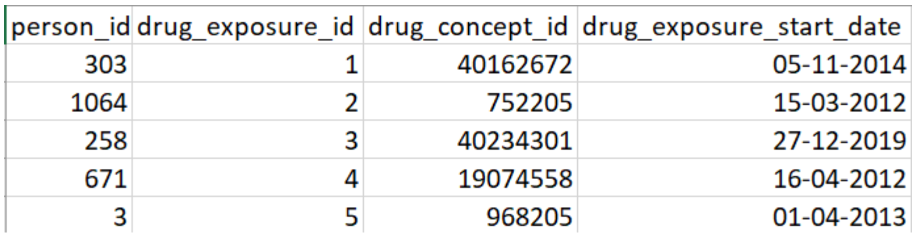


person_id	condition_occurrence_id	condition_concept_id	condition_start_date	condition_start_datetime
909	1	380378	04-08-2020	04-08-2020 08:37
1196	2	75909	16-04-2012	16-04-2012 08:37
156	3	438409	04-07-2019	04-07-2019 08:37
1064	4	435875	04-08-2019	04-08-2019 08:37
925	5	80502	20-05-2012	20-05-2012 08:37
1248	6	4081648	11-02-2019	11-02-2019 08:37
524	7	194133	26-10-2015	26-10-2015 08:37
1194	8	140168	25-07-2011	25-07-2011 08:37

Figure 4.5: Condition Table

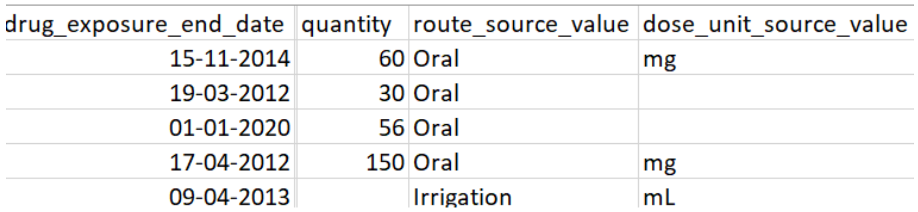
The Condition occurrence table and the drug exposure table have the concept IDs of the conditions and the drugs respectively. These concept IDs are SNOMED-CT codes. We use a dictionary containing codes for conditions, symptoms and drugs to map these codes to their clinical names.

These three datasets are in raw form in layer 1. We do some processing on each data, for example, correcting the format of dates, adding a new column like age of the person in the table 4.4 from the existing column values. We also map the concept id of conditions and drugs to their clinical terms using the SNOMED-CD code dictionary. We now have the transformed/processed data that forms the view of layer 2.



person_id	drug_exposure_id	drug_concept_id	drug_exposure_start_date
303	1	40162672	05-11-2014
1064	2	752205	15-03-2012
258	3	40234301	27-12-2019
671	4	19074558	16-04-2012
3	5	968205	01-04-2013

(a) Drug Table 1



drug_exposure_end_date	quantity	route_source_value	dose_unit_source_value
15-11-2014	60	Oral	mg
19-03-2012	30	Oral	
01-01-2020	56	Oral	
17-04-2012	150	Oral	mg
09-04-2013		Irrigation	mL

(b) Drug Table 2

Figure 4.6: Drug Table

4.3.1 Conversion to Knowledge Graphs

The next step is to convert these heterogeneous data sources to their meaningful domain semantics. By this, we mean that we want an approach to convert the data in layer 2 to a knowledge graph with edges (the properties) and nodes (entities like each column info) based structure based on the vocabulary defined by the ontology. To reiterate the advantage of this process, we convert the data into an ontology defined schema that standardizes all the independent data from various heterogeneous sources into a homogeneous format.

Neo4j and RDF graphs are two technologies available for the creation of knowledge graphs. These graphs are stored as labelled property graphs or LPG in Neo4j and as triple stores in RDF. The issues with Neo4j for creating the knowledge graphs were that it does not have a robust support for OWL2 based ontologies. Also, it uses cypher as the query language which I am not familiar with. We opted for RDF as it's rules are coherent with the OWL ontologies, and RDF SPARQL as the query language.

We use YARRML, an extension of RML (RDF mapping language) that converts CSV to RDF triples data based on the mapping written. YAML is used to write the mappings in YARRML. Below is a code snippet that shows this conversion for Person table. 4.4.

```
prefixes:
  ex: http://www.example.org/btp_ontology/individual/
  schema: http://www.semanticweb.org/btp/clinical/ontology/
  xsd: http://www.w3.org/2001/XMLSchema#
mappings:
  person:
    sources:
      - ['person_transformed.csv~csv']
```

```
        s: ex:patient_$(person_id)
        po:
          - [a, schema:Person]
          - [schema:hasPatientID, subject_id_$(person_id)]
          - [schema:gender, $(gender_source_value)]
          - p: schema:age
            o:
              value: $(age)
              datatype: xsd:int

          - p: schema:dateOfBirth
            o:
              value: $(birth_datetime)
              datatype: xsd:datetime
```

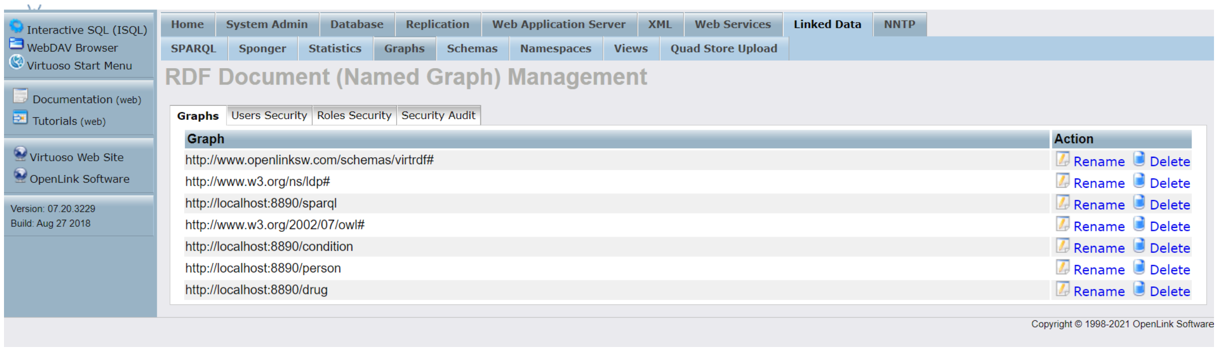
In the snippet, the mapping looks for the person IDs in Person table 4.4 and maps the corresponding columns such as gender, age, dateOfBirth. Below are the RDF triples in the turtle format (a format to represent RDF triples), for row corresponding to patient with id 0.

```
ex:patient_0 a schema:Person;
  schema:age "69"^^xsd:int;
  schema:dateOfBirth "1951-09-18"^^xsd:datetime;
  schema:gender "F";
  schema:hasPatientID "subject_id_0" .
```

Similar mappings are written for the Condition Occurrence and Drug Exposure datasets. Each dataset has a separate mapping and a separate RDF graph. Let us call the mappings

4.3.2 Integrated Data and Querying

Now we have the RDF graphs as triples in turtle files (.ttl) for all the datasets. All the three RDF graphs are now following the same set of vocabularies defined by the ontology. We want to load them to a triple store as shown in the step 3 of the architecture 4.3. The triple store we use is Openlink Virtuoso. We can simply upload the turtle files to the triple store, and provide a unique identifier to the graph in the form of a URI.



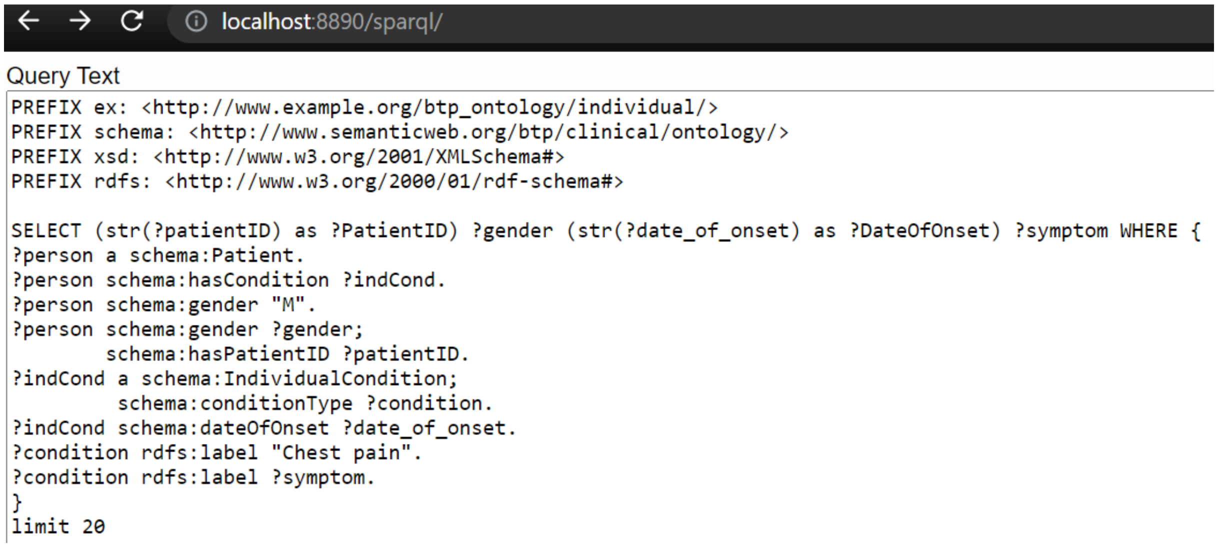
Graph	Action
http://www.openlinksw.com/schemas/virtrdf#	Rename Delete
http://www.w3.org/ns/ldp#	Rename Delete
http://localhost:8890/sparql	Rename Delete
http://www.w3.org/2002/07/owl#	Rename Delete
http://localhost:8890/condition	Rename Delete
http://localhost:8890/person	Rename Delete
http://localhost:8890/drug	Rename Delete

Figure 4.7: Uploaded Graphs in the Triple store

Once the graphs are uploaded, the datasets are now homogeneously integrated in the triple store. We can now query on the integrated data using a SPARQL endpoint located at http://localhost:8890/sparql.

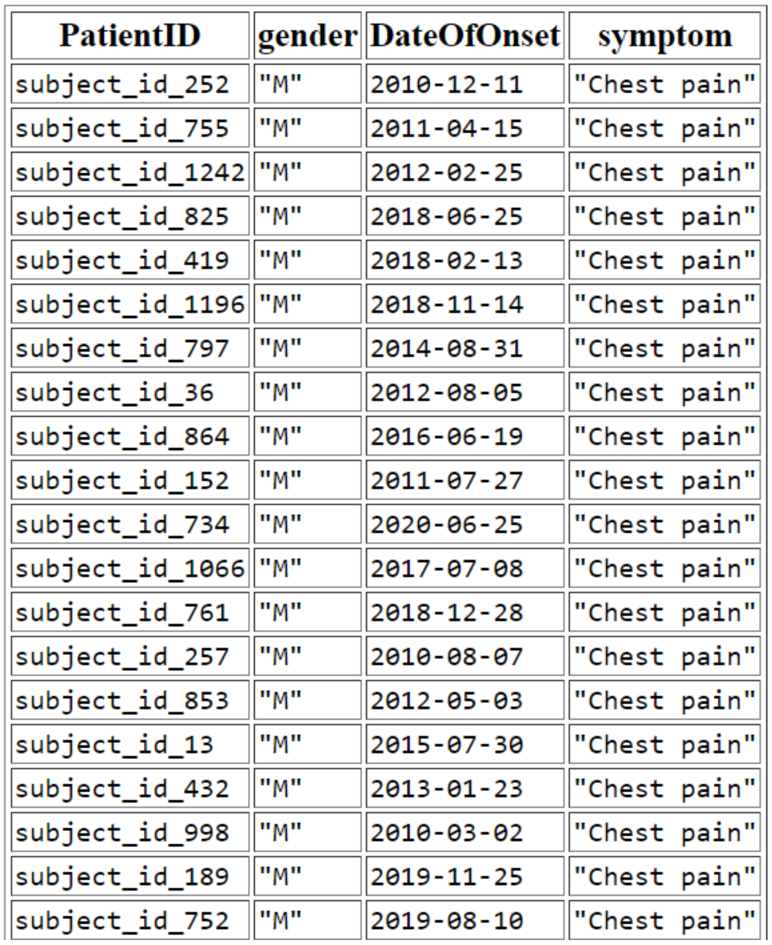
Some example queries are given below.

1. Query 1: Get all the Male Patients with chest pain and the date of onset of the symptom of such patients. This query uses the patient table and condition occurrence table. fig 4.8
2. Query 2: Get the distinct drugs administered to people with Dyspnea. This query uses condition occurrence table and drug exposure table. fig 4.10
3. Query 3: Get the Date of Onset of patients with condition SNOMED ID 433736. This query uses patient table and condition occurrence table. fig 4.12
4. Query 4: Get the Routes of drug administration for people with age greater than 60 having fever. This query uses patient table, condition occurrence table and drug exposure table. fig 4.14



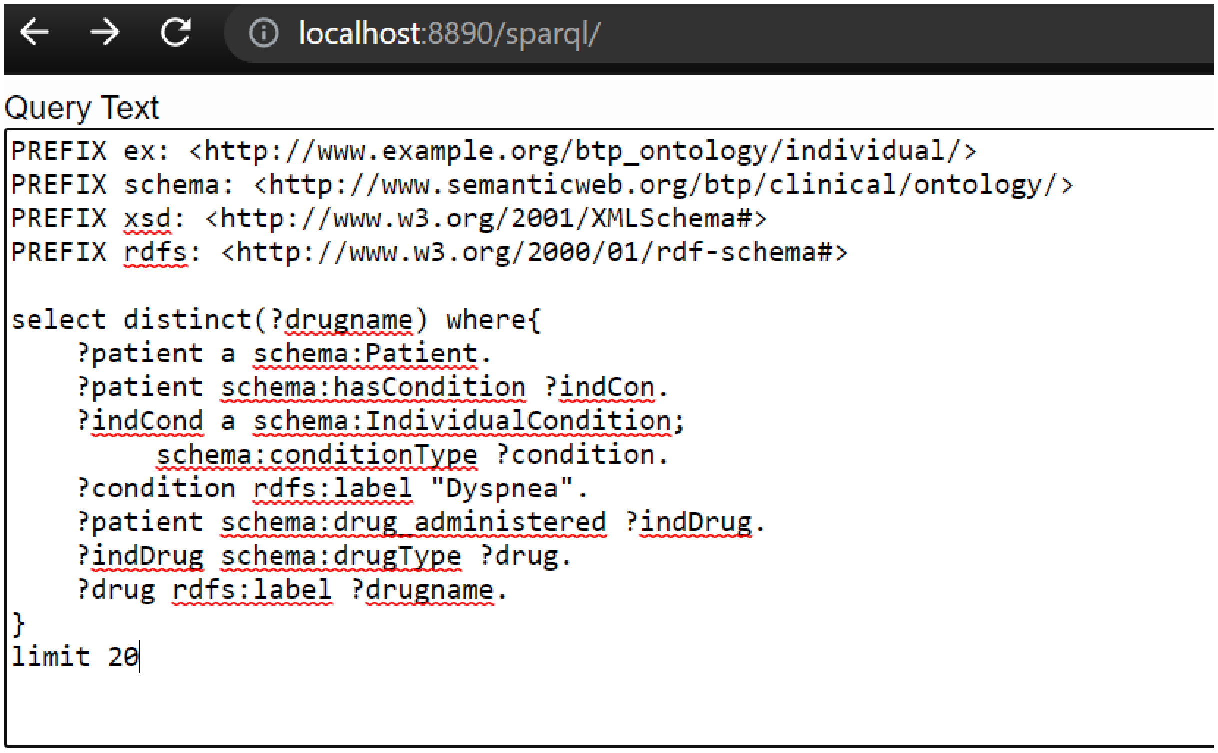
```
PREFIX ex: <http://www.example.org/btp_ontology/individual/>
PREFIX schema: <http://www.semanticweb.org/btp/clinical/ontology/>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>

SELECT (str(?patientID) as ?PatientID) ?gender (str(?date_of_onset) as ?DateOfOnset) ?symptom WHERE {
?person a schema:Patient.
?person schema:hasCondition ?indCond.
?person schema:gender "M".
?person schema:gender ?gender;
        schema:hasPatientID ?patientID.
?indCond a schema:IndividualCondition;
         schema:conditionType ?condition.
?indCond schema:dateOfOnset ?date_of_onset.
?condition rdfs:label "Chest pain".
?condition rdfs:label ?symptom.
}
limit 20
```

Figure 4.8: Query 1



PatientID	gender	DateOfOnset	symptom
subject_id_252	"M"	2010-12-11	"Chest pain"
subject_id_755	"M"	2011-04-15	"Chest pain"
subject_id_1242	"M"	2012-02-25	"Chest pain"
subject_id_825	"M"	2018-06-25	"Chest pain"
subject_id_419	"M"	2018-02-13	"Chest pain"
subject_id_1196	"M"	2018-11-14	"Chest pain"
subject_id_797	"M"	2014-08-31	"Chest pain"
subject_id_36	"M"	2012-08-05	"Chest pain"
subject_id_864	"M"	2016-06-19	"Chest pain"
subject_id_152	"M"	2011-07-27	"Chest pain"
subject_id_734	"M"	2020-06-25	"Chest pain"
subject_id_1066	"M"	2017-07-08	"Chest pain"
subject_id_761	"M"	2018-12-28	"Chest pain"
subject_id_257	"M"	2010-08-07	"Chest pain"
subject_id_853	"M"	2012-05-03	"Chest pain"
subject_id_13	"M"	2015-07-30	"Chest pain"
subject_id_432	"M"	2013-01-23	"Chest pain"
subject_id_998	"M"	2010-03-02	"Chest pain"
subject_id_189	"M"	2019-11-25	"Chest pain"
subject_id_752	"M"	2019-08-10	"Chest pain"

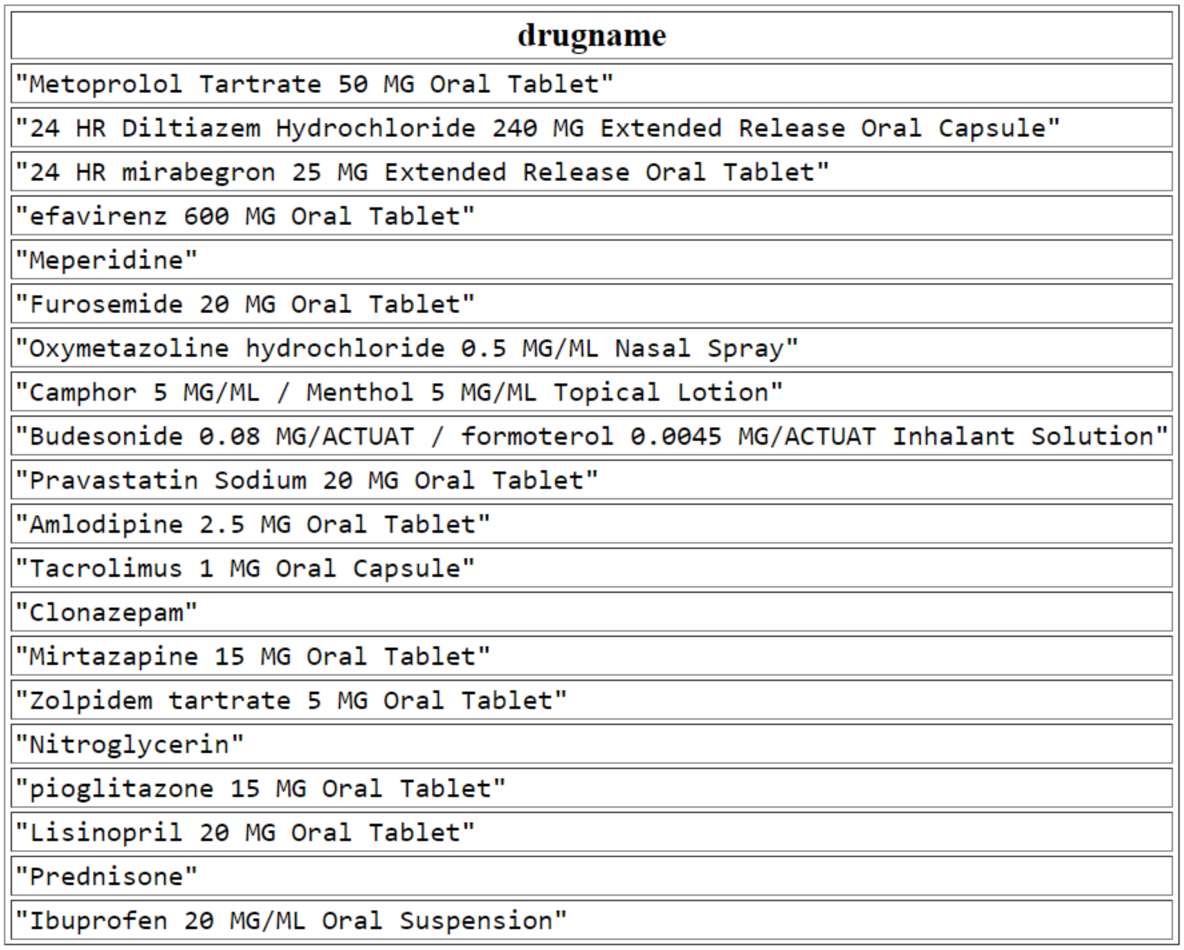
Figure 4.9: Result for the query in Fig 4.8


localhost:8890/sparql/

Query Text

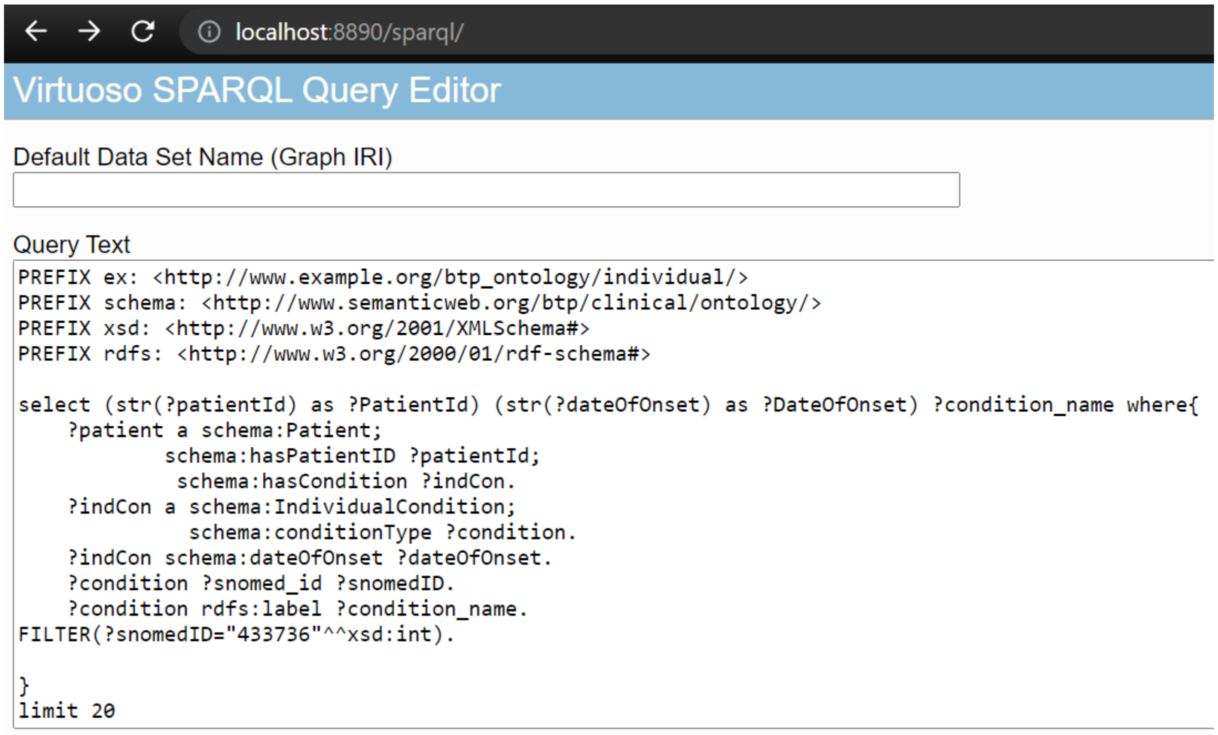
```
PREFIX ex: <http://www.example.org/btp_ontology/individual/>
PREFIX schema: <http://www.semanticweb.org/btp/clinical/ontology/>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>

select distinct(?drugname) where{
    ?patient a schema:Patient.
    ?patient schema:hasCondition ?indCon.
    ?indCond a schema:IndividualCondition;
         schema:conditionType ?condition.
    ?condition rdfs:label "Dyspnea".
    ?patient schema:drug_administered ?indDrug.
    ?indDrug schema:drugType ?drug.
    ?drug rdfs:label ?drugname.
}
limit 20
```

Figure 4.10: Query 2



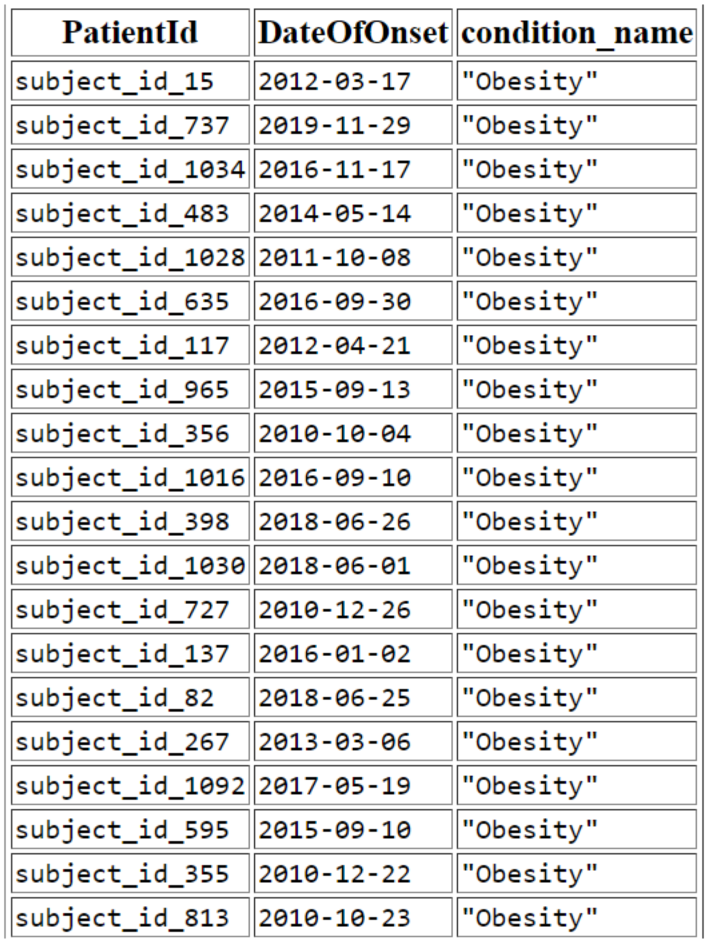
drugname
"Metoprolol Tartrate 50 MG Oral Tablet"
"24 HR Diltiazem Hydrochloride 240 MG Extended Release Oral Capsule"
"24 HR mirabegron 25 MG Extended Release Oral Tablet"
"efavirenz 600 MG Oral Tablet"
"Meperidine"
"Furosemide 20 MG Oral Tablet"
"Oxymetazoline hydrochloride 0.5 MG/ML Nasal Spray"
"Camphor 5 MG/ML / Menthol 5 MG/ML Topical Lotion"
"Budesonide 0.08 MG/ACTUAT / formoterol 0.0045 MG/ACTUAT Inhalant Solution"
"Pravastatin Sodium 20 MG Oral Tablet"
"Amlodipine 2.5 MG Oral Tablet"
"Tacrolimus 1 MG Oral Capsule"
"Clonazepam"
"Mirtazapine 15 MG Oral Tablet"
"Zolpidem tartrate 5 MG Oral Tablet"
"Nitroglycerin"
"pioglitazone 15 MG Oral Tablet"
"Lisinopril 20 MG Oral Tablet"
"Prednisone"
"Ibuprofen 20 MG/ML Oral Suspension"

Figure 4.11: Result for the query in Fig 4.10


localhost:8890/sparql/
Virtuoso SPARQL Query Editor
Default Data Set Name (Graph IRI)
Query Text

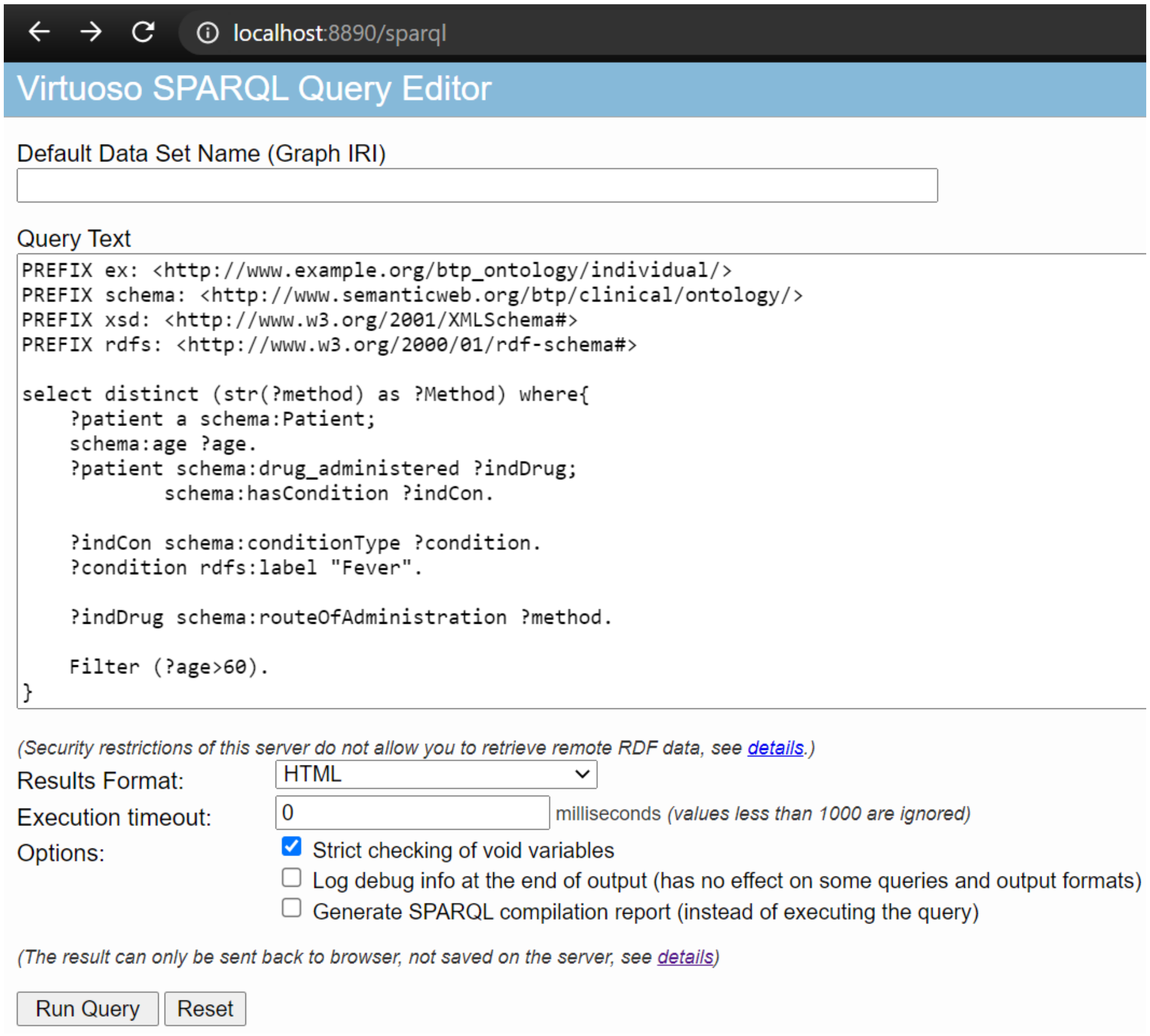
```
PREFIX ex: <http://www.example.org/btp_ontology/individual/>
PREFIX schema: <http://www.semanticweb.org/btp/clinical/ontology/>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>

select (str(?patientId) as ?PatientId) (str(?dateOfOnset) as ?DateOfOnset) ?condition_name where{
    ?patient a schema:Patient;
            schema:hasPatientID ?patientId;
             schema:hasCondition ?indCon.
    ?indCon a schema:IndividualCondition;
              schema:conditionType ?condition.
    ?indCon schema:dateOfOnset ?dateOfOnset.
    ?condition ?snomed_id ?snomedID.
    ?condition rdfs:label ?condition_name.
FILTER(?snomedID="433736"^^xsd:int).

}
limit 20
```

Figure 4.12: Query 3



PatientId	DateOfOnset	condition_name
subject_id_15	2012-03-17	"Obesity"
subject_id_737	2019-11-29	"Obesity"
subject_id_1034	2016-11-17	"Obesity"
subject_id_483	2014-05-14	"Obesity"
subject_id_1028	2011-10-08	"Obesity"
subject_id_635	2016-09-30	"Obesity"
subject_id_117	2012-04-21	"Obesity"
subject_id_965	2015-09-13	"Obesity"
subject_id_356	2010-10-04	"Obesity"
subject_id_1016	2016-09-10	"Obesity"
subject_id_398	2018-06-26	"Obesity"
subject_id_1030	2018-06-01	"Obesity"
subject_id_727	2010-12-26	"Obesity"
subject_id_137	2016-01-02	"Obesity"
subject_id_82	2018-06-25	"Obesity"
subject_id_267	2013-03-06	"Obesity"
subject_id_1092	2017-05-19	"Obesity"
subject_id_595	2015-09-10	"Obesity"
subject_id_355	2010-12-22	"Obesity"
subject_id_813	2010-10-23	"Obesity"

Figure 4.13: Result for the query in Fig 4.12


localhost:8890/sparql
Virtuoso SPARQL Query Editor
Default Data Set Name (Graph IRI)
Query Text

```
PREFIX ex: <http://www.example.org/btp_ontology/individual/>
PREFIX schema: <http://www.semanticweb.org/btp/clinical/ontology/>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>

select distinct (str(?method) as ?Method) where{
    ?patient a schema:Patient;
    schema:age ?age.
    ?patient schema:drug_administered ?indDrug;
            schema:hasCondition ?indCon.

    ?indCon schema:conditionType ?condition.
    ?condition rdfs:label "Fever".

    ?indDrug schema:routeOfAdministration ?method.

    Filter (?age>60).
}
```

(Security restrictions of this server do not allow you to retrieve remote RDF data, see details.)

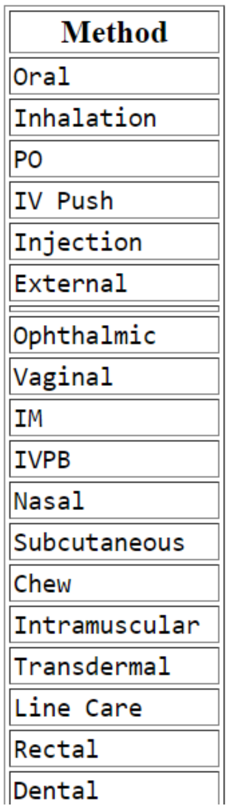
Results Format: HTML

Execution timeout: 0 milliseconds *(values less than 1000 are ignored)*

Options:
- ✓ Strict checking of void variables
- ☐ Log debug info at the end of output (has no effect on some queries and output formats)
- ☐ Generate SPARQL compilation report (instead of executing the query)

(The result can only be sent back to browser, not saved on the server, see details)

Figure 4.14: Query 4



Method
Oral
Inhalation
PO
IV Push
Injection
External
Ophthalmic
Vaginal
IM
IVPB
Nasal
Subcutaneous
Chew
Intramuscular
Transdermal
Line Care
Rectal
Dental

Figure 4.15: Result for the query in Fig 4.14

4.4 Future Work

In the architecture fig 4.3 the data in layer 1 is raw, unprocessed, and unstructured, which requires a step to process this data and store it as structured and clean data in layer 2. In the current example, we already know the attributes of the raw datasets; hence pre-processing steps are already apparent. Often, the structure of the incoming raw data is not known. Therefore, we need a technique that automates the pre-processing.

Similarly, in the conversion of data in layer 2 to RDF graphs, we will not always know the structure of the data. This conversion needs to be automated using some NLP-based interface, which identifies the attributes and maps them to their domain semantics. Moving further, we also want to incorporate a system to check the validity of the RDF graphs built using ontology schema using validation tools such as SHACL. For example, if ontology has an axiom where some drugs cannot be taken together, then validation should be able to highlight such contradictions in the data.

Finally, the data currently being used is integrated at a low-level granularity. By this, we mean that all three datasets are stored in the same triple store. While this solves the problem of standardization into domain semantics, most datasets in real life are isolated and hence require a dedicated triple store on the end of the data provider similar to a microservice.

Chapter 5

Summary

An increase in the volume of data in healthcare has lead to an increase in heterogeneous data sources. This growth of healthcare information means more and more data is non-standardized and non-interoperable amongst the stakeholders. We use an ontology-based approach to make the heterogeneous data more interoperable. After discussing the advantages of ontologies, we build a domain ontology that defines the schema for the clinical data. With the help of domain ontology, we now have a way to standardize the data and facilitate the data exchange better.

Further, We discuss the architecture that we follow and all the layers and steps in the architecture. Finally, we take a real example, where we have three datasets in OMOP format. We convert each of the datasets to RDF graphs based on concepts defined in the ontology. After uploading the RDF graphs Virtuoso triple store, we query on the integrated data using SPARQL. Finally, we discuss the scope for future work.

The github repository contains the codebase including the mappings to convert CSV to RDF, as well as the SPARQL queries. https://github.com/raghav17083/FederatedHealthcarePlatform

Bibliography

[1] H. Liyanage, P. Krause, S. de Lusignan, Using ontologies to improve semantic interoperability in health data, Journal of innovation in health informatics 22 (2015) 309–15. doi:10.14236/jhi.v22i2.159.

[2] C. Martínez-Costa, S. Schulz, Validating ehr clinical models using ontology patterns, Journal of Biomedical Informatics 76 (2017) 124–137. doi:https://doi.org/10.1016/j.jbi.2017.11.001.
URL https://www.sciencedirect.com/science/article/pii/S1532046417302381

[3] H. Sun, K. Depraetere, J. De Roo, G. Mels, B. De Vloed, M. Twagirumukiza, D. Colaert, Semantic processing of ehr data for clinical research, Journal of Biomedical Informatics 58 (2015) 247–259. doi:https://doi.org/10.1016/j.jbi.2015.10.009.
URL https://www.sciencedirect.com/science/article/pii/S1532046415002312

[4] M. Wilkinson, M. Dumontier, I. J. Aalbersberg, G. Appleton, M. Axton, A. Baak, N. Blomberg, J.-W. Boiten, L. O. Bonino da Silva Santos, P. Bourne, J. Bouwman, A. Brookes, T. Clark, M. Crosas, I. Dillo, O. Dumon, S. Edmunds, C. Evelo, R. Finkers, B. Mons, The fair guiding principles for scientific data management and stewardship, Scientific Data 3 (03 2016). doi:10.1038/sdata.2016.18.

[5] C. Martínez Costa, M. Menárguez-Tortosa, J. T. Fernández-Breis, Clinical data interoperability based on archetype transformation, Journal of Biomedical Informatics 44 (5) (2011) 869–880. doi:https://doi.org/10.1016/j.jbi.2011.05.006.
URL https://www.sciencedirect.com/science/article/pii/S1532046411000979

[6] B. Gonçalves, G. Guizzardi, J. G. Pereira Filho, Using an ecg reference ontology for semantic interoperability of ecg data, Journal of Biomedical Informatics 44 (1) (2011) 126–136, ontologies for Clinical and Translational Research. doi:https://doi.org/10.1016/j.jbi.2010.08.007.
URL https://www.sciencedirect.com/science/article/pii/S1532046410001188

[7] I. Berges, J. Bermudez, A. Illarramendi, Toward semantic interoperability of electronic health records, IEEE Transactions on Information Technology in Biomedicine 16 (3) (2012) 424–431. doi:10.1109/TITB.2011.2180917.

[8] N. F. Noy, D. L. McGuinness, Ontology development 101: A guide to creating your first ontology, Tech. rep. (march 2001).
URL http://www.ksl.stanford.edu/people/dlm/papers/ontology-tutorial-noy-mcguinness-abstract.html

[9] M. Fernandez-Lopez, A. Gomez-Perez, N. Juristo, Methontology: from ontological art towards ontological engineering, in: Proceedings of the AAAI97 Spring Symposium, Stanford, USA, 1997, pp. 33–40.

[10] B. Dutta, M. Debellis, Codo: An ontology for collection and analysis of covid-19 data, 2020. doi:10.5220/0010112500760085.

[11] A. Dimou, M. V. Sande, J. Slepicka, P. Szekely, E. Mannens, C. Knoblock, R. V. d. Walle, Mapping hierarchical sources into rdf using the rml mapping language, in: 2014 IEEE International Conference on Semantic Computing, 2014, pp. 151–158. doi:10.1109/ICSC.2014.25.