UTPL

Universidad Técnica Particular de Loja

Sistemas Informáticos y Computación

Integrantes:

- Kevin José Quito Medina
- Freddy Stalin Villavicencio Espinoza

Asignatura: Sistemas Basados en Conocimiento

Docente: Ing. Janneth Chicaiza

Tabla resumen de datos recolectados: A continuación, se presenta el número de instancias que se generarán a partir de los datos recolectados para cada clase del modelo ontológico creado.

Clases del modelo ontológico	Número de instancias a generar
dbo: Place	500 aproximadamente
dbo: Continent	1
dbo: Country	54
dbo: Province	250 aproximadamente
dcat: Dataset	57
dcat:Catalog	2
foaf: Organization	1
newOnto: CaseCovid	75000 aproximadamente
sio: Patient	75000 aproximadamente
newOnto: Medical_Information	75000 aproximadamente
newOnto: Confirmed_cases	3388
newOnto: Death_cases	1276
newOnto: Recovered_cases	2263

Además, cabe mencionar que nosotros no vamos a utilizar todas las clases del modelo, puesto que consideramos que es innecesario porque no tenemos los datos relacionados con dichas clases como por ejemplo la clase de *newOnto:ContainmentMeasures.*

Por otra parte, reconsideramos algunas cuestiones del anterior modelo para la creación y la configuración del repositorio de datos que en nuestro caso es MySQL, en la figura 1, se presenta el modelo final que se utilizará para los siguientes pasos del actual proyecto.

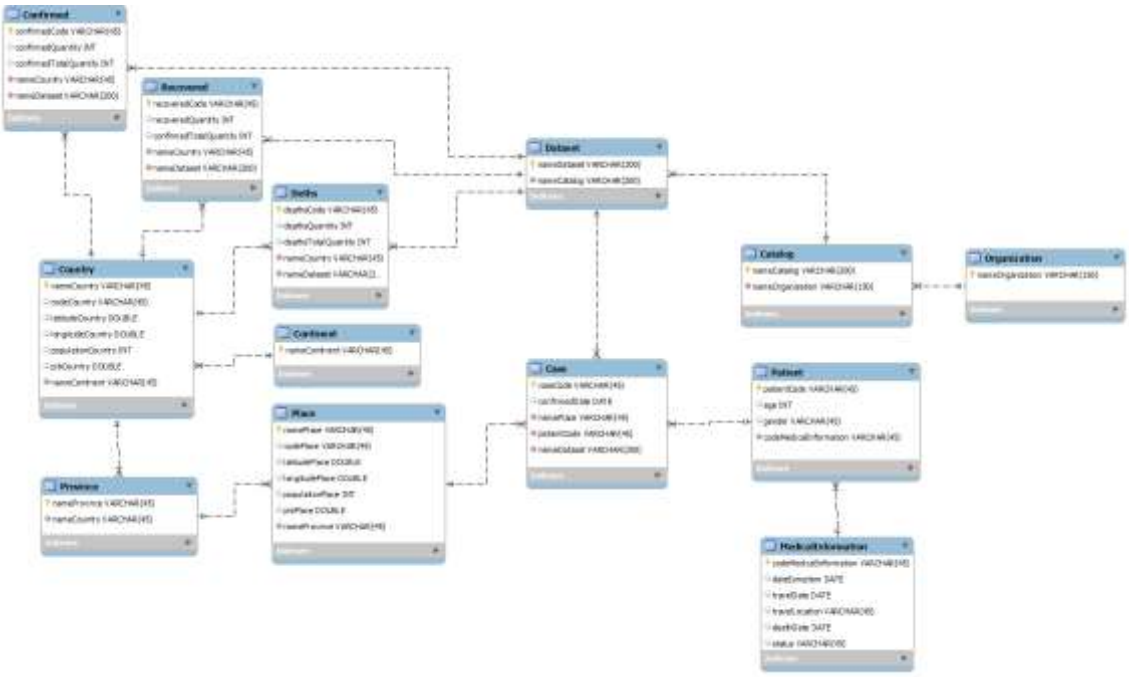


Figura 1. Nuevo modelo de base de datos para la adaptación del modelo ontológico

Preprocesamiento de datos: Indicar qué tareas de limpieza o transformación de datos se realizaron antes de generar RDF.

Inicialmente encontramos 4 datasets sobre el tema relacionado al COVID-19 en el continente africano, más tarde se analizaron y se observaron que dos de ellos compartían información desde algunas fuentes similares por ello decidimos trabajar con 2 datasets que tienen características y atributos dentro del modelo ontológico.

Antes de la generación de datos RDF, se aplicó un proceso de selección y limpieza de los datos de los 2 datasets elegidos. En la fase de recolección de datos recuperamos diferentes datasets generalmente en formato *csv.* los cuales en la gran mayoría se necesitaba realizar una clasificación intensiva para insertarlos en el modelo de base de datos generado. Durante la clasificación y ordenamiento de los datos se fueron adaptando los diferentes atributos al tipo de dato que corresponde para facilitar la posterior utilización y para generar datos RDF mediante Jena.

Por otra parte, actualmente estamos en el proceso de obtención de los datos de ciudades y provincias del continente africado, principalmente para poder realizar un esquema de datos bastante estable con el fin de que se pueda reutilizar para las relaciones existentes entre los diferentes conceptos de las entidades establecidas.

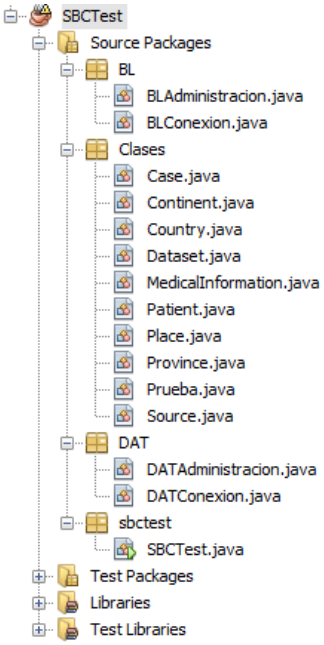
Finalmente se realizó una limpieza de datos y separación por entidades para la posterior carga en la base MySQL como se presenta a continuación.

Figura 2. Limpieza de datos en Excel

idCountry	name	countryCode	latitude	longitude	populationTotal	pib	Continent_idContinent
1	Angola	AO	-11.202692	17.8738861	32886268	6800	1
2	Argelia	DZ	28.033886	1.659626	43851043	15200	1
3	Benin	BJ	9.3076897	2.315834	12123198	2300	1
4	Botsuana	BW	-22.328474	24.684866	2351625	17000	1
5	Burkina Faso	BF	12.2383327	-1.5615931	20903278	1900	1
6	Burundi	BI	-3.3730559	29.9188862	11890781	700	1
7	Cabo Verde	CV	16.0020828	-24.0131969	555988	7000	1
8	Camerún	CM	7.3697219	12.354722	26545864	3700	1
9	Chad	TD	15.4541664	18.7322063	16425859	2300	1
10	Comoras	KM	-11.875001	43.8722191	869595	1600	1
11	Costa de Marfil	CI	7.539989	-5.54708	26378275	3900	1
12	Egipto	EG	26.8205528	30.8024979	102334403	12700	1
13	Eritrea	ER	15.1793842	39.7823334	3546427	1600	1
14	Etiopía	ET	9.1450005	40.4896736	114963583	2200	1
15	Gabón	GA	-0.803689	11.6094437	2225728	18100	1
16	Gambia	GM	13.443182	-15.3101387	2416664	2600	1

Figura 3. Datos cargados en MySQL

Transformación de datos: Indicar la lógica del motor de transformación de datos basado en Jena e indicar porcentaje de avance respecto del total de datos a convertir. En este punto se puede indicar algún esquema que resuma los métodos y demás objetos que se codificarían.



Para la transformación de los datos hemos aplicado una lógica y arquitectura en 3 capas, en la cual se encuentra inicialmente la ***capa de datos*** que se encarga de gestionar todo lo relativo a la base de datos y a la creación, edición y borrado de datos de ésta.

Luego se realiza la conexión con la siguiente capa que es la ***capa de negocio*** en donde se gestiona la lógica de la aplicación. Es donde se dice que se hace con los datos y esta capa estará conectada con la capa de persistencia para poder realizar sus funciones.

Y finalmente en la **capa de presentación** contamos con una clase en donde realizamos en base a las consultas de base de datos la transformación de estos a formato RDF con el framework *JENA*.

En la imagen presentada anteriormente en el paquete *Clases*, se encuentran varias clases .java creadas en base a clase de nuestra ontología sobre datos *COVID-19*.

A continuación, se realizará la explicación breve del proceso que realizamos para llegar a la transformación de datos.

```
public Connection getConnection () throws ClassNotFoundException, SQLException{
    String driver="com.mysql.jdbc.Driver";
    String user="uoarrkef3lziqgp2";
    String password="qeI2HMecZjOTi7ojrHRv";
    String url="jdbc:mysql://uoarrkef3lziqgp2:qeI2HMecZjOTi7ojrHRv@bnyufdebhe-mysql.services.clever-cloud.com:20476
    Class.forName(driver) ;//Diver jdbc para trabajar con acceso
    con =DriverManager.getConnection(url,user,password);
    return con;//retorna la cioneccion url+ruta bd//retorna la cioneccion url+ruta bd
}
```

Figura 4. Conexión a la base de datos de MySQL

Primeramente, realizamos la conexión a la base de datos llamada ***datacovid*** como se muestra en la figura 4.

Luego realizamos una consulta multi tabla para obtener todos los datos relacionados a un caso de *COVID-19* este proceso se muestra en la figura 5.

```
public ResultSet ConsultarCaso() throws ClassNotFoundException, SQLException{
    PreparedStatement pst = c.AbrirConexion().prepareStatement("SELECT * FROM casecovid cc, continent cont,
    ResultSet rs = pst.executeQuery();//recuper un un ResultSet y envio la varible a executeQuery
    return rs;
}
```

Figura 5. Consulta a la base de datos

Finalmente, como se muestra en la figura 6, procesamos el resultado de la consulta anterior instanciando los objetos de las clases .java para poder crear un objeto de un caso de covid y almacenarlo en un arraylist del mismo tipo para posteriormente poder manipularlo y crear tripletas RDF en base al mismo.

```
public ArrayList<Case> consultarCaso() throws SQLException, ClassNotFoundException{
    ArrayList<Case> lstcases = new ArrayList<Case>();
    ArrayList<Source> lstsource = new ArrayList<Source>();
    ResultSet rs = mp.ConsultarCaso();
    while(rs.next()){

        //////Continent//////////
        int idContinent = rs.getInt("idContinent");
        String nameContinet = rs.getString("name");
        Continent continent = new Continent(idContinent, nameContinet);
        //////Country/////////////
        int idCountry = rs.getInt("idCountry");
        String nameCountry = rs.getString("name");
        String codeCountry = rs.getString("countryCode");
        double latitudeCountry = rs.getDouble("latitude");
        double longitudeCountry = rs.getDouble("longitude");
        int populationCountry = rs.getInt("populationTotal");
        double pibCountry = rs.getDouble("pib");
        Country country = new Country(idCountry, nameCountry, codeCountry, latitudeCountry,
        //////Province///////////////
        int idProvince = rs.getInt("idProvince");
        String nameProvince = rs.getString("nombre");
        Province province = new Province(idProvince, nameProvince, country);
        ///////Place//////////////////////
        int idPlace = rs.getInt("idPlace");
        String namePlace = rs.getString("name");
```

Figura 6. Procesamiento de consulta de base de datos

En el arraylist almacenamos temporalmente el resultado de la consulta de la base de datos anteriormente mencionada.

```
ArrayList<Case> lstcases = new ArrayList<Case>();
lstcases = manejador.consultarCaso();
```

Para creación de las diferentes propiedades que corresponden al modelo ontológico definimos los prefijos necesarios como se muestra en la figura 7.

```
//Set prefix for the URI base (data)
String dataPrefix = "http://utpl.edu.ec/lod/dataCOVID/";
model.setNsPrefix("data", dataPrefix);
//Vocab and models present in JENA
//SCHEMA
String schema = "http://schema.org/";
model.setNsPrefix("schema", schema);
Model schemaModel = ModelFactory.createDefaultModel();
//Dbpedia Ontology- DBO
String dbo = "http://dbpedia.org/ontology/";
model.setNsPrefix("dbo", dbo);
Model dboModel = ModelFactory.createDefaultModel();
//Geonames - gn
String gn = "http://www.geonames.org/ontology#";
model.setNsPrefix("gn", gn);
Model gnModel = ModelFactory.createDefaultModel();
//Dublincore - DBR
String dc = "http://purl.org/dc/elements/1.1/";
model.setNsPrefix("dc", dc);
Model dcModel = ModelFactory.createDefaultModel();
//DCat - dcat
String dcat = "http://www.w3.org/ns/dcat#";
model.setNsPrefix("dcat", dcat);
Model dcatModel = ModelFactory.createDefaultModel();
//Prov - prov
String prov = "http://www.w3.org/ns/prov#";
model.setNsPrefix("prov", prov);
Model provModel = ModelFactory.createDefaultModel();
//SIO - sio
String sio = "http://semanticscience.org/resource/";
model.setNsPrefix("sio", sio);
Model sioModel = ModelFactory.createDefaultModel();
//newOnto - newOnto
String newOnto = "http://utpl.edu.ec/lod/dataCOVID/ontology/";
model.setNsPrefix("newOnto", newOnto);
Model newOntoModel = ModelFactory.createDefaultModel();
```

Figura 7. Definición de prefijos

A continuación, se presenta la reutilización los métodos proporcionados por el docente para la creación de datos RDF con los datos del arraylist como se muestra a continuación en la figura 8.

Figura 8. ArrayList para creación de datos RDF

Al ejecutar el código anterior se generan las tripletas RDF de cada caso de COVID-19 con sus respectivos datos y relaciones como se muestra a continuación en la figura 9.

```
MODELO RDF------
<rdf:RDF
    xmlns:dc="http://purl.org/dc/elements/1.1/"
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:ov="http://open.vocab.org/terms/"
    xmlns:prov="http://www.w3.org/ns/prov#"
    xmlns:newOnto="http://utpl.edu.ec/lod/dataCOVID/ontology/"
    xmlns:dbo="http://dbpedia.org/ontology/"
    xmlns:schema="http://schema.org/"
    xmlns:data="http://utpl.edu.ec/lod/dataCOVID/"
    xmlns:sio="http://semanticscience.org/resource/"
    xmlns:dcat="http://www.w3.org/ns/dcat#"
    xmlns:gn="http://www.geonames.org/ontology#"
    xmlns:j.0="http://xmlns.com/foaf/0.1/">
  <newOnto:CaseCovid rdf:about="http://utpl.edu.ec/lod/dataCOVID/Case-028376">
    <gn:locatedIn>
      <dbo:Place rdf:about="http://utpl.edu.ec/lod/dataCOVID/Bilqas_city">
        <dbo:grossDomesticProductNominalPerCapita>0.0</dbo:grossDomesticProductNominalPerCapita>
        <dbo:populationTotal>0</dbo:populationTotal>
        <schema:longitude>0.0</schema:longitude>
        <schema:latitude>0.0</schema:latitude>
        <gn:countryCode></gn:countryCode>
        <dbo:name>Bilqas</dbo:name>
      </dbo:Place>
    </gn:locatedIn>
    <prov:wasDerivedFrom>
      <dcat:Dataset rdf:about="http://utpl.edu.ec/lod/dataCOVID/line-list-egypt">
        <dcat:dateModification>2020-05-30</dcat:dateModification>
        <dc:dateSubmited>2020-03-01</dc:dateSubmited>
        <dcat:downloadURL>https://github.com/defni/covidl9africa/blob/master/data/line_lists/line-list-egypt.csv</dcat:downloadURL>
        <dbo:description></dbo:description>
        <dbo:name>line-list-egypt</dbo:name>
      </dcat:Dataset>
    </prov:wasDerivedFrom>
    <newOnto:hasData>
      <sio:Patient rdf:about="http://utpl.edu.ec/lod/dataCOVID/Patient-028376">
        <newOnto:hasData>
          <newOnto:Medical_Information rdf:about="http://utpl.edu.ec/lod/dataCOVID/">
            <newOnto:travel_history_location>Cameroon</newOnto:travel_history_location>
            <newOnto:travel_history_location>France, Italy</newOnto:travel_history_location>
            <newOnto:travel_history_location></newOnto:travel_history_location>
            <newOnto:travel_history_date>2020-03-03</newOnto:travel_history_date>
            <newOnto:travel_history_date>2020-03-04</newOnto:travel_history_date>
            <newOnto:travel_history_date>2020-02-17</newOnto:travel_history_date>
            <newOnto:travel_history_location>Spain</newOnto:travel_history_location>
            <newOnto:travel_history_date>null</newOnto:travel_history_date>
            <newOnto:date_first_symptom>null</newOnto:date_first_symptom>
            <newOnto:date_first_symptom>2020-03-15</newOnto:date_first_symptom>
            <newOnto:travel_history_location>France</newOnto:travel_history_location>
            <newOnto:travel_history_date>2020-03-01</newOnto:travel_history_date>
            <dbo:currentStatus></dbo:currentStatus>
            <newOnto:travel_history_location>Italy</newOnto:travel_history_location>
            <newOnto:travel_history_date>2020-03-14</newOnto:travel_history_date>
```

Figura 9. Resultado de la ejecución del código para la generación de datos RDF

Como último punto mencionamos que tenemos un avance aproximado del 75% sobre la generación de datos procesados en *JENA* y en cuanto a la limpieza y clasificación de los datos tenemos un aproximado 90% puesto que se solicitaron más cambios al modelo ontológico.