DUY TÂN UNIVERSITY
International School

CAPSTONE PROJECT 2

CMU-SE-451 / CMU-IS-451 / CMU-CS-451

ARCHITECTURE DOCUMENT

Version 2.0

Date: 09 - May - 2021

EXPERT-DRIVEN SMART DASHBOARD APPLICATION

Submitted by

Vo Van Hoa
Pham Van Tin
Ky Huu Dong
Tran Thi Thanh Kieu

Approved by

Capstone Project 2 - Mentor:

Name	Signature	Date
Binh, Thanh Nguyen		26 - May - 2020

PROJECT INFORMATION			
Project Acronym	EDSDA		
Project Title	Expert-Driven Smart Dashboard Application		
Project Web URL	https://sda-research.ml/		
Start Date	01 - Mar - 2021		
End Date:	02 - Jun - 2021		
Lead Institution	International School, Duy Tan University		
Project Mentor	Ph.D Binh, Thanh Nguyen		
Scrum Master	Hoa, Vo	hoavo.dng@gmail.com	0935.193.182
Team Members	Tin, Pham Van	tinphamvan123@gmail.com	0932.535.175
	Dong, Ky Huu	kyhuudong@gmail.com	0898.246.980
	Kieu, Tran Thi Thanh	thanhkieutran391@gmail.com	0358.583.251

DOCUMENT INFORMATION			
Document Title	Architecture Document		
Author(s)	Team C2SE.06		
Role	[EDSDA] Architecture_Document_v2.1		
Date	01 - Mar - 2021	Filename	[EDSDA] 004 Architecture Document
URL	https://github.com/sdateamdtu2020/SDA-v2.0		
Access	Project and CMU Program		

REVISION HISTORY

Version	Person(s)	Date	Description	Approval
Draft	Hoa, Vo	01 - Mar - 2020	Initiate document	x
2.0	All members	09 - May- 2020	Finish content of document	x
2.1	All members	20 - May - 2020	Update content of low level architecture	x

TABLE OF CONTENTS

PROJECT INFORMATION 1
REVISION HISTORY 2
TABLE OF CONTENTS 3
1. INTRODUCTION 5
1.1. PURPOSE 5
1.2. DEFINITIONS, ACRONYMS AND ABBREVIATIONS 5
1.3. DOCUMENTS REFERENCES 5
2. PROBLEM STATEMENT 6
2.1. PROJECT OVERVIEW 6
2.2. BUSINESS DRIVERS 6
2.3. PROJECT GOAL 7
3. ARCHITECTURE DRIVERS 7
3.1. HIGH-LEVEL REQUIREMENTS 7
3.2. SYSTEM CONTEXT 8
3.3. QUALITY ATTRIBUTES 10
4. CONSTRAINTS 12
4.1. BUSINESS CONSTRAINTS 12
4.2. TECHNICAL CONSTRAINTS 12
5. HIGH-LEVEL ARCHITECTURE 13
5.1. COMPONENT AND CONNECTOR VIEW (C&C VIEW) 13
5.2. MODULE VIEW 17
5.2.1. Module View based on Web Application 17
5.2.2. Module View based on Server 21
5.3. ALLOCATION VIEW 23
6. LOW-LEVEL ARCHITECTURE 24
6.1. DATABASE DESIGN 24
6.1.1. Data warehouse architecture 24
6.1.2. Star schema, Fact Tables and Dimension Tables 25
6.2. DATA LAKE 25
6.3. ETL (EXACT, TRANSFORM, LOAD) 26
6.3.1. Base workflow 26
6.3.2. Import workflow 27
6.3.3. Data pipeline workflow 28
6.4. RDF DATA CUBES 29

6.4.1. RDF data cubes design 29
6.4.2. RDF data cubes architecture 29
6.5. RDF DATA CUBES AUTO GENERATOR 32
6.5.1. Traditional R2RML Methods 32
6.5.2. Automatic RDF Data Cubes Generator 33
6.6. APPLICATION APIS 34
7. REFERENCES 34

1. INTRODUCTION

1.1. PURPOSE

The purpose of the Architecture document is to:

- Define the architecture needs and technology in detail.
- Provide solutions for business needs.
- Provide overview about resources, schedule, solution and budget for the project.

The architecture merely introduces the project to the student development teams, and provides the up-front information necessary for the team to develop a specification.

1.2. DEFINITIONS, ACRONYMS AND ABBREVIATIONS

Acronyms	Definitions
EDSDA	Expert-driven Smart Dashboard Application
GUI	Graphical User Interface
SDK	Software Development Kit
ES	ECMAScript language

1.3. DOCUMENTS REFERENCES

No.	Reference
1	Product Backlog Document for EDSDA
2	Project Plan Document for EDSDA

2. PROBLEM STATEMENT

2.1. PROJECT OVERVIEW

Our environment is always changing. However, at the current rate of urbanization and industrialization, outside of the natural factors, the change of environment is mainly due to human factors. Emissions, population explosion, industrial solid waste, ... are the main causes leading to negative effects on the global environment. To address this at a holistic level, data analysis and aggregation are the first important tasks to be done.

However, analyzing and aggregating data from many different sources takes a lot of effort and money. To solve this problem, based on our knowledge of big data systems, we have built an intelligent data processing system that can be run on a website-platform with an intuitive and easy-to-use dashboard. This system is a prospective and useful tool for environmental experts and policymakers in Vietnam in particular, and worldwide in general. It will collect, analyze and synthesize data about all the factors that can affect the environment, thereby helping users to come up with quick and accurate solutions to solve problems related to the environment.

In addition, data sources that open the environment in a particular aspect such as area emissions, soil erosion, etc. are often scarce and stored only in the user's local computer. Analytical tools of these types of data often require expertise using quite advanced technology, thus limiting interoperability with data stored in individuals. With this in mind, we built a subsystem where users could bring their data to EDSDA and analyze, aggregate, and visualize their data directly on the dashboard along with a wide range of system visualization tools.

2.2. BUSINESS DRIVERS

Business problem:

Our environment is always changing. However, at the current rate of urbanization and industrialization, outside of the natural factors, the change of environment is mainly due to human factors. Emissions, population explosion, industrial solid waste, ... are the main causes leading to negative effects on the global environment. To address this at a holistic level, data analysis and aggregation are the first important tasks to be done.

Business need:

Expert-driven Smart Dashboard Application have specific uses :

- Help users to come up with quick and accurate solutions to solve problems that are related to the environment.
- Help users represent data in many types of charts, diagrams and maps, thereby providing an overview to solve the problem.
- Help users can get the information and insights they need at a glance.
- Giving business predictions based on the chart, diagrams, and maps.

All the things above are based on the functionality of the Expert-driven Smart Dashboard Application system. EDSDA fully meets these requirements. Therefore, the development of EDSDA is very necessary and meaningful.

2.3. PROJECT GOAL

Expert-Driven Smart Dashboard - means a dashboard that is convenient for users to analyze and review data. It will include several datasets about real-time information of the environment, measurement data of air pollutants... EDSDA will connect and analyze data from multiple sources in ways you've never imagined. Then reveal the insights you've been missing...in just a matter of minutes.

With smart suggestions and an intuitive visual interface, SDA makes it easy for any user to combine data and discover hidden insights in one place...without the usual scripting, coding, and IT hand-holding. With this dashboard, individuals or any subjects can take advantage of environmental data to be able to decide the best relevant policies.

Not only for the environmental expert, but EDSDA also helps the user directly interact with their own data source by importing, automatically cleaning their data, and enabling the user to do everything with their own data in the easiest way.

3. ARCHITECTURE DRIVERS

3.1. HIGH-LEVEL REQUIREMENTS

(Refer to the Product Backlog document for EDSDA)

3.2. SYSTEM CONTEXT

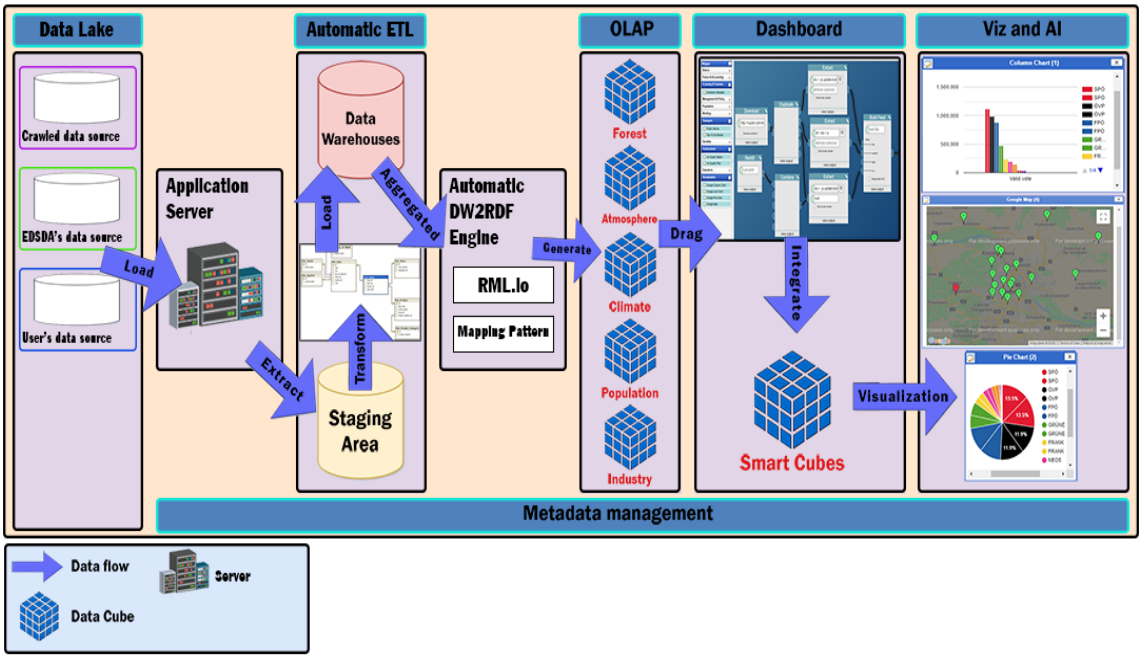

Figure 1: EDSDA Context Diagram

1. **Data Lake:**
   - Collect from environment open data platforms of the governments and NGO organizations.
   - User imported data sources.
   - Use web crawling techniques to crawl data from related environment websites.
   - Data Format : CSV, JSON, XML.
2. **Application Server:**
   - Check for the changes of the data lake when the user imports new data sources.
   - Handle Data Lake Exceptions if any errors occur.
   - Trigger the Data Lake to Staging Area loading process.
3. **Automatic ETL (Extract, Transform, Load)**

   3.1. Extract:

   In this step, we extract structured data after restructuring the raw data from the data lake. Mainly in this step, we focus on exacting data correctly from the structured data from the data lake.

   3.2. Transform:

   Based on the business's constraints we have to transform the data to be compatible with our process to handle the data for displaying on the dashboard. Getting exacted data, transforming the data, and ready to load into the DW.

3.3. Load:

Loading data into the target data warehouse database is the last step of the ETL process. We load all the data we have transformed into the staging areas of DW and based on the data we have loaded into staging areas we used to load the data from the staging areas to dimension tables, and fact tables.

3.4. Perform automatic ETL:

Performing automatic ETL using tools for automating exacting data, transforming data, and loading data to DW. We use the available platform for scheduling to perform ETL at a specific time in a day. In addition, for performing ETL automatically we also build a data pipeline to make the data flow as we wanted.

4. **Automatic DW2RDF Engine:**
   - Using RML.io with our custom features, it can generate RDF Data cubes automatically when users import new data sources to the data warehouse.
   - Suitable with all data sources with a general mapping method.
5. **OLAP**

   In this step, we store the data of the data warehouse as OLAP cubes. And then, for better query performance, data binding, and scalability, in addition to information transparency, we will automate converting OLAP cubes into RDF Data Cubes.
6. **Dashboard**

   In this step, the user can drag any data cubes that appear as items on the sidebar and drop them onto the main content board, then connect between them and use the operator such as statistics merge, geo merge to build a new data cube that matches the user requirement.
7. **Viz & AI**

   This step will perform the data cube which was created by the user with the form they want. It can be a map, a column chart, a line chart, or a pie chart.

3.3. QUALITY ATTRIBUTES

ID	QA01
Quality Attributes	Performance
Stimulus	Uses the statistics merge operator for merging multiple data cubes.
Source(s) of stimulus	User
Artifacts	System
Environment	Normal mode
System response	The SDA returns a new data cubes as fast as possible
Response measure(s)	Within 3 seconds

Table 3.3.1: Quality Attributes: Performance

ID	QA02
Quality Attributes	Availability
Stimulus	The power is off while server is running
Source(s) of stimulus	Power
Artifact	During peak usage load
Environment	Hardware and software
System response	System will use the cloud server to save the work so we don't need to worries about power incident occurred
Response measure(s)	All work always can be saved

Table 3.3.2: Quality Attributes: Availability

ID	QA03
Quality Attributes	Availability
Stimulus	Can't get a specific data cubes when select it from widget
Source(s) of stimulus	User
Artifact	System
Environment	Normal mode
System response	System will log the fault immediately
Response measure(s)	Within immediately

Table 3.3.2: Quality Attributes: Availability

ID	QA03
Quality Attributes	Portability and compatibility
Stimulus	Open the browser and access to http://sda-research.ml/
Source(s) of stimulus	User
Artifact	System
Environment	Normal mode
System response	The system can run on any web browser.
Response measure(s)	Within immediately

Table 3.3.2: Quality Attributes: Portability and compatibility

ID	QA03
Quality Attributes	Security
Stimulus	User access to EDSDA and import a file
Source(s) of stimulus	User
Artifact	System
Environment	Normal mode
System response	System will not track any individual information about the user
Response measure(s)	Within immediately

Table 3.3.2: Quality Attributes: Availability

4. CONSTRAINTS

4.1. BUSINESS CONSTRAINTS

- Project will be started on: 01 - Mar - 2021
- Project will be finished on: 02 - June - 2021
- Duration: 13 weeks

4.2. TECHNICAL CONSTRAINTS

Main Programming Language: Javascript, Python.

Data Crawling:

- Programming Language: Python, Javascript.
- Database: PostgreSQL.
- Library: Beautiful Soup 4, Selenium.

Data Warehouses:

- Programming Language: Python.
- Database: PostgreSQL.
- Library: Psycopg2, CSV.

Perform ETL automating:

- Tool for ETL process: Talend.
- Schedule for ETL process flatform: Apache Airflow.

Data Cubes

- Programming Language: Java, SPARQL.
- Tool for converting from Data Warehouse to RDF Data Cubes Storage: RML.io (with customized phase).
- Network Accessing: RDF-REST API.

Server:

- Programming Language: Javascript.
- Framework / Libraries: ExpressJS (NodeJS), enapso-graphdb-client.
- Operating System: Windows, Linux, macOS.
- Deployment Environment: Google Cloud with App Engine and SQL Services.
- Network Accessing: HTTP methods (POST, GET) via RESTful API.

Client:

- Programming language: HTML, CSS, Javascripts.
- Framework/Libraries: React, Redux, Material-UI, React Flow, React Charts,...
- Deployment Environment: Google Firebase Hosting
- Web Browser: Chrome, Firefox, Microsoft Edge, Coccoc
- Network Accessing: World Wide Web (WWW), HTTP methods (POST, GET) via RESTful API with Axios.

5. HIGH-LEVEL ARCHITECTURE

5.1. COMPONENT AND CONNECTOR VIEW (C&C VIEW)

The diagram below shows the overview architecture including components and other related components. We have representations and behaviors for import components in the following sections

References C&C View on attached page.

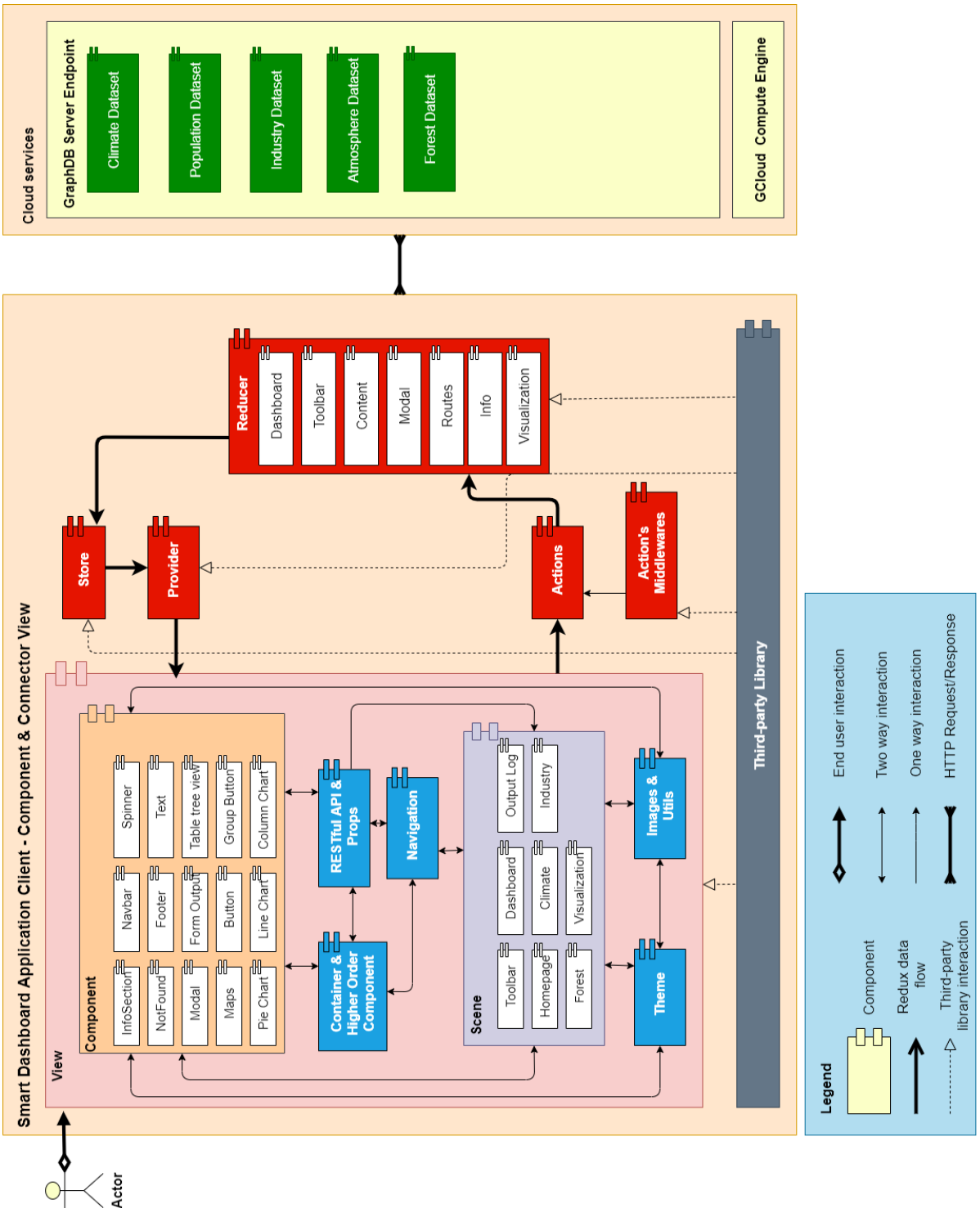

Figure 2: EDSDA C&C View

Prose:

When the end-user opens the application, GUI will appear. Users will interact with the View component. The View component includes some small components like Toolbar, Content, Description and Util, Scene, Navigation, Container and Higher Order component, and some other components (The View component will be detailed in Model View). If the User has any action, View will call the Actions component. Actions will format the requirement and send it to the Reducer. The Reducer will send a state tree to the Store. The Store will replace the old state tree with a new one. The Provider is a component that helps connect Store and View, it will get the new state from Store and send it back to View. Third party libraries will help the application run faster and reduce the code time.

Any time when the application needs to use data, it will call Cloud service. Cloud service includes Graph server endpoint. We place our server on GCloud and implement the application API on it.

Role & Responsibility	Description
View	
Component	Components let you split the app into independent, reusable pieces, and think about each piece in isolation.
Scene	The Scene transform represents the app in UI.
Container & Higher Order Component	Pattern that has proven to be very valuable for several React libraries
Restful API & Props	Provides a data in your API & render in child
Navigation	Provides an easy to use navigation solution,
Theme	Defining a set of styles.
Images & Utils	Storing images or utils library.
Store	The Store is the object that brings all together.
Provider	Make the store available to all container components in the application without passing it explicitly.
Actions	Actions are payloads of information that send data from your application to your store.
Actions's Middlewares	It provides a third-party extension point between dispatching an action, and the moment it reaches the reducer.

Role & Responsibility	Description
Reducer	Actions describe the fact that something happened, but don't specify how the application's state changes in response. This is the job of reducers.
Third-party Library	
GraphDB Server Endpoint	Build and ship our APIs faster and more consistently. Not having to worry about authentication, performance and status monitoring has reduced the time and effort we need to build great APIs
API:	Set of functions and procedures that allow the creation of applications which access the features or data of an operating system, application, or other service.

5.2. MODULE VIEW

5.2.1. Module View based on Web Application

References Module View on attached page.

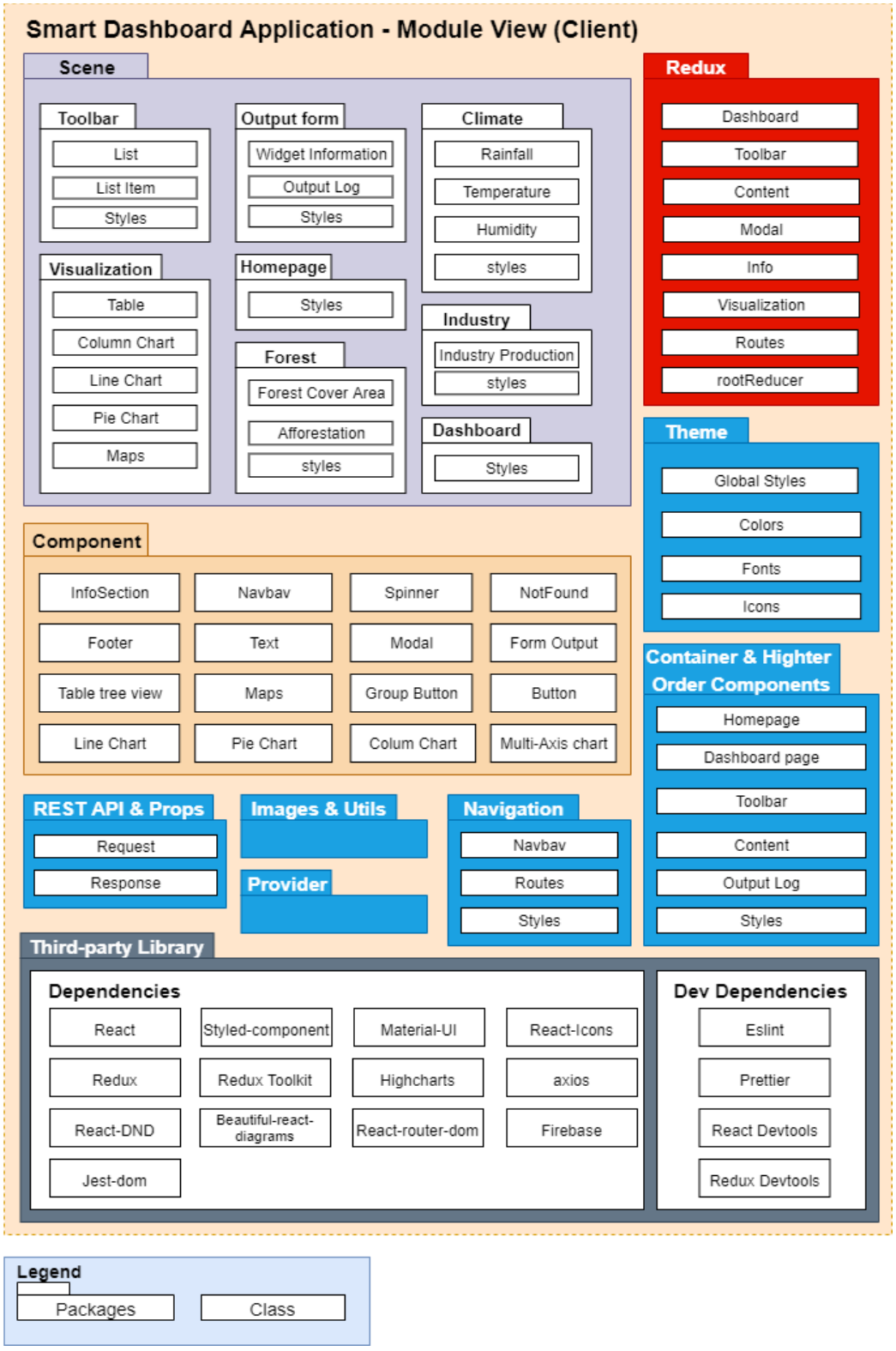

Figure 3: Module View Client

Prose:

The EDSDA client application includes 10 packages that help the app run effectively.

The Scene package includes

Component package which has 16 classes which is often used and we custom it to fit our requirement. We also have Theme package to define the app format. It will define the app color, font, icon, and metric.

Navigation package contains a Navbar, Routes and its Style.

Redux package also has Dashboard, Toolbar, Content, Modal and root Reducer class to manage the state. Our Component and Higher Order Components is an advanced technique in React that is used in reusing components. Higher Order Components are not part of the React API. Specifically, a higher-order component is a function and it takes the argument as a component and returns a new component.

Images and Utils package contain the app image and functions to solve app general problems.

In third-party libraries, we have Dependencies and Devdependencies class.

Beside it, We have Dev Dependencies class like EsLint, Prettier, React Dev Tool and Redux Dev Tool to manage the code and make sure that our code follows the general format.

Finally, when the app wants to use data, it will connect to cloud service. Cloud service includes GraphDB and GCloud service

Role & Responsibility	Description
Styles	Defining styles
Scene	
Toolbar	Display list of datacube & dataset
Output form	Widget Log
User Guide	Help user can use app easily
Homepage	Home page of web
Dashboard	Higher Order Component that wrap toolbar, mainboard, Information Widget
Climate	DataCube
Industry	DataCube
Forest	DataCube
Component	

Role & Responsibility	Description
InfoSection	Section Information
Navbar	Navigation
Spinner	An UI when page is loading
NotFound	An UI when Page when 404 error
Footer	Footer
Form Input	A text field is used for form
Text	A general text is used for all text in this application
Modal	Modal Component is inherited Modal of Material-UI
Button	Button
Group Button	List of button
Custom Node	Widget of each datacube & visualization
Line Chart	Visualization data from data cube in chart
Column Chart	Visualization data from data cube in chart
Pie Chart	Visualization data from data cube in chart
Maps	Visualization data from data cube in maps
Redux	
Dashboard	A Redux store & reducer for handling state of Dashboard
Toolbar	A Redux store & reducer for handling state of Toolbar
Content	A Redux store & reducer for handling state of Content
Modal	A Redux store & reducer for handling state of Modal
Routes	Define routes of scenes for navigation
rootReducer	A combination of all defined reducer & third party reducer
Theme	
Global Styles	Defining styles
Container & Higher Order Components	
Homepage	Responsible for handling homepage, about, contact routes
Dashboard page	Responsible for handling dashboard routes &
Toolbar	Handle list item & drag & drop from Toolbar to Content

Role & Responsibility	Description
Content	Handle widget node & connector in Mainboard
Outputlog	Handle widget information & data log
Rest API & Props	
Request	Used to send request to get data from server
Response	Used to get data when requesting success, such as humidity, temperature,....
Images & Utils	
Provider	
Navigation	
Navbar	A List of buttons such as: New, Help, Example1,...
Routes	Define routes of scenes for navigation
Third-party Library	
Dependencies	A list of dependencies that is required for operating the application
DevDependencies	A list of dependencies that is required for development environment

5.2.2. Module View based on Server

References Module View - Server on attached page.

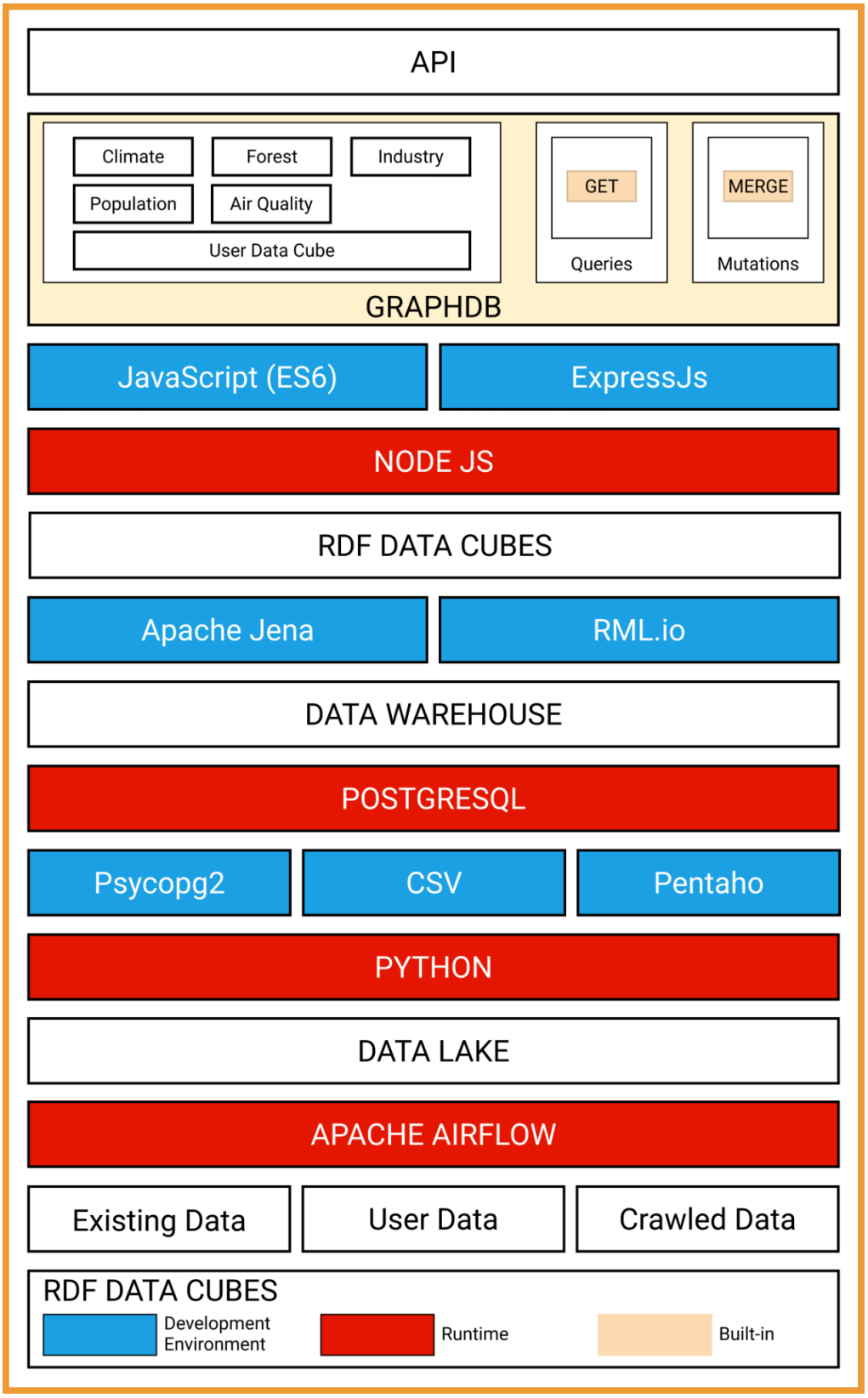

Figure 4: Module View - based on Server

Prose:

Role & Responsibility	Description
API	Execute the responses & requests
GraphDB	RDF Data Storing & Querying platform
Climate	Data of climate data cube (humidity, rainfall, temperature values)
Forest	Data of forest data cube (afforestation area, forest cover value)
Industry	Data of industry data cube
Population	Data of population data cube
Merge	Execute merging 2 or 3 or multi data cubes for creating a new data cube
Get	Get data of data cubes
Javascript ES6	Javascript ES6 Standard
ExpressJS	Library for programming API
NodeJS	Library for programming API
Apache Jena	Library for creating RDF Data Cube API
RML.io	Framework for building automatic Relational Data Warehouse to RDF Generator
Data Warehouse	Place where all the processed data is stored
PostgreSQL	Database system that help building the Data Warehouse
Psycopg2, CSV, Pentaho	Python frameworks that is used to programming auto ETL method
Python	Programming Language for ETL and importing the data sources
Data Lake	Place where storing all the non-processed data sources
Existing Data	Data sources of SDA
Crawled Data	Data sources that continuously updated with the crawler
User's Data	Data sources that are imported from user

5.3. ALLOCATION VIEW

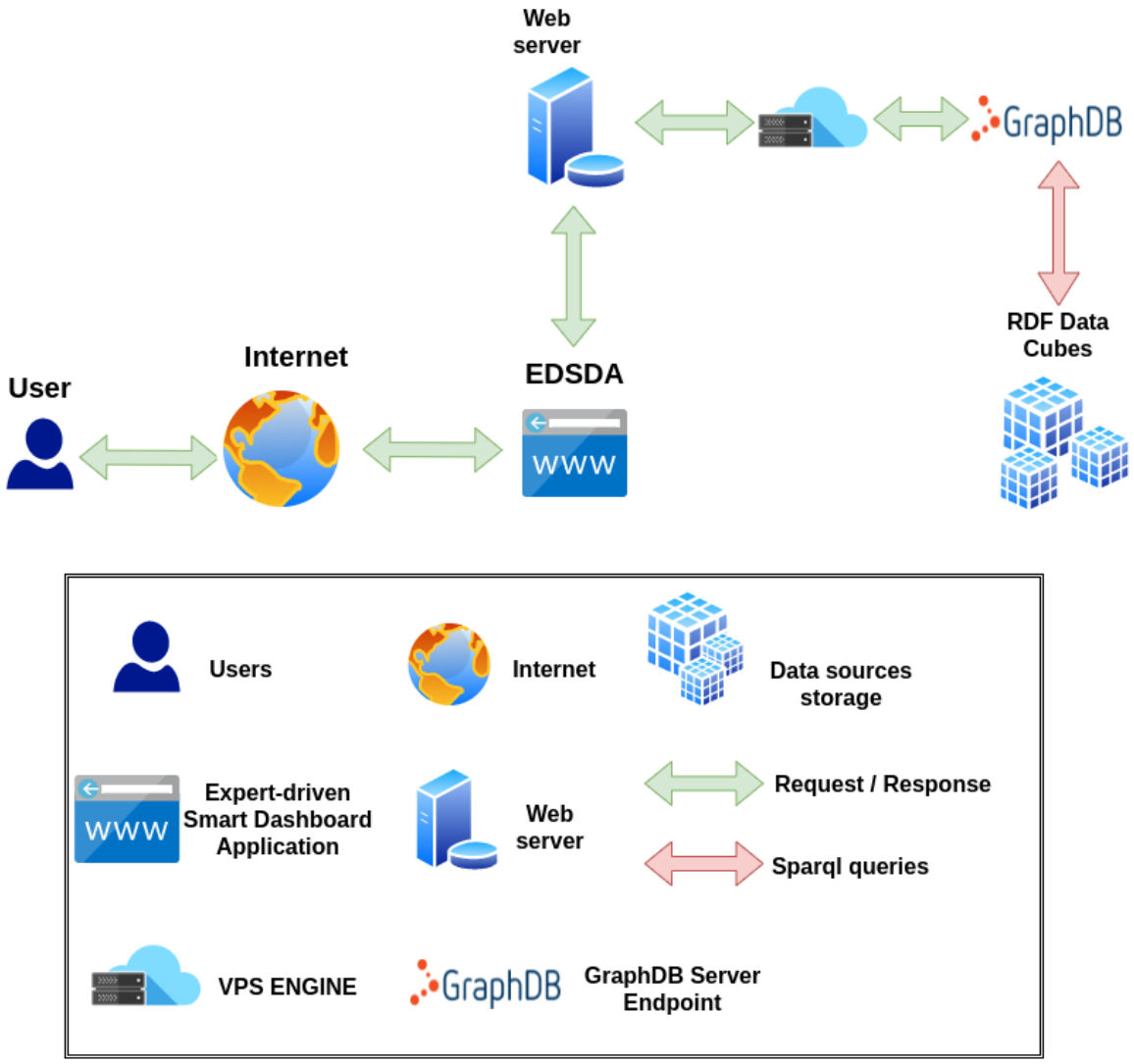

Figure 5: EDSDA Allocation View

Prose:

The user can access our web app via the internet. When they use it, the web app will connect to the GCloud server to get the data via the internet.

Role & Responsibility	Description
Data sources storage	RDF Data Cubes is stored on GraphDB
User	User that interact with EDSDA
Web server	HTTP Server with NodeJS
Expert-driven Smart Dashboard Application	Our application.
VPS Engine	Store and Execute EDSDA's server

Role & Responsibility	Description
GraphDB Server Endpoint	Stage to received and run SPARQL for interacting with the data cubes
Request / Response	Get request metadata from client and response the data to client
SPARQL queries	Query language for execute the method of RDF Data Cubes

6. LOW-LEVEL ARCHITECTURE

6.1. Database Design

6.1.1. Data warehouse architecture

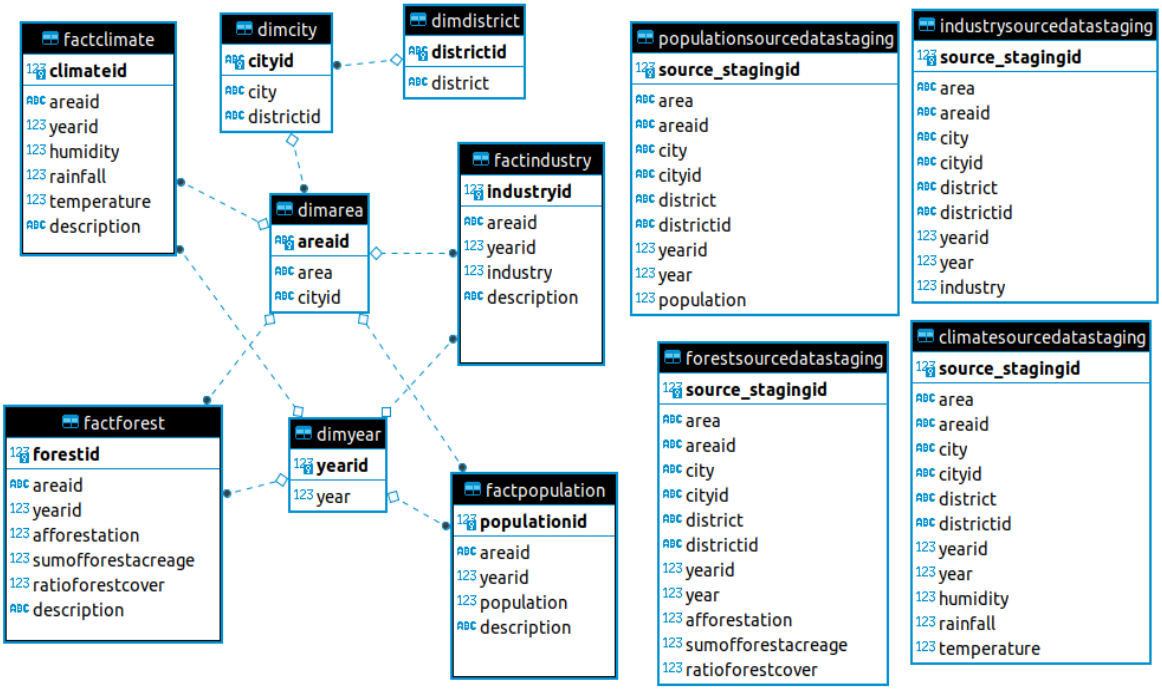

Figure 6: Data warehouse Schema

A data warehouse is subject-oriented as it offers information regarding a theme instead of companies' ongoing operations. These subjects can be sales, marketing, distributions, etc.

A data warehouse never focuses on the ongoing operations. Instead, it put emphasis on modeling and analysis of data for decision making. It also provides a simple and concise

view around the specific subject by excluding data which is not helpful to support the decision process.

6.1.2. Star schema, Galaxy schema, Fact Tables, and Dimension Tables

The star schema architecture is the simplest data warehouse schema. It is called a star schema because the diagram resembles a star, with points radiating from a center. The center of the star consists of a fact table and the points of the star are the dimension tables.

From the star schema data warehouse architecture, relating more dimension tables. It becomes the Constellation Schema (galaxy) with dimensions in this schema separated into separate dimensions based on the various levels of hierarchy. Moreover, it is possible to build this type of schema by splitting the one-star schema into more Star schemes.

A fact table typically has two types of columns: foreign keys to dimension tables id and measures those that contain numeric facts. A fact table can contain fact's data on a detailed or aggregated level.

A dimension is a structure usually composed of one or more hierarchies that categorize data. If a dimension hasn't got hierarchies and levels it is called flat dimension or list. The primary keys of each of the dimension tables are part of the composite primary key of the fact table. Dimensional attributes help to describe the dimensional value. They are normally descriptive, textual values. Dimension tables are generally smaller in size than fact tables.

Typical fact tables store data about sales while dimension tables data about the geographic region (markets, cities), clients, products, times, channels.

The EDSDA database system is designed based on the data warehouse platform. In EDSDA data warehouse, there are three main concepts (table types), those are staging area table, dimensional table, and fact table with star schema:

- Staging area tables: Climate Staging area, Forest Staging area, Industry Staging area, Population Staging area.
- Dimensional tables: dimyear, dimarea, dimcity, dimdistrict.
- Fact tables: factclimate, factforest, factindustry, factpopulation.

6.2. Data lake

Data lake is a centralized repository that allows storing structured and unstructured data at any scale. Data is collected from multiple sources, and moved into the data lake in its original format. This process allows us to scale to data of any size, while saving time of defining data structures, schema, and transformations.

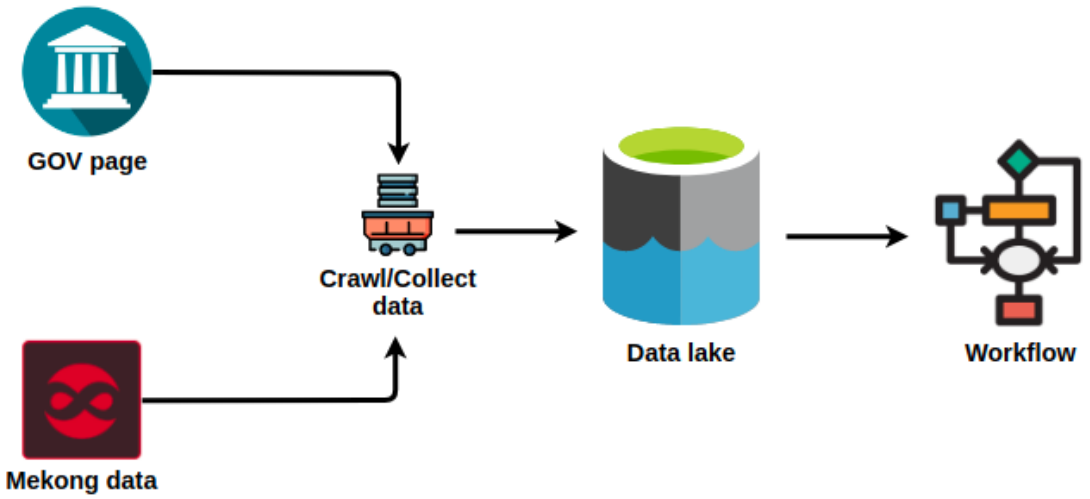

Figure 7: Data lake storing data.

Collecting and crawling data from the government page, Mekong site and starting the loading process to load data into the data warehouse with the design workflows.

6.3. ETL(Extract, Transform, Load) workflows

6.3.1. Base workflow

The first workflow created by the available data sources found.

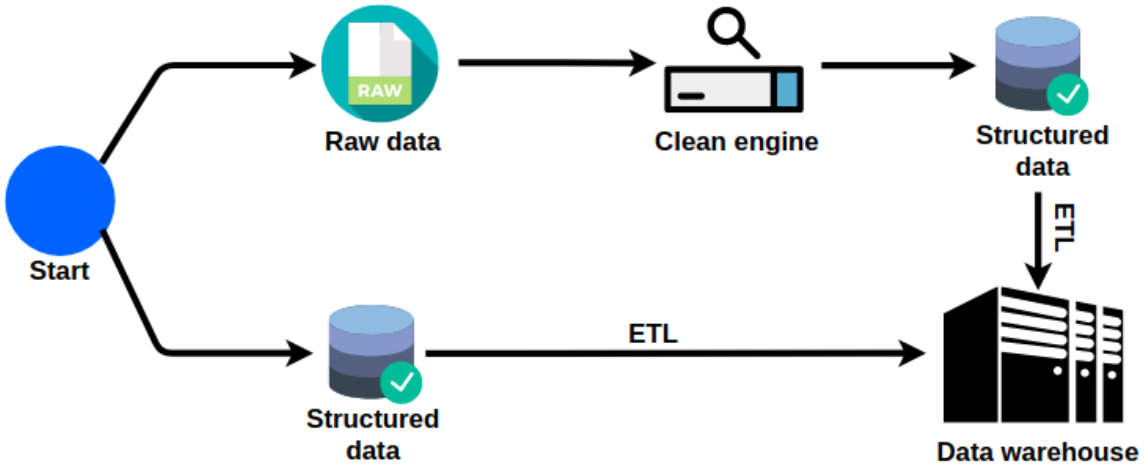

Figure 8: Workflow for the base database.

The purpose of this workflow is to create a base data warehouse, some of the raw data is really messy and can not be used for loading data to the Data warehouse so that having a process makes things easier.

- The collected raw data will get through the clean engine, for each raw data source will have a specific clean engine to handle with a specific raw data. After

getting through the clean engine the raw data becomes structured ready to load into the Data warehouse.

- Clean engine contains some of the ETL works, removing the unnecessary parts of the raw data.
- Nothing much to do with structured data. It is ready to load data into the data warehouse.
- Loading structured data into the data warehouse, before this process happens we can perform our business constraints, removing unnecessary parts, adding logic, and etc ... before we load it into the data warehouse.

6.3.2. Import workflow

The data imported from the user interface gets through the process to ETL into the data warehouse.

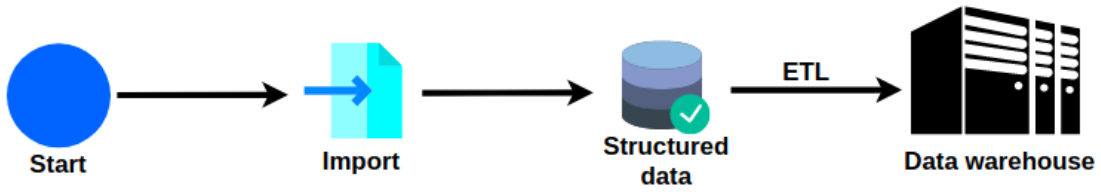

Figure 9: Workflow for importing data.

The data imported must be structured data, constraining the input values really necessary because the input data are a variety of types.

Handling the imported input, letting the user select their own dimensions and measures. After the process, using HTTP requests to get data from the client and ready to import the data into the data warehouse.

6.3.3. Data pipeline workflow

Automating perform tasks, scheduling in the specific time to perform the data pipeline.

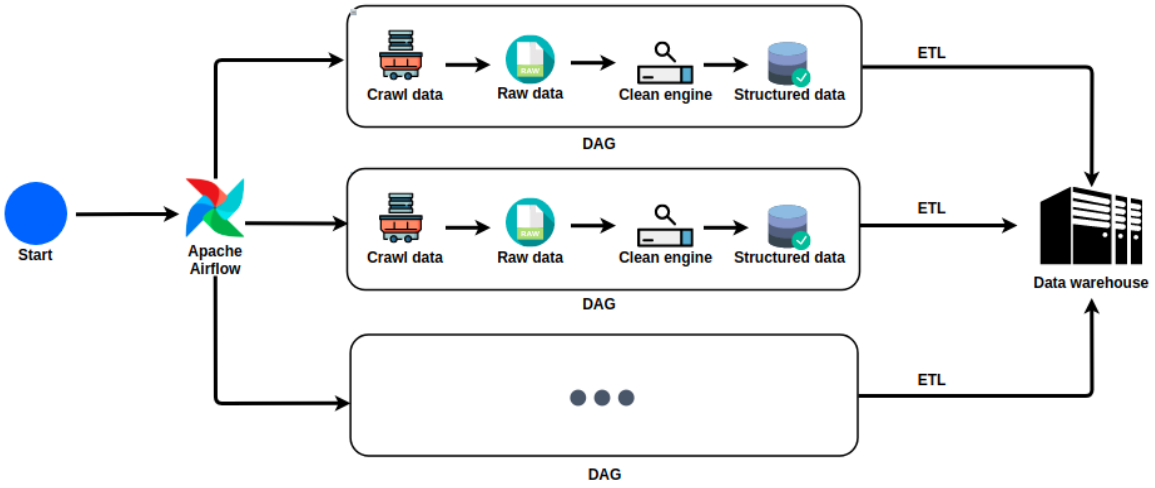

Figure 10: Data pipeline workflow with Apache Airflow managing DAGS.

Apache airflow allows us to schedule, automate performing tasks by using DAG.

DAG(Directed Acyclic Graph) shows us the data pipeline we are running, using dag easily manages the workflow by showing the error, exact phase in the flow is wrong, and logs ...etc.

Each DAG has its own page to crawl and a clean engine to clean. Basically, Each page has its own content and each content has its own structure because of that we have to design a whole new crawl script and clean script to perform with the new DAG.

Scheduling the time to perform specific DAG to get the workflow automatically.

6.4. RDF DATA CUBES

6.4.1. RDF DATA CUBES DESIGN

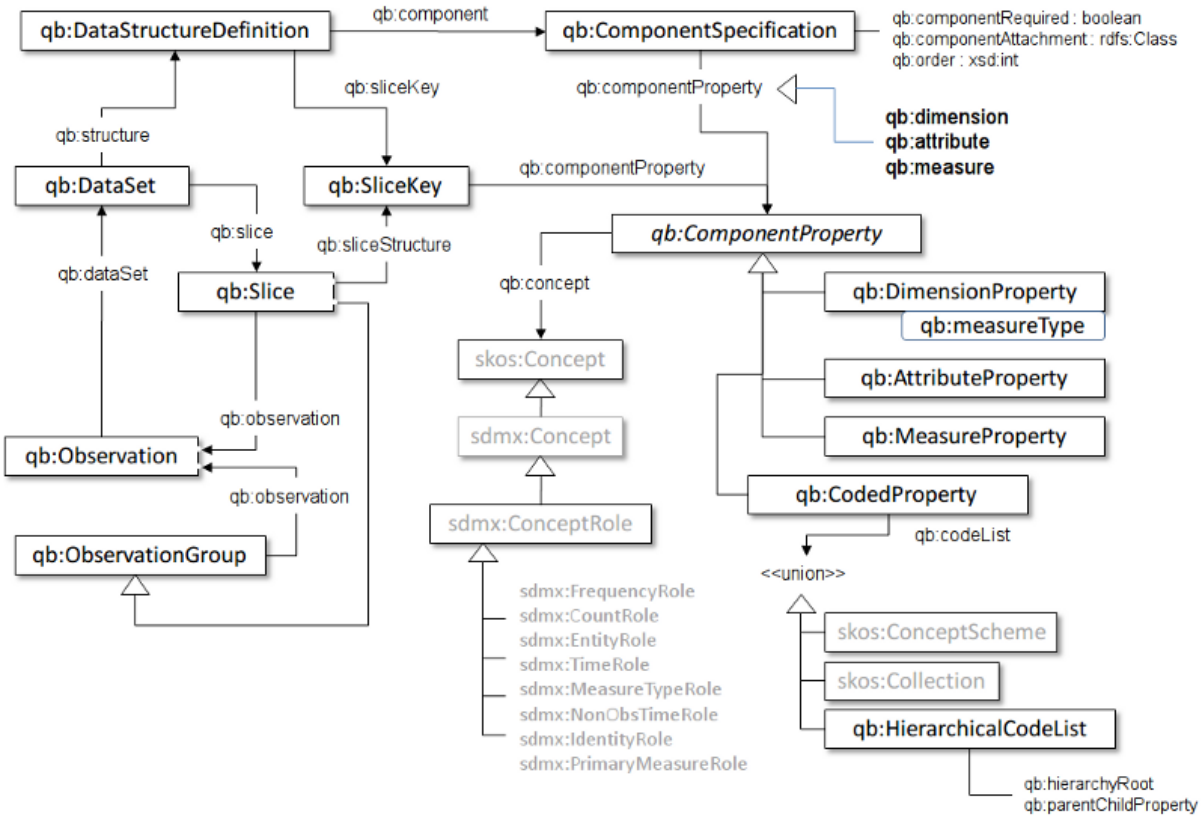

Figure 11: Pictorial summary of key terms and their relationship.

6.4.2. RDF Data cubes Architecture

The vocabulary of the RDF Data Cube was recently proposed as a W3C Recommendation for the destruction of various mathematical data using the RDF format. The model under this vocabulary is based on the SDMX model, the ISO standard for representing statistical data that can be shared between organizations. . The data cube, called the dataset, is an example of class qb:Dataset. Cube data cells, called observation, are the conditions of phase qb:Observation. Each look can have one or more attributes, sizes, and dimensions of a structure. Features defining qb type steps: MeasureProperty. This positive relationship correlates with recognition and measurement values (e.g., value of humidity), which have a range of numbers. Qualities and sizes of species qb:AttributeProperty and qb:DimensionProperty, respectively. The size represents the viewing context (e.g., the city in which the humidity is calculated), while the attributes provide additional information about the measure, such as the unit of measure.

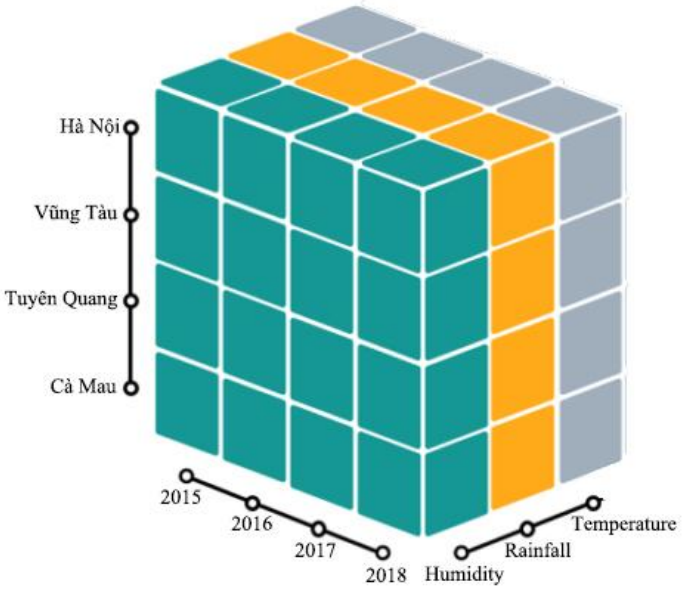

Figure 12: Climate Data Cube in 3D

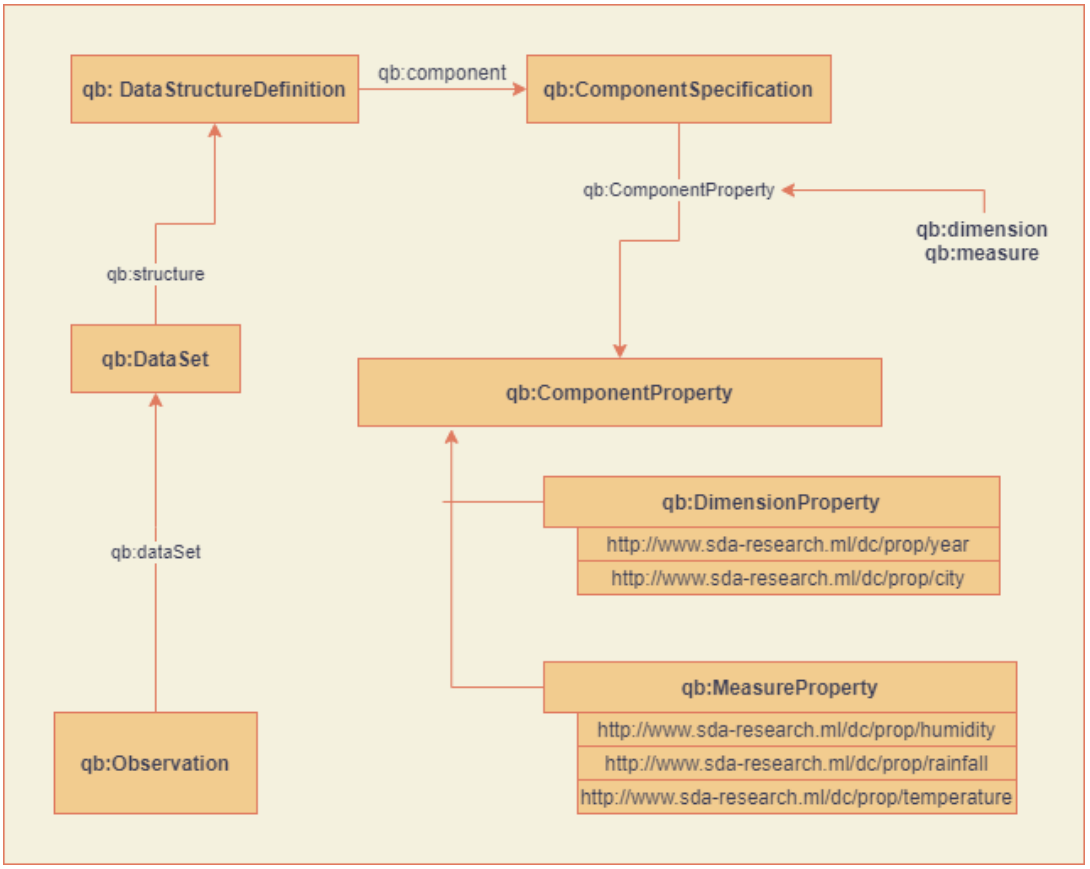

Figure 13: Climate Data Cube Structure

As the schema above, the Data Structure Definition has a child node ComponentSpecification, where qb:ComponentProperty is configured. The qb:ComponentProperty includes two main concepts, qb:DimensionProperty and

qb:MeasureProperty - the DSD definitions for each dimension and measure in the data warehouse. Each class has an specific URI that is regulated for querying with SPARQL Endpoint and makes the Linked Data Structure for the Data Cube.

For instance, The Climate Data Cube structure has:

- Dimensions:
    - City : Hà Nội, Vũng Tàu, Tuyên Quang, Cà Mau,etc...
    - Year : 2015, 2016, 2017, 2018,etc...
- Measures:
    - Humidity
    - Rainfall
    - Temperature
- Observations:
    - Humidity value of Hà Nội in 2015, 2016.
    - Rainfall value of Cà Mau in 2015, 2016.
    - Temperature of Tuyên Quang in 2017, 2018.
    - ...etc...

When a user need to figure out a set of statistical data that depends on multiple categories, such as humidity of an area/city in a specific time period (year), the data cube will refer to the statistics value (observation) of the measure (humidity) that match the dimension (city, year).

For instance, this is the observation of the rainfall value of Tuyen Quang in 2015.

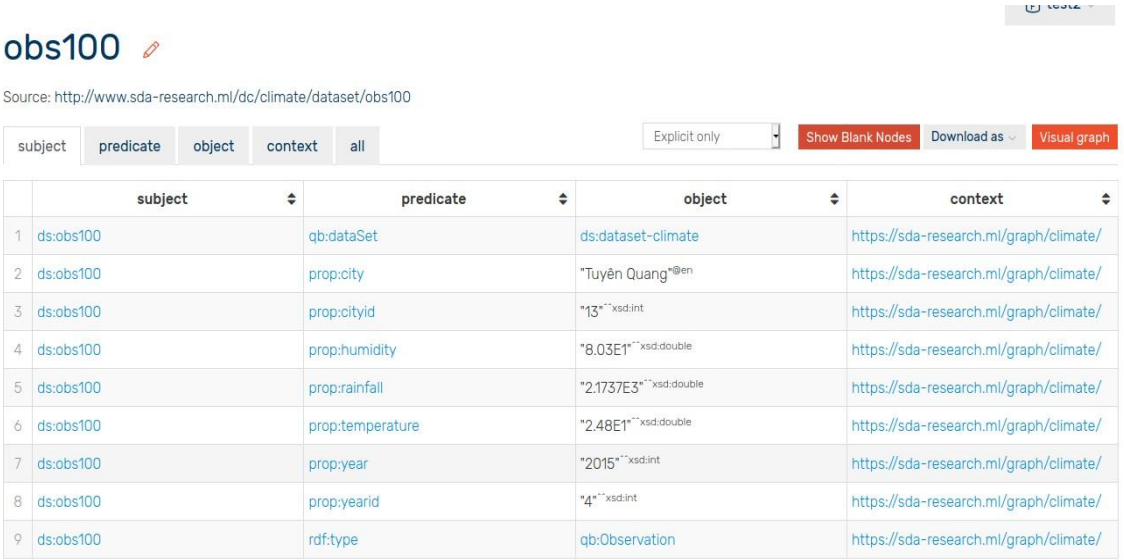

Figure 14: Observation 100th.

6.5. RDF DATA CUBES AUTO GENERATOR

6.5.1. Traditional R2RML Methods

At the present time, although there are many tools to convert data from tabular to linked data, there is still a lack of self-synchronization to make conversion flows fully automated without needing to rely on manual work each time.

In manual conversion methods, for example with classic R2RML engines like figure 6.1.4, we can see that the conversion relies on mapping files with the R2RML language. These mapping files are required to be manually defined by default, which leads to a lot of limitations in expanding the linked data stores.

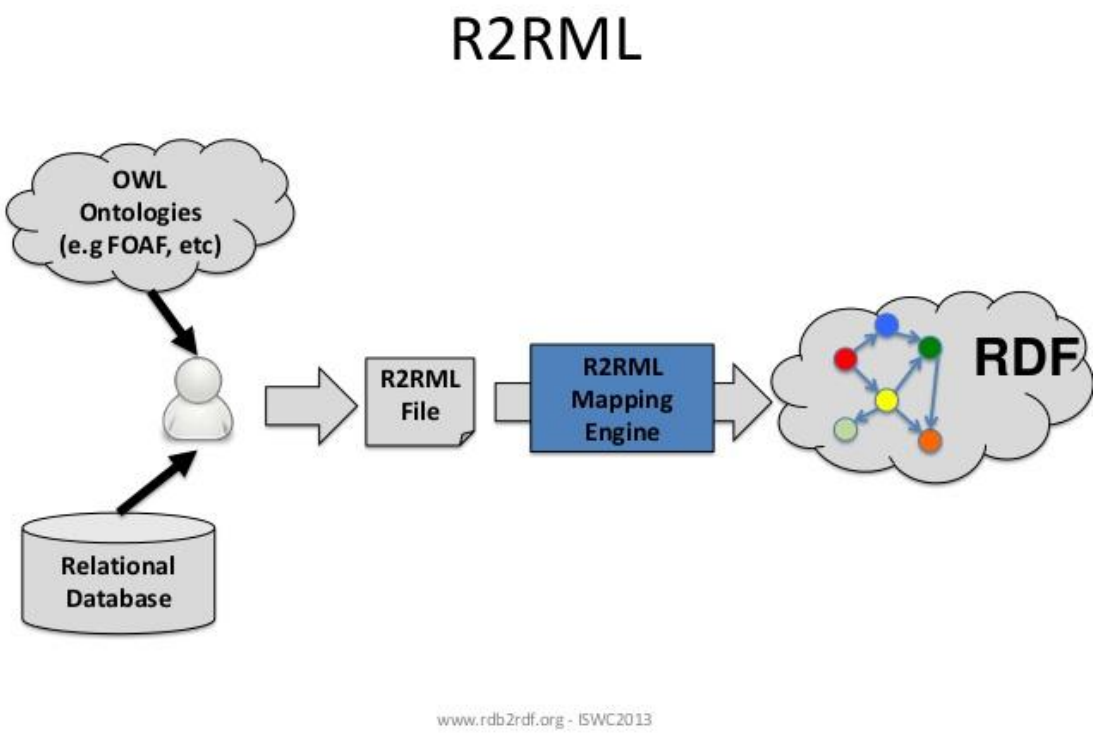

Figure 15: Data flow of Manual Relational Database to RDF process.

Let's suppose, we have a system where data is crawled and processed continuously, not just a static database. With a traditional R2RML engine, this cannot be executed automatically, because as we know, each data source is specified by a mapping file, and it must be defined manually with the R2RML vocabulary. If there is a new data source, we must write an additional mapping file and restart the r2rml engine. This is completely inappropriate for systems where data pipelines are run and fed into a persistent data warehouse.

6.5.2. Automatic RDF Data Cubes Generator

For EDSDA, a system where data is scraped continuously and allows users to upload their primary data source to the system, it is not appropriate to use traditional R2RML methods. Therefore, we have researched and implemented an automatic conversion method from Relational Database and Tabular Data to RDF Data Cubes (Figure 16).

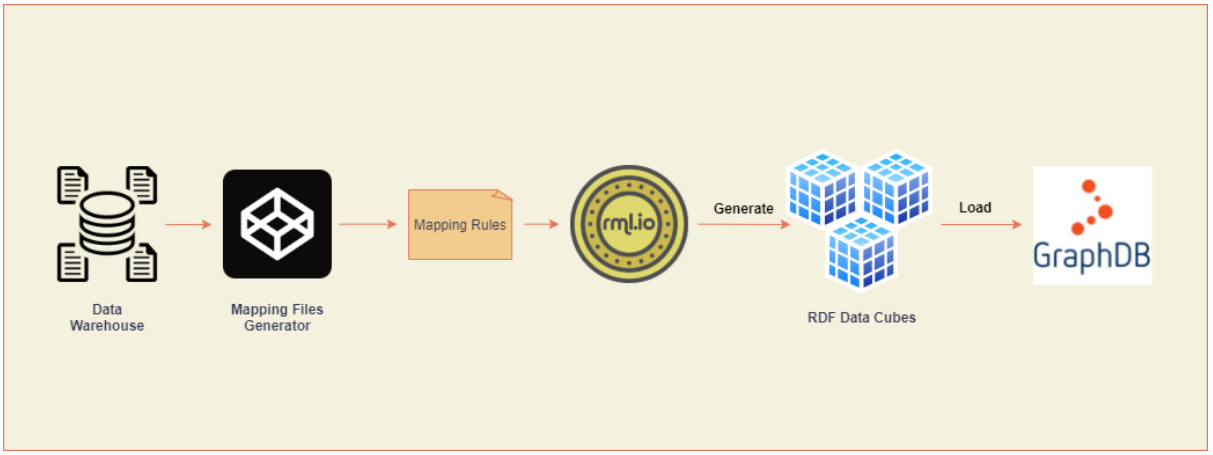

Figure 16: Data flow of Automatic RDF Data Cubes Generator.

As illustrated for the process above, the basic EDSDA RDF Data Cubes Generator will have 6 main parts:

- Data Warehouse: At first, data will be loaded manually from the data warehouse. Then, every time users upload new data sources, the system will check if the data warehouse is updated and start getting the new data source from it.
- Mapping Files Generator: After a new data source is obtained, the Mapping Files Generator will be launched. In the first version, it will handle the cases where the fact table has 1,2 or 3 measures and dimensions. Names of the fact table, measures, and dimensions will be assigned as variables in the R2RML mapping language.
- Mapping Rules: Mapping Files generated from Mapping Files Generator.
- RML.io: To execute the process of generating RDF Data Cubes from Mapping Rules and data sources, we use the RMLMapper and RMLStreamer frameworks provided by the RML.io organization. They are platforms that help users produce high quality linked data and knowledge graphs that are fast and easy to install.
- RDF Data Cubes that are generated from the previous parts.

- OntoText GraphDB: RDF data blocks will be uploaded to the Ontotext GraphDB datastore through the API to respond to data requests from the server and publicize open linked data to end users.

6.6. APPLICATION APIS

For interacting between client and server, we choose to deploy EDSDA onto a Linux Virtual Private Server. Using Linux VPS, it is very easy and simple to develop, maintain and especially run the server in the most stable way.

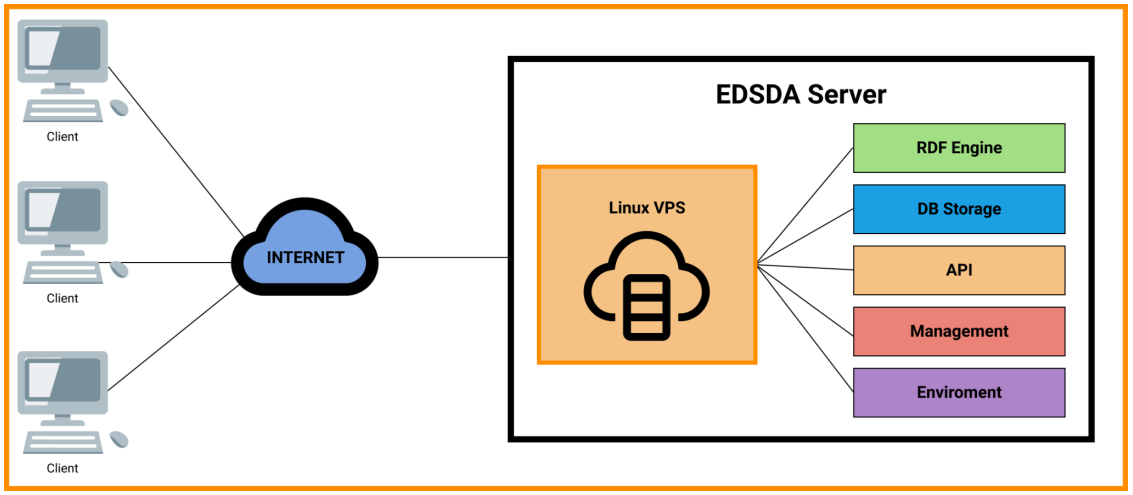

Figure 17: Deployment Diagram

We write the APIs of EDSDA system with enapso-graphdb library, a Javascript library that give developers the most flexible way to configure, connect and interact with OntoText GraphDB platform, and the help of Express.JS, the most famous also strongest NodeJS library for building REST API server.

7. REFERENCES

- Server Endpoint: http://server.sda-research.ml
- Figma: https://figma.com
- ETL (Extract, Transform, and Load) Process: https://www.guru99.com/etl-extract-load-process.html
- Extract, transform, load: https://en.wikipedia.org/wiki/Extract,_transform,_load
- What is Extract, Transform, Load? Definition, Process, and Tools: https://www.talend.com/resources/what-is-etl/
- ETL - Extract, Transform, Load: https://www.webopedia.com/TERM/E/ETL.html
- What is OLAP? Cube, Operations & Types in Data Warehouse: https://www.guru99.com/online-analytical-processing.html
- What is OLAP: https://www.ibm.com/cloud/learn/olap
- OLAP cube: https://en.wikipedia.org/wiki/OLAP_cube
- OLAP (Online analytical Processing): https://techterms.com/definition/olap
- The RDF Data Cube Vocabulary: https://www.w3.org/TR/vocab-data-cube/

- Optimizing RDF Data Cubes: http://ceur-ws.org/Vol-1426/paper-02.pdf
- CubeQA—Question Answering on RDF Data Cubes: https://www.researchgate.net/publication/313073658_CubeQA-Question_Answering_on_RDF_Data_Cubes
- RDF Data Cube - graphical representation : https://www.researchgate.net/figure/RDF-Data-Cube-graphical-representation_fig3_265690180
- Linked clinical data cube architecture aligned with the RDF data cube: https://www.researchgate.net/figure/Linked-clinical-data-cube-architecture-aligned-with-the-RDF-data-cube-Depicts-the_fig2_274700091
- The RDF Data Cube: https://www.w3.org/TR/eo-qb/#Datacube
- Data Warehouse Architecture: Types, Components, & Concepts: https://www.astera.com/type/blog/data-warehouse-architecture/
- Data Warehousing - Architecture: https://www.tutorialspoint.com/dwh/dwh_architecture.htm
- Data Warehouse Architecture: Traditional vs. Cloud: https://panoply.io/data-warehouse-guide/data-warehouse-architecture-traditional-vs-cloud/
- RDB2RDF Tutorial (R2RML and Direct Mapping) at ISWC 2013: https://www.slideshare.net/juansequeda/rdb2-rdf-tutorial-iswc2013
- RML.io: https://github.com/RMLio
- Generating Executable Mappings from RDF Data Cube Data Structure Definitions: https://link.springer.com/chapter/10.1007/978-3-030-02671-4_21
- Apache airflow documentation: https://airflow.apache.org/docs/apache-airflow/stable/