COMMITMENT

We assure that this is our independent scientific research. The data used in the dissertation analysis has a clear, published source in accordance with the regulations. The research results in our thesis are self-study, analyzing honestly, objectively, and in accordance with the reality of Vietnam.

Implemented by Team 06

ACKNOWLEDGEMENT

We are overwhelmed in all humbleness and gratefulness to acknowledge our depth to all those who have helped us to put these ideas, well above the level of simplicity and into something concrete.

We would like to express our special thanks of gratitude to my mentor - Nguyen Thanh Binh as well as our principal who gave me the golden opportunity to do this wonderful project on the topic "Smart Dashboard Application", which also helped us in doing a lot of Researches and we came to know about so many new things. We are really thankful to them.

Any attempt at any level can not be satisfactorily completed without the support and guidance of our parents and friends. We would like to thank my parents and friends who helped us a lot in gathering different information, collecting data, and guiding me from time to time in making this project. Despite their busy schedules, they gave us different ideas in making this project unique.

Team 06

ABSTRACT

Problem statement:

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. In this data booming world, some traditional technologies can no longer serve the need to analyze the large volume of data, especially data about the environment. So that the need for a faster and cheaper way of analyzing and aggregating environmental data is necessary.

Solution:

To solve this problem, based on collecting data sources, Data ETL Technology and RDF Data Cubes, 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.

Result:

Building a data warehouse gathered from Vietnam's environmental data sources. In this context, data have been extracted, transformed and loaded (ETL) into RDF data cubes for online analytical processing (OLAP).

Building a smart dashboard application to help environmental experts and policymakers synthesize and visualize data directly and easily.

SDA provides users with many different types of data graphs, tables and visualization maps, from which users can provide insight and assessment of environmental data of one or more quickly and conveniently between different regions.

TABLE OF CONTENTS

LIST OF ACRONYMS AND TERMS

Abbreviations / Terms	Explain
SDA	Smart Dashboard Application
API	Application Programming Interface
GUI	Graphic User Interface
CSV	Comma Separated Values
XML	Extensible Markup Language
JSON	JavaScript Object Notation
RDF	Resource Description Framework
SDMX	Statistical Data and Metadata eXchange
ISO	International Organization for Standardisation
ETL	Extract, Transform, Load
NGO	Non-Governmental organization
OLAP	On-Line Analytical Processing
Viz	Visualization
C&C	Components & Connectors
GCloud	Google Cloud

LIST OF FIGURES

LIST OF TABLES

INTRODUCTION

1. Problem statement

Vietnam Nam has made a transition from an inter-governmental economy to a market-focused program with unprecedented success. In 2010, the Vietnam economy grew by 6.8%, supported by global economic recovery, monetary policy, and consumer growth. In 2010, per capita income increased by more than $ 1,000 a year, and in 2008 poverty levels dropped to 14% from 60% in 1993. Vietnam Nam became a member of the World Trade Organization in January 2007. In 2010, the country experienced a low level of middle income.

The biggest challenge for Vietnam is to manage its rapid development in a sustainable manner and to prevent the negative effects of environmental degradation and climate change. Industrialization, urbanization, and agricultural intensification have had a devastating effect on air, land, and water, and have far-reaching effects on the energy and transportation sector that have led to increased greenhouse gas emissions and reduced climate change.

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 cost. As we know, the environmental data of Vietnam spread widely on the Internet and there are very few projects for helping to collect and analyze these data in just one centralized system.

The main challenge for that problem is how can we handle a widely spreading environmental data source, centralize and make it easy for the user to interact with it. It is hard to transform a huge amount of distributed data into an interactive system that anyone can use.

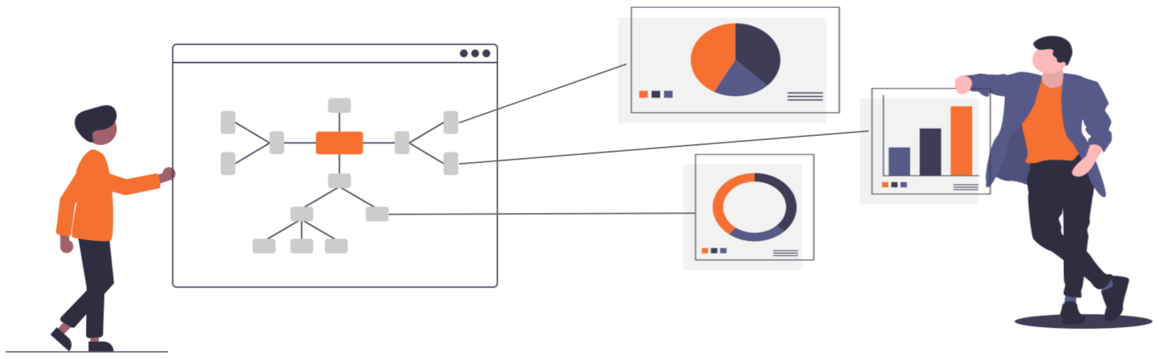


Figure 1: Wireframe of Application

2. Prior Art

There are many research topics on data science applied to policy-making. Here are some researches that come from data scientist groups around the world

Data science empowering the public (https://www.waze.com/): The Smart City dashboards in Rio de Janeiro, Brazil, were created to solve problems related to public transportation and traffic. For this, an infrastructure, a dashboard, and a data portal with more than three thousand datasets and seven APIs for real-time data use (www.data.rio) were developed and used by the Center of Operations Rio (COR).

Interactive Dashboards (http://www.data.rio/): Using Visual Analytics for knowledge Transfer and Decision Support (Samar Al-Hajj, Ian Pike, Brian Fisher) -A visual analytics dashboard that reflects the needs and preferences of injury stakeholders. The types of visualizations were selected to efficiently illustrate trends and patterns in injury data.

Triangulum City Dashboard (https://moovitapp.com/): An Interactive Data Analytic Platform for Visualizing SmartCity Performance (Mina Farmanbar, Chunming Rong).

It can be seen that data science is increasingly weaving and useful in all areas of life, from urban management to public policy management, and help in making decisions. But unfortunately, the world today is facing a vital problem, the environment. However, in today's market, there are very few data analysis platforms to help policymakers make the right decisions about environmental protection.

This is the reason that makes us build a smart dashboard called Smart Dashboard Application - SDA, the first environment data analysis dashboard in the market now.
Smart Dashboard Application - SDA - means that a dashboard that is convenient for users to analyze and review data. It includes several hundred datasets about real-time information of the environment, such as rainfall, humidity, temperature, population, afforestation, measurement data of air pollutants… SDA 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 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. The users can easily choose and integrate any data that are satisfied with their needs and visualize it in many kinds of charts and maps so that they - can consider the factors affecting the environment in the most comprehensive and intuitive way. With this dashboard, individuals or any subjects can take advantage of environmental data to be able to decide the best relevant policies.

3. Decision-Making Dashboard

Environment is the most basic and important human life factor. It concerns all aspects of the human being, like transportation, housing, health and sanitation, security and protection, infrastructure and communication systems in continuous interaction and change. In this scenario and based on the deployment of digital technology, a huge amount of environmental data is produced on a daily basis. This data can be used to build and measure indicators to study and analyze different environmental phenomena. Information obtained through such indicators enables us to understand environmental events, develop public policies and implement corrective actions.

Successful support for administrative decision-making depends to a large extent on the availability of integrated, high-quality information organized and presented in a fast and easy-to-understand way. Data Warehouse has already met this requirement. They serve as an integrated repository for internal and external data - intelligence critical to understand and evaluate a business in its natural state. With the addition of models, analytics tools, and user networks, they have the opportunity to provide useful information resources that support practical problem and opportunity identification, critical decision-making, and strategic planning, implementation, and evaluation. So that, with the help of RDF Data Cube data platform, Data Warehouse in a semantic environment is one of the best solutions for building a Decision-Making Dashboard.

4. Challenges

The major challenge of Smart Dashboard Application is the lack of open data of the environment in Vietnam. As we all know, public statistics of Vietnam have only developed in the last few decades, as well as many data sources are still not digitized and public openly. Therefore, scratching the data and gathering data sources for the environmental topic and related aspects becomes very difficult.

Besides, because of the limited research time, we encountered some problems in accessing new technologies and applying them to the project. Besides the lack of data, it is also a huge challenge for us.

OBJECTIVES

The topic focuses on the following main goals:

- Research on data technology: ETL processing, Linked Data and RDF Data Cube and Data Warehouse.

- Apply RDF Data Cube technology to synthesize, analyze, filter, and visualize data on the environment (climate, population, industry, ...) of Vietnam.

RESEARCH SCOPE

- Smart Dashboard Application helps users access information about the environment in Vietnam directly with a single system without having to search for information manually.

- With the biggest goal of giving users direct experiences and the easiest way in manipulating data, not only with information retrieval, but Smart Dashboard Application also provides a method of data aggregation. on many different aspects related to the environment in Vietnam, thereby maximizing time and manipulation in analyzing and synthesizing environmental information compared to manual information research.

- Smart Dashboard Application provides users with a variety of charts, tables, and data visualization maps, from that users can give insight and assessment of the environmental data for one or more and many different regions quickly and conveniently.

RESEARCH METHODS

- Researching Linked Data data technology (linked data) and RDF Data Cube (multidimensional data cube).

- Design a data warehouse in Vietnam's environment and convert them to linked data for multidimensional retrieval and data synthesis and analysis.

- Design a system of smart information dashboards to help users synthesize and visualize data on the environment directly and easily.

REPORT STRUCTURE

CHAPTER 1 (page 13): Background of research: An overview of SDA's data-based technology, including Data Warehouse and RDF Data Cubes Technology.

CHAPTER 2 (page 19): Clarify problem solving, bring up business problems, and architectural deployment.

CHAPTER 3 (page 37): Implement and perform research.

CHAPTER 1. BACKGROUND

1.1 Data Warehouse

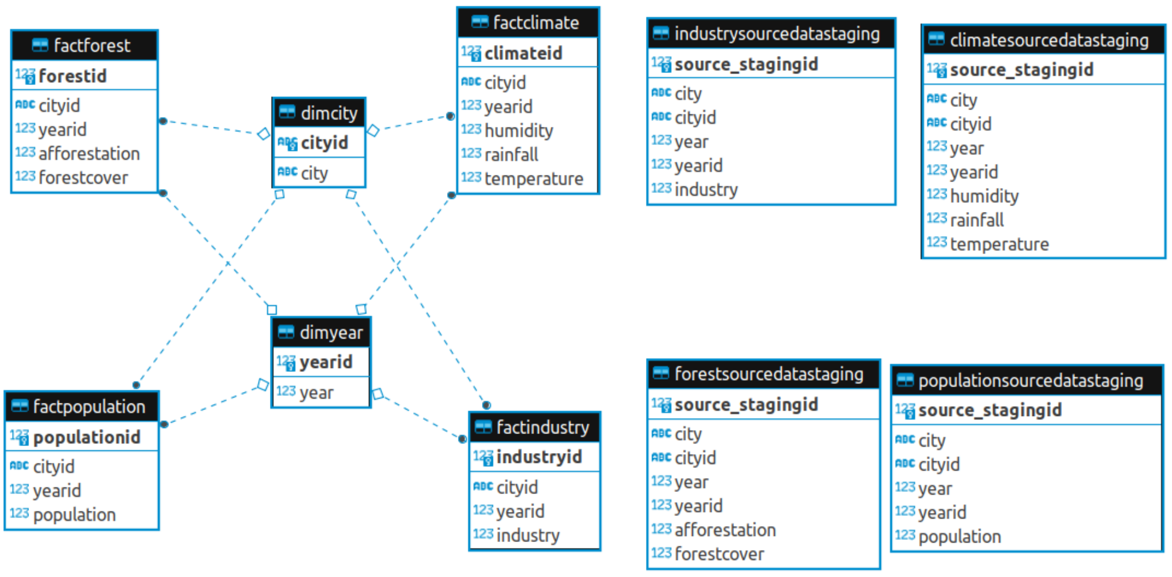


Figure 2: Data warehouse Schema

For more than a decade, Business Intelligence systems have been widely used to provide complex information during decision-making. Performance data is periodically extracted, converted, and uploaded to a stable and integrated storage facility, called Data Warehouse (DW), which organizes data into multiple cubes. Decision-making is based solely on datasets to provide an overview of the organization's activities. Under today's highly competitive business content, additional information from outside the organization, which is widely available on the web, should also be included in the analysis to provide more feedback to decision-makers

A data warehouse is directed to the topic as it provides information about the theme instead of the ongoing corporate activities. These topics can be marketing, distribution or even policy,.. etc.

A data warehouse does not focus on continuous operation. Instead, it emphasized modeling and analysis of decision-making data. It also provides a simple and concise view surrounding a particular topic by not including data that does not help support the decision-making process.

In the SDA system, we use Data Warehouse Technology to build a pre-integration database, before extracting it to the Linked Data platform as RDF Data Cubes.

1.2 Linked Data

Statistical data is currently published in a wide range of formats and standards that do not allow connectivity to all data sets. It is used as a basis for policy prediction, planning and adjustment, and therefore has a significant impact on society (from citizens to business to government). The Linked Data paradigm has opened up new opportunities and strategies for government agencies to open up information and exchange information. The data is open when it is legally open (available in standard machine-readable format, which means it can be retrieved and properly processed by a computer program) and officially opened (explicit license in a way that allows commercial and non-commercial use and re-use without restrictions).

The Linked Data method enables data sets to connect together by referring to common concepts. The database is represented in the form of a graph, using the Resource Description Framework (RDF) as the most commonly understood language. The linked data publishing process refers to a set of tasks related to the issuance, conversion, validation, and extraction of RDF data shares from various sources (e.g., Information) on the Web. The RDF ready-to-use database can be stored locally or registered in a metadata catalog.

Linked Data, a set of four design principles put forward by Tim Berners-Lee in 2006, serve exactly that purpose. Hence, in order to publish Linked Data, publishers should:

- Use Uniform Resource Identifiers (URIs) as names for things, e.g. http://dbpedia.org/resource/Brussels can be used for referring to the city of Brussels.

- Use HTTP URIs, so that people can look up those names.
- When someone looks up a URI, provide useful information, using the standards (i.e. RDF, SPARQL).
- Include links to other URIs, so more things can be discovered, e.g. from http://dbpedia.org/resource/Brussels a link is available to http://dbpedia.org/resource/Belgium.

Links and relationships can be identified between:

- Overlapping data resources, i.e. data resources that refer to the same entity (often sharing some common information). In this case, the linking takes place at the unique identifier level (i.e. URI) of the different data resources. For example, the DBpedia resource for the city of Brussels (accessible at http://dbpedia.org/resource/Brussels) can be linked to the one maintained by the Statistics Belgium (accessible at http://location.testproject.eu/so/au/AdministrativeUnit/STATBEL/21004). Linking these two data resources allows us to get richer information about Brussels.
- Complementary data resources, i.e. data resources that refer to different entities that somehow relate.

1.3 RDF Data Cube

1.3.1. RDF Data Cube Vocabulary

The concept of multidimensional modeling, as it is known today, was suggested by Kimball and later deepened and developed in Kimball. According to Kimball, the main advantage of the multi-model model is its simplicity, which is important to empower users to understand data and allow for efficient recovery. Multidimensional models are designed to store statistical data, compiling a collection of observations done in certain places in a logical place. This collection is characterized by size that defines a set of each observation and metadata that defines the value, as measured and how the observation is expressed. Statistical data can be set multidimensional in space, like a hypercube. A cube is arranged according to a set of dimensions, attributes, and measures.

Nearly all of these have an RDFS translation/definition in the RDF Data Cube vocabulary (in what follows, qb is the prefix for http://purl.org/linked-data/cube#):

Data cube	qb:DataSet
Dimension	qb:DimensionProperty
Observation	qb:Observation
Measurement	qb:MeasureProperty
Unit	Not in RDF Data Cubes but can be found in other vocabularies, e.g. QUDT.
Slice	qb:Slice

Using that vocabulary and adding a few new "words", a data cube has been defined as follows:

A qb:DataSet has a qb:DataStructureDefinition that defines the structure of the cube. The structure is specified by means of a qb:ComponentSpecification containing a number of qb:ComponentProperty sets, detailing qb:DimensionProperty to define the dimensions of the cube, qb:MeasureProperty to define the measured variables, and qb:AttributeProperty to define structural metadata such as the unit of measurement. The observations themselves are included as a qb:Observation for each cell of the cube.

For those who are interested to know more of the technical details, each of these classes has a formal semantic definition in the RDF Data Cube vocabulary specification. We'll limit ourselves to just one example here: the definition of the Observation Class.

```
qb:Observation
  rdf:type rdfs:Class ;
  rdfs:comment "A single observation in the cube, may have one or more
   associated measured values"@en ;
  rdfs:isDefinedBy <http://purl.org/linked-data/cube> ;
  rdfs:label "Observation"@en ;
  rdfs:subClassOf qb:Attachable ;
  owl:equivalentClass scovo:Item ;
```

Some of this will already be sufficiently familiar and readable:

The subject of our statements here is the resource identified as qb:Observation.

We use rdf:type to say that qb:Observation is a resource of type rdfs:Class.

With rdfs:comment, we add a human-readable description of qb:Observation.

We use rdfs:isDefinedBy to indicate that qb:Observation is a resource described by the vocabulary known as <http://purl.org/linked-data/cube>.

We use rdfs:label to assign the literal string “Observation” as the label in English.

The formal semantics are expressed in the last two statements:

qb:Observation is an rdfs:subClassOf the resource known as qb:Attachable.

qb:Observation is an owl:equivalentClass of the resource known as scovo:Item.

Omitting the prefixes for a moment, the use of subClassOf means that every instance of an Observation is also an instance of the class Attachable (an abstract superclass for everything that can have attributes and dimensions.). The use of equivalentClass means that every instance of an Observation is also an instance of the class Item defined in another vocabulary called The Statistical Core Vocabulary (SCOVO, meanwhile deprecated), and conversely every such Item is also an instance of Observation.

1.3.2. Transfer from Data Warehouse to RDF Data Cubes

Relational DW tools are not flexible, because uploading RDF data to a related analysis schema can lead to substantial facts or incomplete or repeated steps; the latter is not compatible with the multidimensional relationship settings and DW tools. Most importantly, the full utilization of RDF graphs, variations and rich semantics of RDF data should be maintained through a series of inventory until the query is analyzed. In particular, RDF analytical questions should be allowed to ask for a shared schema and data, e.g.,. Changes to the basic database should not cause the asset schema to be rebuilt; instead, new resources (and their structures) should flow smoothly into the analysis schemes and cubes.

R2RML and Direct Mapping are the two important pieces for the solution. Direct Mapping is a default mapping that automatically generates RDF from the relational data, with the push of a button. R2RML is a mapping language where a user can customize which relational tables and columns get mapped to RDF using a specific vocabulary/ontology. Direct Mapping and R2RML complement each other. As a first step, a user may want to run the Direct Mapping first to see what the RDF looks like and have a pre-populated R2RML file. Afterwards, the user can customize the R2RML. Consider the following SQL:

```
CREATE TABLE Employee (
id int PRIMARY KEY,
name VARCHAR(100));
INSERT INTO Employee (id, name) VALUES (1, Hoa Vo);
```

The following R2RML:

```
@prefix rr: <http://www.w3.org/ns/r2rml#> .
@prefix ex: <http://ex.com/onto/> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
<TriplesMap1>
  a rr:TriplesMap;
  rr:logicalTable [ rr:tableName "Employee" ];
  rr:subjectMap [ rr:template "http://mycompany.com/Employee/{id}";
                  rr:class ex:Employee; ];
  rr:predicateObjectMap [ rr:predicateMap [ rr:constant ex:id ];
                  rr:objectMap    [ rr:column "id" ] ];
  rr:predicateObjectMap [ rr:predicateMap [ rr:constant foaf:name ];
                  rr:objectMap    [ rr:column "Name" ] ] .
```

would generate the following RDF:

```
<http://mycompany.com/Employee/1>
    <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
    <http://ex.com/onto/Employee>.
<http://mycompany.com/Employee/1>
    <http://ex.com/onto/id>
    "1"^^<http://www.w3.org/2001/XMLSchema#integer>.
<http://mycompany.com/Employee/1>
    <http://xmlns.com/foaf/0.1/name>
    "Hoa Vo".
```

It's very convenient that there is a platform for automatically executing these processes, called Google Open Refine. In SDA, we mostly use it to generate RDF Data Cubes from the Data Warehouse.

2.1. Business Problems

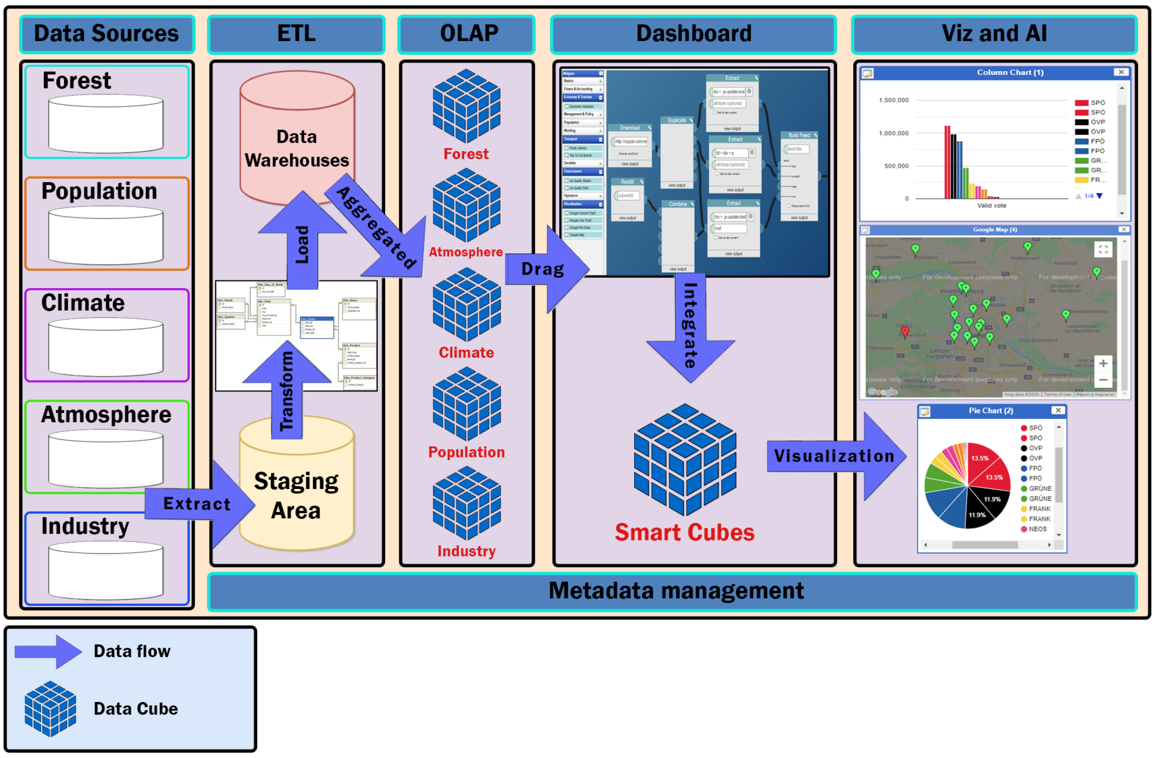


Figure 3: Context Diagram of System

Prose:

- **Data Sources :**
  - Collect from environment open data platforms of the governments and NGO organizations.
  - Use web crawling techniques to crawl data from related environment websites.
  - Data Format : CSV, JSON, XML.
- **ETL (Extract, Transform, Load)**
  - **Extract :**
    - In this step, data is extracted from the source system into the staging area. Transformations if any are done in the staging area so that performance of the source system is not degraded. Also, if corrupted data is copied directly from the source into the data warehouse database, rollback will be a challenge. The staging

        area gives an opportunity to validate extracted data before it moves into the Data warehouse.

    - **Transform :**
        - Data extracted from the source server is the raw data and not usable in its original form. Therefore it needs to be cleansed, mapped, and transformed. In fact, this is the key step where the ETL process adds value and changes data such that insightful reports can be generated. In the transformation step, we filter, clean, split and integrate the data to match with the system requirements and data warehouse architecture.
    - **Load :**
        - Loading data into the target data warehouse database is the last step of the ETL process.
- **OLAP**
    - In this step, we export the data from the data warehouse as RDF data cubes for better query performance, data binding, and scalability, in addition to information transparency.
- **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.
- **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.

2.2. Proposed Solution

2.2.1 Technical Constraints

Main Programming Language: Javascript, Python.

Data Warehouses:

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

Data Cubes

- Programming Language: RDF-Graph, SPARQL.
- Tool for converting from Datawarehouse to RDF Data Cube: OpenRefine.
- Storing Platform & Endpoint : GraphDB.
- Network Accessing: RDF-Rest API.

Server:

- Programming Language: Javascript.
- Framework: ExpressJS (NodeJS).
- Libraries: enapso-graphdb-client
- Operating System: Windows, Linux, macOS.
- Deployment Environment: VPS Engine.
- Network Accessing: HTTP methods (POST, GET) via RESTful API.

Client:

- Programming language: HTML, CSS, Javascripts.
- Framework: React, Redux.
- Libraries: Material-UI, react-dnd, beautiful-react-diagrams, highcharts
- Deployment Environment: Google Firebase Hosting
- Operating System: Windows, Linux, Mac OS.
- Web Browser: Chrome, Firefox, Microsoft Edge, Coccoc
- Network Accessing: World Wide Web (WWW), HTTP methods (POST, GET) via RESTful API

2.2.2 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 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 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: Quality Attributes: Availability

2.3 Architectural Overview

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 the attached page.

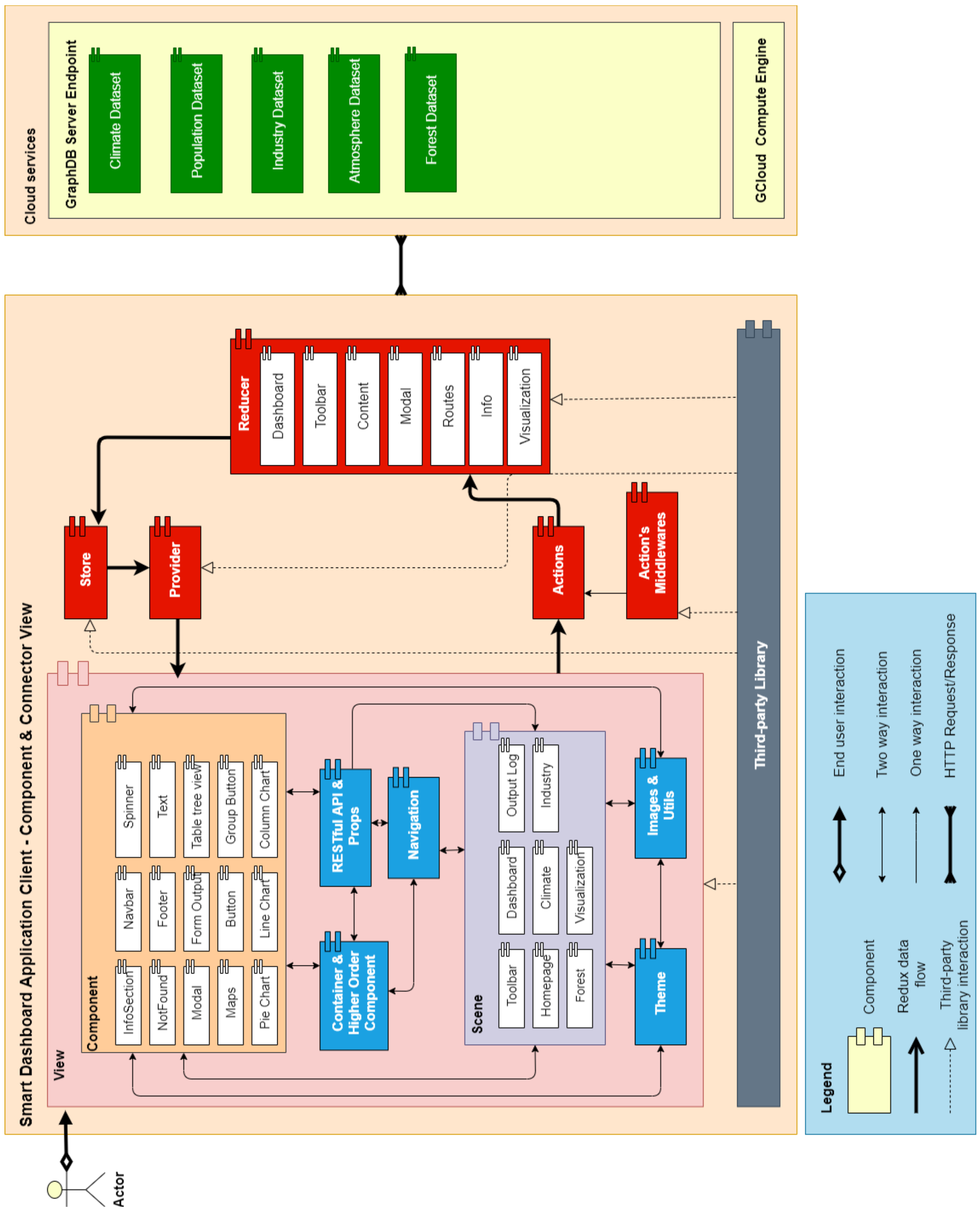


Figure 4: C&C View

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.

Table 4: C&C Role & Responsibility Description

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	The pattern that has proven to be very valuable for several React libraries
Restful API & Props	Provides 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.

Role & Responsibility	**Description**
Actions	Actions are payloads of information that send data from your application to your store.
Actions’ Middlewares	It provides a third-party extension point between dispatching an action, and the moment it reaches the reducer.
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 that access the features or data of an operating system, application, or another service.

2.4. Modules view

2.4.1. Module View based on Web Application

References Module View on the attached page.

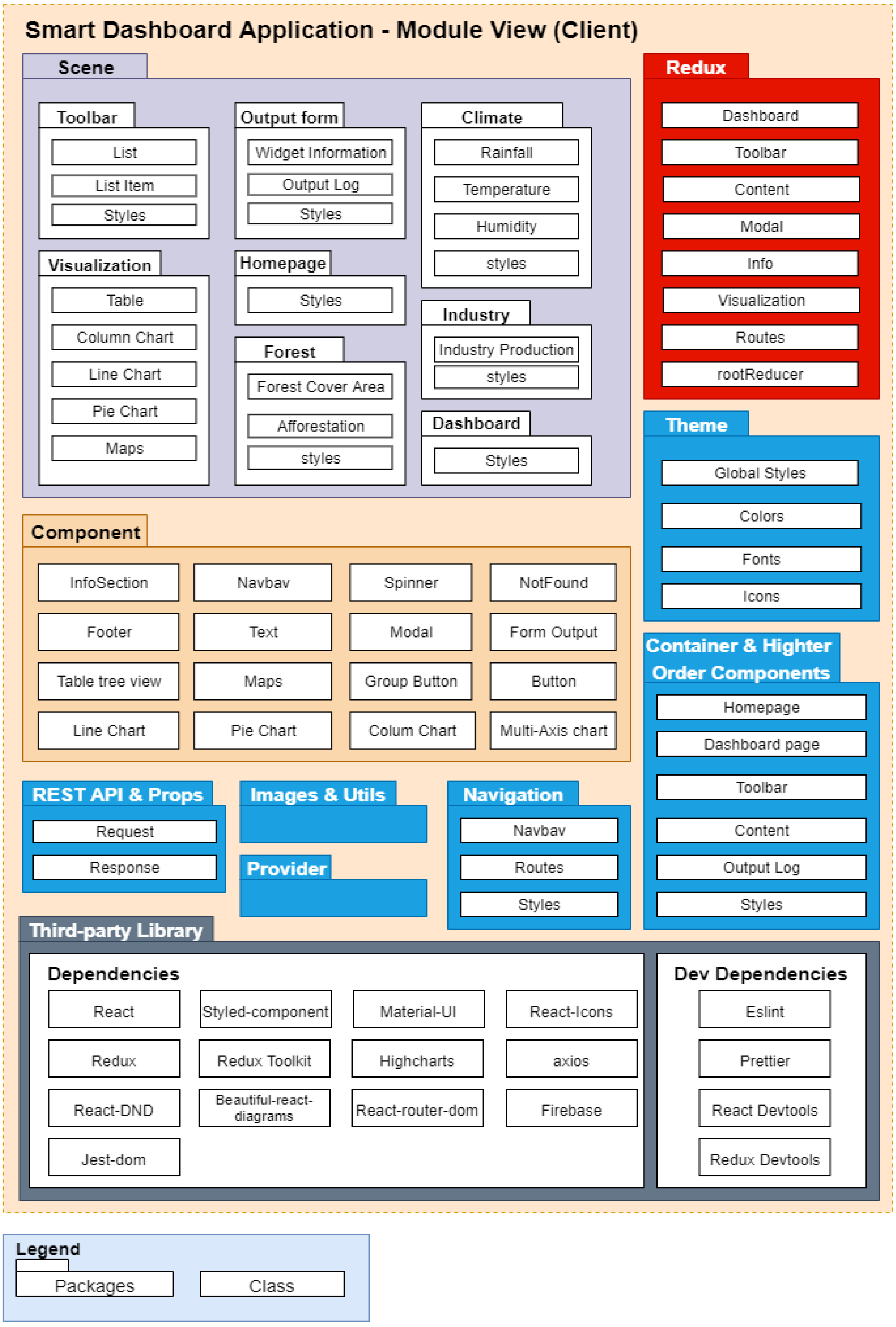


Figure 5: Module View Client

Prose:

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

The Scene package includes:

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

The 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.

Besides 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 a cloud service. Cloud service includes GraphDB and GCloud service

Table 5: Modules View base on Client Role & Responsibility Description

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

Role & Responsibility	Description
Climate	DataCube
Industry	DataCube
Forest	DataCube
Component	
InfoSection	Section Information
Navbar	Navigation
Spinner	A UI when the page is loading
NotFound	A 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 the chart
Column Chart	Visualization data from data cube in the chart
Pie Chart	Visualization data from data cube in the 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 the state of Toolbar
Content	A Redux store & reducer for handling state of Content
Modal	A Redux store & reducer for handling the state of Modal
Routes	Define routes of scenes for navigation

Role & Responsibility	Description
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
Content	Handle widget node & connector in Mainboard
Output log	Handle widget information & data log
Rest API & Props	
Request	Used to send the request to get data from the 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 are required for operating the application
DevDependencies	A list of dependencies that are required for the development environment

2.4.2. Module View based on Server

References Module View - Server on the attached page.

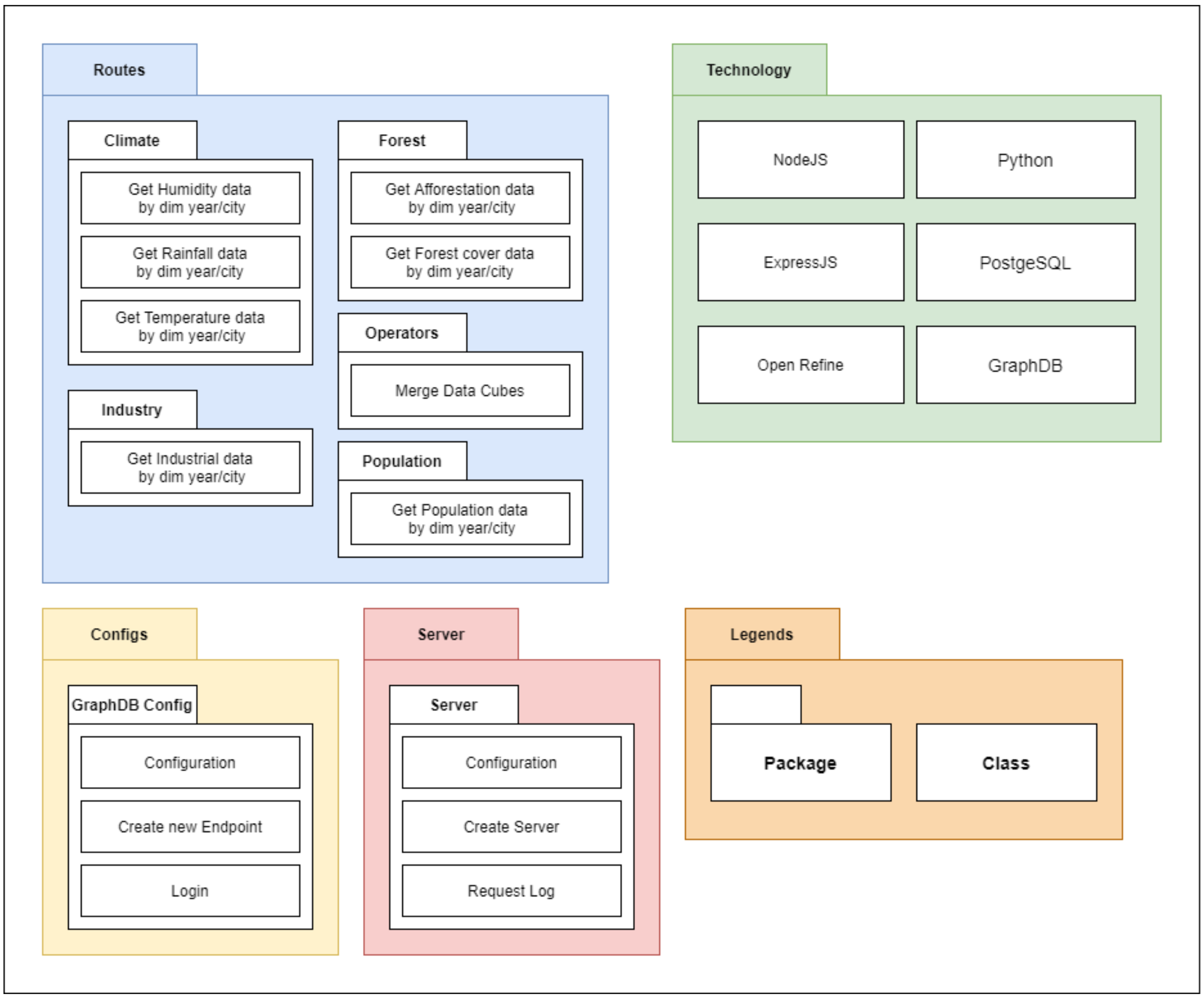

Routes
Climate
Get Humidity data by dim year/city
Get Rainfall data by dim year/city
Get Temperature data by dim year/city
Industry
Get Industrial data by dim year/city
Forest
Get Afforestation data by dim year/city
Get Forest cover data by dim year/city
Operators
Merge Data Cubes
Population
Get Population data by dim year/city
Technology
NodeJS
Python
ExpressJS
PostgeSQL
Open Refine
GraphDB
Configs
GraphDB Config
Configuration
Create new Endpoint
Login
Server
Server
Configuration
Create Server
Request Log
Legends
Package
Class

Figure 6: Module View - based on Server

Prose:

Table 6: Modules View base on Server Role & Responsibility Description

Role & Responsibility	Description
Routes	GET HTTP actions
Climate	Get data of climate data cube (humidity, rainfall, temperature values) by dimension year and city
Forest	Get data of forest data cube (afforestation area, forest cover value) by dimension year and city
Industry	Get data of industry data cube by dimension year and city

Role & Responsibility	Description
Population	Get data of population data cube by dimension year and city
Operators	Execute merging 2 or 3 data cubes for creating a new data cube
Configs	
GraphDB Config	Initiate configuration values for GraphDB, create a new SPARQL Endpoint, login to GraphDB
Server	
Server	Initiate configuration values for creating an ExpressJS server, log the requests from clients
Technology	
ExpressJS	Library for programming API
NodeJS	Library for programming API
Open Refine	A reconciliation service for registered SPARQL endpoints, a graphical user interface(GUI) for exporting data of Google Refine projects in RDF format. The export is based on mapping the data to a template graph using the GUI.
Python	Programming Language for building ETL process

2.5 Data Warehouse Architecture

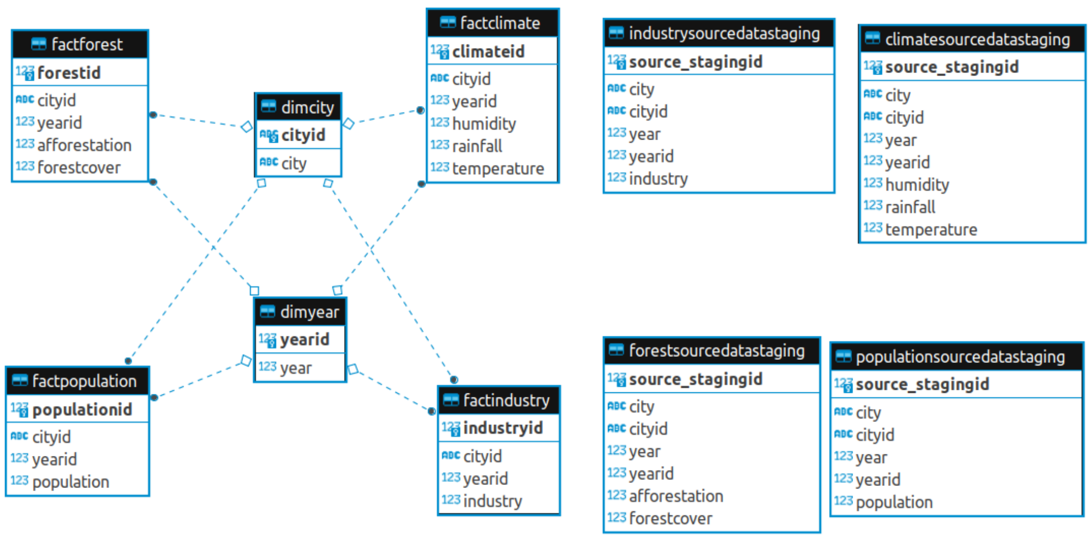


Figure 7: Data warehouse Schema

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.

Star 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.

A fact table typically has two types of columns: foreign keys to dimension tables and measures those that contain numeric facts. A fact table can contain fact data on a detail 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 a 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.

SDA database system is designed based on the data warehouse platform. In SDA 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 table, dimcity table.
- Fact tables: factclimate table, factforest table, factindustry table, factpopulation table.

2.6 RDF Data Cube Architecture

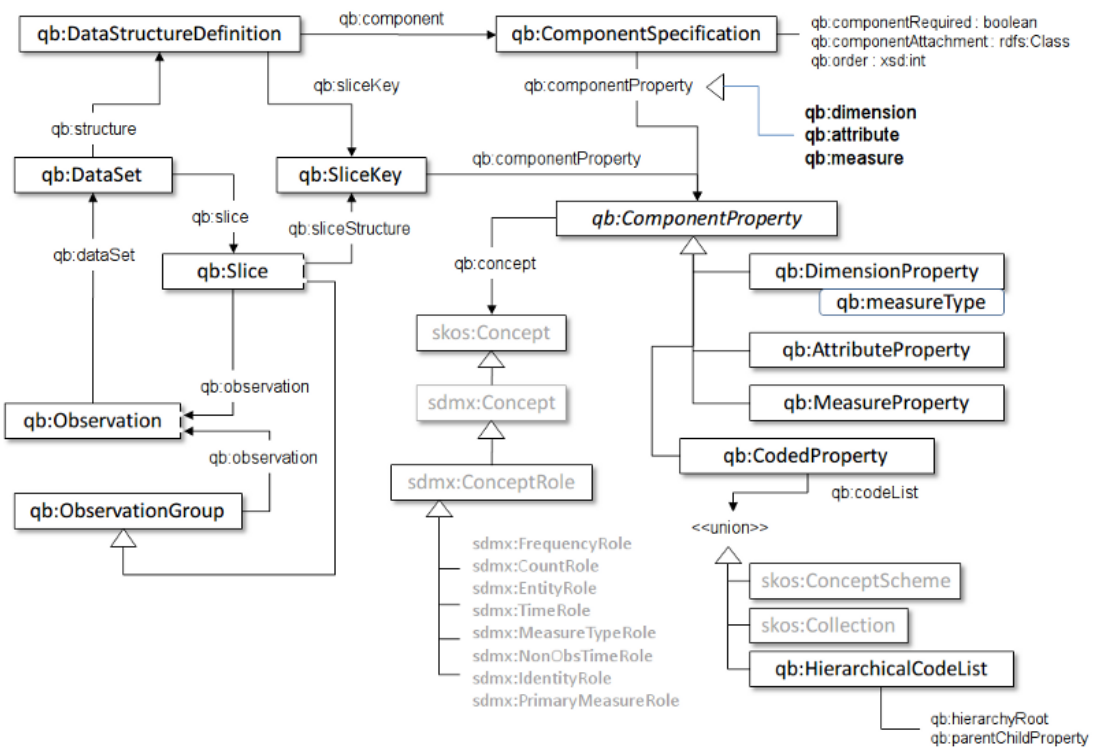


Figure 8: Pictorial summary of key terms and their relationship.

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., the 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.

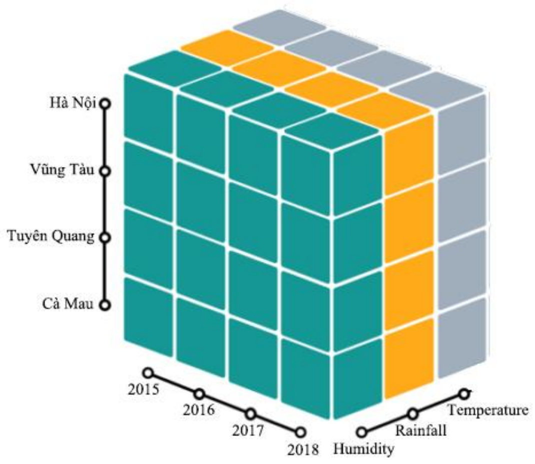


Figure 9: Climate Data Cube in 3D

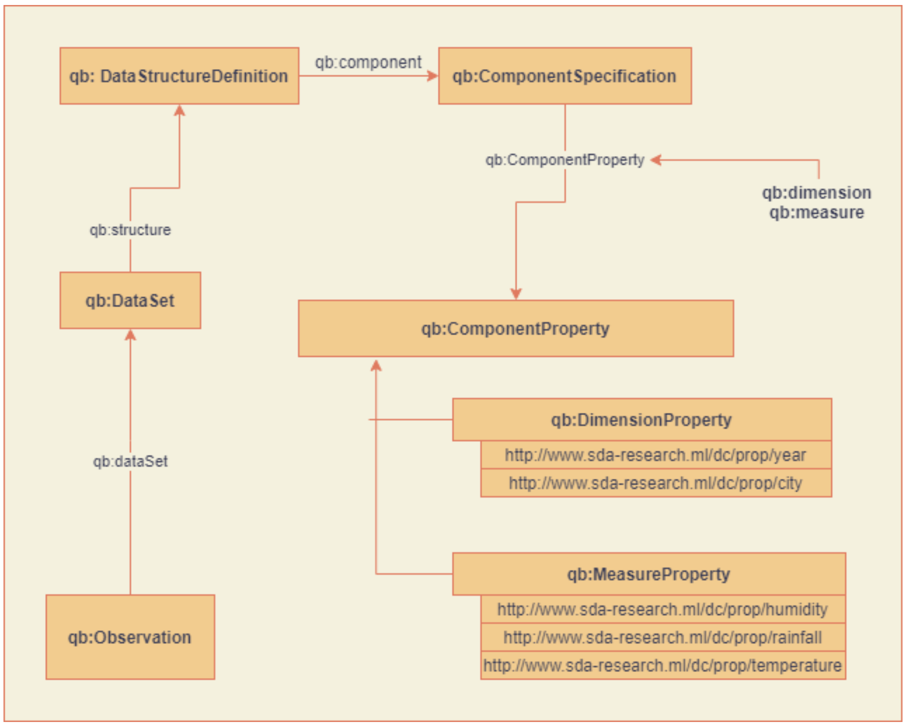

qb: DataStructureDefinition
qb:component
qb:ComponentSpecification
qb:ComponentProperty
qb:dimension
qb:measure
qb:structure
qb:DataSet
qb:ComponentProperty
qb:DimensionProperty
http://www.sda-research.ml/dc/prop/year
http://www.sda-research.ml/dc/prop/city
qb:dataSet
qb:MeasureProperty
http://www.sda-research.ml/dc/prop/humidity
http://www.sda-research.ml/dc/prop/rainfall
http://www.sda-research.ml/dc/prop/temperature
qb:Observation

Figure 10: Climate Data Cube Structure

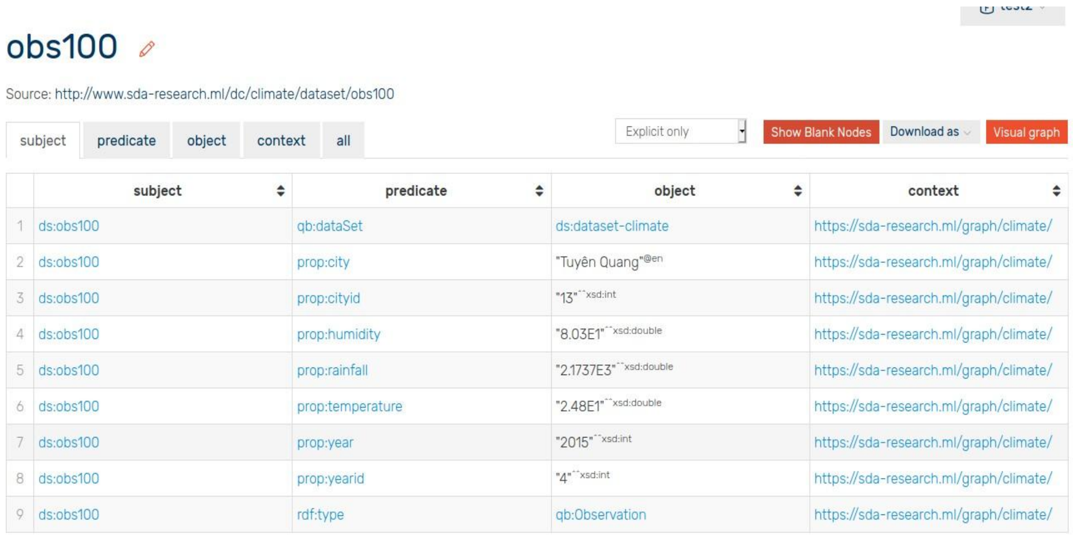
As the schema above, the Data Structure Definition has a child node ComponentSpecification, where qb:ComponentProperty is configured. The qb:ComponentProperty includes two main concepts, qb:DimensionProperty and qb:MeasureProperty - the DSD definitions for each dimension and measure in the data warehouse. Each class has a 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 needs to figure out a set of statistical data that depends on multiple categories, such as the humidity of an area/city in a specific time period (year), the data cube will refer to the value of the statistic (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.



obs100

Source: http://www.sda-research.ml/dc/climate/dataset/obs100

subject | predicate | object | context | all

	subject	predicate	object	context
1	ds:obs100	qb:dataSet	ds:dataset-climate	https://sda-research.ml/graph/climate/
2	ds:obs100	prop:city	"Tuyên Quang"@en	https://sda-research.ml/graph/climate/
3	ds:obs100	prop:cityid	"13"^^xsd:int	https://sda-research.ml/graph/climate/
4	ds:obs100	prop:humidity	"8.03E1"^^xsd:double	https://sda-research.ml/graph/climate/
5	ds:obs100	prop:rainfall	"2.1737E3"^^xsd:double	https://sda-research.ml/graph/climate/
6	ds:obs100	prop:temperature	"2.48E1"^^xsd:double	https://sda-research.ml/graph/climate/
7	ds:obs100	prop:year	"2015"^^xsd:int	https://sda-research.ml/graph/climate/
8	ds:obs100	prop:yearid	"4"^^xsd:int	https://sda-research.ml/graph/climate/
9	ds:obs100	rdf:type	qb:Observation	https://sda-research.ml/graph/climate/

Figure 11: Observation 100th.

CHAPTER 3. EXPERIMENT

3.1. Allocation view of SDA

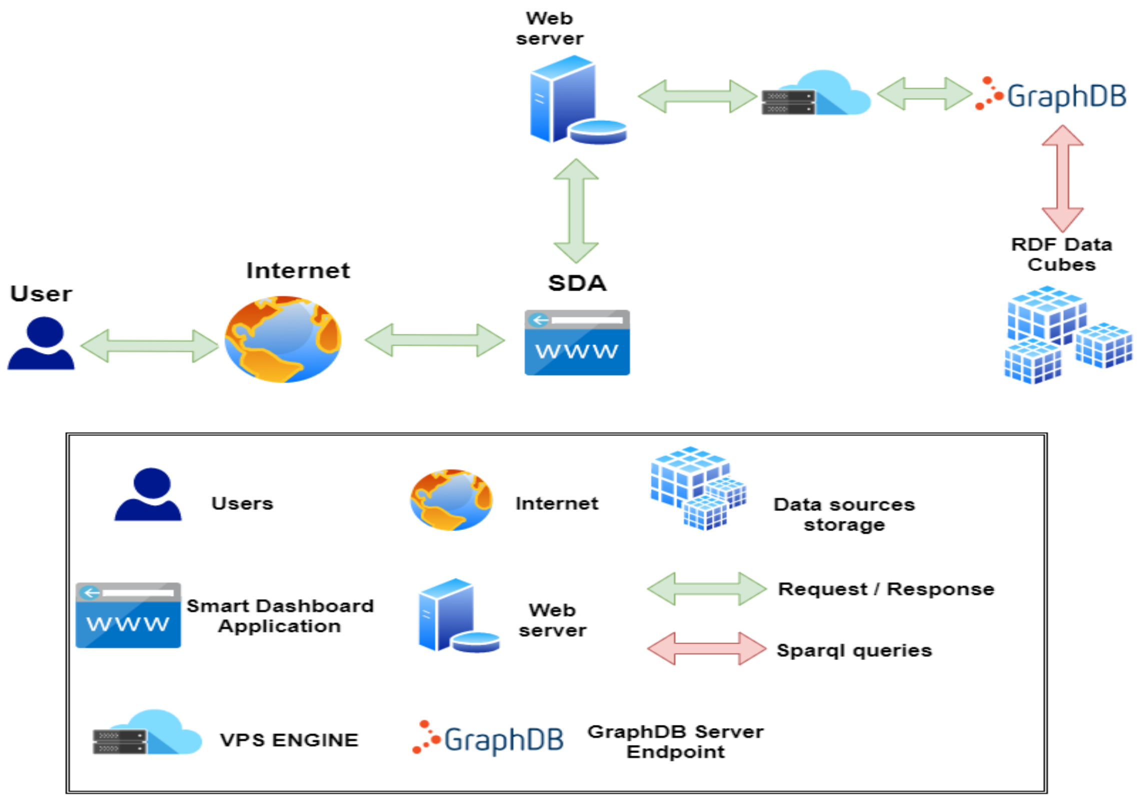


Figure 12: Allocation View

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.

Table 7: Allocation View Role & Responsibility description

Role & Responsibility	Description
Data sources storage	RDF Data Cubes is stored on GraphDB
User	The user that interact with SDA
WebServer	HTTP Server with NodeJS
Smart Dashboard Application	Our application.
VPS Engine	Store and Execute SDA'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 the client and respond the data to the client
SPARQL queries	A query language for executing the method of RDF Data Cubes

3.2. Implementation strategy

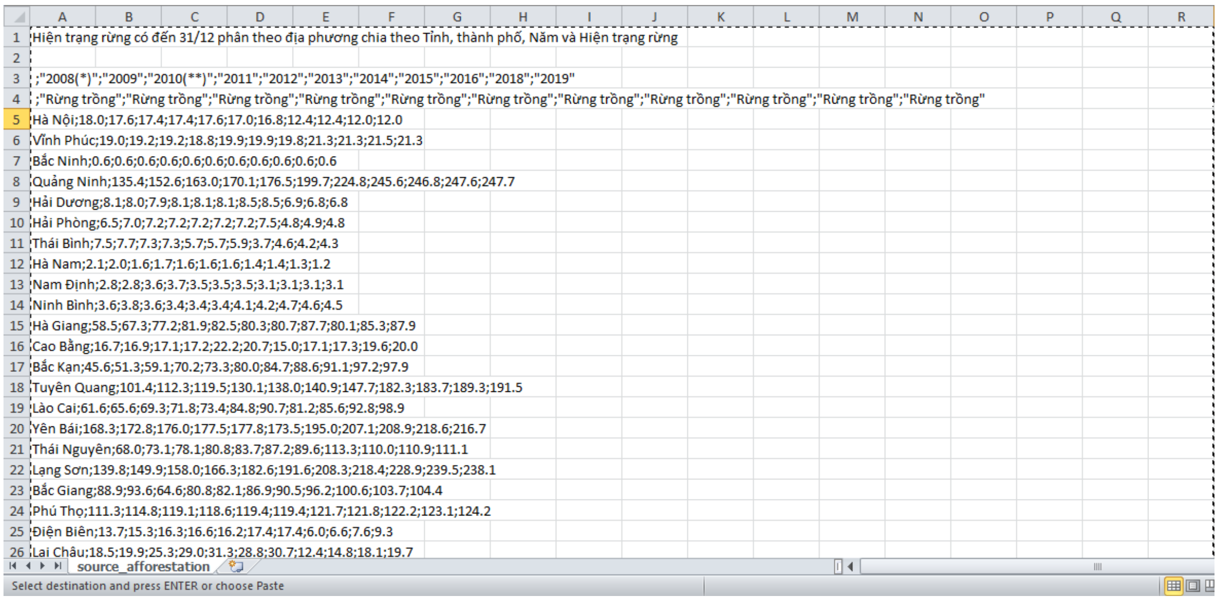
3.2.1 Building the Data Warehouse

Implementing a data warehouse consist of 4 main steps to define all these properties below:

- **Fact tables**: The fact tables contain the measures, and foreign keys which refer to primary keys in the dimension tables.
- **Dimension tables**: Dimension tables contain descriptive attributes, and these dimension tables used as a query constraining, and filtering.
- **Staging area tables**: Staging area for validation, removing constraints of the data types for data process, and data integrity.

So based on the design requirements of the data warehouse and the environment data we have found. We have built a data warehouse using Python for ETL(Extract, Transform, and Load) processes.

STEP 1: Extracting the data from CSV files, Transform the data and Load data into the Staging Area Tables.



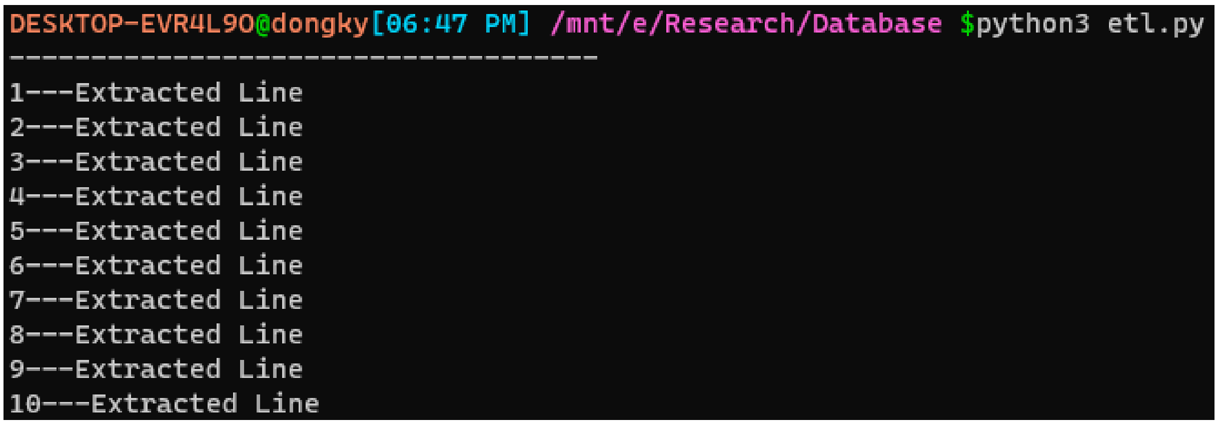
	A
1	Hiện trạng rừng có đến 31/12 phân theo địa phương chia theo Tỉnh, thành phố, Năm và Hiện trạng rừng
2	
3	;"2008(*)";"2009";"2010(**)";"2011";"2012";"2013";"2014";"2015";"2016";"2018";"2019"
4	;"Rừng trồng";"Rừng trồng";"Rừng trồng";"Rừng trồng";"Rừng trồng";"Rừng trồng";"Rừng trồng";"Rừng trồng";"Rừng trồng";"Rừng trồng";"Rừng trồng"
5	Hà Nội;18.0;17.6;17.4;17.4;17.6;17.0;16.8;12.4;12.4;12.0;12.0
6	Vĩnh Phúc;19.0;19.2;19.2;18.8;19.9;19.9;19.8;21.3;21.3;21.5;21.3
7	Bắc Ninh;0.6;0.6;0.6;0.6;0.6;0.6;0.6;0.6;0.6;0.6;0.6
8	Quảng Ninh;135.4;152.6;163.0;170.1;176.5;199.7;224.8;245.6;246.8;247.6;247.7
9	Hải Dương;8.1;8.0;7.9;8.1;8.1;8.1;8.5;8.5;6.9;6.8;6.8
10	Hải Phòng;6.5;7.0;7.2;7.2;7.2;7.2;7.2;7.5;4.8;4.9;4.8
11	Thái Bình;7.5;7.7;7.3;7.3;5.7;5.7;5.9;3.7;4.6;4.2;4.3
12	Hà Nam;2.1;2.0;1.6;1.7;1.6;1.6;1.6;1.4;1.4;1.3;1.2
13	Nam Định;2.8;2.8;3.6;3.7;3.5;3.5;3.5;3.1;3.1;3.1;3.1
14	Ninh Bình;3.6;3.8;3.6;3.4;3.4;3.4;4.1;4.2;4.7;4.6;4.5
15	Hà Giang;58.5;67.3;77.2;81.9;82.5;80.3;80.7;87.7;80.1;85.3;87.9
16	Cao Bằng;16.7;16.9;17.1;17.2;22.2;20.7;15.0;17.1;17.3;19.6;20.0
17	Bắc Kạn;45.6;51.3;59.1;70.2;73.3;80.0;84.7;88.6;91.1;97.2;97.9
18	Tuyên Quang;101.4;112.3;119.5;130.1;138.0;140.9;147.7;182.3;183.7;189.3;191.5
19	Lào Cai;61.6;65.6;69.3;71.8;73.4;84.8;90.7;81.2;85.6;92.8;98.9
20	Yên Bái;168.3;172.8;176.0;177.5;177.8;173.5;195.0;207.1;208.9;218.6;216.7
21	Thái Nguyên;68.0;73.1;78.1;80.8;83.7;87.2;89.6;113.3;110.0;110.9;111.1
22	Lạng Sơn;139.8;149.9;158.0;166.3;182.6;191.6;208.3;218.4;228.9;239.5;238.1
23	Bắc Giang;88.9;93.6;64.6;80.8;82.1;86.9;90.5;96.2;100.6;103.7;104.4
24	Phú Thọ;111.3;114.8;119.1;118.6;119.4;119.4;121.7;121.8;122.2;123.1;124.2
25	Điện Biên;13.7;15.3;16.3;16.6;16.2;17.4;17.4;6.0;6.6;7.6;9.3
26	Lai Châu;18.5;19.9;25.3;29.0;31.3;28.8;30.7;12.4;14.8;18.1;19.7

Figure 13: Example of the data source before ETL

STEP 2: Distinct value cities, and years from staging areas tables into dimension tables.

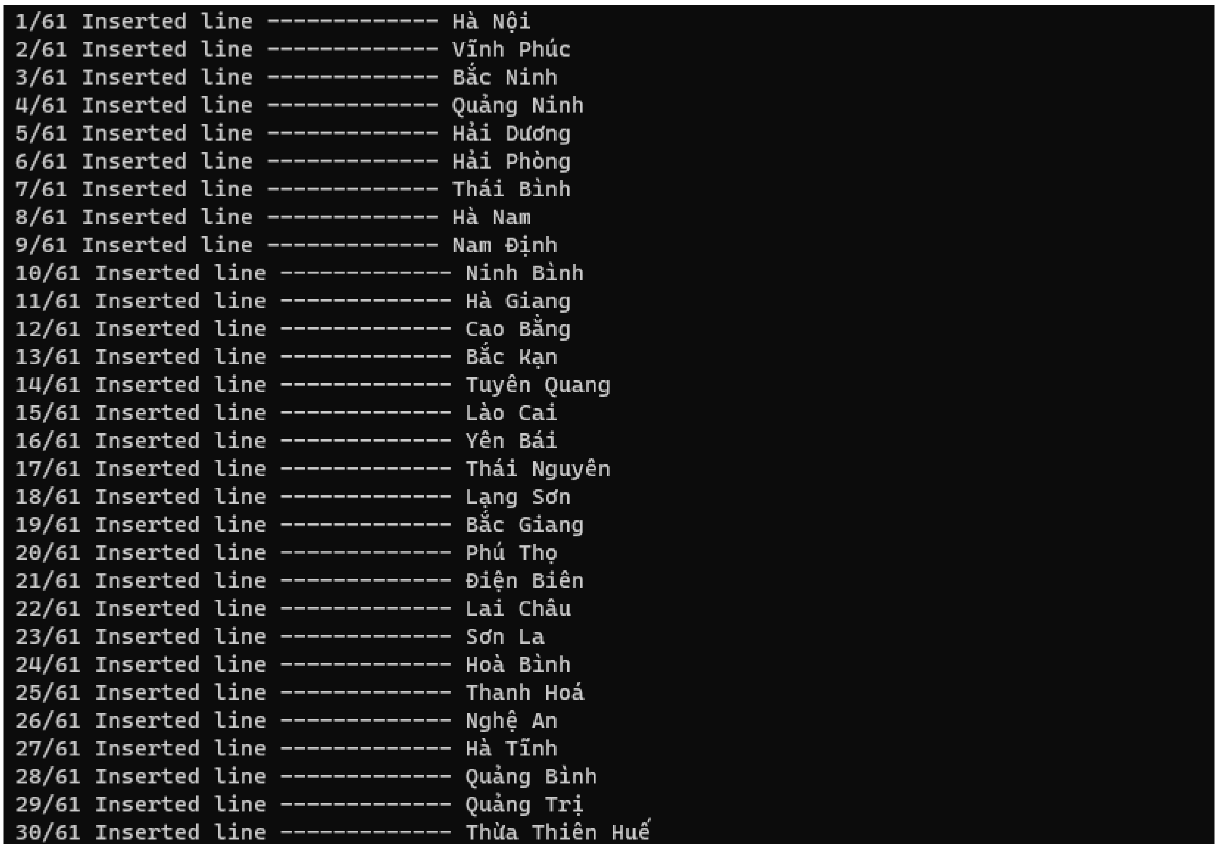
STEP 3: After filling all the data in all the dimension tables then ETL data into empty columns(yearid, and cityid) in staging areas .

STEP 4: Loading all the available columns of staging area tables into fact tables.



```
DESKTOP-EVR4L90@dongky[06:47 PM] /mnt/e/Research/Database $python3 etl.py
------------------------------------
1---Extracted Line
2---Extracted Line
3---Extracted Line
4---Extracted Line
5---Extracted Line
6---Extracted Line
7---Extracted Line
8---Extracted Line
9---Extracted Line
10---Extracted Line
```

Figure 14: Running ETL process



```
1/61 Inserted line ------------- Hà Nội
2/61 Inserted line ------------- Vĩnh Phúc
3/61 Inserted line ------------- Bắc Ninh
4/61 Inserted line ------------- Quảng Ninh
5/61 Inserted line ------------- Hải Dương
6/61 Inserted line ------------- Hải Phòng
7/61 Inserted line ------------- Thái Bình
8/61 Inserted line ------------- Hà Nam
9/61 Inserted line ------------- Nam Định
10/61 Inserted line ------------- Ninh Bình
11/61 Inserted line ------------- Hà Giang
12/61 Inserted line ------------- Cao Bằng
13/61 Inserted line ------------- Bắc Kạn
14/61 Inserted line ------------- Tuyên Quang
15/61 Inserted line ------------- Lào Cai
16/61 Inserted line ------------- Yên Bái
17/61 Inserted line ------------- Thái Nguyên
18/61 Inserted line ------------- Lạng Sơn
19/61 Inserted line ------------- Bắc Giang
20/61 Inserted line ------------- Phú Thọ
21/61 Inserted line ------------- Điện Biên
22/61 Inserted line ------------- Lai Châu
23/61 Inserted line ------------- Sơn La
24/61 Inserted line ------------- Hoà Bình
25/61 Inserted line ------------- Thanh Hoá
26/61 Inserted line ------------- Nghệ An
27/61 Inserted line ------------- Hà Tĩnh
28/61 Inserted line ------------- Quảng Bình
29/61 Inserted line ------------- Quảng Trị
30/61 Inserted line ------------- Thừa Thiên Huế
```

Figure 15: Running ETL process

Basically, in this phase so far we have just manually done the ETL processes, not the automated way. In the future we focus on data resources to enrich more data like crawling data, gathering data automatically, automating the ETL processes.

Currently, The data is complicated to ETL. The data we collected was really messy for the ETL process. Additionally, with the requirements we added into the Data warehouse we have faced the many problems with the unhandling data which means the data is hard to find, process, and especially for automating ETL processes.

3.2.2 Linking Datasets

To able to linking between the data cubes, they have to designed based on following principles:

Dimensions

Table 8: Dimensions description

Column	Description
city	The area belongs to dataset
cityid	ID name transformed from city data
year	The time period belongs to dataset

Measures

Table 9: Measures description

Column	Description
humidity	Observed humidity value
rainfall	Observed rainfall value
temperature	Observed temperature value
forestarea	Observed forest area value
forestcover	Observed forest cover value
deforestation	Observed deforestation value
naturalforestarea	Observed natural forest area value
industry	Observed industrial value
population	Observed population value

Structure, Patterns, and Local Prefixes

Climate Data Cube

Table 10. Structure, Patterns, Prefixes

Item **[prefix]** {pattern} Description	**Value for Project**
Cube Name (Dataset name)	climate
BaseURI	http://sda-research.ml/
Data Cube {BaseURI}dc/{cube name}	http://sda-research.ml/dc/climate
DataSet [ds] {BaseURI}dc/{cube name}/dataset Includes the qb:DataSet, the qb:DataStructureDefinition and the qb:Observation. The values of each dimension (specified as the value of the cube property in each dimension as part of qb:Observation) are also placed here because they are values that are a part of the cube. This would change if codelists are used. Slices [qb:Slice, qb:SliceKey] would also be included here, if used.	http://sda-research.ml/dc/climate/dataset
Properties [prop] {BaseURI}dc/{cube name}/prop/ Properties of the Data Cube. a) qb:ComponentProperty, qb:DimensionProperty, qb:MeasureProperty, qb:AttributeProperty, qb:CodedProperty b) qb:component defined under each Data	http://sda-research.ml/dc/climate/prop/
Cube Component Specifications [dccs] {BaseURI}dc/{cube name}/dccs/	http://sda-research.ml/dc/climate/dccs/

Cube Component specifications. qb:ComponentSpecification	

Table 11. URI's for Dimensions and Measures

Component Pattern	Value for Project
dimension	1. http://sda-research.ml/dc/climate/prop/city 2. http://sda-research.ml/dc/climate/prop/cityid 3. http://sda-research.ml/dc/climate/prop/year
measure	http://sda-research.ml/dc/climate/prop/humidity http://sda-research.ml/dc/climate/prop/rainfall http://sda-research.ml/dc/climate/prop/temperature

Industry Data Cube

Table 12. Structure, Patterns, Prefixes

Item **[prefix]** {pattern} Description	**Value for Project**
Cube Name (Dataset name)	Industry
BaseURI	http://sda-research.ml/
Data Cube {BaseURI}dc/{cube name}	http://sda-research.ml/dc/industry
DataSet [ds] {BaseURI}dc/{cube name}/dataset Includes the qb:DataSet, the qb:DataStructureDefinition and the qb:Observation. The values of each dimension (specified as the value of the cube property in each dimension as part of qb:Observation) are also placed here because they are values that are a part of the cube. This would	http://sda-research.ml/dc/industry/dataset

change if codelists are used. Slices [qb:Slice, qb:SliceKey] would also be included here, if used.	
Item **[prefix]** {pattern} Description	**Value for Project**
Properties [prop] {BaseURI}dc/{cube name}/prop/ **Properties of the Data Cube.** a) qb:ComponentProperty, qb:DimensionProperty, qb:MeasureProperty, qb:AttributeProperty, qb:CodedProperty b) qb:component defined under each Data	http://sda-research.ml/dc/industry/prop/
Cube Component Specifications [dccs] {BaseURI}dc/{cube name}/dccs/ Cube Component specifications. qb:ComponentSpecification	http://sda-research.ml/dc/industry/dccs/

Table 13. URI's for Dimensions and Measures

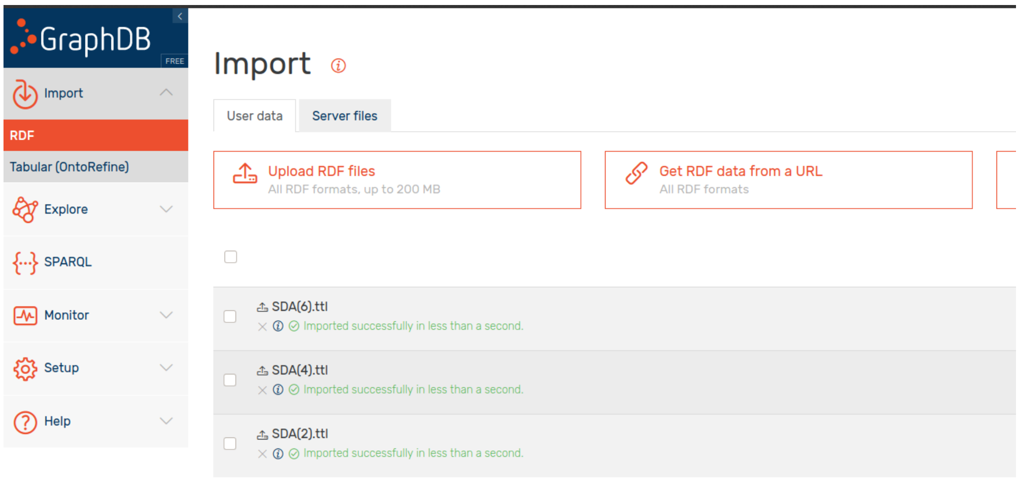
Component Pattern	Value for Project
dimension	1. http://sda-research.ml/dc/industry/prop/city 2. http://sda-research.ml/dc/industry/prop/cityid 3. http://sda-research.ml/dc/industry/prop/year
measure	1.http://sda-research.ml/dc/industry/prop/humidity

External vocabularies

Table 14 External Vocabularies

Prefix	URI	Comment
qb	http://purl.org/linked-data/cube#	Cube spec.
rdfs	http://www.w3.org/2000/01/rdf-schema#	Labels, comments
xsd	http://www.w3.org/2001/XMLSchema#	Data types
dcat	http://www.w3.org/ns/dcat#	Distribution information
dct	http://purl.org/dc/terms/	Creator, issued date, title, description...
prov	http://www.w3.org/ns/prov#	Provenance

- After the data cubes are generated, we upload them to GraphDB Endpoint.



SDA(6).ttl
Imported successfully in less than a second.

SDA(4).ttl
Imported successfully in less than a second.

SDA(2).ttl
Imported successfully in less than a second.

Figure 16: GraphDB Endpoint

- We write the APIs of SDA 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.
- The data cubes with same dimensions and properties are linked based on SPARQL query in GraphDB Cloud Endpoint. There are variety linking ways we provide on the server APIs that have listed below:
- We have osme

GET

/merge/dc1/:dc1/dc2/:dc2/s1/:s1/s2/:s2/city/:cityId/fYear/:fYear/tYear/:tYear

Example:

/merge/dc1/dcclimate/dc2/dcclimate/s1/rainfall/s2/temperature/city/danang/fyear/2012/tyear/2018

GET

/merge/dc1/:dc1/dc2/:dc2/dc3/:dc3/s1/:s1/s2/:s2/s3:/s3/city/:cityId/fYear/:fYear/tYear/:tYear

Example:

/merge/dc1/dcclimate/dc2/dcindustry/dc3/dcforest/s1/rainfall/s2/industry/s3/forestcover/city/danang/fyear/2012/tyear/2018

3.2.3 Visualizing

Technologies Stack:

- React *v16.13.1*
- Redux / @reduxjs/toolkit *v1.4.0*
- highcharts *v8.2.2* / highcharts-react-official *v3.0.0*
- beautiful-react-diagrams *v0.5.1*
- react-dnd *v11.1.3*
- axios *v0.21.0*
- @material-ui *v4.11.0*
- styled-components *v5.2.0*

Installation Guide

Clone this project

- *git clone https://github.com/sdateamdtu2020/sda-client*

Install dependencies

- *npm install*

Start the server (project is run on port 3000: **localhost:3000**):

- *npm start*

Plan of Action

- Initial Project
- Setup **React** & Install dependencies

- Setup Router
- create **Navbar** component
- create **InfoSection** component
- Add data to homepage
- Create **Navbar Dashboard**
- Implement **Redux toolkit** to app
- Update Navbar Dashboard
- Import Material-UI
- create Grid layout for **Dashboard**
- create **Toolbar** component
- Initial **Content** component
- Setup **draw diagrams** with react-beautiful-diagrams
- Refactor Toolbar component with Redux
- create function **addNewNode** & **removeNode**
- create First layout of Node
- Viz **Climate/Humidity** in TreeView
- Initial **LineChart** & **ColumnChart**
- create **Info** container
- create **Properties** component
- create **Widget Infos** component
- create **Output Log** component
- Initial **Maps**
- **Fetch API** from server (http://server-sda-research.ml)
- Viz **Industry** in year 2012 in Maps
- Viz **Humidity** in LineChart
- Viz **Humidity** in ColumnChart
- Draw connector when onClick on RUN btn
- Clear all node when onClick on NEW btn
- Visualization **Humidity** (year, city, period of city)
- Visualization **Temperature** (year, city, period of city)
- Visualization **Rainfall** (year, city, period of city)
- Visualization **Industry** (year, city, period of city)
- Visualization **Forest Cover Area** (year, city, period of city)
- Visualization **Afforestation** (year, city, period of city)
- Visualization **Population** (year, city, period of city)
- **Merge** two dimensions
- Visualization in **multi-YAxis-Chart** when merge two dimensions
- **Merge** three dimensions
- Visualization in **multi-YAxis- Chart** when merge three dimensions

- **Deploy** project to Firebase with HTTPS (https://sda-research.ml)
- **Deploy** project to surge.sh with HTTP (http://sda-research.surge.sh)

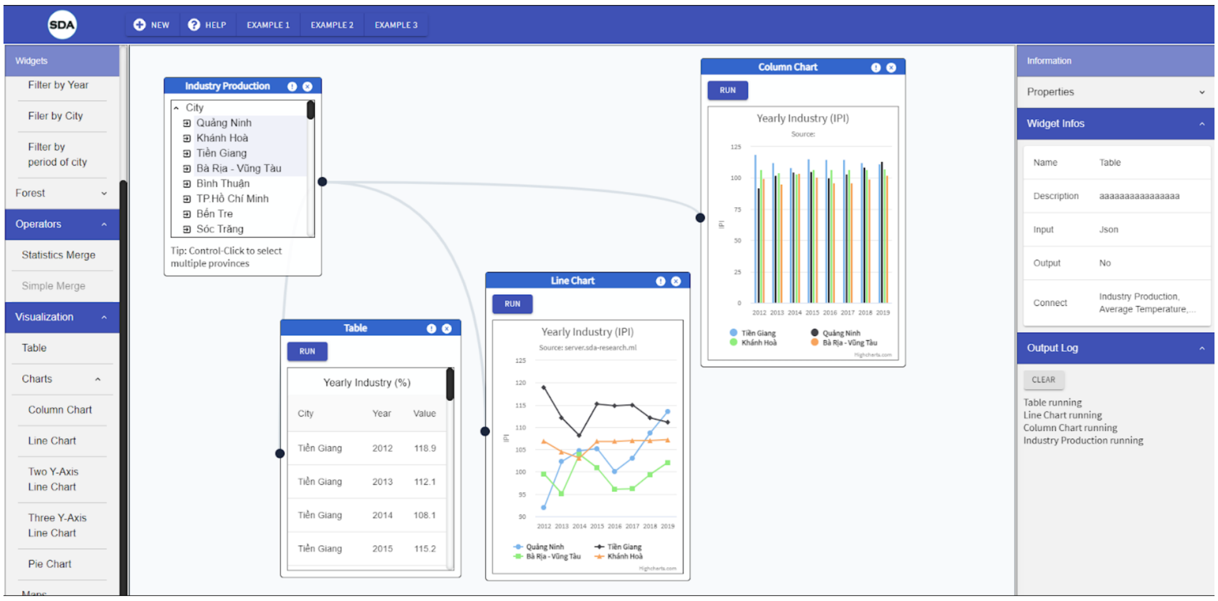
3.2.4 Discussion

With the innovative use of data technologies, Smart Dashboard Application has basically solved the problem that we initially posed: how to create a Vietnam environmental data system with sufficient information but still user-friendly. This is very significant with the trend of digitizing decision-making tools of experts as well as lawmakers in Vietnam today. However, during the limited study period, SDA still had many limitations, for example the lack of data from some provinces in some data areas, some procedures still being carried out. Manually, the methods of aggregating data blocks are still not really flexible. In the future, SDA will continue to be developed and we will try to thoroughly overcome the limitations and further improve the functions that are most useful for users.

3.2.5 Example work

There are screenshots of our web-application with examples:

- Visualize **Industry** in Line/Column Chart, and Table
- Visualize **Humidity** by Cities in Line/Column Chart and by Year in Table, Maps
- Visualize **Industry** by Cities in year on Column Chart, and Maps
- Merge Two Dimensions: **Humidity, Temperature** by city in years in Two Y-Axis Line Chart
- Merge Three Dimensions: **Rainfall, Humidity, Temperature** by city in years in Three Y-Axis Chart



Table

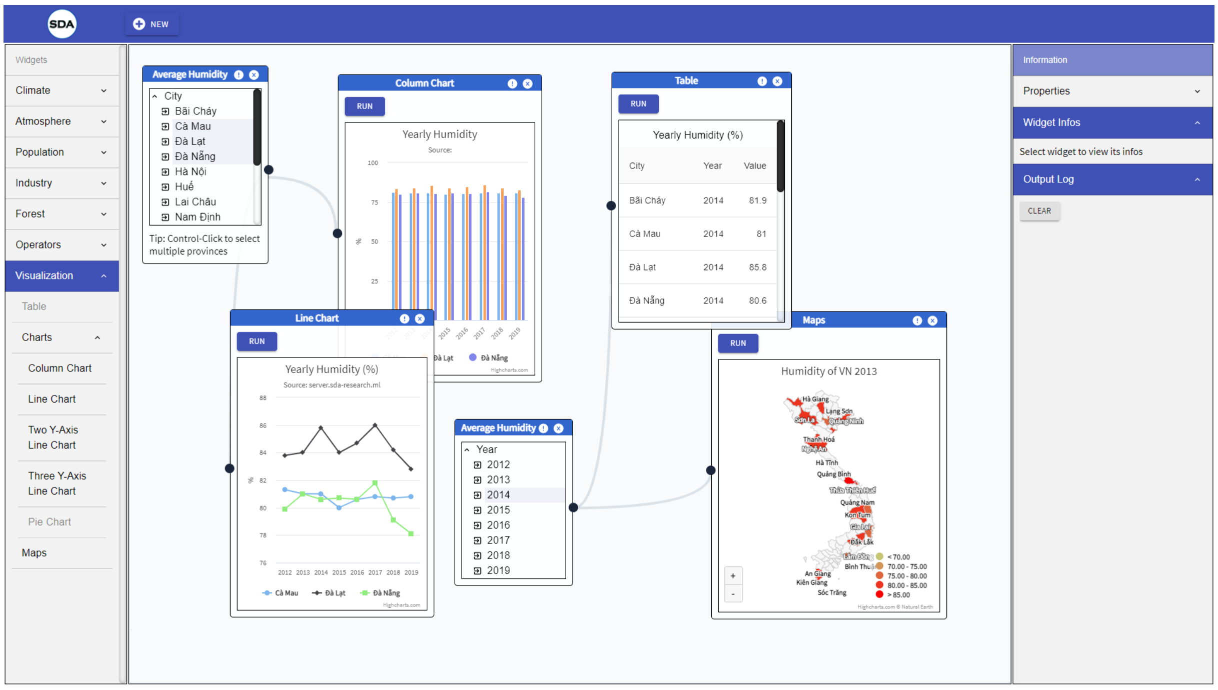
Yearly Industry (%)

City	Year	Value
Tiền Giang	2012	118.9
Tiền Giang	2013	112.1
Tiền Giang	2014	108.1
Tiền Giang	2015	115.2

Name	Table
Description	aaaaaaaaaaaaaaaa
Input	Json
Output	No
Connect	Industry Production, Average Temperature,...

```
Table running
Line Chart running
Column Chart running
Industry Production running
```

Figure 17: Visualize Industry Production in Line/Column Chart, and Table



Table

Yearly Humidity (%)

City	Year	Value
Bãi Cháy	2014	81.9
Cà Mau	2014	81
Đà Lat	2014	85.8
Đà Nẵng	2014	80.6

Select widget to view its infos

Figure 18: Visualize Humidity by Cities in Line/Column Chart and by Year in Table, Maps

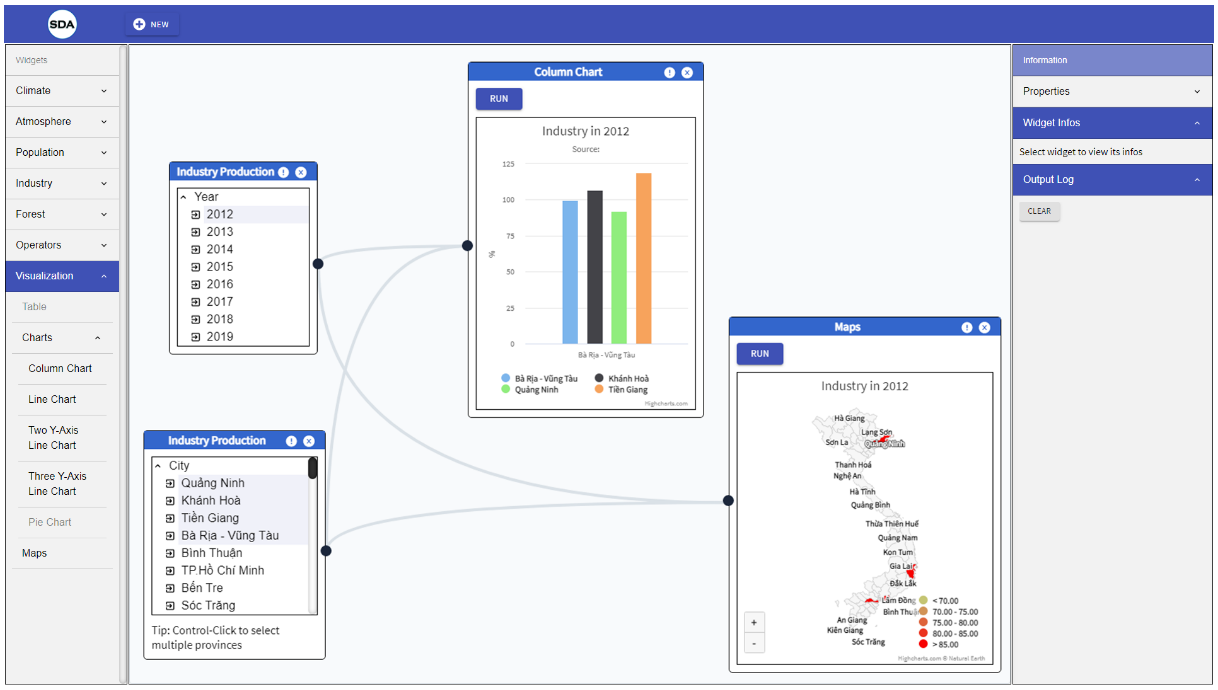


Figure 19: Visualize Industry by Cities in year on Column Chart, and Maps

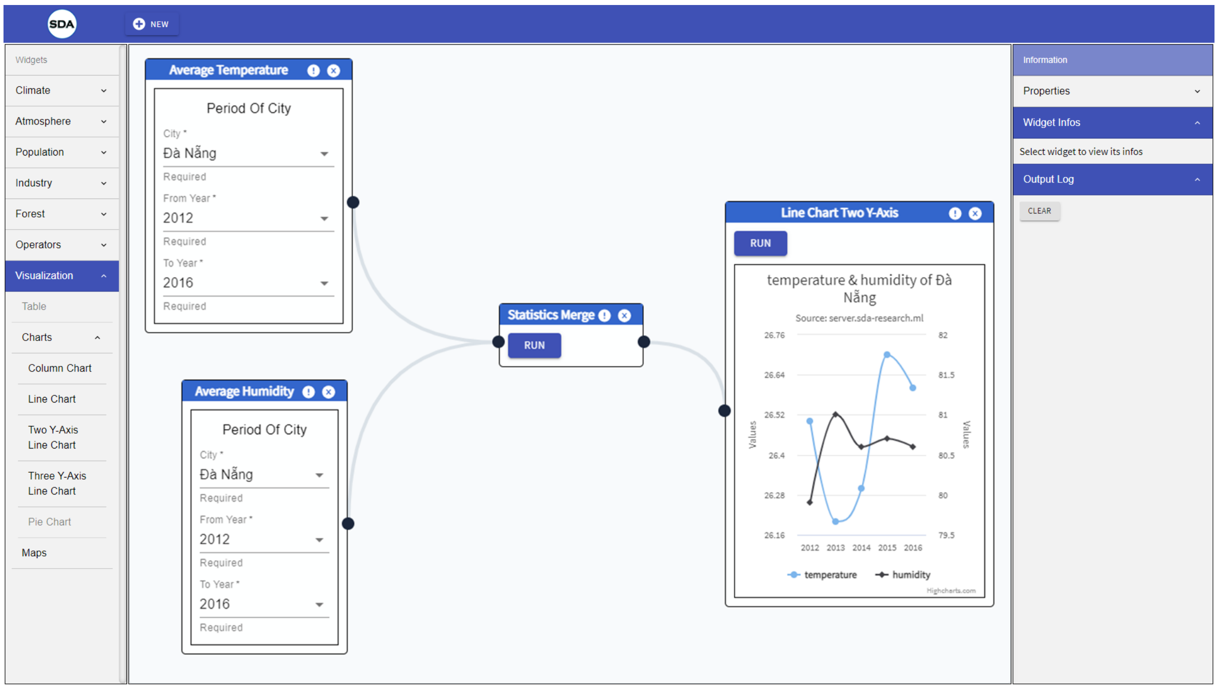


Figure 20: Merge Two Dimensions: Humidity, Temperature by city in years in Two Y-Axis Line Chart

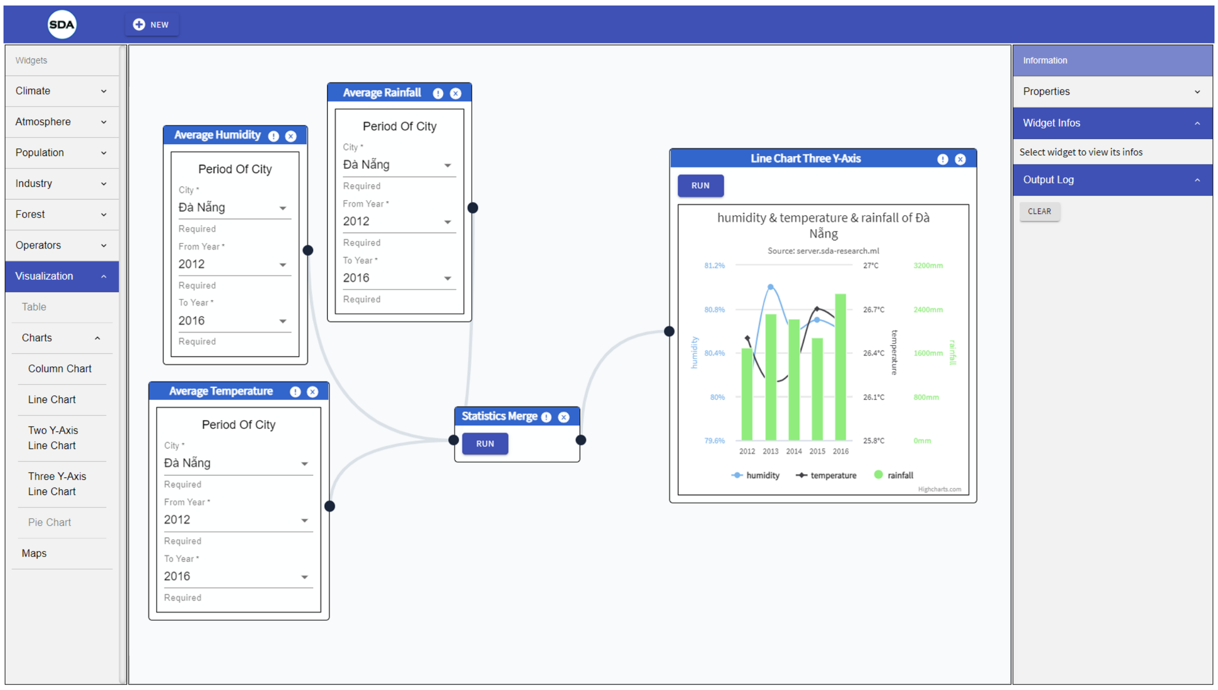


Figure 21: Merge Three Dimensions: Rainfall, Humidity, Temperature by city in years in Three Y-Axis Chart

CONCLUSION

Smart Dashboard Application promises to be a convenient and useful tool for not only environmental experts and policymakers but also those who love to protect the environment. The confusion about a large data system and reviewing it in a general way to come up with precise planning will no longer be a problem, even for those who don't have technology expertise. SDA will contribute to help our society to accurately and effectively fight against environmental damage and global warming.

REFERENCE

Client with HTTPS: https://sda-research.ml/dashboard

Client with HTTP: http://sda-research.surge.sh/dashboard

Server Endpoint: http://server.sda-research.ml

Draw.io: https://www.draw.io

ETL - Extract, Transform, Load: https://www.webopedia.com/TERM/E/ETL.html

What is OLAP: https://www.ibm.com/cloud/learn/olap

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 Warehouse Architecture: Traditional vs. Cloud: https://panoply.io/data-warehouse-guide/data-warehouse-architecture-traditional-vs-cloud/

React Redux — Concept, Workflow & Cheatsheet: https://medium.com/@javascript_7596/react-redux-concept-workflow-cheatsheet-be00e3ffa853

Interactive Dashboards: Using Visual Analytics for knowledge Transfer and Decision Support: https://www.researchgate.net/publication/299633100_Interactive_Dashboards_Using_Visual_Analytics_for_knowledge_Transfer_and_Decision_Support

Dashboard technology based solution to decision making : https://www.researchgate.net/publication/277140671_DASHBOARD_TECHNOLOGY_BASED_SOLUTION_TO_DECISION_MAKING