## 

| **International School**  **CAPSTONE PROJECT 1**  **CMU-SE-450 / CMU-IS-450 / CMU-CS-450**  DATABASE DESIGN DOCUMENT  Version 1.2  Date: 12 - Aug - 2020  **SMART DASHBOARD** APPLICATION  Submitted by  Vo Van Hoa  Pham Van Tin  Ky Huu Dong  Tran Thanh Kieu  **Approved by**  **Capstone Project 1 - Mentor:**  Name Signature Date  Binh, Thanh Nguyen \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_14 - Dec- 2020    Name Signature Date  Huy, Truong Dinh \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
| --- |

| **PROJECT INFORMATION** | | | |
| --- | --- | --- | --- |
| **Project Acronym** | SDA | | |
| **Project Title** | Smart Dashboard Application | | |
| **Project Web URL** | <https://sda-research.ml/> | | |
| **Start Date** | 12 - Aug - 2020 | | |
| **End Date:** | 15 - Dec - 2020 | | |
| **Lead Institution** | International School, Duy Tan University | | |
| **Project Mentor** | PhD Binh, Nguyen Thanh; MSc Huy, Truong Dinh | | |
| **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 Thanh | thanhkieutran391@gmail.com | 0358.583.251 |

| **DOCUMENT INFORMATION** | | | |
| --- | --- | --- | --- |
| **Document Title** | Database Design Document | | |
| **Author(s)** | Team C1SE.06 | | |
| **Role** | [SDA] Database\_Design\_v1.2 | | |
| **Date** | 21 - Nov - 2020 | File name | [SDA] Database\_Design\_v1.2 |
| **URL** | <https://github.com/sdateamdtu2020/sda-documents> | | |
| **Access** | Project and CMU Program | | |

## 

## REVISION HISTORY

| **Version** | **Person(s)** | **Date** | **Description** | **Approval** |
| --- | --- | --- | --- | --- |
| Draft | Hoa, Vo | 12 - Aug - 2020 | Initiate document | x |
| 1.0 | All members | 14 - Nov - 2020 | Finish content of document | x |
| 1.1 | Hoa, Kieu, Dong | 16 - Nov - 2020 | Update content of Physical Design | x |
| 1.2 | Hoa, Dong | 21 - Nov - 2020 | Update RDF Data Cubes Design & Entity Mapping |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

## 

# 

# TABLE OF CONTENTS

PROJECT INFORMATION 1

DOCUMENT INFORMATION 1

REVISION HISTORY 2

**TABLE OF CONTENTS 3**

1. **INTRODUCTION 5**
   1. PURPOSE 5
   2. DOCUMENT OBJECTIVES 5
   3. INTENDED AUDIENCE 5
   4. SCOPE, APPROACH AND METHODS 6
   5. SYSTEM OVERVIEW 6
   6. ACRONYMS AND ABBREVIATIONS 6
   7. KEY PERSONNEL 6
2. **ASSUMPTIONS, CONSTRAINTS AND DEPENDENCIES 7**
   1. ASSUMPTIONS 7
   2. CONSTRAINTS 7
3. **SYSTEM OVERVIEW 7**
   1. DATABASE MANAGEMENT SYSTEM CONFIGURATION 7
   2. DATABASE SOFTWARE UTILITIES 7
   3. SUPPORT SOFTWARE 7
4. **ARCHITECTURE 8**
   1. HARDWARE & SOFTWARE ARCHITECTURE 8
   2. DATASTORES 8
5. **DATABASE-WIDE DESIGN DECISIONS 9**
   1. KEY FACTORS INFLUENCING DESIGN 9
   2. PERFORMANCE AND AVAILABILITY DECISIONS 9
6. **DATABASE ADMINISTRATIVE FUNCTIONS 9**
   1. RESPONSIBILITY 9
   2. APPLICATIONS/SYSTEMS USING THE DATABASE 10
   3. PHYSICAL DESIGN 10
      1. FACT-INDUSTRY 10
      2. FACT-POPULATION 10
      3. FACT-CLIMATE 11
      4. FACT-FOREST 12
      5. DIMCITY 12
      6. DIMYEAR 13
      7. FOREST-SOURCE-DATA-STAGING 13
      8. CLIMATE-SOURCE-DATA-STAGING 13
      9. POPULATION-SOURCE-DATA-STAGING 14
      10. INDUSTRY-SOURCE-DATA-STAGING 14
   4. RDF DATA CUBES DESIGN 15
      1. DIMENSION 15
      2. MEASURE 15
      3. STRUCTURE, PATTERNS AND LOCAL PREFIXES 16
      4. EXTERNAL VOCABULARIES 19
      5. CLASS HIERARCHY 20
      6. CLASS RELATIONSHIPS 21
   5. ENTITY MAPPING 21
      1. ENTITY MAPPING DIAGRAM 21
      2. INDUSTRY 22
      3. CLIMATE 22
      4. FOREST 23
      5. POPULATION 23
7. **REFERENCES 24**

# INTRODUCTION

The Database Design maps the logical data model to the target database management system with consideration to the system’s performance requirements. The Database Design converts logical or conceptual data constructs to physical data constructs (e.g tables,...) of the target Database Management System.

## PURPOSE

The purpose of the Database Design is to ensure that every database transaction meets or exceeds its performance requirements. This document takes into account data and transaction volume to produce a schema and environment that will meet necessary performance

## DOCUMENT OBJECTIVES

The Database Design Document has the following objectives:

* To describe the design of a database, that is, a collection of related data stored in one or more computerized files that can be accessed by users or developers via a DBMS
* To serve as a basis for implementing the database and related software units. It provides the acquirer visibility into the design and provides information necessary for software development.

## INTENDED AUDIENCE

This document is intended for the following audiences:

Technical developers, who must evaluate the quality of this document

Developer including:

* Architects, whose overall architecture design must meet the requirements specified in this document.
* Designers, whose design must meet the requirements specified in this document.
* Developers, whose software must implement the requirements specified in this document.
* Quality Assurance personnel, whose test cases must validate the requirements specified in this document.

## SCOPE, APPROACH AND METHODS

The Database Design for the SDA is composed of definitions for database objects derived by mapping entities to tables, attributes to columns, unique identifiers to unique keys and relationships to foreign keys.

## SYSTEM OVERVIEW

| **System Overview** | **Details** |
| --- | --- |
| **System Name** | SMART DASHBOARD APPLICATION |
| **System type** | Web application |
| **Operational status** | In development |

## ACRONYMS AND ABBREVIATIONS

| **Acronym/Abbreviation** | **Meaning** |
| --- | --- |
| SDA | **S**MART **D**ASHBOARD **A**PPLICATION |
| PG | **P**ost**g**res |
| RDMS | **R**elational **D**atabase **M**anagement **S**ystem |
| DBMS | **D**ata**b**ase **M**anagement **S**ystem |
| RDC | **R**DF **D**ata **C**ube |

## KEY PERSONNEL

| **Full Name** | **Email** | **Phone number** | **Role** |
| --- | --- | --- | --- |
| Hoa, Vo | hoavo.dng@gmail.com | 0935.193.182 | Scrum master |
| Tin, Pham Van | tinphamvan123@gmail.com | 0932.535.175 | Team member |
| Dong, Ky Huu | kyhuudong@gmail.com | 0898.246.980 | Team member |
| Kieu, Tran Thanh Thi | thanhkieutran391@gmail.com | 0358.583.251 | Team member |

# ASSUMPTIONS, CONSTRAINTS AND DEPENDENCIES

## ASSUMPTIONS

* Users can drag the widgets to generate charts, map, or any available solution on GUI.
* Users can link nodes to the others nodes to generate charts based on their needs.

## CONSTRAINTS

* Users can only use all the functions available on GUI.

# SYSTEM OVERVIEW

## DATABASE MANAGEMENT SYSTEM CONFIGURATION

* **System**: GraphDB
* **Vendor**: Ontotext
* **Services enabled**:
  + RDF Data storage.
* **System**: Linux VM
* **Vendor**: Google
* **Services enabled**:
  + Linux Virtual Machine

## DATABASE SOFTWARE UTILITIES

| **Vendor** | **Product** | **Version** | **Comments** |
| --- | --- | --- | --- |
| pgAdmin | PGAdmin | 4.0.0 | This application enables the ability to mage the cloud PG from local machine |

## SUPPORT SOFTWARE

| **Product** | **Version** | **Purpose** |
| --- | --- | --- |
| DBeaver 7.2.0 | Offline software | Help showing the table itself and its relations with other tables inside the schema on PostgreSQL platform |

# ARCHITECTURE

## HARDWARE & SOFTWARE ARCHITECTURE

Handled by GraphDB

## DATASTORES

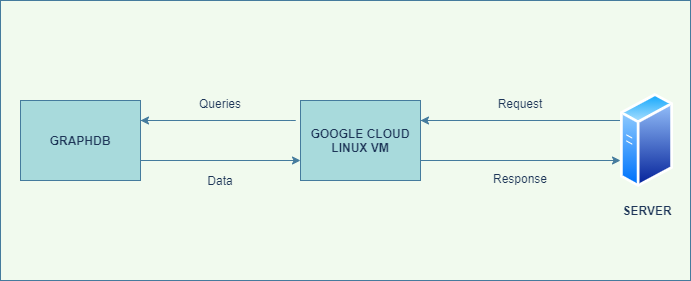


Figure 1. Datastores

GraphDB is a family of well-functioning, robust and awesome RDF databases. It reduces the load and use of connected cloud databases, as well as the RDF resources. We use GraphDB on an Linux Virtual Machine that is runned on Google Cloud platform for better querying, high performance processing and faster response.

# DATABASE-WIDE DESIGN DECISIONS

## KEY FACTORS INFLUENCING DESIGN

* The database should be designed independent when the frontend and backend are still being developed.
* The database should be designed to meet the data warehouse principles.

## PERFORMANCE AND AVAILABILITY DECISIONS

* We use common dimensional tables for all the fact tables for better performance.
* The data warehouse should be designed with the star schema for speed-up querying and processing time.
* The data warehouse should be redundant for reducing response time and meet up with the backend design pattern of the system.
* Data warehouse should be stored with PostgreSQL.
* RDF Data Cubes should be fully defined.
* RDF Data Cubes should be deployed on GraphDB.
* Class hierarchy and relationships should be done before extracting the data.
* RDF Data Cubes should be validated correctly before writing RestAPI.

# DATABASE ADMINISTRATIVE FUNCTIONS

## RESPONSIBILITY

| **Role** | **Name** | **Responsibility** | **Email Address** |
| --- | --- | --- | --- |
| Database Administrator | Dong, Ky Huu |  | kyhuudong@gmail.com |
| System Administrator | Hoa, Vo |  | hoavo.dng@gmail.com |
| Security Administrator | Hoa, Vo |  | hoavo.dng@gmail.com |

## 

## APPLICATIONS/SYSTEMS USING THE DATABASE

| **System ID** | **Model** | **Version** | **System Code** |
| --- | --- | --- | --- |
| SDA-Web | NA | In Development | NA |

## PHYSICAL DESIGN

### FACT-INDUSTRY

| **factindustry** | | | | |
| --- | --- | --- | --- | --- |
| **Field** | **Type** | **Constraint** | **Nullable** | **Description** |
| Industryid | int(auto increment) | PK | No | Industry id is primary key to specific unique row |
| cityid | VARCHAR | FK | No | City id is the foreign key and is the primary key of the city dimension |
| yearid | int | FK | No | Year id is the foreign key and is the primary key of the city dimension |
| industry | double |  | Yes | Industrial data |

### FACT-POPULATION

| **factpopulation** | | | | |
| --- | --- | --- | --- | --- |
| **Field** | **Type** | **Constraint** | **Nullable** | **Description** |
| populationid | int(auto increment) | PK | No | Population id is primary key to specific unique row |
| cityid | VARCHAR | FK | No | City id is the foreign key and is the primary key of the city dimension |
| yearid | int | FK | No | Year id is the foreign key and is the primary key of the city dimension |
| population | double |  | Yes | Population data |

### FACT-CLIMATE

| **factclimate** | | | | |
| --- | --- | --- | --- | --- |
| **Field** | **Type** | **Constraint** | **Nullable** | **Description** |
| climateid | int(auto increment) | PK | No | Climate id is primary key to specific unique row |
| cityid | VARCHAR | FK | No | City id is the foreign key and is the primary key of the city dimension |
| yearid | int | FK | No | Year id is the foreign key and is the primary key of the city dimension |
| humidity | double |  | Yes | Humidity data |
| rainfall | double |  | Yes | Rainfall data |
| temperature | double |  | Yes | Temperature data |

### FACT-FOREST

| **factforest** | | | | |
| --- | --- | --- | --- | --- |
| **Field** | **Type** | **Constraint** | **Nullable** | **Description** |
| forestid | int(auto increment) | PK | No | Forest id is primary key to specific unique row |
| cityid | VARCHAR | FK | No | City id is the foreign key and is the primary key of the city dimension |
| yearid | int | FK | No | Year id is the foreign key and is the primary key of the city dimension |
| afforestation | double |  | Yes | Afforestation data |
| forestcover | double |  | Yes | Forest Cover data |

### DIMCITY

| **dimcity** | | | | |
| --- | --- | --- | --- | --- |
| **Field** | **Type** | **Constraint** | **Nullable** | **Description** |
| cityid | VARCHAR(SEQUENCE(‘xxxx’), 1000) | PK | No | City id is primary key to specific unique row |
| city | VARCHAR |  | No | City name data |

### DIMYEAR

| **dimyear** | | | | |
| --- | --- | --- | --- | --- |
| **Field** | **Type** | **Constraint** | **Nullable** | **Description** |
| yearid | int(auto increment) | PK | No | Year id is primary key to specific unique row |
| year | int |  | No | Year data |

### FOREST-SOURCE-DATA-STAGING

| **forestsourcedatastaging** | | | | |
| --- | --- | --- | --- | --- |
| **Field** | **Type** | **Constraint** | **Nullable** | **Description** |
| Source\_stagingid | int(auto increment) | PK | No | Source\_staging id is primary key to specific unique row |
| city | VARCHAR |  | Yes | City name data |
| cityid | VARCHAR |  | Yes | City id from dimcity |
| year | int |  | Yes | Year data |
| yearid | int |  | Yes | Year id from dimyear |
| afforestation | double |  | Yes | Afforestation data |
| forestcover | double |  | Yes | Forestcover data |

### CLIMATE-SOURCE-DATA-STAGING

| **climatesourcedatastaging** | | | | |
| --- | --- | --- | --- | --- |
| **Field** | **Type** | **Constraint** | **Nullable** | **Description** |
| source\_stagingid | int(auto increment) | PK | No | Source\_staging id is primary key to specific unique row |
| city | VARCHAR |  | Yes | City name data |
| cityid | VARCHAR |  | Yes | City id from dimcity |
| year | int |  | Yes | Year data |
| yearid | int |  | Yes | Year id from dim year |
| humidity | double |  | Yes | Humidity data |
| rainfall | double |  | Yes | Rainfall data |
| temperature | double |  | Yes | Temperature data |

### POPULATION-SOURCE-DATA-STAGING

| **populationsourcedatastaging** | | | | |
| --- | --- | --- | --- | --- |
| **Field** | **Type** | **Constraint** | **Nullable** | **Description** |
| source\_stagingid | int(auto increment) | PK | No | Source\_staging id is primary key to specific unique row |
| city | VARCHAR |  | Yes | City name data |
| cityid | VARCHAR |  | Yes | City id from dimcity |
| year | int |  | Yes | Year data |
| yearid | int |  | Yes | Year id from dimyear |
| population | double |  | Yes | Population data |

### INDUSTRY-SOURCE-DATA-STAGING

| **industrysourcedatastaging** | | | | |
| --- | --- | --- | --- | --- |
| **Field** | **Type** | **Constraint** | **Nullable** | **Description** |
| source\_stagingid | int(auto increment) | PK | No | Source\_staging id is primary key to specific unique row |
| city | VARCHAR |  | Yes | City name data |
| cityid | VARCHAR |  | Yes | City id from dimcity |
| year | int |  | Yes | Year data |
| yearid | int |  | Yes | Year id from dimyear |
| industry | double |  | Yes | Industrial data |

## RDF Data Cubes Design

### Dimension

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

### Measure

| **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 1. 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**/  Cube Component specifications. **qb:ComponentSpecification** | http://sda-research.ml/dc/climate/dccs/ |

**Table 2. 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 | 1.<http://sda-research.ml/dc/climate/prop/>humidity  2. http://sda-research.ml/dc/climate/prop/rainfall  3.http://sda-research.ml/dc/climate/prop/temperatur-e |

#### Industry Data Cube

**Table 1. 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 change if codelists are used. Slices [qb:Slice, qb:SliceKey] would also be included here, if used. | http://sda-research.ml/dc/industry/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/industry/prop/ |
| **Cube Component Specifications**  **[dccs]**  {BaseURI}dc/{cube name}/**dccs**/  Cube Component specifications. **qb:ComponentSpecification** | http://sda-research.ml/dc/industry/dccs/ |

**Table 2. URI's for Dimensions and Measures**

| **Component Pattern** | **Value for Project** |
| --- | --- |
| dimension | 1. [http://sda-research.ml/dc/industry/prop/](http://sda-research.ml/dc/climate/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/](http://sda-research.ml/dc/climate/prop/)humidity |

### 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 |

### 

### Class hierarchy

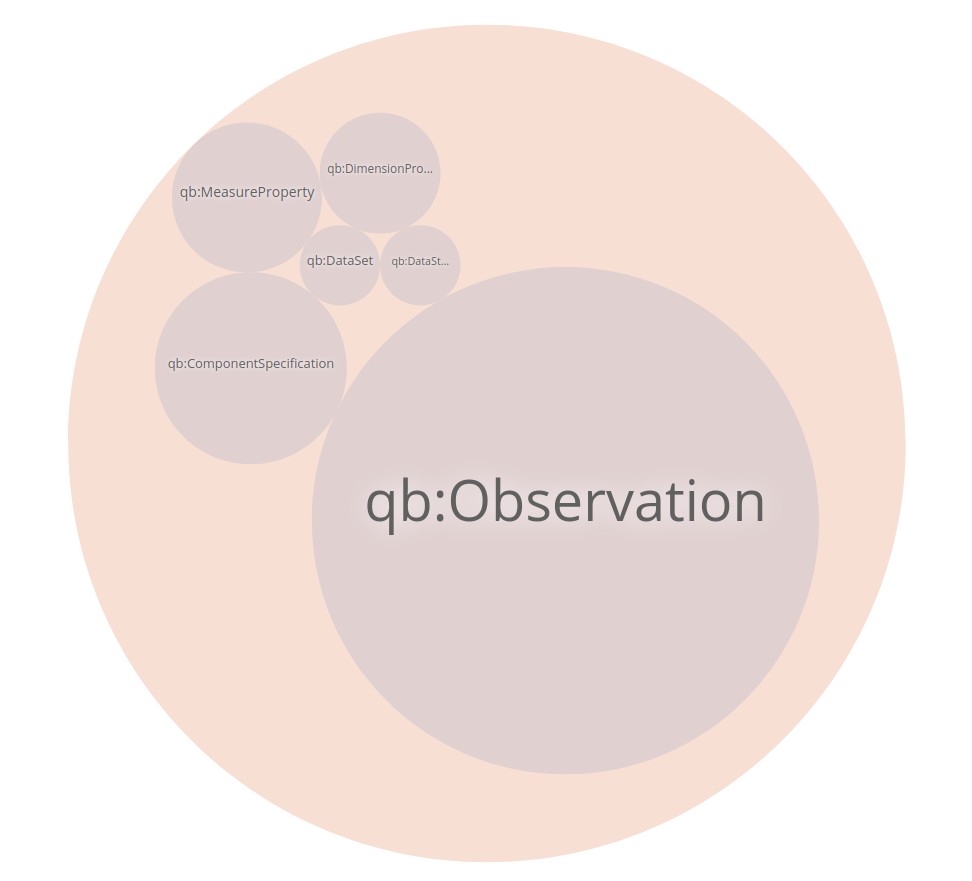


Figure 2. Class Hierarchy

### Class relationships

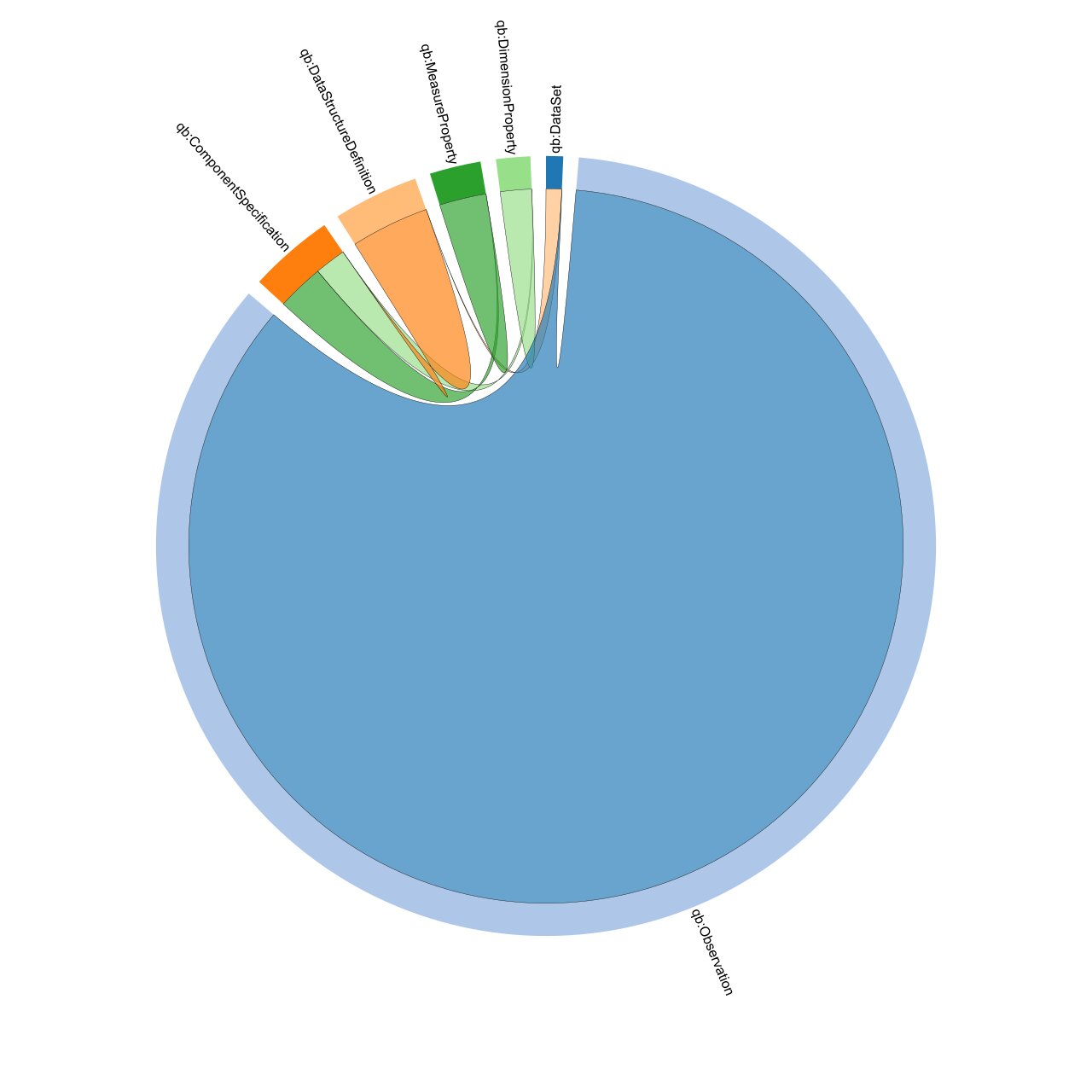


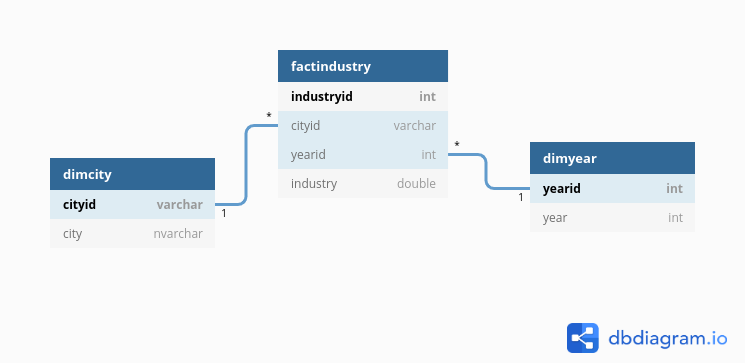
Figure 3. Class Relationships

## ENTITY MAPPING

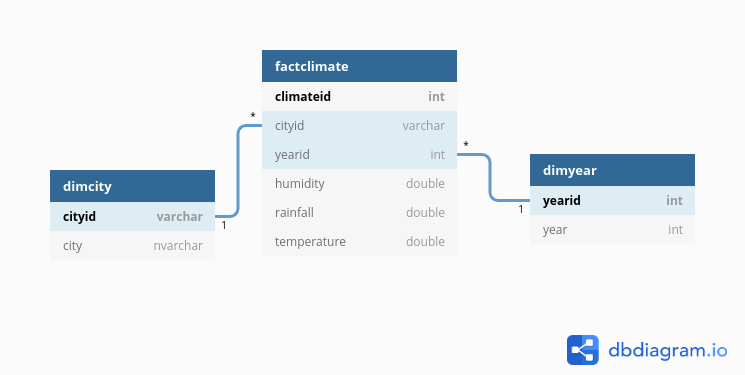
### ENTITY MAPPING DIAGRAM

* Please see the attached image for more details.

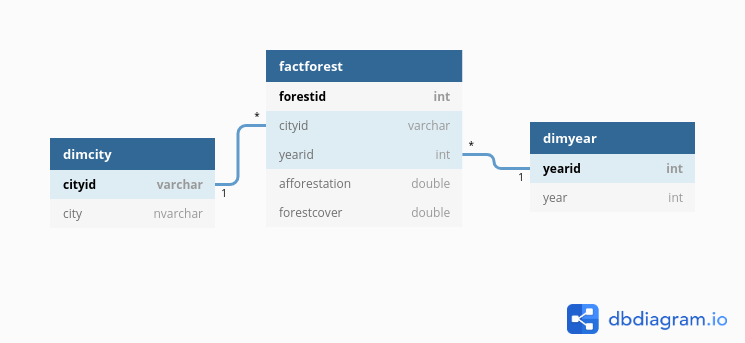
### INDUSTRY



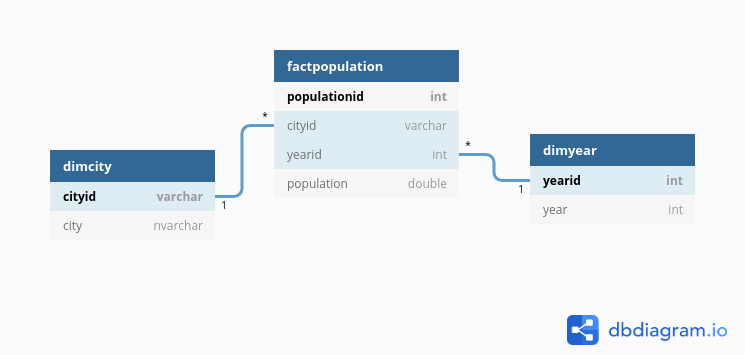
### CLIMATE



### FOREST



### POPULATION



# REFERENCES

* Technologies Stack Document
* dbdiagram.io : <https://dbdiagram.io>
* The RDF Data Cube: <https://www.w3.org/TR/eo-qb/#Datacube>
* Entity Relationship Mapping: <https://docs.oracle.com/cd/A97688_16/generic.903/a97677/ormap.htm>
* Entity Mapping Diagram For modeling ETL processes: <https://www.researchgate.net/profile/Ali_El-Bastawissy/publication/236030320_Entity_Mapping_Diagram/links/00463515e17cb192b4000000/Entity-Mapping-Diagram.pdf>