Facciolo Eugenio 2065516

Climate-Water DataWarehouse

Analyzing the Impact of Climate Change on Water Quality

Project for the course Data Management 2023/24

Index

Objectives

- Create a centralized and structured database that consolidates climate and water-related data from various sources

- Perform relevant analysis on the relationship between Climate Data and Water Quality Data, considering locations, parameters, date, and sources data

Back to Index

Schedule

Dataset Collection

- Set up a Python environment with the required libraries
- Identify and download datasets

Data Integration

- Design a possible dataset's structure and key statistics
- Produce a DFM and a StarSchema

ETL Operations

- Ensure the datasets are in a consistent and usable format
- Load the cleaned and integrated data into the PostgreSQL database

Analysis and Optimization

- Generate insights from the data using OLAP operations

Back to Index

Tools

Visual Studio Code + GitHub

For developing the code

Pandas + Numpy + SqlAlchemy

For data manipulation and database interaction

PostgreSQL + pgAdmin

For managing the database

Back to Index

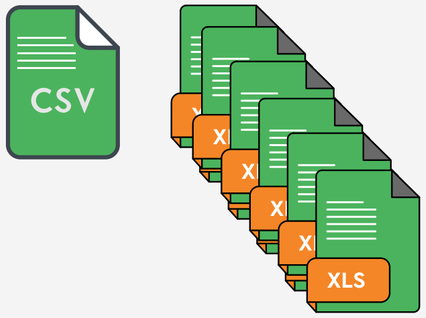
Datasets

Back to Index

Climate Data

Climate Insights Dataset

Global Environmental Indicators



Extra Data

World Map of Köppen-Geiger

Latitude and Longitude

Countries by Continent

Water Quality Data

Global Water Quality GEMStat Dataset

- *Click on dataset names to be redirected to the source*

DFM

Climate Data

+

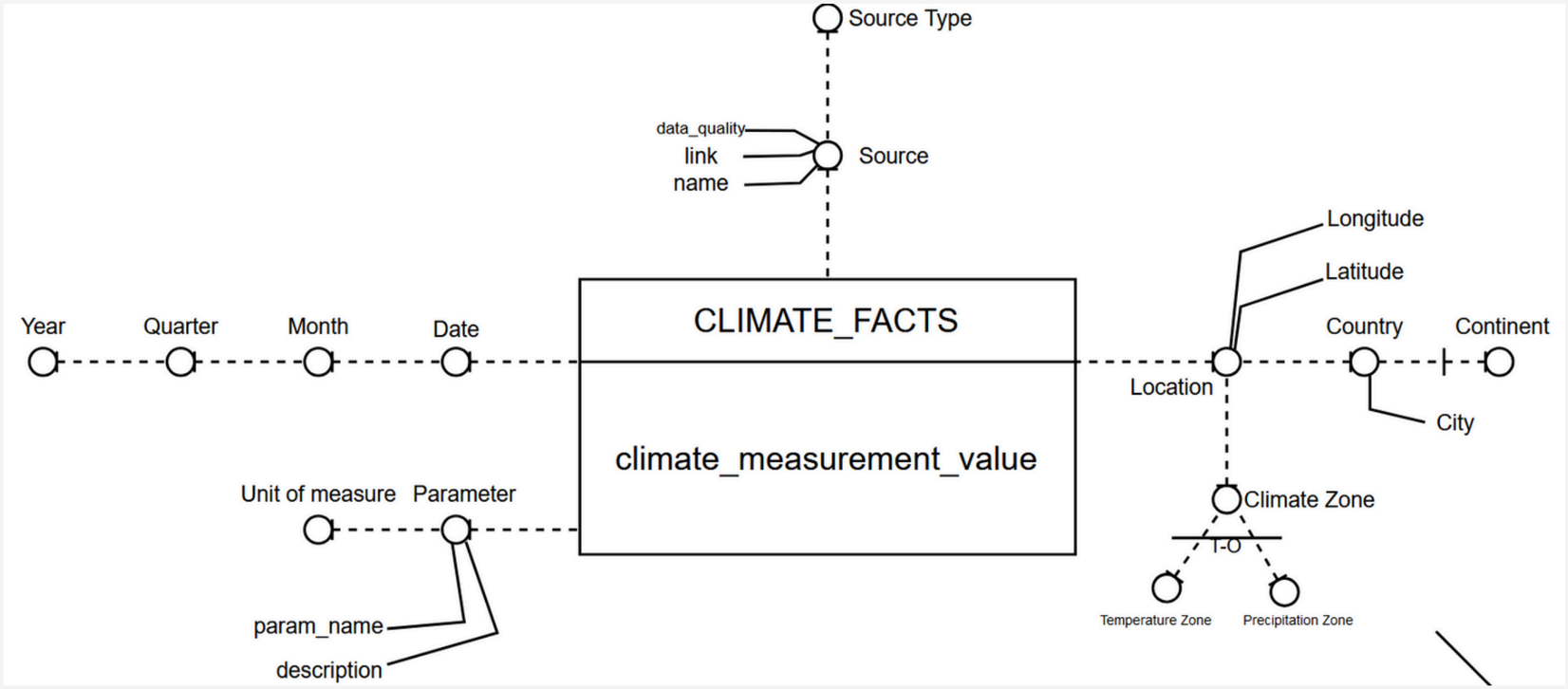
Extra Data

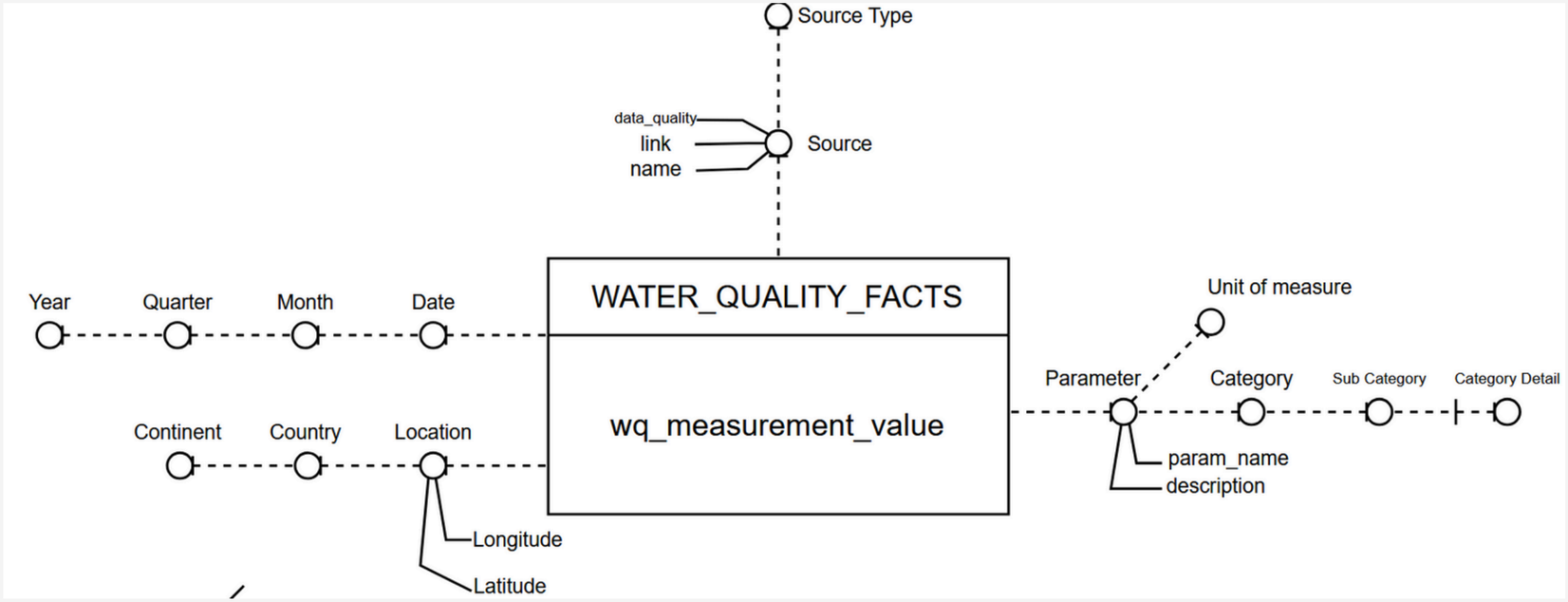
Water Quality Data

+

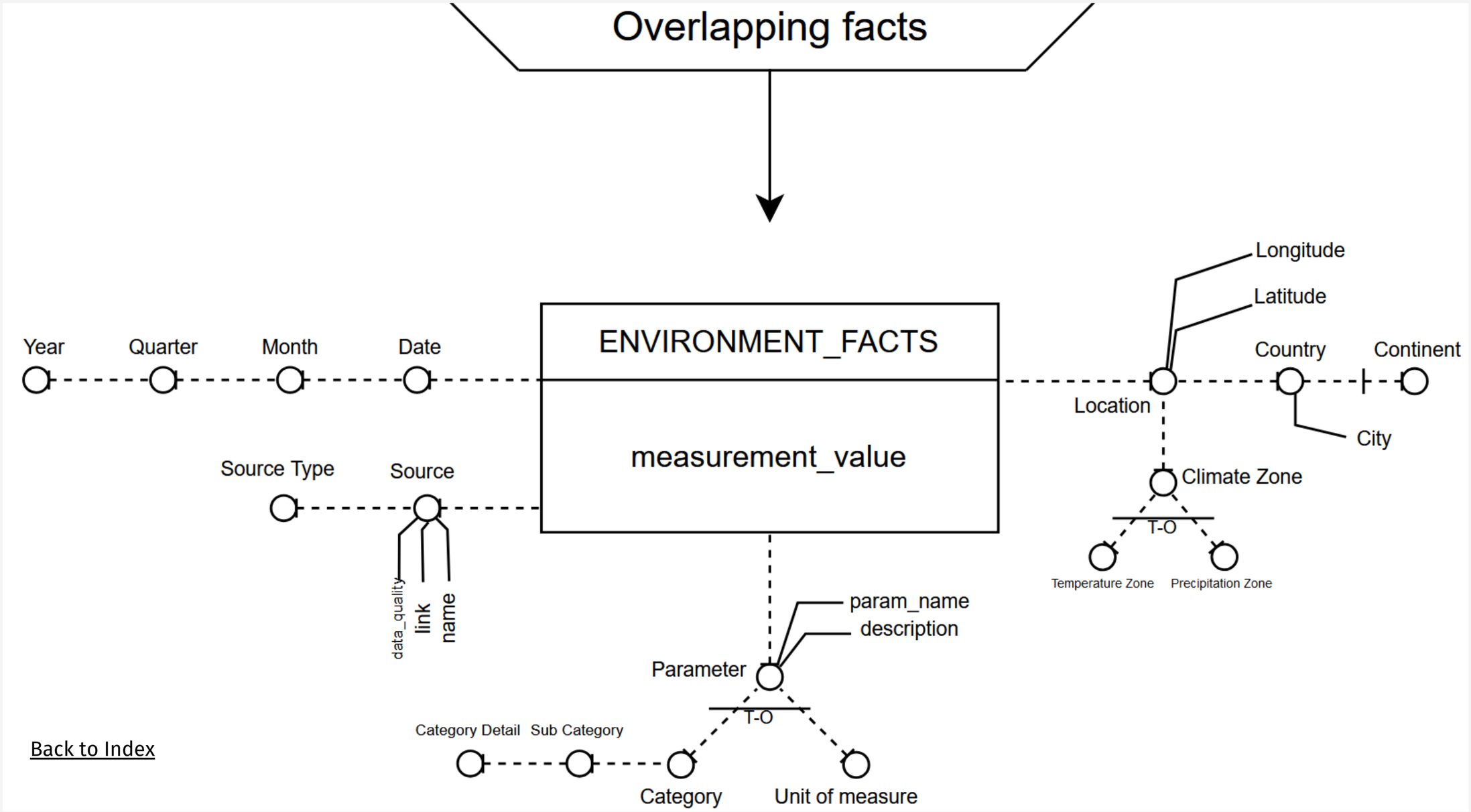
Extra Data

Back to Index



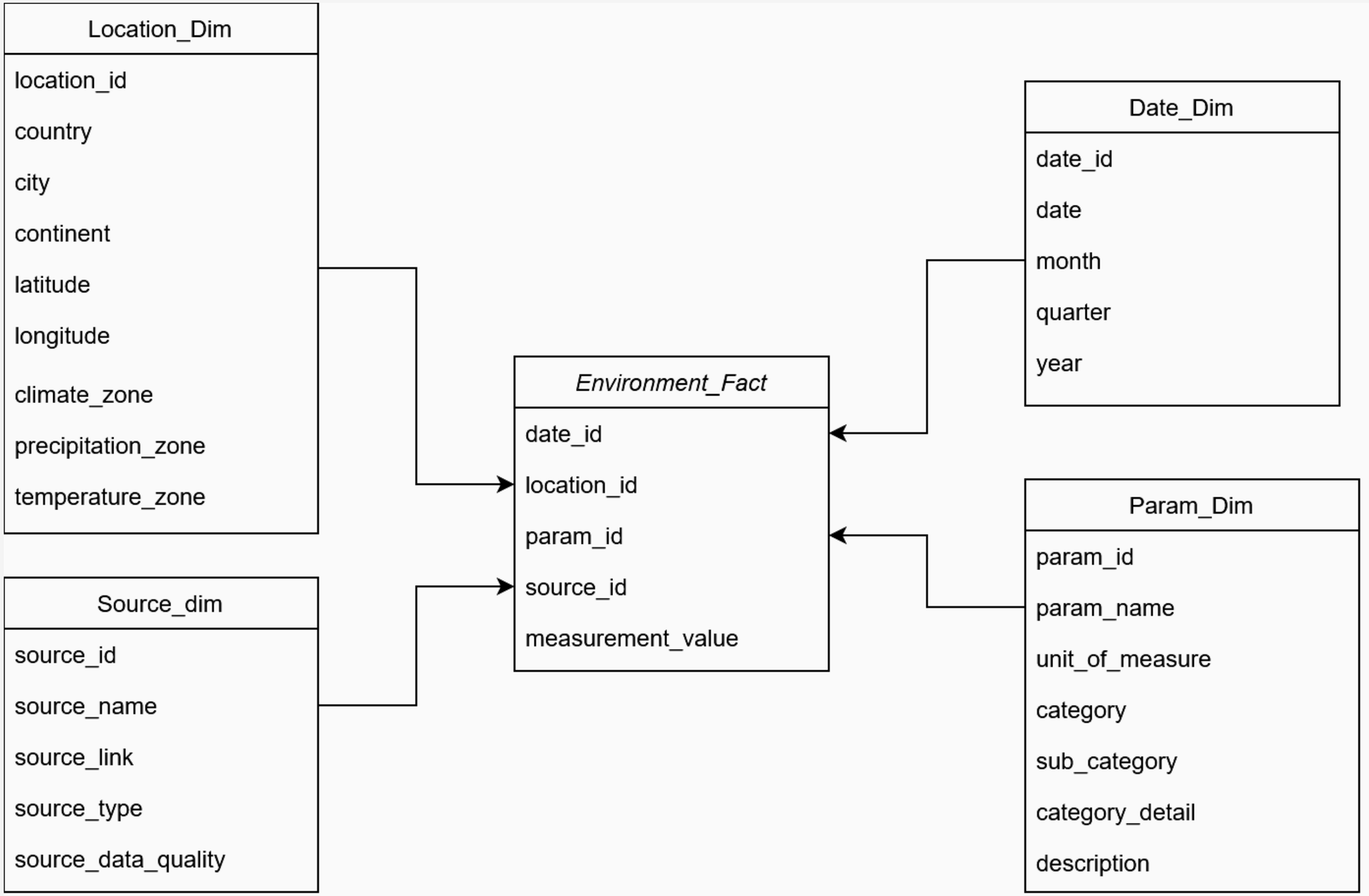


Climate Data + Extra Data + Water Quality Data



Back to Index

Star Schema



Back to Index

ETL Operations

Extraction

Extract useful data from datasets and create dataframes

- *read_csv_file(climatePath + '1.1_Climate_Insights_Dataset/climate_change_data.csv', ",", 0, None, None)*
- *read_excel_file(GEIPath + 'Air and Climate/GHG_Emissions.xlsx', 0, 16, None, "B,C,D,F,H,J", 0, False, 20)*

Trasformation

Map dataframe structure and **Standardize** dataframe values

- *df.rename(columns={'latest year available': 'Date'}, inplace=True)*

- *numericalDf = numericalDf.map(standardize_numerical_format)*
- *non_numericalColumns['Date'] = non_numericalColumns['Date'].map(standardize_datetime_format_CID)*
- *dfFormatted = pd.concat([non_numericalColumns, numericalDf], axis=1).reset_index(drop=True)*

Back to Index

Cleansing

Clean data and perfom **Imputation** on missing values :

- df.dropna(axis=0, thresh=row_threshold, inplace=True)
- df.dropna(axis=0, subset=['Country', 'Date'], inplace=True)
- df.drop_duplicates(inplace=True)

- *imputer = IterativeImputer(max_iter=10, random_state=0)*
- *imputed_array = imputer.fit_transform(numericalDf)*

Loading

Load the cleaned and integrated data into the PostgreSQL database

- *insert_country = text('''*
- *INSERT INTO "Location_Dim" (country, city)*
- *VALUES (:country, :city)*
- *RETURNING location_id;*
- *'')*
- *new_insert = connection.execute(insert_country, {'country': country, 'city': city})*
- *location_id = new_insert.fetchone()[0]*

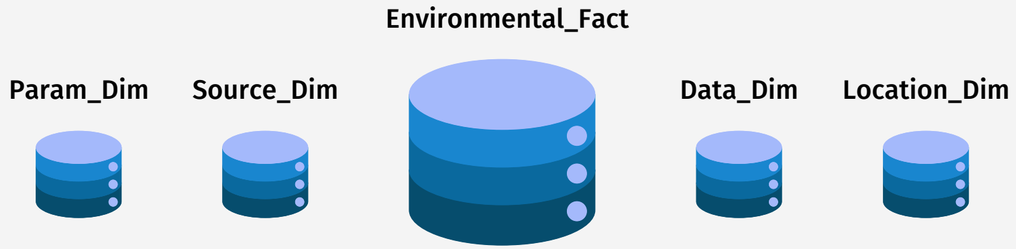
Back to Index

Database

PostgreSQL

Back to Index

ClimateWaterDataWarehouse Schema and Tables



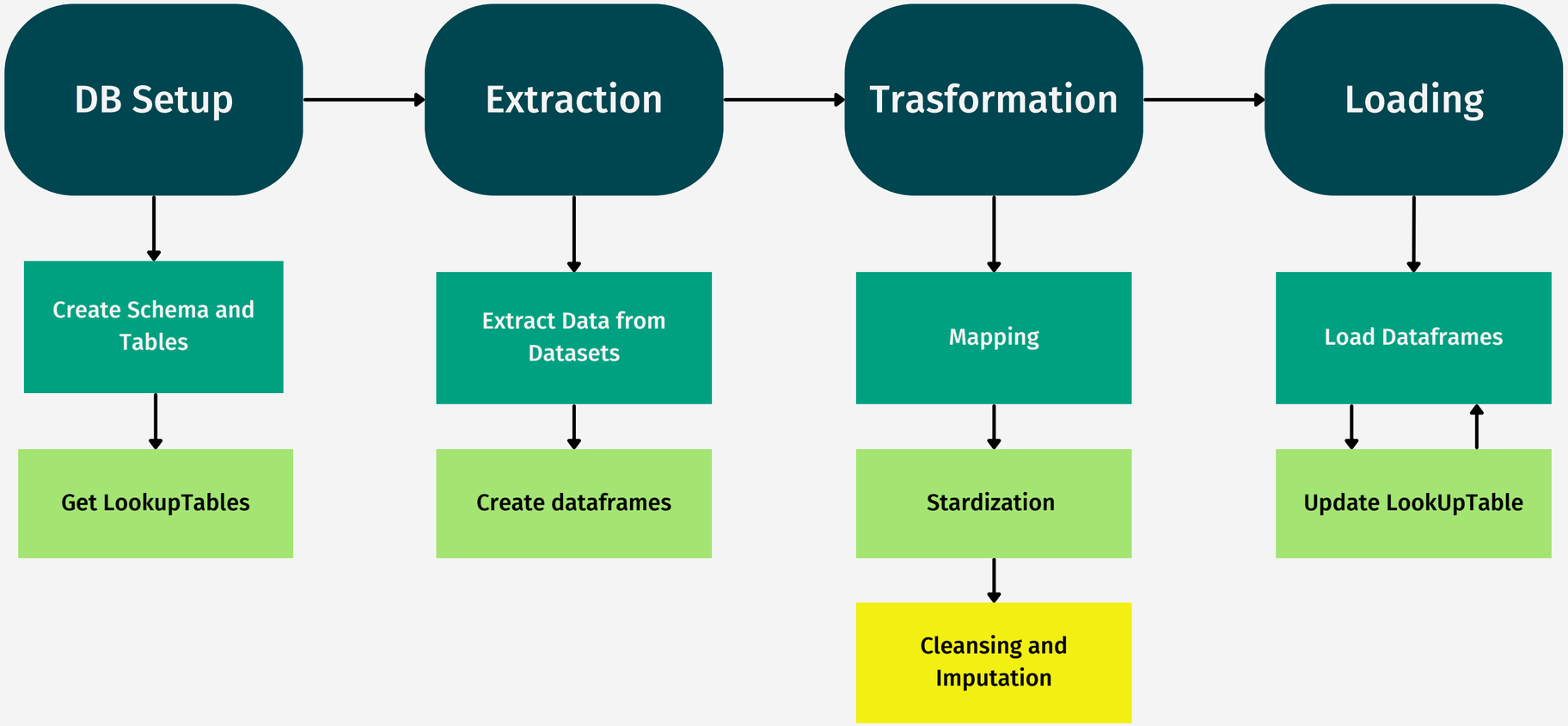

```
connection.execute(text('''
CREATE TABLE IF NOT EXISTS "Environment_Fact" (
date_id INT REFERENCES "ClimateWaterDataWarehouse"."Date_Dim"(date_id),
location_id INT REFERENCES "ClimateWaterDataWarehouse"."Location_Dim"(location_id),
param_id INT REFERENCES "ClimateWaterDataWarehouse"."Param_Dim"(param_id),
source_id INT REFERENCES "ClimateWaterDataWarehouse"."Source_Dim"(source_id),
measurement_value FLOAT(24),
CONSTRAINT unique_mv UNIQUE (date_id, location_id, param_id, source_id));'''))
```

LookUpTables


```
param_ids = {}
param_data = connection.execute(text('''
SELECT param_name, param_id
FROM "Param_Dim"
''')).fetchall()
param_ids = {row[0]: row[1] for row in param_data}
```

Back to Index

Data Workflow



Back to Index

OLAP Operations

Roll-Up

Aggregate measurement_value by year instead of by month.

```
SELECT D.year, P.param_name, AVG(EF.measurement_value) AS avg_measurement
FROM Environment_Fact AS EF
JOIN Date_Dim AS D ON EF.date_id = D.date_id
JOIN Param_Dim AS P ON EF.param_id = P.param_id
GROUP BY D.year, P.param_name
ORDER BY D.year, P.param_name;
```

Drill-Down

Increase the granularity of the analysis from year to month

```
SELECT D.year, P.param_name, AVG(EF.measurement_value) AS avg_measurement
FROM Environment_Fact AS EF
JOIN Date_Dim AS D ON EF.date_id = D.date_id
JOIN Param_Dim AS P ON EF.param_id = P.param_id
GROUP BY D.year, D.month, P.param_name
ORDER BY D.year, D.month, P.param_name;
```

Back to Index

Slice-and-Dice

Slice on ‘Water Temperature’ in ‘Europe’ and **Dice** on ‘Warm Summer’ temperature zones.

Back to Index

```
SELECT L.country, L.temperature_zone, D.year, AVG(EF.measurement_value) AS avg_temp
FROM Environment_Fact AS EF
JOIN Param_Dim AS P ON EF.param_id = P.param_id
JOIN Location_Dim AS L ON EF.location_id = L.location_id
JOIN Date_Dim AS D ON EF.date_id = D.date_id
WHERE P.param_name = 'Water Temperature'
AND L.continent IN ('Europe')
AND L.temperature_zone = 'Warm Summer'
GROUP BY L.country, L.temperature_zone, D.year
ORDER BY L.country, L.temperature_zone, D.year;
```

Pivoting

Change the view from ‘Avg Measurement of Continent’ for each year to ‘Avg Measurement of Continent-Year’ combination

```
SELECT P.param_name,
    AVG(CASE WHEN L.continent = 'North America' AND D.year = 2020 THEN EF.measurement_value END) AS "Avg Measurement of North America in 2020",
    AVG(CASE WHEN L.continent = 'North America' AND D.year = 2021 THEN EF.measurement_value END) AS "Avg Measurement of North America in 2021",
    AVG(CASE WHEN L.continent = 'Europe' AND D.year = 2020 THEN EF.measurement_value END) AS "Avg Measurement of Europe in 2020",
    AVG(CASE WHEN L.continent = 'Europe' AND D.year = 2021 THEN EF.measurement_value END) AS "Avg Measurement of Europe in 2021"
FROM ... JOIN ... AS ... ON ...
WHERE P.category = 'Chemical'
GROUP BY P.param_name;
```

Drill-Across

Combining data from two or more dataframes (During Mapping Phase)

```
SELECT
    SM."GEMS Station Number" AS station_number,
    SM."Country Name" AS country,
    SM."Station Type" AS station_type,
    S."Data" AS sample_date,
    PM."Param_Name" AS parameter_name,
    PM."Unit of Measure" AS unit_of_measure,
    S."Value" AS measurement_value
FROM
    Station_Metadata AS SM
JOIN
    samples AS S
    ON SM."GEMS Station Number" = S."Station Code"
JOIN
    Param_Metadata AS PM
    ON S."Param_Code" = PM."Param_Code"
ORDER BY
    SM."Country Name", SM."Station Type", S."Data";
```

Back to Index

Thanks for your attention

Demo Time

Facciolo Eugenio
Matricola 2065516

GitHub Repository
Repo of WaterClimateWarehouse

Back to Index