Florence Consulting Group website analysis

University of Calabria

Giacomo Galavotti

July 12, 2025

Introduction

This report aims at describing the Data Warehouse project developed for the analysis of the Florence Consulting Group website data. In particular, the conceptual and logical schemas, along with the ETL processes used on the website data are going to be explained.

Objective

The main objective for the project has been to create a Data Warehouse solution for analyzing performance and analytical metrics of the Florence Consulting Group website traffic, in order to gain insight on the best approaches to further develop the website. Specifically, we wish to analyze traffic performance data based on geographical location, and gain more analytical insight on users' interests based on demographic and usage data.

Data Analysis

As a first step on the engineering process of the Data Warehouse for the Florence Consulting Group website, the original source dataset provided by the company has been analyzed. The dataset consists on 11 `.csv` files, of which 7 of them represent functional data regarding site traffic, while the remaining 4 are metadata that are used to describe some of the numerical values of the traffic data. We present some indices[1] that represent the data quality of each source file present in the source dataset:

File	Completeness	Uniqueness	Validity	Consistency
`OS_listing.csv`	100.0	100.0	100.0	100.0
`browser_list.csv`	100.0	100.0	100.0	100.0
`country_listing.csv`	100.0	100.0	100.0	100.0
`pages_listing.csv`	100.0	97.9	100.0	100.0
`pages_topics.csv`	100.0	100.0	100.0	100.0
`session_listing.csv`	100.0	100.0	100.0	100.0
`visited_pages.csv`	100.0	100.0	90.0	100.0

A re-engineering step has been designed to compose a Relational Model schema to fit the data present in the source dataset, in order to better understand the relationships between the `.csv` files and give them a structure. The resulting Relational Model is as follows:

- **visit_page**(<u>url*</u>, <u>Session_Id*</u>, <u>OperationDate</u>, Country_code*, OS*, Browser*, Page_load_timing, Image_load_timing, Text_load_timing)

[1] Some data quality indices have been automatically calculated, while others were manually applied. Automatic indices were calculated via a python pandas script, present in file `data_quality.ipynb`.

- **page**(Page_Url, Page_name, Topic*, Gateway_Pass, Firewall_Pass)
- **session**(Session_Id, Gender, Age, Registered_User)
- **topic**(Topic_code, Topic_description)
- **os**(OS_code, OS_description)
- **browser**(Browser_code, Browser_description)
- **country**(Country_code, Country_name)

Conceptual and Logical Design

Following the design of the Relational Model for the source dataset, the resulting model has been used to construct the DFM schema of the dataset. The first step is the choice of the fact, whereby, considering the objective of the project, the choice has fallen to setting **page visits** as the fact of interest. Then, starting from the fact chosen, the attribute tree has been built as depicted in Figure 1.

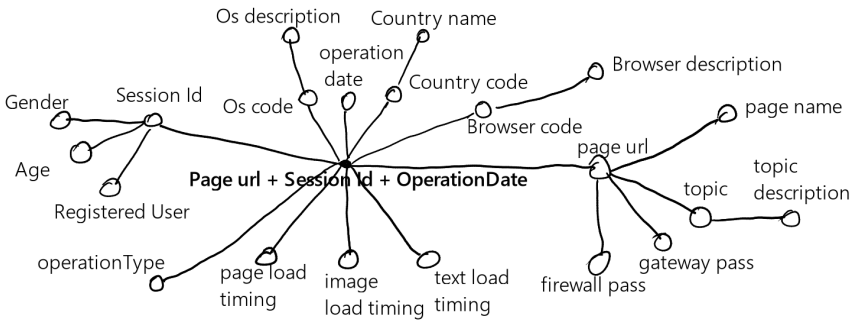


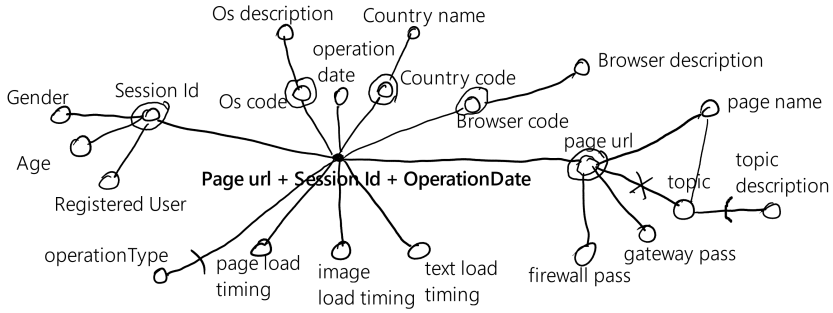
Figure 1: Initial attribute tree

Starting from the attribute tree, the process of pruning and grafting to keep only data important for the analysis has been initialized. By looking at the initial tree and at the content of the dataset the following modifications to the attribute tree have been implemented:

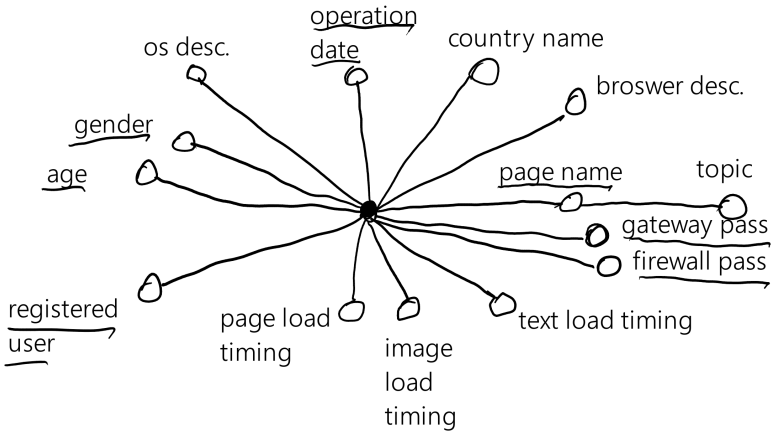
- We can ascertain that the data between identifiers of `OS`, `Country`, `Browser` and `url` and their respective child attributes have a 1-to-1 relation, implying that the former can be grafted and the latter can be kept to create a functional dependency with the fact node.
- `operationType` has been pruned from the tree since it contained the same value for all rows of the dataset, while `Topic description` has been pruned since it contains the same information as `Topic`.

- After grafting `Page url`, we wish to keep the dimension for `Topic`, so we create a functional dependency between `Page name` and `Topic`, that is possible thanks to the 1-to-1 relation between `Page name` and `Page url`.

The graphical representation of the modifications of the attribute tree is seen on Figure 2a, while the resulting final attribute tree is seen on Figure 2b.



(a) Modified attribute tree.



(b) Cleaned attribute tree.

Figure 2: Pruning and grafting of the attribute tree.

Completed the attribute tree, we can define dimensions, hierarchies and measures for the DFM schema. Since the schema has become temporal, for the changes of granularity in the tree, we explicitely define the measures as follows:

- Number of page visited: COUNT(*)
- Average Latency per page visited: `AVG(page load timing + image load`

`timing + text load timing + (gateway pass * 15) + (firewall pass * 30))`

- Max. Latency per page visited: `MAX(page load timing + image load timing + text load timing + (gateway pass * 15) + (firewall pass * 30))`
- Min. Latency per page visited: `MIN(page load timing + image load timing + text load timing + (gateway pass * 15) + (firewall pass * 30))`

The latency formula uses two numerical constant that were defined in the metadata file `infrastracture_impact.csv`, that keeps data about the amount of latency (ms) to add to a page visit when a gateway or firewall pass is present.

We define the following attributes as dimensions: `Date, OS, Country, Browser, Url name, Age`. Moreover, we define four hierarchies based on the defined dimensions: `OS Family` for `OS`, `Region` for `Country`, `Hour`, `Day`, `Month` and `Year` for `Date` and lastly, `Topic` for `Url name`. The resulting DFM schema, which takes in consideration the measures, dimensions and hierarchies defined beforehand, is depicted in Figure 3a. `Gender` and `registered user` are kept in the schema as descriptive attributes.

Once defined the DFM schema, we generate a logical star schema that better represents the DFM. The resulting star schema is depicted in Figure 3b.

Data Cleaning and Populating

Before creating the DBMS that should hold the reconciled data, structured by following the star schema, it has been decided to apply a cleaning phase on the source data. The cleaning phase was automated by implementing a Pentaho Data Integration transformation[2], that takes in input the source `*.csv` files, and outputs them in a `clean/` folder. The Figure 4a shows the transformation map in Pentaho. In this transformation the `latency` property, used as a measure, is pre-calculated to ease migration to the star schema.

Before applying the transformation, a problem with the source data file `country_listing.csv` has arisen. The file has been encoded using the `'latin'` encoding instead of a Unicode encoding, and unfortunately it has not been possible to change the conversion of this file by using Pentaho. As such, a python pandas script[3] has been developed to cover for this caveat, and needs to be run as a first step for the whole Pentaho data cleaning transformation to work.

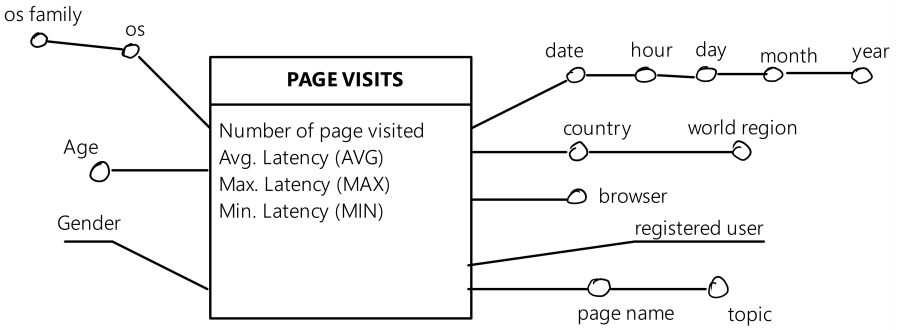
After the process of data cleaning, we can start populating the DBMS with the star schema. SQLite has been chosen as the DBMS to hold the starred data. The ETL process that transforms and loads the reconciled data to the DBMS

[2]The transformation is stored in the project folder in file `"data_cleaning.ktr"`

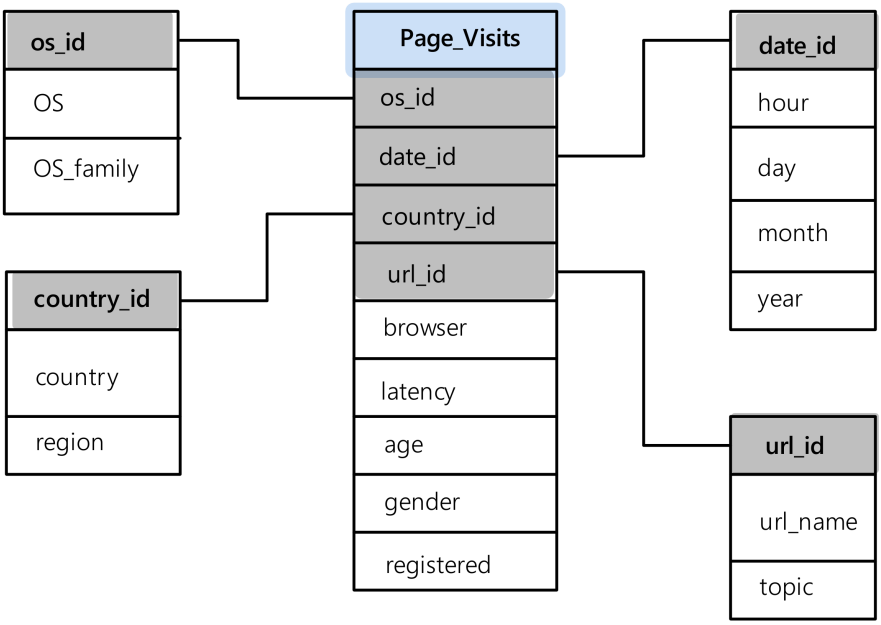
[3]The pandas script to change the encoding of the file is present in the project folder at `"change_encoding.ipynb"`

with the star schema has been implemented with a Pentaho transformation[4], depicted in Figure 4b.

[4]The file of the final transformation from reconciled data to star schema data is in the project folder at file "to_star_dbms.ktr"

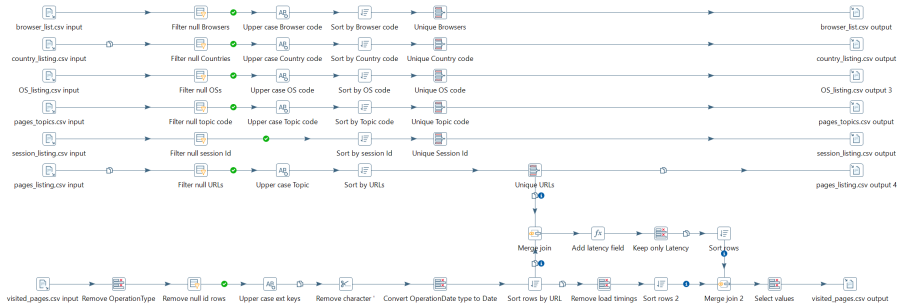


(a) DFM schema of the dataset.

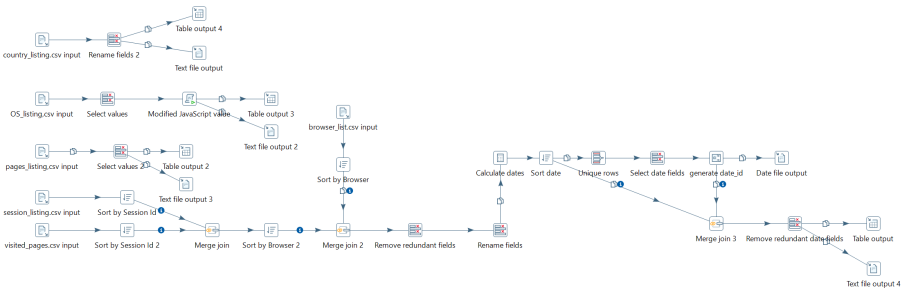


(b) Star schema of the dataset.

Figure 3: Final result of the Conceptual and Logical Design.



(a) Data cleaning transformation in PDI.



(b) Star schema transformation in PDI.

Figure 4: PDI transformations.