Kalpa Publications in Computing

Volume XXX, 2022, Pages 75–77

Proceedings of V XoveTIC Conference. XoveTIC 2022

Component for the visualization of a spatio-temporal data warehouse in a Geographic Information System.

Víctor L. Sardiña, Alejandro Cortiñas, Miguel R. Luaces y Óscar Pedreira

Universidade da Coruña, Centro de Investigación CITIC, Laboratorio de Bases de Datos, Facultade de Informática, Elviña, 15071 A Coruña, Spain {victor.lamas,alejandro.cortinas,miguel.luaces, oscar.pedreira}@udc.es

Abstract

In the Database Laboratory of the University of A Coruña, we have a software product line for the creation of Geographic Information Systems (GIS) on the web. In the case that the data is available in a data warehouse, the current functionality of the product line is not enough. Therefore, it is necessary to create a component in charge of connecting the data warehouse with the user interface providing the usual functionality (filters and aggregations in the different dimensions of the data warehouse).

Given that integrating the functionality and variability of the exploration of a data warehouse in a software product line is complicated, we have opted for a first approximation to define an exploration component in a GIS of any data warehouse. The component receives, utilizing a domain-specific language, a description of the data model (i.e. the facts and the hierarchy of dimensions, how each of them is related, and the way to retrieve their values) and makes the user interface components independent from the management of the communication with the data warehouse.

1 Introduction

Currently, in the Database Laboratory (LBD) we are working on three Geographic Information Systems (GIS), all of them related to the visualization of Spatio-temporal data stored in *data warehouses*. The projects in question are CEDCOVID, a system that allows the visualization of geographic data related to the incidence of COVID cases during the pandemic; MAGIST, a system that allows the visualization of the state of traffic in cities; and SIGTRANS, a system that allows the visualization of data related to the use of public transport. These three systems have in common that one of their main functionalities is the visualization of queries resulting from filtering and aggregation of the dimensions of data stored in a data warehouse.

The LBD has been working for years with a Software Product Line (SPL) for the generation of web-based GIS [1]. Although the basic functionality of each of the three systems can be implemented through the LPS, the functionality related to *data warehouses* is not available and it is not feasible to implement it as part of the LPS without a prior exploration of the different functionalities required in each product.

To avoid having to implement the logic that manages the different levels of aggregation and spatio-temporal filtering of the *data warehouses* as many times as projects are to be developed, and to acquire the necessary knowledge to address future integration in the LPS, a component has been designed that allows a declarative description of the dimensions, their relationships (dependencies) and how to retrieve the possible values of each dimension.

On the other hand, it is also desirable to decouple the visual components of the data retrieval process, with the option that the values of the dimensions can come from different sources since there is the possibility that there are dimensions that can have static or dynamic values depending on the application.

2 Proposal

To support any data warehouse that follows a snowflake scheme with dimensions connected to the fact table, we have designed a component that receives a specification defining the dimensions and the relationships between them. In addition, the developer can indicate how the possible values for each dimension are retrieved taking into account those selected in other related dimensions, and how the user interface is redrawn when necessary. With this information, the component takes care of managing changes in the user interface and communication with the data warehouse when necessary.

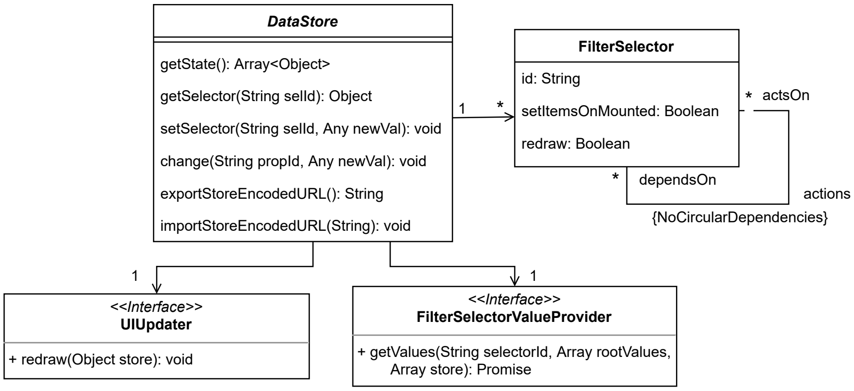
DataStore
getState(): Array<Object>
getSelector(String selId): Object
setSelector(String selId, Any newVal): void
change(String propId, Any newVal): void
exportStoreEncodedURL(): String
importStoreEncodedURL(String): void
1
*
FilterSelector
id: String
setItemsOnMounted: Boolean
redraw: Boolean
* actsOn
* dependsOn
actions
{NoCircularDependencies}
1
<<Interface>>
UIUpdater
+ redraw(Object store): void
1
<<Interface>>
FilterSelectorValueProvider
+ getValues(String selectorId, Array rootValues, Array store): Promise

Figure 1: Diagrama de Clases del componente propuesto

Figure 1 shows a class diagram of the component. `DataStore` is the main class that controls and manages the internal state of the selectors and is used by the developer integrating the library. `FilterSelector` represents the set of classes corresponding to the dimensions and relations of the data warehouse. They are created from a specification (see example in fig. 2). `FilterSelectorValueProvider` is the interface that is implemented with the logic to retrieve the dimension data. For example: for the *province* level of the *spatial* dimension, a list of provinces should be retrieved. `UIUpdater` is the interface that defines the behavior of the application when the view needs to be refreshed (usually updating the geometries of a map or the elements of a list).

As an example, in the left part of Figure 2 we see the specification that would implement the *data warehouse* model. From this specification, views with selectors are created (right-hand side of the fig. 2). The first filter, `MUNICIPALITY_FILTER`, will be automatically populated with the values of the municipalities, which will be retrieved using the concrete implementation of the interface `FilterSelectorValueProvider`. When a municipality is selected, the `DISTRICT`

```
[{ "id": "MUNICIPALITY_FILTER",
  "label": "Municipality Filter",
  "type": "selector",
  "actions": ["DISTRICT_FILTER"]
}, {
  "id": "DISTRICT_FILTER",
  "label": "District Filter",
  "type": "selector",
  "actions": ["SECTION_FILTER"]
}, {
  "id": "SECTION_FILTER",
  "label": "Section Filter",
  "type": "selector",
  "redraw": true
}, {
  "id": "DATE_FILTER",
  "label": "Date Filter",
  "type": "date",
  "redraw": true
}, {
  "id": "TEMPORAL_FILTER",
  "label": "Temporal Filter",
  "type": "selector",
  "actions": ["DATE_FILTER"] }]
```

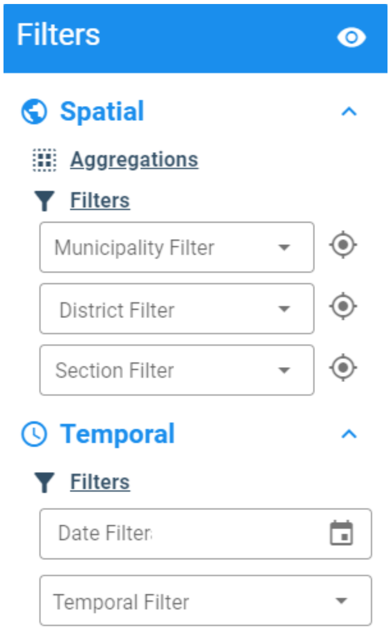

Figure 2: Specification and resulting visual component

element shall be updated, as indicated in the specification (property `actions`). To update the values of this new selector, the implementation of `FilterSelectorValueProvider` shall be used again. In addition, the values of `SECTIONFILTER` will be updated, and it will continue with the rest of the filters. It is also possible to define a property `redraw`, which tells the component that, when the selector value changes, a request must be launched to update the UI, using the implementation of `UIUpdater`.

3 Conclusions and future work

The proposed solution defines a component that helps code reuse for systems using data warehouses that perform spatio-temporal data aggregations and filters. By defining the dimensions, their relationships, and the way in which their possible values are retrieved, the proposed component manages the selectors and avoids developing the logic for each of the systems.

Funding: MCIN/AEI/10.13039/501100011033, NextGenerationEU/PRTR, FLATCITY-POC: PDC2021-121239-C31; MCIN/AEI/10.13039/501100011033 MAGIST: PID2019-105221RB-C41; MCIN/AEI/10.13039/501100011033 EXTRACompact: PID2020-114635RB-I00; GAIN/Xunta de Galicia/ERDF CEDCOVID: COV20/00604; Xunta de Galicia/FEDER-UE GRC: ED431C 2021/53; MICIU/FEDER-UE BIZDEVOPSGLOBAL: RTI-2018-098309-B-C32.

References

[1] A. Cortiñas, M. R. Luaces, O. Pedreira, A. S. Places, and J. Perez. Web-based geographic information systems sple: Domain analysis and experience report. In *Proc. 21st International Systems & Software Product Line Conference (SPLC 2017) Vol.1*, pages 190–194, Sevilla, 2017.