



Universidade Nova de Lisboa - Faculdade de Ciências e Tecnologia

Department of Computer Science

SOLAP+:

Extending the Interaction Model

By Ruben Jorge

Thesis submitted to Faculdade de Ciências e Tecnologia of the Universidade Nova de Lisboa,
in partial fulfillment of the requirements for the degree of **Master in Computer Science**

Supervisor: PhD João Moura Pires

Monte da Caparica, July 2009

Abstract

Decision making is a crucial process that can dictate success or failure in today's businesses and organizations. Decision Support Systems (DSS) are designed in order to help human users with decision making activities. Inside the big family of DSSs there is OnLine Analytical Processing (OLAP) - an approach to answer multidimensional queries quickly and effectively.

Even though OLAP is recognized as an efficient technique and widely used in mostly every area, it does not offer spatial analysis, spatial data visualization nor exploration. Geographic Information Systems (GIS) had a huge growth in the last years and acquiring and storing spatial data is easier than ever. In order to explore this potential and include spatial data and spatial analysis features to OLAP, Bédard introduced Spatial OLAP (SOLAP). Although it is a relatively new area, many proposals towards SOLAP's standardization and consolidation have been made, as well as functional tools for different application areas.

There are however many issues and topics in SOLAP that are either not covered or with incompatible/non general proposals. We propose to define a generic model for SOLAP interaction based on previous works, extending it to include new visualization options, components and cases; create and present a component-driven architecture proposal for such a tool, including descriptive metamodels, aggregate navigator to increase performance and a communication protocol; finally, develop an example prototype that partially implements the proposed interaction features, taking into consideration guidelines for a user friendly, yet powerful and flexible application.

Resumo

A tomada de decisões é um processo crucial que pode ditar o sucesso ou insucesso dos negócios e organizações de hoje. Os Sistemas de Apoio à Decisão (SAD) são desenhados de modo a ajudar os utilizadores humanos nas actividades de tomada de decisões. Dentro da grande família dos SAD está o Processamento Analítico Online (OLAP) - uma aproximação para dar resposta a consultas multidimensionais de modo rápido e eficaz.

Ainda que OLAP seja reconhecido como uma técnica eficiente e amplamente utilizada em praticamente qualquer área, esta não oferece análise espacial, visualização de dados espaciais nem exploração. Os Sistemas de Informação Geográfica (SIG) tiveram um crescimento enorme nos últimos anos e adquirir e armazenar dados espaciais é mais fácil que nunca. De modo a explorar este potencial e incluir dados espaciais e funcionalidades de análise espacial em OLAP, Bédard introduziu o Spatial OLAP (SOLAP). Apesar de ser uma área relativamente nova, várias propostas no sentido de criar *standards* e consolidar SOLAP têm sido efectuadas, assim como ferramentas funcionais para diferentes áreas de aplicação.

Existem no entanto vários problemas e assuntos em SOLAP que ou não estão cobertos ou têm propostas incompatíveis/não gerais. Propomo-nos definir um modelo de interacção genérico para SOLAP baseado em trabalhos prévios, extendendo-o de modo a incluir novas opções de visualização, componentes e casos; criar e apresentar uma arquitectura baseada em componentes para uma tal ferramenta, incluindo metamodelos descritivos, navegador de agregados de modo a melhorar o desempenho e um protocolo de comunicação; finalmente, desenvolver um protótipo de exemplo que implemente parcialmente as funcionalidades propostas, tendo em consideração boas práticas para o desenvolvimento de uma aplicação de fácil utilização, poderosa e flexível.

Agradecimentos

Ao Professor Doutor João Moura Pires, pela aposta em mim para este projecto e por tudo o que me ensinou, não só relativamente à área de investigação mas também no sentido de manter um espírito crítico, rigor e exigência.

Ao António Cruz e ao Pedro Lopes pelo tempo disponibilizado e apoio com questões técnicas durante o desenvolvimento da tese.

A todos os colegas e amigos de faculdade, que de um modo ou de outro me ajudaram ao longo deste percurso. Um agradecimento especial ao Francisco Costa, meu amigo e colega de trabalho ao longo de seis anos e que me ajudou sempre que precisei.

Ao pessoal da OCHC, por terem provado ao longo dos anos os amigos que são, por todas as aventuras e discussões que me ajudaram a crescer.

Aos meus pais, por tudo aquilo que sou - pela minha educação, por todos os valores que me souberam transmitir e por saber que estarão sempre presentes.

Table of Contents

Chapter 1 Introduction	1
1.1. Context and Motivation	2
1.1.1. Multidimensional Model	4
1.1.2. Data Granularity	4
1.1.3. Typical OLAP Operations	4
1.1.4. OLAP Modes	5
1.1.5. SOLAP Challenges	7
1.2. Objective and Contributions	7
1.3. Thesis Structure	8
Chapter 2 State of the Art	9
2.1. Geometrical Data Types in DBMS	10
2.2. Multidimensional Models for Spatial Data	12
2.3. SOLAP Systems	15
2.3.1. StatCan	16
2.3.2. The Spatial One	17
2.4. Clustering	19
2.5. Aggregate Navigators	20
2.6. Other Works on Spatial Performance	21
2.7. Conclusion	21
Chapter 3 Extended Interaction Model	23
3.1. Multidimensional Model Concepts and Extended Framework	24
3.2. Data Representation	26
3.2.1. Map Representation Function (mRF)	29
3.2.2. Support Table Representation Function (stRF)	30
3.2.3. Support Chart Representation Function (scRF)	31
3.2.4. Detail Representation Functions (dtRF and dcRF)	31
3.3. Interaction Cases Overview	31
3.4. Case 1: One Numerical Measure	32
3.4.1. Support Table	32
3.4.2. Map	33
3.4.3. Support Chart	34
3.4.4. Detail Table	37
3.4.5. Detail Chart	37
3.5. Case 2: Multiple Numerical Measures	38
3.5.1. Support Table	38
3.5.2. Map	38
3.5.3. Support Chart	42
3.6. Case 3: Semantic Attributes from Semantic Dimensions	43

3.6.1. Support Table	44
3.6.2. Map	45
3.7. Case 4: Semantic Attributes from a Spatial Dimension	46
3.7.1. Semantic Attribute at the Same or Higher Level than the Spatial Attribute	47
3.7.2. Semantic Attrib. at an Incomparable or Lower Level than the Spatial Attribute	48
3.8. Case 5: Spatial Attributes from the Same Dimension	48
3.8.1. Attributes from Same Hierarchy	49
3.8.2. Attributes from Different Hierarchies	50
3.9. Case 6: Spatial Attributes from Two Dimensions (or with no inclusion)	54
3.9.1. Roll-Up and Drill-Down Operations	57
3.9.2. Visualization Clustering	59
3.10. Spatial Measures	59
3.11. Visualization Complexity	60
3.12. Styles and Legend	62
Chapter 4 Architecture	65
4.1. Architecture Overview	66
4.2. SOLAP+ Server Architecture	67
4.3. Aggregates	69
4.3.1. Dimensions, Levels, Hierarchies and Aggregates Representation	70
4.3.2. Selecting Possible Aggregates	72
4.3.3. Selecting the Best Aggregate	72
4.4. SOLAP+ Client Architecture	72
4.5. Communication Protocol	75
4.5.1. List Cubes	76
4.5.2. Load Cube	76
4.5.3. Get Data	77
4.5.4. Get Distincts	79
4.6. Meta Model	80
4.6.1. Database Element	80
4.6.2. Mapservers Element	81
4.6.3. Multidimensional Element	82
Chapter 5 Implementation	87
5.1. Implemented Features	88
5.2. External Components	88
5.2.1. Spatially Compliant Data Server	88
5.2.2. Map Server	89
5.3. Implemented Components Overview	91
5.4. SOLAP+ Server	92
5.5. SOLAP+ Client	94
5.5.1. Backing Beans	95
5.5.2. Interface / Interaction	96
Chapter 6 Use Case and Validation	105

6.1. Presentation	106
6.2. Data Model	106
6.3. Interaction Examples	107
6.3.1. Example 1	107
6.3.2. Example 2	108
6.3.3. Example 3	109
6.3.4. Example 4	110
6.3.5. Example 5	111
6.3.6. Example 6	112
6.3.7. Example 7	113
6.3.8. Example 8	114
Chapter 7 Conclusions and Future Work.....	117
7.1. Conclusions.....	118
7.2. Future Work.....	119
References	121

Figure Index

Figure 1 - A Star Schema	6
Figure 2 - A Snow Flake.....	6
Figure 3 - Geometry Class Hierarchy from OpenGIS Features.....	11
Figure 4 - Types of spatial dimensions based on the levels' data type.....	12
Figure 5 - Classification of topological relationship for aggregation procedures (Malinowsky and Zimányi)	14
Figure 6 - StatCan interface.....	16
Figure 7 - The Spatial One base framework.....	17
Figure 8 - Alternative 1	18
Figure 9 - Alternative 2	19
Figure 10 - An aggregate navigator.....	21
Figure 11 - The extended framework	25
Figure 12 - Data representation on the extended framework.....	26
Figure 13 - Four combinations of different spatial attributes values leading to four partitions.....	27
Figure 14 - Example partition and respective vector object (vObj)	28
Figure 15 - Geometric Grouping Function.....	29
Figure 16 - Map and support table after SOLAP operations - The semantic attribute "Region" is presented on (...)	31
Figure 17 - An example support table displaying one numerical measure	32
Figure 18 - Example of representation of one single numerical measure using color and legend.....	34
Figure 19 - Chart showing "Population" as a numerical measure and ordered by a hidden function "Area" (...).....	35
Figure 20 - Support table ordered by hidden function "Area" (ascending order) (Source: INE).....	35
Figure 21 - Selected point and auxiliary distance lines.....	36
Figure 22 - Support table ordered by hidden function "Distance to point X" (ascending order) (Source: INE).....	36
Figure 23 - Chart showing "Crime Rate" as numerical measure and ordered using hidden function "Distance to (...)	36
Figure 24 - Support table at lower detail level (Source: INE).....	37
Figure 25 - Detail table at higher detail level (Source: INE).....	37
Figure 26 - Detail chart example (Source: INE)	38
Figure 27 - Example support table with two numerical measures.....	38
Figure 28 - Example of combination of charts and colors to represent four numerical measures	39
Figure 29 - Selected values for clustering algorithm (feature vectors).....	40
Figure 30 - Example support table for clustering.....	40
Figure 31 - Clusters discovered.....	41
Figure 32 - Clusters represented by color.....	41
Figure 33 - Support table and respective chart when there is no active selection.....	42
Figure 34 - Support table and respective chart when selecting a line.....	42
Figure 35 - Support table and respective chart when selecting a column (measure).....	43
Figure 36 - Support table and respective chart when selecting two lines and two columns (measures)	43
Figure 37 - An example support table with year attribute.....	44
Figure 38 - Support table for two semantic attributes from semantic dimensions and one numerical measure.....	44
Figure 39 - Support table for one semantic attribute from semantic dimension and two numerical measures.....	44
Figure 40 - An example map where a numerical measure is represented along with 3 distinct values (...)	46
Figure 41 - Support table after adding sA.....	47
Figure 42 - Map with numerical measure and semantic attribute representation.....	48
Figure 43 - Support table after adding store_type (sA)	48
Figure 44 - Hierarchy diagrams.....	49
Figure 45 - Roll-up and drill-down operations (Average used as aggregation function)	50

Figure 46 - Portuguese administrative subdivisions.....	51
Figure 47 - "Município" subdivision - lowest data granularity.....	51
Figure 48 - "Distrito" subdivision.....	52
Figure 49 - "NUTS II" subdivision.....	52
Figure 50 - Verifying the inclusion principle.....	53
Figure 51 - Intersections between "Distrito" and "NUTS II".....	53
Figure 52 - Support table.....	54
Figure 53 - Map (with legend).....	54
Figure 54 - Example of support table and respective map when representing two spatial attributes (...)	55
Figure 55 - Example of support table and respective map when representing two spatial attributes (...)	56
Figure 56 - Alternative representation.....	56
Figure 57 - Example using relationship-lines.....	57
Figure 58 - New support table with aggregated data.....	57
Figure 59 - New map representation.....	58
Figure 60 - New map representation using charts.....	58
Figure 61 - Support table and map representation without visualization clustering.....	59
Figure 62 - Support table and map representation with visualization clustering.....	59
Figure 63 - Table representation.....	61
Figure 64 - Tree representation.....	61
Figure 65 - Possible and recommended visualization styles.....	62
Figure 66 - SOLAP+ Logical Architecture.....	66
Figure 67 - SOLAP+ Server Architecture.....	67
Figure 68 - Server's Data Request Processor Architecture.....	68
Figure 69 - Levels organized in hierarchies.....	70
Figure 70 - Relationships between two levels.....	70
Figure 71 - The all level.....	71
Figure 72 - Two incomparable levels.....	71
Figure 73 - SOLAP+ Client Architecture.....	73
Figure 74 - Client's Data Processor Architecture.....	74
Figure 75 - Meta Model Root Element.....	80
Figure 76 - MapViewer's Architecture (Extracted from Pro Oracle Spatial).....	90
Figure 77 - Implementation Overview.....	92
Figure 78 - SOLAP+ Request Lifecycle.....	93
Figure 79 - Support Table Structure.....	96
Figure 80 - SOLAP+ Client Interface.....	97
Figure 81 - Interface with both tables minimized and hidden right panel.....	98
Figure 82 - Dimensions panel where "Emissao" dimension and "Poluente" level are expanded.....	98
Figure 83 - A typical SOLAP+ session with all panels expanded.....	99
Figure 84 - Map Control detail example.....	99
Figure 85 - Slice panel with an active slice.....	100
Figure 86 - Dialog box for creating/editing a slice.....	100
Figure 87 - Slice panel with an active slider.....	101
Figure 88 - Dialog box for creating/editing a spatial slice.....	101
Figure 89 - Dialog box for creating/editing a measure filter.....	102
Figure 90 - Defining measures and attributes order.....	102
Figure 91 - Removing elements from the analysis.....	103
Figure 92 - Loading a model/session.....	103
Figure 93 - Saving a session.....	104
Figure 94 - Data model for the use case.....	107
Figure 95 - Spatial hierarchies for dimension Installation.....	107
Figure 96 - Output example for one measure and a point spatial attribute.....	108

<i>Figure 97 - Output example for one measure and a polygon spatial attribute</i>	<i>109</i>
<i>Figure 98 - Output example for one measure and spatial objects generated by intersection.....</i>	<i>110</i>
<i>Figure 99 - Output example for two measures.....</i>	<i>111</i>
<i>Figure 100 - Output example when using header attributes</i>	<i>112</i>
<i>Figure 101 - Output example for two header attributes</i>	<i>114</i>
<i>Figure 102 - Generated maps when using a slider for attribute "Pollutant", alternating between "Cadmium", (...).</i>	<i>115</i>

Chapter 1

Introduction

This chapter presents the motivation, context and objective of this thesis, followed by the thesis structure

- 1.1. Context and Motivation2
- 1.2. Objective and Contributions.....7
- 1.3. Thesis Structure8

This chapter starts by presenting the context and motivations for the elaboration of this thesis, including some introductory information about OLAP and SOLAP. Next, it mentions the objective and contributions of this thesis and finally there is an overview of its structure.

1.1. Context and Motivation

Decision Support Systems (DSS) are a class of computer applications that helps human users with decision making activities on business and other organizations. It is considered that the concept of Decision Support Systems became an independent area of research in mid 1970's, evolving to a larger scale during the 1980's. At the conceptual level, Power differentiates DSSs in five different categories [1]:

- A model-driven DSS emphasizes access to and manipulation of a statistical, financial, optimization, or simulation model. Model-driven DSS use data and parameters provided by DSS users to aid decision makers in analyzing a situation, but they are not necessarily data intensive;
- A communication-driven DSS supports more than one person working on a shared task; (collaborative tools);
- A data-driven DSS or data-oriented DSS emphasizes access to and manipulation of a time series of internal company data and, sometimes, external data;
- A document-driven DSS manages, retrieves and manipulates unstructured information in a variety of electronic formats;
- A knowledge-driven DSS provides specialized problem solving expertise stored as facts, rules, procedures, or in similar structures.

Data-driven systems are designed in order to provide decision makers with useful information compiled from the organization's raw data that will help them choose between alternatives in a given situation. The decision-making process is no longer associated with a single central entity - it is spread across multiple users with different backgrounds and positions. This democratization led to the development of this kind of systems. Data-driven DSSs provide data reasoning operations such as comparison, pattern and outliers finding, tendencies, among others. These operations allow the user to improve his efficiency in problem solving and encourages exploration and discovery.

OnLine Analytical Processing (OLAP) is part of the family of Data-driven DSSs and Business Intelligence (BI). It is an approach for answering multidimensional analytical queries quickly and effectively. OLAP tools provide the user with an interaction standard and ease of use. All the usage benefits of a Decision Support System become available to heterogeneous users using typical OLAP operations to explore and analyze data, mainly slice and aggregation. Slice operations enable the user to focus on the essential data for each analysis. Aggregation is characterized by summarization (how data is summarized) and detail level (data granularity). The interaction is made by varying the summarization formulas/algorithms and detail levels, through roll-up and drill-down operations.

While OLAP meets all the requirements for an easy-to-use, fast and intuitive system, it has no support for spatial data visualization and analysis.

Geographic Information System (GIS) is a computer-based information system that enables capture, modeling, storage, retrieval, sharing, manipulation, and presentation of geographically referenced data.

Even though GIS is adequate for most applications that need to display or manipulate geographical data, it does not have the efficiency required from an analysis tool that deals with large amounts of data like OLAP, as Rivest and Bédard mention: “(...) despite interesting spatiotemporal analysis capabilities, it is recognized that actual GIS systems per se are not optimally designed to be used to support decision applications and that alternative solutions should be used” [2].

Nowadays there are large amounts of GIS data available. There are also many techniques to acquire it with increasing accuracy and precision. As more data with higher quality and reliability becomes available, new applications and tools are possible, including the usage of GIS capabilities as tools for analyzing spatial-related data.

It has been estimated that about 80 percent of all data stored in corporate databases are spatial data [3]. This is a clear indicator that spatial data analysis has a very wide application area. This factor along with the advance of GIS tools and DataBase Management Systems (DBMS) supporting GIS data, allowed the development of a new type of tools that combined OLAP and GIS called SOLAP (Spatial OLAP).

Rivest, Bédard et al. defined a SOLAP tool as “a visual platform built especially to support rapid and easy spatio-temporal analysis and exploration of data following a multidimensional approach comprised of aggregation levels available in cartographic displays as well as in tabular and diagram displays” [4]. These tools add spatial analysis and visualization to the standard OLAP interaction.

Several SOLAP tools have been implemented since then, however they are specific to a strict and pre-determined context and of complex usage. This complexity prevents the analysis and consequent decision making process to be made by different users, something that regular OLAP tools achieved through interaction standards and general ease of use. The work of Rosa Matias [5] and subsequent research by Marlene Vitorino and Rodolfo Caldeira [6] led to a generic, although limited, SOLAP interaction model.

A generic interaction model with extended analysis capabilities should be the foundation for standardization of SOLAP tools, opening a way for generic and easy to use applications to be developed instead of the complex and specific already existent.

1.1.1. Multidimensional Model

Using a multidimensional approach to model a system is enforcing simplicity [7]. A multidimensional model is composed of hypercubes, measures, dimensions, levels, hierarchies and attributes.

Dimensions are used to map and categorize data into entities that are relevant to the organization (such as product, date and warehouse in a retail sales model). Attributes are the characteristics of these dimensions (such as name, description and price in product dimension).

Each dimension can index data at several detail/aggregation levels. Hierarchies can then be defined using the levels of a dimension. For example, the *date* dimension can have *day*, *week*, *month*, *year* as a hierarchy and *month*, *trimester*, *year* as another.

A hypercube is a logical structure that has the organization's data indexed by the dimensions. Each of its cells represents an intersection from all the dimensions and contains measures – values associated to this relationship.

“The simplicity of the model is inherent because it defines objects that represent real-world business entities. Analysts know which business measures they are interested in examining, which dimensions and attributes make the data meaningful, and how the dimensions of their business are organized into levels and hierarchies.” [8]

1.1.2. Data Granularity

Regarding data granularity, Joe Oates says in DMReview [9]: “Granularity refers to the level of detail of the data stored fact tables in a data warehouse. High granularity refers to data that is at or near the transaction level. Data that is at the transaction level is usually referred to as atomic level data. Low granularity refers to data that is summarized or aggregated, usually from the atomic level data. Summarized data can be lightly summarized as in daily or weekly summaries or highly summarized data such as yearly averages and totals.”

Data granularity is a very important point in a data warehouse and data analysis. Typical OLAP operations use several data granularity levels in order to summarize and present data to the user. In order to aggregate data, an aggregation operator is needed for each measure: sum, average, etc for numerical measures; union, intersection, etc for spatial measures.

1.1.3. Typical OLAP Operations

As mentioned before, the main OLAP operations are related to either slicing or aggregating data.

Slice performs a selection on one dimension of a given cube. It selects a subset of that cube's elements where certain attributes match a given value or range of values. This operation allows the

analyst to focus on the relevant data. For example, if the analyst is interested in data from January 2009 only, he will apply a slice operation on the *time* dimension, selecting rows where the *year* attribute is equal to '2009' and *month* is equal to 'January'.

A user can filter data based on measure values, excluding elements that do not match the required criteria (Example: "Consider only sales of over 1,000 units"). Selection operations relative to measure values are also possible, such as Top/Bottom X elements. Along with slice, this enables queries such as "What were the top 10 selling products on Lisbon region in 2008?". Note that filtering is applied before data is aggregated, while top/bottom operations are applied after.

Data aggregation levels are changed using two operations: Roll-up and drill-down. Roll-up consists on summarizing or aggregating data by going up a hierarchy towards the least detailed level. Drill-down is the opposite operation, going down a hierarchy towards the most detailed level. By using drill-down operations, the analyst can understand the reasons behind summarized data. By using roll-up operations, the analyst can see the impact of the facts on a larger scale. As an example, consider a minimalist multidimensional model of a sales company containing two dimensions: *product* and *time*. The lowest data granularity represents the sale of a certain product at a certain time. To the store manager, a monthly report detailing the sales of each product in each *hour*, wouldn't be very helpful for his decision making process. Probably a report depicting sales per *product type* and per *week* (roll-up) would be more reasonable. However, if the analyst detects an abnormal sales amount on one of those weeks, he may perform a drill-down operation to explain the data at a lower level (*product*, *day* or *hour*) in order to detect the cause.

1.1.4. OLAP Modes

There are three different modes for storing cubes in a data warehouse: Multidimensional OLAP (MOLAP), Relational OLAP (ROLAP), and Hybrid OLAP (HOLAP).

In MOLAP data is stored in multidimensional cubes built for fast data retrieval. Aggregations are pre-calculated, meaning that response times are very good. Also because of this pre-calculation, it can only handle a limited amount of data. This can be countered by pre-calculating only part of the aggregations.

In ROLAP, the underlying data is stored in a relational database with multiple related tables. It's conceptual model gives the appearance of traditional OLAP operations such as slicing and roll-up/drill-down using the dimensions. It can handle large amounts of data but it has slower response times. It is also limited by the SQL functionalities when executing queries. To address this limitation there are extensions to the SQL language especially designed to deal with multidimensional models.

HOLAP is an approach that tries to combine the strengths of both MOLAP and ROLAP: It uses data cubes for summarized data and a relational model for drilling-down a hierarchy.

There are two main ways of arranging tables when using a ROLAP approach: Star-schema and snowflake schema. The star-schema (see Figure 1) is the simplest one and it consists of fact tables (or just one) referencing multiple dimension tables.

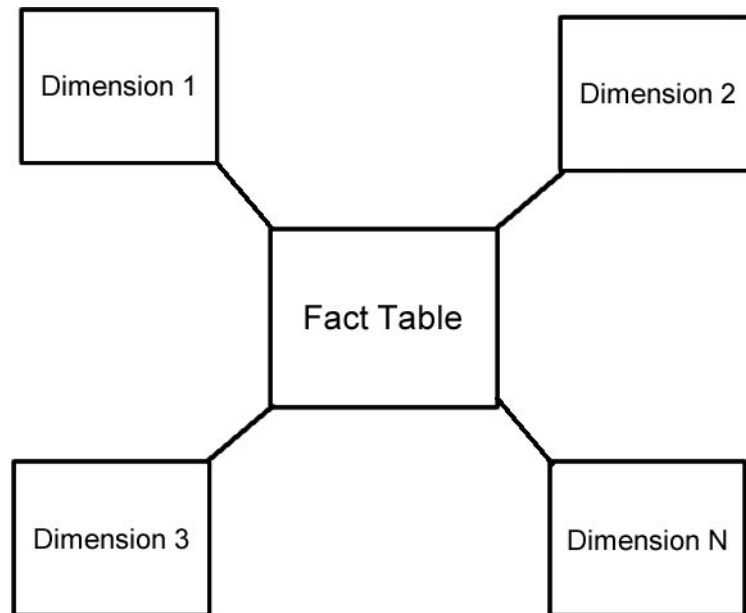


Figure 1 - A Star Schema

In the snowflake schema (Figure 2), the fact tables are also centralized and reference dimension tables, but dimensions are normalized into multiple related tables, resembling a snowflake.

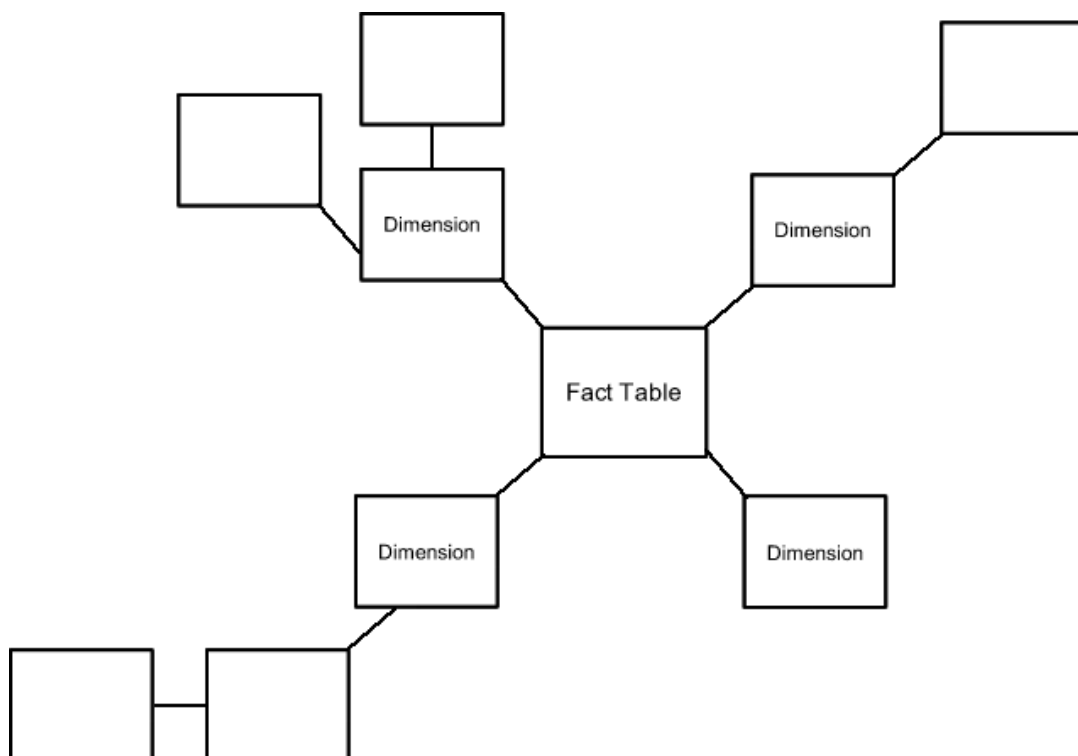


Figure 2 - A Snow Flake

1.1.5. SOLAP Challenges

SOLAP poses a lot of challenges. Some of those are shared with standard OLAP tools but it also has specific ones that were not an issue before.

Semantic and spatial data integration

Integrating spatial and non-spatial data is the base for a SOLAP tool. This integration poses several issues regarding how and where to store both types of data (data structures, databases vs GIS, pointers to spatial data, etc) in order to work together as a single tool.

Performance

A fast response to queries is a critical requirement for OLAP tools and its importance to SOLAP is not lower. Because SOLAP deals with spatial data and operations, an acceptable response time is harder to achieve, since those operations usually involve much more processing and thus more time to perform than those using regular data.

Ease of use

Being easy to use is a very important characteristic for an analysis tool, especially when it is designed to be used by different classes of users. This allows the user to focus on the actual data analysis rather than how to do it. Dealing with maps and other geographical information poses another obstacle to interface design and user interaction.

Standardization

Defining standards is an approach to support innovation, increasing productivity and assuring the quality of a product or group of related products. SOLAP standards are far from being reached - Lots of subjects are still under discussion or reasearch and several different methods are proposed for the same objective such as multidimensional models for spatial data, spatial measures, spatial hierarchies, interaction models, topological operators, etc.

1.2. Objective and Contributions

The objective of this thesis is to extend the generic SOLAP interaction model by including new components on the base framework and redefining the interaction and presentation on the different cases and scenarios. The initial models' limitations are presented as well as the new proposal to overcome them and extend its capabilities. An architecture for such a system and a prototype to exemplify some of the features proposed in the interaction model will also be developed.

With thesis we hope to achieve relevant contributions on:

- SOLAP Interaction Model - Extending and improving:
 - Visualization components
 - Interaction cases
 - Analysis options
- Metamodel for SOLAP - A new proposal that allows the usage of aggregates in order to improve performance
- Prototype - An application that partially implements the proposed features in the interaction model:
 - General
 - Easy to use
 - Flexible
 - Aggregate-aware

1.3. Thesis Structure

The content of this thesis is structured in seven chapters and one annex. Follows the listing and description of each chapter:

Chapter one (Introduction) presents the context, motivation and objective of this thesis, followed by its structure. In chapter two (State of the Art), the state of the art in Spatial OLAP and related technologies is presented. Chapter three (Extended Interaction Model) contains our proposal for an extended SOLAP interaction model, including the definition of new visualization components, options and interaction cases. Chapter four (Architecture) describes the architecture of our system, as well as the communication protocol and metamodels. Chapter five (Implementation) describes implementation decisions and details of our prototype. In chapter six (Use Case and Validation) we present several interaction examples of the implemented prototype. The thesis is concluded in chapter seven (Conclusions and Future Work), where conclusions are drawn and possible future work is presented. The annex includes XML Schemas for the communication protocol and metamodels. It is present in the digital version of this document only.

Chapter 2

State of the Art

This chapter presents the state of the art in SOLAP related works, presenting technologies, tools, multidimensional models for spatial data and previous interaction models.

2.1.	Geometrical Data Types in DBMS.....	10
2.2.	Multidimensional Models for Spatial Data	12
2.3.	SOLAP Systems	15
2.4.	Clustering	19
2.5.	Aggregate Navigators	20
2.6.	Other Works on Spatial Performance.....	21
2.7.	Conclusion.....	21

The introduction of spatial data can be a huge step in the decision making process of many organizations, even those already using conventional OLAP tools for their spatio-temporal analysis. With a plain OLAP tool, analysts don't have the features to visualize spatial data, perform spatial analysis or explore the data using a map. [4].

It has been estimated that about 80 percent of all data stored in corporate databases is spatial data [3]. Among these large numbers is a wide variety of areas onto which spatial analysis could be applied: Airline companies that work with plane routes, railroad companies dealing with tracks, phone operators registering calls' origins and destinations, pharmaceutical companies knowing the locations of hospitals, health centers, etc. [5].

Even though SOLAP is a relatively new area of research, there have been important works that are slowly leading to its applicability and standardization. Some researchers have managed to define many concepts related to spatial data and several multidimensional models have been proposed.

Next in this chapter we will give an overview on previous work on geometrical data types, operations and their implementation on conventional DBMSs that eventually allowed spatial data analysis to progress. The next section refers works on multidimensional models for spatial data - the base foundation for SOLAP tools, including spatial dimensions, hierarchies and measures. We will then present SOLAP systems/generic interaction models already existent and the possible usage of clustering in those tools.

2.1. Geometrical Data Types in DBMS

An entity is geographic if it has attributes that are associated with the earth coordinate system. These entities can have two kinds of components: semantic or descriptive attributes and spatial attributes. For example, a county has attributes that represent it's name, population, unemployment rate, etc (semantic) and attributes that depict its boundaries, cities, etc (spatial).

DataBase Management Systems were initially designed and optimized to deal with simple alphanumeric data. With the growth of GIS and the expansion of available spatial data, DBMSs started to include structures to handle this kind of data. It is required that the spatial module of a DBMS has the following properties [5]:

- Abstract types for spatial data - The possibility to create conceptual and logical models using spatial data
- Spatial query language - It is necessary that the SQL language is extended in order to handle spatial data
- Optimized techniques for query execution - Getting an acceptable response time when dealing with spatial data is much more difficult than using alphanumeric data. It is then

required that optimized mechanisms be developed and implemented in order to efficiently execute spatial queries.

The *Open Geospatial Consortium* (OGC) was founded in 1994 by a group of both public and private organizations with the intent of promoting development and usage of non-proprietary norms or standards on spatial data processing. To do this, the OGC develops and revises specifications where protocols and interfaces are described.

The *OpenGIS Simple Feature Specification for SQL* [10] is an OGC specification that contains a norm defining spatial objects in a class diagram (see Figure 3). From the abstract class *Geometry* derives *Point*, *Curve*, *Surface* and *GeometryCollection*.

Point is used to describe objects with zero dimensions, such as city locations on a map. *Curve* is used to represent objects like roads or rivers. *Surface* enables us to represent regions, areas or any other two-dimensional objects such as counties, natural reservations, etc. *GeometryCollection* allows more complex objects resulting from the combination of multiple objects by using spatial functions such as union, intersection, etc.

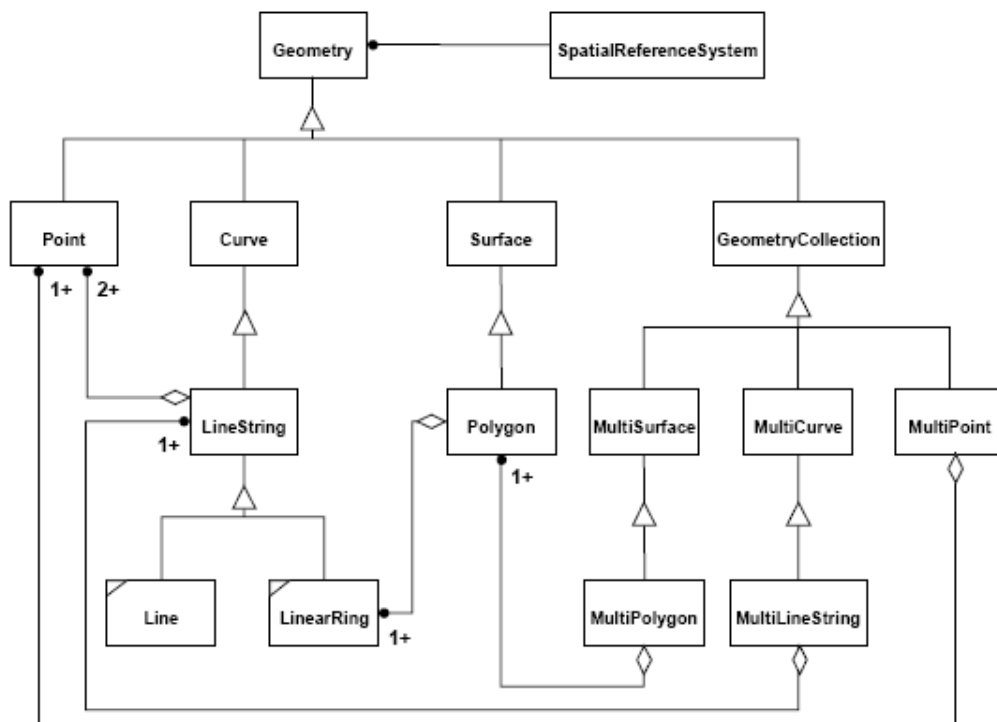


Figure 3 - Geometry Class Hierarchy from OpenGIS Features

Besides describing spatial objects, the *OpenGIS Simple Feature Specification for SQL* also describes multiple functions to test a geometry's properties, calculate numerical values based on a geometry, obtain a geometry using another, test topological and directional relationships between geometries

and distance functions. Some of these functions are general for all kinds of geometries (returning the description of a geometry type) but most of them are specific for each type - returning the X coordinate of a point (*Point*), returning the length of a curve (*Curve*), return the area (*Surface*), etc.

2.2. Multidimensional Models for Spatial Data

Even though multidimensional models have been used for data analysis for a long time and have solid foundations, the introduction of spatial data requires a rethinking of the concepts associated with it, mainly those related to dimensions, hierarchies and measures.

The usage of spatial dimensions allows the analyst to execute new and powerful slices using topological operators for example “Amount of fishing products sales in shops that are within a 5 km buffer of a river or lake”, “What is the crime rate percentage in districts that are at least 2 km away from a police station?”, “Return the top 10 plants on CO₂ emissions that are inside a protected region”, etc.

Several works [4][2][11][12] categorize spatial dimensions based on the type of data on each of its hierarchy levels: *a)* Non-geometric (when all levels contain only descriptive data); *b)* geometrical to non-geometrical (when the finest granularity levels contain geometrical data and the coarser granularity levels contain descriptive data) and *c)* fully geometrical (when all the levels contain geometrical data only). These types are depicted in Figure 4:

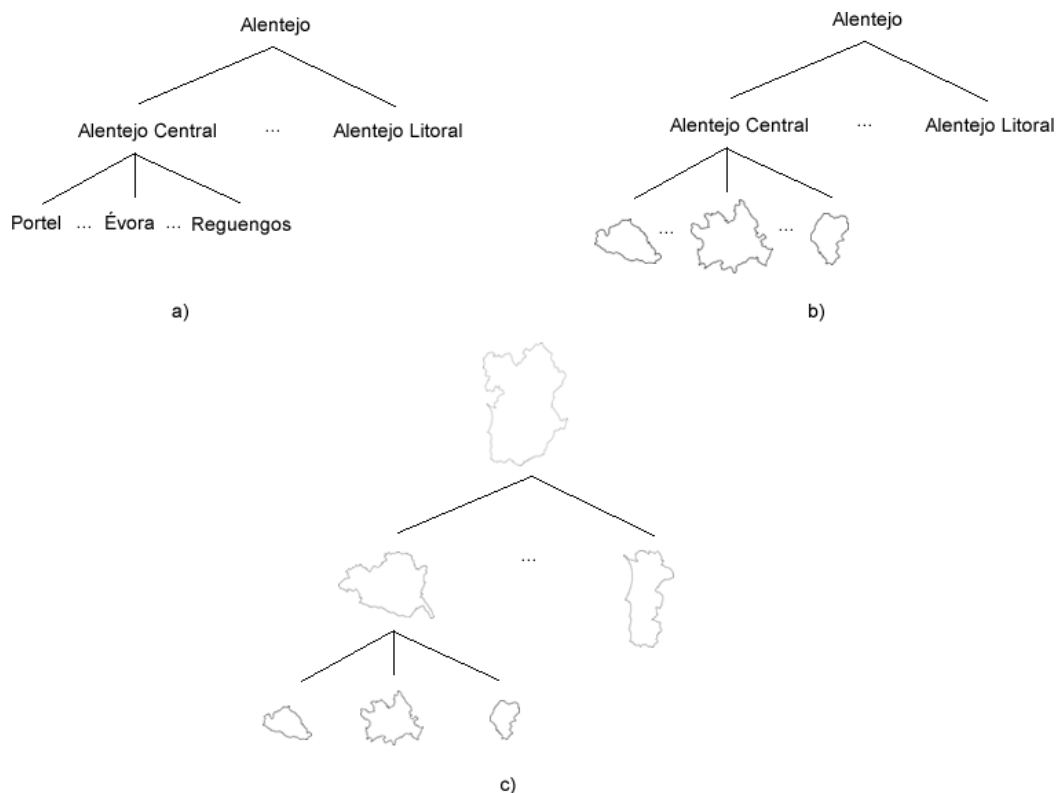


Figure 4 - Types of spatial dimensions based on the levels' data type

Fidalgo et al. [11] proposed a framework to be used as guidance for designing geographical dimensional schemas, considering geographical (only geographical attributes) and hybrid (geographical and conventional attributes) dimensions. It includes modeling guidelines in order to minimize redundancy of spatial data by normalizing *Composed Geographical Dimensions* (a dimension with multiple spatial attributes) into *Primitive Geographical Dimensions* (one that contains only one spatial attribute).

Spatial dimension hierarchies and the associated aggregation/summarization issues are subject to research in some works [13][11][14][15]. In particular, Malinowsky and Zimányi [13] defined different kinds of spatial hierarchies and gave a conceptual model for them. The summarizability problems that arises for some of those hierarchies is also studied. The conceptual model for the hierarchies is simple and effective, covering relationships between levels (denoted by *child* and *parent*) and respective cardinalities, spatial data types and topological relationships. Using real-world examples/applications, a generic classification of hierarchies (both spatial and non-spatial) is defined: *Simple*, *Non-Strict*, *Multiple* and *Parallel*.

Simple spatial hierarchies are those where the relationships between their members can be represented as a tree, for example a hierarchy depicting a relationship between stores, counties and states. *Symmetric hierarchies* have at the schema level only one path where all levels are mandatory. This implies that, at the instance level, the members form a tree where all the branches have the same length. *Asymmetric hierarchies* have only one path at the schema level but some of the lower levels of the hierarchy are not mandatory, causing the members at the instance level to generate a non-balanced tree. *Generalized hierarchies* contain multiple exclusive paths sharing some levels among them. At the instance level, each member belongs to only one path.

Non-strict spatial hierarchies are those that have at least one many-to-many cardinality. The previous hierarchy types can also be *non-strict* in addition to their already defined type. Most of the *non-strict* hierarchies arise when a partial containment relationship exists (only part of a river belonging to a county for example).

Multiple alternative spatial hierarchies have multiple non-exclusive *simple spatial hierarchies* sharing some levels. At the instance level, these hierarchies form a graph, since a *child* member can be associated with more than one *parent* member belonging to different levels.

Parallel spatial hierarchies arise when a dimension has multiple spatial hierarchies for different analysis criteria. In a *parallel independent hierarchy*, hierarchies do not share levels, as opposed to a *parallel dependent hierarchy*.

A method for identifying allowed topological relationships between spatial levels (considering those will be used for measure aggregation purposes) (Figure 5) is also presented. This method is

based on the result of the intersection between the geometric union of the spatial extent of *child* members and the spatial extent of their associated *parent* member.

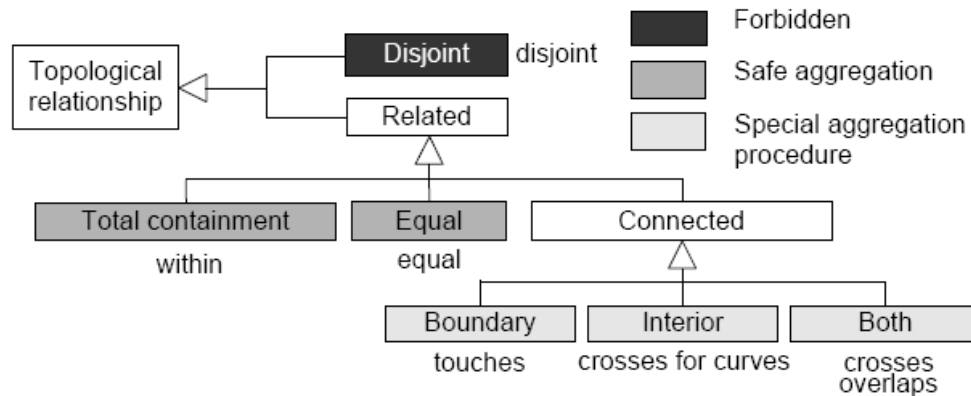


Figure 5 - Classification of topological relationship for aggregation procedures [13]

The same authors also proposed a conversion for this multidimensional model into an object-relational schema, based on SQL [16]. In this article, examples of spatial functions including aggregation functions can also be found.

The usage of spatial measures is a widely discussed subject far from having a standard defined, both on when and how to apply them. In the works that address this subject, there are three main approaches on how to define spatial measures:

1) Using the concept of geographical measure: an entity described by geometric, metric and thematic (textual) attributes. Complex aggregation functions must be defined for these entities since multiple attributes may need to be aggregated using different functions [17][18][14][19]. An example of a geographical measure using this approach would be the following element:

$\{Name = "Region1"; Shape = (PolygonCoordinates); Type = "A"; MeasureValue = "100"\}$

2) As a numerical value that is calculated by applying a spatial or topological operator like area or distance to an existent geometric object [16][2]. In this case, a spatial measure could be: $SUM(Region1.area() + Region2.area())$

3) A geometric shape obtained by combining two or more spatial attributes from spatial dimensions using functions like union, intersection, etc [16][2]. In [16] this is called a *spatial fact relationship* and it also considers the existence of numerical measures associated with it. A spatial fact relationship example would be the line resulting from the intersection between a road and a county.

Several problems related to spatial measure aggregations arise. Different aggregation functions are required, since there are different data types (spatial and semantic) with distinct meanings for each measure. To deal with the geometric part of a measure, a spatial aggregation function is needed in order to receive possibly multiple objects as input and return a single object as the output. This function can become especially complex and ineffective when the geometric objects are themselves calculated using spatial operations, causing a chain of time-costly geometric operations.

In conclusion, in most of the studied works, the geometric part of the spatial measure is an existent object or a new one computed based on existent ones from spatial dimensions – either a county, region or any other logical of administrative boundary. The only slight reference to an arbitrary geometric shape directly and solely related to a fact is present in a work by Matias and Moura-Pires [20]. In this work, an arbitrary spatial measure is proposed to represent the pollutant emission cloud as a polygon. The emission cloud is only related to the event of an emission, not to any spatial attribute or hierarchy from the dimensions.

2.3. SOLAP Systems

Bédard et. al. [4] defined and presented the features of a SOLAP system in three main areas: *Visualization of Data*, *Exploration of Data* and *Structure of Data*.

Under *Visualization of Data*, the authors defend the need for cartographic and non-cartographic displays, and their flexibility, either by using multiple displays at the same granularity or at different levels. The representation of multiple measures is another focused point as well as an interactive legend that allows the user to visualize numerical information at different levels while remaining at the same spatial level. The importance of using charts and guidelines for their construction is discussed by Tufte [21] and MacEachren [22]. Usage of context information is encouraged as it helps the user locate the displayed information.

Related to *Exploration of Data*, the authors explain the need for the availability of the data exploration operations in all display types, both cartographic and non-cartographic; availability of both topological and metric functions for data analysis and a timeline component for manipulating the temporal dimension. Also in this section is placed emphasis on calculated measures (measures calculated from other measures stored in the cube), filtering on the dimension members (slicing operations) and the display of significant data aggregates only, by eliminating irrelevant or empty cells from the aggregation process.

Under *Structure of Data* the need for tools to support multiple spatial dimensions is explained, as well as full-support for the standard geometric primitives. Automatic generalization of data sets to be displayed based on user actions such as zooming in/out, support for storage of historical geometric

data (changing over time) and support for different data sources, mainly the three OLAP modes (MOLAP, ROLAP, HOLAP) are also noted.

Even though this work provides many important guidelines for the development of SOLAP applications, it lacks detail on how to model and implement the proposed features throughout the article.

2.3.1. StatCan

StatCan [23] is a tool to analyze data from the Census of Canada. The usual census data is available such as age, gender, residence status, etc. The main SOLAP tools are available: slice, roll-up and drill-down. It includes several possible measures such as *population*, *births* and *deaths*.

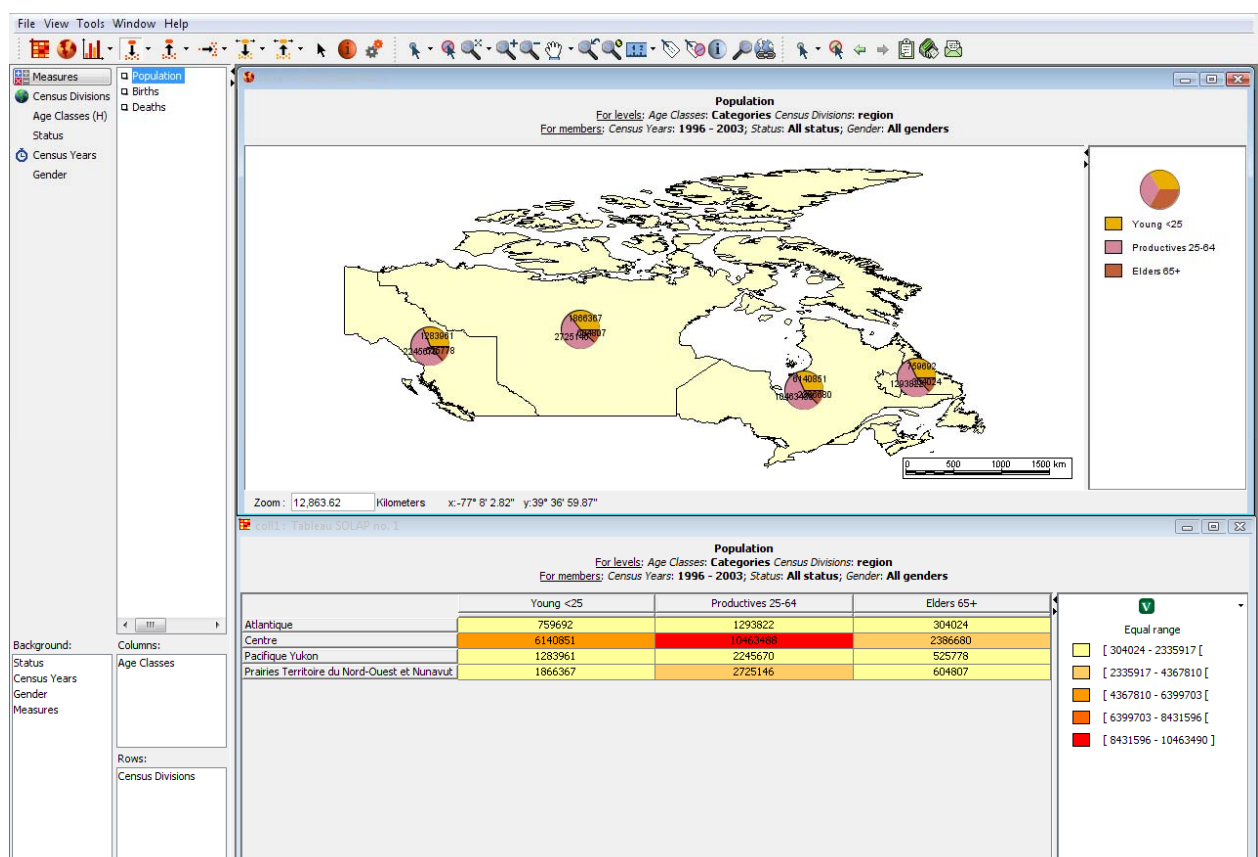


Figure 6 - StatCan interface

The interface (see Figure 6) contains a panel for the map and one for the table. At any time the analyst can request a chart based on the information presented. In that case, a new panel is opened and the related chart is presented.

The table works in a matrix format, adapting its rows and columns by dragging semantic attributes or numerical measures to the analysis. It allows expanding and collapsing data into lower or higher level spatial attributes, representing elements at different levels of granularity on the table.

The map offers the possibility to display multiple numerical measures by using charts associated with each region. It also has the basic features of zoom, spatial selection and labeling.

Slice operations are made by selecting one or more values for each attribute. Roll-up and drill-down operations can be made on either the map panel or the table panel. Applying those operations to one panel causes the other one to adapt its representation to the appropriate granularity level and executing the necessary summarization.

2.3.2. The Spatial One

Vitorino, M. and Caldeira, R. developed a generic SOLAP interaction model [6] based on a previous interaction model by Matias, R. [5]. They defined both the components and the behavior of the system based on several generic interaction cases from the user. Those interaction cases are the base for any specific SOLAP application that may use the generic model defined. This model features the usage of spatial slices mentioned before, namely the possibility to slice by distance to a certain point, by topological operator (such as “contained in”, “touches”, ...) and by the top/bottom elements (ex.: “the 10 nearest factories to river X”).

The base framework proposed has three main components: The map, the support table and the detail table (Figure 7).

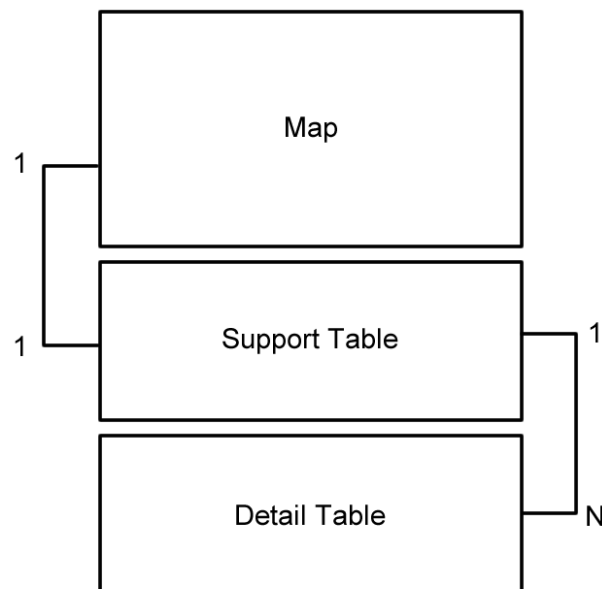


Figure 7 - The Spatial One base framework

The map is where reference maps, spatial objects and other SOLAP information are presented to the user.

The support table is used to show the base data from which the map representation depends. This includes attributes from dimensions (semantic attributes) and measures.

The detail table is a tool to provide the user with more in-depth analysis. In order to maintain the 1:1 relationship between the support table and the map, drill-down operations resulting in certain detail levels should not be expressed in the support table, because they can't be represented in the map (The 1:1 relationship need is explained further on). In these cases, the detail table is used to analyze that data at a lower aggregation level.

Any one of the three components can also show numerical measures.

The 1:1 relationship between the map and the support table is an important property that should be kept in any situation. It helps the analysis process as to each spatial object on the map, there is a corresponding line in the support table.

To justify this property we have to look at the alternatives to a 1:1 relationship. Those are:

1. Multiple lines in the support table to one spatial object on the map (Figure 8)

In this case, we would have multiple rows associated with each spatial object. Selecting a spatial object in the map would select multiple rows, possibly spread along the table, making it difficult for the user to relate the information in the table to the objects on the map.

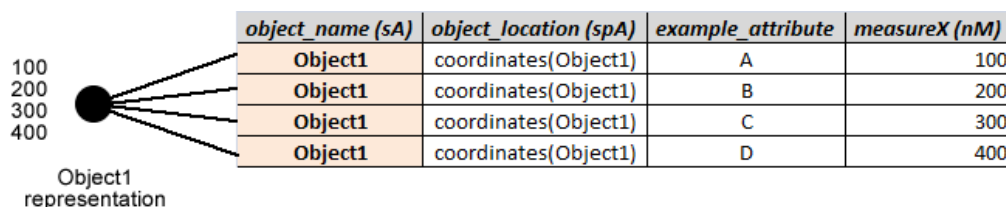


Figure 8 - Alternative 1

2. One line in the support table to multiple spatial objects in the map (Figure 9)

On the other hand, having one measure (one support table line) associated to multiple spatial objects would prevent the analyst from realizing the contribution of each spatial object to the global value while looking at the map.

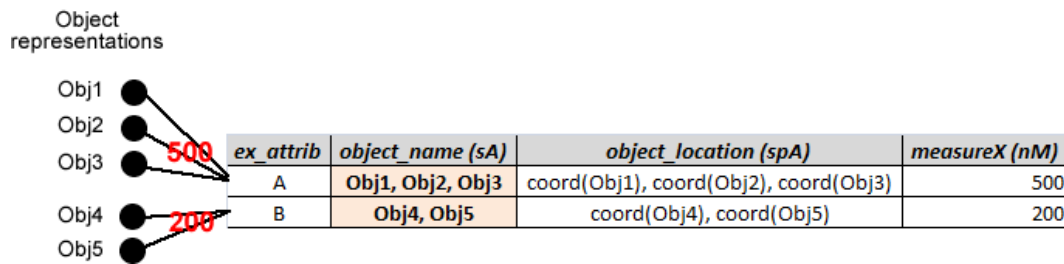


Figure 9 - Alternative 2

Data in the support table is usually aggregated at a certain level. However, if the user requires a more in-depth analysis, he/she can use the detail level, where the same data is shown but at a lower level of aggregation. This means that for each line in the support table, we will possibly have multiple lines in the detail table that represent the same data, thus the 1:N relationship.

Using this framework, the authors defined a generic interaction model contemplating the following cases:

1. One numerical measure
2. Multiple numerical measures
3. Using semantic attributes from semantic dimensions
4. Using semantic attributes from spatial dimension
5. Using spatial attributes from the same dimension
6. Using spatial attributes from different dimensions

2.4. Clustering

Even though clustering is not directly related to the SOLAP area, it will have a very important role in our interaction model proposal, used at several levels and with different goals, as we will explain further on.

Data analysis is a process that requires ease of use and fast response times so that the user can maintain his train of thought. The amount of data presented to the user and its level of detail is also extremely important: Too much or too detailed data presented at one time to the user will prevent him from drawing conclusions.

When dealing with SOLAP tools, these aspects have to be considered not only for the support table but also for the respective map visualization. If we try to represent a lot or too close spatial objects on the map, we can end up with a cluttered map where the analysis process becomes impossible.

To address this problem we propose clustering or merging nearby spatial objects into groups represented by a single entity on the map. These groups have no semantic meaning and the spatial objects they contain are similar only on their geographical position, so that would be the only feature to be used as an input parameter on the clustering algorithm.

CLARANS [24] is a k-medoid clustering algorithm which identifies circular clusters based on a distance function. Each point is assigned to the cluster to which the distance to its respective k-medoid point has the smaller value. This algorithm requires an input parameter K – the number of clusters to be generated. Since we are only considering grouping by geographical proximity, circular clusters identified by CLARANS seems to be adequate in this case.

Clustering techniques can also be used to discover similar elements based on their associated measures. Note that in this case the geographical location is not relevant, only the measures related to them.

DBSCAN [25] is a density-based clustering algorithm that relies on notions of core and border points to determine which points belong to which cluster. Since it is density-based and not distance-based like CLARANS, it generates clusters with arbitrary shapes and doesn't need an input parameter specifying the number of clusters to generate. In order to identify clusters based on feature vectors, DBSCAN is more accurate (as it does restrict itself to circular clusters) and with better response times than CLARANS. It is adequate for clustering needs that are not location/geography based, such as identifying similar elements based on multiple measures.

More recent works [26][27] aim to define techniques and algorithms for clustering spatial objects. Those can be interesting for simplifying visualization when dealing with polygons.

2.5. Aggregate Navigators

According to Ralph Kimball [28], providing a proper set of aggregate records is the best way to improve performance of a data warehouse. These aggregate records are summarized records obtained from the original data. Since they are summarized, operations over them are faster than having to compute all the original records.

There is, however, the problem of the user knowing when and which aggregate to use with a certain query. Hardcoding the end-user applications with knowledge of the aggregates removes the database administrator's flexibility to add and remove aggregates, as the applications would have to be recoded. The answer is to use an Aggregate Navigator [29][28][30] - a component that stands between the end-user applications and the DBMS, capturing the SQL queries and converting them to use the best possible aggregate (if available):

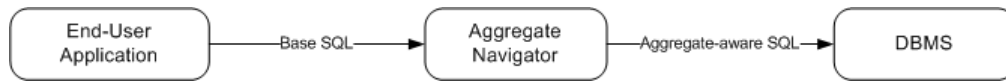


Figure 10 - An aggregate navigator

The SQL generated by the end-user application always refers the base multidimensional model's fact tables and dimensions - it is the responsibility of the aggregate navigator to convert that base SQL into aggregate-aware SQL, where aggregate tables are referenced in case a usable aggregate is present.

2.6. Other Works on Spatial Performance

Even though this issue is not within the scope of our work, we believe a few considerations on this are important. Response time is a major concern in any analysis tool: Users expect to be able to see the results of their requested operations in timely fashion, allowing him to maintain his train of thought. An already crucial issue to a standard OLAP tool is taken to an even higher level when dealing with SOLAP, since manipulating spatial data is usually much more time costly.

Stefanovic et. al. [31] proposed time efficient methods for computing spatial measures, merging spatial objects and selecting pre-aggregated cuboids based on an adapted greedy algorithm for spatial databases.

Papadias et. al. [32] presented a method that uses aggregation trees based on Minimum Bounding Rectangles to discover arbitrary spatial hierarchies. These newly discovered hierarchies are then added to a greedy algorithm that will decide which pre-aggregates to materialize.

Spatial operations are very time costly, both on I/O and CPU. In order to minimize the cost of polygon amalgamation operations, an algorithm that reduces the number of polygons used in those operations to the absolute minimum was developed [33]. This algorithm works by identifying the internal polygons which would not be used in the amalgamation process. Two methods can be used for this identification, polygon adjacency or spatial indexes.

2.7. Conclusion

In order to achieve standard levels for SOLAP, there is still much to be done. Even though there are several proposals for most of the issues presented, many of those proposals are not compatible with each other, leading to the development of specific applications rather than generic interaction models or tools. Spatial measures is a particular case of an issue with different and incompatible proposals which has a lot of open possibilities to explore.

The presented generic interaction models by Matias [5] and Vitorino and Caldeira [6] opened many possibilities, however, many of the proposed solutions are far from optimal and flexible.

Chapter 3

Extended Interaction Model

This chapter presents the extended SOLAP interaction model, namely its components and interaction cases

3.1.	Multidimensional Model Concepts and Extended Framework.....	24
3.2.	Data Representation	26
3.3.	Interaction Cases Overview	31
3.4.	One Numerical Measure	32
3.5.	Multiple Numerical Measures.....	38
3.6.	Semantic Attributes from Semantic Dimensions	43
3.7.	Semantic Attributes from a Spatial Dimension.....	46
3.8.	Spatial Attributes from One Dimension.....	48
3.9.	Spatial Attributes from Two Dimensions (or with no inclusion).....	54
3.10.	Spatial Measures.....	59
3.11.	Visualization Complexity	60
3.12.	Styles and Legend.....	62

Our work proposal is to extend the generic SOLAP interaction model previously presented [6]. We aim to redefine the behavior on several interaction aspects as well as adding new components to the visualization and new interaction cases that were not considered before.

This chapter starts by introducing a few concepts related to our multidimensional model and presenting our extended framework components followed by an explanation on how data is represented. An overview of the considered interaction cases is then presented. Finally, each of the interaction cases is exposed in detail.

3.1. Multidimensional Model Concepts and Extended Framework

We consider two types of dimensions in our model, depending on what kind of data they contain: *Semantic dimensions* and *Spatial dimensions*.

A dimension is constituted by levels. These levels are organized within one or more hierarchies inside the dimension and each level can contain multiple attributes. Levels can be compared with each other, being higher or lower in case they are on the same hierarchy or incomparable if they are on different hierarchies. This issue is addressed on section 4.3 when studying aggregates.

A dimension is semantic if it has *semantic attributes* only. If the dimension has at least one *spatial attribute*, then it is considered a spatial dimension.

Definition 1: Semantic attribute (sA) is a textual or numerical attribute from a dimension.

Definition 2: Spatial attribute (spA) is an attribute from a dimension containing a coordinate or group of coordinates that represent a spatial object. Each spatial attribute has an associated semantic attribute used to represent it in a table or another textual component. The associated semantic attribute to a spatial attribute is represented as **sA(spA)**.

Definition 3: Spatial object (spO) is a GIS element that is either a point (ex.: client location), line (ex.: road) or polygon (ex.: county). It is not necessarily connected to a dimension.

Our interaction model considers the two most common multidimensional model architectures: Star Schema and Snow Flake. The fact table is where the relationships between the dimensions as well as possible measures are present. *Measures* can also be of two kinds: *Numerical measures* and *spatial measures*.

Definition 4: Numerical measure (nM) is a numerical value associated with a fact and stored in the fact table.

Definition 5: Spatial measure (spM) is a coordinate or group of coordinates that represent a spatial object. It is associated with a fact and stored in the fact table. Note that these are *arbitrary spatial measures* only, as mentioned in section 2.2.

The extended framework we propose (see figure below) has two new components and some changes to the previous ones on how information is presented. The new components are the support chart and the detail chart.

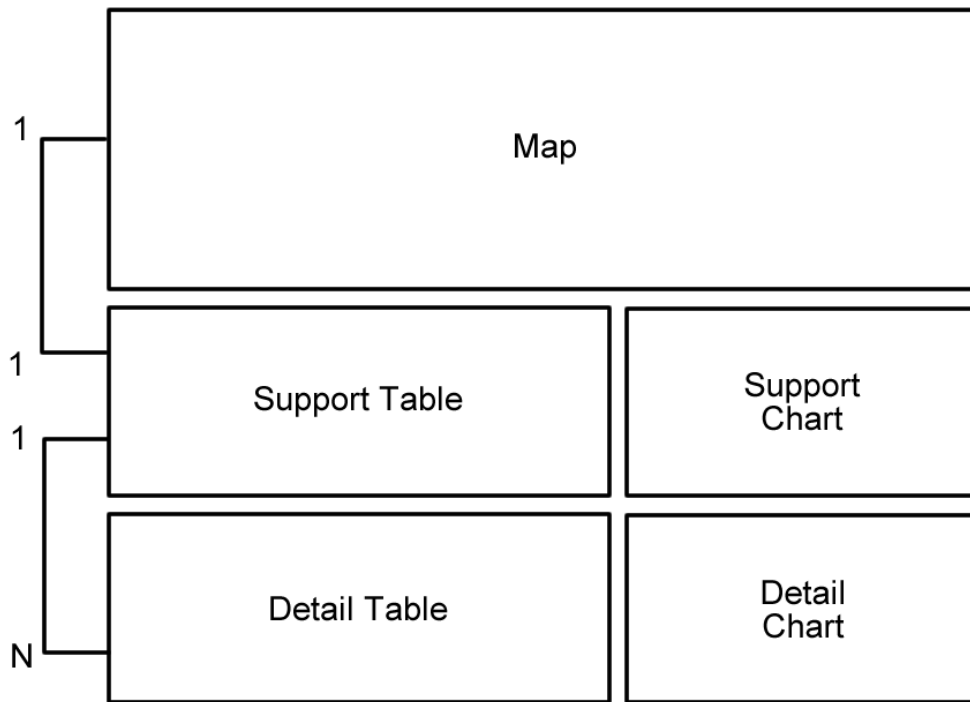


Figure 11 - The extended framework

The **map** is used to present reference maps, spatial objects and other SOLAP information to the user.

The **support table** is used to show textual and numerical data at the same granularity as the map. This includes attributes from dimensions and measures. The **detail table** is a tool to provide the user with more in-depth analysis, usually at a finer granularity level by applying drill-down operations to the data in the support table.

The **support chart** is a tool used to visualize data referring to the support table. The **detail chart** is related to the data in the detail table.

These components will be addressed in detail as we expose the interaction cases.

3.2. Data Representation

The framework components are used to represent data in different ways and at different granularity levels at the same time. The map, support table and support chart present data at a coarser level while the detail table and detail chart do it at a finer level, as we can see in Figure 12:

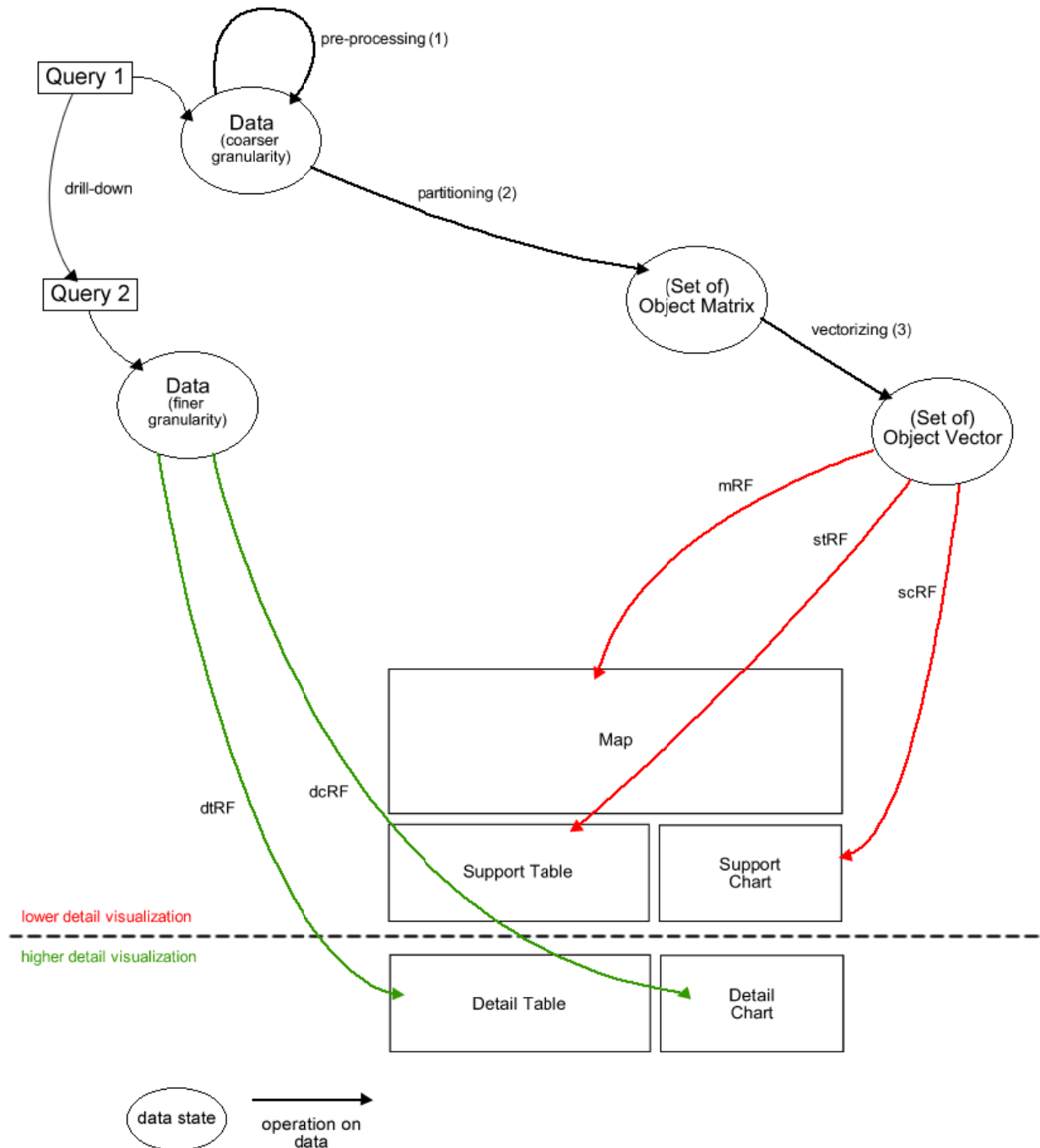


Figure 12 - Data representation on the extended framework

The two required data sets (low and high detail) are obtained by executing database queries where *Query 2* is the result of a drill-down operation over *Query 1*. This data comes in the form of *rowsets* - a set of rows where each one refers to a combination of attributes (both spatial and

semantic) defined in an SQL GROUP BY clause. These rows also contain the selected measure values for the respective attribute combination.

In our interaction model, we introduce the possibility of representing multiple numeric values associated with a single spatial object, while maintaining the one to one relationship between the map and the support table. This is done by merging multiple rows that refer to the same spatial object into one. In order to do this, the rowset returned is subject to several transformations:

Pre-processing (1) is optionally applied to the query data set (Clustering for visualization purposes (discussed below) is an example) by creating virtual rows from the original ones.

The next step is *partitioning (2)* the rowset returned after executing the query: n groups will be created, where n is the number of combinations between spatial attributes' values in the query. Each one of these groups will represent a spatial object. See Figure 13 for an example of partitioning:

spA 1	spA 2	sA	M1	M2
□	○	B	3	5
□	○	C	7	10
□	△	A	10	20
□	△	B	10	15
□	△	C	5	10
○	□	A	12	10
△	○	C	2	8

Figure 13 - Four combinations of different spatial attributes values leading to four partitions

Each group is then converted into a vector (3) that represents an object (*vector object* or *vObj*)(Figure 14):

spA 1	spA 2	sA	M1	M2	
□	△	A	10	20	
□	△	B	10	15	
□	△	C	5	10	

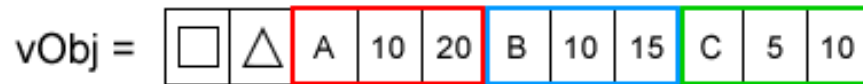


Figure 14 - Example partition and respective vector object (vObj)

These transformations are not required for the detail components - the vector objects are directly obtained from the lower granularity rowset. Having multiple rows referring the same spatial object is not a problem in these cases because for each element in the higher level components (map, support table, support chart) there are N elements in the lower level components (detail table and chart), allowing a more in-depth analysis.

Each component has a specific *representation function*. A **representation function (RF)** is a function that maps a set of vector objects ($\{vObj\}$) to a visual representation on a certain component, namely *map RF (mRF)*, *support table RF (stRF)*, *support chart RF (scRF)*, *detail table RF (dtRF)* and *detail chart RF (dcRF)*:

$$(\{vObj\}) \xrightarrow{RF} \text{Visual Representation}$$

According to the component, the output from the respective representation function will be a spatial object on the map (Map), a line on a table (Support table, Detail table) or a chart (Support chart, Detail chart). All these functions take a sub-set of the vectors' elements as input. While presenting the interaction cases, the input of the representation functions will be adapted to the given scenarios.

Generally, a representation function maps a set of vector objects to a component. The visual representation of each vector object may depend not only of the object being represented but also on the values or properties of all the other vector objects on the same set. For example, if we want to create visual elements representing arbitrary groups of objects based on their geographical proximity (using clustering techniques), the entire set of objects has to be considered.

There are however, two particular cases of the representation function: The first case is when each vector object can be represented individually, without the need for any other information, for instance, a row in the detail table. This representation function could be defined as:

$RF(\{vObj\}) = \sum_i rf(vObj_i)$, where $\sum rf()$ represents the successive call of a representation function for each vector object.

The second particular case is when each vector object can be represented individually but requiring some *context information*; an example would be a bar in a chart - it can be represented individually but it would require a maximum and minimum scale value, both of which are context variables, not information from the object itself. This representation function could be defined as:

$RF(\{vObj\}) = \sum_i rf(vObj_i, summary(\{vObj\}))$, where $\sum rf()$ represents the successive call of a representation function for each vector object and $summary(\{vObj\})$ is the context information, previously extracted from the global set.

As mentioned before, there are three high granularity and two low granularity representation functions, one for each component: Map, Support Table and Support Chart as higher level visualization components and Detail Table and Detail Chart as lower level components. The following sub-sections describe those representation functions.

3.2.1. Map Representation Function (mRF)

The different types of parameters taken by the representation functions determine different spatial object properties on the map. The selected sub-set of the spatial attributes in the rowset determines the location of the respective spatial object on the map. If only one spatial attribute is used, then its location is determined by the spatial attribute's plain coordinates. If more than one spatial attribute is used in the representation function, an additional spatial object is computed using a Geometric Grouping Function (ggF) that takes multiple spatial attributes as input. Then, the new spatial object's coordinates determine its location on the map. ggF can be defined as: $(spA_1, \dots, spA_i) \xrightarrow{ggF} (spObj)$, where i is the number of spatial attributes used in the representation function (see Figure 15).



Figure 15 - Geometric Grouping Function

ggF varies depending on the interaction case: It can be a spacial intersection, union, difference or any other function that generates a spatial object from multiple spatial objects. Representing only one spatial object on the map for each element ensures the 1:1 relationship between the support table and the map explained above.

The selected sub-set of the rowset's semantic attributes and measures determines the color, pattern, shape, associated chart or any other characteristic regarding that spatial object.

In the simplest case we would have one spatial attribute representing a point and one measure. A single point would be marked on the map (at the same coordinates as the ones from the spatial attribute) with a certain color or size (determined by the measure value).

The most complex case would be multiple spatial attributes that represent polygons and multiple measures and/or semantic attributes. In this case we have more than one spatial attribute, therefore *ggF* would be used, taking those spatial attributes as input and returning a representative spatial object. Then, the representation function takes all the measures/semantic attributes and determines the new object's color, associated chart, pattern, etc.

Slicing and filtering the data for analysis is a crucial step in any OLAP/SOLAP interaction. However, even after executing those operations, the map representation when dealing with multiple spatial attributes can still be complex. To minimize this problem we propose using clustering algorithms and grouping/aggregating data sets based on their spatial location. Those ad-hoc groups do not have any semantic meaning other than being geographically near each other. Groups can be created or decomposed by user interaction or automatically related to zoom variation. This group generation is also the responsibility of *ggF*, even though it's results have to be available to the other components, in order to maintain the consistency among the represented data.

3.2.2. Support Table Representation Function (stRF)

stRF maps a set of vector objects to a table representation. In most cases, this representation function falls into the first particular case presented above, where it can be decomposed into multiple representation functions, each of them mapping a vector object individually. This is not the case when using visualization clustering for example. In this case we would need the general *RF* approach of mapping the entire set of vectors.

An example of the second particular case would be if we wanted to set the background color of each line of the support table with the same color used for the respective spatial object in the map. We can still map each vector object individually, but we will require some context information: the color to use (Figure 16).

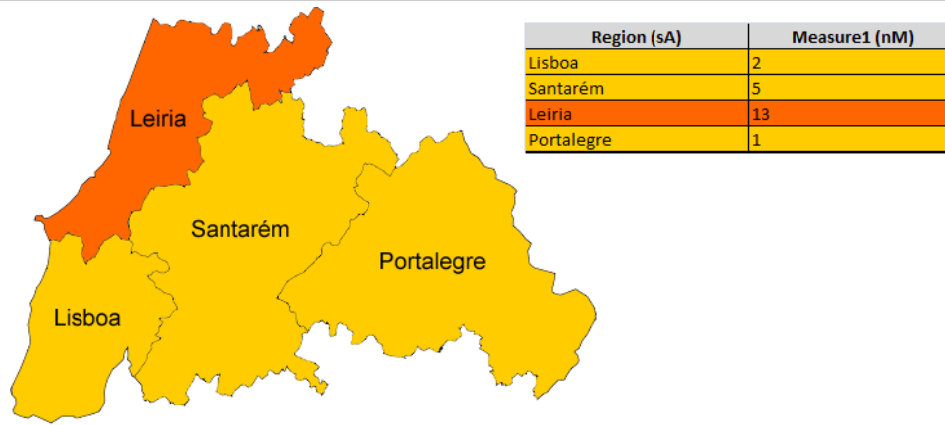


Figure 16 - Map and support table after SOLAP operations - The semantic attribute "Region" is presented on the map to establish a connection to the respective line in the support table

It is also very important that every spatial attribute *spA* has an associated semantic attribute *sA*. This helps the user establish a connection between those elements in the map and the support table. As mentioned before, these associated semantic attributes are represented by *sA(spA)*.

3.2.3. Support Chart Representation Function (scRF)

scRF's objective is to map a set of vector object's to a visual chart representation. Just like *stRF*, it usually can be decomposed into multiple calls to a representation function for each vector object, either with some context information (such as chart style, size, etc) or without it.

3.2.4. Detail Representation Functions (dtRF and dcRF)

The representation functions for the detail table and chart components are similar to *stRF* and *scRF*, keeping in mind that the set of vector objects taken as input is not the same as the one used by the higher components - it is at a lower granularity, more detailed level.

3.3. Interaction Cases Overview

There are several representation cases that we will address using our framework. Each of the following cases start from a base scenario where interaction is applied.

Base scenario: One spatial attribute and corresponding semantic attribute is present:

1. One numerical measure
 $vObj = \{spA, sA(spA), nM\}$
2. Multiple numerical measures
 $vObj = \{spA, sA(spA), nM_1, \dots, nM_n\}$

Base scenario: One or more numerical measures are present in addition to a spatial attribute and corresponding semantic attribute:

3. Semantic attributes from semantic dimensions

$$vObj = \{spA, sA, nM_1, \dots, nM_n, sA_1, \dots, sA_i\}$$

4. Semantic attributes from spatial dimension

$$vObj = \{spA, sA, nM_1, \dots, nM_n, sA_1, \dots, sA_i\}$$

5. Spatial attributes from the same dimension

$$vObj = \{spA_1, sA_1, \dots, spA_i, sA_i, nM_1, \dots, nM_n\}$$

6. Spatial attributes from different dimensions

$$vObj = \{spA_1, sA_1, \dots, spA_i, sA_i, nM_1, \dots, nM_n\}$$

The last (7th) group of interaction cases is related to spatial measures.

Even though they are not mentioned in the following interaction cases, the spatial slice possibilities used in the works of Vitorino and Caldeira [6] and mentioned in section 2.3.2 are adopted - distance to a point, topological operators and neighbours (top/bottom).

Some of the given examples use data from *Instituto Nacional de Estatística* (INE).

3.4. Case 1: One Numerical Measure

The first interaction case depicts the representation of a single numerical measure associated to a spatial attribute, where $vObj = \{spA, sA(spA), nM\}$. $sA(spA)$ is the semantic attribute associated with the spatial attribute spA .

3.4.1. Support Table

In the spatial dimensions, for each spatial attribute spA there is a corresponding semantic attribute sA . The support table has a column for the semantic attributes and one column for the respective numerical measure. The spatial attribute is not represented in the support table (as you can see in Figure 17), therefore: $stRF = f(sA(spA), nM)$.

Region (sA)	Measure1 (nM)
Leiria	13
Santarém	5
Castelo Branco	0
Coimbra	15
Setúbal	8
Viseu	7

Figure 17 - An example support table displaying one numerical measure

Each line can be colored with the same color from the map's legend, in order to more easily establish a relation between a line and the corresponding spatial object in the map. The map's legend component is addressed further on.

3.4.2. Map

As it was pointed out before, the visualization has a one to one relationship with the support, meaning that for each line in the support table, there is a corresponding spatial object on the map.

The representation function for each $vObj$ is then defined as $mRF = f(spA, sA(spA), nM)$. This function generates a spatial object on the map based on the input values - in this case the spatial object's properties (such as size, color, etc) are conditioned by the measure value and the type of object (point, line, polygon).

spA determines each spatial object's location, while nM determines the color, shape, size, or any other characteristic that can graphically map a value.

As an example, we will consider spA is a polygon that represents a Portuguese region. That said, mRF can't use neither size nor shape as those are relevant for the location. Color has been chosen because it is easy to recognize for the user and it also offers more personalization options than the pattern approach for example. For each spatial object $spObj$, mRF sets its color to a value that is determined by the respective numerical measure nM (using a mapping function).

A legend is used to map the colors to the measure values (or ranges of values). In the figure below, we have one possible result of applying mRF to 18 vector objects of $\{spA, sA(spA), nM\}$, where the spatial attributes represent polygons, resulting in 18 instances of $spObj$ differentiated by color. The semantic attribute has not been used in this example, however they are often used to label the spatial objects on the map, establishing a direct connection to the respective table rows.

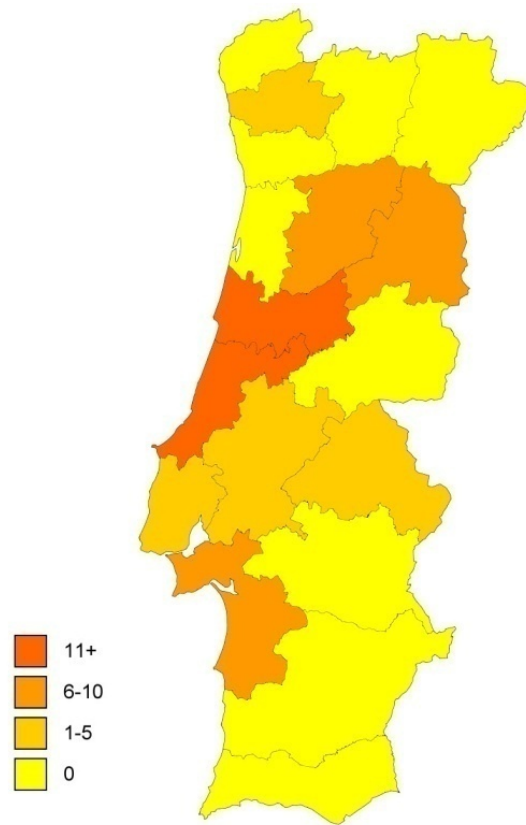


Figure 18 - Example of representation of one single numerical measure using color and legend

3.4.3. Support Chart

In the extended model, associated with our support table we have a support chart. This component's function varies with the type and number of attributes and measures being used for the analysis. The support chart representation function ($scRF$) takes a set of vector objects as input ($vObj = \{spA, sA(spA), nM\}$) to produce the chart.

In this case, semantic associated attributes ($sA(spA)$) are represented along one axis and numerical measures values (nM) along the other. Besides the possibility to order the chart bars by alphabetic order (semantic attribute), ascending/descending measure value or user-defined, we can also order the chart by the return values of a function not shown in the chart (hF). The values returned by hF are represented on the support table.

This function takes one spatial attribute as input and returns an orderable, real number.

Let $hF = f(spA)$, so we have: $(spA) \xrightarrow{hF} (n)$, where n is a real number.

Examples for that function would be "Area" or "Distance to point X".

Ordering the chart by one of these functions would allow the user to possibly draw certain conclusions on the relation between the measure values and the function values for each element, for example “Smaller regions tend to have a larger population” (Using “Area”), as seen in Figure 19 and Figure 20:

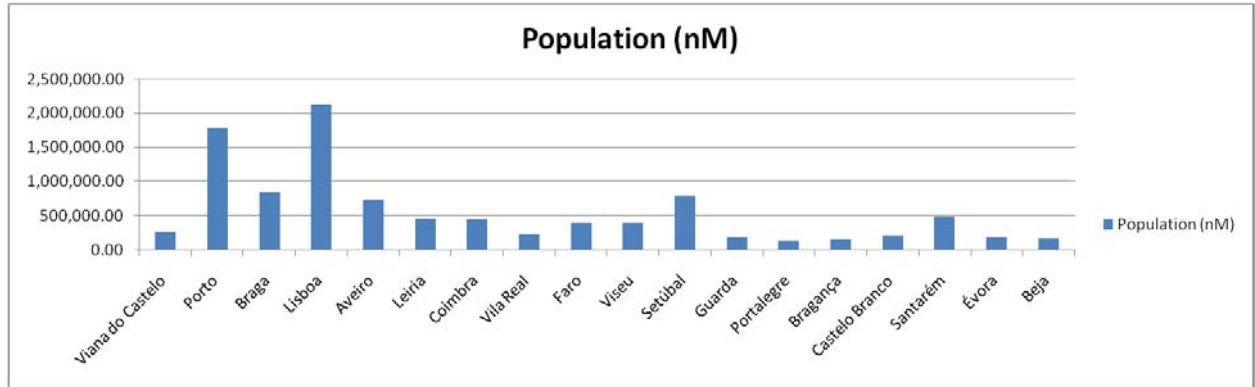


Figure 19 - Chart showing "Population" as a numerical measure and ordered by a hidden function "Area" (Source: INE)

Region (sA)	Population (nM)	Area (Sq. Km) (hF)
Viana do Castelo	250,273.00	2,255
Porto	1,781,826.00	2,395
Braga	831,368.00	2,673
Lisboa	2,135,992.00	2,761
Aveiro	713,578.00	2,808
Leiria	459,450.00	3,515
Coimbra	441,245.00	3,947
Vila Real	223,731.00	4,328
Faro	395,208.00	4,960
Viseu	394,927.00	5,007
Setúbal	788,459.00	5,064
Guarda	173,716.00	5,518
Portalegre	127,018.00	6,065
Bragança	148,808.00	6,608
Castelo Branco	208,069.00	6,675
Santarém	475,344.00	6,747
Évora	173,408.00	7,393
Beja	161,211.00	10,225

Figure 20 - Support table ordered by hidden function "Area" (ascending order) (Source: INE)

In the example, the support table's data and order is the same as the chart, but that is not mandatory.

As another example, “Crime rate significantly reduces with distance from point X” (Using “Distance to point X”)(See Figure 21, Figure 22 and Figure 23):

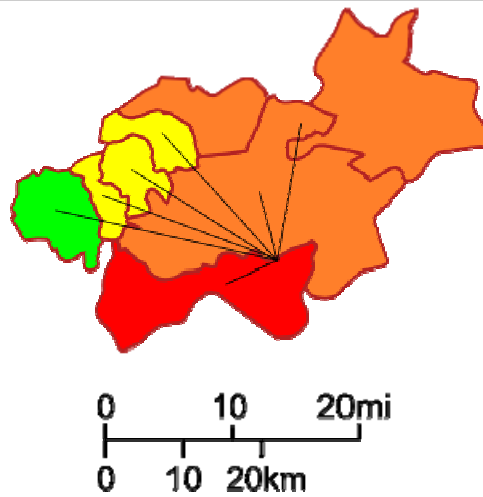


Figure 21 - Selected point and auxiliary distance lines

Name	Crime Rate (‰)	Dist. Point X (Km) (hF)
Setúbal	58.7	9.74
Palmela	46.9	11.16
Montijo	45.3	20.22
Moita	39.2	29.18
Barreiro	38.7	29.94
Alcochete	40.6	30.17
Seixal	33.7	33.63

Figure 22 - Support table ordered by hidden function "Distance to point X" (ascending order) (Source: INE)

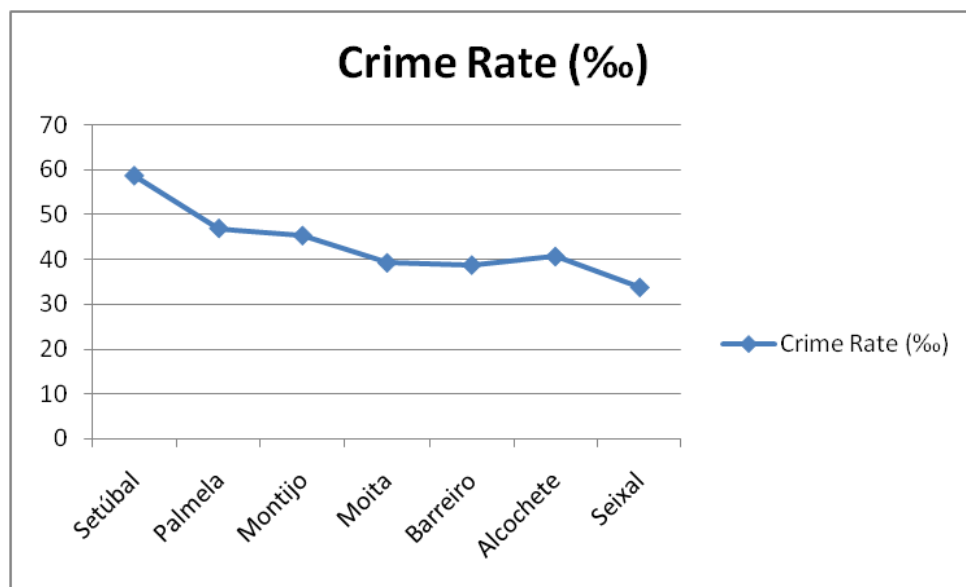


Figure 23 - Chart showing "Crime Rate" as numerical measure and ordered using hidden function "Distance to point X" (Source: INE)

3.4.4. Detail Table

The detail table is a component used to explain data in greater detail (lower granularity) or without the need to change the current visualization on the map or support table. It can display extra semantic attributes and numerical measures not present in the map/support table. Depending on the levels to which those attributes belong, it can cause a drill-down on the data.

The rows presented in the detail table vary - it can present rows related to all the elements on the support table or only the ones related to user selection.

In the example below ($dtRF = f(sA, nM)$), depicted by Figure 24 and Figure 25, we have the support table displaying data at NUTS II level. When the user selects “Alentejo”, lower granularity data used to produce that aggregation is shown in the detail table:

NUTS II	Telephones per 100 inhabitants
Norte	26.26
Centro	30.50
Lisboa	36.68
Alentejo	31.61
Algarve	41.78

Figure 24 - Support table at lower detail level (Source: INE)

NUTS III	Telephones per 100 inhabitants
Alentejo Litoral	30.10
Alto Alentejo	35.00
Alentejo Central	32.55
Baixo Alentejo	30.42
Lezíria do Tejo	30.53

Figure 25 - Detail table at higher detail level (Source: INE)

3.4.5. Detail Chart

The detail chart is a direct chart representation of the data in the detail table (see Figure 26 for an example). That data can be represented using different kinds of charts, depending on the analysis needs and/or semantic meaning of those values. Just like the support chart, the input for the respective representation function ($dcRF$) is a set of object vectors.

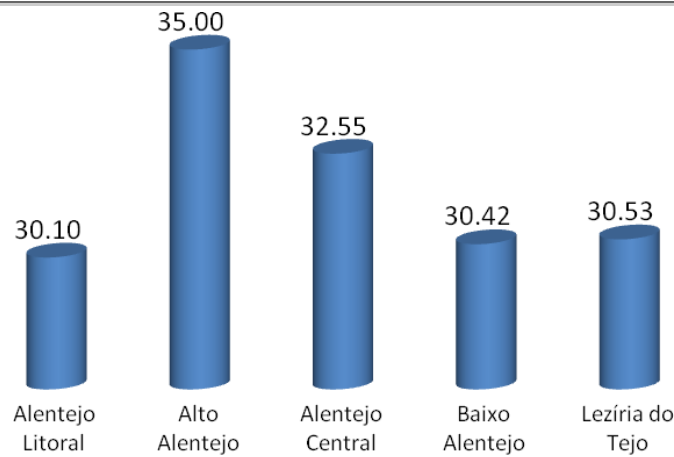


Figure 26 - Detail chart example (Source: INE)

3.5. Case 2: Multiple Numerical Measures

The second interaction case is characterized by the usage of multiple numerical measures in the analysis, therefore: $vObj = \{spA, sA(spA), nM_1, \dots, nM_n\}$

3.5.1. Support Table

The support table for multiple numerical measures has the same structure as the one for a single measure, with the exception that it has a column for each measure (Figure 27). $stRF = f(sA(spA), nM_1, nM_2)$.

Region (sA)	Measure1 (nM1)	Measure2 (nM2)
Leiria	13	520
Santarém	5	125
Castelo Branco	0	541
Coimbra	15	195
Setúbal	8	269
Viseu	7	301

Figure 27 - Example support table with two numerical measures

3.5.2. Map

In this interaction scenario, the representation function will use the spatial attribute spA to determine the spatial object's location, just as before. Now, multiple measures will have to be represented for each spatial object. We can associate nM_1 with color, nM_2 with pattern, nM_3 with line style, and so on. However, this approach is limited to a couple or a few numerical measures, because human perception is lost with too many similar graphical elements (for example: pattern, line type and line weight can be easily confused).

In order to simplify and make perception easier for the user, additional measures are represented in a chart for each spatial object. The kind of chart used depends on the values to compare and type of analysis desired. If we consider that a chart can represent up to 3 or 4 measures, this approach would cover the large majority of spatial analysis needs.

As an example, consider we want to represent 4 numerical measures named nM_1 , nM_2 , nM_3 , nM_4 . The map representation function would be $mRF = f(spA, sA(spA), nM_1, nM_2, nM_3, nM_4)$. In the example below (Figure 28) we have associated nM_1 with color and the remaining measures with a pie-chart. Representation styles, such as colors and charts are a complex matter and will be discussed in section 3.12.

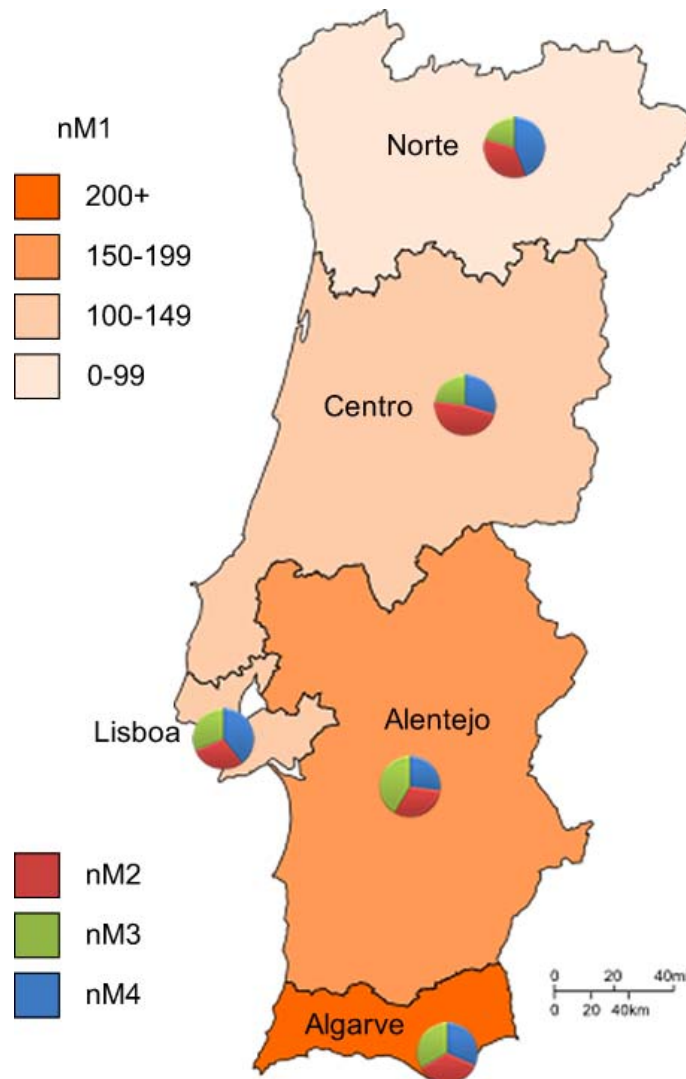


Figure 28 - Example of combination of charts and colors to represent four numerical measures

3.5.2.1. Discovery of Spatial and Measure Values Relations

Besides representing numerical measure on the map, it would be interesting to find relations between the numerical measures and the areas/regions on the map. To do this, we propose to use

the numerical measures' values from each line as feature vectors in a clustering algorithm (Figure 29). This will create clusters of similar elements based on their measure values.

Region	Financial Income of Enterprises (€)	Employment Rate (%)
Minho-Lima	23,632,598	45.6
Cávado	112,662,914	57.4
Ave	218,511,497	60.3
Grande Porto	2,024,235,904	56.4

Feature Vectors

Figure 29 - Selected values for clustering algorithm (feature vectors)

After the clusters are determined, to each geometrical element is associated a color (or any other property) on the map and conclusions may be drawn.

In the following example (Figure 30), we have two numerical measures associated with a number of regions: "Financial Income of Enterprises (€)" and "Employment rate".

Region	Financial Income of Enterprises (€)	Employment Rate (%)
Minho-Lima	23,632,598	45.6
Cávado	112,662,914	57.4
Ave	218,511,497	60.3
Grande Porto	2,024,235,904	56.4
Tâmega	142,876,671	55
Entre Douro e Vouga	615,296,379	59.2
Douro	17,875,308	42.8
Alto Trás-os-Montes	15,486,898	39.6

Figure 30 - Example support table for clustering

We take those values associated with each region and use a clustering algorithm to determine similar counties (Figure 31):

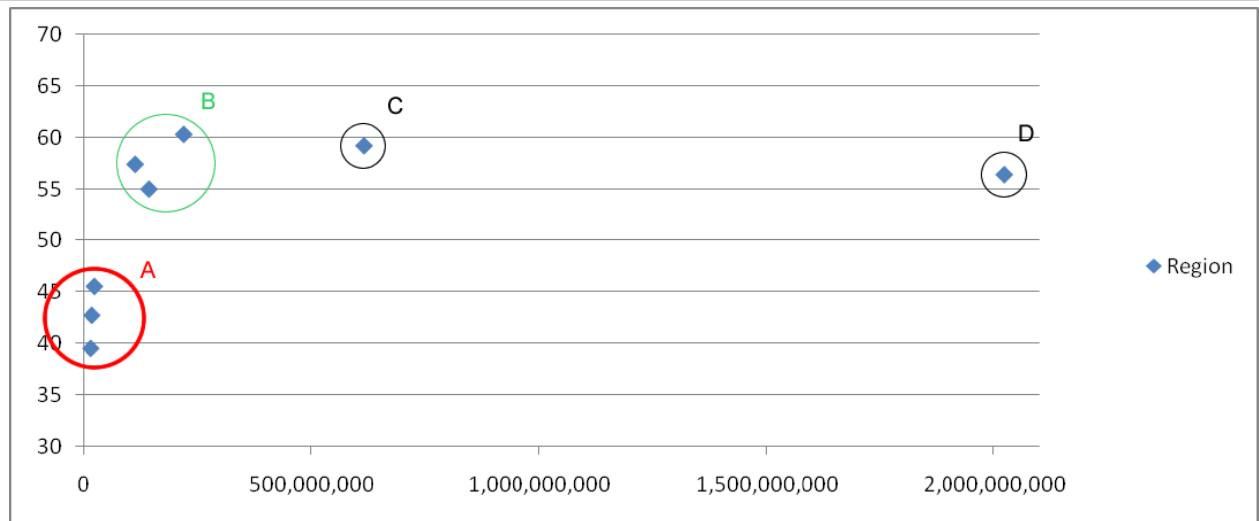


Figure 31 - Clusters discovered

We clearly notice 4 distinct clusters, denoted by A, B, C and D.

C and D are isolated points and won't be considered for further analysis. Cluster A is constituted by the regions named "Douro", "Alto Trás-os-Montes" and "Minho-Lima". Cluster B is constituted by the regions named "Cávado", "Ave" and "Tâmega".

By representing on the map the clusters to which each region belongs (Figure 32) we realize that South-Western regions are similar regarding those two aspects, as well as North and North-Eastern are similar among them, establishing a geographical relation with the numerical data.

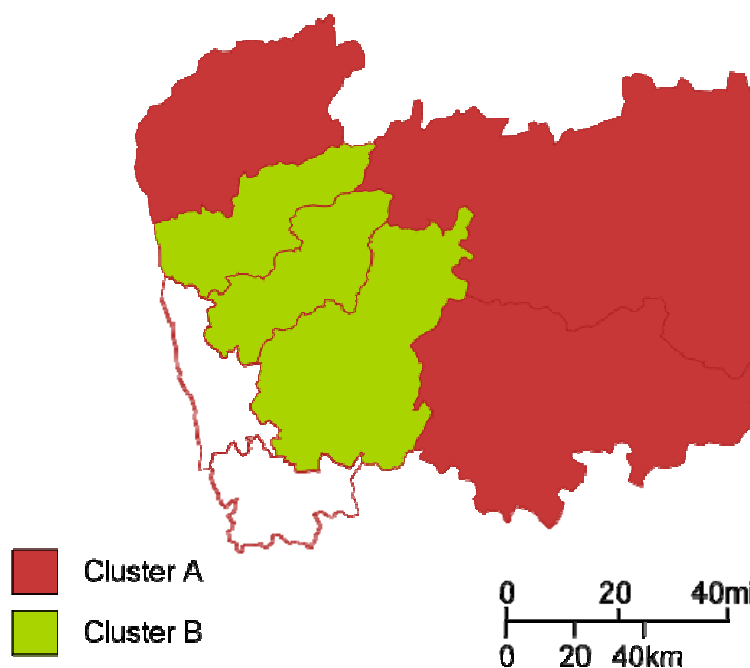


Figure 32 - Clusters represented by color

This simple example only took two numerical measures into consideration, however, most clustering algorithms can work with a larger number of measures.

3.5.3. Support Chart

The support chart can be used to visualize information relative to all lines, a single line, one column or a combination of lines and columns, depending on user selection.

If no specific lines or columns are selected in the table, the chart can be used to represent an overview of the whole information (Figure 33):

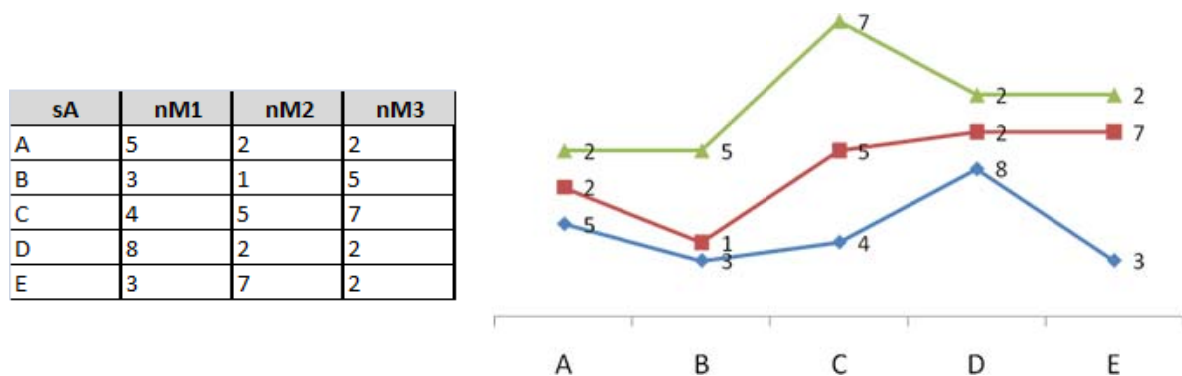


Figure 33 - Support table and respective chart when there is no active selection

If the user selects a line, the information presented on the chart are all the measure values related to the selected element (Figure 34):

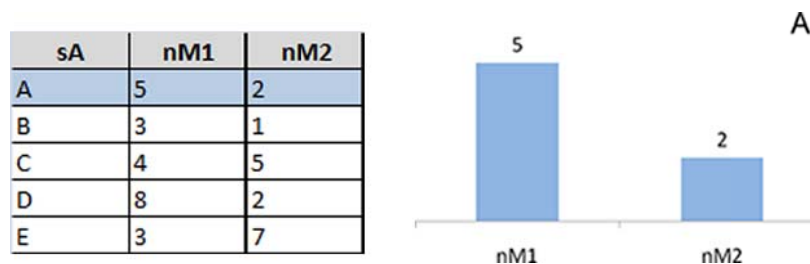


Figure 34 - Support table and respective chart when selecting a line

If the user selects a column, the information displayed in the chart are the values of that measure for all the elements (Figure 35):

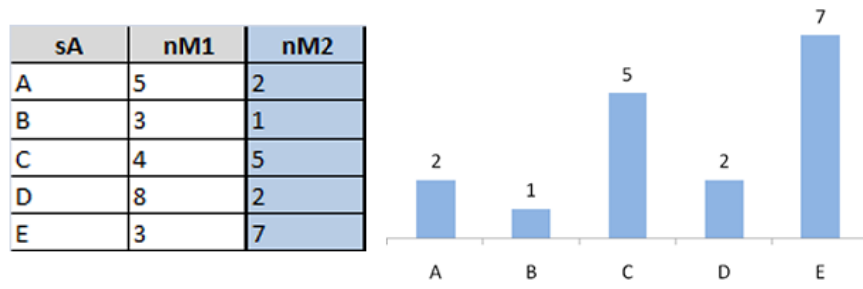


Figure 35 - Support table and respective chart when selecting a column (measure)

The user can also select a combination of lines and columns, detailing which measures from which elements he wants to see represented in the chart (Figure 36):

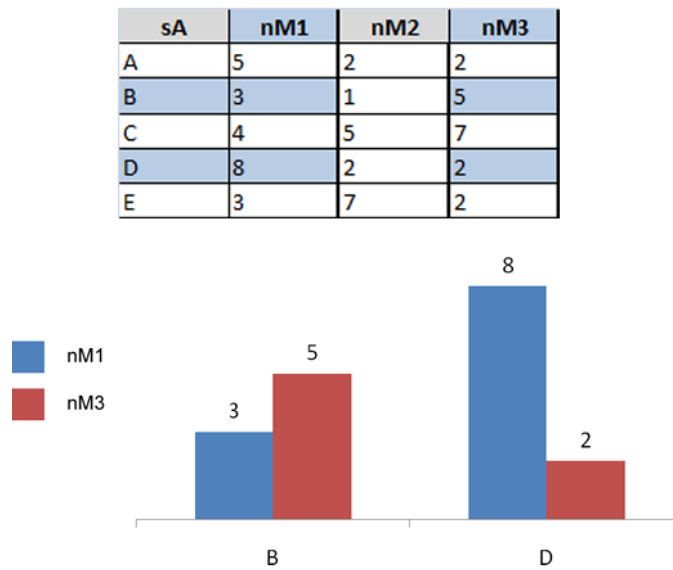


Figure 36 - Support table and respective chart when selecting two lines and two columns (measures)

Note: In order to correctly visualize multiple measures in the same chart (Chart A) all of them should be represented using the same scale, otherwise the bars' size would be disproportionate. More information about representation styles and charts can be found in section 3.12.

3.6. Case 3: Semantic Attributes from Semantic Dimensions

In this case we define how data is represented when using semantic attributes (sA). Note that the base scenario is one or more numerical measures (nM_i) in addition to a spatial attribute (spA) and respective associated attribute $sA(spA)$. The object vectors in these cases have the following structure: $vObj = \{spA, sA(spA), nM_1, \dots, nM_n, sA_1, \dots, sA_i\}$. sA_i are the new semantic attributes.

Consider in the examples below that *store_location* is the spatial attribute spA (representing a point), *sales* is the numerical measure nM and *store_name* is the semantic attribute associated with *store_location*.

year and *product_type* are semantic attributes from semantic dimensions.

3.6.1. Support Table

As we add a new semantic attribute (*year* (*sA*)) to the analysis, the support table representation function receives a new argument: $stRF = f(sA(spA), sA, nM)$. After it is applied to every *vObj*, the following table is generated (Figure 37):

	<i>year</i>		
<i>store_name</i>	2006	2007	2008
Store1	120	135	149
Store2	110	112	101
Store3	80	75	99
Store4	93	97	92
Store5	112	110	105

Figure 37 - An example support table with year attribute

“2006”, “2007” and “2008” are possible values from *year* (*sA*). Values inside the table cells are *sales* (*nM*) for each *store_name* and *year*. You can also see a couple of examples of support tables when adding more semantic attributes or numerical measures in Figure 38 and Figure 39.

	<i>product_type</i>					
	A			B		
	<i>year</i>			<i>year</i>		
<i>store_name</i>	2006	2007	2008	2006	2007	2008
Store1	70	90	87	50	45	62
Store2	60	62	56	50	60	55
Store3	72	67	75	8	8	24
Store4	60	59	70	33	38	22
Store5	92	101	80	20	9	25

Figure 38 - Support table for two semantic attributes from semantic dimensions and one numerical measure

	<i>year</i>					
	2006		2007		2008	
<i>store_name</i>	<i>sales</i>	<i>expenses</i>	<i>sales</i>	<i>expenses</i>	<i>sales</i>	<i>expenses</i>
Store1	120	101	135	102	149	107
Store2	110	111	112	98	101	98
Store3	80	89	75	99	99	102
Store4	93	65	97	78	92	79
Store5	112	100	110	101	105	98

Figure 39 - Support table for one semantic attribute from semantic dimension and two numerical measures

It is important to note that this is a new approach considering the previously defined model by Vitorino and Caldeira [6]: Using that approach we could not represent multiple numerical columns associated with each spatial object, as it would disrupt the 1:1 relationship between the map and support table. Using pivot-like tables (also called *matrix view*) allows us to keep this relationship while representing multiple numerical columns both on the table and on the map.

This matrix view of the table is used in some applications such as pivot tables or the presented StatCan application [23]. However, while in the pivot tables and StatCan it is permitted to use attribute values on the table lines, our interaction model does not allow this approach, as it would disrupt the one to one relationship between the table and the map. Because of this support table design, adding too many semantic attributes and/or numerical measures, would make it extremely complex to the user. The topic of visualization complexity is discussion in section 3.11.

3.6.2. Map

In order to represent the measure values associated with each value of a semantic attribute sA , the previous interaction model [6] used a slider with multiple maps. In each of those maps, a possible value for sA was chosen, and measure values related to that attribute were represented.

Even though the slider with multiple maps approach allows us to view the changes to the measures across several values of a semantic attribute sA , it doesn't give the analyst a global view of those values, making it difficult to draw conclusions based on non-adjacent values. It is, however, useful for orderable attributes, especially from data/time dimensions since it allows the user to view the evolution of values through time.

The most commonly used group of attributes of this kind are probably those from date/time dimensions (Year, trimester, month, ...), so we're going to use them on the examples below.

The proposed method to allow the comparison among several measure values associated with the respective attribute's value is to add/adapt a chart to each spatial object on the map. The chart should have the measure values on one axis (Y) and the orderable attribute's values on the other (X). That way a user can easily compare several values at the same time without the need to scroll back and forth.

This method assumes that there is a relatively low number of distinct values of the semantic attribute, or only a low number of those values is used at the same time.

In the example below (Figure 40), $mRF = f(spA, sA(spA), nM, sA)$:

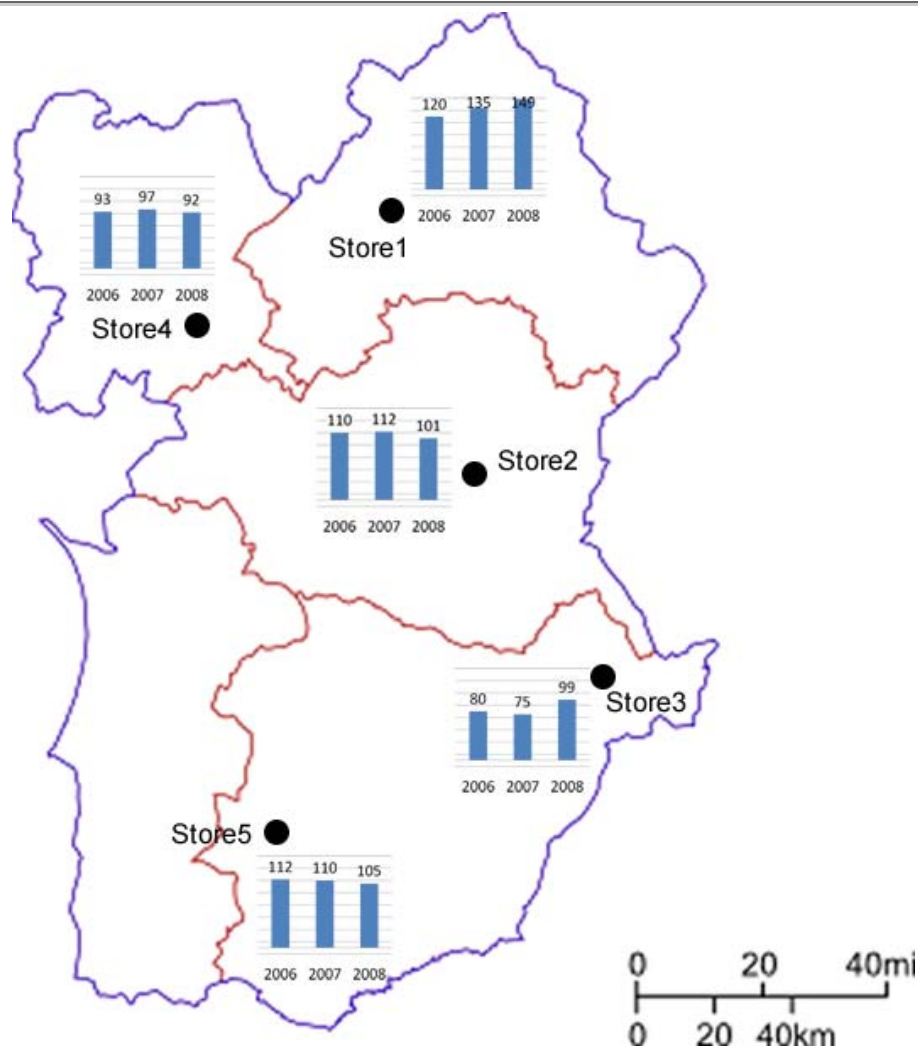


Figure 40 - An example map where a numerical measure is represented along with 3 distinct values for a semantic attribute

Multiple measures and semantic attributes can be used on this approach but, problems arise on the map, as this component becomes excessively cluttered with text and graphical data. In order to represent multiple measures/attributes on the map we would need to adapt the chart for this representation. This can be done by creating multiple groups of bars, where each one represents either a measure distributed by several attribute's values or the opposite situation.

This method ensures that a 1:1 relationship is kept between the support table and the map, where for each line in the first corresponds a spatial object in the latest.

3.7. Case 4: Semantic Attributes from a Spatial Dimension

Like in the previous interaction case (presented in section 3.6), we define how data is represented when dealing with semantic attributes (*sA*), with the difference that *sA* is from a spatial dimension this time. Note that there is already a spatial attribute (*spA*) and at least a numerical measure (*nM*)

present, as you can see by a representative vector object for this case: $vObj = \{spA, sA(spA), nM_1, \dots, nM_n, sA\}$, where sA is the new semantic attribute.

When adding a semantic attribute (sA) from a spatial dimension it is important to consider the level at which both spA and sA are (comparing levels is described in detail in section 4.3). If sA is at the same or higher level than spA , then there is only one value of sA for each spA .

On the other hand, if sA is at a lower or incomparable level than spA , then there are possibly multiple values of sA for each spA . In this case, a different approach is needed in order to keep the 1:1 relationship between the map and the support table as we'll see.

3.7.1. Semantic Attribute at the Same or Higher Level than the Spatial Attribute

3.7.1.1. Support Table

In this case, the number of lines in the support table is not altered. A new column indicating the value of sA for each line is added ($stRF = f(sA(spA), sA, nM)$)(Figure 41):

Point (sA)	Semantic1 (sA)	Measure1 (nM1)
A	Z	13
B	Z	5
C	Y	0
D	Z	15
E	X	8
F	X	7

Figure 41 - Support table after adding sA

3.7.1.2. Map

In order to represent the semantic attribute, a spatial object property has to be chosen. The spatial objects we consider in this example are points. Since we are using color to represent the measure (as proposed in the first interaction case), we could choose among size, shape, pattern or any other. In the following map (Figure 42), shape was used to represent the semantic attribute and color to represent the measure ($mRF = f(spA, sA(spA), nM, sA)$):

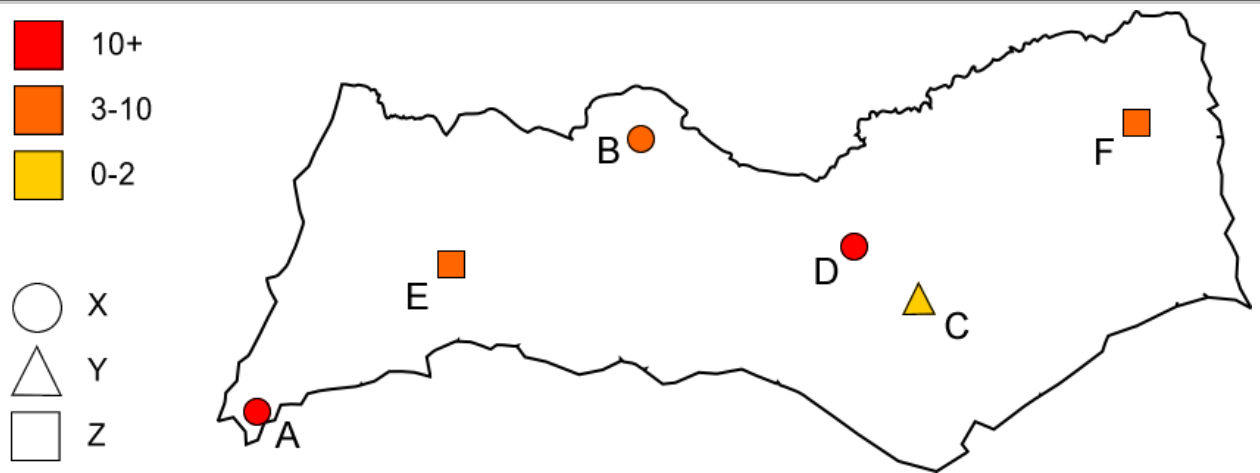


Figure 42 - Map with numerical measure and semantic attribute representation

3.7.2. Semantic Attribute at an Incomparable or Lower Level than the Spatial Attribute

When the semantic attribute sA is at a lower level than the spatial attribute spA or at an incomparable level (meaning that there is no *path* between the two levels), the support table representation function receives the same arguments, but as we can see in Figure 43, the table structural representation is different:

	<i>store_type</i>		
<i>store_area</i>	A	B	C
Area1	2	2	2
Area2	3	1	1
Area3	0	5	3
Area4	2	1	1

Figure 43 - Support table after adding *store_type* (sA)

Notice that this case is identical to the one presented in the previous section (3.6). The representation function mRF would produce a similar map, since the input support table has the same structure.

Once again this matrix structure ensures the 1:1 relationship between the lines in the support table and the spatial objects represented in the map.

3.8. Case 5: Spatial Attributes from the Same Dimension

The base scenario in this case contains one or more numerical measures (nM_i) in addition to a spatial attribute (spA) and respective associated attribute $sA(spA)$.

When the user adds a new spatial attribute from the same dimension as the one already present, two cases are considered: Both attributes belong to the same hierarchy or they are from different hierarchies. A vector object in this case has the following structure: $vObj = \{spA_1, sA(spA)_1, \dots, spA_i, sA(spA)_i, nM_1, \dots, nM_n\}$.

3.8.1. Attributes from Same Hierarchy

If the spatial attribute we are adding is in the same hierarchy tree as the one already present in the analysis, this is either a drill-down or roll-up operation, depending on the hierarchy level of both attributes (Figure 44):

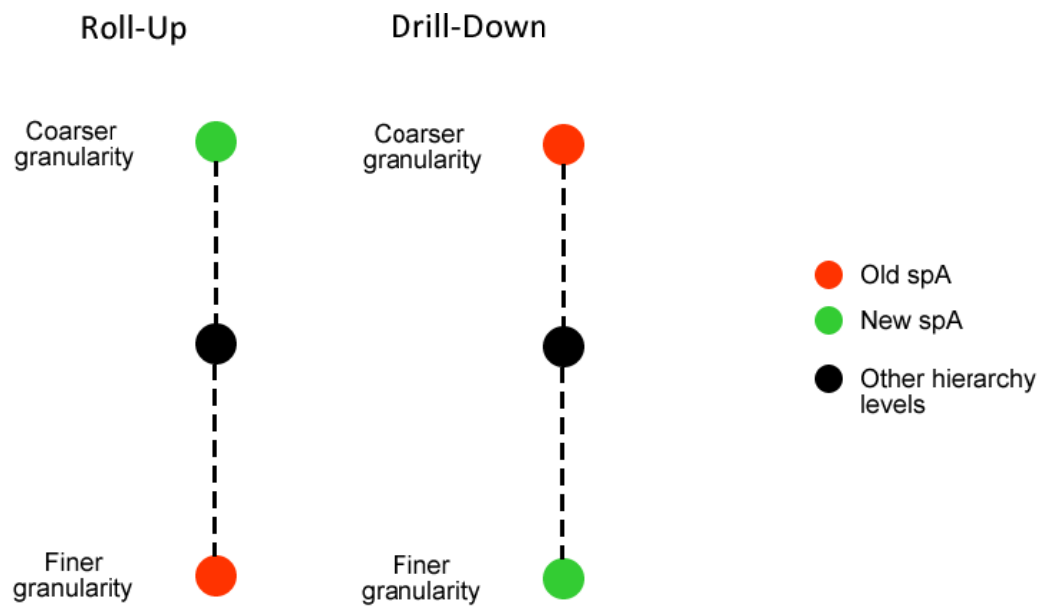


Figure 44 - Hierarchy diagrams

Old <i>spA</i>	New <i>spA</i>	Operation	Example
Higher hierarchy level (Coarser granularity)	Lower hierarchy level (Finer granularity)	Drill-down	Old <i>spA</i> represents a country. New <i>spA</i> represents a region. When we add the new <i>spA</i> , the analysis required is at the region level, thus finer granularity.
Lower hierarchy level (Finer granularity)	Higher hierarchy level (Coarser granularity)	Roll-up	Old <i>spA</i> represents a city. New <i>spA</i> represents a region. When we add the new <i>spA</i> , the analysis required is at the region level, a generalization, thus coarser granularity.

3.8.1.1. Support Table

The support table's old *spA* is replaced with the new *spA* and data is presented at the respective level.

3.8.1.2. Map

The map representation function mRF is used in the same way as the previous interaction cases, since we have only one spatial attribute and one or more numerical measures. Figure 45 shows examples of roll-up and drill-down operations.

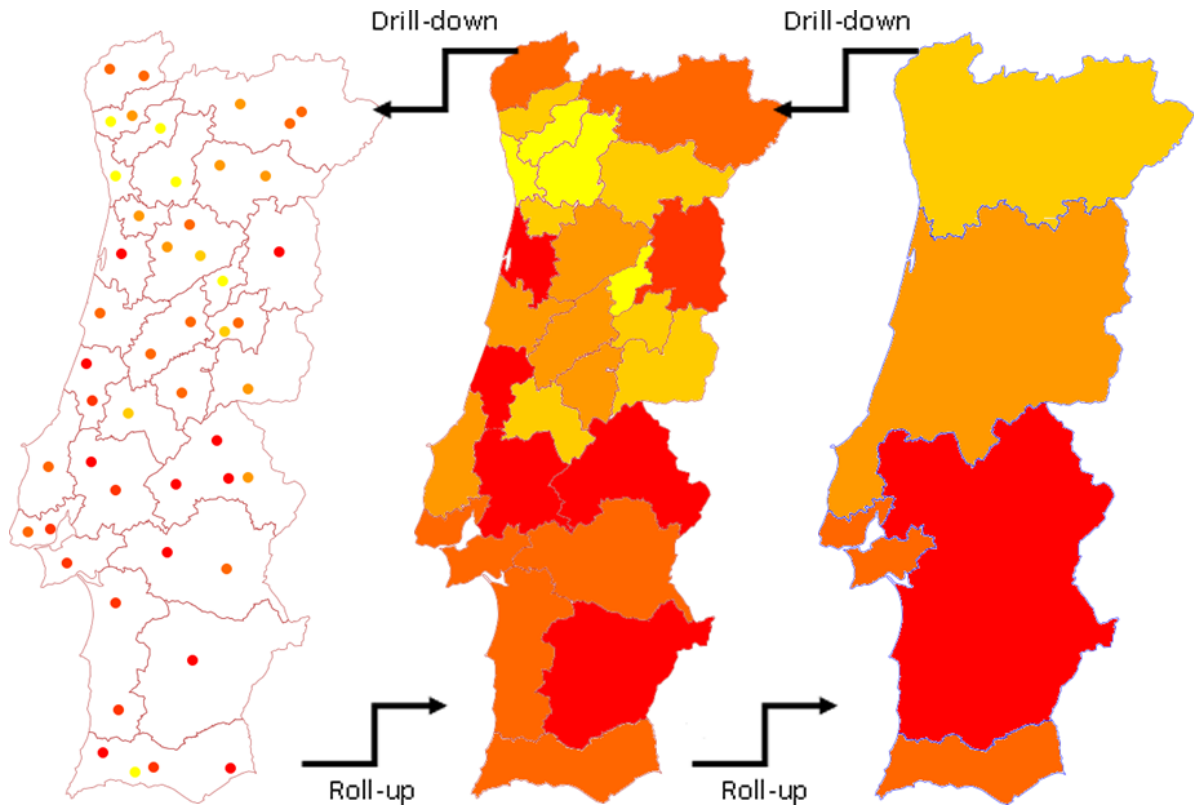


Figure 45 - Roll-up and drill-down operations (Average used as aggregation function)

3.8.2. Attributes from Different Hierarchies

If the spatial attribute we are adding is not in the same hierarchy tree as the one already present in the analysis, there are two possible approaches, depending on whether the inclusion principle is verified or not in the situation.

Let spA_1 be the spatial attribute already present in the visualization and spA_2 the spatial attribute we are adding. spA_{base} is the spatial attribute at the lowest level of granularity on the same spatial dimension.

Definition 6: The **inclusion principle** between spA_{base} , spA_1 and spA_2 is verified iff $spA_{base} \subset spA_1$ and $spA_{base} \subset spA_2$.

If the inclusion principle among the three relevant attributes is verified, both the support table and the map will generate and represent intersections between spA_1 and spA_2 , as depicted in the following example:

Consider the following hierarchies in the Portuguese administrative subdivisions (Figure 46).

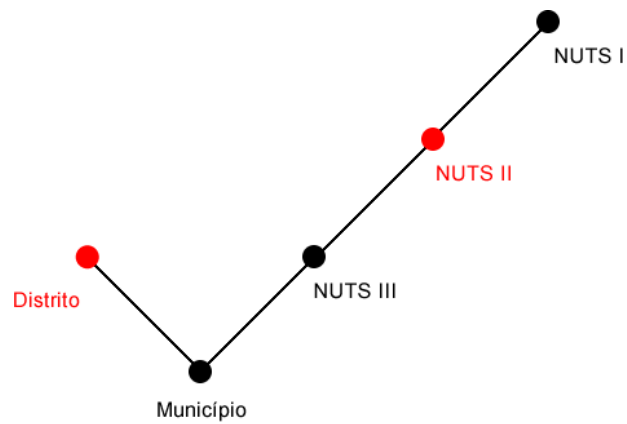


Figure 46 - Portuguese administrative subdivisions

Consider data (store sales) at the “Município” level (spA_{base}). spA_1 will be “Distrito” and spA_2 will be “NUTS II” (see Figure 47, Figure 48 and Figure 49).



Figure 47 - “Município” subdivision - lowest data granularity



Figure 48 - "Distrito" subdivision

The three regions marked in red (Figure 48) are the ones we will consider in our example.



Figure 49 - "NUTS II" subdivision

Figure 50 shows that the inclusion principle is verified among the three spatial attributes considered:

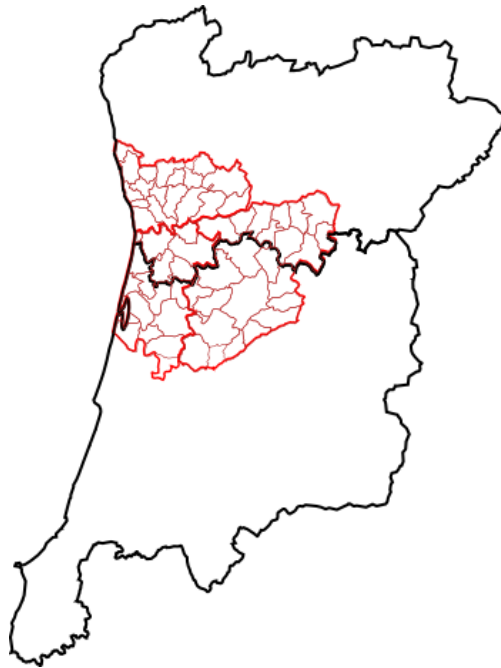


Figure 50 - Verifying the inclusion principle

Since the inclusion principle is verified, we will have five elements to represent, which are the result of the intersection between spA_1 and spA_2 , shown in Figure 51:

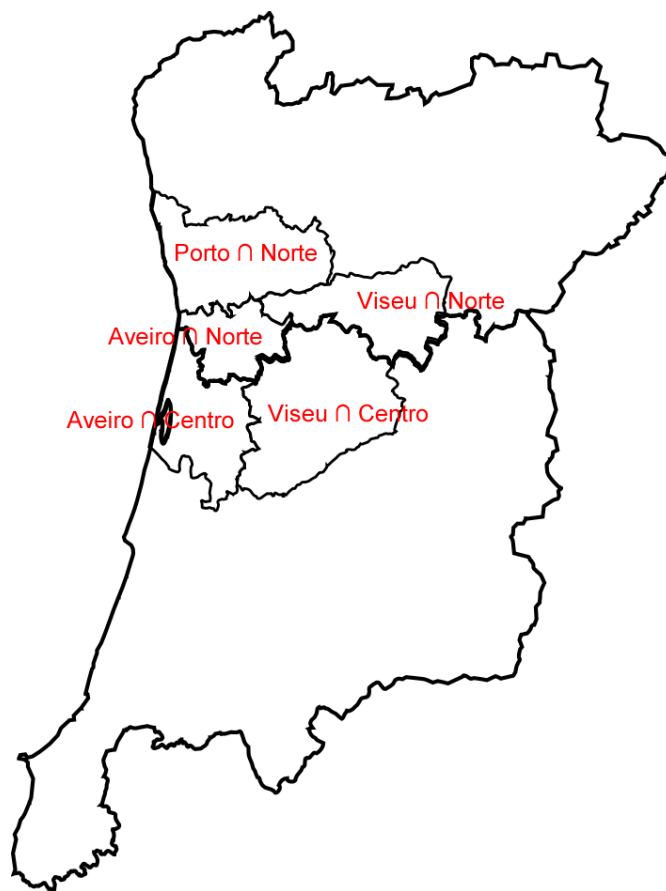


Figure 51 - Intersections between "Distrito" and "NUTS II"

The support table will have five lines (Figure 52), one for each intersection. The map maintains the 1:1 relationship, having five corresponding spatial objects (Figure 53) (Notice that the Geometric Grouping Function described above is used in these cases, since we will have two spatial attributes as input for one spatial object):

Distrito	NUTS II	Sales (nM)
Aveiro	Centro	120
Aveiro	Norte	112
Porto	Norte	198
Viseu	Centro	93
Viseu	Norte	101

Figure 52 - Support table

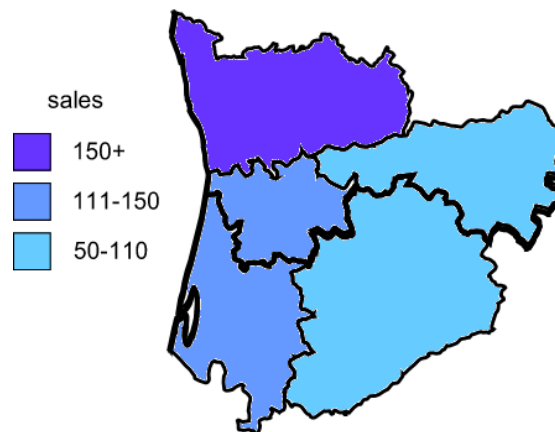


Figure 53 - Map (with legend)

If the inclusion principle between the three parts is not verified, a different approach is required, presented in the next section.

3.9. Case 6: Spatial Attributes from Two Dimensions (or with no inclusion)

In this case, we will often refer to spA_1 and spA_2 . The first one is the spatial attribute already present in the analysis, while spA_2 is the spatial attribute being added.

In the previously defined interaction model [6], visualizing spatial attributes from different dimensions would only be possible by fixing a value of spA_1 . That way, multiple maps and support tables would be created, one for each distinct value of spA_1 .

Even though this approach allows the user to analyze data for each value of spA_1 , it doesn't offer a comprehensive view on the relations between the two spatial attributes, making it extremely hard to make comparative analysis among them.

To address this issue, we propose a new approach to these cases, specific to two spatial attributes, where only one map and the respective support table is used. More than two spatial attributes would require further study, we are confident however that the great majority of the necessary analysis for most areas could be done by using up to two spatial attributes. Each line in the support table establishes relations between spA_1 and spA_2 . Those relations are represented in the map by mRF as lines that connect the determined locations of the respective spatial attributes.

A representative vector object is $vObj = \{spA_1, sA(spA)_1, \dots, spA_i, sA(spA)_i, nM_1, \dots, nM_n\}$.

Consider in the following examples that sA_1 is the semantic attribute related to spA_1 , having {A, B, ..., E} as values and sA_2 is the semantic attribute related to spA_2 , having {1, 2, ..., 5} as values.

The map representation function is defined as $mRF = f(spA_1, sA(spA)_1, spA_2, sA(spA)_2)$ as you can see in Figure 54:

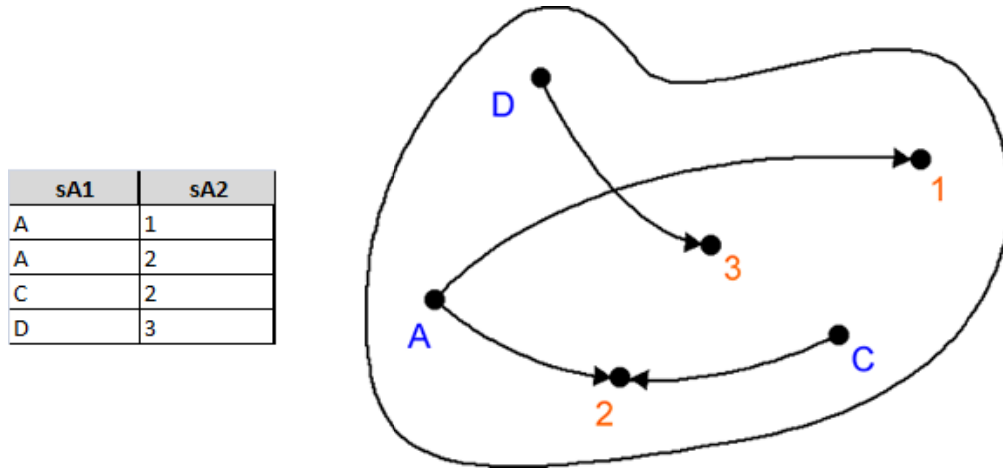


Figure 54 - Example of support table and respective map when representing two spatial attributes from different dimensions

Notice that the 1:1 relation between the support table and the map is maintained: To each line in the table corresponds a relationship-line in the map. In this simple example there are no measures represented. In a more realistic scenario, we will have one or more numerical measures associated with each relation. In those cases, measures will be represented on the relationship-lines itself or in an associated chart. The example below (below) shows four relationship-lines and two numerical measures. nM_1 is represented by line thickness and nM_2 is represented by color ($mRF = f(spA_1, sA(spA)_1, spA_2, sA(spA)_2, nM_1, nM_2)$):

sA1	sA2	nM1	nM2
A	1	8	2
A	2	5	9
C	2	3	1
D	3	2	7

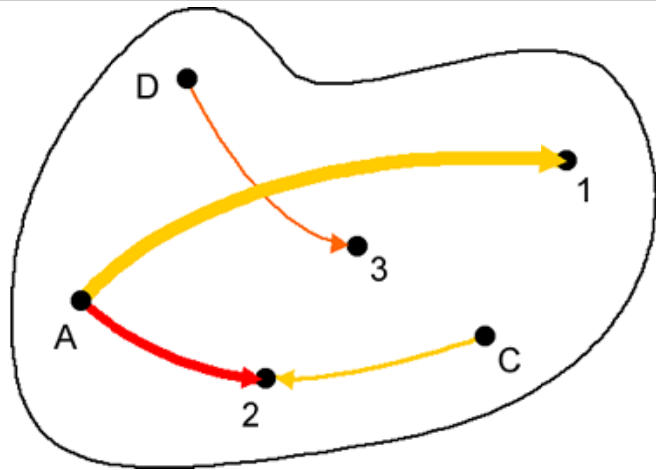


Figure 55 - Example of support table and respective map when representing two spatial attributes from different dimensions and two numerical measures

We propose yet another map representation for these cases, based on charts. In this representation, to each spatial object is assigned a color. Then, to each line in the support table will correspond a bar or slice (depending on the chart type used) placed on the destination (second) spatial object, whose color corresponds to the initial (first) spatial object. As an example, consider the same table as in Figure 54 but with only one measure (nM_1). It would be represented by *mRF* as seen in Figure 56:

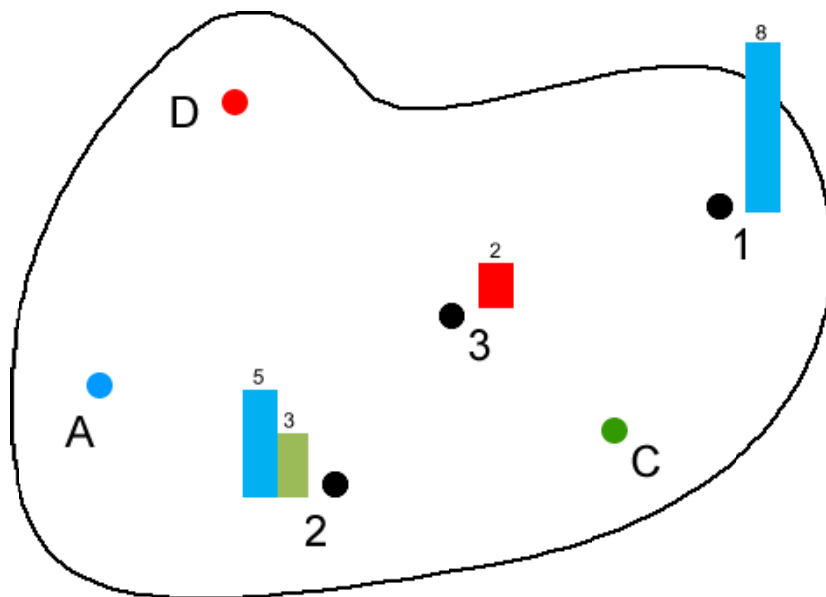


Figure 56 - Alternative representation

These representation methods could be extremely useful and intuitive for many areas, namely transportation or tourism. If we have the *departures* and *arrivals* spatial dimensions, such map visualizations would give a very useful overview of passenger movements.

3.9.1. Roll-Up and Drill-Down Operations

When representing spatial attributes from different dimensions, roll-up and drill-down operations can still be used by following each attribute's hierarchies. The previous example considered both spatial attributes at a level where their respective spatial object is a point, however, when a roll-up operation is performed on one or both attributes, their representation may be an area (polygon).

Consider the same support table and representation using relationship-lines in Figure 57 (the above hierarchy level to which the points will roll-up is marked by the Greek letters α , β and γ):

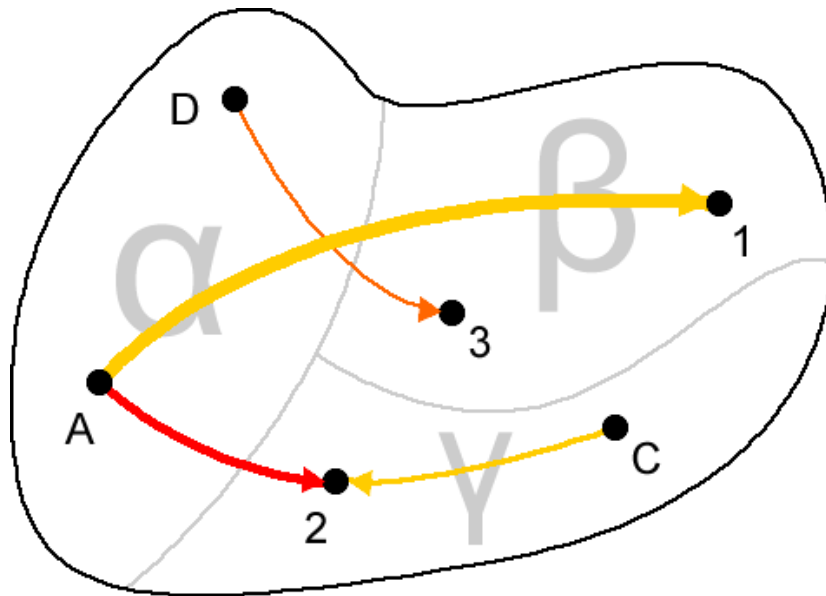


Figure 57 - Example using relationship-lines

If the user applies a roll-up operation on spA_1 (to which $sA(spA)_1$ is the related semantic attribute) an aggregation will take place and the respective spatial objects will be polygons (α , β and γ). In this example, the aggregation functions applied were sum to nM_1 and average to nM_2 , producing the new support table in Figure 58:

new sA1	new sA2	nM1	nM2
α	β	10	4.5
α	γ	5	9
γ	γ	3	1

Figure 58 - New support table with aggregated data

Based on the new granularity data, mRF produces the following representation (see Figure 59) (A ghost point (definition 7, see below) is used for each area):

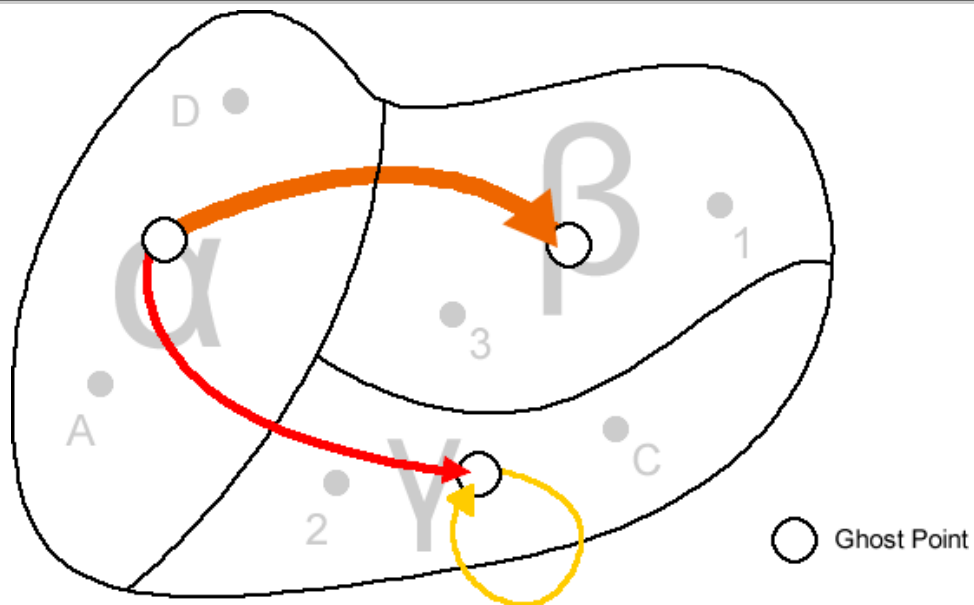


Figure 59 - New map representation

The same principle applies when using the chart-based representation (only nM_1 is represented)(Figure 60):

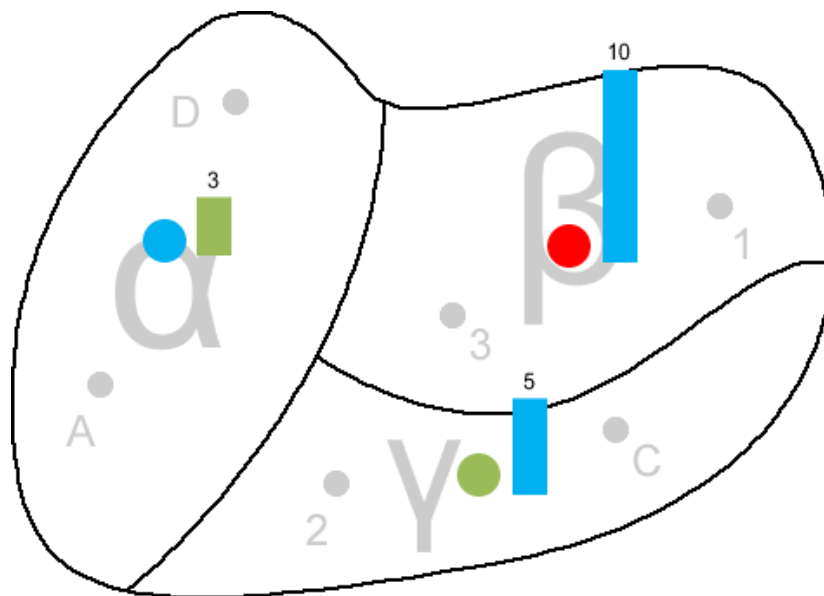


Figure 60 - New map representation using charts

Definition 7: Ghost point is an auxiliary GIS object (point) that is used to represent an area. Several approaches can be used when determining an area's (polygon) ghost point such as: User-defined, horizontal/vertical centered relative to the polygon, computed based on lower-level points, etc.

3.9.2. Visualization Clustering

As it was mentioned in section 3.2 (Data Representation), clustering may be very important to make the analysis process easier for the user. As an example of automatic group creation consider the following support table and direct map representation by *mRF* (without visualization clustering)(Figure 61):

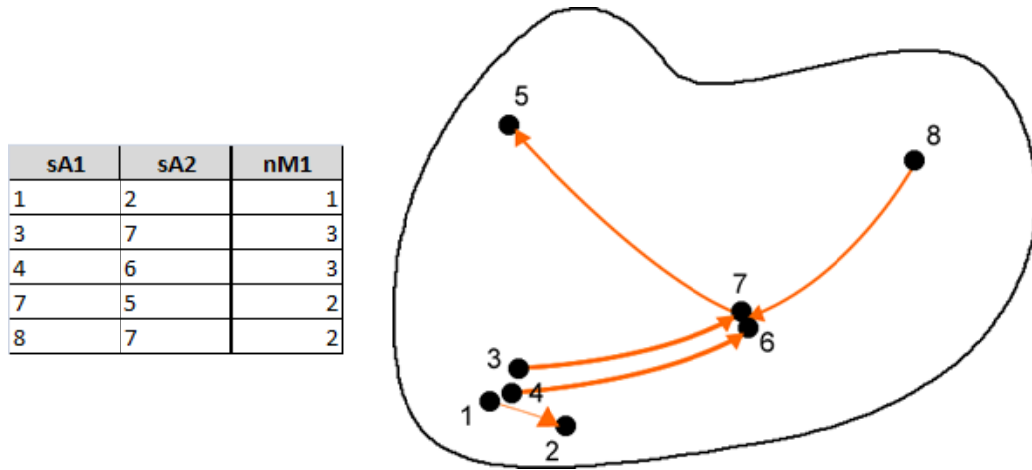


Figure 61 - Support table and map representation without visualization clustering

Using visualization clustering, a new aggregated support table (used sum as aggregation function in this example) would be generated and therefore a new representation by *mRF* (A and B are *ad hoc* groups/clusters with no semantic meaning)(Figure 62):

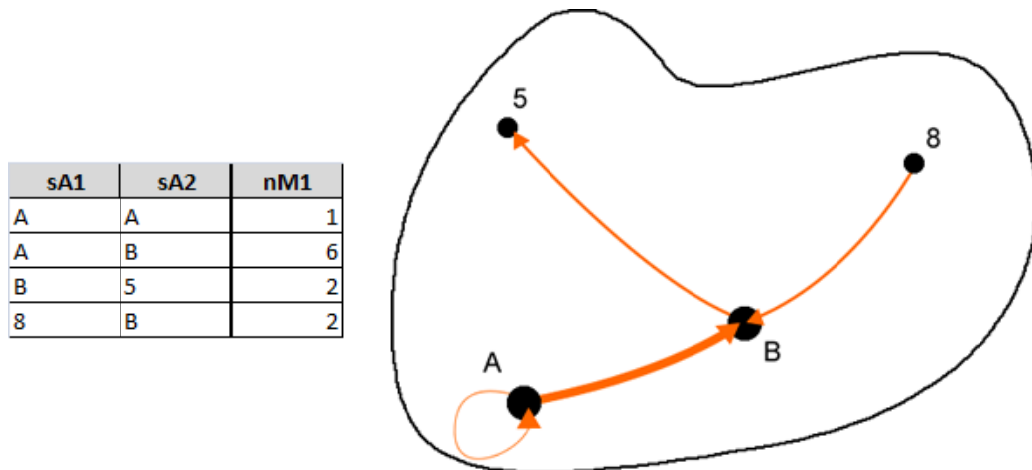


Figure 62 - Support table and map representation with visualization clustering

3.10. Spatial Measures

Spatial measures were not considered in the previous interaction model even though there are several works in which they are proposed using different approaches (see Chapter 2).

As it was mentioned, none of the studied works covers spatial measures that are arbitrary geometric shapes directly and solely related to a fact/event. These arbitrary spatial measures allow a much higher flexibility - they are not related to already existent spatial attributes and therefore have no restrictions on their shape and size. Using arbitrary spatial measures we can represent the area of a fire, the radioactive cloud of a nuclear incident or any other spatial value that is not related to pre-defined spatial hierarchies in a multidimensional model.

We have to take into consideration that since the spatial measures are stored in the fact table, the number of spatial elements to store in the database and process for analysis is much higher than when we're dealing with spatial attributes from dimensions. Being measures and not attributes from dimensions also influences greatly on both map representation and aggregation of spatial data.

Representing and aggregating spatial measures are the two main issues that require further research in order to reach a compliant and useful interaction model supporting them. We would suggest starting with spatial measures represented by points for two reasons: 1) it is easier to acquire real world data where events are characterized by simple coordinates (points) than areas (polygons) and 2) visualization, clustering and aggregation is more intuitive using points. This subject is not studied further on this thesis.

3.1.1. Visualization Complexity

A table's complexity is associated with the number of rows, columns and cells it contains. The number of cells can be calculated simply by multiplying the number of rows by the number of columns in the table.

In our interaction model it is assumed that the support table has a one to one relationship with the map at all times. This means that for each row in the support table there is a spatial object in the map. Adding a new spatial object to the analysis means that a new row will be added to the table.

The number of columns is more complex - even though it is related to the number of measures and attributes, it is not a linear relationship as it would be if we were using a regular table. In order to maintain the "one row, one spatial object" property, our table can have nested headers. This happens when the added attribute has either a lower level, is from a different dimension or from a different hierarchy than the current spatial attribute. For the remaining of this section, whenever we mention "header attribute", we're referring only to these. You can think of such a table as a tree where nodes contain header attribute values and leaves (cells) contain the measure values associated with a specific combination of header attributes values. For example, the following two representations of the same table are equivalent (both diagrams represent a table with two measures (M1 and M2) and two header attributes, each of which has two distinct values ('X', 'Y' and 'A', 'B')):

X				Y			
A		B		A		B	
M1	M2	M1	M2	M1	M2	M1	M2
...							

Figure 63 - Table representation

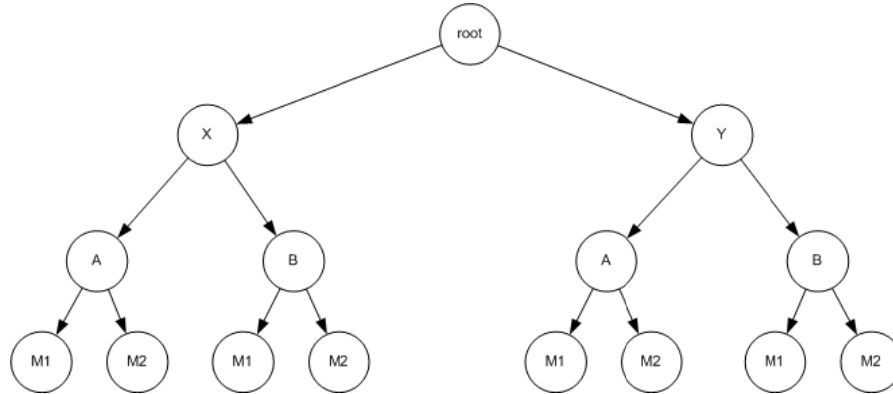


Figure 64 - Tree representation

Whenever a measure or attribute is added to the analysis, it will possibly add multiple columns to the table. The total number of columns in the support table can be calculated using the following formula:

$$NumColumns = NumInLineAttribs + NumMeasures * \prod_{i=1}^{NHeaderAttribs} NDistincts(i)$$

As we can see from the formula, while adding In-Line attributes (those that are at the same or higher level than the considered spatial attribute) leads to a linear growth in the number of columns, adding a header attribute leads to a much higher growth that depends on the number of distinct values.

In order to maintain a one to one relationship between the rows in the support table and the spatial objects in the map, we have to be able to represent all the values associated with each row in the same spatial object. This means that we need a visual property of the spatial object to be controlled for each column in our support table.

When we have only one measure and no header attributes, our support table has one numerical column, containing the measure's value. This can be easily represented by mapping the respective value to the size of the spatial object (if it is a point) or its color (in case it's a polygon). When we have more than one measure and/or header attributes, our support table will have more than one column - each object has multiple values associated with it. In this case we have to control multiple visual properties of the spatial objects, one for each column. Instead of associating each value with

properties such as pattern or dash, which would be extremely hard to analyze, our approach involves charts - one chart for each object, where each bar maps a column/value.

The cognitive process of a user is then conditioned not only by the table's complexity but also by the complexity of the spatial objects that derive from the first. Keeping the number of columns in the table to a small number (and therefore the number of bars in a chart) is essential. As it was explained earlier, this can be done by limiting the number of in-line attributes and measures, but more effectively by limiting the number of header attributes and their respective distinct values.

3.12. Styles and Legend

While spatial objects indicate *what* to display on a map, a style is a description of *how* to display it. Using different styles we can control whether we want to display a variable size point marker, areas with a color gradient, a chart, etc.

The possible/recommended representations (and therefore the applied style) depend on two main factors: The geometry of the spatial objects and the number of values to represent on each of those objects.

The following figure shows some of the possible styles and the recommended style for each relevant combination:

	Geometry type								
	Point			Polygon (Area)			Line		
	Number of numerical columns			Number of numerical columns			Number of numerical columns		
	1			> 1			1		
Possible styles	Variable size marker	Bar chart		Color gradient	Bar chart		Color gradient	Bar chart	
	Variable color marker	Pie chart		Bar chart	Pie chart		Line width	Pie chart	
	Bar chart						Bar chart		
Recommended style	Variable size marker	1 Meas.	> 1 Meas.	Color gradient	1 Measure	> 1 Measure	Color gradient	1 Meas.	> 1 Meas.
		Pie chart	Bar chart		Pie chart	Bar chart		Pie chart	Bar chart
		Bar chart			Bar chart			Bar chart	

Figure 65 - Possible and recommended visualization styles

The cases where there is more than one column but only one measure happen when the user adds a header attribute, which adds a new GROUP BY element and causes the measure value to split into multiple. In these cases the recommended style would depend on the type of analysis the user is looking for: If the user wants to compare the values for each spatial object between themselves, the

pie chart is the optimal choice. On the other hand, if the user wants to compare values among different spatial objects, the bar chart is more clear.

In certain cases it is important to know if our numerical values are *components* of a measure: Consider $v_1, \dots, v_i$ as being the numerical columns we want to represent. $v_1, \dots, v_i$ are the **components** of a measure X if $X = v_1 + \dots + v_i$. Example: “Public expense” is a component of “Expense” because “Expense” = “Public expense” + “Private expense”.

A pie chart is usually used to represent a whole divided into its parts (the components of a measure). When we have only one measure split into multiple values due to a header attribute, it is exactly the case where a pie chart is a good representation. The pie chart style is not recommended in the other cases because if we have more than one measure, we are representing values related to different concepts. Two different measures are not part of a whole, so it would be illogical to use a pie chart to represent them - in these cases, measures can be represented by a simple bar or line chart, where to every measure corresponds a bar/line. The aim of the analysis in these cases is usually the absolute values of each measure, and not the proportion among them.

There are many subjects related to styles and legend that are open for discussion - the usage of local and global scale for different charts, adapting the bar's scale according to the measure it represents, using both constant and variable value groups in a legend, style/legend editor, etc. These subjects are out of the scope of this work but research on these topics would probably lead to the implementation of many new analysis features related to styles and legends.

Chapter 4

Architecture

This chapter presents the SOLAP+ general architecture, its components, models and communication protocol

4.1.	Architecture Overview.....	66
4.2.	SOLAP+ Server Architecture	67
4.3.	SOLAP+ Client Architecture.....	72
4.4.	Communication Protocol	75
4.5.	Meta Model	80

This chapter presents the proposed architectural design for the prototype to be developed. It starts by an overview of the entire system, its components and their communication, followed by a more detailed insight into the SOLAP+ Server and Client. It is then presented the Communication Protocol between the SOLAP+ Server and Client and finally the Meta Model definition.

4.1. Architecture Overview

There are five main components in our proposed logical architecture: SOLAP+ Client, SOLAP+ Server, Spatially Compliant Data Server, Map Server and Metadata Repository.

The SOLAP+ Client handles all user interaction, data presentation and request generation. It communicates with the SOLAP+ Server to send data requests and receive responses. It also communicates with the Map Server to request maps and present them to the user. The SOLAP+ Server is responsible for listening to client requests, processing them using the required metadata information from the Metadata Repository and retrieving the appropriate data stored in the Spatially Compliant Data Server. It can also communicate with the Map Server to retrieve map data. The Metadata Repository is where all the required metadata for the system is stored. It is accessed only by the SOLAP+ Server in several situations described in this chapter.

Map Server and Spatially Compliant Data Server are external components that permit generation of dynamic maps and retrieval of data respectively, both based on request/response approaches.

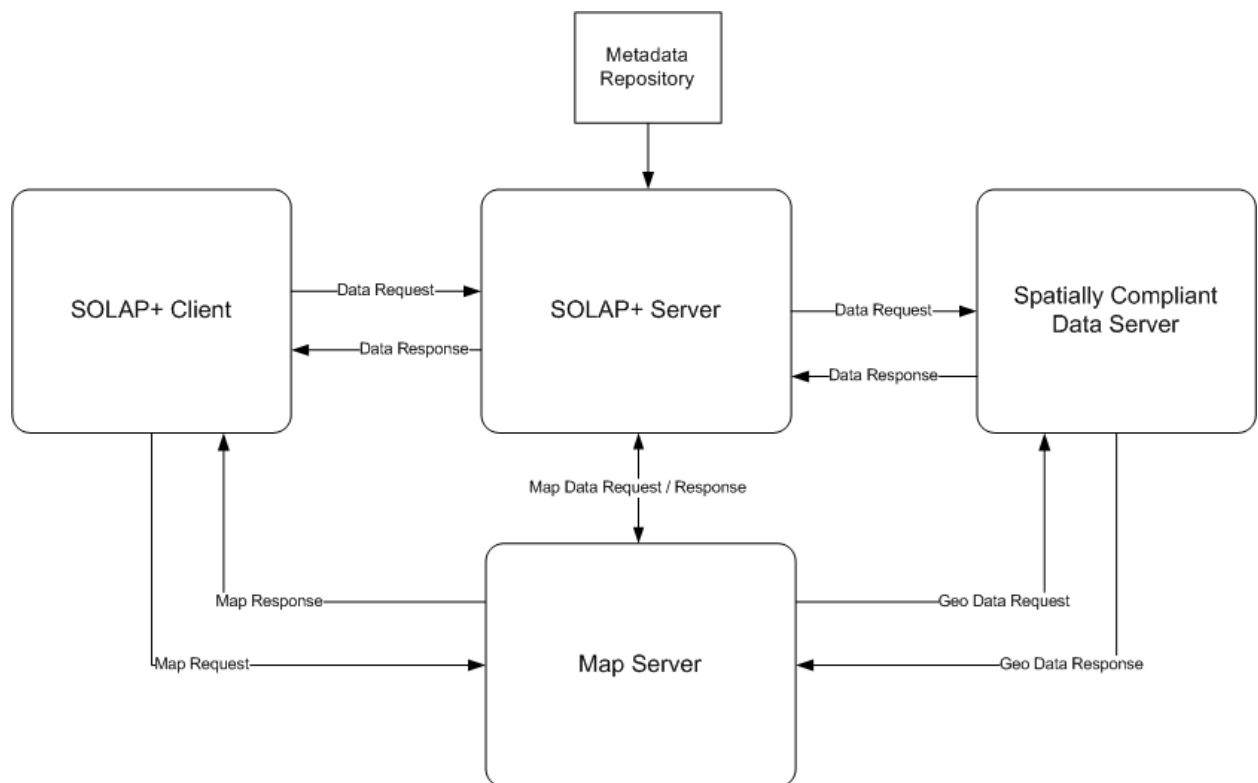


Figure 66 - SOLAP+ Logical Architecture

4.2. SOLAP+ Server Architecture

A SOLAP+ request is handled by the server as follows: First the request goes through a parser, where it is validated. The parameters are then extracted and sent to a processor. Requests that do not require information from the database (such as List Cubes) are processed using the request parameters and metadata only, producing response parameters that are sent back. Other kinds of requests (such as Get Data) require access to the database. In these cases, the processor uses the request parameters, auxiliary metadata and database-retrieved information to generate the appropriate response parameters. Finally, the actual XML response is generated based on the received response parameters and sent back to the client.

In this process we can identify four main components inside the SOLAP+ Server: Communication Handler, Metadata Request Processor, Data Request Processor and SQL Query Processor:

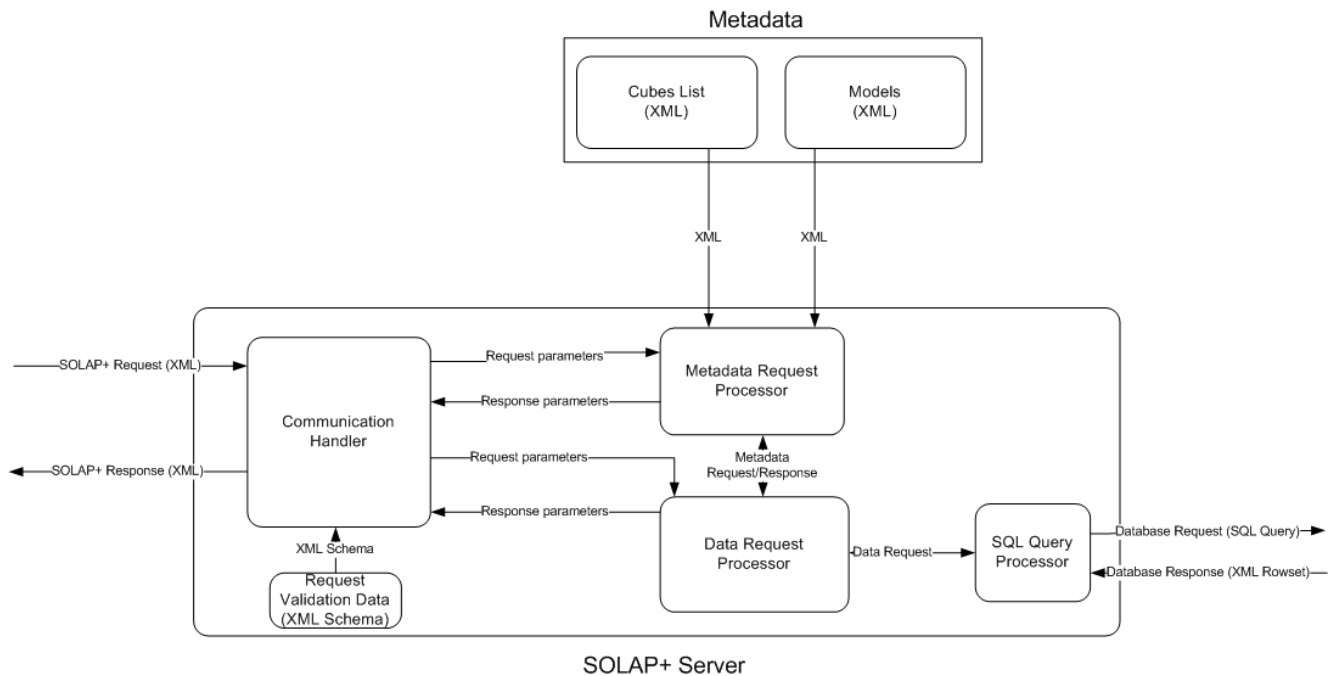


Figure 67 - SOLAP+ Server Architecture

Communication Handler receives the XML request from the client and validates it using an internal XML Schema. If the request is valid, it also extracts the parameters from the request and sends them to the appropriate component (Metadata Request Processor or Data Request Processor) depending on the type of request.

As mentioned above, the Metadata Request Processor component deals only with requests that do not require database access. The received request parameters along with the model information from the Metadata allows this component to produce response parameters and send them to the Response Builder without the need for any other component.

The Data Request Processor deals with requests that involve database access. It receives request parameters and model information metadata from the Metadata Request Processor, producing requests for database information. This information is received and compiled, creating response parameters and sent to the Communication Handler. This component is fairly complex and will be detailed further on.

SQL Query Processor acts like a database connector - it receives queries from other components and executes them on the DBMS. It is also responsible for retrieving the results and sending them back to the requesting component.

Finally, the Communication Handler component is also responsible for compiling all the response parameters received and generates a SOLAP+ XML response with the required information. This XML response is then sent back to the client.

As mentioned before, the Data Request Processor (Figure 68) handles requests that require database access as well as metadata and request parameters. Once the request parameters are received, the first action taken is the choice of an aggregate by the Aggregate Navigator (detailed ahead). Once an aggregate is chosen, this information along with the request parameters are sent to the SQL Generator which in turn calls one or more auxiliary components (Table SQL Generator, Spatial SQL Generator, Distincts SQL Generator or any other SQL module) depending on the request type. When the response parameters are ready, they are sent back to the Communication Handler component.

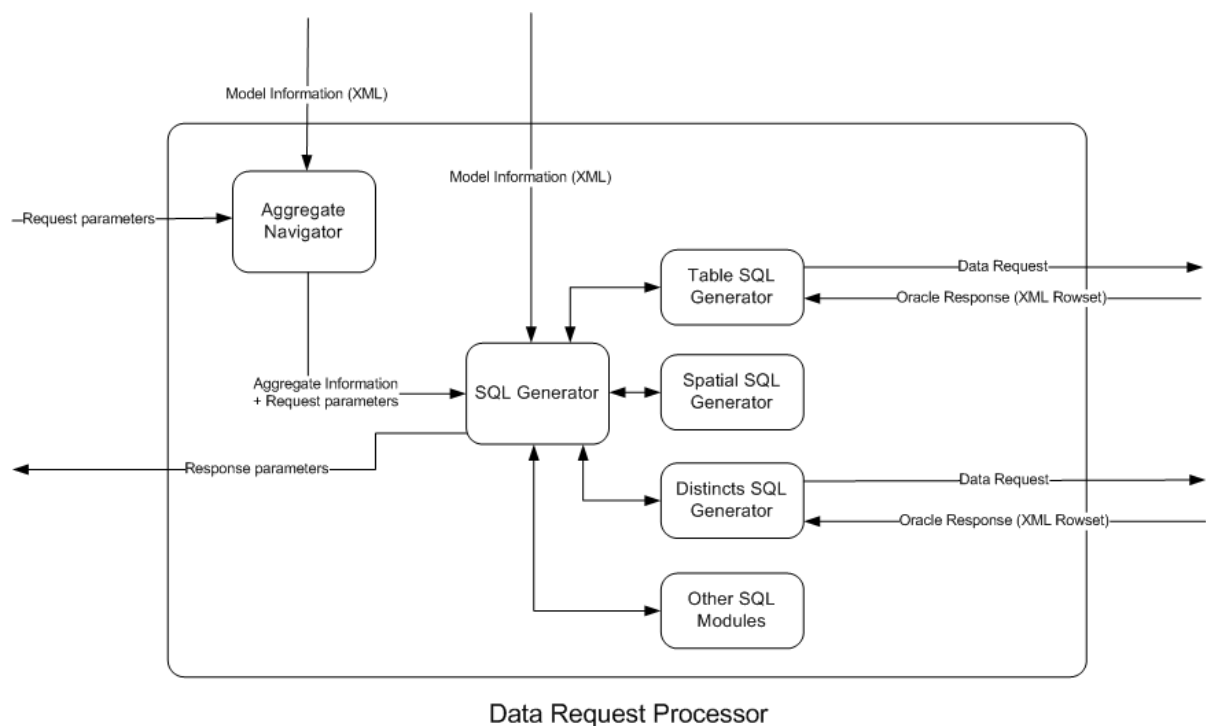


Figure 68 - Server's Data Request Processor Architecture

SOLAP+ XML requests are not aggregate-aware. This means that a request sent to the server always refers the original multidimensional model's entities such as fact tables and dimensions. It is the Aggregate Navigator's responsibility to select the best possible aggregate to use for each request/situation based on the request parameters.

Receiving requests that always refer the original multidimensional model takes away the complexity of choosing an aggregate from the analyst/client. As explained before when presenting aggregates, the possible ones are selected based on the referenced levels in the request parameters/query. If there is an aggregate that can be used, it will be selected and all further processing will use the aggregate's tables instead of the original ones; otherwise, the original fact table and dimensions will be used. In case there is more than one possible aggregate for a certain query, an heuristic will have to be used to select one of them. Comparing aggregate's performance and selecting the best aggregate from within a list is, however, out of the scope of this work.

After the aggregate is selected, all the required information is sent to the SQL Generator (request parameters and aggregate information). This component determines, based on the request parameters, which generator(s) are to be used: If the request is *get_distincts*, only the Distincts SQL Generator is used. If the request is *get_data*, the Table SQL Generator is called, along with the Spatial SQL Generator in case the *spatial* flag is set to *true*; that is, if the client requires a spatial SQL query to be generated. Other SQL generator modules can be implemented.

Both the Distincts SQL Generator and Table SQL Generator components generate SQL queries based on the request parameters and send them to the SQL Query Processor to be executed on the server. They then receive a data rowset that is sent back and used to build a response.

The Spatial SQL Generator works in a different way: While the other two components execute their generated SQL queries and receive rowsets as results, the Spatial SQL Query's objective is only to produce an SQL query. This query is attached to the response that goes back to the client and it should then be forwarded to the Map Server to generate a map that represents the same information as the associated rowset produced by the Table SQL Generator.

4.3. Aggregates

There are two main issues regarding aggregates that need to be addressed: Which aggregate tables to build and which of the existent aggregates to use in each of the user's interactions/queries. In our work, we're assuming that the first issue is already solved, i.e., we already have a defined set of aggregates to use (aggregates can be described using our metamodel proposal, as seen in section 4.6.3). We now have to define techniques to select the best aggregate for each situation from our initial set. This is done in two steps: 1) Select the possible aggregates and 2) Select the best aggregate from among the possible.

Next in this section we present a representation and definition of concepts used further on, after which we address the two steps mentioned.

4.3.1. Dimensions, Levels, Hierarchies and Aggregates Representation

For representation purposes, we consider a dimension as an oriented graph where each level is a node and arcs represent hierarchic relationships between two levels. The root node is always the lowest level of a dimension (finer granularity). Following the arcs starting from the root node we can determine one or more *hierarchies* (see Figure 69).

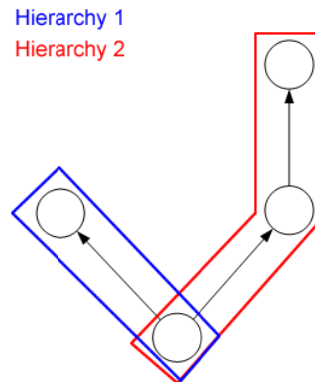


Figure 69 - Levels organized in hierarchies

Consider α and β as levels in a dimension: β is *higher* ($>$) than α if a graph path exists from α to β . In this case, α is *lower* ($<$) than β . This graph relationship means that the lower level (α) has finer granularity and the higher level (β) has coarser granularity (see Figure 70).

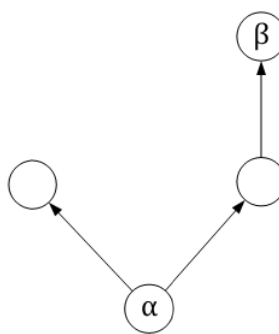
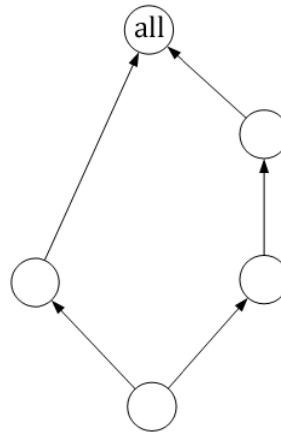


Figure 70 - Relationships between two levels

A special level named *all* can exist in any dimension. It is the highest level for every hierarchy. Using the example from the previous figure but now representing the *all* level:

Figure 71 - The *all* level

α and β are *incomparable* if there is no graph path that contains both levels. This means that α and β belong to different hierarchies and a granularity relationship between them cannot be established (Figure 72).

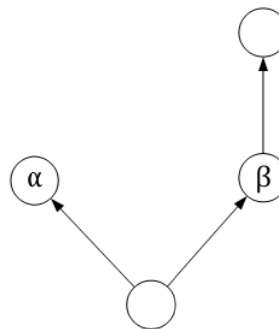


Figure 72 - Two incomparable levels

An aggregate is characterized by a level or set of levels for each dimension it indexes. Those levels define the granularity at which that aggregate is and will be used to determine if that aggregate is suitable to answer a query or not.

$$Ag = (\{Dim_1Lvl_1, \dots, Dim_1Lvl_n\}, \dots, \{Dim_kLvl_1, \dots, Dim_kLvl_i\})$$

Even though it is possible (and maybe relevant in some cases) to create aggregates with integrated slice and/or filter expressions, those will not be considered in our work. However, slice/filter operations can obviously be executed over the aggregated data.

Aggregates can also be compared, establishing a hierarchy among them. An aggregate Ag_1 is *higher or equal* ($\geq$) than an aggregate Ag_2 iff all levels from Ag_1 are higher or equal to the respective levels from Ag_2 :

$$Ag_1 \geq Ag_2 \leftrightarrow \forall Ag_{1i_{Dim_iLvl_j}} \in Ag_1: Ag_{1i_{Dim_iLvl_j}} \geq Ag_{2i_{Dim_iLvl_j}}$$

After parsing a query and extracting the levels it references for each dimension, we can build a representation of it that follows the same structure as an aggregate:

$$Q = (\{Dim_1Lvl_1, \dots, Dim_1Lvl_n\}, \dots, \{Dim_kLvl_1, \dots, Dim_kLvl_i\})$$

4.3.2. Selecting Possible Aggregates

An aggregate Ag is suitable to answer a query Q if, for each dimension, each of the levels in Q is higher than at least one of the levels in Ag :

$$\forall Q_{level_i} \in Q, \exists Ag_{level_k}: Q_{level_i} \geq Ag_{level_k}$$

Note that when a level is not present it is considered as *all*, which is the highest in any hierarchy.

4.3.3. Selecting the Best Aggregate

Using aggregates improves a system's performance because aggregation operations up to a certain level are already made, meaning that this new table has less rows than the base fact table, and therefore further operations will involve fewer rows.

At this point we already have a set of possible aggregates that can be used to answer a query. The best aggregate to use will be the one where all levels are equal to the ones referenced by the query, named Ag_Q . If that aggregate does not exist, we should use (one of) the aggregate(s) immediately below Ag_Q in the hierarchy, as those are the closest to the best.

If more than one aggregate is selected this way, an heuristic will have to be used in order to choose one of them. Following the reasoning explained in the first paragraph, one of the possible approaches would be to select the aggregate that has the lowest number of rows in its fact table. That would result in fewer operations and therefore better performance.

4.4. SOLAP+ Client Architecture

The SOLAP+ Client is responsible for generating and sending requests to the SOLAP+ Server as well as processing the responses and displaying results. These requests are triggered by user interaction in the interface, generating a message according to the communication protocol and sent to the server.

When a reply is received, it is parsed and processed, producing a visual representation of the respective data (such as a table). If there is a spatial query included in the response, it is used to request a new map to the Map Server, which is finally displayed in the user interface.

In this process we can identify four main components that constitute our SOLAP+ Client: User Interface, Communication Handler, Data Processor and Map Control (Figure 73):

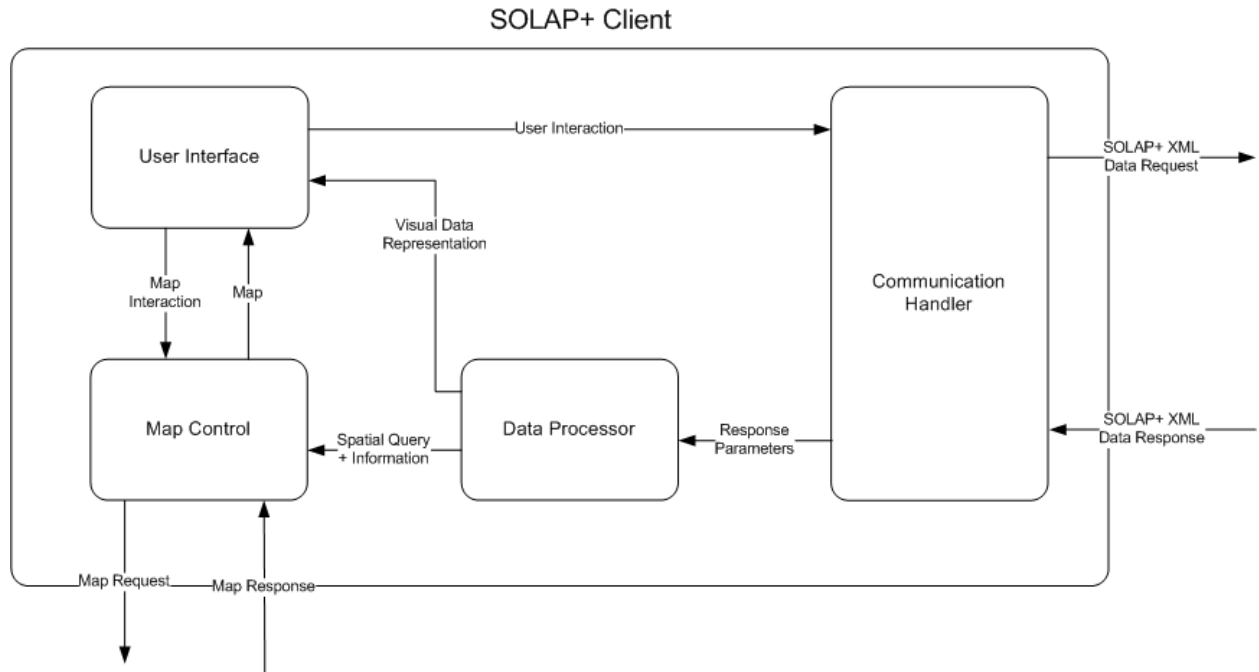


Figure 73 - SOLAP+ Client Architecture

The User Interface component is responsible for triggering events based on user interaction (such as adding attributes and measures, slices, zooming on the map, etc) as well as displaying visual information, namely the support and detail tables and the map.

These events (except the ones directed only at the map) are captured by the Communication handler which produces XML messages that comply with the specified communication protocol based on the user's actions. They are then sent to the SOLAP+ Server and processed as described in the previous section. When a response from the SOLAP+ Server is received, it is parsed, extracting its parameters and sent to the Data Processor.

The Data Processor component is where the response data is actually processed. There are different types of responses according to the type of request that was issued, but it always comes down to generating a visual representation of the received data, whether it is displaying the information from a newly loaded cube/session or rendering tables and map. The most common request is *get data*. The response to such a request includes a rowset which, along with some extra information, permits the Data Processor to generate a visual representation of it, namely the support

table and/or detail table. In case it was flagged as a spatial request, a spatial query is also included in the response. This SQL query, along with some information required for style generation is sent to the Map Control component. The Data Processor's inner architecture will be detailed further on.

Map Control is the bridge between the SOLAP+ Client and the external Map Server component. As stated before, it receives a spatial SQL query and some extra information from the Data Processor. Its main objective is to request appropriate maps to the Map Server. In order to do this, it needs to generate dynamic themes (using the spatial SQL query) and dynamic styles (using the extra information, such as number of measures, the spatial attribute's type of geometry, etc). After the themes and styles are defined, the map request is sent and a new map is received, which is then forwarded to the User Interface. The Map Control component also captures some of the interaction events from the interface, such as zooming, panning, layer displaying, etc. These events are handled by sending requests to the Map Server and receiving a new map.

The client's Data Processor (Figure 74) is basically a collection of components to produce visual representations of the received data. Based on the received response parameters, one or more of these auxiliary components are used to produce an output, which is then sent to the User Interface for presentation:

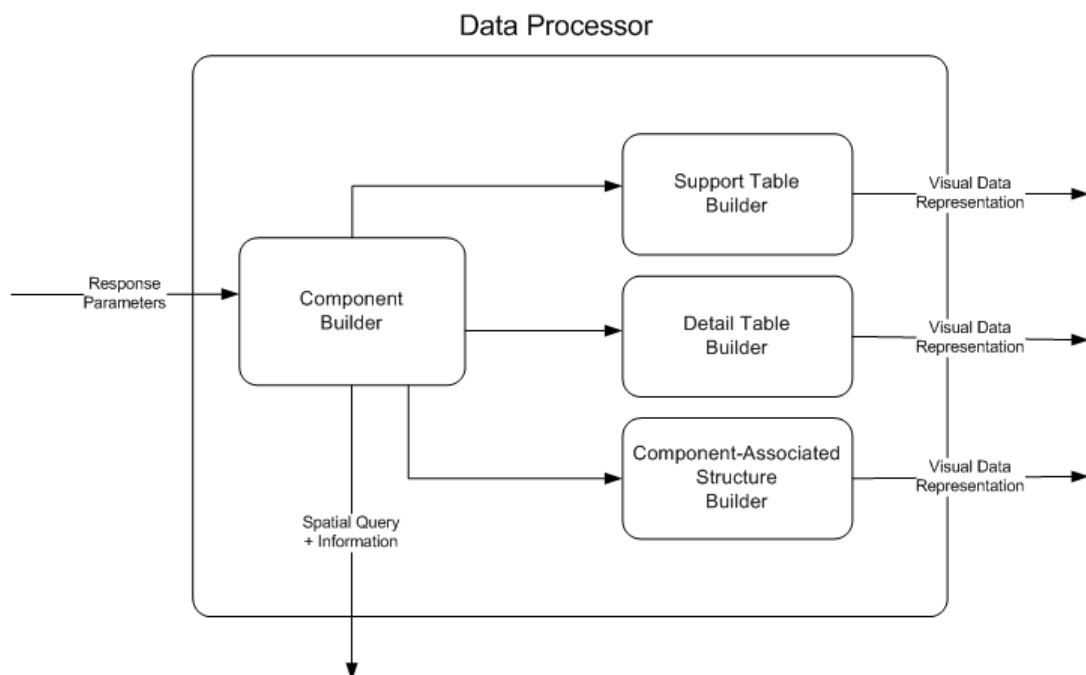


Figure 74 - Client's Data Processor Architecture

The Component Builder is responsible for forwarding the spatial-related information to the Map Control component, as well as deciding which auxiliary builder component(s) to call. In a normal *get data* request, the Support Table Builder and/or the Detail Table Builder components are used to generate visual representations of the rowset(s) received, according to the interaction model. Other

kinds of requests, such as *get distincts* or *load cube* require drop-down lists, trees or other components to be associated with data structures, such as arrays, vectors, etc. This is done by the Component-Associated Structure Builder. All of these auxiliary components produce a visual data representation, which is sent to the User Interface, associated with a certain component and presented to the user.

4.5. Communication Protocol

The communication between the client and server follows a request/response pattern. As described before, the SOLAP+ Server acts as a web service and listens for XML requests from clients. When one is received, it is validated and processed, generating a XML response which is sent back to the client.

Client requests can be split into two types: session requests and data requests. Session requests refer to loading or saving session information to a file. Data requests require that the SOLAP+ server sends information back to the client, such as a request for a new map / support table.

All SOLAP+ requests have the same XML base format:

```
<?xml version="1.0" encoding="UTF-8"?>
<solapplus>
  <request call="request_name" spatial="boolean_value">
    ...
  </request>
</solapplus>
```

Where *request_name* is the request identifier and *boolean_value* determines if a spatial query is to be produced by the server. Most of the requests require additional information which is supplied in the form of extra XML elements under the <request> element.

In the same way, all SOLAP+ responses follow the same base structure:

```
<?xml version="1.0" encoding="UTF-8"?>
<solapplus>
  ...
</solapplus>
```

The specific response data is added inside the root element.

Data requests are the most frequent and most important requests as they provide the means to perform the actual data analysis. There are four different data requests in our communication protocol: “List Cubes”, “Load Cube”, “Get Data” and “Get Distincts”. The responses to the first two are derived from the metamodels only, while the latest two require data to be retrieved from the

database. Next in this section we will present the detailed structure of each data request to the SOLAP+ server, as well as their respective responses.

4.5.1. List Cubes

List cubes is a base request aimed at getting information about the available cubes in the server. It has the following structure:

```
<?xml version="1.0" encoding="UTF-8"?>
<solapplus>
  <request call="list_cubes" />
</solapplus>
```

This request does not require any additional information apart from the request identifier, therefore it's structure is exactly the same as the base structure presented above.

The response to *list_cubes* is basically a list of the available cubes in the server, as well as some associated information, namely the filename where they are described and a textual description of what they represent:

```
<?xml version="1.0" encoding="UTF-8"?>
<solapplus>
  <cube id="1" name="CubeA" filename="cubea.xml" description="A desc." />
  <cube id="2" name="CubeB" filename="cubeb.xml" description="B desc." />
</solapplus>
```

4.5.2. Load Cube

Load cube is the request to get all the necessary information regarding a model so that a client can interact with it.

The XML request contains the cube ID and filename:

```
<?xml version="1.0" encoding="UTF-8"?>
<solapplus>
  <request call="load_cube">
    <params cubeId="1" filename="cubea.xml" />
  </request>
</solapplus>
```

The response includes not only cube-specific data such as dimensions and measures, but also auxiliary information such as connection specifications for the map server, the base map to use and available layers:

```
<?xml version="1.0" encoding="UTF-8"?>
```

```

<solapplus>
  <cube id="1" name="CubeA" description="A description">
    <maps>...</maps>
    <dimensions>...</dimensions>
    <measures>...</measures>
  </cube>
</solapplus>

```

4.5.3. Get Data

Get Data is the request for the actual data to be displayed and analyzed by the user. The base request contains the cube ID and the spatial boolean flag. Specifying the information we want and the restrictions to apply is done by including additional elements inside <request>, as in the following example:

```

<?xml version="1.0" encoding="UTF-8"?>
<solapplus>
  <request call="get_data" spatial="true">
    <params cubeId="1" spatial="true"/>
    <measure id="1" operator="SUM" />
    <level dimensionId="1" levelId="3" />
    <attribute dimensionId="1" levelId="3" attributeId="1" />
    <slice dimensionId="1" levelId="3" attributeId="1" operator="LESS"
value1="100" />
    <spatialSlice dimensionId="1" levelId="3" attributeId="7" layerId="2"
operator="INSIDE" />
    <fieldFilter measureId="1" operator="GREATER" value1="2500" />
    <measureFilter measureId="1" operator="GREATER" value1="100"
measureOperator="SUM"/>
    <nFilter measureId="1" operator="TOP" nRows="10" measureOperator="AVG"/>
  </request>
</solapplus>

```

This example contains one of each possible element in a SOLAP+ request, but there can be multiple elements of the same type. For details on these elements please check the XML Schema for the request messages (attached document).

<measure> element

A measure can be either numeric or calculated. A numeric measure has a set of possible aggregation operators associated with it, so one has to be specified along with the measure ID:

If we want to add a calculated measure, only the ID is needed (no operator required), since a calculated measure has a specific formula associated with it.

<level> element

In order to get data at a certain aggregation level, the server needs both the dimension and the level's ID.

<attribute> element

An attribute is added to the analysis by specifying it's own ID and the respective ID's from the dimension and level it belongs to.

<slice> element

A slice is the base for applying constraints or restrictions on the data. A slice is specified by indicating the attribute on which we want to create a restriction (identified by dimension, level and attribute), the operator to apply and the value to be compared.

Some operators (such as *BETWEEN*) require two values to be used on comparison operations instead of just one; in these cases, an extra attribute (*value2*) is added to the <slice> element. The possible operators are in the XML Schema.

<spatialSlice> element

Spatial slices allow the user to apply spatially-related constraints (the same as the ones presented in section 2.3.2). These require both a spatial attribute and a spatial object (or layer) to be compared according to the chosen operator.

Other operators may require additional attributes on the <spatialSlice> element; for example the *WITHIN_DISTANCE* operator requires a measuring unit (*unit*) and a distance (*value*).

The *NEIGHBOURS* operator allows selecting only the closest N elements to a certain spatial object. In this case, the *value* attribute determines the number of neighbors to select.

<fieldFilter> element

A measure field filter restricts the considered data to only those elements that comply with the respective condition before the aggregation operators are applied. It can only be applied to numeric measures and requires the measure ID, an operator and a value.

<measureFilter> element

This element applies a filter to the aggregate value of a measure and can be used with both types of measures. Just like when adding a measure, if it is numeric, then an aggregation operator is required.

<nFilter> element

This element defines a restriction on the returned rows to the top or bottom *X* rows, ordered by a certain measure. An aggregation operator is required when using numeric measures.

The response to these requests includes two main sections: Spatial information and table information. Spatial information is compiled within the <query> element, containing both a spatial SQL query and the type of geometry to be generated. Table information includes the number of rows in the rowset, the number of measures, the data rowset itself, additional information about the associated attributes, selected attributes' levels and their relation to the considered spatial attribute:

```
<?xml version="1.0" encoding="UTF-8"?>
<solapplus>
  <query sql="SELECT ... " geometryType="polygon" />
  <table count="7" nMeasures="2">
    <rowset>...</rowset>
    <associatedAttributes>...</associatedAttributes>
    <attributesLevels>...</attributesLevels>
  </table>
</solapplus>
```

4.5.4. Get Distincts

When creating a slice on a certain attribute, it is helpful to provide the user with a list of the existing values in the database for that attribute. The base format for this request specifies an attribute identified by its dimension, level and its own ID:

```
<?xml version="1.0" encoding="UTF-8"?>
<solapplus>
  <request call="get_distincts">
    <params cubeId="1" filename="cubea.xml" />
    <distinct dimensionId="1" levelId="1" attributeId="5" />
    ...
  </request>
</solapplus>
```

Optionally, <slice> elements can be added to the request inside the <request> element. The <slice> element has the same syntax as the one presented in the "Get Data" request. Here is an example of a "Get Distincts" request including two slices:

```
<?xml version="1.0" encoding="UTF-8"?>
<solapplus>
  <request call="get_distincts">
    <params cubeId="1" filename="cubea.xml" />
    <distinct dimensionId="1" levelId="1" attributeId="5" />
    <slice dimensionId="1" levelId="1" attributeId="4" operator="EQUAL"
```

```

    value1="X" />
    <slice dimensionId="1" levelId="1" attributeId="2" operator="EQUAL"
    value1="AAA" />
  </request>
</solapplus>

```

The response to this request is basically a list of values, structured as the example below:

```

<?xml version="1.0" encoding="UTF-8"?>
<solapplus>
  <distincts>
    <distinct value="A1" />
    <distinct value="A2" />
    <distinct value="B1" />
  </distincts>
</solapplus>

```

4.6. Meta Model

A meta model file is a metadata description of all needed information for a data model to be used with SOLAP+. This includes not only information about the data warehouse itself, but also about external spatial components to be used, such as the map generator server.

The purpose of a meta model file is mainly to define spatial cubes by mapping the multidimensional model of the data warehouse to the relational model used by the DBMS. A meta model is described by a XML file in accordance with a specific XML Schema (provided in attachment). In order to present these files' structure, we will use parts of one possible XML file that complies with the mentioned schema as examples.

There are three elements at the root level: *databases*, *mapservers* and *multidimensional*. These are related to the relational model, the map servers and the multidimensional model respectively.

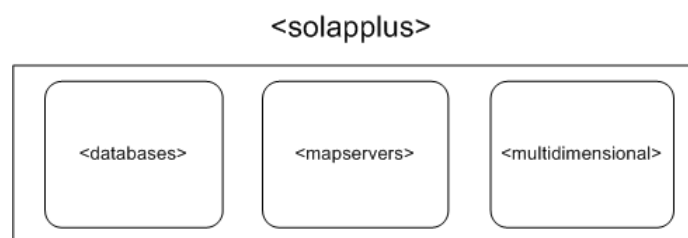


Figure 75 - Meta Model Root Element

4.6.1. Database Element

This element describes the databases available for cube and dimension definition. It contains one *database* element for each available database.

A *database* element contains an identification attribute and two sub-elements: *connection* and *tables*. The first one contains necessary information so that the system can connect to the DBMS, namely the host, port, user authentication and database name:

```
<connection
  name="fct"
  host="localhost"
  port="1521"
  username="SYSTEM"
  password="abc"
/>
```

The *tables* element is constituted by multiple *table* elements, each of which describes a physical table in the database.

table includes an ID, the table name (physical table name as in the database) and sub-elements *columns* and *constraints*. Each *column* contains an ID, a name and a type attribute. *constraints* is where primary and foreign keys for the table are defined. Each of these keys can be constituted by one or more table columns:

```
<table id="1" name="atividade">
  <columns>
    <column id="1" name="atividade_id" type="number"/>
    <column id="2" name="codigo" type="string"/>
    <column id="3" name="categoria_id" type="string"/>
    <column id="4" name="categoria" type="string"/>
  </columns>
  <constraints>
    <primaryKey>
      <column columnRef="1"/>
    </primaryKey>
  </constraints>
</table>
```

4.6.2. Mapservers Element

This element describes the available map servers that can be used for map, theme and style generation. It contains one *mapserver* element for each server.

A *mapserver* element contains an ID attribute and three main elements: *connection*, *layers* and *maps*. The first element contains information necessary to connect to the map server, such user authentication and data source:

```
<connection
  driver="mapviewer"
  host="my-host"
```

```
port="8888"
name="mapviewer"
datasource="solap"
/>
```

The *layers* element contains multiple *layer* elements, each of which defines a layer by referring a certain table and column. For example, the following defines a layer containing the Portuguese rivers:

```
<layer
  id="3"
  tableRef="12"
  columnRef="85"
  label="false"
  name="SOLAP.RIOS"
  title="Rios"
  object="false"
/>
```

The *label* attribute defines if this is a geometrical object layer (false) or a plain text label layer (true). The *object* attribute indicates if this should also be used as a map viewing layer (false) or used only for spatial slices (true).

Finally the *maps* element can contain multiple *map* elements. Each *map* element defines a base map for this model by specifying a set of one or more layers to be used:

```
<map id="1" name="SOLAP.PORTUGAL" title="Países" srid="8307">
  <layers>
    <layer layerRef="1"/>
  </layers>
</map>
```

The *srid* attribute indicates the coordinate system type, which will be used later as a parameter on map requests to the map server.

4.6.3. Multidimensional Element

This is the most complex element in a meta model. It describes the whole multidimensional model (star schema or snow flake), including dimensions and their respective levels/attributes, measures and aggregates. This element is divided into two main categories: *dimensions* and *cubes*.

The *dimensions* element contains multiple *dimension* elements, where each defines a dimension and all its associated information, such as ID, name and table reference. This table is the main dimension table, since snow-flaked dimensions are also supported. A *dimension* element is constituted by *levels* and *hierarchies*:

```
<dimension id="1" name="Actividade" masterTableRef="1">
  <levels/>
  <hierarchies/>
</dimension>
```

The *levels* element has an attribute that indicates which of the following *level* should be considered as the base level. It is important to mention that our model does not consider varying-size levels/hierarchies. This kind of levels/hierarchies was studied by Malinowsky and Zimányi [13] (Also see section 2.2). *Asymmetric spatial hierarchies* are also in this case as they have one or more lower levels that are not mandatory for each member. The few cases that require them can be adapted to our model by creating auxiliary levels.

Each *level* element inside *levels* contains an ID, name, references to its primary, display and sort attribute, a list of *attribute* and an indication of the levels above it. Each *attribute* has an ID, a name and a reference to a column in the dimension's master table:

```
<level id="1" name="..." primaryAttribute="1" displayAttribute="1"
sortAttribute="1">
  <attribute id="1" columnRef="1" name="..."/>
  <attribute id="2" columnRef="2" name="..."/>
  <upperLevels>
    <upperLevel levelRef="2"/>
  </upperLevels>
</level>
```

A *hierarchy* element is defined by an ID, name, type (semantic, spatial or hybrid) and a list of *level*:

```
<hierarchy id="1" name="..." type="semantic">
  <level levelRef="1"/>
  <level levelRef="2"/>
</hierarchy>
```

The *cubes* element can contain multiple *cube* elements. Each one of these has an ID, a name and description, and also references to the respective fact table, database and map server. Inside the *cube* element there are five main elements: *maps*, *dimensions*, *measures*, *aggregates* and *aggregateChildren*:

```
<cube id="1" name="..." factTableRef="6" databaseRef="1" mapserverRef="1"
description="...">
  <maps/>
  <dimensions/>
  <measures/>
  <aggregates/>
  <aggregateChildren/>
</cube>
```

The *maps* element contains a *basemap* element and a *layers* element. The *basemap* has information that will be used to setup the initial map, namely the map reference, the center coordinates and the zoom level. The *layers* element contains a list of the layers that can be applied to this map.

```
<maps>
  <basemap mapRef="1" centerX="-7" centerY="39.56" zoomLevel="1"/>
  <layers>
    <layer layerRef="2"/>
    <layer layerRef="3"/>
  </layers>
</maps>
```

The *dimensions* element defines a list of the dimensions used in this cube:

```
<dimensions>
  <dimension dimensionRef="1"/>
  <dimension dimensionRef="2"/>
  <dimension dimensionRef="3"/>
  <dimension dimensionRef="4"/>
  <dimension dimensionRef="5"/>
</dimensions>
```

The *measures* element contains a list of *measure* elements, each one defining a measure for this cube. It includes an ID, name, a type (numeric or calculated) and a format (which can be used as a guide to format values for display). If this is a calculated measure, we need a formula to calculate the values. In case this is a numeric measure (as it happens most of the time), we need a reference to a column in the fact table and a list of the possible aggregation operators:

```
<measure id="1" name="..." columnRef="60" type="numeric"
format="999999999999.99">
  <aggregationOperators>
    <numeric operator="SUM"/>
    <numeric operator="AVG"/>
  </aggregationOperators>
</measure>
```

To define an aggregate we need an ID, a name and a reference to a fact table as attributes. We also need a list of dimensions and their respective conform dimensions for this aggregate:

```
<aggregate id="2" name="..." factTableRef="17">
  <dimensions>
    <dimension dimensionRef="1" conformDimensionRef="1"/>
    <dimension dimensionRef="2" conformDimensionRef="2"/>
    <dimension dimensionRef="3" conformDimensionRef="14"/>
    <dimension dimensionRef="4" conformDimensionRef="4"/>
  </dimensions>
</aggregate>
```

```
<dimension dimensionRef="5" conformDimensionRef="5"/>
</dimensions>
</aggregate>
```

The *aggregateChildren* element defines a hierarchy among the existing aggregates by indicating, for each aggregate, the aggregates at a lower level. This element can be calculated using the *upperLevels* element in the dimensions. It exists as an auxiliary structure for the Aggregate Navigator component to choose the appropriate aggregates for each query. The *top* attribute indicates that this aggregate is not a child of any other:

```
<aggregateChildren>
  <aggregateChild aggregateRef="1" top="true">
    <child aggregateRef="2"/>
  </aggregateChild>
  <aggregateChild aggregateRef="2" top="false">
  </aggregateChild>
</aggregateChildren>
```


Chapter 5

Implementation

This chapter presents the chosen external components/technologies and implementation details of the prototype

5.1. Implemented Features.....	88
5.2. External Components.....	88
5.3. Implemented Components Overview.....	91
5.4. SOLAP+ Server	92
5.5. SOLAP+ Client.....	94

In the previous chapter we have described our system's architecture. Now we will present an overview of the prototype's features, followed by a description of the technologies/applications used to implement our components, the reasons that lead to that choice and some important implementation details, as well as an interface/interaction description.

5.1. Implemented Features

Regarding the main components, the map, support table and detail table were implemented. The map visualization supports several styles that vary according to the context of analysis (variable size marker, color gradient, bar and pie charts). The support table adapts accordingly to maintain the 1:1 relationship, turning into a pivot-like table whenever needed. The support and detail charts were not implemented.

All of the interaction cases were implemented in the prototype except for case 6 (two spatial attributes from different dimensions) and 7 (spatial measures). Case 6 was not implemented due to time and technology constraints, but it can be included using the current system architecture. Case 7 requires further studying before considering actual implementation.

Visualization clustering is also not present in the prototype. In order to be implemented, it would require some pre-processing server-side. The client architecture would require no major changes, as it would simply present the spatial objects returned by the server. Changes would be focused on interacting with the *ad-hoc* groups, such as expanding/collapsing them.

Calculated measures are not implemented. They are, however, considered in the meta model and communication protocol.

Post-aggregation measure filters and n-row filters are implemented and handled by the server, but they were not implemented in the client simply as a design decision, as they have low relevance in a SOLAP system.

5.2. External Components

Both the Spatially Compliant Data Server and the Map Server components were not implemented, therefore external applications were considered for these roles.

5.2.1. Spatially Compliant Data Server

The Spatially Compliant Data Server is basically a Database Management System which is able to deal with both regular and spatial data, according to the standard Open GIS Consortium (OGC) specifications [10]. The possibility to define indexes over the spatial data in order to improve performance is also a requirement, as operations on this kind of data usually require intense processing.

Two Database Management Systems that fit these requirements were considered: Oracle (with Spatial module) and Microsoft SQL Server 2008. Even though they have similar features regarding spatial data, Oracle has had spatial data implementation for a few versions, while in the SQL Server it has just been integrated in the last version (2008). The decision to use Oracle was based on their experience with spatial data and proof given with previous versions.

Oracle Spatial is compliant with both the OGC standards for data types and operations and also with SQL/MM. Spatial data (points, lines, polygons) is stored in table columns with a specific data type named SDO_GEOMETRY. In order to create spatial indexes that facilitate the execution of spatial operations over these columns, it is necessary to add spatial metadata about the table and column in which we are creating the index.

5.2.2. Map Server

Regarding the Map Server there were also some alternatives such as Oracle's MapViewer [34], Esri's ArcView and MapInfo. Considering that our interaction model needs for representation such as charts, color gradients, legends, etc are covered in Oracle's MapViewer, along with multiple API's and easy integration with the DBMS, it was the chosen application as the Map Server component.

Oracle's MapViewer is a J2EE service for rendering maps using spatial data managed by Oracle Spatial. This application was deployed to WebLogic server [35]. MapViewer has four main components: Map Rendering Engine, Map Definitions, Map Builder and APIs.

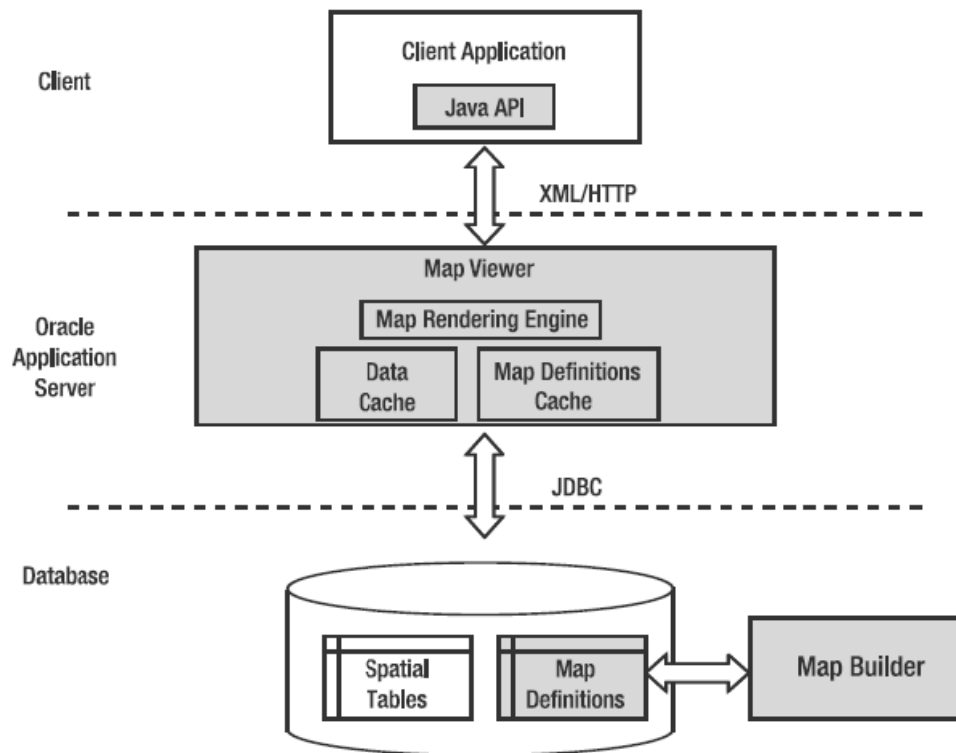


Figure 76 - MapViewer's Architecture (Extracted from *Pro Oracle Spatial*)

The Map Rendering Engine is available as a servlet that accepts and processes client requests and generates responses. It is responsible for fetching the necessary spatial information from the database and producing maps in one of several image formats (GIF, PNG, SVG or JPEG).

Map Definitions is descriptive information, stored in the database, about maps: which tables and columns to use, styles (see ahead) to apply, etc. Map Builder is an external tool developed in order to help with developing base maps, styles and layers, i.e., managing those map definitions.

Application Programming Interfaces (or APIs) allow programmers to access MapViewer's features from a variety of development environments. These APIs include XML, PL/SQL, Java and JavaScript (AJAX) interfaces. Two of these APIs were used to develop the prototype (see section 5.2): XML and JavaScript/AJAX. The XML API is the lower level API; it can be used in any development environment as long as the XML request syntax is respected. It can also be used in addition with other APIs for advanced requests. Oracle Maps is a suite of technologies that allows building mapping applications based on the JavaScript/Ajax API. It includes many useful map control features such as zooming and panning.

In our interaction model, a map is constituted by a reference map and spatial objects. MapViewer also makes this distinction, so in order to generate a map, it requires information about the reference map (*base map*) and the objects to represent (*theme*). The base map is a static map associated with a

certain SOLAP+ model, used in every analysis as the reference map. The theme is a dynamic selection of spatial objects related to the current analysis criteria (slices, filters, etc).

According to our interaction model, the maps we want to present should be customized (such as area colors, marker size or bar charts) taking into consideration the information we are representing on it (type of geometry, number of measures, ...). This customization is made by creating *styles* - information on how to render a map and the spatial objects in it. Styles can be static or dynamic: Static styles are defined once, saved physically in the database and applied to multiple generated maps. Dynamic styles are generated in execution time, specifically for the current map and usually discarded afterwards.

5.3. Implemented Components Overview

The SOLAP+ Server was presented as a component that listens to requests, processes them and returns a response to the client. The most important requirement this component should have is independence from the client's implementation, both regarding operating system and programming language. In order to create a component with such a behavior, we have decided to implement it as a Web service, taking advantage of its properties, mainly the interoperability between various software applications running on different platforms. By using the HTTP protocol, they can work through most firewalls and their text-based structure is easier to understand.

The chosen programming language was Java. The reasons for this choice were mainly because of its portability and exhaustive API support for many tasks. Two of those tasks that are important to mention are the XML messages manipulation and the database connectivity. The first one is related to XML message parsing and generation, both on the client and server components - the Java Document Object Model (DOM) API was used for this purpose. For the database connectivity we used the Java Database Connectivity (JDBC) API - a set of classes and interfaces that allow the execution of SQL queries in a relational database.

Considering that SOLAP clients are used by a wide variety of users with different backgrounds, positions and technical knowledge, it is very important to keep in mind it's usability when designing not only the interface but also the interaction procedures. It was decided to implement the SOLAP+ Client as a Web application - this eliminates the need for program installation on the user's computer (since it just needs a web browser) and makes the update process much easier, as it is a one-side job. To do this, we've decided to use Java Server Faces (JSF) [36], a relatively new and very promising technology that allows the development of rich web applications by combining reusable and customizable components. The application logic is coded in Java and the interface components are used for data presentation and user interaction.

In order to interact with Oracle's Map Viewer, a few API's were available, including *Oracle Maps* - an AJAX API that includes several typical features (zooming, panning, ...) out-of-the-box. This was the main map control API and every map request has to be sent through it, that's why there is no communication between the SOLAP+ Server and the Map Server. JavaScript functions were also created from scratch to deal with user interaction such as layer visualization. A combination of JavaScript and Map Viewer's low-level XML API was also used to generate dynamic themes, styles and legends when requesting maps.

Regarding the Metadata Repository, it was implemented as simple XML/XML Schema files. However, the information required could have been supplied to the server in any other way, as long as the XML elements structure is maintained.

The following picture shows an overview of the main physical architecture, as well as the most important technologies used:

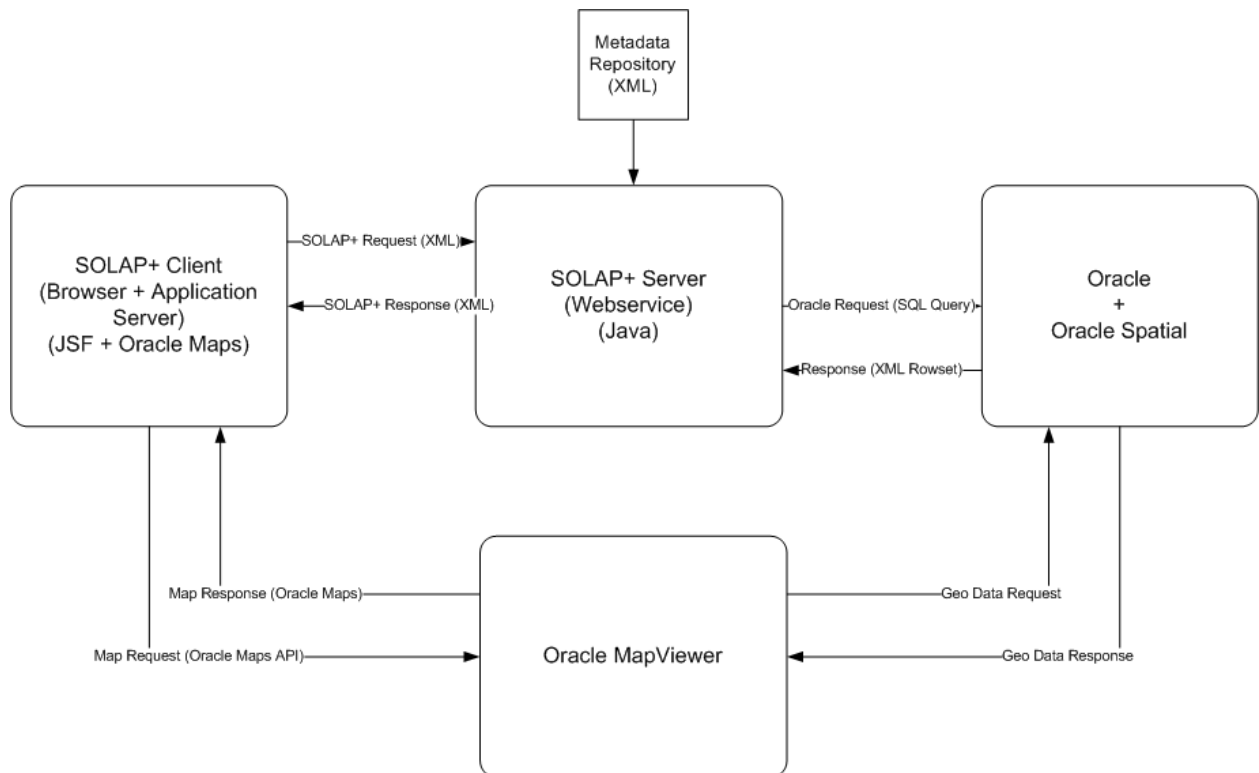


Figure 77 - Implementation Overview

5.4. SOLAP+ Server

The SOLAP+ Server was implemented as a *stateless* web service using Java. This means that each request is handled independently and no previous interaction information is ever needed. This allows a client to be served by different hosts without conflicts.

When a request is received, it is validated using an internal XML Schema. If it is valid, it is parsed using the Java DOM API and the information is stored in *parameter objects*. In case it is a *get data* request, those parameters are processed and *clause objects* are generated. These clauses are actually SQL fragments that are later combined to create SQL queries:

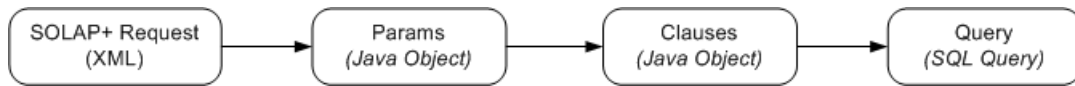


Figure 78 - SOLAP+ Request Lifecycle

There is a particular case when generating a spatial SQL query: When an attribute that causes a measure to split among multiple values (creating multiple columns in the support table) is present in the current session, the spatial SQL query cannot be similar to the table SQL query. This happens because Oracle Maps associates a mapping representation (such as a color or a bar from a chart) to each numeric value in the returned rows.

For example, consider a retail business where the analyst wants to check the sum of the sales for two product types (A and B) in three regions (R1, R2 and R3). This would work perfectly if the rowset returned by the database had a structure similar to the following:

```

<rowset>
  <row>
    <region>R1</region>
    <sales_product_A>5000</sales_product_A>
    <sales_product_B>7000</sales_product_B>
  </row>
  <row>
    <region>R2</region>
    <sales_product_A>1000</sales_product_A>
    <sales_product_B>2000</sales_product_B>
  </row>
  <row>
    <region>R3</region>
    <sales_product_A>700</sales_product_A>
    <sales_product_B>1300</sales_product_B>
  </row>
</rowset>
  
```

In that case, Oracle Maps would associate each numeric value in each row with a bar from a chart, generating three bar charts (one for each region) with two bars each (two distinct values for the attribute, or two numeric columns in each row).

However, if we execute an SQL query where the *product_type* attribute is in the GROUP BY clause (as usual), the returned rowset has this structure:

```
<rowset>
  <row>
    <region>R1</region>
    <product>A</product>
    <sales>5000</sales>
  </row>
  <row>
    <region>R1</region>
    <product>B</product>
    <sales>7000</sales>
  </row>
  <row>
    <region>R2</region>
    <product>A</product>
    <sales>1000</sales>
  </row>
  <row>
    <region>R2</region>
    <product>B</product>
    <sales>2000</sales>
  </row>
  <row>
    <region>R3</region>
    <product>A</product>
    <sales>700</sales>
  </row>
  <row>
    <region>R3</region>
    <product>B</product>
    <sales>1300</sales>
  </row>
</rowset>
```

If we use a similar spatial SQL query in Oracle Maps, instead of creating three spatial objects with two numeric values each, it will create six spatial objects (three of them overlapping the other three on the map, as they refer the same location) with only one numeric value each.

The workaround is then to generate a temporary table with a physical column for each value we want to represent, filled with the actual data to be represented in the client. This means manipulating the returned rowset and generating new (and less) rows. The spatial SQL query returned is then a simple select statement over this temporary table, without any constraints.

5.5. SOLAP+ Client

The SOLAP+ Client was implemented as a browser front-end and a web application deployed in an application server. The next sections describe implementation details regarding the application logic and interface/interaction.

5.5.1. Backing Beans

The SOLAP+ Client has two main responsibilities:

1. Produce XML data requests compliant with the defined communication protocol according to the user's actions
2. Process data responses, producing visual representations

As mentioned before, the client was developed using Java Server Faces (JSF). This technology allows associating interface components with data structures or methods from Java classes, called *backing beans*. The two main backing beans in the client implementation are directly connected to the two main responsibilities listed: `SessionBean.java` for data request generation and `MainBean.java` for response retrieval and data processing.

Session Bean

The session bean contains all the structures of Java objects required to represent and interact with a loaded SOLAP+ Model, such as dimensions, measures and layers. When the user interacts with the components associated with these data structures, events are fired and processed in this backing bean, generating XML request elements. For example, when the user adds a slice, a method in this bean is called, which will create a `<slice>` element to be included in the XML request, based on the options provided by user interaction (attribute, values, slice operator, ...). The SOLAP+ Client interface and respective interaction is presented in the next section (5.5.2).

Main Bean

The main bean is responsible for executing the requests generated by the session bean, as well as receiving the responses, processing them and presenting the formatted data. The spatial data information is forwarded to the Map Control component, which is implemented by Oracle Maps and other JavaScript functions. The processing inside this bean refers mainly to the support and detail tables.

While each row in the detail table is a visual representation of a row in the received rowset from the database, the support table representation requires this rowset to be transformed before it is represented, since it is not a direct mapping of the rowset rows (refer to the Extended Interaction Model chapter).

The support table has two sections, produced separately: Headers and body. Headers are generated taking into consideration the `<attributesLevels>` element present in a SOLAP+ XML data response (see Communication Protocol ?). The content of this element, along with the number of measures (also in the XML response) allows headers to be created and possibly nested depending

on the attributes represented, as described in the Extended Interaction Model chapter. In the table body, the associated attributes are displayed first, followed by the in-line attributes and finally the measure values:

		Headers
Associated Attributes	In-Line Attributes	Measure Values

Figure 79 - Support Table Structure

The associated attributes are the ones present in the <associatedAttributes> response element. In-Line attributes are determined based on <attributesLevels> element. Measures are always the first elements in the returned rowset. Each measure value from a row in the returned rowset will then correspond to a column in the support table - the column number is calculated based on the combination of attribute values and measure it is associated with.

Other Beans

An additional backing bean was created to handle the dynamic panel interface presented in the next section. Each main panel is associated with an object whose properties are changed whenever the user interacts with them, such as minimizing a panel.

5.5.2. Interface / Interaction

For the interface/interaction development, a component library named RichFaces [37] was used. These are enhanced or new JSF components that allow skinnability using CSS.

The SOLAP+ Client interface is constituted by five main panels. Map (1), Support Table (2) and Detail Table (3) are data presentation panels, at the center; Control Panel (4) and SOLAP Model (5) are the side panels:

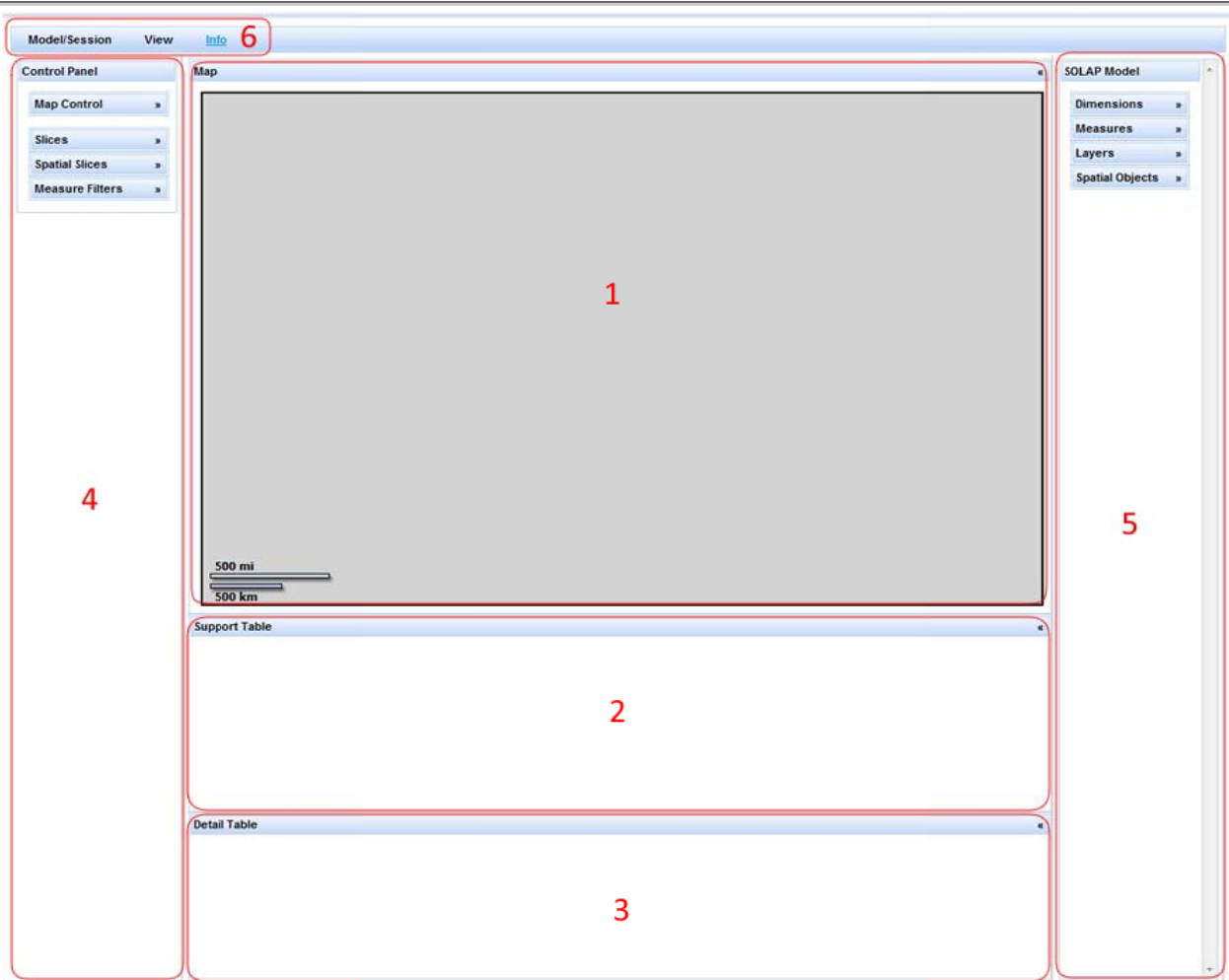


Figure 80 - SOLAP+ Client Interface

All of these panels can be minimized or hidden. When that happens, the remaining panels take up the space. These options produce a flexible environment and allows the user to focus on certain aspects of the analysis by displaying only the key panels at each time. The two side panels' visibility is controlled by the *View* menu in the top toolbar (6). All other panels and sub-panels can be minimized/restored by simply clicking on their title bar. An example of this flexibility is presented in the figure below, where we have minimized both tables and hidden the right panel (SOLAP Model):

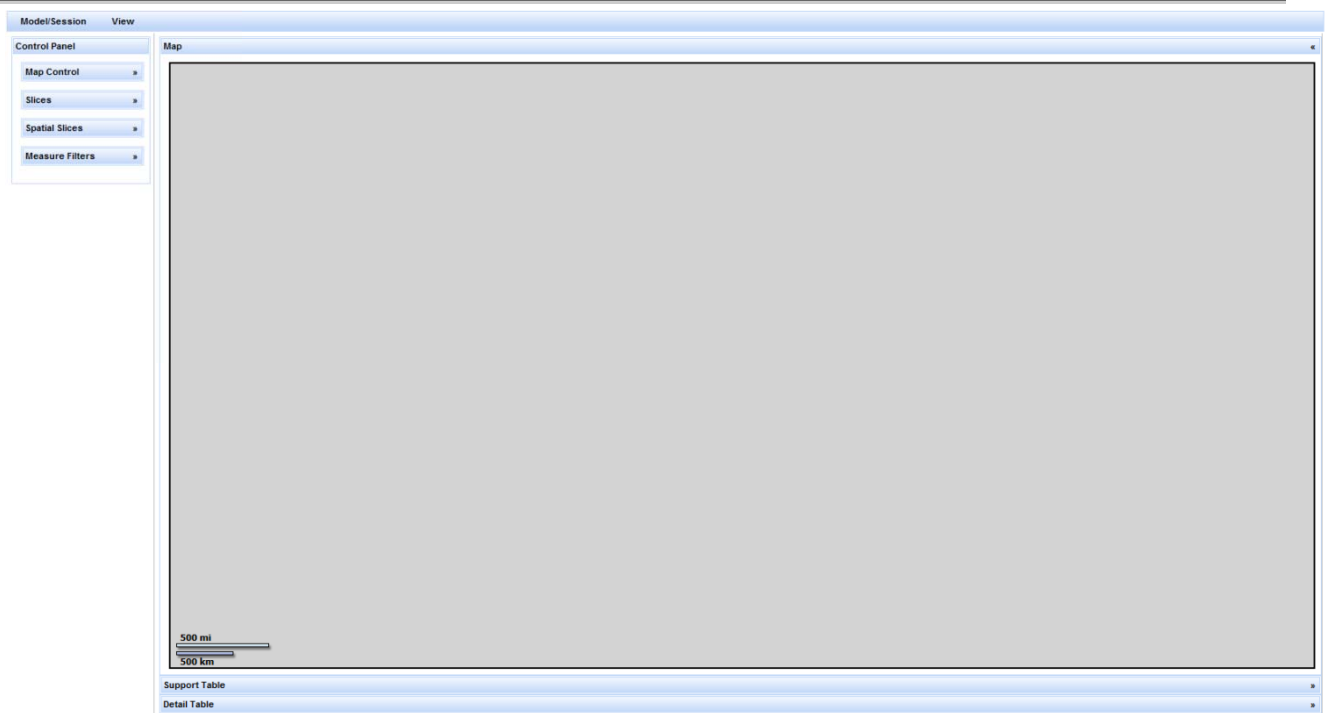


Figure 81 - Interface with both tables minimized and hidden right panel

The SOLAP Model panel (right side) contains the necessary information about the currently loaded cube, such as dimensions (along with levels and attributes), measures (and respective aggregation operators), layers and spatial objects. All of these items are draggable to other panels in order to add elements to the analysis or to create slices/filters.

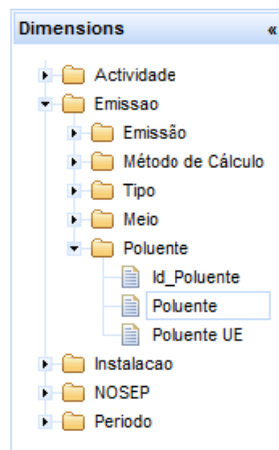


Figure 82 - Dimensions panel where “Emissao” dimension and “Poluente” level are expanded

Just like as presented in our interaction model, the Map is where base maps, spatial objects and legends are rendered. Simple map operations such as zooming in/out and panning are also performed on this component. This component always has a one to one relationship with the Support Table, meaning that to each spatial object represented in the Map always corresponds a row in the Support Table. The Detail Table provides more detailed information about the selected line(s)

in the Support Table. Measures and attributes are added to the analysis by dragging the respective elements from the SOLAP Model panel to either the Support Table or Detail Table.

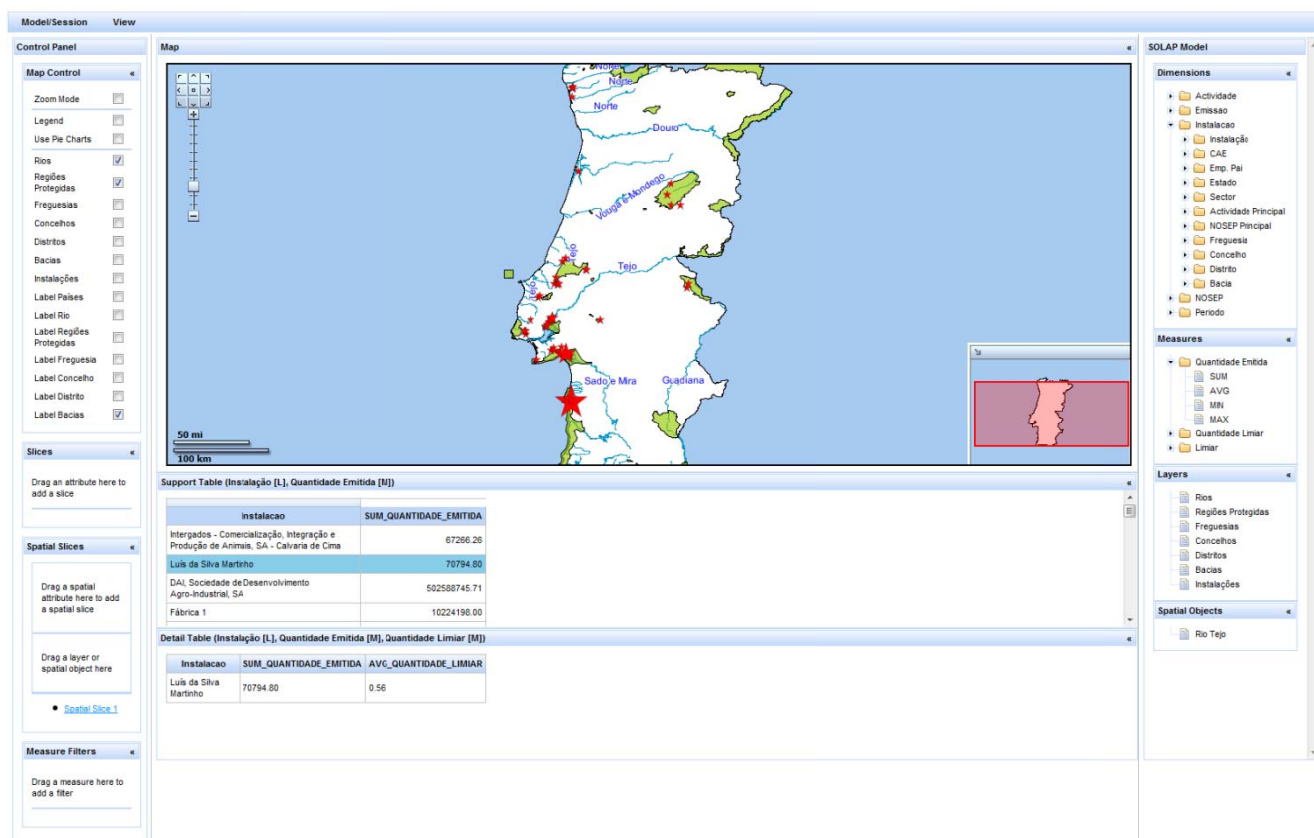


Figure 83 - A typical SOLAP+ session with all panels expanded

The Control Panel (on the left side) has four sub-panels: Map Control, Slice, Spatial Slice and Measure Filter. The first one allows the user to control some of the map display properties, namely switching on and off rectangle zoom mode, legend, usage of pie charts and layer display (Figure 84).

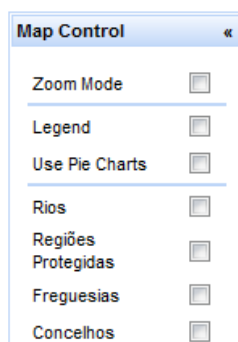


Figure 84 - Map Control detail example

The other three sub-panels under Control Panel are used to create filters/restrictions over the data and to display the active ones (Figure 85).



Figure 85 - Slice panel with an active slice

A new slice/measure filter is added by dragging the appropriate elements from the SOLAP Model and dropping them on the respective panel, which will open the add/edit dialog for a slice (Figure 86), spatial slice (Figure 88below) or measure filter (Figure 89):

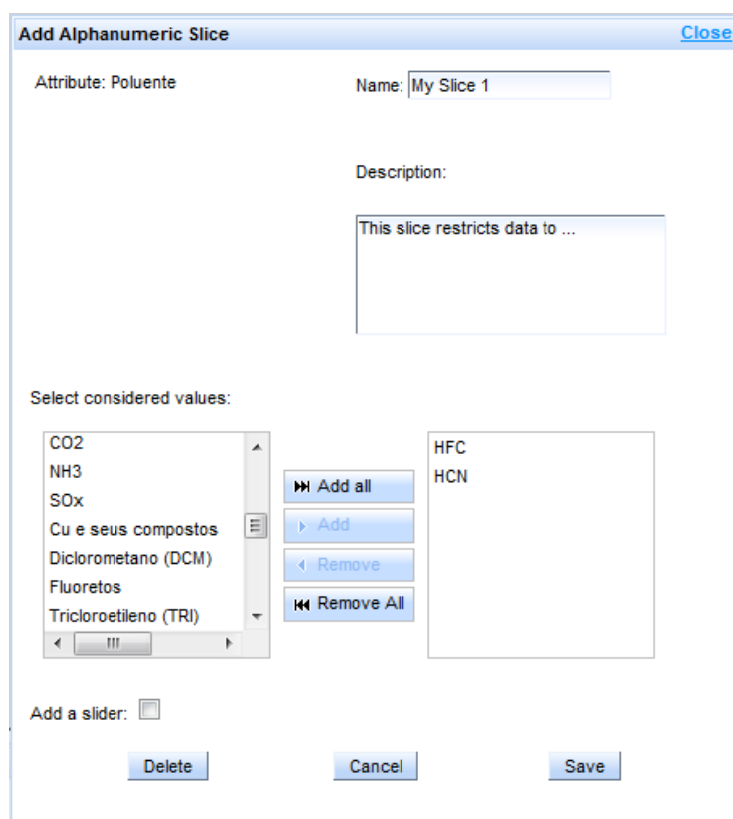


Figure 86 - Dialog box for creating/editing a slice

When adding a slice the user has the possibility to create a slider over the selected attributes. If that option is selected, a slider component will appear in the list of active slices, allowing the user to switch the value to which the analysis is referring to (see figure below).

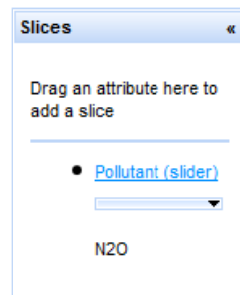
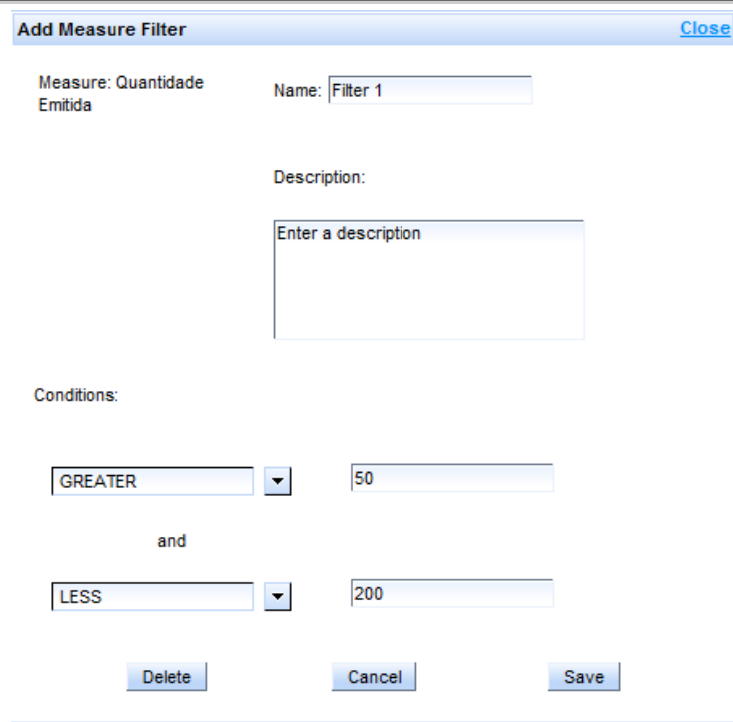


Figure 87 - Slice panel with an active slider

The dialog box for creating a spatial slice is presented to the user when both a spatial attribute and a spatial object (or layer) have been dragged to the spatial slices panel. In this dialog box, the user should select the spatial operator (*inside*, *within distance* or *neighbors*), a value and a measurement unit (if required).

Figure 88 - Dialog box for creating/editing a spatial slice

In the dialog box for a measure filter there are two conditions that will define a value interval (or a single value if both are set to the same). Please note that these are pre-aggregation filters, meaning that they will be applied before measure aggregation operations take place.



Add Measure Filter Close

Measure: Quantidade Emitida Name: Filter 1

Description:

Enter a description

Conditions:

GREATER 50

and

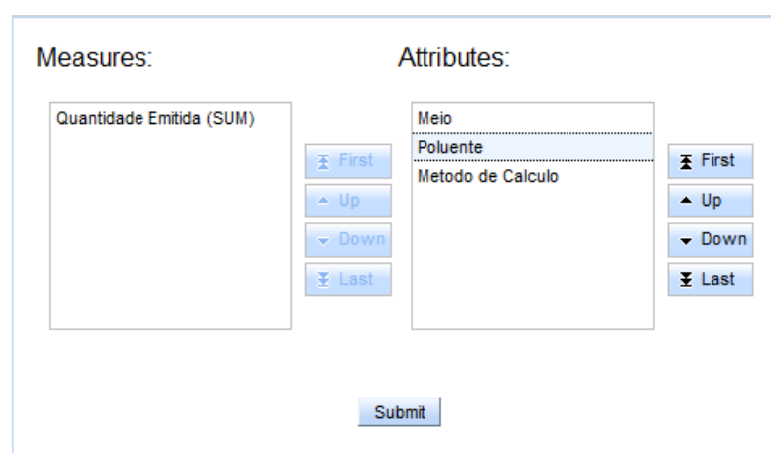
LESS 200

Delete Cancel Save

Figure 89 - Dialog box for creating/editing a measure filter

Any active slices/filters can be edited by clicking their name. The respective dialog box appears where parameters can be changed or the entire slice/filter can be deleted.

The support and detail tables have a context menu (triggered by a right-mouse button click) that allows the user to re-order the attributes and measures' display (Figure 90) or remove them from the analysis (Figure 91) (Note that any measures/attributes in the support table are also in the detail table):



Measures:

Quantidade Emitida (SUM)

Attributes:

Meio
Poluente
Metodo de Calculo

First Up Down Last

First Up Down Last

Submit

Figure 90 - Defining measures and attributes order

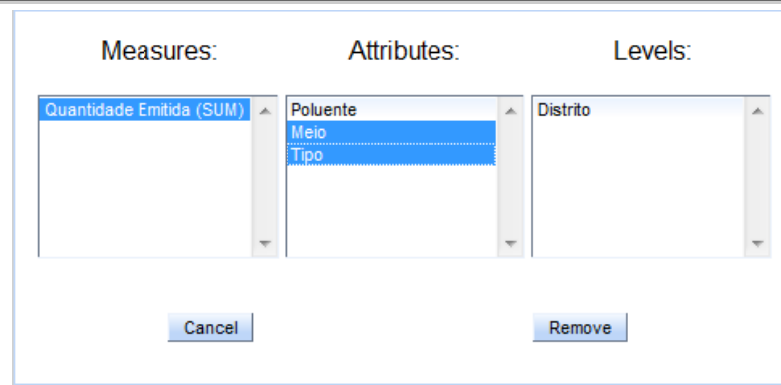


Figure 91 - Removing elements from the analysis

The top toolbar has two options: *Model/Session* and *View*. The first menu allows the user to load a model/session (Figure 92) (whether start a new session for a certain model or load an existing one) and save the current session (Figure 93). The *View* menu allows switching the side panels' visibility as mentioned before.

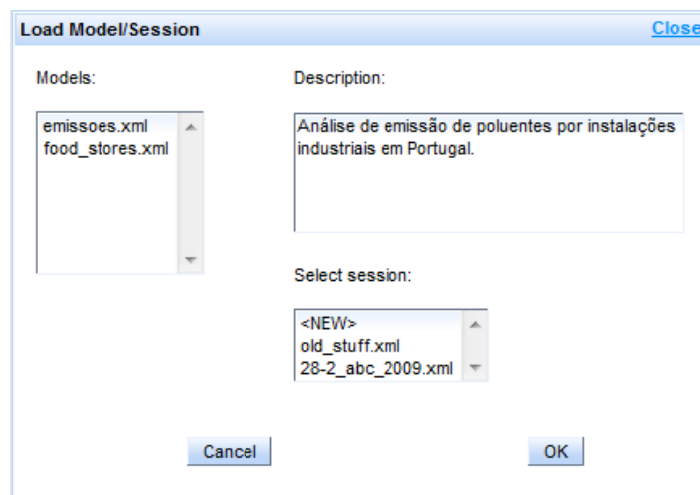


Figure 92 - Loading a model/session

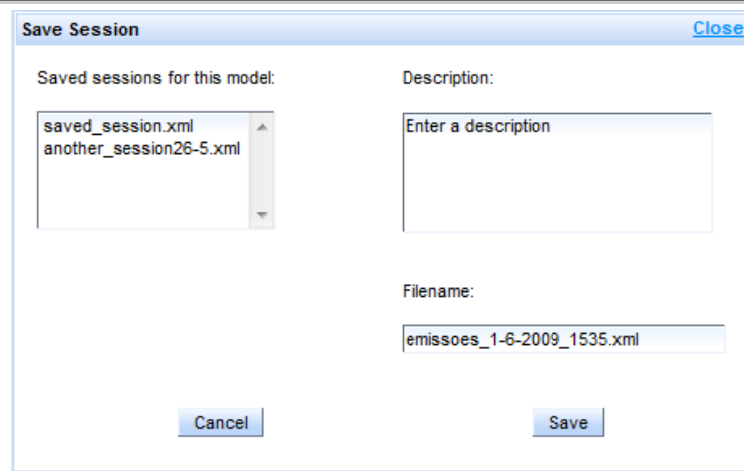


Figure 93 - Saving a session

Chapter 6

Use Case and Validation

This chapter presents a use case and interaction examples in order to validate the implemented prototype.

6.1.	Presentation.....	106
6.2.	Data Model.....	106
6.3.	Interaction Examples	107
6.4.	Conclusions.....	115

The prototype we have implemented is general, i.e., it can be used with many different databases as long as the XML metamodel for that database is provided. In order to test and validate the prototype, we used a specific database for which we created the XML metamodel that describes it.

6.1. Presentation

The Portuguese Environment Institute (*Instituto do Ambiente Português*) is a branch of the Portuguese Ministry of Environment (*Ministério do Ambiente, Ordenamento do Território e Desenvolvimento Regional*). It is responsible for studying, planning, coordinating and giving technical support in the areas of environment management and sustainable development.

One of its objectives is to make reports regarding the environment status in Portugal. To do this, they have records of industrial installations and their respective pollutant emissions. There are characteristics related to these emissions (such as the pollutants involved, the type of emission (air/water), etc) and associated with the installations (geographical and administrative location, main activity, responsible, etc). These emissions are closely associated with spatial data, as each of those emissions occurs at a location and affects an area/region.

Spatially analyzing the available data on pollutant emissions can help to: 1) Obtain indications on air and water quality; 2) identify environmentally endangered zones; 3) uncover possible causes on water course contamination; 4) study the concentration of chemical agents on the ground; 5) establish connections between pollution and health; and 6) verify the compliance with international protocols regarding pollutant emission.

6.2. Data Model

Each industrial installation has an associated primary activity such as energy production or metal transformation. They also have processes related to their industrial activities, named NOSEP.

Each pollutant emission is characterized by the industrial installation, activity, type of emission (pollutant, mean, etc), NOSEP and time. Those activities and NOSEP processes may not be the same as the installation's primary activity/NOSEP.

The data model has the following structure:

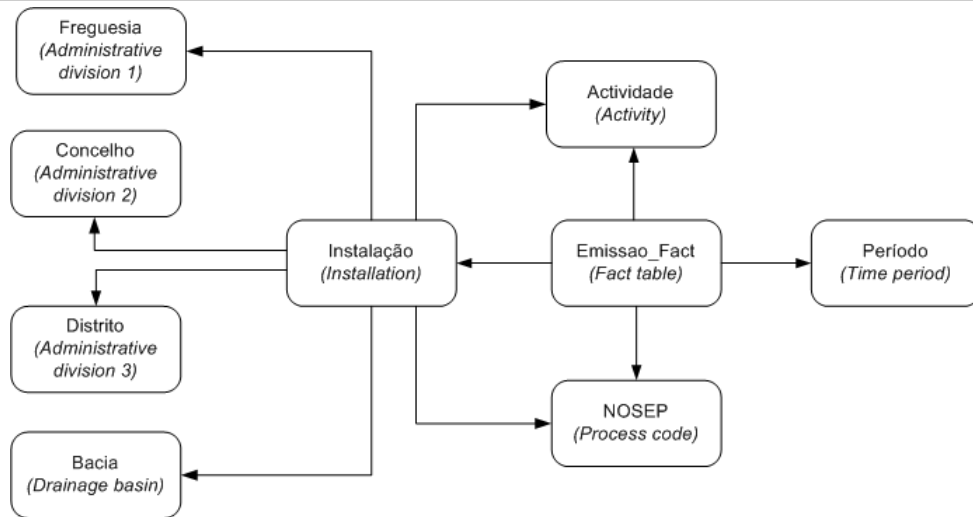


Figure 94 - Data model for the use case

Installation (*Instalação*) is the spatial dimension. It has five spatial attributes: installation location, drainage basin, *Freguesia*, *Concelho* and *Distrito*. The last three are Portuguese administrative divisions. Hierarchically those levels are organized as seen in Figure 95.

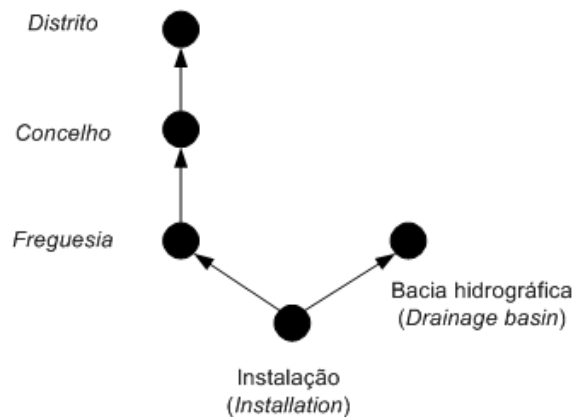


Figure 95 - Spatial hierarchies for dimension Installation

6.3. Interaction Examples

A series of interaction examples using our prototype with the previously described database were produced. These examples are closely related to most of the cases proposed in our extended interaction model.

6.3.1. Example 1

In this first example we wanted to analyze the nitrogen emissions amount for the ten closest industrial installations to the Tagus river (*Rio Tejo*).

To do this we have dragged the desired level at which we want to visualize the data (“Installation” or *Instalação*) to the support table, along with the measure we’re interested in (“SUM of Emitted Quantity” or *SUM Quantidade Emitida*). To limit emissions to nitrogen, we drag the “Pollutant” or *Poluente* attribute to the slice panel and select the appropriate value in the dialog box.

So far this gives us the required information for *all* industrial installations. Since we are only interested in the ten closest to the Tagus river, we add a spatial slice by dragging the spatial attribute (“Point Location” under the “Installation” level) and spatial object to the spatial slice panel. We chose the “NEIGHBOURS” operator and set its value to “10”. The following map and support table are produced (layers containing the rivers and labels for *Distritos* have been activated):

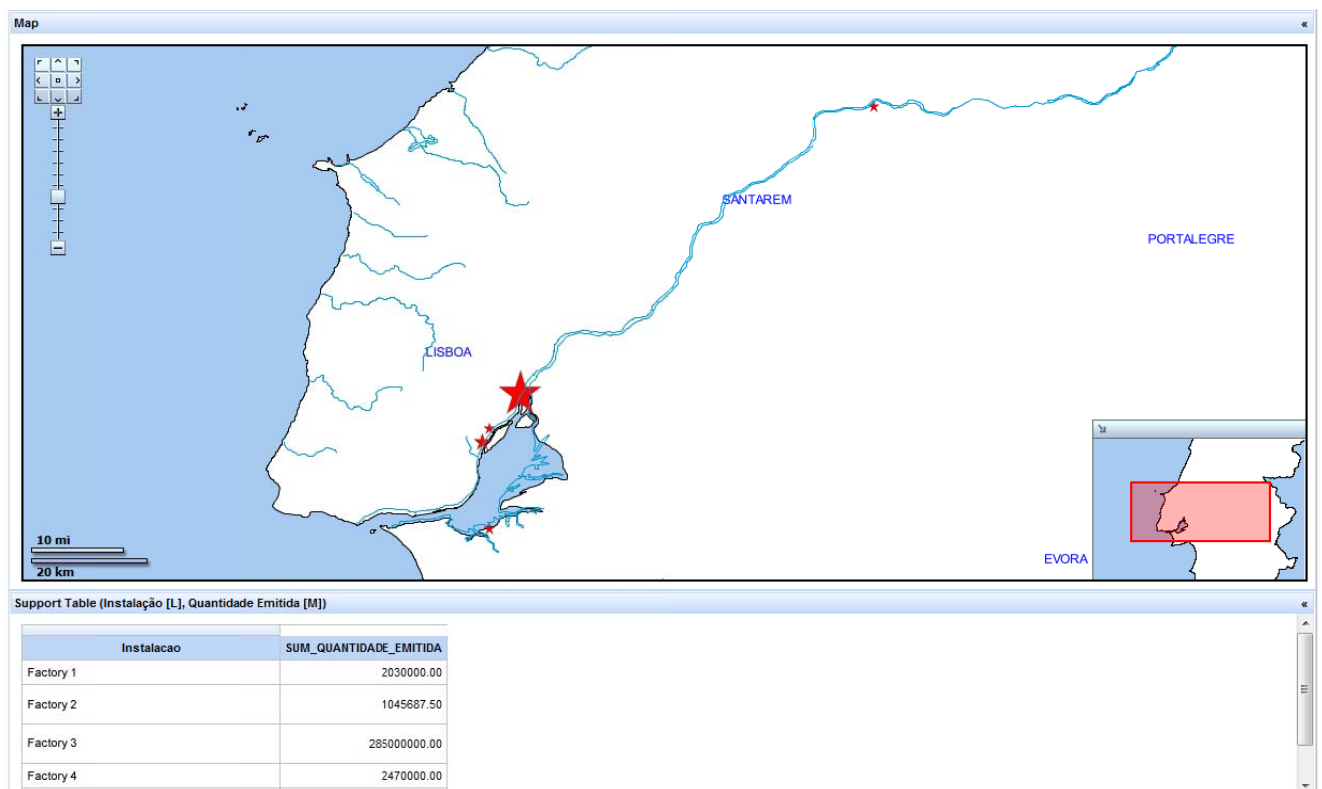


Figure 96 - Output example for one measure and a point spatial attribute

Notice that the spatial objects on the map depend on the applied style. In this case, our prototype has chosen the *variable size marker* style, since we are dealing with only one measure and our spatial attribute is a point. As always, the number of spatial objects and rows in the support table maintain a one to one relationship.

6.3.2. Example 2

For this example assume we want to visualize data at a higher level (*Concelho*) and using a different measure (“SUM Threshold Quantity” or *SUM Quantidade Limiar*). We are also only interested in CO_2 emissions.

Like before we have dragged the respective level and the measure to the support table. To restrict the data to CO_2 emissions, we need to create a slice, so we drag the respective attribute ("Pollutant" or *Poluente*) to the slice panel and select " CO_2 " as the considered value. This interaction produces the following output:

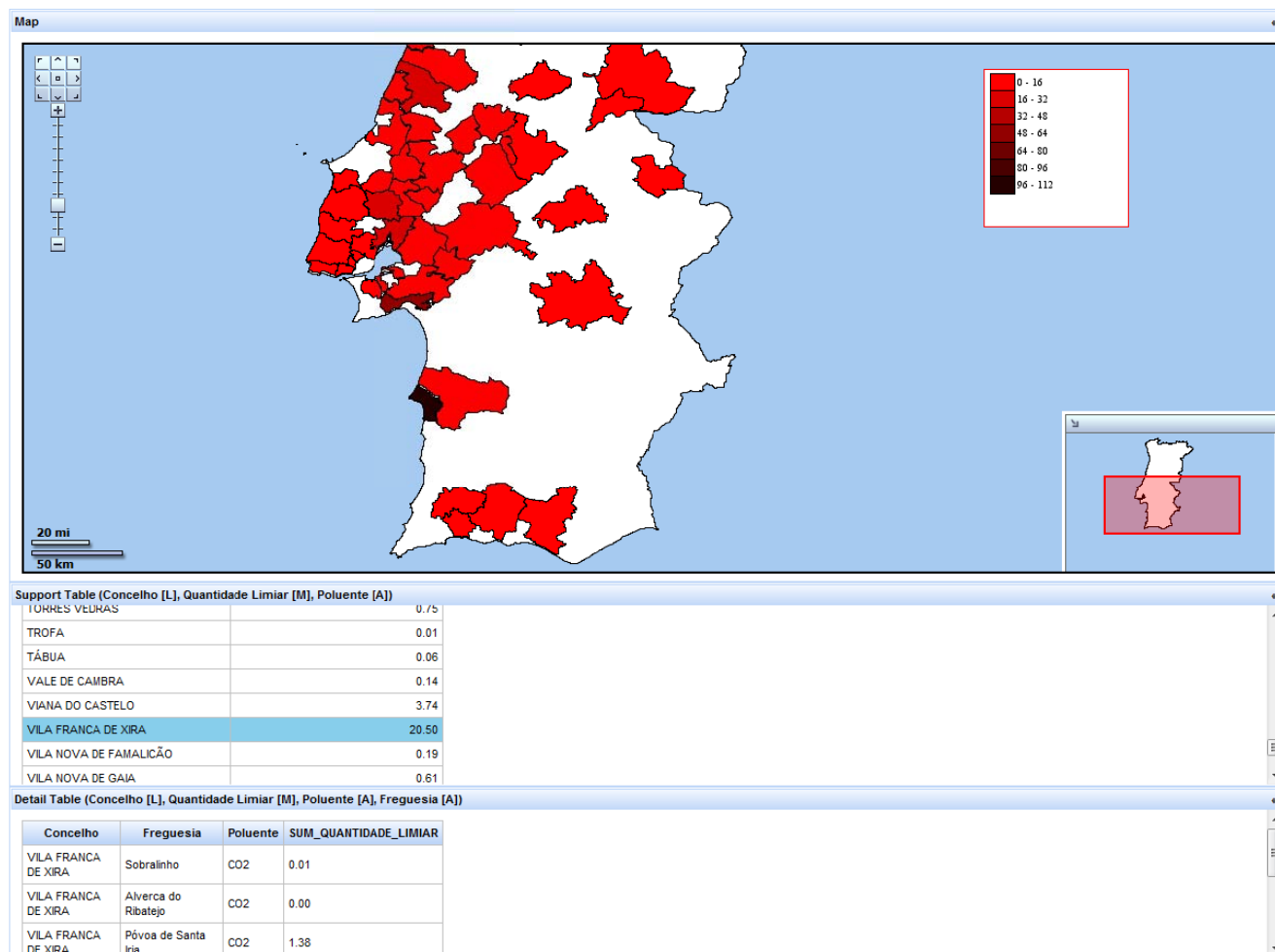


Figure 97 - Output example for one measure and a polygon spatial attribute

Notice that we have included the detail table in this example: By clicking on an element in the support table, that element is detailed below. We have dragged the *Freguesia* attribute (which is at a lower level than *Concelho*, used in the support table and map) to the detail table so that we can analyze data at a finer granularity level.

On the map, the applied style was *color gradient* since we're dealing with one measure and a polygon spatial attribute. We have also activated the legend in the Map Control Panel.

6.3.3. Example 3

In this example we are not interested in an existing administrative geometry. We want to analyze data grouped by the areas resulting from the intersection of *Distrito* and *Bacias Hidrográficas*

(Drainage basins) related to chlorine emissions. To simplify, assume we're only interested in the two North drainage basins, *Douro* and *Norte*.

We have dragged the required level (*Distrito*) and the second spatial attribute to create the intersection ("Drainage Basin Polygon" or *Polígono Bacia*), along with the measure we want to analyze. A slice is also applied to limit the data to the two specified drainage basins and to the Chlorine pollutant. After this, the following output is generated:

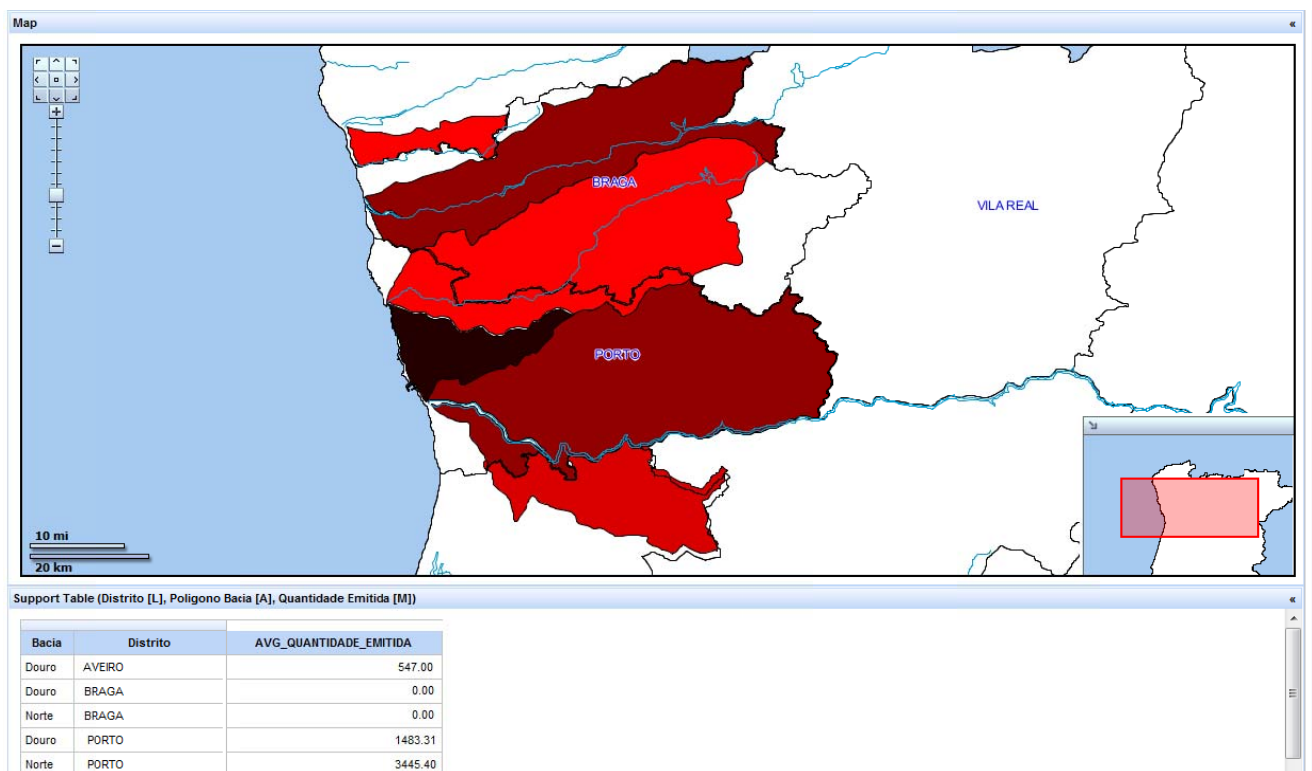


Figure 98 - Output example for one measure and spatial objects generated by intersection

The *Distritos* of *Porto* and *Braga* belong to both drainage basins, so there are two rows in the support table for each of them, as well as two spatial objects that are the intersection of the respective *Distrito* and *Bacia* polygons, maintaining the one to one relationship. *Distritos* polygon and label layers are visible, as well as rivers.

6.3.4. Example 4

So far we have only used one measure in our analysis, representing its values on the map by a *variable size marker* or *color gradient* style, depending on whether the spatial objects were points or polygons. In the next example we want to analyze both the average and maximum values of emitted pollutants at the *Concelho* level, restricted to the *Distrito* of *Viseu*.

This is achieved by dragging the level (*Concelho*) and both the measures we're interested in ("AVG Emitted Quantity" or *AVG Quantidade Emitida* and "MAX Emitted Quantity" or *MAX Quantidade*

Emitida). A slice over the *Distrito* attribute allows us to restrict data to *Viseu* only. This produces the following output:

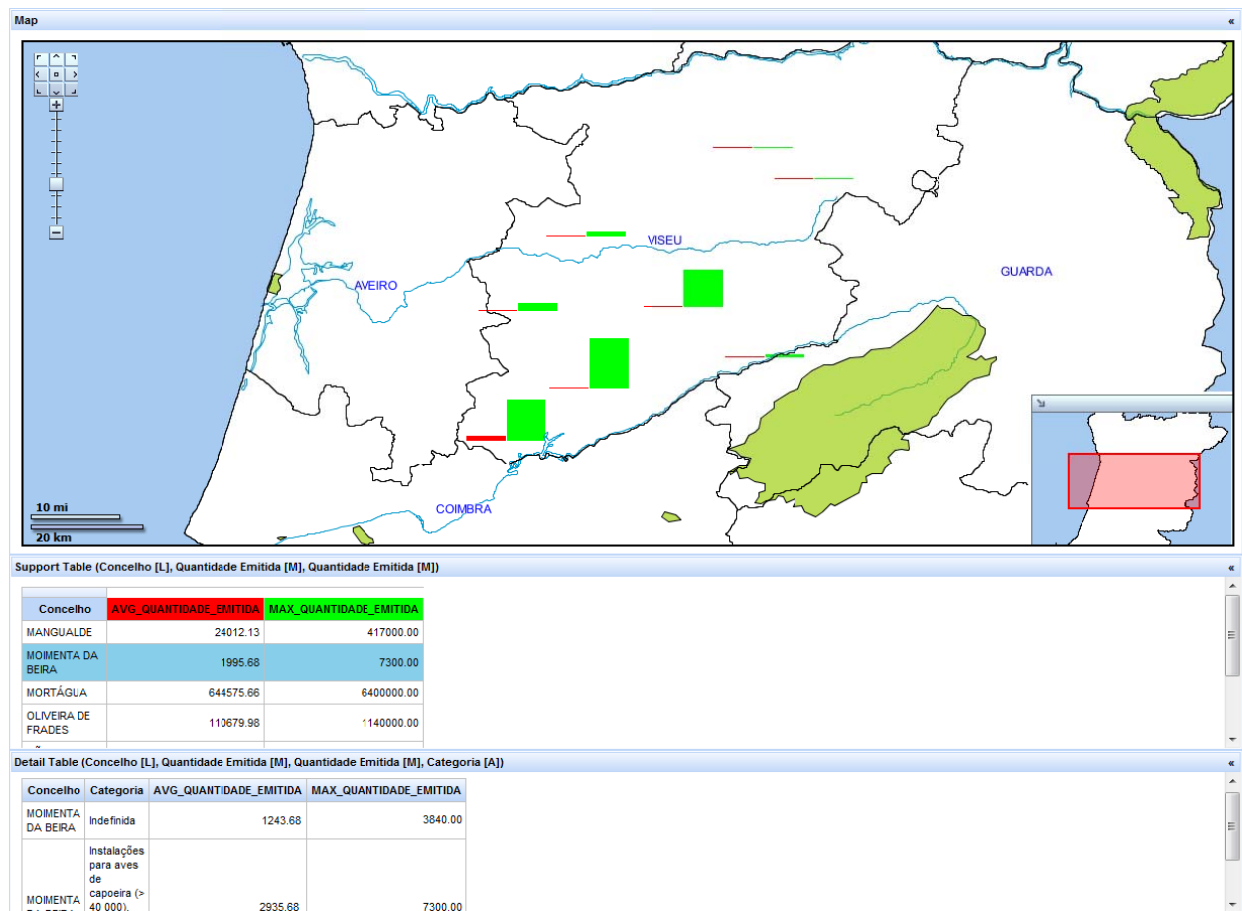


Figure 99 - Output example for two measures

Since we are representing two numerical values, the prototype uses the *barchart* style, where each bar represents a numerical column. This relationship is established by color, present in the bars and the table headers.

We can also see the detail table where we added an extra attribute for visualization - "Categoria" or *Category*. Four layers are also active: *Distritos* polygon layer and their labels, rivers (blue lines) and protected regions (green areas).

6.3.5. Example 5

In this case we are interested in analyzing the pollutant threshold amounts by *Distrito*, but we want to separate air from water pollution. We are also only interested in pollutant emissions made by industrial sites that are within a 5 Km radius from a protected region.

What we need to do is drag the level, the measure and then the attribute we want to use to group the data by, in this case, "Mean" or *Meio*. We drag both the spatial attribute (*Distrito* polygon) and

the layer/spatial object (“Protected regions” or *Regiões protegidas*) to the spatial slice panel, restricting data as required. The “Mean” attribute is at a different, incomparable level than our main level (*Distrito*), therefore it will generate a matrix-like table. Since this attribute has only those two possible values (“Water” (*Água*) and “Air” (*Ar*)), the following is presented to the user:

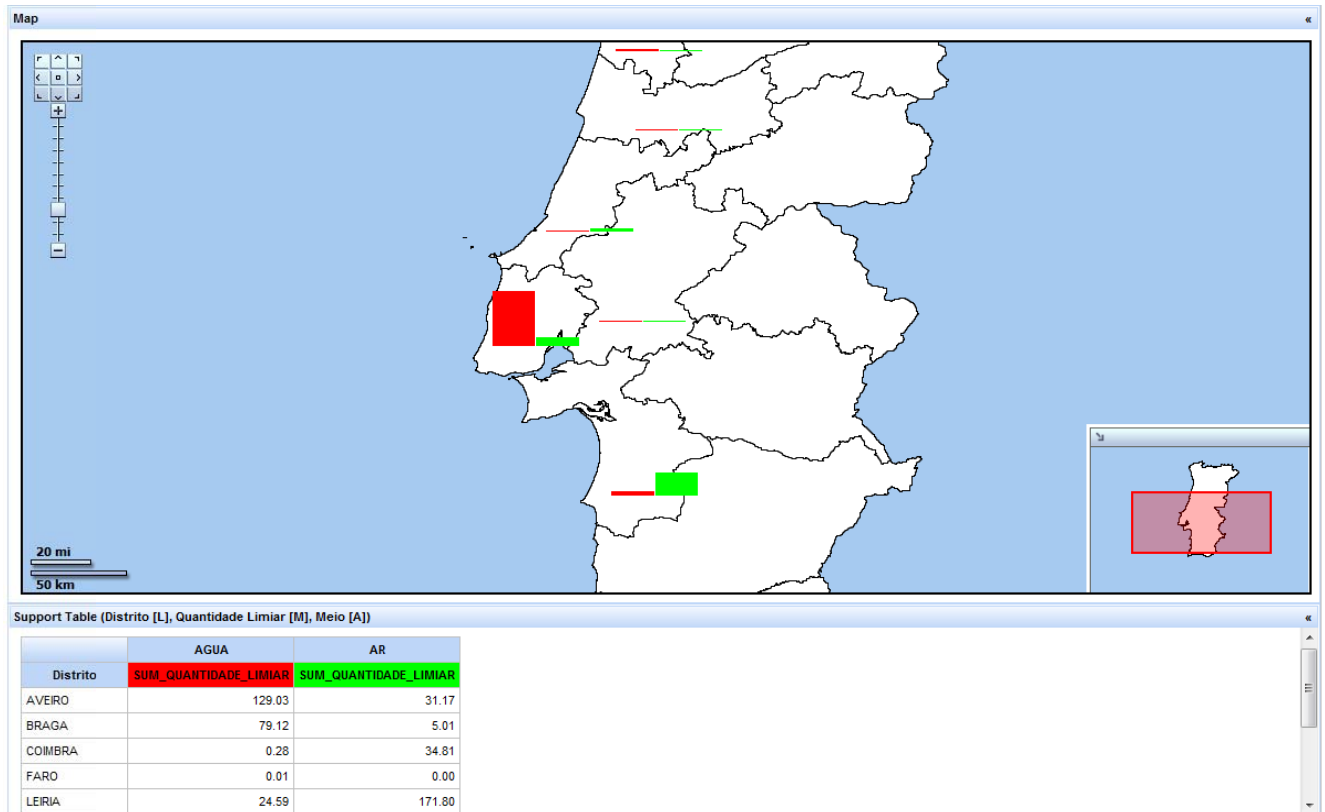


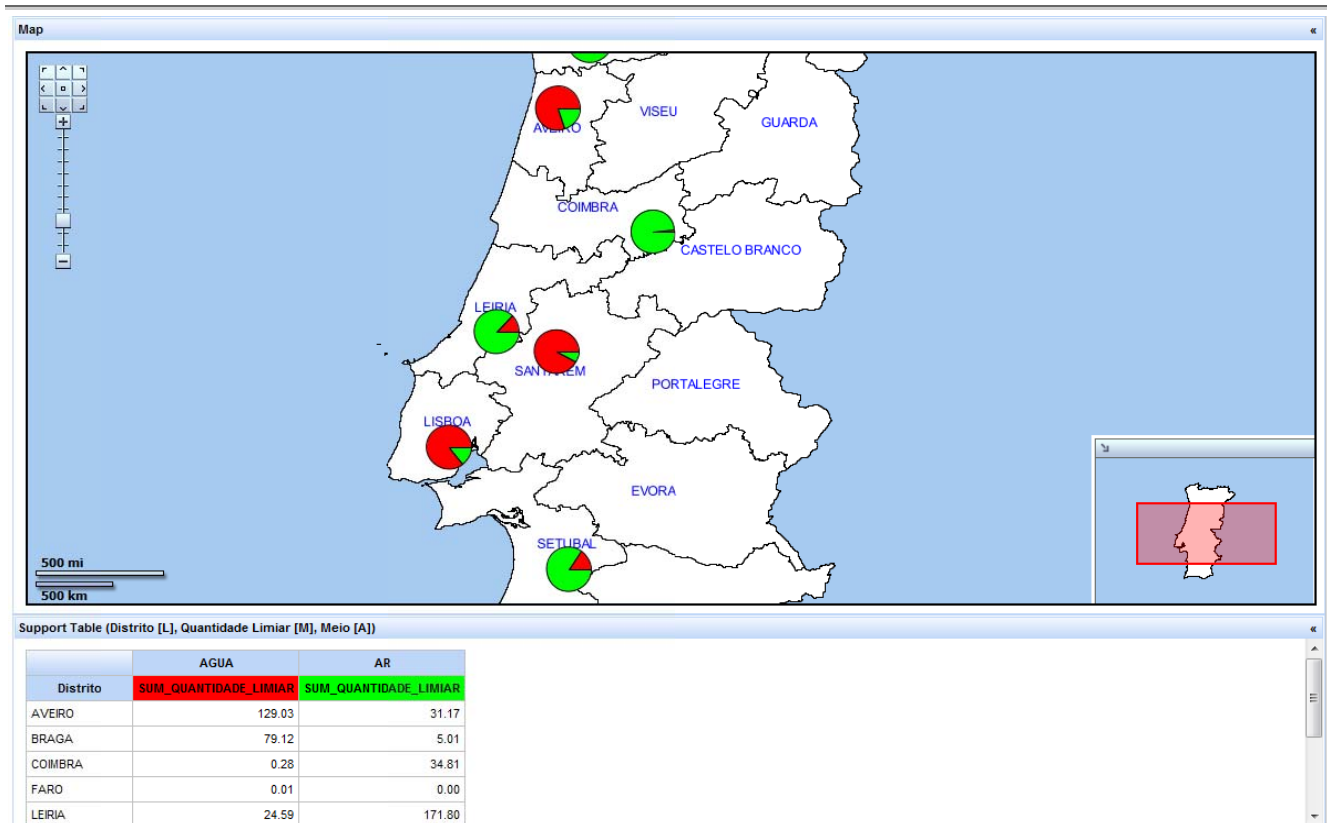
Figure 100 - Output example when using header attributes

Despite having only one measure, it is divided by the different attribute values. In this case there are two distinct values considered, therefore our prototype presents two numerical columns. *Barchart* style is used to represent both values in the map. The *Distrito* polygon layer is also enabled to help visualization.

6.3.6. Example 6

This example is similar to the previous, with the difference that now we are more interested in the proportion of water and air emissions in each *Distrito*, rather than their absolute values. Of course this kind of analysis could be done by comparing the values in the support table for each row, or comparing both bars in each bar chart, however there is a much more clear and helpful style that can be applied: *piecharts*.

Clicking on the respective checkbox causes the prototype to render the map with the new style:



Note that it is logical to use pie charts in this case because total pollutant emissions is equal to the sum of water emissions with air emissions, making them measure components (see section 3.12). The usage of pie charts makes the proportion between water and air emissions much more clear to the user. Color is used to establish a relationship between values in the table and pies in the map.

6.3.7. Example 7

In the next example we want to analyze the sum of quantity emitted by *Distrito*, separated by pollutant and also by the type of water emission (attribute “Type” or *Tipo*), which can be either “direct” or “indirect”. We are also only interested in two water pollutants, *Cianetos* (“Cyanides”) and *Fluoretos* (“Fluoride”).

As usual, the level and measures are dragged to the support table, as well as the two mentioned attributes. Restricting data to the two specified pollutants is done by creating a slice over the “Pollutant” or *Poluente* attribute. Both these attributes are at different or incomparable levels than the main level, therefore they will create a matrix-like table once again:

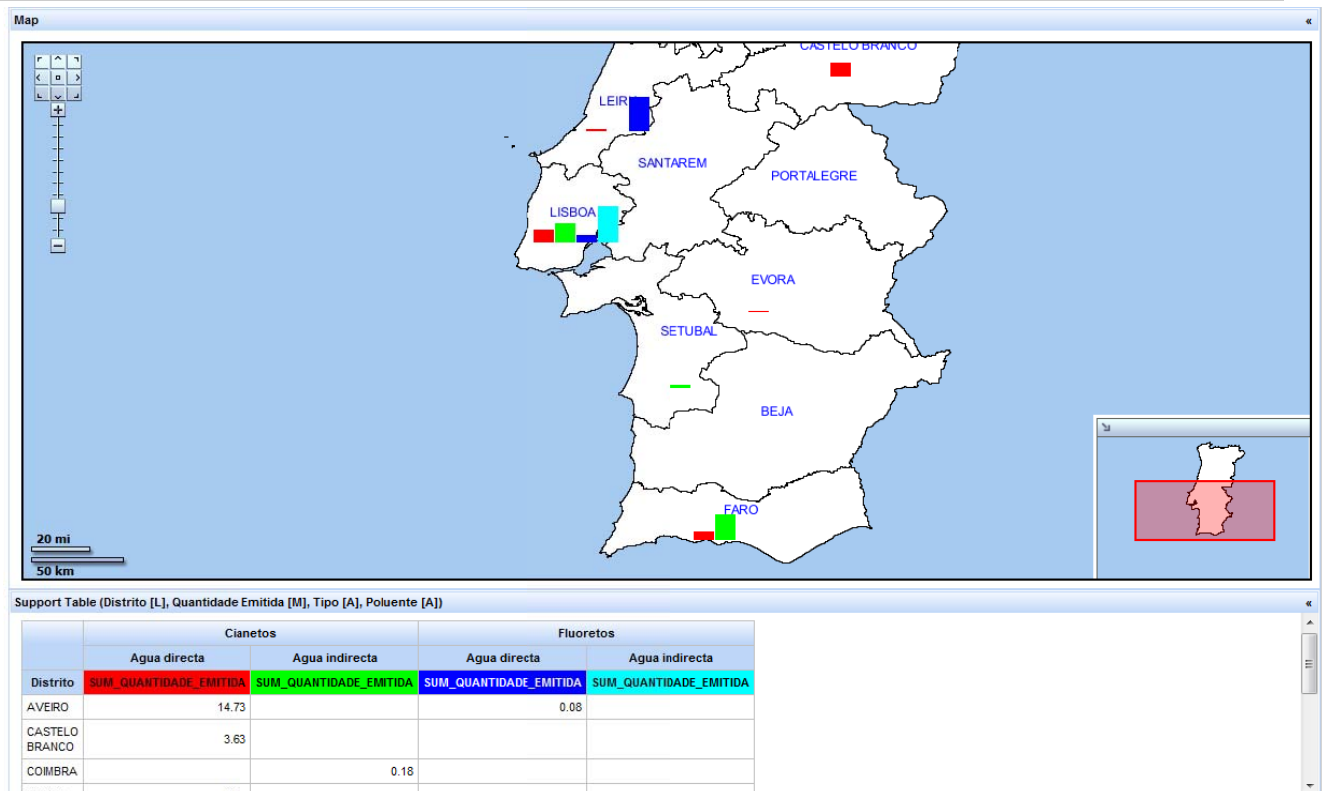


Figure 101 - Output example for two header attributes

The *barchart* style is used to represent the values in the map. As you can see, adding header attributes to the analysis has the potential to greatly expand the support table's columns and respective bars in the map. This issue has been covered in section 3.11.

6.3.8. Example 8

Suppose we want to get an overview of different pollutant emissions. Instead of creating several analysis sessions where we create a new slice in each, restricting data to a certain pollutant, we can create a slider with all the pollutants we are interested in, switching the output by simply dragging a visual element to the next value. In this example we want to get an overview for the emissions of the “Cadmium”, “PM10” and “Trichlorobenzene”, grouped by *Concelho*.

We add the level (*Concelho*) and a slider over the “Pollutant” attribute by dragging it to the slice panel, selecting the possible values and checking the “Add a slider” box.

As we slide the component present in the active slices/sliders section, our tables and map switch and adapt their output according to the selected value. The generated maps would be as presented in the figure below:

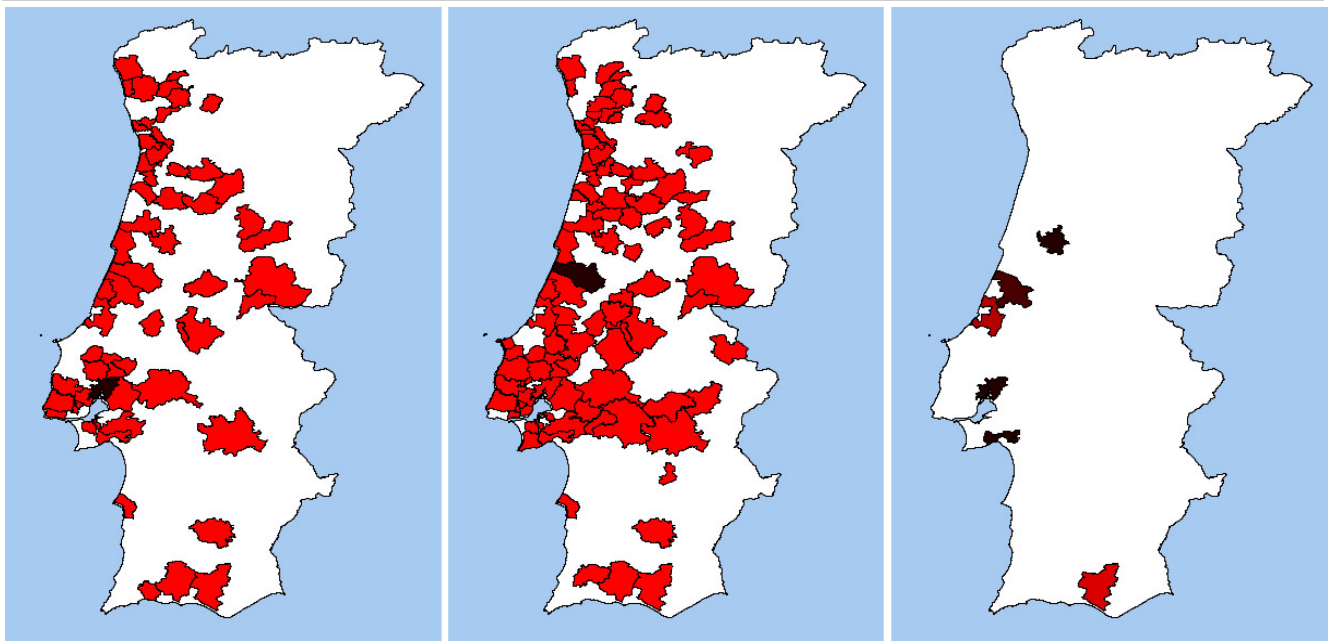


Figure 102 - Generated maps when using a slider for attribute "Pollutant", alternating between "Cadmium", "PM10" and "Trichlorobenzene"

Sliders are also very useful and intuitive when dealing with time-related dimensions. In those cases we could have a slider based on the "year" attribute and visualize data for multiple years by switching the values.

6.4. Conclusions

In the previous sections we have seen some examples of the possibilities of our system applied to a specific, testing environment: 1) Displaying/considering industrial installations according to their geographical position, topological relation to an administrative or natural boundary and proximity to a certain landmark; 2) aggregating and displaying one or more measures using spatial hierarchies to perform drill-down and roll-up operations; 3) visualizing and analyzing data related to areas generated *on-the-fly* by spatial operations; 4) using different map styles to adequately represent different types of values and spatial objects, such as point markers, color gradients, bar charts and pie charts; 5) slicing and grouping data based on attribute values; 6) filtering data based on measure values; 7) displaying layers in order to help visual recognition and discovery; 8) using the detail table to visualize data at a lower, more detailed level and 9) using a slider to quickly switch from a slice value to another, adapting the visualization on multiple components.

Considering the previous examples presented and the expected contributions by using the "Pollutant Emissions" model described in section 6.1, such as identifying environmentally endangered zones or uncover possible causes on water course contamination, we can surely say that the SOLAP+ system is able to provide insight and very helpful spatial and numerical information to an analyst by using all the presented features adequately.

The examples given here were all based on the pollutant emission scenario, however it is important to mention that this is a generic SOLAP system, able to adapt to any case where spatial information is a key analysis feature. From transportation to retail sales, from medical care to civil protection - they all can benefit from the discovery and explanation of spatially-related patterns, trends, anomalies and clusters, helping them perform their duties more efficiently by focusing on the core of the problem.

Chapter 7

Conclusions and Future Work

This chapter draws final conclusions on the design and implementation of this thesis as well as presenting future work activities

7.1. Conclusions.....	118
7.2. Future Work.....	119

This chapter draws final conclusions on the design and implementation of this thesis and presents future work.

7.1. Conclusions

In this thesis we have proposed an extended SOLAP interaction model that provides a user with both spatial and numeric analysis options while keeping a simple and intuitive interaction. This allows users with different backgrounds and positions to be able to use this decision support system without the need for specific technical knowledge. This model adds new interaction cases and visualization options as well as redefining some already presented in previous works. The main contributions regarding the interaction model are:

- Using pivot-like tables in order to open new analysis possibilities or enrich existing ones. This new approach allows the representation of multiple numerical values (both multiple measures and values grouped by attributes) while maintaining the 1:1 relationship between the map and the support table.
- Proposing an innovative approach for the analysis process when dealing with two spatial attributes from different dimensions
- Using clustering techniques to control the amount of spatial objects represented on the map by creating *ad-hoc* groups for both the map and support table, based on the elements' geographical proximity
- Integrating charts associated with the support table including spatial aspects/options
- Using clustering techniques on non-spatial data in order to draw spatially-related conclusions and/or patterns

Even though the subject was not thoroughly studied, the state of the art analysis on spatial measures and the identification of arbitrary spatial measures (those related solely to an event and not the result of a spatial operation between spatial attributes) is worth mentioning.

After the interaction model was defined, an architecture for a system implementing that model was proposed. This architecture is based on a *client-server* approach with external auxiliary components. The SOLAP+ Client provides the user with a graphical interface, converting the interaction into requests for the SOLAP+ Server. Communication between these components is achieved through a *request-response* protocol, in which the SOLAP+ Server is *stateless*, i.e., all requests are treated independently, allowing a client to interact with multiple servers without conflicts. The SOLAP+ Server processes the requests, generates a response and sends it back to the SOLAP+ Client, which displays both the textual/numeric and spatial information in the interface.

A specific communication protocol defining the possible requests from the client and respective responses from the server was created using XML and XML Schema for validation. This allows the

implementation of other client applications in different programming languages or platforms, as long as the XML request structure is maintained.

The SOLAP models to be used in this system also follow a defined XML Meta Model where the necessary information for a certain analysis case is present, such as database, map and multidimensional information. This metamodel description permits to use any multidimensional model in our system, given its XML description.

A prototype of a SOLAP+ system was implemented in order to exemplify some of the proposed features in the interaction model. The main visualization component's behavior is visible when dealing with one or multiple measures and different spatial object geometries (points and polygons) that can be static or dynamic (area intersection). The numeric relationships between the map, support table and detail table are exemplified and preserved in all cases. It allows the addition of alphanumeric slices and sliders, spatial slices with multiple possible spatial operators and measure filters, all using a simple yet flexible GUI based mainly on drag-and-drop items, dialog boxes and collapsible panels. Outside the scope of the interaction model, an aggregate navigator was also implemented in the server, allowing the system to retrieve data from aggregate tables (faster execution) whenever possible, based on the client's requests and the existing aggregates.

Regarding the prototype, the following contributions should also be considered:

- Implementation using standard technologies and a web-based client
- Using dynamic styles for the map, based on a modular architecture in order to suggest appropriate styles depending on the analysis context
- Using aggregates, by extending the meta model and implementing an aggregate navigator that selects and uses the appropriate tables without the need for any user indication

This prototype follows the defined architecture with some minor changes due to technological and time constraints. The implemented features were applied in a case study, exemplifying and validating the proposed interaction model for a fully functional and general Spatial OLAP application.

7.2. Future Work

This section presents future activities related with the SOLAP interaction model. Regarding these activities, the following points are suggested:

Clustering - Further research on dynamic visualization clustering and group creation based on geographical proximity and zoom level can lead to a very useful analysis feature when dealing with large numbers of spatial objects.

Spatial measures - The ability to calculate and visualize arbitrary spatial objects associated with events (facts) would make spatial analysis even more interesting in many areas, providing new analysis options and draw conclusions that are not possible at the moment. However, spatial measures still need profound research in order to reach a well-defined and usable model.

Representing spatial attributes from different dimensions - Due to technological and time constraints, it was not possible to implement the features regarding the representation of spatial attributes from different dimensions presented in our interaction model. A prototype that exemplifies this behavior would be interesting as it would allow a different kind of spatial analysis that is not possible when representing one attribute at a time.

Charts - Some ideas for the usage of support and detail charts were given in our interaction model but not implemented in the prototype. Besides ordering the charts elements based on spatial functions, they could be used to represent sub-sets of data or particular selected elements.

Legend and styles - The process of choosing an appropriate style for a map representation and creating the respective legend is complex. In our interaction model/prototype implementation, we have used a simple approach of selecting a recommended style based on a couple of factors, however, there is a lot of potential in customizing the visualization of data in the map depending on the kind of analysis, attributes and measure types, user preferences, etc. This is a promising research area that would enrich the interaction model and data visualization.

References

1. Power, D.. *Decision Support Systems: Concepts and Resources*. Quorum Books division. Greenwood Publishing (2002). ISBN-13: 978-1567204971
2. Rivest, S. [et al.]. SOLAP: A New Type of User Interface to Support Spatio-Temporal Multidimensional Data Exploration and Analysis (2003)
3. Franklin, C. An Introduction to Geographic Information Systems: Linking Maps to Databases. Database. (1992), p. 13-21
4. Rivest, S.; Bédard, Y.; Marchand, P.. Toward Better Support for Spatial Decision Making: Defining the Characteristics of Spatial On-Line Analytical Processing (SOLAP). (2001)
5. Matias, R.. Integração de Informação Geográfica em Sistemas OLAP. (2006)
6. Vitorino, M.; Caldeira, R.. The Spatial One. (2008)
7. Kimball, R. *The Data Warehouse Toolkit*. Wiley. (2002). ISBN-13: 978-0471200246
8. Oracle® OLAP Application Developer's Guide - Accessed in Januray 9th at http://download.oracle.com/docs/cd/B14117_01/olap.101/b10333/multimodel.htm
9. DMReview- Accessed in January 2nd 2009 at <http://www.dmreview.com/news/5460-1.html>
10. OpenGIS Simple Features Specification For SQL. Open GIS Consortium, Inc. (1999)
11. Fidalgo, R. [et al.]. Providing Multidimensional and Geographical Integration Based on a GDW and Metamodels. (2004)
12. Bédard, Y.; Merrett, T.; Han, J.. Fundamentals of Spatial Data Warehousing for Geographic Knowledge Discovery. Taylor & Francis, Vol. Research Monographs in GIS, No. Chap. 3, p. 53-73 (2001)
13. Malinowski, E.; Zimányi, E.. Spatial Hierarchies and Topological Relationships in the Spatial Multi DimER Model. (2005)
14. Bimonte, S.; Tchounikine, A.; Miquel, M.. Spatial OLAP: Open Issues and a Web Based Prototype. (2007)
15. Bédard, Y. [et al.]. Merging Hypermedia GIS with Spatial On-Line Analytical Processing: Towards Hypermedia SOLAP. (2005)
16. Malinowski, E.; Zimányi, E.. Logical Representation of a Conceptual Model for Spatial Data Warehouses. Springer Science + Business Media, LLC. (2007)
17. Bimonte, S.; Tchounikine, A.; Miquel, M.. GeoCube, a Multidimensional Model and Navigation Operators Handling Complex Measures: Application in Spatial OLAP. (2006)
18. Bimonte, S. [et al.]. GeWOlap: A Web Based Spatial OLAP Proposal. (2006)
19. Bimonte, S.; Tchounikine, A.; Miquel, M.. Towards a Spatial Multidimensional Model. (2005)
20. Matias, R.; Moura-Pires, J.. Spatial On-Line Analytical Processing (SOLAP): A Tool to Analyze the Emission of Pollutants in Industrial Installations. (2005)
21. Tufte, E. *The Visual Display of Quantitative Information*. Graphics Press. (1992). ISBN-13: 978-0961392109

-
22. MacEachren, A.. An Evolving Cognitive-Semiotic Approach to Geographic Visualization and Knowledge Construction.
 23. Statistics Canada. Accessed in December 23rd 2008 at <http://www.statcan.gc.ca/> and <http://132.203.82.216:8081/solap/index.html>
 24. Ng, R.; Han, J.. Efficient and Effective Clustering Methods for Spatial Data Mining. (1994)
 25. Ester, M. [et al.]. A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. (1996)
 26. Sander, J. [et al.]. Density-Based Clustering in Spatial Databases: The Algorithm GDBSCAN and its Applications. (1998)
 27. Zhang, J.; Samal, A.; Soh, L.-K.. Polygon-Based Spatial Clustering. (2005)
 28. Kimball, R.. Aggregate Navigation With (Almost) No Metadata. DBMS Data Warehouse Supplement August 1996. (1996). Access in March 20th at <http://www.dbmsmag.com/9608d54.html>
 29. Kimball, R.. The Aggregate Navigator. DBMS Data Warehouse Supplement November 1995. (1995). Accessed in March 20th at <http://www.dbmsmag.com/9511d05.html>
 30. Adamson, C.. *Mastering Data Warehouse Aggregates - Solutions for Star Schema Performance*. Wiley. (2006). ISBN-13: 978-0-471-77709-0
 31. Stefanovic, N.; Han, J.; Koperski, K.. Object-Based Selective Materialization for Efficient Implementation of Spatial Data Cubes. (2000)
 32. Papadias, D. [et al.]. Efficient OLAP Operations in Spatial Data Warehouses. (2001)
 33. Zhou, X.; Truffet, D.; Han, J.. Efficient Polygon Amalgamation Methods for Spatial OLAP and Spatial Data Mining. (1999)
 34. Kothuri, R.; Godfrind, A.; Beinat, E.. *Pro Oracle Spatial*. Apress. (2007). ISBN-13: 978-1590598993
 35. WebLogic Application Server website at Oracle. Accessed in March 3rd at <http://www.oracle.com/appserver/weblogic/weblogic-suite.html>
 36. Java Server Faces (JSF) website at Sun. Accessed in March 5th at <http://java.sun.com/javaee/jaserverfaces/>
 37. Katz, M.. *Practical RichFaces*. Apress. (2008). ISBN-13: 978-1-4302-1055-9