ISSN 2278-3091
Volume 8, No.4, July – August 2019
International Journal of Advanced Trends in Computer Science and Engineering
Available Online at http://www.warse.org/IJATCSE/static/pdf/file/ijatcse02842019.pdf
https://doi.org/10.30534/ijatcse/2019/02842019

Meta-Modeling of Big Data visualization layer using On-Line Analytical Processing (OLAP)

Allae Erraissi[1], Abdessamad Belangour[2]
[1,2]Laboratory of Information Technology and Modeling, Hassan II University, Faculty of sciences Ben M'Sik, Casablanca, Morocco, erraissi.allae@gmail.com

ABSTRACT

Big data is about collecting, storing, managing, processing massive quantities of daa, but it is also about presenting various insights into the collected data through visualization. Visualization layer is located on the top of a layered architecture composed of Data Sources, Ingestion, Storage, Management, Monitoring, and Security layers. While each of these layers has its own challenges, visualization is presenting a particular challenge because of physical limitations of the display area and the chosen mode of data presentation. Along with the visualization modes, OLAP is a powerful technology for data discovery, including capabilities for limitless report viewing, complex analytical calculations, predictive scenario planning, and visualization. Based on our previous comparative studies in which we identified key concepts of visualization layer of major Big Data distributions, we propose in this paper to map this layer to OLAP visualization technology. To achieve this goal, we apply techniques related to Model Driven Engineering "MDE" to propose a universal meta-model for visualization layer in a Big Data systems, in which we integrated a meta-model of OLAP for data presentation.

Key words: Meta-model, Model Driven Engineering, Big Data, Visualization layer, OLAP, Data Viz.

1. INTRODUCTION

The field of visualization is concerned with how to transform data into a graphical representation. This graphic representation is called visualization. At this point, we deem it necessary to point out that we have identified earlier the key concepts of visualization layer through our comparative studies of major Big Data distributions [1,2]. Our main goal there was to outline the benefits and shortcomings of each distribution and thus gather the pertinent raw material necessary for the study. Yet, the following work is an interim report of our previous proposals for meta-models for layers: Data Sources, Ingestion [3], Storage [4,5], and Management [6,34]. In this article, we are proposing a universal meta-model for visualization layer and OLAP by applying techniques related to Model Driven Engineering 'MDE' [7]. These meta-models together with the previous ones, which we have proposed for the other layers, can be used as an independent cross-platform Domain Specific Language.

2. RELATED WORK

The amount of data created by people, machines, and corporations around the world is growing every year. Thanks to innovations such as the Internet of Things, this trend will continue, giving rise to the creation of Big Data. Indeed, many researchers have worked on big data, particularly on its visualization tools. Accordingly, in our previous works [1,2], we provide a specific comparison of the top five big data distributions. These comparatives studies along with the evaluations done by Forrester Wave [8], Robert D. Schneider [9] and V. Starostenkov [10] on the same Hadoop distributions led us to define key concepts and features of visualization layer. At the level of the global architecture of a big data system, the common architecture is composed of Data sources, Ingestion, visualization, Hadoop Platform management, Hadoop Storage, Hadoop Infrastructure, Security, and Monitoring Layers. In our way for a unified abstract implementation, we proposed in a previous work a meta-model for data sources, ingestion [3], and Storage layers [4,5]. In this paper, we continue our effort by proposing a universal meta-model for visualization layer in a Big Data system including OLAP. The main goal of this universal meta-modeling is to enable Big Data distribution providers to offer standard and unified solutions for a Big Data system.

3. MDA-MDE

The model approaches proposed by the Model Driven Architecture (MDA) [11] and the Model Driven Engineering (MDE) [12] are approaches that come from the field of software engineering. Their primary goal is to streamline and simplify software design processes. For this purpose, they rely on the concepts of models, modeling languages and model transformations. Historically, the goal of the MDA was to reduce the gap between the software (and its abstract models) and the platform on which it must run. MDA's main idea was to rationalize and capitalize on good software development practices. The MDA then considers the software architecture from two points of view: platform-specific (Platform Specific Model) and platform independent model (PIM). The MDA is built around a Y cycle [13]. The objective of this approach is to start from a model realized in a generic framework and to project it in various contexts, according to languages and platforms of specific executions. This approach has brought one of the first important concepts that will be found in the

MDE: the reuse of models. The second important notion introduced by the MDA is that of the modeling language model, also called meta-model. From this notion, the MDA proposed the 4-level pyramid (figure 1) defining the levels of abstraction existing between the different types of possible models.

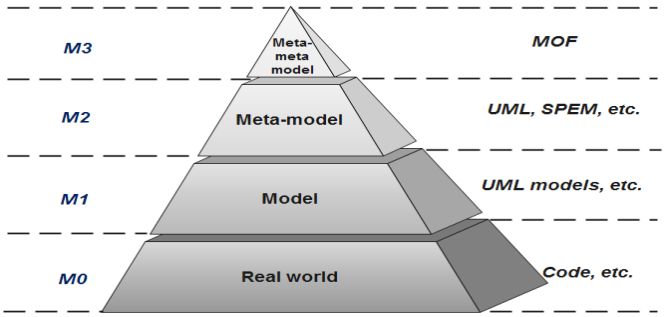

Figure 1: OMG modeling pyramid [14].

A specific action led by the CNRS and bearing the name of Specific Action MDA gave birth to the publication of the book [15] and laid the foundations of the MDE that generalize the MDA. Yet, MDE or Model Driven Engineering puts models at the heart of software development. Like the MDA, MDE is based on the notions of model, meta-model, modeling language, and model transformation. It is crucial since it makes it possible to envisage a more rational implementation of software systems and thus allowing the integration of heterogeneous domains. Eventually, model transformations allow the analysis, modification or synthesis of a model according to rules defined at the meta-model level. In fact, several types of transformations are possible: exogenous transformations and endogenous transformations. On the one hand, exogenous transformations are used to translate a model defined in a modeling language into a model in another modeling language. a typical example is code generation. On the other hand, endogenous transformations are used for transforming a model into another model within the same modeling language. The best example is the optimization of a model (refactoring, etc.).

4. VISUALIZATION

4.1 History of visualization:

The field of visualization is concerned with how to transform data into a graphical representation. This graphic representation is called visualization. At this stage, it is important to give a brief history of visualization by presenting two examples: one by William Playfair [16] and the other by Dr. Jhon [18]. Although visualizations, especially in the form of maps, have existed since antiquity, the graphic representation of abstract information emerges especially in the nineteenth century. William Playfair (1759-1823) is one of the pioneers of the graphic representation of information. This Scottish engineer and economist is the inventor of several types of visualization. His inventions such as the temporal curve and the histogram are widely used today. One of his best-known visualizations, visible in Figure 2, combines these two idioms to compare the evolution of the price of wheat and the weekly salary of a "good mechanic". This visualization, along with a few others, accompanies Playfair's letter on agriculture to the British Parliament in 1821 [16]. It goes on to say that the price of wheat has never been so cheap relative to the cost of labor. Here, the visualization is used to expose the conclusion of an analysis to convince an interlocutor. We are talking about descriptive visualization.

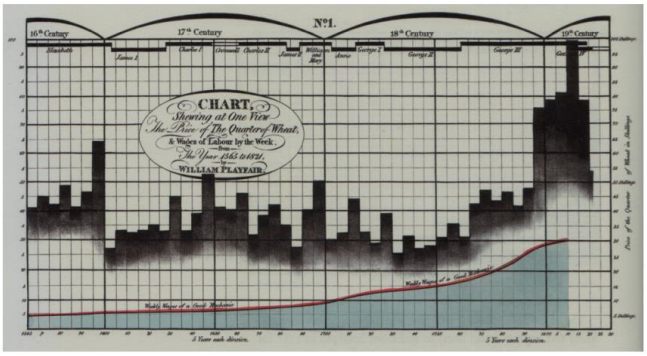

Figure 2: Diagram of wheat price by William Playfair [16,17]. This visualization puts the evolution of the wheat price in perspective with that of a weekly reference wage. Playfair wants to show here that wheat has never been cheaper if we take into account the evolution of wages. Even if the information is well contained in the diagram, the conclusion still requires an additional analysis step. In representing the relationship between the two values, this conclusion would be more obvious.

Visualization can also help the decision by making the data collected clearer. This is clearly illustrated by the map produced by Dr. John Snow during the London cholera outbreak in 1854 (see Figure 3) [18].

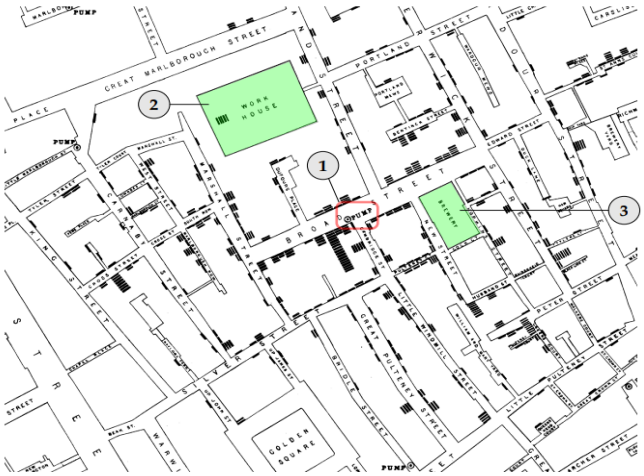

Figure 3: Detail of the map of London established by Dr. John Snow during the 1854 cholera outbreak. If we look closely at the map, we can realize that each point represents a death attributed to the disease. The map also shows the location of the pumps used by the population to draw drinking water. As Dr. Jhon found a greater density of deaths around the Broad Street pump (1), he deduced that the area was probably infected. However, two areas of the low density of deaths can be detected in the map despite their proximity to the pump. The first one is a "workhouse" (2), where only 5 of the 530 residents became ill. The second one is a brewery (3), where no case was recorded. In both cases, the inhabitants did not consume water from

the Broad Street pump. The "workhouse" had its own well and the brewery gave a ration of beer to its employees.

In brief, these visualizations, like all those of their time, were specially designed for a particular case. Besides, they were made by hand because no automation process was available at that time. The realization of visualization needed a lot of time and its image remained statics. Conversely, today and with the rise of computing, it is possible to automate many aspects of visualization. Firstly, the final drawing (conversion into an image) can be done automatically. Secondly, the processing of the information to produce a type of visualization can be entrusted to algorithms, and thus making the comparison of several datasets easier and faster. Lastly, visualization is no longer conceived as a static image, but as an interactive system, on which the end user has feedback power. The raw data must follow certain steps before being presented to the user, who can then act on each transformation to modify the final visualization and learn new lessons. This process is summarized in the visualization pipeline (see Figure 4), originally developed by Haber and McNabb [19] for scientific visualization and later enriched by Dos Santos and Brodlie [20] for information visualization.

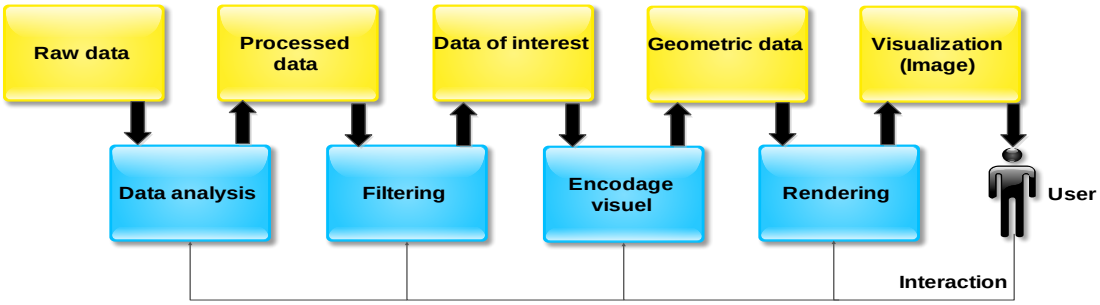

Figure 4: The pipeline of visualization, adapted from [20]. In an interactive system, it can be executed many times per second.

This pipeline is divided into several stages that allow you to go from raw data to visualization:

- The data preparation stage makes it possible to choose what will be visualized. At this stage, the raw data are refined in order to keep the interesting aspects or to derive existing data values (average, difference, map projection ...). The result of this step is a set of more or less structured data ready to be transformed into visualization.
- The prepared data is then filtered to retain only a subset of interest. This filtering can be, for example, temporal (the result of the exploitation of a particular year), semantic (only the friends of a person of interest) or even geographical (election results on a commune).
- Once the data of interest is identified, the step of visual encoding comes. It there where it is decided in what form the data would be represented. Indeed, this step produces a geometric representation that describes the desired visualization. It is at this stage that data is transformed into visualization.
- Finally, the rendering step consists of drawing the geometry produced by the visual encoding step. This step is most often performed by a computer and may involve the use of a graphics processor. The resulting image is the final visualization that can be presented to the user.

As a matter of fact, all steps in this pipeline can be modified at any time by user interaction. Accordingly, when using an interactive system, part or even the entire pipeline can be executed several times per second. In most cases, the user's interactions focus on the filtering and visual encoding steps. The purpose of an information visualization system is to allow the user to derive meaning from what is presented to him. The various steps followed by the user in his exploration of the data are summarized in Shneiderman's mantra: "Overview first, zoom and filter, then details-on-demand." (Ben Shneiderman [21]). The system must first present an overview of the data, permit to filter and enlarge interactively, and hence display the details on request of the user.

4.2 Big Data visualization tools:

It is sometimes difficult to present data in an understandable way to people who are not specialized. Fortunately, some tools make it easy to visualize data. These are Data Visualization tools, abbreviated as "Dataviz" [22]. Data visualization is a general term that describes an effort to help staff understand the meaning of data by placing it in a visual context. Therefore, data visualization software will highlight and identify patterns, trends, and correlations, which might go unnoticed within textual data.
Today's data visualization tools go beyond the standard charts and curves used by Excel spreadsheets. They display data in more sophisticated modes, particularly through infographics, dials and gauges, maps, sparklines, heat maps, and detailed bar graphs in sectors and progress (fever chart).

Images can incorporate interactive capabilities that allow the user to manipulate or explore the data for querying and analysis. These tools may also have indicators designed to alert the user when the data is updated or when previously defined conditions are met. The majority of Business Intelligence (BI) [23] software vendors incorporate data visualization tools into their products, either by developing their own technology or by sourcing from visualization vendors.

5. PROPOSED META-MODELS

5.1 Meta-model of data Sources and Ingestion Big Data layer:

Relying on our previous work, we have succeeded to propose meta-models for layers Data Sources, Ingestion [3], Hadoop Storage [4,5], and Hadoop Platform Management [6] of Big Data architecture. Correspondingly, our evaluation of the Big Data architecture [24] lead us to find that there is a direct link between the Data Sources, Ingestion and Management layers, and that all sorts of data must pass through by the steps constituting the layer of Ingestion before being used by the other layers of the Big Data system. Yet, we have expressed this link with the following meta-package diagram that represents the meta-packages IngestionPkg, DataSourcesPkg, HadoopPlatformManagementPkg, and VizualisationPkg and the dependency relationship that links them:

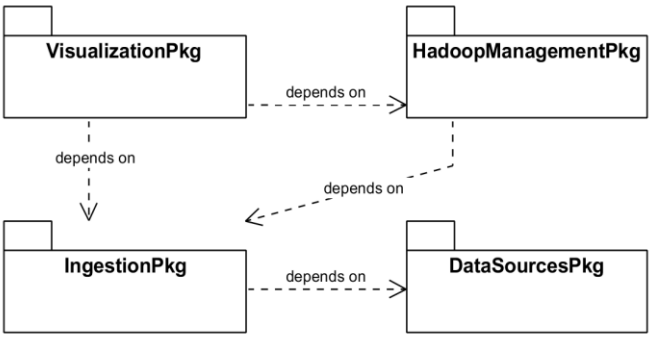

Figure 5: Ingestion, Data Sources, Hadoop Management, and Visualization Meta-Packages.

The following figure shows the meta-model that we proposed for the two Data Sources and Ingestion layers:

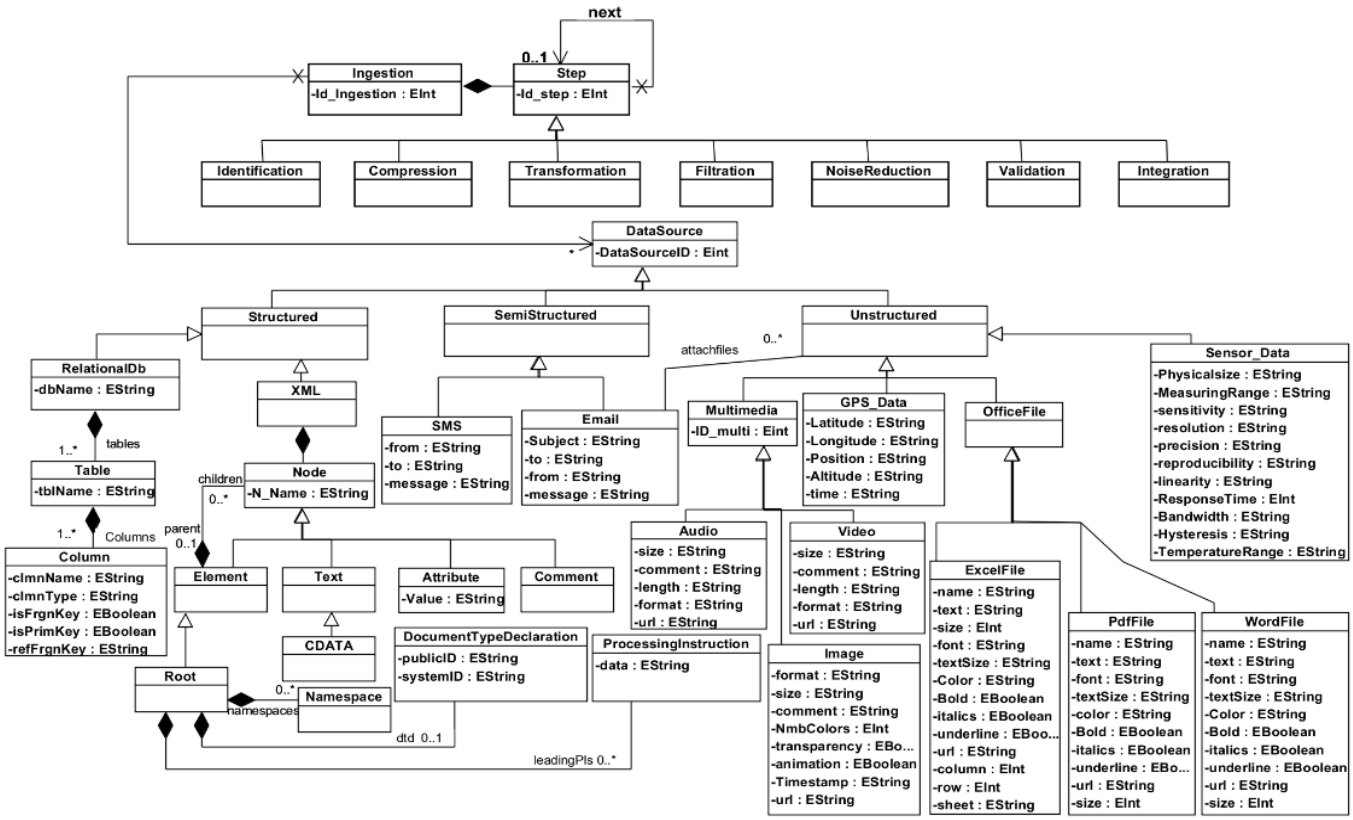

Figure 6: Generic Meta-Model of Data Sources and Ingestion Big Data layers [3].

5.2 Meta-model of Visualization Big Data layer:

In continuous efforts, we have found that companies are faced with an increasing amount of data transiting through their information system. This mass of information makes it difficult for them to operate properly. Therefore, there was an urgent need for a Big Data system to have a visualization layer that will make it possible to exploit this large amount of data in order to generate reports and make decisions. At this point, it is important to show that the meta-model we proposed for visualization layer includes a main meta-class called Visualization, which groups the meta-classes: HadoopAdministration, DataAnalystIDE/SDK, and VisualizationTool. The meta-class Visualization communicates directly with Big Data analytics applications. The following figure shows the proposed meta-model for the visualization layer within Big Data, as well as its relationship to analytics applications:

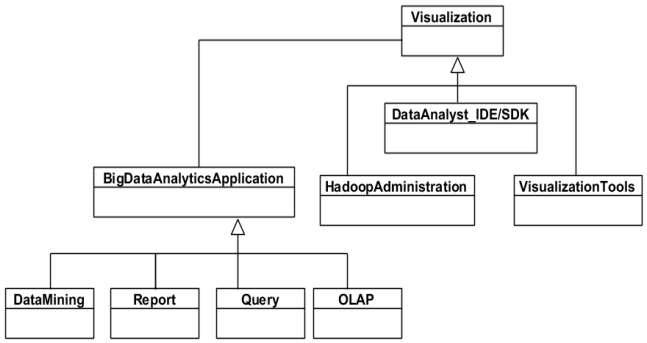

Figure 7: Meta-model of visualization layer.

As seen before, we have managed to create the meta-models for Data Sources, Ingestion, and management layers. In the next step, we shall focus on the creation of models respecting these meta-models. Then we shall define the transformation rules between these meta-models using the transformation language ATL (Atlas Transformation Language) [25,26].

5.3 Meta-model of OLAP

5.3.1 Major Classes and Associations

Nowadays, decision-makers need a synthetic and global vision of the information circulating in their organization to guide and adapt their decision-making. To facilitate this process, they employ decision support systems. These tools allow decision-makers to have a global view of a company's business through quick and interactive access to a set of organized data views to reflect the multi-dimensional nature of business data [27]. In 1993, EF Codd, suggests the use of systems that improve the decision-making process by consulting and analyzing large amounts of data: online analysis systems, "On-Line Analytical Processing (OLAP) [28]. These OLAP systems are intended to provide a quick response to analytical queries of a multidimensional nature. Analytical queries are analyzes that rely on a data centralization tool called: Data Warehouse [29,30].

The following figure shows the meta-model that we proposed for OLAP:

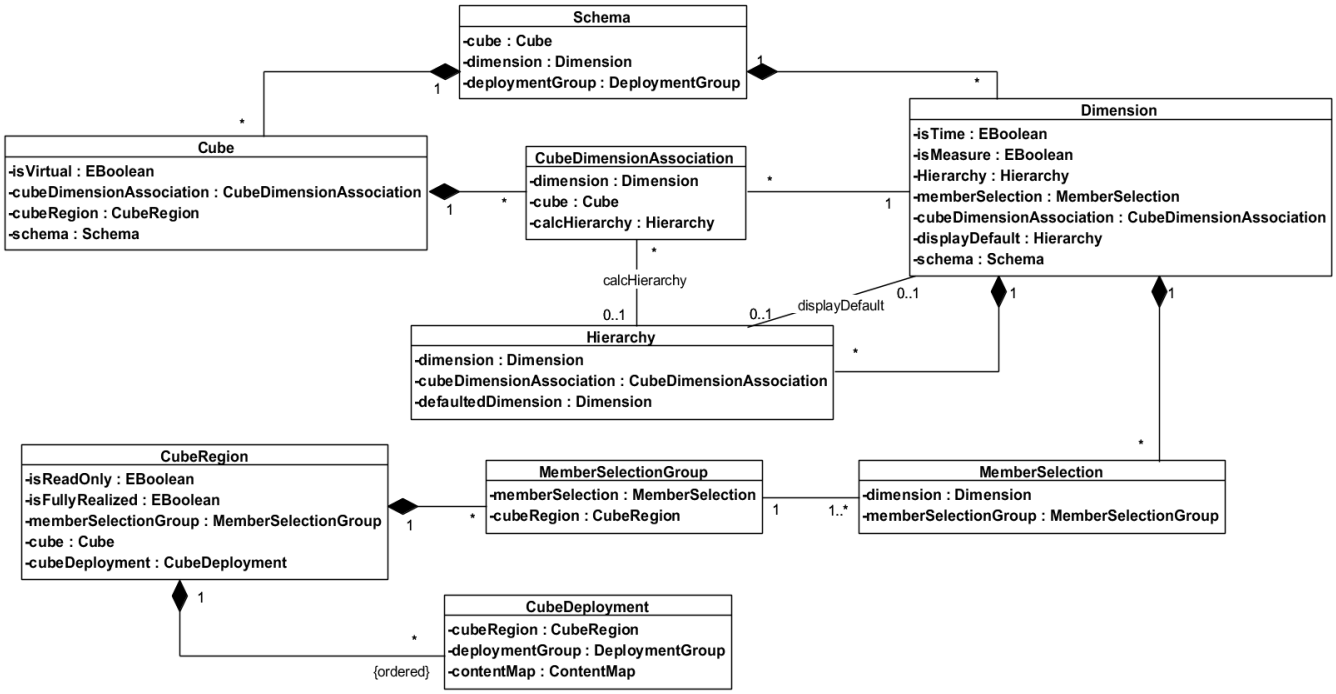

Figure 8: OLAP meta-model: Major Classes and Associations.

The major classes and associations of the OLAP meta-model are shown in Figure 8. Schema is the logical container of all elements comprising an OLAP model. It is the root element of the model hierarchy and marks the entry point for navigating OLAP models. A Schema contains Dimensions and Cubes. A Dimension is an ordinate within a multidimensional structure and consists of a list of unique values (i.e., members) that share a common semantic meaning within the domain being modeled. Each member designates a unique position along its ordinate. A Cube is a collection of analytic values (i.e., measures) that share the same dimensionality. This dimensionality is specified by a set of unique Dimensions from the Schema. Each unique combination of members in the Cartesian product of the Cube's Dimensions identifies precisely one data cell within a multidimensional structure. CubeDimensionAssociation relates a Cube to its defining Dimensions. Features relevant to Cube-Dimension relationships (e.g., calcHierarchy) are exposed by this class.

A Dimension has zero or more Hierarchies. A Hierarchy is an organizational structure that describes a traversal pattern through a Dimension, based on parent/child relationships between members of a Dimension. Hierarchies are used to

define both navigational and consolidation/computational paths through the Dimension (i.e., a value associated with a child member is aggregated by one or more parents). For example, a Time Dimension with a base periodicity of days might have a Hierarchy specifying the consolidation of days into weeks, weeks into months, months into quarters, and quarters into years. A specific Hierarchy may be designated as the default Hierarchy for display purposes (e.g., a user interface that displays the Dimension as a hierarchical tree of members). CubeDimensionAssociation can also identify a particular Hierarchy as the default Hierarchy for consolidation calculations performed on the Cube. MemberSelection models mechanisms capable of partitioning a Dimension's collection of members. For example, consider a Geography Dimension with members representing cities, states, and regions. An OLAP client interested specifically in cities might define an instance of MemberSelection that extracts the city members. CubeRegion models a sub-unit of a Cube that is of the same dimensionality as the Cube itself. Each "dimension" of a CubeRegion is represented by a MemberSelection of the corresponding Dimension of the Cube. Each MemberSelection may define some subset of its Dimension's members. CubeRegions are used to implement Cubes. A Cube may be realized by a set of CubeRegions that map portions of the logical Cube to physical data sources. The MemberSelections defining CubeRegions can also be grouped together via MemberSelectionGroups, enabling the definition of CubeRegions with specific semantics. For example, one can specify a CubeRegion containing only the "input level" data cells of a Cube. A CubeRegion may own any number of CubeDeployments. CubeDeployment is a metaclass that represents an implementation strategy for a multidimensional structure. The ordering of the CubeDeployment classes may optionally be given some implementation-specific meaning (e.g., desired order of selection of several possible deployment strategies, based on optimization considerations).

5.3.2 Dimension and Hierarchy

Figure 9 shows Dimension and Hierarchy, along with several other classes that model hierarchical structuring and deployment mappings:

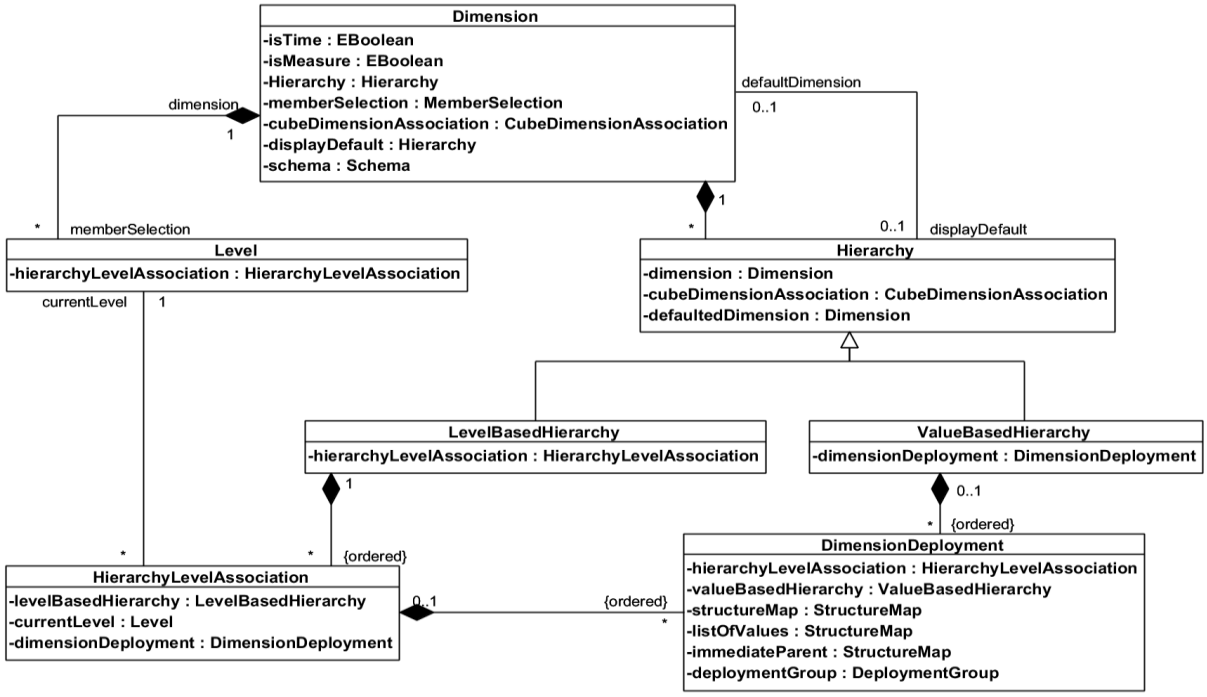

Figure 9: OLAP meta-model: Dimension and Hierarchy.

5.3.2.1 Dimension

The OLAP meta-model defines two special types of Dimension: Time and Measure. A Time Dimension provides a means of representing time-series data within a multidimensional structure. The members of a Time Dimension usually define some base periodicity (e.g., days of the week). The implementation of a Time Dimension might provide support for advanced "time-intelligent" functionality, such as the ability to automatically convert between different periodicities and calendars. The members of a Measure Dimension describe the meaning of the analytic values stored in each data cell of a multidimensional structure.

5.3.2.2 Hierarchy

The OLAP metamodel specifies two subclasses of Hierarchy: LevelBasedHierarchy and ValueBasedHierarchy.

- **LevelBasedHierarchy**

LevelBasedHierarchy describes hierarchical relationships between specific levels of a Dimension. LevelBasedHierarchy

is used to model both "pure level" hierarchies (e.g., dimension-level tables) and "mixed" hierarchies (i.e., levels plus linked nodes). Dimensional levels are modeled by the Level class, a subclass of MemberSelection that partitions a Dimension's members into disjoint subsets, each representing a distinct level. LevelBasedHierarchy contains an ordered collection of HierarchyLevelAssocations that defines the natural hierarchy of the Dimension. The ordering defines the hierarchical structure in top-down fashion (i.e., the "first" HierarchyLevelAssociation in the ordered collection represents the upper-most level of the dimensional hierarchy). A HierarchyLevelAssociation may own any number of DimensionDeployments. DimensionDeployment is a metaclass that represents an implementation strategy for hierarchical Dimensions. The ordering of the DimensionDeployment classes may optionally be given an implementation-specific meaning (e.g., desired order of selection of several possible deployment strategies, based on optimization considerations).

- **ValueBasedHierarchy**

A ValueBasedHierarchy defines a hierarchical ordering of members in which the concept of level has little or no significance. Instead, the topological structure of the hierarchy conveys meaning. ValueBasedHierarchies are often used to model situations where members are classified or ranked according to their distance from a common root member (e.g., an organizational chart of a corporation). In this case, each member of the hierarchy has some specific "metric" or "value" associated it with it. ValueBasedHierarchy can be used to model pure "linked node" hierarchies (e.g., asymmetric hierarchical graphs or parent-child tables). As with LevelBasedHierarchy, ValueBasedHierarchy also has an ordered collection of DimensionDeployments, where the ordering semantics are left to implementations to define.

5.4 Generic meta-model for visualization layer

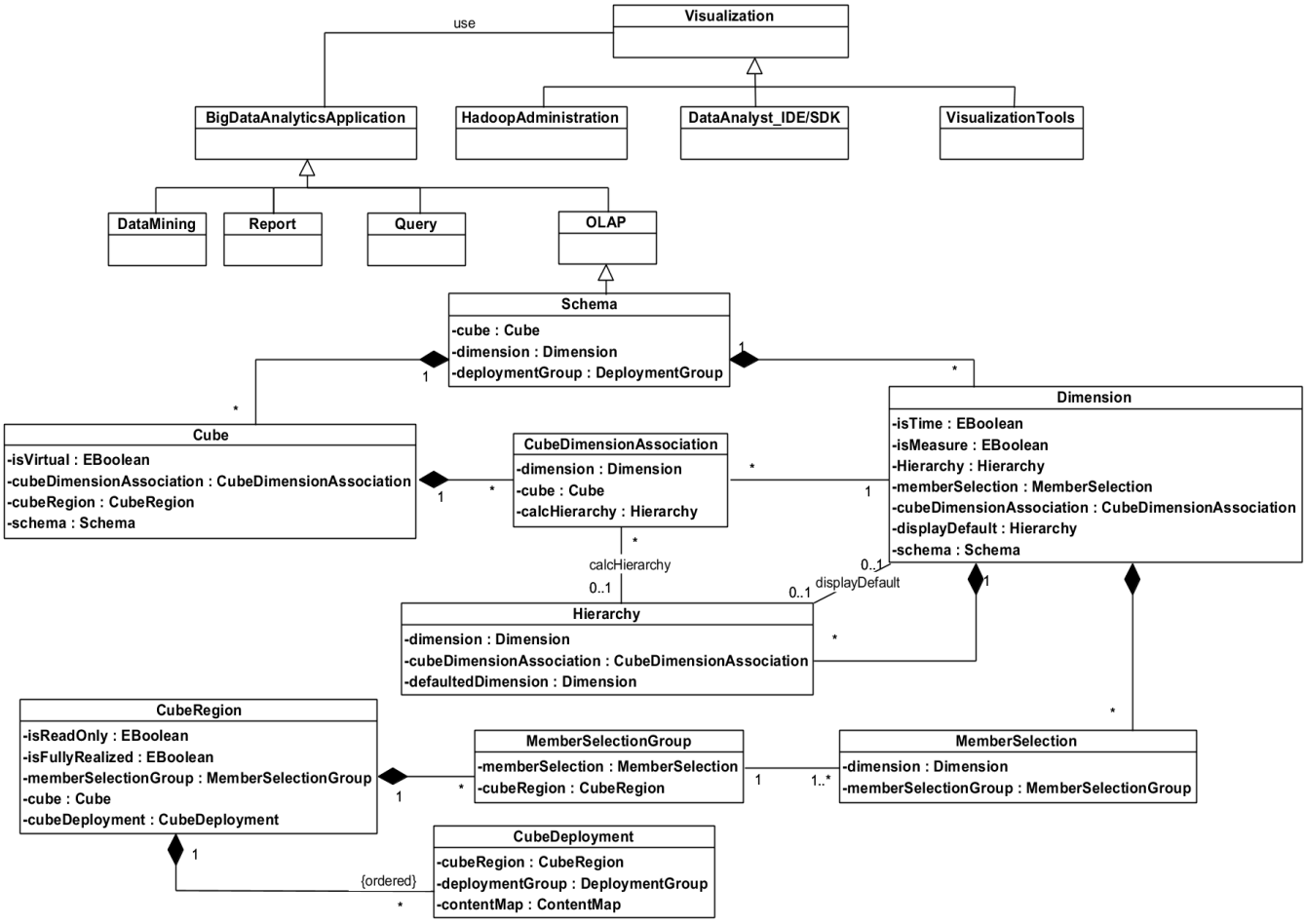

Figure 10: Generic meta-model for visualization layer.

These meta-models are platform independent according to Model Driven Architecture pattern [31,33], which describes visualization layer and its relation with the Big Data analytics application independently from any specific platform.

6 FUTURE WORK

Visualization today has ever-expanding applications in science, engineering, education, interactive multimedia, medicine, etc. Model-based visualization components are commonly more sophisticated. Our approach has been used to generate meta-model for visualization layer using techniques related to Model-driven Engineering, we have also proposed meta-model for OLAP like a Big Data analytics application that helps decision makers make a decision. We are currently continuing our work by creating a complete library of visualization meta-models and model-driven visualization components. The meta-models for visualization layer and the other layers, which we have proposed before in our previous works, will be used for creating a universal meta-modeling of a Big Data system. Model-based visualization components are also being developed for integration within Eclipse using the EMF framework.

7 CONCLUSION AND PERSPECTIVES

In short, the exponential growth of data produced every day by consumers and businesses makes it necessary to develop new tools to make them easier to read. In fact, big data management needs new methods and powerful technologies to deal with these huge data sets. More precisely, the visualization layers remains an efficient mean to exploit and deal with this large amount of data. The trend that we have seen emerge for some years now: data visualization (or DataViz). At this stage, it is worth mentioning that in this article we applied the modeling techniques (Model Driven Engineering) by proposing meta-models for the Big Data layers. Our main goal is to create standardized concepts at the Big Data level. In the MDA (Model Driven Architecture) approach, the generic meta-model proposed for Big Data layers can be used as an independent cross-platform Domain Specific Language [32]. Thus, further work needs to be carried out in order to create a powerful system capable of ensuring a more accurate and reliable result for big data management.

REFERENCES

1. Allae Erraissi, Abdessamad Belangour, Abderrahim Tragha. "**A Big Data Hadoop Building Blocks Comparative Study**." International Journal of Computer Trends and Technology. Accessed June 18, 2017. https://doi.org/10.14445/22312803/IJCTT-V48P109
2. Allae Erraissi, Abdessamad Belangour, and Abderrahim Tragha, "**Digging into Hadoop-based Big Data Architectures,**" Int. J. Comput. Sci. Issues IJCSI, vol. 14, no. 6, pp. 52–59, Nov. 2017. https://doi.org/10.20943/01201706.5259
3. Erraissi, A., & Belangour, A. (2018). **Data sources and ingestion big data layers: meta-modeling of key concepts and features.** International Journal of Engineering & Technology, 7(4), 3607–3612. https://doi.org/10.14419/ijet.v7i4.21742
4. Erraissi, Allae, and Abdessamad Belangour. « **Hadoop Storage Big Data Layer: Meta-Modeling of Key Concepts and Features** ». International Journal of Advanced Trends in Computer Science and Engineering 8, nº 3 (2019): 646-53. https://doi.org/10.30534/ijatcse/2019/49832019
5. A., Erraissi A., Belangour A. (2019) **Capturing Hadoop Storage Big Data Layer Meta-Concepts**. In: Ezziyyani M. (eds) Advanced Intelligent Systems for Sustainable Development (AI2SD'2018). AI2SD 2018. Advances in Intelligent Systems and Computing, vol 915. Springer, Cham
6. A. Erraissi and A. Belangour, "**Meta-modeling of Zookeeper and MapReduce processing**," 2018 International Conference on Electronics, Control, Optimization and Computer Science (ICECOCS), Kenitra, Morocco, 2018, pp. 1-5. doi: 10.1109/ICECOCS.2018.8610630.
7. Royer, Jean-Claude, and Hugo Arboleda. **Model-Driven and Software Product Line Engineering**. 1st Edition. London, UK : Hoboken, NJ, USA: Wiley-ISTE, 2012.
8. Read, W., Report, T., & Takeaways, K. (2016). The Forrester WaveTM: Big Data Hadoop Distributions, Q1 2016.
9. R. D. Schneider, "**HADOOP BUYER'S GUIDE**," 2014.
10. V. Starostenkov, R. Senior, and D. Developer, "**Hadoop Distributions** ".
11. Kleppe, A.G., Warmer, J., Warmer, J.B. and Bast, W., 2003. **MDA explained: the model driven architecture: practice and promise**. Addison-Wesley Professional.
12. Schmidt, Douglas C. "**Model-driven engineering.**" COMPUTER-IEEE COMPUTER SOCIETY- 39.2 (2006): 25. https://doi.org/10.1109/MC.2006.58
13. Booch, Grady. "**UML in action**." Communications of the ACM42.10 (1999): 26-26. https://doi.org/10.1145/317665.317672
14. ISO/IEC/JTC 1/SC 32. ISO/IEC 19502:2005, Information Technology **- Meta Object Facility**. Multiple. Distributed through American National Standards Institute, 2007.
15. Favre, Jean-Marie, Jacky Estublier, and Mireille Blay-Fornarino. **L'ingénierie dirigée par les modèles: au-delà du MDA.** Hermes Science, 2006.
16. W. PLAYFAIR. **A Letter on Our Agricultural Distresses, Their Causes and Remedies: Accompanied with Tables and Copper-plate Charts, Shewing and Comparing the Prices of Wheat, Bread and Labour from 1565 to 1821**... Sams, 1822.
17. E. R. TUFTE. « **The visual display of quantitative information** ». Edward R. Tufte. Cheshire, Conn.: Graphics Press, c1983. (1983).
18. Brody, Howard, et al. "**Map-making and myth-making in Broad Street: the London cholera epidemic, 1854**." The Lancet356.9223 (2000): 64-68. https://doi.org/10.1016/S0140-6736(00)02442-9
19. R. B. HABER and D. A. MCNABB. « **Visualization idioms: A conceptual model for scientific visualization**

systems ». Visualization in scientific computing 74 (1990), p. 93.
20. S. DOS SANTOS and K. BRODLIE. « **Gaining understanding of multivariate and multidimensional data through visualization** ». Computers & Graphics 28.3 (2004), p. 311–325. https://doi.org/10.1016/j.cag.2004.03.013
21. B. SHNEIDERMAN. « **The eyes have it: A task by data type taxonomy for information visualizations** ». Visual Languages, 1996. Proceedings. IEEE Symposium on. IEEE. 1996, p. 336–343.
22. Healy, Kieran. **Data Visualization: A Practical Introduction.** 1 edition. Princeton, NJ: Princeton University Press, 2018.
23. Steffine, Gregory P. Hyper: **Changing the Way You Think about, Plan, and Execute Business Intelligence for Real Results, Real Fast! Aliquippa**, Penn.: Sanderson Press, LLC, 2015.
24. N. Sawant and H. (Software engineer) Shah, **Big data application architecture Q & A a problem-solution approach**. Apress, 2013. https://doi.org/10.1007/978-1-4302-6293-0
25. "**ATL**: Atlas Transformation Language Specification of the ATL Virtual Machine."
26. "**ATL**: Atlas Transformation Language ATL Starter's Guide," 2005.
27. Colliat, G. (1996). **OLAP, relational, and multidimensional database systems**. SIGMOD Record 25(3), 64–69. https://doi.org/10.1145/234889.234901
28. **Providing OLAP to User-Analysts:** An IT Mandate by E F Codd, S B Codd and C T Salley, ComputerWorld, 26 July 1993.
29. Kimball, Ralph. **The Data Warehouse Toolkit: Practical Techniques for Building Dimensional Data Warehouses**. New York, NY, USA: John Wiley & Sons, Inc., 1996.
30. Inmon, W. H. **Building the Data Warehouse** (2Nd Ed.). New York, NY, USA: John Wiley & Sons, Inc., 1996.
31. Mouad Banane, and Abdessamad Belangour. « **RDFMongo: A MongoDB Distributed and Scalable RDF Management System Based on Meta-Model** ». International Journal of Advanced Trends in Computer Science and Engineering 8, no 3 (25 juin 2019): 734-41. https://doi.org/10.30534/ijatcse/2019/62832019.
32. Pastor, Oscar, and Juan Carlos Molina. **Model-Driven Architecture in Practice: A Software Production Environment Based on Conceptual Modeling**. 2007 edition. Berlin ; New York: Springer, 2007.
33. Banane, M., & Belangour, A. (2019). **New Approach based on Model Driven Engineering for Processing Complex SPARQL Queries on Hive**. *International Journal of Advanced Computer Science and Applications (IJACSA)*, 10(4).
34. Erraissi, Allae, and Abdessamad Belangour. « **Meta-Modeling of Big Data Management Layer** ». *International Journal of Emerging Trends in Engineering Research* 7, no 7 (15 July 2019): 36-43. https://doi.org/10.30534/ijeter/2019/01772019.