Data Warehousing in Big Data: From Multidimensional to Tabular Data Models

Maribel Yasmina Santos
ALGORITMI Research Centre
University of Minho
Guimarães, Portugal
+351 253 510 308
maribel@dsi.uminho.pt

Carlos Costa
ALGORITMI Research Centre
University of Minho
Guimarães, Portugal
+351 253 510 319
id6011@alunos.uminho.pt

ABSTRACT
Data warehouses are central pieces in business intelligence and analytics as these repositories ensure proper data storage and querying, being supported by data models that allow the analysis of data by different perspectives. Those perspectives support users and organizations in the decision-making process. In Big Data environments, Hive is used as a distributed storage mechanism that provides data warehousing capabilities. Its data schemas are defined attending to the analytical requirements specified by the users. In this work, multidimensional data models are used as the source of those requirements, allowing the automatic transformation of a multidimensional schema into a tabular schema suited to be implemented in Hive. To achieve this objective, a set of rules is proposed and tested in a demonstration case, showing the applicability and usefulness of the proposed approach.

Categories and Subject Descriptors
H.2.8 [**Information Systems**]: Data management systems – Database design and models.

General Terms
Data Models, Management.

Keywords
Analytical Data Model; Big Data; Data Warehousing; Hive.

1. INTRODUCTION
Information systems started to consider in their technological infrastructures repositories able to enhance business intelligence and analytics in Big Data environments. In those contexts, the volume, variety or velocity of data can justify the adoption of NoSQL databases as storage repositories, supporting operational or analytical applications [1].

Considering analytical applications, the multidimensional data model is extensively used to model the data requirements that will guide the implementation of a data warehouse using traditional environments like a relational or a multidimensional database [2].

When the volume of data, for instance, justifies the implementation of a data warehouse in a big data context, a data model also needs to be identified for defining the structure of the tables that need to be created in Hive [3], [4], the data warehousing repository available in Hadoop [5], which facilitates managing and querying large datasets that are in distributed storage. However, and due to the characteristics of Hive, whose data schema must be defined considering the queries that need to be answered, few methodological guidelines are available for guiding this process. Usually, users need to specify the analytical requirements that will afterwards guide the definition of the data schema.

As multidimensional modeling is extensively used in traditional business intelligence environments, and many data warehouses are already available to support organizational decision-making processes, this paper presents a set of rules that automatically transform a multidimensional data model into tables suited to be implemented in Hive. Following this, the obtained tables simulate the different analytical perspectives that an on-line analytical processing system offers, providing different analytical perspectives and different aggregations on data. As advantages, there is no need to question users about the analytical requirements, as those were previously considered in the multidimensional data model. Also, the process is guided through a well-defined set of steps that ensure that no data requirements are left behind.

The work here proposed is aligned with the current research trends, being one of these related with the integration of multidimensional data sources into the Hadoop lifecycle, a stimulating research challenge for the next generation of data warehousing and on-line analytical processing research [6].

This paper is organized as follows. Section 2 presents related work, pointing the main limitations identified in the state-of-the-art. Section 3 describes the definitions and rules proposed for the automatic transformation of a multidimensional data model in a Hive data schema. Section 4 uses a demonstration case to show the applicability and usefulness of the proposed approach. Section 5 concludes with some remarks and guidelines for future work.

2. RELATED WORK
Data models are central pieces in business intelligence and analytics as these repositories ensure proper data storage and querying, being supported by data models that allow the analysis of data by different perspectives. This means that information systems design and development, which support business intelligence and analytics contexts, need to pay particular attention to those data models.

In a traditional organizational environment, the multidimensional data model is very popular [2], integrating the fact tables with the measures or indicators to be analyzed, and the several dimensions,

C3S2E'16, July 20–22 2016, Porto, Portugal.

DOI: http://dx.doi.org/10.1145/2948992.2949024

or perspectives, on which those measures will be explored. In the data-modeling task, specific rules are available for identifying the business processes to be considered, the granularity to be adopted, the dimensions to be included, and the measures to be analyzed [2]. However, when we move to an emergent context like the one represented by Big Data, and due to the characteristics of the NoSQL databases here used, the data modeling task is seen in another perspective, as those databases are schema-free. This means that the data model can change in runtime, due to the data storage or analytical needs [7]. The models, in this context, are designed considering the queries that need to be answered. Although schema-free, these databases also need data models that ensure the proper storage of the data.

In the case of Hive, a database is constituted by a set of tables [4], [8], organized according to the queries that need to be answered, being data schemas highly denormalized, as happens in a multidimensional data model. In order to facilitate the transformation of a multidimensional data model to a Big Data context, the work of [9] recognizes that the design of big data warehouses differs considerable from traditional data warehouses, as their schema should be based on novel logical models that allow more flexibility than the relational model does. Therefore, their design methodology requires new principles, such as automation and agile techniques, which allow both fast realization and reaction to changes in business requirements. For that, the authors propose a design methodology for the representation of a multidimensional schema at the logical level, based on a particular type of NoSQL repository, the key-value model. The proposed methodology is considered hybrid as it integrates a data-driven approach design, using data repositories as the main source of information, and a requirements-driven approach design, using information from the decision makers. In this case, the needs of decision makers are investigated and are represented as business goals considering the actors of the system.

The works of [10], [11] and [7] are dedicated to implementing multidimensional data warehouses in NoSQL databases. While [10] and [11] propose a set of rules for mapping start schemas into column-oriented and document-oriented data models, the work of [7] proposes three approaches for implementing big data warehouses in column-oriented data models. [10] and [11] consider a star schema as a logical data model in the multidimensional context and transform it into a table in a columnar-oriented model or a document-oriented model, where all attributes from the dimensions and fact tables are stored together grouped attending to the specific provided rules. In [7], the three approaches are the normalized logical approach, denormalized logical approach and denormalized logical approach by using column family, and were compared for evaluating the performance of a data warehouse in a distributed environment.

In the mentioned works, the authors focused on the conversion of a star schema into a columnar table or a document-oriented model, not addressing more complex scenarios like a constellation of star schemas, with the need of integration of data from different fact tables for analytical purposes.

As we are going to see, the work here proposed considers a constellation schema and proposes the automatic integration of different fact tables, providing useful insights on the data under analysis. Moreover, the work here proposed does not require any additional information, like the specification of the queries to be answered, and provides guidelines on how to optimize the Hive data model.

3. DATA WAREHOUSING IN BIG DATA

3.1 Overall Overview

In a Big Data context, there is the need to add structure to the data when analytical tasks are required, which is the reason why specific guidelines about how to structure a Hive data model are here proposed for the transformation of multidimensional data models, usually used to model a data warehouse repository in a traditional Business Intelligence environment, to a tabular data model that enhances the analysis of data with Hive, allowing the concretization of a data warehouse in Big Data. In the proposed process, it is possible to derive a set of different tables providing different analytical perspectives on the data. Traditionally, a tabular data model is specified taking into consideration the set of queries that need to be answered. This requires the identification of the users' requirements and the modeling of the data repositories. With the approach here proposed, all analytical capabilities are foreseen in an approach that imitates how the cubes of an on-line analytical processing system work, proposing tables that include the different combinations of the dimensions and indicators, providing those different perspectives in data analysis tasks.

When the volume of data constrains the use of a traditional Business Intelligence environment, the work here presented ensures a well-guided process in which the analytical needs, expressed in the multidimensional data model, are addressed.

In the context of the work here presented, a multidimensional data model is considered an Analytical Data Model (*ADM*), which can be implemented in a traditional Business Intelligence environment (ADM_{BI}), or in a Big Data environment (ADM_{BD}), in which Hive or a columnar NoSQL database can be used in its implementation. This work is dedicated to the transition to a Hive data model, although specific hints will also be given if the context also requires a columnar database in the implementation.

Hive is a data storage mechanism in Big Data environments used as a data warehousing software that facilitates querying and managing large datasets that are in distributed storage [3], [4]. Having the data in a distributed file system, Hive adds structure to the data and allows querying through the use of the HiveQL (Hive Query Language), a SQL-like language [8]. In Hive, three elements exist to organize the available data: Tables, Partitions and Buckets.

Tables are common structures with columns and rows that are stored in a directory of HDFS (Hadoop Distributed File System) [5]. Partitions are defined over the tables to make horizontal slices of the data and speed up query processing. Partitions are stored in a sub-directory of a table's directory, being possible to have several partitions associated to a table. The partitions are used to prune the data that is searched in a specific query, having a strong impact on the time that a query takes to process. Buckets can be defined associated to tables or to partitions, being stored in a file within the partition's or table's directory, depending if the table is partitioned or not, and are used as a technique to cluster large datasets of data to optimize query performance. When a table is created, the user can specify the number of buckets needed and the column that will be used to bucket the data. Again, this information is used to prune the data in case the user runs the query on a sample data [8].

Let us suppose that the table *Customer* will be created in Hive and that the attribute *name* is used as a partition. In this case, associated to the directory of the table, there will exist as many sub-directories as customers, splitting the data. When a WHERE

condition is used in a query, to filter a specific customer, only the sub-directory associated to that condition will be searched. If we take as an example *Customer* and Location, using also the name of the customer as partition, but using the city of the location as a bucketing attribute then, per customer name, groups of cities will be created taking into consideration the number of defined buckets.

Having a Hive data model for supporting an efficient analytical context, and besides the definition of the structure of the tables, the challenge is how to optimize the Hive data model. Moreover, as joins between tables are possible in Hive, different analytical tables can be joined to provide the answer to a specific analytical question.

While the identification of the Hive data model will be constrained by the multidimensional data model taken as input, considering the relationships between the fact tables and the dimension tables, the definition of partitions and buckets is completely dependent on the available data like, for instance, the number of different customers, locations or products, as data should not be over partitioned. If partitions are relatively small then the cost of searching many directories becomes more expensive than simple scanning all the data. Also, partitions should be similar in size to prevent a single long running. Considering the city attribute mentioned above and supposing that more than 80% of the clients are from 2 or 3 different cities, partitions using this attribute would not be adequate. Bucketing would be preferred as different cities can be clustered in the same bucket. Joining two tables for the same city, for instance, allows Hive to join bucket by bucket, which performs even better if buckets are sorted.

In general, partitioning helps in reducing data used in a WHERE clause, while bucketing helps in organizing data in each partition using multiple files. The same set of data is always written in the same bucket, which helps in the joining of columns. In general, bucketing works well when the attribute has high cardinality and partition when the cardinality is not too high and the available records are well distributed by the different attribute's values. Partitioning can be done on multiple attributes, while bucketing can only be applied to a single one.

3.2 Proposed Transformation

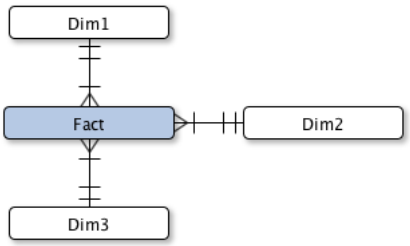
Let us now consider the definition of an ADM_{BI} and propose the definition of an ADM_{BD} based on a tabular data model and define the rules that allow the automatic transformation from one to the other:

Definition 1. An Analytical Data Model in a Business Intelligence context, $ADM_{BI} = (D, F, R, M)$, includes a set of dimension tables, $D = \{D^1, D^2, \dots, D^l\}$, a set of fact tables, $F = \{F^1, F^2, \dots, F^m\}$, a set of relationships, $R = \{R_{F^1}^{D^1}(c_f : c_d), R_{F^1}^{D^2}(c_f : c_d), \dots, R_{F^m}^{D^l}(c_f : c_d)\}$, which associate facts included in a specific fact table, F_1 for example, with a set of dimensions, D_1, D_2, ..., being the cardinality $(c_f : c_d)$ of the relationship of n: 1 between the fact table and the dimension table. An ADM_{BI} also includes a set of measures, $M = \{M^1, M^2, \dots, M^n\}$, used for analyzing the business processes.

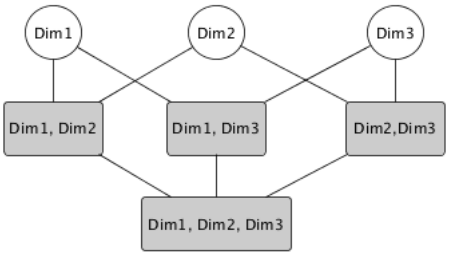
Definition 2. An Analytical Data Model in a Big Data context, $ADM_{BD} = (T, CG)$, includes a set of tables $T = \{T^1, T^2, \dots, T^n\}$, where each table integrates one or more column groups, as $T^i = \{CG^1, \dots, CG^m)\}$. Each column group integrates different columns representing the atomic values to be stored in the Hive tables, $CG^j = \{C_j^1, \dots, C_j^k)\}$.

Considering the characteristics of a tabular data model, like the ones considered by Hive, the following rules can be adopted in an organizational context to automatically transform an ADM_{BI} into an ADM_{BD}, starting by the identification of tables that will be included in Hive.

Rule ADM_{BD}.1: Identification of Tables. The identification of tables considers the dimension tables present in the ADM_{BI}, and computes the Dimensions Lattice to verify the different combinations of dimensions, which is useful for the analytical capabilities of the ADM_{BD}. Considering an abstract example in which a star schema integrates a fact table (*Fact*) and 3 dimensions (Dim_1, Dim_2 and Dim_3) as depicted in Figure 1 a), the Dimensions Lattice, shown in Figure 1 b), integrates the 7 final tables with all the combinations between dimensions plus the original ones. The table that combines all the dimensions of a fact table will contain all the transactions present in the fact table, as the granularity of the data does not suffer any change. All the other tables will include aggregated data, with different levels of aggregation, providing the different analytical perspectives on data.



a) ADM_{BI} example data model



b) Dimensions Lattice

Figure 1. Example of a Dimensions Lattice of a star schema

Following the approach proposed in this rule for the identification of the tables, it is possible to calculate the maximum number of tables that can be obtained. One of the levels of the lattice, the more granular one, contains each one of the dimensions as an independent table. Level *n* (where *n* is the number of dimensions) will have all the dimensions integrated in one. Level *n-1* will have combinations of *n-1* dimensions, level *n-2* will have combinations of *n-2* dimensions, and so on. When we are dealing with a star schema, only one fact table, the total number of combinations (*NumCombinations*) is obtained by Equation 1, where *n* represents the number of different dimensions, so the number of levels. With 3 dimensions, 7 combinations, representing the tables, are obtained.

$$NumCombinations = \sum_{i=1}^{n} \frac{n!}{i! \times (n-i)!} \quad (1)$$

When a constellation schema is considered, meaning that more than one fact table is available, the number of tables must consider the combination of dimensions and, also, the fact tables to which

they are related, as fact tables can share different dimensions. Considering now an example with a constellation schema as provided in Figure 2, in which two fact tables ($Fact_1$ and $Fact_2$) share two dimension tables (Dim_1 and Dim_2), the total number of combinations is 15 (following Equation 1), but the total number of tables to be implemented is lower, as there are fact tables that do not share dimensions.

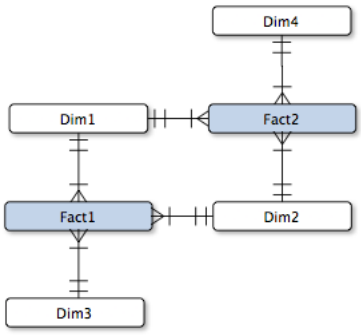


Figure 2. ADM_{BI} example data model (two fact tables)

Considering the example in Figure 2, in which Dim_3 and Dim_4 are not shared by the two fact tables, all the combinations that include these two dimensions must be excluded from the final set of tables, as those do not make sense in the logical data model under analysis. These combinations are identified in the red dashed lines in Figure 3.

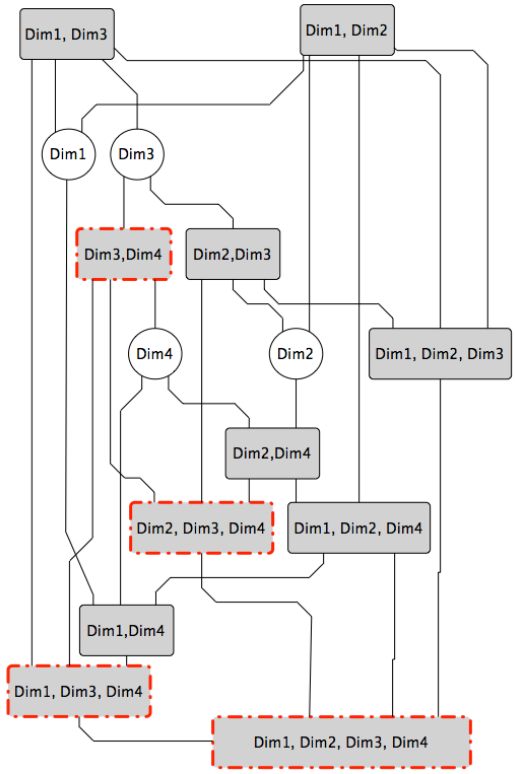


Figure 3. Dimensions Lattice with combinations that must be excluded

After the identification of the combinations that do not share dimensions tables, and their correspondent removal from the set of tables to be considered, 11 different tables need to be considered, as shown in Figure 4.

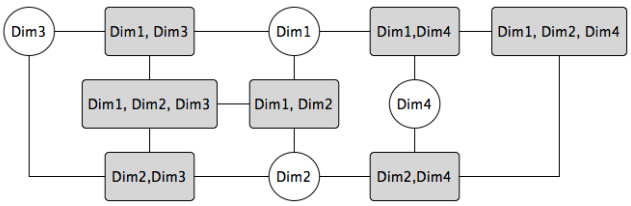


Figure 4. Example of a Dimensions Lattice for a constellation schema

Rule ADM_{BD}.2: Identification of Column Groups. The identification of column groups of an ADM_{BD} follows a two-step approach:

Rule ADM_{BD}.2.1: Identification of Descriptive Column Groups. All the dimensions present in the ADM_{BI} will give origin to descriptive column groups, one column group per dimension, providing the descriptive attributes that add semantics to the analytical ones and the different analytical perspectives. The attributes of a descriptive column group are constituted by the set of non-key attributes (excluding primary or foreign keys) present in the corresponding dimensions of the ADM_{BI}.

Rule ADM_{BD}.2.2: Identification of Analytical Column Groups. All the fact tables present in the ADM_{BI} will give origin to analytical column groups, one column group per fact table, providing the analytical attributes that will be analyzed considering the related column groups identified in the previous rule. The attributes of an analytical column group are constituted by the set of non-key attributes (excluding primary or foreign keys) present in the corresponding fact tables of the ADM_{BI}. Those attributes correspond to the measures or business indicators under analysis.

Rule ADM_{BD}.3: Association of Column Groups to Tables. A specific table integrates a set of descriptive and analytical column groups, which may vary depending on the table. For identifying the several descriptive and analytical column groups, the following rules must be followed:

Rule ADM_{BD}.3.1: Descriptive Column Groups. Attending to the dimensions lattice identified in Rule ADM_{BD}.1, each table will integrate the column groups of the corresponding dimensions. For example, the table derived from Dim_1 will include the column group obtained from Dim_1, while the table derived from Dim_1 and Dim_2 will include the column groups obtained from Dim_1 and Dim_2, and so on.

Rule ADM_{BD}.3.2: Analytical Column Groups. Considering the relationships existing between the fact tables and the dimension tables of the ADM_{BI}, each table in the ADM_{BD} will include as analytical column groups the ones inherited from the relationships expressed in the ADM_{BI}. For example, in Figure 2, the table derived from Dim_1 will include the analytical column groups derived from $Fact_1$ and from $Fact_2$, while the table derived from Dim_2 and Dim_4 will only include the analytical column group derived from $Fact_2$. As already mentioned, all these tables – with exception of the ones that combine all the dimensions of a fact table (maintaining the same granularity of the fact table) – need an aggregation function for the attributes present in the analytical column groups, as different summaries of the measures need to be calculated, attending to the descriptive column groups present in a specific table. By default, it is assumed that the aggregation function is SUM, although another one can be adopted as, for instance, AVG, COUNT, MIN or MAX.

Being column groups associated to tables, the work proposed in ([12]) can be used to implement the model obtained so far in a columnar database, as long as a specific key is generated for each table, as also addressed in ([12]) for HBase.

Rule ADM_{BD}.4. Columns of Hive Tables. The identification of columns for each table identified by Rule ADM_{BD}.1, and that will be implemented in Hive, follows a two-step approach.

Rule ADM_{Hive}.4.1. Descriptive Columns. The attributes that integrate the several descriptive column groups of the ADM_{BD} give origin to descriptive columns of a table in Hive.

Rule ADM_{Hive}.4.2. Analytical Columns. The attributes that integrate the several analytical column groups of the ADM_{BD} give origin to analytical columns of a table in Hive.

Rule ADM_{BD}.5. Identification of Partitions and Buckets. The identification of partitions and buckets will consider the balance that must exist between the cardinality of the attributes and their distribution. For each column group that gave origin to descriptive attributes by Rule ADM_{Hive}.2.1, the cardinality and distribution of the attributes that define the granularity of the column group must be analyzed. As a table in an ADM_{BD} can integrate one or more descriptive columns, it is necessary to analyze the cardinality and distribution of the several columns, to support the identification of the appropriate ones for partitioning and for bucketing. For that, and as in a multidimensional data model, the attribute that defines the granularity of the dimension is the one that provides the highest detail to the table, this rule suggests the analysis of the cardinality and distribution of those attributes. Considering different descriptive columns, and their characteristics in terms of cardinality and distribution, the column with lowest cardinality and best distribution must be used as the partitioning attribute, while a column with higher cardinality could be used as the bucketing attribute. These suggestions aim to balance the data and enhance query performance. Nevertheless, the suggested model can be fine-tuned by the user or database modeler in order to add other attributes to the partition, for instance, or to change the choices previously suggested based on the cardinality and distribution of the attributes.

4. DEMONSTRATION CASE

In order to verify the applicability of the proposed rules and their relevance in an analytical scenario in Big Data contexts, let us consider a demonstration case that integrates two fact tables, *Orders* and *OrderProducts*, as seen in Figure 5, which represent the orders made in a given organization, and the products purchased in each order. The *Orders* fact table is a factless fact table that includes an event-tracking counter (*Event Order*), used to provide summaries of the orders by the different considered dimensions, *Calendar*, *Customer* and *Location*. The *OrderProducts* fact table also includes an event-tracking counter (*Event Product*), the ordered quantity (*Quantity*) and the ordered value (*Value*) to provided summaries of the ordered products, by the different considered dimensions, *Calendar*, *Customer* and *Product*. In this model, and for the keys, the following nomenclature is used: PK (Primary Key), FK (Foreign Key) and SK (Surrogate Key).

Taking into consideration the rules expressed for the transformation of an ADM_{BI} into an ADM_{BD}, we start by the identification of the dimensions lattice that identifies the tables of the ADM_{BD} model (Rule ADM_{BD}.1), in this case 11 resulting tables (Figure 6). Although in a specific implementation context different nomenclatures can be adopted for naming the resulting tables, in this work the integration of a small abbreviation of each dimension is used to combine the tables' name, and the prefix *agg* is added to those tables that integrate pre-aggregations of the data: $aggCal_T$, $aggCus_T$, $aggLoc_T$, $aggProd_T$, $aggCusProd_T$, $aggCalProd_T$, $aggCalCus_T$, $aggCusLoc_T$, $aggCalLoc_T$, $CalCusProd_T$ and $CalCusLoc_T$.

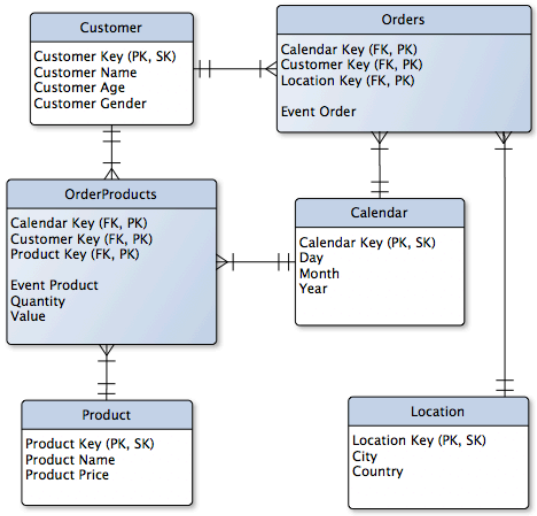


Figure 5. Example of an ADM_{BI} for Orders

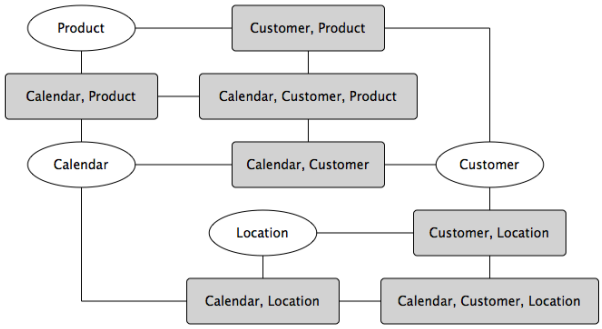


Figure 6. Dimensions Lattice for the ADM_{BI} Orders

Proceeding with Rule ADM_{BD}.2 for the identification of the different column groups, Rule ADM_{BD}.2.1 allows the identification of four descriptive column groups, one for each dimension in the ADM_{BI}: $Calendar_{CG}$, $Customer_{CG}$, $Location_{CG}$ and $Product_{CG}$. For the analytical column groups, Rule ADM_{BD}.2.2, two column groups are identified: $Orders_{CG}$ and $OrderProducts_{CG}$. Rule ADM_{BD}.3 allows the association of the column groups to the corresponding tables. For the descriptive column groups (Rule ADM_{BD}.3.1), it is necessary to follow the dimensions lattice previously identified (Figure 6).

Let us take two different tables as example. The $aggCustomer_T$ was derived from the *Customer* dimension, meaning that it will only integrate the column group derived from it, $Customer_{CG}$. For $CalCusProd_T$, as it was derived from three dimensions, *Calendar*, *Customer* and *Product*, it will integrate the three corresponding column groups: $Calendar_{CG}$, $Customer_{CG}$ and $Product_{CG}$. For the analytical column groups (Rule ADM_{BD}.3.2), it is necessary to follow the relationships between the dimensions and the fact tables present in the ADM_{BI}. Considering again the same two tables as example, $aggCus_T$ was derived from the *Customer* dimension, which is related with the *Orders* and *OrderProducts* fact tables (Figure 5), meaning that this table will integrate the

column groups of these two fact tables: $Orders_{CG}$ and $OrderProducts_{CG}$. Once that $CalCusProd_T$, considering the dimensions that gave origin to it, is only related with the *OrderProduct* fact table, this table will only include the $OrderProducts_{CG}$.

For the analytical column groups, it is also necessary the definition of the aggregation functions that will be used to summarize the data of the tables in the ADM_{BD}. All tables, excluding the two that do not need an aggregation function ($CalCusLoc_T$ and $CalCusProd_T$) use the SUM function. Before proceeding with the next rule, with the structure of Hive tables, Figure 7 shows the tables obtained so far and their corresponding column groups.

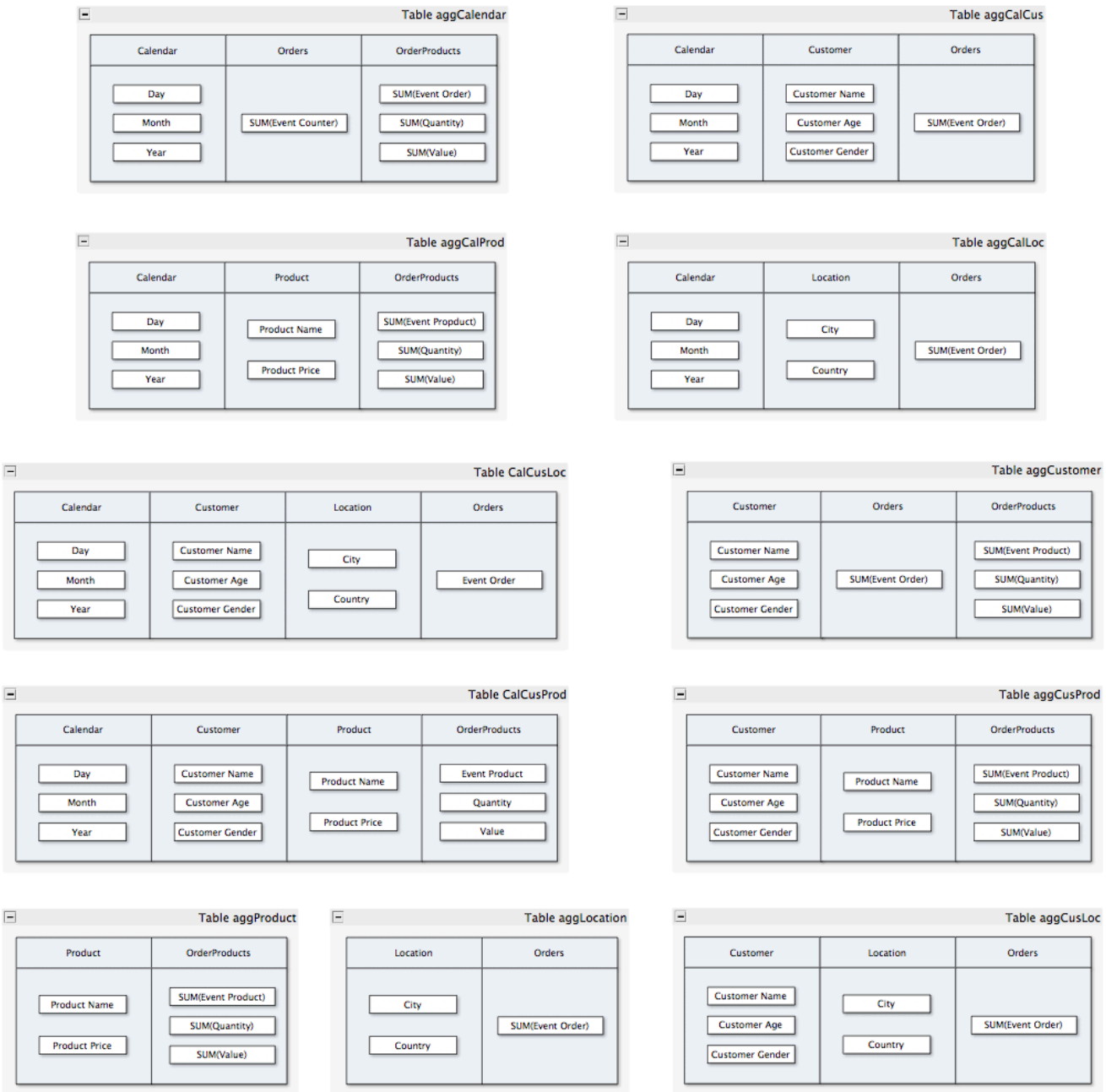


Figure 7. Resulting set of tables and column groups for Orders

Analyzing Figure 7, it is possible to see that the obtained tables present different aggregates of data. Taking as an example $aggCustomer_T$, this table will integrate one row per customer with the descriptive attributes of that customer, as name, age and gender, as well as the total number of orders that particular customer did (*SUM(Event Order)* in the $Orders_{CG}$), the number of times products were ordered (*SUM(Event Product)* in the $OrderProducts_{CG}$), the total quantity of ordered products

(*SUM(Quantity)* in the $OrderProducts_{CG}$) and the total amount spent in the orders (*SUM(Value)* in the $OrderProducts_{CG}$). Tables $CalCusProd_T$ and $CalCusLoc_T$, containing all detailed records available in the fact tables of the ADM_{BI}, can be used (if needed) with the HiveQL to calculate different aggregates on data, summarizing the data in a different way.

Having identified the tables and their column groups, Rule ADM_{BD}.4 allows the identification of the Hive tables, some of them with pre-computed aggregates, like $aggCal_T$, $aggCus_T$, $aggLoc_T$, $aggProd_T$, $aggCusProd_T$, $aggCalProd_T$, $aggCalCus_T$, $aggCusLoc_T$, and $aggCalLoc_T$, and that can be directly used for analysis and visualization purposes, and those that have all the details available in the fact tables, like $CalCusProd_T$ and $CalCusLoc_T$, to allow different computations on data.

By this rule, the columns present in all column groups previously identified and presented in Figure 7, are columns of the Hive tables, now presented in Figure 8.

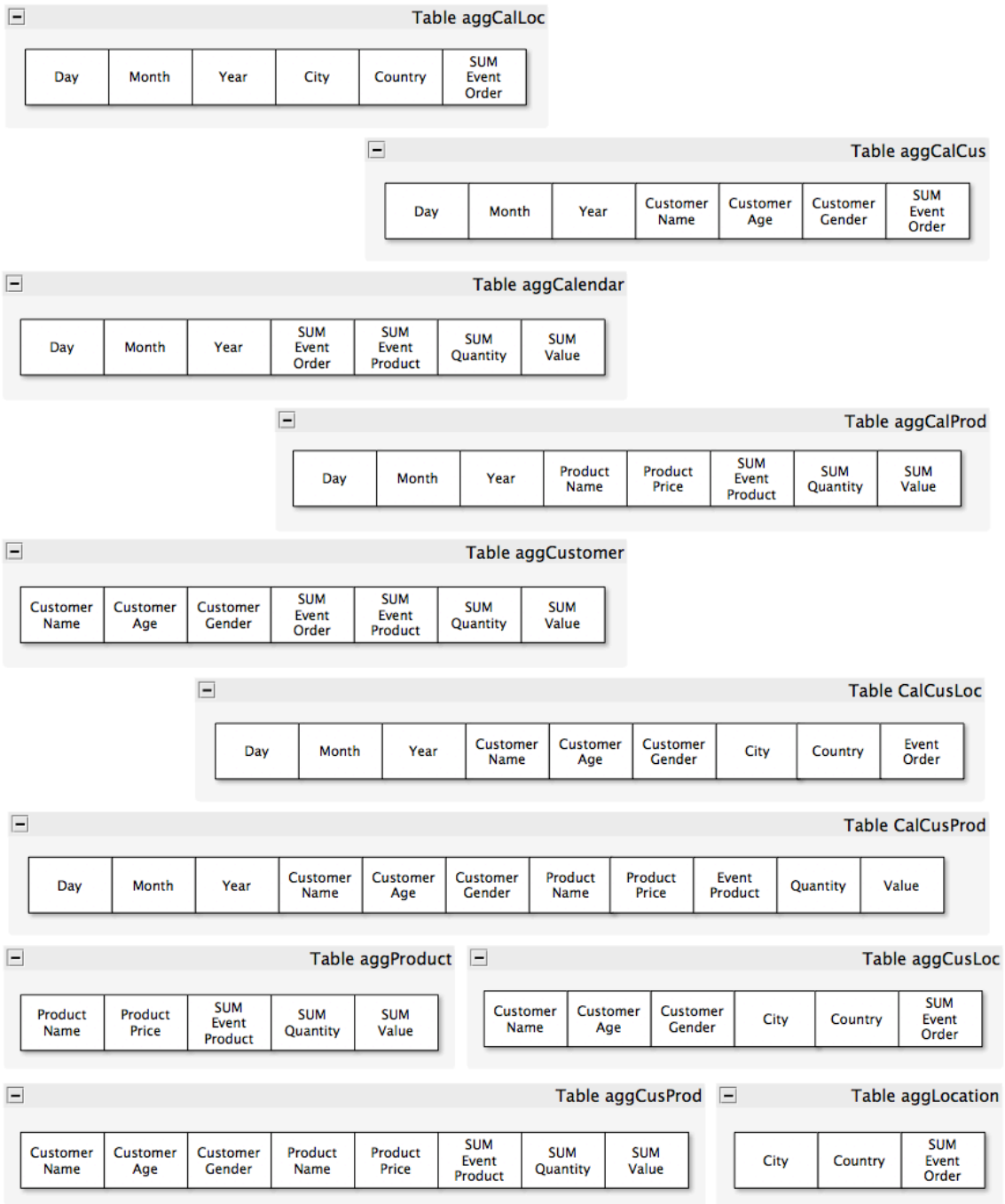


Figure 8. Resulting set of Hive tables for Orders

Regarding Rule ADM_{BD}.5, for partitioning and bucketing, and as those activities are dependent on the available data, only an example will be provided. Taking as an example table *aggCusProd*, with the summaries of orders by customer and products, and considering that the names of the clients have lower cardinality than the names of the products, the first one could be used as a partitioning attribute, and the other one as a bucketing attribute. Moreover, the age or gender of the clients could be used as a sub-partitioning attribute, for instance.

In order to verify how these tables complement each other and the usefulness of the proposed aggregates, Figure 9 presents a small example that includes 10 rows in the *OrderProducts* fact table (Figure 5). As can be seen, the orders were made between 10 and 13 of February 2016, and 5 different products were ordered by 5 different customers. The event-tracking attribute (*Event Product*) allows the counting of the different transactions, while *Quantity* and Value give the amount of ordered products and the corresponding amount value, respectively.

Following the several steps proposed in the previous section, and taking as example the aggregated table *aggCalCus*, Figure 10 shows how data start to be aggregated providing summaries of the total quantities and values of the ordered products, by customer and day. Moreover, the event-tracking attribute also reflects the number of transactions that were grouped.

					Customer Name / Customer Age / Customer Gender													
					Customer 1			Customer 2			Customer 3			Customer 4			Customer 5		
					25			33			27			42			54		
					M			M			F			F			F		
Year	Month	Day	Product Name	Product Price	SUM Event Product	SUM Quantity	SUM Value	SUM Event Product	SUM Quantity	SUM Value	SUM Event Product	SUM Quantity	SUM Value	SUM Event Product	SUM Quantity	SUM Value	SUM Event Product	SUM Quantity	SUM Value
2016	2	10	Product A	€25	1	10	€250												
			Product B	€10				1	30	€300									
			Product D	€17	1	20	€340												
		11	Product D	€17	1	22	€374												
			Product E	€5							1	7	€35						
		12	Product C	€12	1	12	€144							1	10	€120			
			Product E	€5										1	12	€60			
		13	Product B	€10													1	15	€150
			Product D	€17													1	5	€85

Figure 9. Available data in the *OrderProducts* fact table

			Customer Name / Customer Age / Customer Gender													
			Customer 1			Customer 2			Customer 3			Customer 4			Customer 5		
			25			33			27			42			54		
			M			M			F			F			F		
Year	Month	Day	SUM Event Product	SUM Quantity	SUM Value	SUM Event Product	SUM Quantity	SUM Value	SUM Event Product	SUM Quantity	SUM Value	SUM Event Product	SUM Quantity	SUM Value	SUM Event Product	SUM Quantity	SUM Value
2016	2	10	2	30	€590	1	30	€300									
		11	1	22	€374				1	7	€35						
		12	1	12	€144							2	22	€180			
		13													2	20	€235

Figure 10. Available data in the *aggCalCus* table

The tables that were derived from the dimensions, like calendar, customer or products, represent the more aggregated level. In those, the summaries of all indicators are achieved in a detail in which totals by customers, products or dates are obtained. As can be seen in Figure 11 a) and b), for the *aggCus* and *aggProd* tables, using the columns present in each one of these tables, the highest level of summarization is presented, with a row per customer or product. Moreover, and as also shown in this figure (Figure 11 c) and d)), data analysis and visualization tools can be used to inspect the available data with specific charts or graphs. In this case, the tool used was Tableau [13], either for visualization as tables or as charts. In those, clear perspectives on the customers that purchased more or the best-seller products are provided.

Customer Name / Customer Age / Customer Gender

	Customer 1 25 M	Customer 2 33 M	Customer 3 27 F	Customer 4 42 F	Customer 5 54 F
SUM Event Product	4	1	1	2	2
SUM Quantity	64	30	7	22	20
SUM Value	€1,108	€300	€35	€180	€235

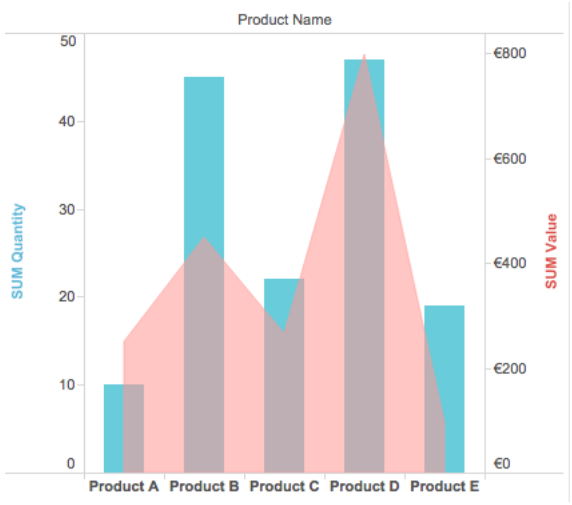
a) Table for *aggCus*

Product Name / Product Price

	Product A €25	Product B €10	Product C €12	Product D €17	Product E €5
SUM Event Product	1	2	2	3	2
SUM Quantity	10	45	22	47	19
SUM Value	€250	€450	€264	€799	€95

b) Table for *aggProd*

c) Chart for *aggCus*



d) Chart for *aggProd*

Figure 11. Data available in the *aggCus* and *aggProd* tables

5. CONCLUSIONS

This paper presented a set of rules for the automatic transformation of a multidimensional data model into a Hive data schema, which allows the implementation of a data warehouse in Big Data environments, supporting business intelligence and analytics in organizations. Considering the increasing ability provided by information technologies to collect, store and process data, organizational information systems need to be rethought, mainly in their supporting technologies, to face the challenges imposed by the volume, variety and velocity of Big Data.

Given the characteristics of Hive, a distributed storage repository for data warehousing purposes integrated in the Hadoop ecosystem, the on-line analytical processing capabilities of a traditional business intelligence context can be translated to a Big Data context, where different tables and aggregates on data are made available to the users, supporting their decision-making needs.

As future work, it is planned the application of the proposed transformation to a data warehouse with voluminous data, in order to fine-tune the rule for the definition of the partitions and buckets.

6. ACKNOWLEDGMENTS

This work has been supported by COMPETE: POCI-01-0145-FEDER-007043 and FCT – *Fundação para a Ciência e Tecnologia*, within the Project UID/CEC/00319/2013 (ALGORITMI). This work has also been funded by the SusCity project, MITP-TB/CS/0026/2013.

7. REFERENCES

[1] Chen, H., Chiang, R.H. and Storey, V.C. 2012. Business Intelligence and Analytics: From Big Data to Big Impact. *MIS quarterly*. 36, 4 (2012), 1165–1188.

[2] Kimball, R. and Ross, M. 2013. *The Data Warehouse Toolkit: The definitive Guide to Dimensional Modeling*. John Wiley & Sons, Inc.

[3] Hive 2016. Apache Hive, https://hive.apache.org.

[4] Thusoo, A., Sarma, J.S., Jain, N., Shao, Z., Chakka, P., Zhang, N., Antony, S., Liu, H. and Murthy, R. 2010. Hive-a petabyte scale data warehouse using hadoop. *Data Engineering (ICDE), 2010 IEEE 26th International Conference on* (2010), 996–1005.

[5] Hadoop 2016. Apache Hadoop, http://hadoop.apache.org.

[6] Cuzzocrea, A., Song, I.-Y. and Davis, K.C. 2011. Analytics over large-scale multidimensional data: the big data revolution! *Proceedings of the ACM 14th international workshop on Data Warehousing and OLAP* (2011), 101–104.

[7] Dehdouh, K., Bentayeb, F., Boussaid, O. and Kabachi, N. 2015. Using the column oriented NoSQL model for implementing big data warehouses. *Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA)*.

[8] Capriolo, E., Wampler, D. and Rutherglen, J. 2012. *Programming Hive*. O'Reilly & Associates.

[9] Di Tria, F., Lefons, E. and Tangorra, F. 2014. Design process for Big Data Warehouses. *Data Science and Advanced Analytics (DSAA), 2014 International Conference on* (2014), 512–518.

[10] Chevalier, M., El Malki, M., Kopliku, A., Teste, O. and Tournier, R. 2015. Implementation of multidimensional databases with document-oriented NoSQL. *Big Data Analytics and Knowledge Discovery*. Springer. 379–390.

[11] Chevalier, M., El Malki, M., Kopliku, A., Teste, O. and Tournier, R. 2015. Implementing Multidimensional Data Warehouses into NoSQL. *17th International Conference on Entreprise Information Systems (ICEIS), Barcelona, Spain* (2015).

[12] Santos, M.Y. and Costa, C. 2016. Data Models in NoSQL Databases for Big Data Contexts. *Proceedings of the International Conference on Data Mining and Big Data*, LNCS, Springer-Verlag, (Bali, Indonesia, Jun. 2016).

[13] Tableau 2016, http://www.tableau.com.