Data Models in NoSQL Databases for Big Data Contexts

Maribel Yasmina Santos(✉) and Carlos Costa

ALGORITMI Research Centre, University of Minho, Guimarães, Portugal
maribel@dsi.uminho.pt, id6011@alunos.uminho.pt

Abstract. Data models are a central piece in information systems, being the relational data models very popular and extensively used. In Big Data, and due to the characteristics of the NoSQL databases, the data modeling task is seen in another perspective, as those databases are considered schema-free. Nevertheless, these databases also need data models that ensure the proper storage and querying of the data. Considering the vast amount of relational databases and the ever-increasing volume of data, the importance of data models in Big Data increases. In this work, a specific set of rules is proposed for the automatic transition between a traditional and a Big Data environment, considering two specific objectives: the identification of a columnar data model for HBase supporting operational needs and the identification of a tabular data model for Hive supporting analytical needs. The obtained results show the applicability of the proposed rules and their relevance for data modeling in Big Data environments.

Keywords: Data model · Big Data · Data warehousing · NoSQL databases

1 Introduction

Organizations are constantly being challenged in several areas, and the challenges emerge from the markets, stakeholders, information technologies, business needs, among others. In what concerns information technologies, the evolution in the capacity to collect, store and process data has started the need of developing new environments, using different technologies, which require a redesign of the organizations' information systems, and their business intelligence and analytics capabilities [1].

Data models are a central piece in information systems design and development, and their analytical capabilities, as they ensure that data needs are properly considered. In a traditional organizational environment, the relational data models are very popular and are defined following strict rules about the requirements these models should consider. However, when we move to an emergent context like the one represented by Big Data, and due to the characteristics of the NoSQL databases, the data modeling task changes of perspective, as those databases are schema-free. This means that the data model can change in runtime, due to the needs of data storage or analytical tasks. The models, in this context, are designed considering the queries that need to be answered [2]. Although schema-free, these databases do need data models that ensure the proper storage of the data and their search with specific queries.

If we think on the vast amount of relational databases already implemented and foresee the need of transferring these databases to a columnar format, due to the ever-increasing volume of data, for instance, the importance of data models in Big Data increases. In contexts where we are expecting a fast transition from one environment to the other, the proposal of objective rules for this transition will benefit the users and will ensure proper data handling.

In this paper, a specific set of rules is proposed for a transition between a traditional organizational environment and a Big Data environment, considering two specific objectives: the identification of a columnar data model supporting operational needs and the identification of a tabular data model supporting analytical needs, both in Big Data contexts. While the first is suited to be implemented in a NoSQL database like HBase, the second is the basis for the data warehouse storage of Hive.

This paper is organized as follows. Section 2 presents related work. Section 3 describes the definitions and rules proposed for the automatic transformation of models. Section 4 uses a demonstration case to show the applicability of the proposed approach. Section 5 concludes with some remarks and guidelines for future work.

2 Related Work

In a Big Data context, where NoSQL databases are usually used, logical data models are schema-free, meaning that different rows in a table may have different data columns (less rigid structures) or that the defined schema may change on runtime [3]. In practical terms, those databases do have a data schema, but its definition follows a different approach. Instead of reflecting the relevant entities in a particular domain and the relationships between those entities, for instance, data schemas are defined having into consideration the queries that need to be answered, being data replicated as many times as needed, as long as the queries can be quickly answered. In this context, all queries that need to be answered must be known in advance [4], which could be a drawback if we consider analytical environments where data is available but the analytical needs depend on many different users and different contexts.

The work of [4] proposes an algorithm that automatically determines the most cost efficient data structure for a selected subset of NoSQL data stores, where cost efficiency is calculated based on the fees the developers need to pay in a cloud environment, which charges users based on the size of the stored data and the queries that need to be performed. This work uses the column-oriented data stores of Windows Azure Table and Amazon DynamoDB. The proposed approach starts from a predefined relational data schema and a set of queries, and performs automatic schema denormalization to find the optimal storage schema regarding a given query load. Also, mappings from the original queries to the newly created schemas are provided [4]. In this work, it is important to point out that queries must be known beforehand and that the proposed algorithm performs heavy denormalization, not optimizing the different number of generated data schemas.

Another work that proposes a heuristic-based approach for transforming a relational database into a columnar database, for HBase, is [3], and includes two phases. In the first one, three guidelines are used for transforming the relational database into the

HBase schema. In the second one, relationships between the two schemas are expressed and used to create a set of queries that transform the source relational data into the target representation. In the first phase, the authors recognize that business requirements and access/write patterns of the application are needed, and that this work is done by application experts, not proposing a semi-automatic or automatic process for dealing with the whole transformation.

The work of [5] 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. The authors propose a design methodology for the representation of a multidimensional schema at the logical level based on the key-value model. The proposed methodology is considered hybrid as integrates a data-driven approach design, using data repositories as main source of information, and a requirements-driven approach design, using information of the decision makers. In this case, the needs of decision makers are investigated, requiring additional information, besides input data models.

3 Data Schemas in Big Data Environments

With the advent of Big Data, the ability to collect, process and store data increases in a context where NoSQL databases are schema-free, allowing the storage of huge amounts of data without many concerns about the data structure. Those concerns emerge later on a schema-on-read approach, in which data is parsed, formatted and cleaned at runtime. Although having fewer concerns at the beginning of the collection phase, this adds several tasks latter when there is the need to develop specific applications to analyze the data. At some point, these schema-free repositories need to be transformed into some structured data model that allows data analysis by the users.

Considering that in a Big Data context there is the need to add structure to the data when analytical tasks need to be performed, this paper proposes a transformation process that leverages data modeling capabilities to the Big Data environment. The advantage of the proposed approach is that it considers all the business needs expressed in the operational data model that supports on-line transaction processing (OLTP) and uses that information for identifying useful data schemas in Big Data when analytical capabilities are needed. In this paper, a relational data model is used to automatically identify the appropriate data models in a NoSQL context, using the columnar data model of HBase and the tabular data model of Hive, here for a Big Data Warehouse context (BDW). In an operational context, those columnar data models allow data analysis in a real-time fashion, while the BDW allows analytical processes that run in a periodic fashion.

3.1 From an Operational Data Model to a Columnar Model

A column-oriented database in a NoSQL context is constituted by a set of tables defined row by row, but organized by groups of columns usually named column-families. This organization is followed by HBase and makes a vertical partitioning of the data. Each column-family may contain a variable number of columns and allows the lack of some columns between different rows of a same table [6]. Given this context, following are presented definitions and rules to formalize the transformation of a Relational Data Model to a Columnar Data Model.

Definition 1. A Relational Data Model, $RDM = (E, A, R)$, includes a set of entities $E = \{E^1, E^2, \ldots, E^n\}$, the corresponding attributes, $A = \{A^1, A^2, \ldots, A^n\}$, where A^1 is the set of attributes for entity E^1, $A^1 = \{A^1_1, A^1_2, \ldots, A^1_k\}$, and a set of relationships among entities $R = \{R^1_1(c_o|c_d), R^1_2(c_o|c_d), \ldots, R^l_m(c_o|c_d)\}$, where R^l_m express a relationship between E^m and E^l of cardinality c_o in the origin entity E^m and c_d in the destination entity E^l. Cardinalities can be of type $1 : n$, $m : n$ or $1 : 1$, with the optional 0 when needed.

Definition 2. A Columnar Data Model, $CDM = (T, CF)$, includes a set of tables $T = \{T^1, T^2, \ldots, T^n\}$, where each table integrates a key and a set of column-families, as $T^i = \left\{key^i, CF^1, \ldots, CF^m\right\}$. Each column-family integrates a set of columns representing the atomic values to be stored, $CF^j = \left\{C^1_j, \ldots, C^k_j\right\}$.

Rule CDM.1. Identification of Column-Families. The identification of column-families of a *CDM* follows a two-step approach:

Rule CDM.1.1. Identification of Descriptive Column-Families. Each entity only receiving cardinalities of 1 ($c_o = 1$ and $c_d = 1$) in the *RDM* corresponds to a descriptive column-family in the *CDM*, as this entity is used to complement the description of other entities (entity usually derived from the normal forms). The attributes of a descriptive column-family are constituted by the set of non-key attributes (excluding primary or foreign keys) present in the corresponding entities in the *RDM*.

Rule CDM.1.2. Identification of Analytical Column-Families. All entities present in the *RDM* and not identified by Rule CDM.1.1 as descriptive column-families, give origin to analytical column-families integrating the indicators or measures that can be analyzed considering the several descriptive column-families, as long as those entities have attributes of the relationships they are representing, which means that besides keys (primary or foreign), these entities need to integrate other attributes. The attributes of analytical column-families are constituted by the set of non-key attributes (excluding primary or foreign keys) present in the corresponding entities in the *RDM*.

Rule CDM.2. Identification of Tables. Two types of tables are proposed for a *CDM* supporting an operational system:

Rule CDM.2.1. Descriptive Tables. Descriptive tables are those tables that support specific data management tasks in an operational system. Correspond to the column-families identified by Rule CDM.1.1.

Rule CDM.2.2. Analytical Tables. For the identification of the set of analytical tables there is the need of identifying the data workflows present in the *RDM* and, from the set of data workflows, those that are able to represent all the data circulating in the *RDM*. For identifying the data workflows of a *RDM*, all entities receiving only cardinalities of 1 (used to identify the descriptive column-families by Rule CDM.1.1) start a data workflow following the *1:n* relationships associated to it, and all other *1:n* relationships that follows. The data workflow ends when a *n:1* relationship is found, meaning that it was possible to join a coherent piece of information that is related with each other and that was split in the *RDM* by the normalization forms. All identified workflows that are not fully contained by other workflows give origin to analytical tables.

Rule CDM.3. Integration of Column-Families into Tables. A specific table integrates a key, which is a very important component of a table in a *CDM*, and a set of column-families, which may vary depending on the type of table.

Rule CDM.3.1. Column-Families of Descriptive Tables. A descriptive table includes as a column-family, the column-family derived from the corresponding entity in the *RDM* (Rule CDM.1.1).

Rule CDM.3.2. Column-Families of Analytical Tables. An analytical table includes as column-families all the descriptive and analytical column-families associated with the entities included in the data workflow that gave origin to a specific table.

Rule CDM.4. Definition of the Tables' Key. A table's key should be able to assure an adequate performance throughout read and write access patterns from the application. The key represents a set of one or more attributes (concatenated) that have the potential to form a natural key that properly identifies each row in the *CDM*. This key must serve the applications' get, scan and put patterns, keeping them as short as possible, while maintaining the potential for adequate access patterns [6]. The order in which the attributes are concatenated plays a relevant role in the design of the key, since HBase stores keys in a sorted order [7].

To exemplify the proposed rules, let us consider a small example in which the *RDM* is represented by the relational data model depicted in Fig. 1.

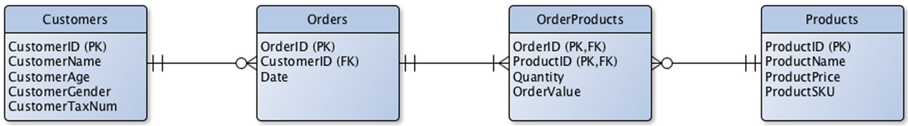

Fig. 1. Example of a *RDM* for orders

Taking into consideration the rules expressed for the transformation of a *RDM* into a *CDM*, the obtained model is depicted in Fig. 2, where the descriptive column-families $Customers_{CF}$ and $Products_{CF}$ (Rule CDM.1.1) are identified. By Rule CDM.1.2, the analytical column-families $Orders_{CF}$ and $OrderProducts_{CF}$ are identified. For tables, the tables of the *CDM* include as descriptive tables $Customer_{T}$ and $Products_{T}$ (Rule CDM.2.1). For analytical tables, two data workflows are identified, one starting in *Customers* and ending in *Products*, and the other starting in *Products* and ending in *Orders* (Rule CDM.2.2). However, as these workflows overlap, being one completely contained by the other, the contained one must be ignored.

This leads to the identification of one analytical table, here named as *Orders*$_T$. The integration of the column-families in the corresponding tables is achieved following Rule CDM.3. Rule CDM.4 must form a concatenated key obtained from the available attributes, as shown in Fig. 2. In the case of the customers' key, as well as in other keys, it might be needed to use the inverse of the key (with regard to the *CustomerTaxNum*), to avoid the creation of hot regions in the storage cluster.

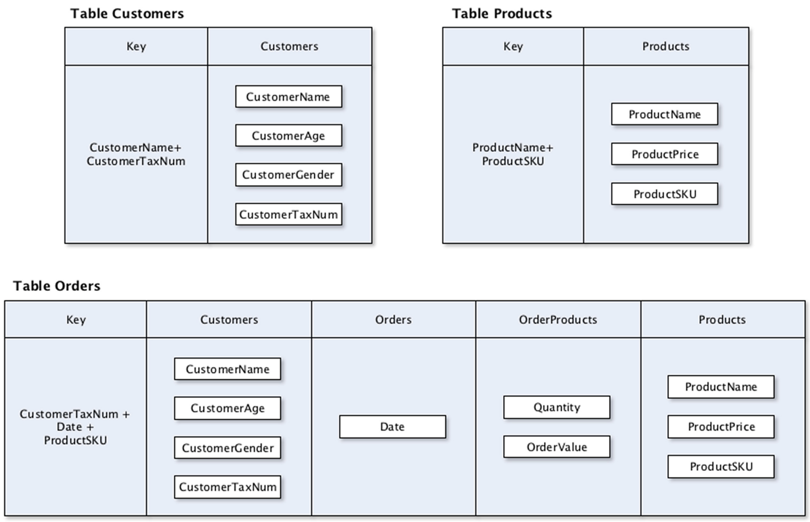

Fig. 2. Resulting *CDM* for orders

3.2 From a Columnar Model to a Tabular Data Model

In an analytical context for decision support, it is expected that the user could ask for aggregates based on the available data. In this work, it is proposed that the data warehouse model in the Big Data context, using Hive, be created based on the NoSQL columnar model (CDM). Hive [8] 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 [9]. 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 [10].

At this point, the *CDM* enables the derivation of a tabular data model that is formally defined as follows.

Definition 3. A Tabular Data Model, *TDM* $= (T, C)$, includes a set of tables $T = \{T^1, T^2, \ldots, T^n\}$, where each table integrates a set of columns, $T^i = \{C^1, \ldots, C^m\}$, storing the atomic values.

Rule TDM.1. Identification of Tables. All the analytical tables in a *CDM* give origin to tables in the *TDM*, as those will include the relevant data for data analysis and decision support.

Rule TDM.2. Identification of Columns. The identification of columns for each table identified by Rule TDM.1 follows a two-step approach.

Rule TDM.2.1. Identification of Descriptive Columns. The attributes that integrate the several descriptive column-families of the *CDM* give origin to descriptive columns of a table in the *TDM*.

Rule TDM.2.2. Identification of Analytical Columns. The attributes that integrate the several analytical column-families of the *CDM* give origin to analytical columns of a table in the *TDM*. As the *TDM* contains data that can be aggregated, the user can adopt different aggregation functions as SUM, MIN, MAX, AVG or COUNT to summarize the analytical columns.

Considering the example presented in previous subsection, one table is identified to be included in the TDM (Fig. 3), $Orders_T$ (Rule TDM.1). For $Orders_T$, the descriptive attributes *CustomerName*, *CustomerAge*, *CustomerGender*, *CustomerTaxNum, Date*, *ProductName*, *ProductPrice* and *ProductSKU* are identified as descriptive columns (Rule TDM.2.1), while *Quantity* and *OrderValue* are identified as analytical columns (Rule TDM.2.2). These analytical columns can now be manipulated in Hive, providing different analytical perspectives with different aggregations.

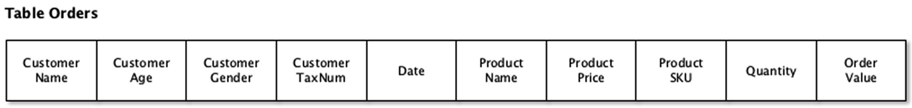

Table Orders

Customer Name	Customer Age	Customer Gender	Customer TaxNum	Date	Product Name	Product Price	Product SKU	Quantity	Order Value

Fig. 3. Resulting *TDM* for the orders

4 Demonstration Case

This section presents a more complete example of the model transformations proposed in this paper, for defining the data models in a Big Data environment taking into consideration an operational data model expressed in relational data model. The *RDM* used as starting point is presented in Fig. 4, from [11]. The model integrates eight entities, several attributes, and the relationships among the entities.

From this data model, we will start by the rules expressed in Sect. 3.1 for the identification of a columnar NoSQL data model. The descriptive column-families present in the diagram of Fig. 4 are $Hotels_{CF}$, $POIs_{CF}$, $Guests_{CF}$ and $Amenities_{CF}$ (Rule CDM.1.1) as these entities only receive relationships with cardinality of 1. All other entities in the model give origin to analytical column-families (Rule CDM.1.2), namely $Rooms_{CF}$ and $Reservations_{CF}$ (*HotelPOIs* and *RoomAmenities* are not identified as column-families as they do not integrate other attributes besides keys). The attributes of these column-families are the attributes of the corresponding entities in the *RDM*, excluding the keys, either primary (PK) or foreign (FK) keys. Regarding Rule CDM.2, the descriptive tables are $Hotels_T$, $POIs_T$, $Guests_T$ and $Amenities_T$ associated with the identified descriptive column-families (Rule CDM.2.1), while for the identification of the analytical tables there is the need of identifying the data workflows of the *RDM*

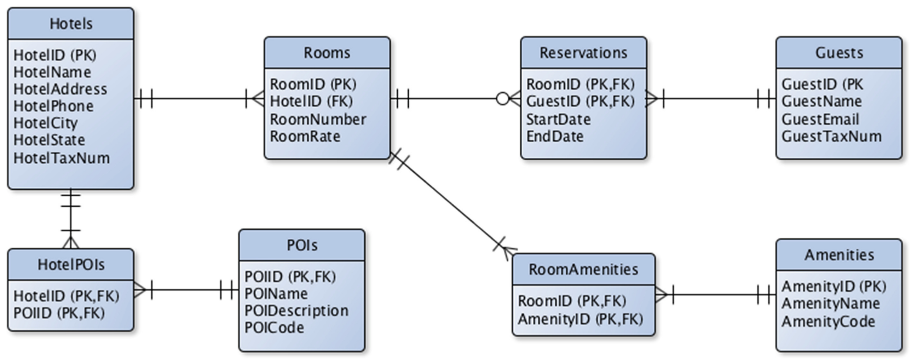

Fig. 4. *RDM* for the hotel's demonstration case (Adapted from: [11])

(Rule CDM.2.2). Starting by all entities only receiving relationships with cardinalities of 1, Fig. 5 presents the three identified data workflows, which lead to three analytical tables from now on named *Reservations*$_T$, *RoomAmenities*$_T$ and *HotelPOIs*$_T$.

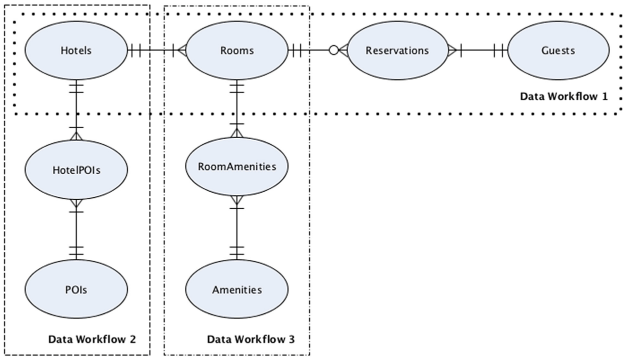

Fig. 5. Data Workflows for the *RDM*

Having identified the column-families and the several tables, it is now time to assign to each table its respective column-families. Each descriptive table integrates a column-family derived from its corresponding entity. For the analytical tables, *Reservations*$_T$ integrates four column-families, *Hotels*$_{CF}$, *Rooms*$_{CF}$, *Reservations*$_{CF}$ and *Guests*$_{CF}$, *RoomAmenities*$_T$ integrates two column-families, *Rooms*$_{CF}$ and *Amenities*$_{CF}$, and *HotelPOIs*$_T$ also integrates two column-families, *Hotels*$_{CF}$ and *POIs*$_{CF}$. Figure 6 depicts the identified *CDM* with the seven resulting tables.

Looking to the obtained model, and comparing it with the one presented in [11], it was possible to see that although both models present the same number of tables, seven, they are organized in a different way. Nevertheless, both are able to answer the same questions. At this point, it is important to mention that there are many possible ways to organize a columnar data schema [11]. In this paper, we presented one that was

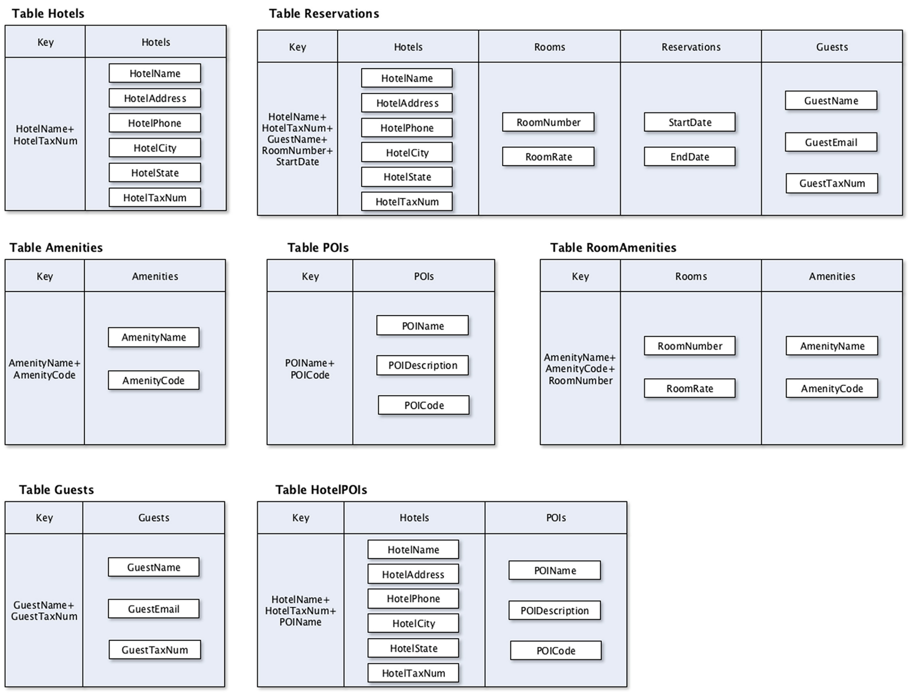

Fig. 6. *CDM* for the hotel's demonstration case

obtained through an automatic process that aims to help the users or data modelers in this task.

After the identification of the *CDM*, it is possible to proceed with the rules specified in Sect. 3.2 for the identification of a *TDM*. In this model, given the characteristics of Hive as a data warehouse for Big Data contexts, Rule TDM.1 allows the identification of the Hive tables based on the analytical tables of the *CDM*, which leads to the identification of *Reservations*$_T$, *RoomAmenities*$_T$ and *HotelPOIs*$_T$. By Rule TDM.2, these tables will integrate several descriptive columns derived from the descriptive column-families *Hotels*$_{CF}$, *POIs*$_{CF}$, *Guests*$_{CF}$ and *Amenities*$_{CF}$ (Rule TDM.2.1) and analytical columns derived from the analytical column-families *Rooms*$_{CF}$ and *Reservations*$_{CF}$ (Rule TDM.2.2). For these analytical tables, Fig. 7 presents the obtained *TDM*.

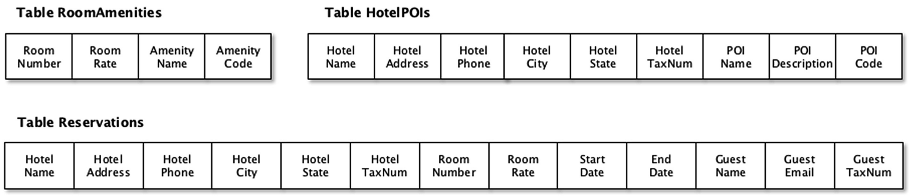
Table RoomAmenities

Room Number	Room Rate	Amenity Name	Amenity Code

Table HotelPOIs

Hotel Name	Hotel Address	Hotel Phone	Hotel City	Hotel State	Hotel TaxNum	POI Name	POI Description	POI Code

Table Reservations

Hotel Name	Hotel Address	Hotel Phone	Hotel City	Hotel State	Hotel TaxNum	Room Number	Room Rate	Start Date	End Date	Guest Name	Guest Email	Guest TaxNum

Fig. 7. *TDM* for the hotel's demonstration case

5 Conclusions

This paper presented a specific set of rules in a two-step approach for identifying a columnar data model suited to be implemented in HBase, and a tabular data model prepared to be implemented in Hive. Both consider that there are organizational contexts where an operational database cannot continue to be supported by a relational database in a traditional technological environment, and that the transition for a Big Data context is needed. In this case, this work proposes the columnar data model for supporting operational tasks, while the tabular data model can support analytical needs in a data warehouse environment for Big Data.

As advantages of the proposed approach we point out that the process is fully automatic, guiding the identification of the tables of the columnar data model, as well as their column-families, using as input the information available in a relational data model. Moreover, a process to generate the key of these tables is also suggested, due to the importance of the key in searching for data. Using the information of the columnar data model, the transition for a tabular data model is also addressed in an automatic way. Besides these, it is an integrated process that considers all the data needs expressed in the relational data model, ensuring that the business needs that were initially modeled are still applicable.

As future work, it is planned the application of the proposed approach to a more complex scenario, in order to complement this work. Other data models, like the one supported by Cassandra, should also be considered, as well as other particular organizational needs that may be expressed in the relational data model used as input of the proposed process.

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

References

1. Chen, H., Chiang, R.H., Storey, V.C.: Business intelligence and analytics: from Big Data to Big Impact. MIS Q. **36**, 1165–1188 (2012)
2. Durham, E.-E., Rosen, A., Harrison, R.W., et al.: A model architecture for Big Data applications using relational databases. In: 2014 IEEE International Conference on Big Data (Big Data), pp. 9–16. IEEE (2014)
3. Li, C.: Transforming relational database into HBase: a case study. In: 2010 IEEE International Conference on Software Engineering and Service Sciences (ICSESS), pp. 683–687. IEEE (2010)
4. Vajk, T., Feher, P., Fekete, K., Charaf, H.: Denormalizing data into schema-free databases. In: 2013 IEEE 4th International Conference on Cognitive Infocommunications (CogInfoCom), pp. 747–752. IEEE (2013)
5. Di Tria, F., Lefons, E., Tangorra, F.: Design process for Big Data warehouses. In: 2014 International Conference on Data Science and Advanced Analytics (DSAA), pp. 512–518. IEEE (2014)

6. HBase: Apache HBase (2016). https://hbase.apache.org
7. Khurana, A.: Introduction to HBase schema design. White Paper, Cloudera (2012)
8. Hive: Apache Hive (2016). https://hive.apache.org
9. Thusoo, A., Sarma, J.S., Jain, N., Shao, Z., Chakka, P., Zhang, N., Antony, S., Liu, H., Murthy, R.: Hive-a petabyte scale data warehouse using hadoop. In: 2010 IEEE 26th International Conference on Data Engineering (ICDE), pp. 996–1005. IEEE (2010)
10. Capriolo, E., Wampler, D., Rutherglen, J.: Programming Hive. O'Reilly & Associates, Sebastopol (2012)
11. Hewitt, E.: Cassandra: The Definitive Guide. O'Reilly, Beijing (2011)