

Startable: Multidimensional Modelling for Column-Oriented NoSQL

Leandro Mendes Ferreira¹^a, Solange Nice Alves-Souza¹^b and Luciana Maria da Silva^{2,3}^c

¹*Departamento de Engenharia de Computação e Sistemas Digitais (PCS), Universidade de São Paulo (USP), Brazil*

²*Department of Mathematical Sciences, Durham University, U.K.*

³*Centro de Estudos de Petróleo (CEPETRO), University of Campinas (Unicamp), Brazil*

Keywords: NoSQL, Multidimensional Modelling, Data Modelling, Database Query Processing and Optimisation (Theory).

Abstract: NoSQL Database Management Systems (DBMS) can be an alternative for analytical systems and have been used in this regard. As analytical systems often use multidimensional data modelling, which was created for relational databases, a new logical data modelling is necessary for NoSQL Column-Oriented Databases. We developed logical modelling (Startable) and applied NoSQL oriented to the family of columns. Furthermore, a logical model for the application of Startable modelling is presented in this research and a benchmark where the performance of the proposed modelling is demonstrated compared to traditional approaches to multidimensional modelling.

1 INTRODUCTION

Decision-making systems are fundamental to many organisations in the most diverse businesses. These systems are considered key parts of organisations' business development, and for many years, they were seen as a competitive differential. In the 1990s, these decision-making systems were defined as Business Intelligence (BI) and comprised modelling tools and technologies to analysis tools (Duan and Xu, 2012).


Data Warehouse (DW), a database dedicated to the decision-making context, is one of the main components in BI environments (Yessad and Labiod, 2016). According to Lins and Ferreira (2016), the multidimensional model is the most adopted.


The big data systems with Hadoop and NoSQL have significant differences concerning the Relational Database Management Systems (RDBMS) in several characteristics. NoSQL uses non-relational data structures, does not have an SQL query language (present in the vast majority of RDBMSs), does not meet ACID characteristics (an acronym for Atomicity, Consistency, Isolation, and Durability), and has the persistence of distributed data (Moniruzzaman and Hossain, 2013). Thus, data


modelling with the ER and Multidimensional Modelling are not applied to these new technologies since they are designed to assist RDBMS.

This research has the main focus on analytical systems. We developed the modelling of logical data called Startable, which needs the Column-Oriented NoSQL physical model. This modelling pretends to be simple, based on the business rules of the users and widely known concepts of modelling. Finally, we propose that the modelling improves the efficiency of the return of queries.

This paper is structured as follows: Session 2 presents Column-Oriented NoSQL; Session 3 presents related research and with subsidies for the development of this article; Session 4 offers an explanation of the importance of partitioning mechanisms for NoSQL; Session 5 shows the building of Startable modelling; Session 6 presents the Star Schema Benchmark (SSB), the benchmark applied for testing of modelling performance; Session 7 presents the modelling performance results over the benchmark in a controlled laboratory; Session 8 presents the conclusion and proposals for future work.

^a <https://orcid.org/0000-0003-0680-1100>

^b <https://orcid.org/0000-0002-6112-3536>

^c <https://orcid.org/0000-0001-9544-9748>

2 COLUMN-ORIENTED NoSQL

NoSQL is a term that groups non-relational DBMS, which serve large volumes of data with high performance and high availability. According to Moniruzzaman and Hossain (2013), NoSQL has the following characteristics:

- Absence of the standard SQL-ANSI query language
- Horizontal scalability
- Schemaless or Schema Free
- Sharding and Replication

NoSQL DBMSs are highly relevant for applications that need a technology which supports large volumes of data management and scalability efficiently and straightforwardly (Lóscio et al., 2011).

Currently, NoSQL DBMSs are classified into four basic types of architectures: Key-Value, Document Oriented, Column Family and Graph Oriented (Moniruzzaman and Hossain, 2013). This research focuses on Oriented Column (Family Column) NoSQL, as shown in Figure 1.

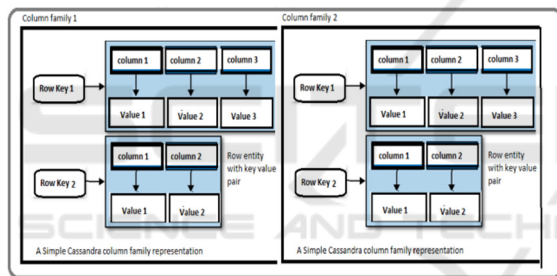


Figure 1: Example of Oriented Column Data Model - (Manoj V, 2014)

Column-family or column-oriented NoSQL has extensible record storage and large columnar stores. This type of NoSQL was initially inspired by Google's Bigtable, a distributed DBMS based on structured data built to handle vast volumes of data. Column-oriented NoSQL data stores are hybrid row/column storage, with significant differences from common relational databases. Despite having a similar concept to column-by-column storage of columnar databases and columnar extensions to row-based databases, column-oriented NoSQL does not persist data in tables. Instead, it stores the data in massively distributed architectures. A key is associated with one or more attributes (columns). This type of columnar storage persists its data to be optimally aggregated to lower I/O costs and offers high scalability in data storage. Data stored in the database is based on column family sort order (Manoj V, 2014). The most popular column-oriented NoSQL

on the market are Apache HBase, Apache Cassandra and Google Big Table.

3 RELATED WORKS

Different researchers developed papers regarding the problem of modelling for analytical systems focusing on the use of a NoSQL database and adaptation of multidimensional modelling for new non-relational database types. We highlight the three significant modelling concentrations proposals:

1°: The purpose of the classic Star Schema model in NoSQL, for example, Carniel et al. (2012) proposed applying a multidimensional normalised model, or the classic "star schema" modelling over columns-oriented NoSQL.

2°: Several authors among them (Carniel et al., 2012; Chevalier et al., 2015a; Ferreira, 2015; Murazza and Nurwidyantoro, 2016) proposed the Fully Non-Normalised Multidimensional Modelling (FNMM), which is a joining of a fact table with its dimensions of Star Schema model in a single table.

The physical implementation of FNMM in column-oriented NoSQL facts and their dimensions are gathered in the same table, using a single column family as seen in Dehdouh et al. (2014). Also, Scabora et al. (2016a) call a similar approach SameCF. Another option for a physical model for column-oriented NoSQL is proposed in Chevalier et al. (2015b), outlined in Figure 2. Facts and dimensions belong to the same table; however, facts and each dimension are represented in separated column families. Scabora et al. (2016a) also proposed this same option, called CNSSB; Bouaziz et al. (2019) suggest the same. This approach for physical implementation is similar to non-nested document-oriented NoSQL.

It is worth highlighting that each specific NoSQL from the same type of family has different implementations. For example, HBase and Cassandra belong to the same NoSQL family, but each has its implementation. Thus, the physical model of non-normalised data for column-family-oriented NoSQL is adapted according to a specific NoSQL.

Using FNMM for complex hierarchies in document-oriented NoSQL, Bonnet et al. (2011a), Chevalier et al. (2015a), and Tournier (2017a) suggest the use of arrays to construct the hierarchies. For column-family-oriented NoSQL, Chevalier et al. (2015b) show that dimension hierarchy should be mapped as a column for each dimension, i.e., physical columns in column families representing each dimension.

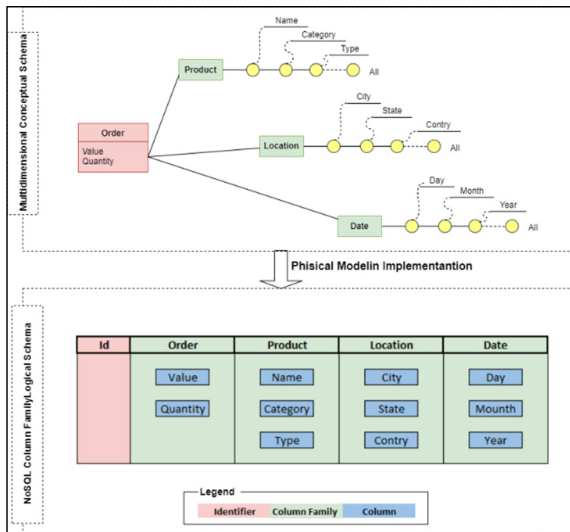


Figure 2: Physical model Implementation in Column-Oriented NoSQL - (Max Chevalier et al., 2015).

3^o: The FNMM variations adding the materialisation of aggregations are pre-computation of fields are used for aggregates to persist along to this model physically. More details can be seen in (Carniel et al., 2012; Zhao and Ye, 2014; Chavalier et al., 2016), among other works. It is worth mentioning that some researchers suggest tools to improve the indexing way of data (Vasilakis et al., 2017), or tools for a better choice of partitioning models, automating this partitioning in an automated way through Machine Learning algorithms (statistical models) (Boussahoua et al., 2017; Cugnasco et al., 2016). These models use techniques such as clustering using K-Means, Linear Regression, or Hash algorithms. They automatically discover the best indexing or partitioning, not considering the users' business rules, query needs, or even data recovery.

Another note is that many works use tools or frameworks in conjunction with NoSQL to aid queries, generally using a traditional query language like SQL ANSI. The authors used the structures of the following frames Apache Hive (Cugnasco et al., 2016), Apache Phoenix (Dehdouh et al., 2014), as well as the built by their researchers for this task (Zhao and Ye, 2014). Regarding the Apache Hive, it is used with NoSQL to assist in the exposure of metadata and queries on NoSQL.

Nevertheless, the modelling developed by (Amirthalingam and Rais, 2018) is also an adaptation of the FNMM, but the authors do not encourage the total merging of dimensions and facts. The authors used a set of rules for joining some dimension tables and tables with pre-computed aggregations, adding structures of partitions and buckets of Hive. This use

depends exclusively on the cardinality of the data, having no relation to business rules.

Concerning the FNMM, Chebotko et al. (2015) proposed that this modelling can be firmly based on the business rules, and the applications will carry out queries. Another predominant factor in the modelling developed by the authors is the primary keys model, which involves hard work in the concepts of and the idea of partitions. Although the model developed by the authors is efficient in meeting the business requirements of several methods, they designed it for transactional systems. Also, this modelling is linked intrinsically to the NoSQL Cassandra and its physical model of persisting data.

4 PARTITIONING AND CLUSTERING OF DATA ON NoSQL

Data partitioning and primary keys modelling are the main mechanisms used by NoSQL databases for a balanced distribution of data in the cluster. These concepts have fundamental importance in the performance of these technologies. Figure 3 presents the effect of data partitioning on a Cassandra cluster.

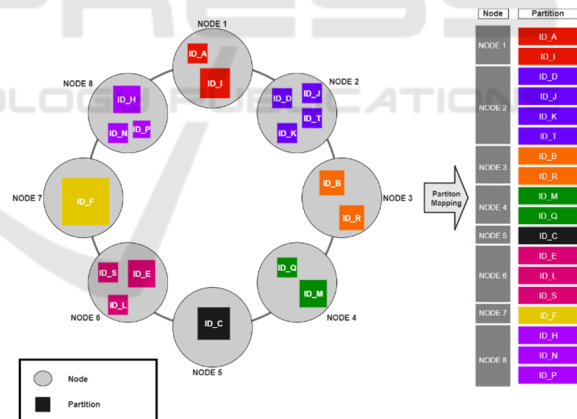


Figure 3: Partitioning on Clustering Cassandra.

As shown in Figure 3, the Cassandra cluster is represented as a ring. The fields belonging to the cluster key are responsible for distributing data in the cluster (Hewitt and Carpenter, 2020).

Besides, Cassandra and other NoSQL and distributed data systems utilise indexing algorithms and hashing based on partitions and buckets to aid in insertion and data query. The incorrect choice of fields in the keys and partitions of each NoSQL and Hive can cause data to unbalance in the cluster. It can have partitions with excess data on a single node or

with little or no data on the nodes. So, the correct use of the designated fields as partitions or buckets can increase the efficiency of the queries. Besides, it is possible to avoid the Full Scan. It also potentiated the indexing and hashing algorithms to aid the systems in retrieving data to answer the queries. See more details in (Hewitt and Carpenter, 2020; Chodorow and Bradshaw, 2018; Du, 2018).

5 STARTABLE – LOGICAL MULTIDIMENSIONAL MODELLING FOR NoSQL

As a basis for the development of Startable, we use the Star Schema multidimensional modelling proposed by (Inmon, 2002; Kimball and Ross 2002) and the principles of modelling presented by (Chebotko, Kashlev and Lu 2015). We modified and adapted the modelling to meet the physical data model present on columns-oriented NoSQL. Beyond the Startable modelling, we proposed a set of principles and a flow of processes that guide the application of the logical model of the data. We tested the modelling and flow (Figure 4) in a controlled environment to verify their applicability and efficiency.



Figure 4: Logical Flow of Startable Modelling.

We highlighted and discussed, Figure 4, the four main stages for creating the Startable modelling and the tasks involved in each of these stages:

1. Star Schema Modelling:

- Develop the standard process of multidimensional modelling seen in (Kimball and Ross, 2002; Inmon, 2002);
- Perform complete denormalisation by transforming the Star Schema model into a single table composed of facts and all dimensions, as proposed in (Chevalier et al., 2015a; Ferreira, 2015; Dehdouh et al., 2014).

2. Partition Selection:

- Identify the dimensions that are most used as filters in queries, for example, Dates, Geolocalization (country, states, cities),

Organizational Data (company name, branch name);

- Identify the mandatory filters applied in proportion to Pareto. According to the theoretical concept of Pareto, in most cases, we asked 80% of the examples to use 20% of the dimensions as filters (Zhu and Xiang, 2016; Vandewalle et al., 2007);
- Identify the dimensions that have a distribution more uniform of data in the cluster; for example, a more granular data dimension is probably more evenly distributed. Therefore, among the aspects that identify geolocation data, for example, the City dimension should probably have a better distribution characteristic between the cluster than the Country or State dimensions;
- Minimise the number of fields for partitions (1 to 5 partition fields generally) (Chebotko et al., 2015);
- Do not use partition fields that are not used as filters in queries. Not using this task can generate a Full Scan in the database.

3. Buckets Selection:

- Identify the frequency of use of filters not required;
- Identify the frequency of ordering data usage (order by);
- Identify the frequency of data usage in the grouping (group by) and drill-down;
- Identify the order of the granularity of the dimensions and insert them in a reverse way (less granular to the more granular). For example, we can have the following hierarchy: Year, Month, Day. The Year field is the least granular of this aspect. If we need the three fields in the buckets, we must first insert the year field, followed by month and day, as this facilitates the sorting and grouping the data and drill-downs.

4. Priority - Partition and Buckets:

- Select, like partitions and buckets, fields that appear in most queries answering the business questions;
- If there is a possibility of performance testing of queries, then we must select partitions and buckets which we can use in the consultations with the most impact on the workload of the analytical system;
- When more than one field is partitioned, first use the minor granularity field, followed by the more granular area. For example, for the dimension Date, we must first use the Year field, then the Month field and so forth;

- In the case of hierarchies of dimensions, use the most specific field as a bucket, for example, for the hierarchy of Date (YYYYMMDD). But, if there is not or need to use more than one hierarchy field, first use the areas with less granularity, for example, (Year, Month, Day);
- All fields used as a partition should be filtered in the queries to avoid a Full Scan.

Another essential point in Figure 4 is the "views," which permeate each stage. "Views," Figure 4, focused on data and the business model concerning the steps of the transition. We highlighted five views in Figure 4:

- **View over Traditional Model** - Develop the Star Schema traditional model on existing data;
- **View over Filters** - Establish the primary filters used in the business for most of the questions that the data in the model will answer;
- **View over Groups** - Establish the main groupings used by the business;
- **View over Order** - Establish the order and priorities that the queries will present;
- **View over Data Distribution** - Focus on the data distribution in the cluster. We need to ponder the model, improving the data distributed through the clusters. We can improve its efficiency in processing the queries.

6 BENCHMARK – DATA MODELLING

We applied an analytical benchmark to the distributed and controlled environment before of test the performance of the logical model proposed in this research (Startable). The chosen Benchmark model was the Star Schema Benchmark (SSB), proposed by O'Neil et al. (2009). In the scientific community, this benchmark stands out for its simplicity of application and for being a specific reference for multidimensional modelling. Also, it is a simplification of the TPH-C benchmark model. TPH-C follows the Snowflake modelling based on Star Schema but with high normalisation of dimensions. SSB transforms the Snowflake model on the Star Schema multidimensional model.

SSB has a data generator and a set of 13 ready queries, by which one can measure the query performance on several aspects such as filters,

equalities, inequalities, groupings, and ordering. SSB data model consists of a fact table and four dimensions modelled in Star Schema format, as shown in Figure 5.

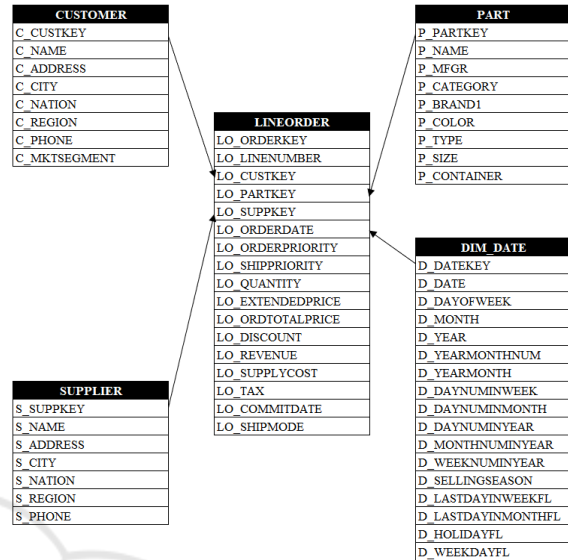


Figure 5: SSB Data Model.

We applied the SSB data model, the FNMM, and the Startable modelling. We pass by the four steps of the Startable modelling flow proposed in item 4. Firstly, we transform the SSB data model into the FNMM data model, in the following, in the Startable model. So, we replicated the Startable modelling over a Cassandra NoSQL database cluster (oriented to the column-family). Also, we measured the performance of the Startable model concerning the Star Schema traditional model and a version of the Star Schema modelling unifying facts and dimensions in a single table. Several authors proposed the last modelling (Carniel et al., 2012; Chevalier et al., 2015a; Ferreira, 2015; Dehdouh et al., 2014; Dehdouh, 2016a).

We cannot interfere in the SSB business model, for example, defining a mandatory field as a filter in all queries. Then, we studied the data and tables involved in understanding the business model implicit in the SSB. After applying the four steps of the Startable model, we kept the views of each of these. We obtained the Startable final model. We counted the fields with 13 queries, facilitating queries and SSB fundamental business models. Table 1 shows how the counting of fields is accomplished in queries of the SSB model.

Table 1: Count of Queries Arguments on SSB Model.

Line Labels	Field Count	Where Sum	Group by Sum	Order by Sum
c_city	3	0	3	0
c_nation	3	1	2	1
c_region	4	4	0	0
d_year	12	7	10	10
d_yearmonth	1	1	0	0
d_yearmonthnum	2	2	0	0
lo_discount	3	3	0	0
lo_quantity	3	3	0	0
p_brand1	4	2	4	4
p_category	3	2	1	1
p_mfgr	2	2	0	0
s_city	4	1	4	1
s_nation	3	1	2	1
s_region	7	7	0	0
General Total	54	36	26	18

In Table 1, the d_year field is often used as an argument in the queries, especially "where." Also, the d_year field appears as an argument in 12 of the 13 queries for the SSB. With this information described and interpreting the data of the SSB model, we applied and analysed the Startable modelling based on the four main steps presented in Figure 5.

Step 1 – Star Schema Model:

- No action was required to perform Star Schema modelling since the SSB data model was already in this modelling.
- We only perform the joining of the facts and dimensions by changing the Star Schema model to an FNMM by suppressing the primary key columns of the dimension tables.

Step 2 – Partition Selection:

- We identified that the d_year field was more used as a filter, following a proportion of Pareto (Zhu and Xiang, 2016; Vandewalle et al., 2007). This field also had a uniform data distribution, which qualified him as a candidate for partition.
- The s_region field could also be used as a partition because we often used it as a filter, and it could distribute the partitions into the smallest segments. However, this could also expressively fragment the partitions, so we only used the d_year field as a partition.

Step 3 – Buckets Selection:

- We identified the fields s_region 7 (seven) times and c_region 4 (four) times as filters.

- We identified the p_mfgr, p_category, and p_brand1 fields at least 2 (two) times, each one in clauses "order by" and "group by."
- Another vital piece of information to note is that we organised the hierarchical fields in the "part" dimension, where the "p_mfgr" is the less granular and the "p_brand1" the most granular.

Step 4 – Priority - Partition and Buckets:

- We applied the d_year field only for partition due to filter clauses' queries and the homogeneous distribution of data.
- We chose the s_region and c_region fields as the primary bucket due to the frequency in the queries of filter clauses. We used "s_region" and "c_region" in 7 queries as filters and did not apply them for grouping or ordination any single time. The "s_region" field has a smaller distribution, so we chose it as the primary bucket, followed by "c_region".
- By frequency of use in grouping and sorting clauses, we used the p_mfgr, p_category, and p_brand1 fields as a bucket following the dimension's hierarchy for this use.

The final modelling of partitions and buckets was defined, as shown in Table 2. We presented the order of the fields will be the order for the physical ordering of the data in the partitions. For NoSQL Apache Cassandra, we have the following structure of Clusters (in the Startable model refers to partitions) and partitions (in the Startable model refers to buckets):

```
"PRIMARY KEY ((D_YEAR), S_REGION, C_REGION, P_MFGR, P_CATEGORY, P_BRAND1))"
```

Table 2: Partitions and Buckets model applied to the SSB.

Startable		
Field	Partitions	Buckets
d_year	X	-
s_region	-	X
c_region	-	X
p_mfgr	-	X
p_category	-	X
p_brand1	-	X

For Apache Cassandra, the most internal parentheses refer to the fields defined as the clusters, and the most external refers to the fields defined as a partition.

We accomplished two tests with loads of different order data from the model developed. The first load had about 1GB of data, and the second load had 10GB of data. We tested two different data models and compared the results (as shown in section 7). Thus, the models tested were the traditional Star Schema, the FNMM, and the modelling proposed in this work: Startable.

To perform the benchmark, we used a distributed cluster. We built the cluster with four machines and used the Google Cloud Platform (as Platform as a Service (PaaS)). The machines used had the following configurations: 200 GB Hard Disk (not SSD), 8 CPUs (Intel Skylake), 48 GB Memory, and Operating System CentOS 7. The version of Cassandra was Cassandra 3.11.4, and we also used PrestoDB 0.214 as an auxiliary framework to perform the queries. The Cassandra Cluster was composed of 4 (four) nodes ring with identical configurations for each node. Cassandra clusters do not have a master/slave architecture, requiring only the processing nodes.

7 PERFORMANCE RESULTS OF THE STARTABLE MODELLING APPLICATION

We present the results concerning the methodology described above. To analyse the Startable model, we carried out two tests with data loads of different orders, about 1GB of data and 10GB of data. Also, we tested for each load of data the three data models: Traditional Star Schema, FNMM, and Startable modelling. Besides, we tested each modelling with the two data volumes on Apache Cassandra.

We can observe in Table 3 the total volume of data and the number of records generated for each modelling. Also, we can see in Table 3 that the volume of data is not much of a difference in storage for each model, as the most significant volume of data is in the line order fact table. There is no high volume increase when we perform the join for the FNMM model. The volume of data in the FNMM and Startable model is identical since, structurally, the data model is different only in partition and bucket selection.

The tests were based on the response time of each SSB query performed on each of the persistence systems. Remember that we performed all the queries to the PrestoDB applied on Apache Cassandra. We triggered the queries to the PrestoDB through a Python script that calculated the time spent for each

Table 3: Benchmark Data Volumetry.

Modelling	Volume	Storage	Quantity
customer	1GB	5.5M	60000
	10GB	47M	510000
supplier	1GB	328K	4000
	10GB	2.8M	34000
dim_date	1GB	228K	2556
	10GB	228K	2556
part	1GB	33M	400000
	10GB	83M	1000000
lineorder	1GB	1.2G	11998051
	10GB	9.9G	101987916
FNMM	1GB	1.2G	11998051
	10GB	10G	101987916
Startable	1GB	1.2G	11998051
	10GB	10G	101987916

of them. After the tests, we compiled the data and showed the results from Figures 6 and 7. We compared the response times of queries in seconds for the three models, Star Schema, FNMM, and Startable.

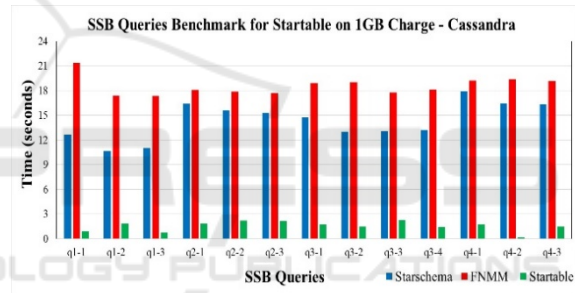


Figure 6: SSB Query Benchmark for Startable over 1GB Charge – Cassandra.

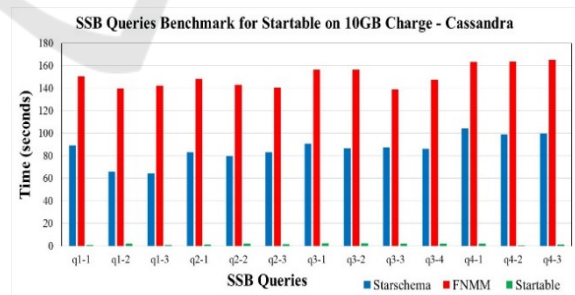


Figure 7: SSB Query Benchmark for Startable over 10GB Charge – Cassandra.

When we analysed the results from 1 GB and 10 GB of data for Startable modelling compared to other modellings, we observed a substantial performance gain in the return of the queries in both volumes. Concerning the charge of 10GB data, the Startable modelling with column-oriented NoSQL (Apache

Cassandra) was dozens of times faster than the Star Schema traditional model. We can verify a performance increase in the application of Startable modelling in large data volumes.

The turnaround time of queries on Startable modelling remained practically the same in 1GB and 10GB volumes; however, in traditional Star Schema and FNMM modelling, there was an increase proportional to the data volume.

We saw that the correct application of the Startable modelling could be 100 to 150 times more efficient than the Star Schema and FNMM modelling on queries in large volumes of data.

Through the tests carried out, we were able to verify that our central hypothesis regarding the Startable modelling proposition showed up to be coherent. Correctly defining the partitioning and buckets according to the needs of the business rules, rules that were expressed in SQL queries, can significantly reduce the query time in analytical environments based on column-oriented NoSQL.

The results of the performance of queries in FNMM modelling compared to traditional Star Schema modelling were surprisingly negative. It is common in the literature to verify that joining into a single table of facts and dimensions brings performance gains in queries (Dehdouh, 2016b; Scabora et al., 2016b; Chevalier et al., 2017; Bouaziz et al., 2017; Bonnet et al., 2011b; Tournier, 2017b). However, in our tests, there was an inverse result. Joining the fact and dimensions into a single table made the performance of queries worse. We believe that the better performance of traditional Star Schema than FNMM is due to optimisations of the PrestoDB processing framework, which applies advanced optimisations in its execution plan to improve performance in federated queries. PrestoDB may have used the primary keys of each of the tables in the Star Schema physical model to improve the performance of the queries, especially in the joining of tables and data selection. On the other hand, tables in the FNMM model have only a single partition primary key, which may have made PrestoDB's optimisations unfeasible. In later works, these suspicions need to be validated by scientific tests in controlled environments.

Finally, we can observe that the increase in performance in the queries on the Startable modelling occurred in all the proposed queries and the two-volume scenarios, maintaining a similar time for the return of the query regardless of the volume, which indicates substantial gains even with the addition of 10 times the volume of data. Other data quantities must be tested in later works to verify the continuity

of the performance gain of the Startable modelling even over vast volumes of data.

8 CONCLUSIONS AND FUTURE WORKS

We proposed in this work logical modelling for analytical systems with persistence in Column-oriented NoSQL, Startable modelling. Our modelling was able to unify two different concepts of multidimensional modelling, the Star Schema and fully non-normalised. Besides, it consolidated facts and dimensions in a single table and focused on modelling partitioning, bucket, and data ordering.

Startable modelling met the initial requirements for independent simplicity applicability of the business domain. It is modelling that bases its development on the business rules of each case, mainly to answer business questions more efficiently.

The proposed methodology for developing the model based on logical flow met the requirements of data queries and the better distribution of the data on the persistence system. We proposed and followed all steps of the workflow. Each of these steps must have a specific view of the process development, facilitating the replicability of the model in any data domain. Concerning the applicability, the results were successful since we obtained them from benchmark tests defined in the academic community and used for performance testing on a logical, analytical data model. We can highlight another reason for success in the analysis and results of modelling. We suggested and followed the steps in the Logical Flow of Development of the Startable modelling.

The performed tests showed significant gains in the performance of the organised data according to the Startable modelling about the classic Star Schema and the FNMM modelling. Several queries had substantial performance gains when running on Startable-organized data, especially on large data volumes (10GB of data) in data persistence on NoSQL Apache Cassandra.

We believe that Startable is flexible enough to adapt to other families of NoSQL, such as document-oriented ones. We also believe that its application can be helpful in Big Data distributed file systems such as HDFS S3, among others. Another exciting implementation that can be performed on top of Startable is on distributed relational databases specific to Data Warehouse, like Amazon Redshift, Azure Synapse, Google Big Query, Snowflake, and

Apache Druid, among others, which can benefit from the partitioning and buckets implementation model. Thus, we suggest implementing Startable over other data persistence models and NoSQL Document-oriented, HDFS/S3 and Distributed Relational Data Warehouse DBMS as future works.

REFERENCES

- Amirthalingam, T, and HM Rais (2018). Automated Table Partitioner (ATAP) in Apache Hive. At *2018 4th International Conference on Computer and Information Sciences (ICCOINS)*. IEEE.
- Bonnet, L, A Laurent, M Sala, B Laurent, and N Sicard (2011a). Reduce, you say: What NoSQL can do for data aggregation and BI in large repositories.
- Bonnet, L, A Laurent, M Sala, B Laurent, and N Sicard (2011b). Reduce, you say: What NoSQL can do for data aggregation and BI in large repositories.
- Bouaziz, S, A Nabli, and F Gargouri (2017). *From Traditional Data Warehouse To Real Time Data Warehouse*. Vol. 557, nan.
- Bouaziz, S, A Nabli, and F Gargouri (2019). Design a data warehouse schema from document-oriented database. vol. 159.
- Boussahoua, M, O Boussaid, and F Bentayeb (2017). Logical Schema for Data Warehouse on Column-Oriented NoSQL Databases. In *Database and Expert Systems Applications*, D. Benslimane, E. Damiani, W.I. Grosky, A. Hameurlain, A. Sheth, R.R. Wagner (eds.). vol. 10439, Springer International Publishing.
- Carniel, AC, A de Aguiar Sa, VHP Brisighello, MX Ribeiro, R Bueno, RR Ciferri, and CD de Aguiar Ciferri (2012). Query processing over data warehouse using relational databases and NoSQL. In *2012 XXXVIII Conferencia Latinoamericana En Informatica (CLEI)*. IEEE.
- Chevalier, M, M El Malki, A Kopliku, O Teste, and R Tournier (2016). Document-oriented data warehouses: Models and extended cuboids, extended cuboids in oriented document. The *2016 IEEE Tenth International Conference on Research Challenges in Information Science (RCIS)*. IEEE.
- Chebotko, A, A Kashlev, and S Lu (2015). A Big Data Modeling Methodology for Apache Cassandra. In *2015 IEEE International Congress on Big Data*.
- Chevalier, M, M El Malki, A Kopliku, O Teste, and R Tournier (2015a). Implementation of Multidimensional Databases with Document-Oriented NoSQL. In *Big Data Analytics and Knowledge Discovery*, S. Madria, T. Hara (eds.). vol. 9263, Springer International Publishing.
- Chevalier, M, M El Malki, A Kopliku, O Teste, and R Tournier (2015b). Implementing multidimensional data warehouses into NoSQL. H.S. Maciaszek L. Maciaszek L., Teniente E. (ed.). vol. 1.
- Chevalier, M, M El Malki, A Kopliku, O Teste, and R Tournier (2017). Document-Oriented Data Warehouses: Complex Hierarchies and Summarizability. In *Advances in Ubiquitous Networking 2*, R. El-Azouzi, D.S. Menasche, E. Sabir, F. De Pellegrini, M. Benjillali (eds.). vol. 397, Springer Singapore.
- Chodorow, K, and S Bradshaw (2018). *MongoDB: The Definitive Guide*. 3rd ed., O'Reilly Media.
- Cugnasco, C, Y Becerra, J Torres, and E Ayguadé (2016). D8-tree: a de-normalised approach for multidimensional data analysis on key-value databases. In *Proceedings of the 17th International Conference on Distributed Computing and Networking - ICDCN' 16*. ACM Press.
- Dehdouh, K (2016a). Building OLAP Cubes from Columnar NoSQL Data Warehouses. In *Model and Data Engineering*, L. Bellatreche, Ó. Pastor, J.M. Almendros Jiménez, Y. Ait-Ameur (eds.). vol. 9893, Springer International Publishing.
- Dehdouh, K (2016b). Building OLAP Cubes from Columnar NoSQL Data Warehouses. In *Model and Data Engineering*, L. Bellatreche, Ó. Pastor, J.M. Almendros Jiménez, Y. Ait-Ameur (eds.). vol. 9893, Springer International Publishing.
- Dehdouh, K, F Bentayeb, O Boussaid, and N Kabachi (2014). Columnar NoSQL CUBE: Agregacion operator for columnar NoSQL data warehouse. At *2014 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. IEEE.
- Du, D (2018). *Apache Hive Essentials: Essential Techniques to Help You Process, and Get Unique Insights from, Big Data, 2nd Edition*. 2nd ed., Packt Publishing.
- Duan, L, and LD Xu (2012). Business Intelligence for Enterprise Systems: A Survey. *IEEE Transactions on Industrial Informatics*, 8(3), 679–687.
- Ferreira, LM (2015). Modelo de Processo para Criação de BI em Banco de Dados NoSQL Orientado a Colunas. In *Conferencia Ibero Americana WWW/Internet - CIAWI*. vol. 1, IADIS.
- Hewitt, E, and J Carpenter (2020). *Cassandra: The Definitive Guide*. 3rd ed., O'Reilly Media, Inc.
- Inmon, WH (2002). *Building the Data Warehouse*, 3rd Edition. 3rd ed., John Wiley & Sons, Inc.
- Kimball, R, and M Ross (2002). *The data warehouse toolkit: the complete guide to dimensional modeling*. 2nd ed, Wiley.
- Lins, MTS, and LM Ferreira (2016). Estudo de Caso de Processamento de ETL em Plataforma Big Data. In *Conferências Ibero-Americanas WWW/Internet e Computação Aplicada 2016*. vol. 1, IADIS.
- Lóscio, BF, H de OLIVEIRA, and J de S PONTES (2011). NoSQL no desenvolvimento de aplicações Web colaborativas. *VIII Simpósio Brasileiro de Sistemas Colaborativos*, 10(1), 11.
- Moniruzzaman, ABM, and S Hossain (2013). NoSQL Database: New Era of Databases for Big data Analytics - Classification, Characteristics and Comparison. *Int J Database Theor Appl*, 6.
- Murazza, MuhR, and A Nurwidiantoro (2016). Cassandra and SQL database comparison for near real-time

- Twitter data warehouse. In the *2016 International Seminar on Intelligent Technology and Its Applications (ISITIA)*. IEEE.
- O'neil, P, B O'neil, and X Chen (2009). The Star Schema Benchmark (SSB).
- O'Neil, P, E O'Neil, X Chen, and S Revilak (2009). The Star Schema benchmark and augmented fact table indexing. Springer.
- Scabora, LC, JJ Brito, RR Ciferri, and CD de Aguiar Ciferri (2016a). Physical Data Warehouse Design on NoSQL Databases OLAP Query Processing over HBase. S. Hammoudi, L. Maciaszek, M. Missikoff, O. Camp, J. Cordeiro (eds.).
- Scabora, LC, JJ Brito, RR Ciferri, and CD de Aguiar Ciferri (2016b). Physical Data Warehouse Design on NoSQL Databases OLAP Query Processing over HBase. S. Hammoudi, L. Maciaszek, M. Missikoff, O. Camp, J. Cordeiro (eds.).
- Tournier, MCEMKT (2017a). *Document-Oriented Data Warehouses: Complex Hierarchies and Summarizability*. Springer Singapore.
- Tournier, MCEMKT (2017b). *Document-Oriented Data Warehouses: Complex Hierarchies and Summarizability*. Springer Singapore.
- V, Manoj (2014). Comparative Study of NoSQL Document, Column Store Databases and Evaluation of Cassandra. *International Journal of Database Management Systems*, 6(4), 11–26.
- Vandewalle, B, J Beirlant, A Christmann, and M Hubert (2007). A robust estimator for the tail index of Pareto-type distributions. *Computational Statistics & Data Analysis*, 51(12), 6252–6268.
- Vasilakis, N, Y Palkhiwala, and JM Smith (2017). Query-efficient Partitions for Dynamic Data. In *Proceedings of the 8th Asia-Pacific Workshop on Systems*. ACM.
- Yessad, L, and A Labiod (2016). Comparative study of data warehouses modeling approaches: Inmon, Kimball and Data Vault. At *2016 International Conference on System Reliability and Science (ICSRS)*. IEEE.
- Zhao, H, and X Ye (2014). A Practice of TPC-DS Multidimensional Implementation on NoSQL Database Systems. In *Performance Characterisation and Benchmarking*, R. Nambiar, M. Poess (eds.). vol. 8391, Springer International Publishing.
- Zhu, Q, and H Xiang (2016). Differences of Pareto principle performance in e-resource download distribution: An empirical study. *The Electronic Library*, 34(5), 846–855.