Physical Data Warehouse Design on NoSQL Databases
OLAP Query Processing over HBase

Lucas C. Scabora[1], Jaqueline J. Brito[1], Ricardo Rodrigues Ciferri[2]
and Cristina Dutra de Aguiar Ciferri[1]
[1]*Department of Computer Science, University of Sao Paulo at Sao Carlos, 13.560-970, Sao Carlos, SP, Brazil*
[2]*Department of Computer Science, Federal University of Sao Carlos, 13.565-905, Sao Carlos, SP, Brazil*

Keywords: Data Warehousing, Physical Design, NoSQL, OLAP Query Processing, HBase, Star Schema Benchmark.

Abstract: Nowadays, data warehousing and online analytical processing (OLAP) are core technologies in business intelligence and therefore have drawn much interest by researchers in the last decade. However, these technologies have been mainly developed for relational database systems in centralized environments. In other words, these technologies have not been designed to be applied in scalable systems such as NoSQL databases. Adapting a data warehousing environment to NoSQL databases introduces several advantages, such as scalability and flexibility. This paper investigates three physical data warehouse designs to adapt the Star Schema Benchmark for its use in NoSQL databases. In particular, our main investigation refers to the OLAP query processing over column-oriented databases using the MapReduce framework. We analyze the impact of distributing attributes among column-families in HBase on the OLAP query performance. Our experiments showed how processing time of OLAP queries was impacted by a physical data warehouse design regarding the number of dimensions accessed and the data volume. We conclude that using distinct distributions of attributes among column-families can improve OLAP query performance in HBase and consequently make the benchmark more suitable for OLAP over NoSQL databases.

1 INTRODUCTION

The comparison among different systems that manipulate huge volumes of data is crucial for modern information systems. Performing analysis over massive volumes of data is a challenge for traditional data warehousing approaches (Chevalier et al., 2015). Data warehouses (DWs) are used for data analysis, in which the data is modeled in a multidimensional schema according to the cube metaphor and on-line analytical processing (OLAP) queries are performed to help the decision-making process. New solutions for big data management are usually implemented on distributed environments, which enables horizontal scalability. Many enterprises use NoSQL (*Not only SQL*) database systems to manage data split in decentralized environments. With the advent of NoSQL systems to store and process data, there is a need to apply systematic techniques for performance comparison, usually conducted by benchmarks.

Benchmarks of DW are tools aimed at answering the question "*Which is the best database system for OLAP query processing?*" (Folkerts et al., 2012). These questions are answered by functional and performance tests, based on properties of each evaluated system. Its goal is to quantify the quality and the performance of a system, in order to make a fair comparison. There are four main requisites of a benchmark (Bog, 2013): relevance, portability, scalability and simplicity. In the context of DW and decision support systems, there are three main benchmarks: TPC-DS (Poess et al., 2002), TPC-H (Moussa, 2012) and Star Schema Benchmark (SSB) (O'Neil et al., 2009). However, they fail in, at least, two requisites. First, they were planned to evaluate relational databases, which can be very different from NoSQL systems, failing on portability. Second, their data generation is centralized and limited, impacting the horizontal scalability of huge data volumes.

There has been a significant amount of work on column-oriented database systems (Abadi et al., 2008). Studies revealed that this type of data storage could support analytical workloads with more than an order of magnitude faster than row-oriented database systems. The performance improvement is related to read-only workload, which reduces the number of I/O operations since most of the queries have to read only the targeted attributes. Based on the as-

Scabora, L., Brito, J., Ciferri, R. and Ciferri, C.
Physical Data Warehouse Design on NoSQL Databases - OLAP Query Processing over HBase.
In *Proceedings of the 18th International Conference on Enterprise Information Systems (ICEIS 2016) - Volume 1*, pages 111-118
ISBN: 978-989-758-187-8

sumption that, for some specific queries, not all of the values are required at the same time, the column-oriented approach is appropriate for deploying a DW. HBase (George, 2011) is a distributed, persistent and strictly consistent column-oriented NoSQL database system. All data stored in HBase is organized in column-families, as described in Section 2.2. As proposed by Cai et al. (2013), there are an important feature that impact the performance of HBase's read and write operations: the attributes stored in the same or among different column-families.

In this paper, we propose the analysis of OLAP query performance over multiple data organization strategies on HBase. These different strategies refer to different physical DW designs. The present investigation consists in implementing and comparing the performance of OLAP queries over different column-families arrangements. We also highlight scenarios based on different report requirements that could benefit from the designs investigated in the paper. Since a benchmark encompasses the schema and workload of a DW, we tackle the problem by exploring different schemas and analyzing the effects on their workloads.

1.1 Motivating Scenarios

Let a DW storing data related to a shopping corporation represented by the multidimensional data cube. This company is interested in reporting the quantity sold per product per filial per day. OLAP queries issued against this DW depend on the business perspectives of interest. Therefore, we introduce two representative scenarios that motivate the investigations carried out in this paper, as follows:

Scenario 1: The shopping corporation is interested in reporting the daily profit, based on the total of products sold in that day. With this information, the corporation can determine the total income of each month. Other possibility is that the enterprise is focused on the daily profit to perform infrastructure investments. In this scenario, the most frequent OLAP queries access only one dimension of the data cube. We call these queries as one-dimensional queries.

Scenario 2: The corporation is interested in analyzing the amount of units sold of each product over time. More specifically, it is focused on reporting the quantity sold of each product in the last month, or the quantity sold of each product in each filial. In this scenario, the most used queries involve two or more dimensions of the data cube. We call these queries as two-dimensional and three-dimensional queries.

1.2 Contributions

The relevance of our paper is to point out appropriate data organization designs to enhance the performance of OLAP queries in a column-based NoSQL database for different DW enterprise scenarios. We investigate the influence of physical design of the DW schema so that databases administrators can optimize the performance of OLAP queries on distributed column-oriented NoSQL database systems. This paper introduces the contributions described as follows:

1. It proposes a new physical DW design, called FactDate, aimed to improve the performance of one-dimensional queries.
2. It analyses three physical DW designs, each one providing better performance results according to a given scenario. This analysis includes a scalability performance evaluation.
3. It extends the SSB workload by proposing two new OLAP queries, which are used to investigate two-dimensional queries.

The remaining part of this paper is organized as follows. Section 2 summarizes the background, Section 3 reviews related work, Section 4 describes the physical DW designs, including the proposed FactDate design, Section 5 details the queries proposed for SSB, Section 6 addresses the experimental tests, and Section 7 concludes the paper.

2 BACKGROUND

2.1 Data Warehouse and OLAP

Business Intelligence (BI) can be defined as a set of technologies and mechanisms to efficiently extract useful business information from large volumes of data. Nowadays, DWs are inserted in many business information technology applications, enabling the effectively utilization and analysis of information for business planning and decision making (Ciferri et al., 2013). A DW stores data used for analytical tasks to support decision making, such as information about sales, customers and profit. DWs are typically related to the day-to-day company's operations, and can contain millions and even billions of business records.

Following the cube metaphor described in Section 1.1, the DW provides a multidimensional organization of data. When a DW is implemented in relational databases, this organization is usually structured as a star schema, where the fact table stores the measures of the business events and the dimension tables, related to the fact tables, contain the con-

text of these measures (Kimball and Ross, 2013). Star schemas provide better query performance by reducing the number of join operations.

While a DW is considered one of the most used infrastructure for BI systems, OLAP can be interpreted as a front-end analyzing tool. OLAP encompasses complex queries that frequently aggregate and consolidate measures of the fact table, including dimensions perspectives. An example of an OLAP query is: "*How many products were sold by brand and by store in the last year?*", where the measure is the quantity of items sold and the dimensions are brand, stores and date, respectively. In this context, the word analytical refers to extracting information from the DW that is useful to the decision making process, focusing on the analyses of the day-to-day company's operations.

2.2 Column-oriented NoSQL Databases

Column-oriented databases store their data grouped by columns, where the values of each column are contiguously stored on disk. This orientation differs from the row-oriented databases, which store each row entirely and contiguously on disk (George, 2011). As stated in Section 1, an advantage of column-oriented organization is the need of reading only the required attributes of the query, which is the case for OLAP. In a column-oriented database, inserting a tuple requires to write each attribute value of the record separately, raising the number of I/O operations. However, column-oriented storage can improve the performance of queries that access only a subset of columns from a wide table. This occurs because unrequired attributes are not read, reducing the I/O consumption of read-intensive workloads, such as OLAP queries.

An example of column-oriented NoSQL database is HBase, which provides access to each tuple using unique keys called *rownum*. These keys are stored lexicographically. For each attribute of each tuple, HBase stores a cell structure with the following format: <*rownum, cf, column, timestamp, value*>. To retrieve an attribute value, it is required to inform the *rownum*, *cf* (column-family) and *column* fields. Column-families group columns that are stored continuously on disk, in the same file, whose structure is denominated HFile. If a query processes attributes from different column-families, the needed HFiles are joined to reconstruct the query result.

To perform OLAP queries over distributed data, the HBase tables can be used as input to the MapReduce or the Spark frameworks, which are designed for parallel processing of massive datasets (Doulkeridis and Nørvåg, 2014). Companies can implement their own queries using these frameworks, or can integrate HBase with some SQL layers, such as Hive and Phoenix. Hive (Thusoo et al., 2010) is a SQL layer that models an infrastructure of DW, allowing queries to be expressed with a SQL-like language called HiveQL. Phoenix, on the other hand, offers another SQL interface, boosting HBase performance.

2.3 Benchmarking Technique

A benchmarking technique aims to measure performance of an information system and compare it with others. This evaluation consists of performing a set of well-defined tests in order to empirically measure the system's functionality and performance. Moreover, the benchmark must contain a set of operations based on the workload scenario that is going to be tested. Regarding SQL statements, the OLAP query processing can be classified as a read-only workload, focused on select transactions (Bog, 2013).

Regarding benchmarks for DWs, they should encompass four main steps (Floratou et al., 2014): *(i)* schema and workload; *(ii)* data generation; *(iii)* metrics; and *(iv)* validation. In step *(i)*, two issues must be tackled. First, a schema that models a typical application in the domain area of the DW. For the workload, it refers to operations on this schema, represented by OLAP queries with respect to variations of the selectivity. In step *(ii)*, the benchmark defines rules to generate synthetic or real data for the schema, allowing data volumes variations and respecting the selectivity of the workload. In step *(iii)*, some quantitative and qualitative metrics are defined for the benchmark, to report important aspects of the system. Finally, in step *(iv)*, metrics are collected after the workload is applied on the generated data. These metrics are compared to others reported by others databases systems.

3 RELATED WORK

The Star Schema Benchmark (SSB) (O'Neil et al., 2009) is an extension of the TPC-H (Poess and Floyd, 2000), which is designed to measure the performance of analytical queries over database products in support to typical data warehousing applications. SSB implements a genuine star schema, which previous work (Kimball and Ross, 2013; O'Neil et al., 2009) argued that this kind of schema can better represent real-world scenarios. The central fact table of SSB is *LineOrder*, which contains information about sales transactions of a retailer, and this information is stored as different types of measures, like profit, units sold and revenue. Also, SSB defines four dimension tables: *Date*, *Customer*, *Supplier* and *Part*. The SSB

workload is composed of 13 OLAP queries organized in four classes that provide not only functional coverage but also variations of selectivity and hierarchical level used. The main limitation of SSB is the restriction to relational OLAP environments. Further, it also defines queries applied over one, three or four dimension tables, lacking of two-dimensional queries.

The Columnar NoSQL Star Schema Benchmark (CNSSB) (Dehdouh et al., 2014) extends SSB by proposing a benchmark adapted to measure the performance of columnar NoSQL DWs. It defines a schema composed of only one table with several columns, i.e. it denormalizes the SBB's star schema by joining the fact table (*LineOrder*) with the dimensions tables. The attributes of the schema are grouped in column-families (CFs) related to their original dimension. Although this adaptation has an intuitive semantic, the physical design of the schema can influence query performance, and therefore this schema may not be the only one recommended.

Related to the performance evaluation of columnar NoSQL, the work of Cai et al. (2013) measures the HBase performance by using two physical DW designs: (i) only one column-family with multiple columns; and (ii) multiple column-families, where every column-family has only one column. In case (i), reading one row means reading the data of all columns, even if the user does not need them. As a consequence, queries using only a few attributes should consume more I/O and bandwidth. On the other hand, in case (ii), the user needs to read the data separately from each CF and combine them to rebuild the row. When the user request data from a fixed set of columns, storing these specific columns in the same CF should provide a better processing performance than splitting them into several different CFs. The experiments only performed generic read and write operations, without evaluating any aspect of query processing, selectivity, and DW schema.

Another extension of SSB to column-oriented NoSQL databases is proposed by Dehdouh et al. (2015). They introduce three approaches to adapt SSB to HBase, called NLA-SSB, DLA-SSB, and DLA-CF-SSB. NLA-SSB refers to the normalized approach of SSB, while DLA-SSB and DLA-CF-SSB join the fact and the dimension tables. While DLA-SSB groups all dimensions in the same CF, DLA-CF-SSB stores each dimension in a distinct CF, such as CNSSB (Dehdouh et al., 2014). They observed that both DLA-SSB and DLA-CF-SSB did not impact query performance when accessing attributes from different dimensions. However, the experiments only processed a fixed data volume, not analyzing the behavior of query processing as data grow. They also did not evaluate the performance of all SSB's queries, which vary the selectivity and perform aggregations based on real-case enterprise scenarios.

In this section, we addressed the limitations of SSB, and its adaptations, to analyze different physical DW designs on NoSQL databases. In this paper, we tackle these issues by measuring the impact of the CF organization on OLAP queries, considering different data volumes. Further, we propose two new types of queries, which are very important because they allow the investigation of two-dimensional queries.

4 PROPOSED INVESTIGATION

We propose an investigation of the physical DW design on HBase column-oriented NoSQL database, by considering different strategies to arrange attributes into column-families (CFs). Our investigation is motivated by the fact that, as presented in Section 2.2, each CF on HBase is stored in a separated HFile, such that when a query accesses data from two or more CFs, it must read each HFile and join them by the *rownum* field to rebuild the tuple. However, recommendations of HBase state that joining more than two CFs leads to a low performance. As a consequence, it is important to analyze the attribute distribution over CFs regarding the OLAP query context. Depending on the analysis, specifically the number of dimensions aggregated, distinct distributions can benefit or degenerate query performance. Further, different enterprise scenarios may benefit from different physical DW designs. As described in Section 1.1, our work focuses on two enterprise scenarios. Figure 1 depicts the three-level architecture for NoSQL column-oriented databases adopted in our work, showing the adaptations for the physical level regarding distributed NoSQL systems.

By adopting this denormalization, we argue that, at the physical level, the database administrator can decide among three DW designs, as follows:

- Physically organize all attributes in the same CF, as proposed by Cai et al. (2013) and presented as DLA-SSB (Dehdouh et al., 2015). We call this schema as **SameCF**.
- Store each dimension in different CFs, regarding CNSSB (Dehdouh et al., 2014) and DLA-CF-SSB (Dehdouh et al., 2015). We call this schema as **CNSSB**.
- Group some of the more frequently used dimensions to the fact table, which represents the new strategy proposed in this paper. We joined dimension *Date* and call this schema as **FactDate**.

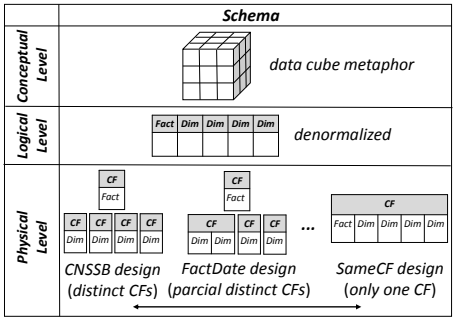


Figure 1: Three-level DW architecture for NoSQL column-oriented databases, with specific adaptations on the physical level that are analyzed in this paper.

First, we implement the CNSSB, which was proposed by Dehdouh et al. (2014) and described in Section 3. Despite the fact that CNSSB is a user understandable organization, any OLAP query that involves at least one dimension of the DW will need to process two CFs, one for the fact's attributes and one for the aggregated dimensions. Our second implementation refers to the SameCF design and consists in storing all the attributes in a single CF, based on the DLA-CF-SSB (Dehdouh et al., 2015) and on the work of Cai et al. (2013). This configuration aims to improve the performance of OLAP queries that analyze a greater number of dimensions, such as Scenario 2 (Section 1.1). Although minimizing the quantity of CFs is a good approach, the organization may decrease the performance of OLAP analysis that uses just a small set of attributes and dimensions.

Finally, we propose a new physical DW implementation, called the FactDate design, which represents an intermediary solution between CNSSB and SameCF, combining the *LineOrder* and *Date* tables. This design is aimed at improving the performance of OLAP queries that use both the fact table and the dimension table *Date*, as most of analytical queries relate their measures to time. This physical design can improve query performance for queries that use both CFs as those described in Scenario 1, i.e. one-dimensional queries. It can also benefit two-dimensional queries, where one of the dimensions used is the dimension *Date*, by decreasing the total number of CFs processed in the query.

5 PROPOSED QUERIES

Another contribution of this paper is the proposal of two new queries to be added to the SSB workload. The need for those queries is to test OLAP aggregations over two dimensions since the SSB workload lacks this type of query. Therefore, the proposed queries are two-dimensional queries. To elaborate queries for a data warehousing benchmark, first we need to determine and vary the selectivity of the queries. The predicates used by OLAP queries defined in SSB have an uniform distribution. Through the cardinality of the attributes, we can define new queries with specific selectivities, similar to the other queries defined by SSB, however performing a two-dimensional analysis. Table 1 defines the cardinality of the predicates used in the proposed queries.

Table 1: Attribute's cardinality of the proposed queries.

attribute	value	attribute	value
d_year	7	*l_quantity*	50
d_yearmonthnum	84	*p_category*	25

The first proposed query, named **Qnew1** (Figure 2), calculates the maximum and minimum revenue, for a given product's category and year, grouped by product's brand. The predicates of this query are defined over the attributes *d_year*, *p_category* and *lo_quantity*, whose combined selectivity is $\frac{1}{7} \times \frac{1}{25} \times \frac{25}{50} = 2.85 \times 10^{-3}$. This value is similar by the same order of magnitude to the queries defined by SSB using other quantities of dimensions, like *Q2.2* (1.60×10^{-3}) and *Q3.2* (1.37×10^{-3}).

```
SELECT p_brand1, max(lo_revenue),
       min(lo_revenue)
FROM lineorder, dates, part
WHERE lo_orderdate = d_datekey
      AND lo_partkey = p_partkey
      AND d_year = 1993
      AND p_category = 'MFGR#11'
      AND lo_quantity < 25
GROUP BY p_brand1 ORDER BY p_brand1;
```

Figure 2: The two-dimensional proposed query **Qnew1**.

The second proposed query is based on a *drill-down* operation over the date hierarchy, changing the analysis from year to month (i.e. attributes *d_year* to *d_yearmonthnum*). This new query, called **Qnew2**, calculates the maximum and minimum revenue for a given product's category and a month of a year, grouped by product's brand. The selectivity of this query is 2.38×10^{-4}, because of the cardinality of the predicate over the attribute *d_yearmonthnum*. Also, this query has the selectivity near to the SSB's queries *Q1.2* (6.49×10^{-4}) and *Q2.3* (2×10^{-4}).

The two new queries are used in the performance evaluation to test the designs detailed in Section 4.

6 PERFORMANCE EVALUATION

In this section, we present the performance evaluation for the three implemented physical designs, CNSSB,

SameCF and FactDate, regarding the execution of the queries of the SSB workload and the queries proposed in Section 5. We also investigate the impact of the data volume scalability. The tests were performed using the following configuration setup:

Hardware: A *cluster* composed of 4 nodes, each node having a quad-core CPU at 3.0 Ghz (i5-3330), 16 GB RAM, 1 TB SATA disk (7200 RPM) and 1 Gb/s network. One node acts only as a dispatcher (*namenode*) and the other three as workers (*datanodes*).

Software: All machines run CentOS (version 7.0). Storage and query processing were performed using HBase (version 0.98.13), Hadoop (version 2.4.1) and ZooKeeper for data partitioning in HBase.

We used the SSB's data generator to generate the dataset, in which the tables were joined into a single denormalized CSV file. This file was first loaded in HBase through conversion to HFile format and then equally distributed among the *datanodes* using MapReduce jobs. Table 2 details, for each Scale Factor (SF), the size of the generated file and the size of the same data after loading it in HBase. Because of the HBase's cell structure, the size of the database is greater than the CSV file size. Each query was implemented using Java (version 1.8) and executed at least five times to collect the average elapsed time.

Table 2: Data volumes used in the experiments.

	Scale Factor (SF)			
	10	**20**	**40**	**80**
CSV (GB)	28	55	109	218
FactDate (GB)	58.4	116.9	233.9	468.2
CNSSB (GB)	58.5	117	234.2	469.9
SameCF (GB)	57.9	115.8	231.8	464.0

6.1 Analyzing the Physical Designs

This experiment evaluates the query processing for the three schemas described in Section 4 using SFs with values of 10 and 20 (Table 2). We organize our discussions considering two aspects: low-dimensional analysis and high-dimensional analysis.

Analysis of Low-dimensional Queries

Regarding Scenario 1, we evaluated OLAP queries involving a few number of dimensions, i.e. we evaluated one and two-dimensional queries. The one-dimensional queries were adapted versions of the SSB query workload. These queries, named *Q1.1*, *Q1.2* and *Q1.3*, depend only on the dimension *Date* and on the fact table. Their differences consist in the predicates involved, which provided different values of selectivity. The two-dimensional queries were the **Qnew1** and **Qnew2** proposed in Section 5.

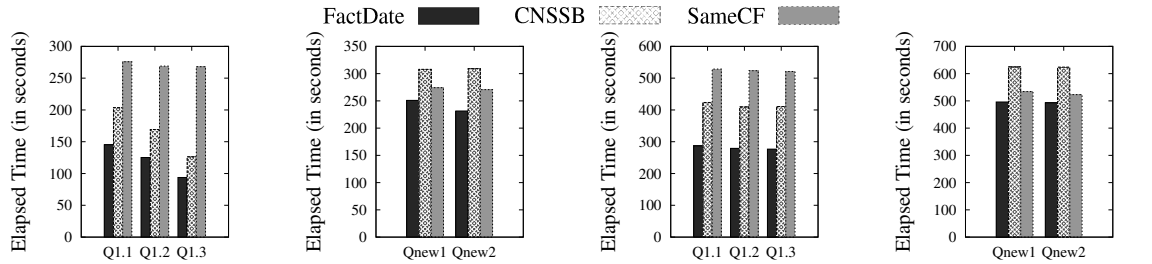
Figure 3 depicts the obtained performance results. Figures 3(a) and 3(c) show that the proposed FactDate outperforms the other designs for one-dimensional queries. FactDate improved the overall performance from 25% to 33% regarding its best competitor, CNSSB. This behavior is justified by the fact that our proposed design processes only one CF, while CNSSB processes two CFs to perform the same query. When comparing the SameCF and the FactDate designs, SameCF contains flatter HFiles because it stores attributes for all dimensions. As a consequence, the one-dimensional query processing requires more time due to larger HFiles. Figures 3(b) and 3(d) illustrate that CNSSB demanded more time to process queries **Qnew1** and **Qnew2**, as it requires accessing three CFs to process them. Also, our proposed FactDate design still outperformed the other designs because it uses only two CFs. Compared to its best competitor, SameCF, FactDate improved query performance from 6% to 14%.

Analysis of High-dimensional Queries

The second part of our analysis evaluated three and four-dimensional queries, which were adapted versions of the SSB workload. These queries are related to Scenario 2, and adapted from SSB's workload. The three-dimensional queries are named *Q2.1* to *Q3.4*, and the four-dimensional queries, *Q4.1*, *Q4.2* and *Q4.3*, accessed all four dimensions of the schema.

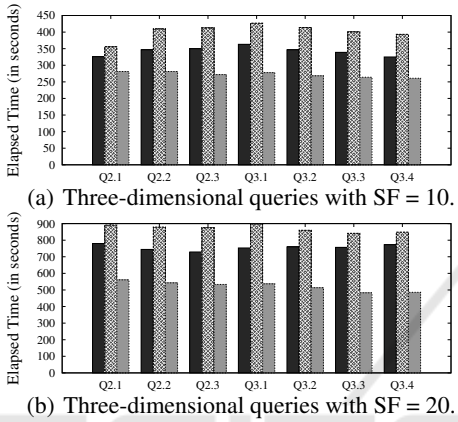
Figure 4 depicts the obtained processing elapsed time. Here, we noticed that processing more than two CFs in the same query provided significant performance losses regarding the FactDate and CNSSB designs. When a query needs to process three or more CFs, the overhead for rebuilding a tuple sharply increased execution time. Figures 4(a) and 4(b) illustrate that SameCF provided the better performance results. They also show that this behavior was maintained when the data volume was increased by two times. Regarding the three-dimensional queries, SameCF improved the overall performance from 14% to 38% when compared to its best competitor, FactDate. Furthermore, when we added more dimensions to the query, the processing time for CNSSB and FactDate increased. They become unsuitable for Scenario 2, as depicted in Figures 4(c) and 4(d). Regarding the four-dimensional queries, the improvement provided by SameCF over FactDate ranged from 47% to 54%.

Both analysis show strong indications about how different enterprise scenarios can require distinct physical DW designs to efficiently attend the most frequent queries. On our next experiment, we analyze how this behavior is related to data volume.



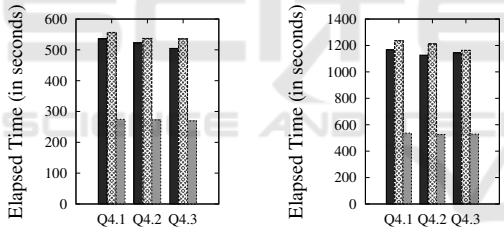
(a) One-dimensional queries with SF = 10. (b) Two-dimensional queries with SF = 10. (c) One-dimensional queries with SF = 20. (d) Two-dimensional queries with SF = 20.

Figure 3: Processing elapsed time for low-dimensional queries, which are related to Scenario 1.



(a) Three-dimensional queries with SF = 10.

(b) Three-dimensional queries with SF = 20.



(c) Four-dimensional queries with SF = 10. (d) Four-dimensional queries with SF = 20.

Figure 4: Processing elapsed time for high-dimensional queries, which are related to Scenario 2.

6.2 Scalability Evaluation

Here, we evaluate query performance, according to the designs described in Section 4, analyzing their behavior as the data volume increases. In this experiment, we used SFs = 10, 20, 40 and 80 (Table 2).

Figure 5 depicts the average processing time for the one-dimensional queries. We can observe that the query performance against SameCF was highly degenerated and produced higher processing times when compared to the other designs. The main difference between FactDate and CNSSB was related to the quantity of CFs, where the queries against FactDate only accessed one CF while the queries against CNSSB accessed two CFs. Regarding FactDate, we observed that reducing the quantity and size of the CFs related to the most frequent queries improved query performance, but if all the dimensions were joined in the same CF, the performance dropped substantially. Further, FactDate boosted the performance by 20% on average when compared to CNSBB.

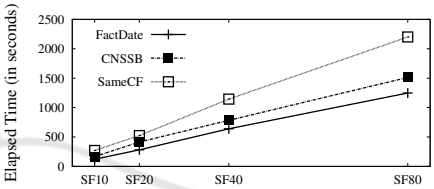


Figure 5: Processing time of one-dimensional queries.

Figure 6 depicts the average processing time for the two-dimensional queries. The three designs showed a similar behavior when processing these queries. However, the proposed FactDate design slightly outperformed the other designs as it processed only two CFs for the two-dimensional queries, while CNSSB design processed three CFs. Comparing FactDate to SameCF, we observed that FactDate deals with two small CFs while SameCF processes only one large CF. Therefore, when two small CFs were joined, FactDate still outperformed the overhead of processing one flatter HFile. This improvement varied from 6% to 11% as the data volume increased.

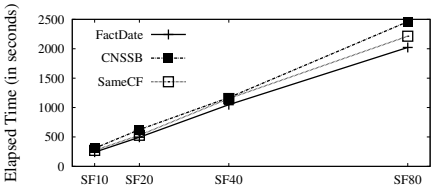
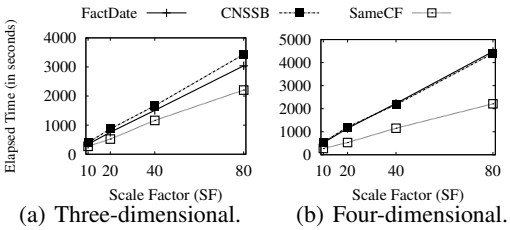


Figure 6: Processing time of two-dimensional queries.

Figure 7(a) shows the average processing time for the three-dimensional queries. For two or more CFs, query performance presents an opposite behavior when compared to the results depicted in Figures 5 and 6. We can observe that grouping all attributes in the same CF boosted the performance of SameCF

over FactDate from 21% to 31% as the data volume increased. Moreover, Figure 7(b) depicts that this improvement, on four-dimensional queries, is up to 54%. We can conclude that, when it comes to processing three or four CFs, FactDate and CNSSB are not suitable for Scenario 2.



(a) Three-dimensional. (b) Four-dimensional.

Figure 7: Processing time of high-dimensional queries.

7 CONCLUSIONS

In this paper, we analyze three physical DW designs, called CNSSB, SameCF, and FactDate. We consider two different enterprise scenarios, determining OLAP queries with different numbers of dimensions. We observe how the attribute arrangement over CFs according to these designs influences OLAP query performance. The results of our experiments showed that storing all data in one CF provided better performance for high-dimensional queries. In this scenario, the SameCF was the most appropriated to be deployed. On the other hand, storing dimensions in different CFs benefited low-dimensional queries. In this scenario, the FactDate and the CNSSB were more appropriated. Further, when processing one-dimensional queries that required data from the dimension *Date*, the FactDate design provided the best performance results. Since data warehousing is characterized by mostly read-only operations, this organization in CFs is an important issue to take into account when comparing NoSQL column-oriented databases.

By using this guideline, the company is able to provide a schema physical design that best suits the most frequent OLAP queries issued against its data warehousing application. Regarding benchmarks, we can conclude that their workload must model different physical designs in order to provide a more accurate evaluation focused on the company interests.

ACKNOWLEDGEMENTS

This work has been supported by the following Brazilian research agencies: FAPESP (Grant: 2014/12233-2), FINEP, CAPES and CNPq.

REFERENCES

Abadi, D. J., Madden, S. R., and Hachem, N. (2008). Column-stores vs. row-stores: How different are they really? In *ACM SIGMOD*, pages 967–980, NY, USA.

Bog, A. (2013). *Benchmarking Transaction and Analytical Processing Systems: The Creation of a Mixed Workload Benchmark and Its Application*. Springer Publishing Company, Incorporated, 1 edition.

Cai, L., Huang, S., Chen, L., and Zheng, Y. (2013). Performance analysis and testing of hbase based on its architecture. In *12th IEEE/ACIS ICIS*, pages 353–358.

Chevalier, M., El Malki, M., Kopliku, A., Teste, O., and Tournier, R. (2015). Implementing Multidimensional Data Warehouses into NoSQL. In *ICEIS*.

Ciferri, C., Ciferri, R., Gómez, L., Schneider, M., Vaisman, A., and Zimányi, E. (2013). Cube algebra: A generic user-centric model and query language for olap cubes. *IJDWM*, 9(2):39–65.

Dehdouh, K., Bentayeb, F., Boussaid, O., and Kabachi, N. (2015). Using the column oriented NoSQL model for implementing big data warehouses. *PDPTA'15*, pages 469–475.

Dehdouh, K., Boussaid, O., and Bentayeb, F. (2014). Columnar NoSQL star schema benchmark. In *MEDI 2014*, pages 281–288.

Doulkeridis, C. and Nørvåg, K. (2014). A survey of large-scale analytical query processing in mapreduce. *The VLDB Journal*, 23(3):355–380.

Floratou, A., Özcan, F., and Schiefer, B. (2014). Benchmarking sql-on-hadoop systems: TPC or not tpc? In *5th WBDB*, pages 63–72.

Folkerts, E., Alexandrov, A., Sachs, K., Iosup, A., Markl, V., and Tosun, C. (2012). Benchmarking in the cloud: What it should, can, and cannot be. In *4th TPCTC*, pages 173–188.

George, L. (2011). *HBase: The Definitive Guide*. O'Reilly Media, 1rd edition.

Kimball, R. and Ross, M. (2013). *The Data Warehouse Toolkit: The Definitive Guide to Dimensional Modeling*. Wiley Publishing, 3rd edition.

Moussa, R. (2012). Tpc-h benchmark analytics scenarios and performances on hadoop data clouds. In *NDT*, volume 293, pages 220–234.

O'Neil, P., O'Neil, E., Chen, X., and Revilak, S. (2009). The star schema benchmark and augmented fact table indexing. In *TPCTC*, pages 237–252.

Poess, M. and Floyd, C. (2000). New TPC benchmarks for decision support and web commerce. *SIGMOD Record*, 29(4):64–71.

Poess, M., Smith, B., Kollar, L., and Larson, P. (2002). TPC-DS, taking decision support benchmarking to the next level. In *SIGMOD Conference*, pages 582–587.

Thusoo, A., Sarma, J. S., Jain, N., Shao, Z., Chakka, P., Zhang, N., Anthony, S., Liu, H., and Murthy, R. (2010). Hive - a petabyte scale data warehouse using hadoop. In *26th ICDE*, pages 996–1005.