A Performance Analysis for Efficient Schema Design in Cloud-Based Distributed Data Warehouses

Fred E. R. Rabelo Ferreira[a] and Robson do Nascimento Fidalgo
Center of Informatics (CIn), Federal University of Pernambuco (UFPE), Recife, PE, Brazil

Keywords: Data Warehouse, Distributed SQL, NewSQL, HTAP Databases, Data Modeling, Performance Analysis.

Abstract: Data Warehouses (DWs) have become an indispensable asset for companies to support strategic decision-making. In a world where enterprise data grows exponentially, however, new DW architectures are being investigated to overcome the deficiencies of traditional relational Database Management Systems (DBMS), driving a shift towards more modern, cloud-based DW solutions. To enhance efficiency and ease of use, the industry has seen the rise of next-generation analytics DBMSs, such as NewSQL, a hybrid storage class of solutions that support both complex analytical queries (OLAP) and transactional queries (OLTP). We understand that few studies explore whether the way the data is denormalized has an impact on the performance of these solutions to process OLAP queries in a distributed environment. This paper investigates the role of data modeling in the processing time and data volume of a distributed DW. The Star Schema Benchmark was used to evaluate the performance of a Star Schema and a Fully Denormalized Schema in three different market solutions: Singlestore, Amazon Redshift and MariaDB Columnstore in two different memory availability scenarios. Our results show that data denormalization is not a guarantee for improved performance, as solutions performed very differently depending on the schema. Furthermore, we also show that a hybrid-storage (HTAP) NewSQL solution can outperform an OLAP solution in terms of mean execution time.

SCITEPRESS SCIENCE AND TECHNOLOGY PUBLICATIONS

1 INTRODUCTION

In today's data-driven world, where the amount of enterprise data has increased exponentially, efficient processing of vast volumes of information has become paramount for organizations seeking to gain insights and make informed decisions (Dumbill, 2013). In this sense, Data Warehouse (DW) has become an indispensable asset for decision support, being the environment responsible for the storage, processing and analysis of large volumes of data (Murphy, 1988). However, the increasing volume of data produced today makes the successful implementation of a DW increasingly challenging, and thus practitioners are looking into new ways of modernizing their current Data Warehousing installations.

Recently, advances in cloud computing have introduced tools that simplify the creation, setup, and scaling of a DW with the emergence of Databases as a Service (DBaaS), which allows for straightforward deployment of Database Management Systems (DBMS) in the cloud. Furthermore, the usage of a DbaaS exempts the user from upfront investments and allows cost optimization, since many DBaaS solutions provide dynamic cloud resource allocation (Idrissi, 2016). Such technology may help businesses that cannot afford expensive DW infrastructures to setup their first analytical application using cloud infrastructure as a service, i.e., a scalable Data Warehouse as a Service (DWaaS) (Krishnan, 2013).

Along with the move from on-premise infrastructure to a cloud infrastructure, new technologies have emerged to substitute traditional Relational Database Management Systems (RDBMS), which typically do not scale horizontally, to a more modern DBMS that supports massively parallel data processing (MPP) and scale natively in a cloud environment. A traditional RDBMS stores data in a row-by-row manner, making it ideal for Online Transactional Processing (OLTP) workloads. In the rowstore structure, entire rows are physically stored together, facilitating efficient insert, update, and delete operations that require access to all fields of a record (Abadi, 2008).

However, to efficiently support Online Analytical Processing (OLAP) workloads, a DBMS will typically use a columnstore, which organizes data by

[a] https://orcid.org/0000-0002-1419-5043

Ferreira, F. and Fidalgo, R.
A Performance Analysis for Efficient Schema Design in Cloud-Based Distributed Data Warehouses.
DOI: 10.5220/0012546200003690
Paper published under CC license (CC BY-NC-ND 4.0)
In *Proceedings of the 26th International Conference on Enterprise Information Systems (ICEIS 2024) - Volume 1*, pages 39-49
ISBN: 978-989-758-692-7; ISSN: 2184-4992
Proceedings Copyright © 2024 by SCITEPRESS – Science and Technology Publications, Lda.

columns, allowing efficient access to specific columns without the need to scan entire rows (Abadi, 2008). This structure offers substantial benefits for a DW, as it enables compression and allows for scanning and aggregating specific columns without accessing unnecessary data, ultimately improving query performance (Jaecksch, 2010). Since 2012, cloud providers such as Snowflake (Snowflake, 2023) and Amazon Web Services (AWS) have begun to offer cloud-based columnar DW solutions with services like Snowflake and Amazon Redshift (Pandis, 2021).

More recently, a new class of DBMS is gaining recognition in the realm of data analytics and business intelligence. Although traditionally designed for OLTP (Stonebraker, 2011) (Grolinger, 2013), NewSQL DBMSs are increasingly demonstrating their ability in handling OLAP queries because of their horizontal scalability, data distribution mechanisms and specially designed storages built to efficiently support Hybrid Transactional/Analytical Processing (HTAP) (Pavlo, 2017): depending on the solution, it can have both in-memory or disk-based rowstore and duplicated columnstore data, or it can have hybrid column-row stores.

Regardless of how the data is physically stored, there are several approaches to how the data can be modeled in an analytical database. For this study, we highlight the Star Schema (SS) and the denormalized Flat Table (FT). SS is a type of database schema where a central fact table is connected to one or more dimension tables through foreign key relationships, allowing for complex queries across multiple dimensions at the cost of the execution of join operations (Kimball and Ross, 2013). On the other hand, FT consolidates all data into a single table, completely eliminating the need for joins, which is why it is commonly known to provide better query performance (G. Lawrence Sanders, 2002). It comes, however, at the cost of increased storage and maintenance complexity due to data redundancy. Snowflake schema was not considered for this study because, as cited in (Kimball, 2002): "Disk space savings gained by further normalizing the dimension tables typically are less than 1% of the total disk space needed for the overall schema".

The Star Schema is a firmly established schema that the community widely embraces for DW modeling (Adamson, 2006). However, the suitability of this schema is unquestionable in the context of a traditional DW architecture within a centralized relational database environment, which scales vertically. We are now interested in understanding whether the suitability of SS still holds when transitioning to a cloud-based environment characterized by parallel and distributed processing. In this new environment, the expense associated with performing join operations within a distributed setup might outweigh the benefits of dealing with a larger volume of redundant data in a distributed Flat Table.

This hypothesis carries particular significance because, in a horizontally scalable DW within a cloud environment, during periods of peak demand, one would typically encounter a cluster comprising many nodes, each with restricted memory capacity to maintain cost efficiency, rather than a single server boasting substantial CPU and memory resources. We understand that there is a lack of methodological approaches in the schema design for distributed Data Warehousing. Therefore, there are questions that remain fairly unanswered by the scientific community and practitioners: Is a multidimensional design such as a star schema a suitable design pattern for a distributed DW? How does the increase in volume caused by data redundancy in FT affect the performance of a distributed DW cluster? How can the availability of memory at nodes affect data processing? How does a NewSQL HTAP DBMS perform compared to solutions built specifically for OLAP?

To answer these questions, an experimental study was conducted by executing the Star Schema Benchmark (SSB) (O'Neil, 2009) and its 13 queries in two data schemas: A Star Schema (SS) and a completely denormalized Flat Table Schema. Three distributed DBMSs were chosen for the experiment: Singlestore, Amazon Redshift and MariaDB Columnstore, specifically a hybrid store (favoring HTAP) and two columnar stores (favoring OLAP). Since rowstores do not favor OLAP queries, no rowstore DBMS was selected. The solutions were chosen based on the following criteria: higher position in the ranking of the 'DB-engines' website; the number of mentions on websites and interests in Google Trends searches; the solution must have a DBaaS offering of the product; and it must provide a free license or free credits to run the tests in the DBaaS.

This work aims to contribute to the literature by performing an evaluation and discussion of adequate data modeling, especially data denormalization, for distributed data warehousing. For each of the solutions, a detailed performance analysis was conducted to analyze their strengths and weaknesses in SS and FT. Furthermore, we also present a comparison of the overall performance to complete the entire workload in order to conclude which solution provides the best efficiency for OLAP.

The rest of this article is organized as follows: Section 2 presents related works. Section 3 describes the design of the experiment, methods used and some theoretical foundation. Section 4 presents the three experiments performed, along with a detailed analysis of the star and flat schema design's performance for each selected solution. Section 5 presents a comparative analysis of the overall performance between the three DW solutions. Finally, conclusion and future work are presented in Section 6.

2 RELATED WORK

We started our literature review by searching for studies related to Data Warehousing that carried out at least one experimental performance analysis, preferably using a benchmark specific for OLAP, and studies that analyzed different data modeling techniques and data schemas using one or multiple relational DBMS, NoSQL or NewSQL solutions.

We found several studies that analyze the performance of a DW using relational DBMSs. The study by (Almeida et al., 2015) performs an experiment to compare two relational DBMS to process OLAP queries using the Star Schema Benchmark (SSB): MySQL InnoDB and Microsoft SQL Server 2012 (MSSQL), in a variety of dataset sizes: 1GB, 3GB, 6GB, 12GB and 24GB, representing a fact table with 6M, 18M, 36M, 71M and 144M rows respectively. The author concludes that MSSQL outperforms MySQL in general, especially for the larger datasets, while MySQL has acceptable performance only for workloads up to 6GB, mainly as a result of the columnstore storage in MSSQL in comparison to MySQL InnoDB, which uses a rowstore storage. In (S Ilić, 2022), the author provides a thorough investigation of rowstores versus columnstores using a variety of different RDBMS: HP Vertica, Microsoft SQL Server, Oracle, MariaDB ColumnStore and MySQL, concluding that all queries were executed in an acceptable time of under 10 seconds, except for the databases using rowstore (MySQL), meaning columnstore-based DBMSs are highly recommended for OLAP, but rowstores are not.

We also found a few studies that analyze the performance of a DW using newer technologies, such as NewSQL, and compare it with relational DBMSs. The work of (Murazzo et al., 2019) analyzes the performance of Google Cloud Spanner, a NewSQL solution, and compares it with MySQL, both running on the Google Cloud Platform (GCP). The author uses the year 2018 from a public historical database of the City of Buenos Aires as a workload, containing 895,000 lines, and saves the average time to execute three different SQL queries. The results obtained showed that Cloud Spanner achieved a lower execution time than MySQL in all scenarios and tests. However, the author points out that for query 3 (the most complex query, as it has a higher number of joins), Spanner's performance is very close to that of MySQL, leading to a preliminary conclusion that the greater the complexity, or the higher the number of joins, the greater the tendency of the two DBMSs to converge towards a greater and equivalent execution time.

In (Oliveira, 2017), the authors conduct experiments on MemSQL (currently Singlestore) and VoltDB using the TPC-H benchmark in SF = 1 (6 million rows, 1GB of size) and on a single machine with 40GB of RAM. The study shows that MemSQL achieves better performance in both load time and average query time, saving 92% of the average time spent compared to VoltDB. According to the authors, the reason for MemSQL's better performance when executing queries is due to its in-memory cache storage, offering time savings for reexecutions. To analyze a distributed DW, the setup of the experiment, however, is not ideal, as only one node is used in the cluster, and the data volume using SF = 1 is too small a volume to simulate a realistic DW environment (data always fit in memory).

In a study by (Costa et al., 2017), the authors investigate the role of data modeling in the processing times of big data DWs implemented using two SQL-on-Hadoop systems: Hive, an open-source, distributed NoSQL DBMS by Apache, and Engine-X (real name omitted due to licensing limitations), an enterprise NoSQL system. The authors specifically benchmark multidimensional star schemas and fully denormalized tables, investigating how data denormalization and partitioning affects the performance of Hive on Tez using the SSB with different Scale Factors: SF = 10, SF = 30, SF = 100 and SF = 300 in a Hadoop cluster with 5 nodes, each with 32GB of RAM. The authors conclude that the usage of the star schema for DWs on Hive may not be the most efficient design pattern: despite saving a significantly larger amount of data (SS was 3x smaller than the FT), Hive still favors the denormalized schema with faster query execution, less memory requirements, and less intensive CPU usage. This conclusion arises directly from evaluating the effort required by the solution to handle data redundancy in comparison to the effort needed to manage join operations in a distributed cluster with 5 nodes. The conclusion, however, cannot be extended to other classes of DWs, since the authors only experiment with SQL-on-hadoop systems.

In a more recent study, in (Eduardo Pina, 2023) the authors used OSSpal methodology to analyze the performance of three NewSQL databases, CockroachDB, MariaDB Xpand and VoltDB in the context of DW using the SSB in SF = 1 and SF = 10, respectively 1GB and 7GB of data. The authors conclude that the DBMS with the best score was MariaDB Xpand, followed by VoltDB and CockroachDB. The authors also showed that VoltDB has better performance in terms of query execution time, as it is an in-memory database. On the other hand, CockroachDB showed better performance when loading data and lower consumption of resources such as CPU and RAM. However, the setup of the experiment is not ideal, as the volume of data used fits entirely in the main memory of the server. In addition to that, the work does not analyze data distribution, since only one node with 28GB of RAM is used in the experiment's cluster. Furthermore, the study does not analyze the impact of the denormalization of the schema in terms of data volume and query performance, nor does it include HTAP NewSQL DBMSs, such as Singlestore. In our extensive literature review, we also did not find papers that specifically evaluated the performance of NewSQL solutions provided as cloud services or DW solutions provided specifically as DWaaS.

3 EXPERIMENTAL DESIGN AND METHODS

This section describes the materials and methods used in this research. Since this paper discusses some best practices for data modeling, we were careful to provide detailed information on our setup and guidelines in order to support experiment replication.

3.1 Star Schema Benchmark

In order to maintain the same process across different setups, we looked at the literature to find a suitable benchmark for our study. There are several benchmark approaches for DW and OLAP systems, with TPC-H(Serlin, 1998), TPC-DS (Nambiar and Poess, 2006) and SSB (O'Neil, 2009) being the most widely accepted by the industry. We chose to continue our experiment using SSB because it is based on the Star Schema, a prevalent structure in many business intelligence and data warehousing systems. Unlike TPC-H, which simulates a broad spectrum of ad hoc business queries, and TPC-DS, which spans a diverse range of decision support functions, SSB offers a simpler but focused assessment for dimensional models based on the TPC-H.

The Star Schema Benchmark is composed of a fact table (Lineorder) and four dimension tables (Supplier, Customer, Part and Date). The data from SSB can be obtained in different scale factors (SF), which determine the amount of data in the fact table and dimension tables, with the exception of the Date table, which has 2556 records in any SF, representing an interval of 7 years. The SSB also provides 13 SQL queries involving common operations used in DW such as drill down, roll up, slice and dice. These queries are classified into four groups that differ from each other in the number of joined tables and in the complexity of the queries:

- Group 1: Queries Q1.1, Q1.2 and Q1.3 restrict revenue, with different ranges and filter factors, to find potential revenue increases. Typically, they calculate the increase in revenue that would result from eliminating certain discounts by a certain percentage for products shipped in a given year.
- Group 2: Queries Q2.1, Q2.2, and Q2.3 restrict the data in two dimensions, comparing revenues for all orders in all years for suppliers in a given region and for a given class of products.
- Group 3: Queries Q3.1, Q3.2, Q3.3 and Q3.4 restrict data in three dimensions, calculating revenue volume over a given period of time by customer country, supplier country, and year within a given region.
- Group 4: Queries Q4.1, Q4.2 and Q4.3 represent a "What-If" sequence. It starts with query Q4.1 using a group by in two dimensions and weak constraints in three dimensions to measure aggregate profit.

3.2 Process

We used the DBGEN software to generate SSB data at the SF = 10 and SF = 50 Scale Factors, where the Lineorder table has approximately 60 million and 300 million rows respectively. The FT schema was created by joining the Star Schema fact table with its dimension tables, discarding foreign keys, and adjusting field names accordingly. As a result, we ended up with a copy of the SSB data organized as a flat table. In addition, we adapted all thirteen queries provided by SSB to fit the new fully denormalized table, this time without the joins, providing the same results as the original SSB dimension and fact tables.

The first stage of the experiment was the creation of the DBMS cluster in the selected solution using credits or a test account provided by the DBaaS. Once the cluster is online, for each solution, we create the SSB tables, first in SS and then in FT. All data (SS and FT) were previously generated and stored in an

Amazon S3 bucket as a .CSV file, to be then loaded into the DBMS through either an import operation or an import pipeline. Once we have the cluster running and the tables created and populated, the experiment is started, one solution at a time and one configuration at a time.

A Python script (code available publicly) was created to run each of the 13 SSB queries 60 times, saving data volume and mean execution time for each execution (Rabelo Ferreira, 2023). We were careful to drop the cache and memcache before each run. The number of runs for each query was chosen according to the central limit theorem, so that we had a sample large enough to obey a normal distribution. Each indicator was saved along with its mean, standard deviation and confidence interval (p=0.95). Once all the queries are executed, we drop the tables, drop the database, and do the whole process again for a new data schema, a new scale factor, and then the next solution to be analyzed.

The proposed experiment aims to analyze the performance of each DBMS in its offer as DBaaS in the cloud, that is, in the standard configurations offered by each supplier. We standardized the clusters in all solutions to contain exactly three distributed nodes, each node with 16GB of RAM. On Singlestore, the DBaaS offering for 16GB of RAM has 2vCPUs. In MariaDB Columnstore, the DBaaS offering for 16GB of RAM has 4vCPUs. In Amazon Redshift, the DbaaS for 16GB of RAM has 2vCPUs but uses 1GB of RAM for internal processing, resulting in an actual 15GB of RAM availability. All solutions were hosted on the Amazon AWS cloud at Region 'us-east-1' (North Virginia).

The usage of 16GB nodes in the cluster was purposely chosen by the author because it is the smallest amount of memory available across the different DBaaS offerings in the market today and represents a real scenario where practitioners will select the cheapest possible infrastructure for the DW in idle time and scale out the number of nodes as the necessity arrives (peak in demand). Furthermore, we specifically designed the experiment to test a scenario in which the volume of the denormalized table does not fit the total memory available in the cluster (SF = 50). In a cluster with 3 nodes, which is used in our experiments, the total amount of memory available is 3 x 16GB = 48GB, forcing the DW to perform main and secondary memory operations along with data distribution to provide the results. These scenarios were chosen because FT is commonly known to provide better query performance since it eliminates the joins of a query (G. Lawrence Sanders, 2002).

4 EXPERIMENTS

This section presents the experiments performed for each DBMS solution selected, along with its results and a thorough analysis comparing the performance of a DW built with a design using the star schema and one with a fully denormalized schema. In some cases, relevant variables such as data volume and memory consumption are also analyzed if pertinent to understand the solution's behavior and performance. The results depicted in this section are relevant to highlight several factors that data managers and engineers must take into account when designing a distributed DW.

Before the experiments began, all the SSB data was loaded into the DBaaS. Tables 1 and 2 below show the total amount of data of the denormalized flat table and the SS's fact table for both Singlestore and Redshift. Due to limitations in the DbaaS, it was not possible to gather data volume for columnstore tables in MariaDB Columnstore in the SkySQL offering. We also present the data compression ratio of the data in each solution compared to the total amount of data organized as a tab-separated .CSV file, along with the total amount of time in seconds to transfer and load the data from Amazon S3 to the solution.

Table 1: Comparison of Fact table (SS) and Flat Table (FT) data volume and data dump elapsed time - SF = 10.

SF - Schema	SF10 - SS		SF10 - FT	
Solution	**Singlestore**	**Redshift**	**Singlestore**	**Redshift**
Volume	3.0GB	2.4GB	8.5GB	12.5GB
Compression Ratio	1.86	2.33	3.04	2.07
Data loading time	477.6	111.5s	2296.9s	674.9s

Table 2: Comparison of Fact table (SS) and Flat Table (FT) data volume and data dump elapsed time - SF = 50.

SF - Schema	SF50 - SS		SF50 - FT	
Solution	**Singlestore**	**Redshift**	**Singlestore**	**Redshift**
Volume	18.7GB	12.7GB	66.8GB	62.7GB
Compression Ratio	1.51	2.23	1.94	2.07
Data loading time	1766.5	546.7s	9246.4s	2313.1s

An analysis of the results shows that, with the exception of FT in SF = 10, Redshift has a higher compression ratio than Singlestore, especially for multidimensional schemas, meaning that its algorithms and storage structure presents an advantage in data compression compared to Singlestore. For SF = 10, Redshift data volume is 20% smaller in SS but 47% larger in FT. For SF = 50, Redshift data volume is 32% smaller in SS and 6% smaller in FT. Regarding the total amount of time to load the data in the solution, Redshift has a clear advantage over Singlestore, probably due to the overload of work necessary to store the data in its hybrid structure to support HTAP: For SF = 10, Redshift loads the data 77% faster than Singlestore in SS and 70% faster in FT. For SF = 50, Redshift loads the data 69% faster than Singlestore in SS and 75% faster in FT.

4.1 Experiment 1: Singlestore

SingleStore (formerly known as MemSQL) is a distributed database focusing on high performance in both transactional and analytical workloads. It can be categorized as a ***Single-Store Architecture HTAP database with Primary Columnstore + In-memory Rowstore***. Officially launched in 2013, it was one of the first general-purpose databases, introducing a proprietary storage engine called S2DB, which combines in-memory rowstore and on-disk columnstore, as well as vectorization techniques, columnar scanning using secondary indexes, among others (Prout, 2022).

The Singlestore structure is composed of S2DB clusters, which have two layers: aggregator nodes, which function as SQL query routers and act as a gateway in the distributed system, and leaf nodes, which are responsible for the distributed and parallel execution of queries, as well as aggregate the result and return it to the customer. Singlestore uses a shared-nothing architecture, with each node in the cluster responsible for a subset of the data. Communication with the client is carried out via SQL commands under a MySQL protocol (Geomar Schreiner, 2019). Last checked in August 2023, the service has three plans: Standard, whose cost starts from $0.65 per hour; Premium, starting at $1.30 per hour; and Dedicated, a unique service for on-demand projects in which the price is based on a dedicated budget. It is also possible to use Singlestore in Self-hosted mode, where the customer hosts and manages his own system (Singlestore, 2023).

Table 3 shows the results of the average execution time for different scale factors, SF = 10 and SF = 50 using Singlestore, with lower values highlighted in bold and followed by a (-); higher values are followed by a (+). According to our experiment, FT outperformed SS in all but one query (Q1.3) in SF = 50. An explanation for this behavior is the fact that group 1 of queries only joins the fact table with the date dimension. Considering that the denormalized flat table is 3.5 times larger than the normalized fact table in SF = 50, there is an overload of work to handle the redundancy of the data in the flat table in which is equivalent to perform the single join operation.

Table 3: SSB average execution times (in miliseconds) on Singlestore: star schema (SS) and flat table (FT).

Query	**SF = 10**		**SF = 50**	
	SS	FT	SS	FT
Q1.1	226 (+)	**136** (-)	793 (+)	**561** (-)
Q1.2	185 (+)	**143** (-)	654 (+)	**539** (-)
Q1.3	159 (+)	**126** (-)	**505**(-)	509 (+)
Q2.1	245 (+)	**144** (-)	1145 (+)	**582** (-)
Q2.2	221 (+)	**149** (-)	1014 (+)	**583** (-)
Q2.3	331 (+)	**124** (-)	1183 (+)	**452** (-)
Q3.1	316 (+)	**177** (-)	1194 (+)	**711** (-)
Q3.2	246 (+)	**169** (-)	983 (+)	**721** (-)
Q3.3	168 (+)	**115** (-)	763 (+)	**356** (-)
Q3.4	158 (+)	**115** (-)	544 (+)	**338** (-)
Q4.1	381 (+)	**181** (-)	1598 (+)	**740** (-)
Q4.2	433 (+)	**1205** (-)	1489 (+)	**848** (-)
Q4.3	299 (+)	**155** (-)	1031 (+)	**915** (-)
TOTAL	3368 (+)	**1939** (-)	12896 (+)	**7855-**
Comparative %		**-42%**		**-39%**

The experiment carried out shows that for Singlestore, the denormalization in the FT scheme brings a clear performance improvement to DW, with an overall performance advantage of 42% in SF = 10 and 39% in SF = 50 compared to SS (meaning that FT can complete the workload up to 42% faster than SS).

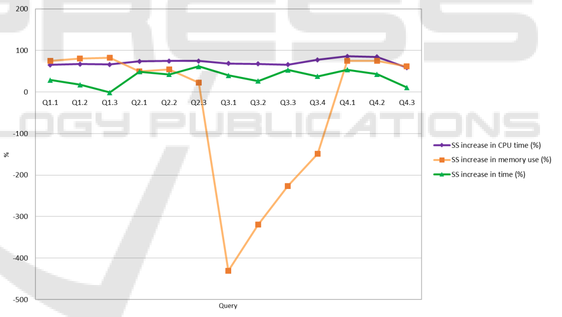

Figure 1: SS increase in execution time, memory usage and CPU time when compared to FT in %; SF = 50.

Figure 1 compares the percentage of increased memory use and CPU time in addition to the query execution time discussed above. It is possible to see that SS has much higher CPU time (71% on average) than FT. Regarding RAM memory consumption, however, we notice that SS's increase in memory usage for query group 3 has fallen below zero, meaning that FT has used much more memory than SS to execute all four queries of group 3, on average 281% more memory. This might be explained by the fact that these queries calculate revenue volume over a given period of time, dealing with even more data than group 4 of queries, which emulates a "what-if" sequence. For all other queries, with the exception of group 3, SS has increased memory consumption compared to FT.

SCITEPRESS SCIENCE AND TECHNOLOGY PUBLICATIONS

4.2 Experiment 2: Amazon Redshift

Launched in 2013 by Amazon Web Services (AWS), it is considered the first 100% cloud-managed enterprise quality DW, focused on massively parallel data processing (MPP) and columnar storage. Redshift brought simplicity and better cost-effectiveness to the market, introducing the concept of DWaaS to the world (Pandis, 2021). As of August 2023, Redshift is offered exclusively as a cloud-based DBaaS, fully managed by the solution and hosted on AWS. The service can be scaled manually and automatically, depending on the required infrastructure (size of infrastructure and cluster nodes) and the necessary availability zone, with costs starting at $0.25 per hour per node. The Redshift cluster architecture is based on a shared-nothing strategy, layered and composed of a coordinator node, called the leader node, and multiple computing nodes. Data is stored in Redshift Managed Storage, located in Amazon S3, cached on compute nodes on SSDs local to the DBMS and compressed in columnar format. Tables are replicated on each node, or partitioned across multiple S3 buckets and distributed across nodes (Pandis, 2021).

An analysis of Table 4 shows that in group 1 of queries (Q1.1, Q1.2, and Q1.3), both in SF = 10 and SF = 50, FT outperforms SS in Redshift, even with an increase in data volume in FT due to data redundancy (the flat table is four to five times bigger than the normalized fact table in Redshift). One can conclude that, for this solution in group 1, the smaller size of SS does not compensate for the overhead of performing the join operation, in which is preferable to handle the larger volume of data in FT. In further analysis of query group 1, we can notice that queries Q1.1, Q1.2 and Q1.3 only filters columns from the Lineorder and Date tables, not using any VARCHAR columns from the Part, Customer or Supplier tables. Because Redshift uses columnar storage, the solution does not need to retrieve the entire distributed flat table to return the results, thus resulting in a better performance in FT. As the queries become more complex, however, we can see that there is a reversal in the performance of FT compared to SS.

Table 4: SSB average execution times (in miliseconds) on Redshift: star schema (SS) and flat table (FT).

Query	SF = 10		SF = 50	
	SS	FT	SS	FT
Q1.1	140 (+)	**85** (-)	630 (+)	**382** (-)
Q1.2	116 (+)	**67** (-)	522 (+)	**297** (-)
Q1.3	108 (+)	**68** (-)	478 (+)	**294** (-)
Q2.1	**266** (-)	1003 (+)	**1142** (-)	4917 (+)
Q2.2	**226** (-)	984 (+)	**1530** (-)	4909 (+)
Q2.3	**210** (-)	804 (+)	**878** (-)	4006 (+)
Q3.1	**438** (-)	1587 (+)	**2360** (-)	8002 (+)
Q3.2	**257** (-)	1557 (+)	**1099** (-)	7874 (+)
Q3.3	**214** (-)	798 (+)	**970** (-)	3953 (+)
Q3.4	**202** (-)	1165 (+)	**902** (-)	5711 (+)
Q4.1	**491** (-)	1393 (+)	**2654** (-)	7029 (+)
Q4.2	**452** (-)	1558 (+)	**1996** (-)	7893 (+)
Q4.3	**320** (-)	1554 (+)	**1284** (-)	7846 (+)
TOTAL	**3440** (-)	12623 (+)	**16345** (-)	63113 (+)
Comparative %	**-72%**		**-74%**	

In all remaining 20 distinct queries in groups 2, 3 and 4, SS outperformed FT in both SF = 10 and SF = 50. The overall performance advantage for the star schema (SS) using Redshift is 72% in SF = 10 and 74% in SF = 50 (meaning SS can complete the workload up to 74% faster than FT). A contributing factor to the advantage of SS in SF = 50 is the fact that it is the workload in which the total size (62.7GB) of the denormalized table (FT) does not fit in the total amount of memory available for querying in the cluster (3 x 16 GB = 48GB), meaning the solution has to perform swaps between main and secondary memory, along with handling the distribution of data in the nodes of the cluster.

Unlike Singlestore, which has hybrid storage for HTAP operations, the optimized columnar storage of Redshift, the distributed allocation of data in the cluster, and the selectivity of SSB's query constraints to filter data selection may be responsible for these results, in which it is preferable to perform the join operations than to handle the larger volume of data in the FT, i.e. data denormalization does not improve the overall performance of a distributed DW in Redshift.

4.3 Experiment 3: MariaDB Columnstore

Released in 2016, MariaDB ColumnStore is a distributed DBMS that provides high performance in executing distributed queries, especially for analytical (OLAP) queries that involve scanning large datasets. Although it also has rowstore storage, its architecture is especially focused on columnar storage (defined when creating the table), which was built by porting InfiniDB 4.6.7 to MariaDB Server. Its architecture is composed of several MariaDB servers, operating as modules and working together to offer linear scalability and high performance using a shared-nothing architecture.

MariaDB Columnstore is offered in three modalities: MariaDB Community Server, an open source solution that can be managed and hosted by the customer in their own cloud, MariaDB Enterprise, a dedicated solution for scaled production environments, with DBMS management and support provided by MariaDB, and SkySQL. SkySQL is a DbaaS solution offering that automates the deployment and sizing of MariaDB Columnstore and MariaDB Xpand DBMS in the cloud, in an environment fully managed by the solution and with costs starting at $0.1318 per hour.

The analysis of mean execution time for MariaDB Columnstore presents, again, a curious scenario, as shown in table 5: In group 1 (Q1.1, Q1.2 and Q1.3), where the queries are simpler (group 1 only joins the fact table with the date dimension), both in SF = 10 and SF = 50, FT outperforms SS, even though there is an increase in data volume in FT due to redundancy. Again, we conclude that the columnstorage architecture of MariaDB Columnstore favors FT as it does not need to touch the entire distributed flat table to return the results, but only a few columns. As the queries become more complex, however, we can see that there is a reversal in the performance of FT compared to SS.

Table 5: SSB average execution times (in miliseconds) on MariaDB Columnstore: star schema (SS) and flat table (FT).

Query	SF = 10		SF = 50	
	SS	FT	SS	FT
Q1.1	568 (+)	**389** (-)	2202 (+)	**1067** (-)
Q1.2	261 (+)	**226** (-)	1122 (+)	**675** (-)
Q1.3	248 (+)	**233** (-)	1089 (+)	**698** (-)
Q2.1	**2468** (-)	4163 (+)	**8735** (-)	10226 (+)
Q2.2	**1957** (-)	4279 (+)	**7111** (-)	10375 (+)
Q2.3	**1191** (-)	4078 (+)	**4163** (-)	9952 (+)
Q3.1	**2856** (-)	6362 (+)	**13615** (-)	15931 (+)
Q3.2	**1815** (-)	5886 (+)	**7781** (-)	14796 (+)
Q3.3	**1174** (-)	7989 (+)	**6478** (-)	19906 (+)
Q3.4	**1010** (-)	8763 (+)	**3769** (-)	21901 (+)
Q4.1	**3300** (-)	9237 (+)	**14883** (-)	22504 (+)
Q4.2	**1968** (-)	2720 (+)	7725 (+)	**6789** (-)
Q4.3	**1465** (-)	1868 (+)	5699 (+)	**4753** (-)
TOTAL	**20281** (-)	56193 (+)	**84372** (-)	139573 (+)
Comparative %	**-63%**		**-39%**	

Regarding the remaining 20 query executions compared in groups 2, 3 and 4, SS outperformed FT in 18 queries, with the exception only of Q4.2 and Q4.3 in SF = 50, where FT outperforms SS by a slight advantage (12% faster in Q4.2 and 16% faster in Q4.3). The overall performance advantage, however, for the star schema (SS) using MariaDB Columnstore is 63% in SF = 10 and 39% in SF = 50 (meaning SS can complete entire workload up to 63% faster than FT). Just like Redshift, the optimized architecture of MariaDB Columnstore, which uses a columnar storage only, makes it preferable to perform the join operations along with the constraints in groups 2, 3, and 4 of queries, rather than handle the larger volume of data distributed in the cluster. One can conclude that the denormalization of data does not improve the overall performance of the distributed DW in MariaDB Columnstore.

5 HORIZONTAL ANALYSIS OF RESULTS

This section presents a comparative analysis of the experiments and its results presented in the previous section. Taking into account the different architectures of the selected DBaaS solutions and different behavior in the test scenarios, a comparative analysis of performance is presented. The results depicted in this section are relevant to highlight some of the variables that CTOs, data managers, and/or data engineers must take into consideration when selecting a cloud solution for a distributed DW.

Figures 2 and 3 compare, respectively, for SF = 10 and SF = 50, the percentage increase in the execution time of SS in relation to FT for each of the solutions studied. It is easy to notice that Singlestore is the only solution with results above zero, meaning that FT is faster than SS. For MariaDB Columnstore and Redshift, we can notice that both favor SS, as SS actually decreased the execution time. We also notice that Redshift, with the exception of Q3.3 and Q3.4 in SF = 10, has even higher performance increase when us-

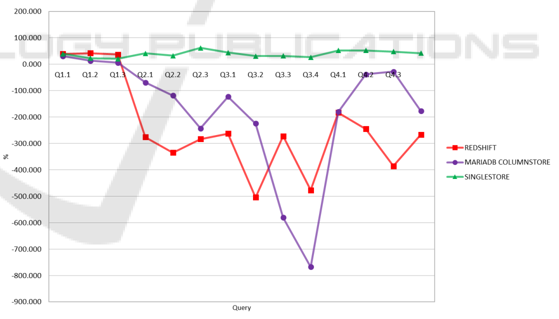

Figure 2: SS increase in execution time when compared to FT in %, per solution; SF = 10.

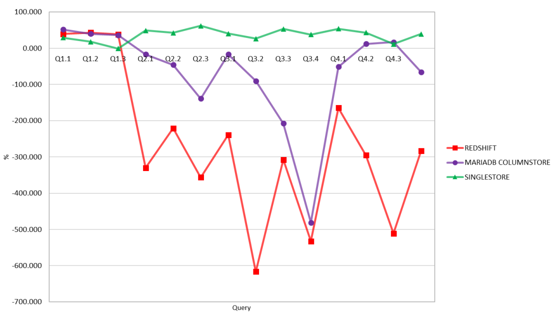

Figure 3: SS increase in execution time when compared to FT in %, per solution; SF = 50.

SCITEPRESS SCIENCE AND TECHNOLOGY PUBLICATIONS

ing SS instead of FT compared to MariaDB Columnstore. One can notice that queries Q3.3 and Q3.4 are the ones with higher selectivity.

Figures 4 and 5 show, respectively for SF = 10 and SF = 50, the overall performance depicted earlier in a comparative scenario per solution and per data schema. It is possible to notice that in both schemes (SS and FT) and in both scale factors, Singlestore has a superior performance, followed by Amazon Redshift and then by MariaDB Columnstore. This result shows that, despite having a hybrid structure to perform OLAP and OLTP queries, and hypothetically an overload of functions in the structure to maintain row and columnar storage, Singlestore's total execution time to conclude the workload was still advantageous compared to other solutions, except only in the comparison of Singlestore's SS schema to Redshift's SS schema in SF = 10, which have equivalent performance considering the confidence interval of the results.

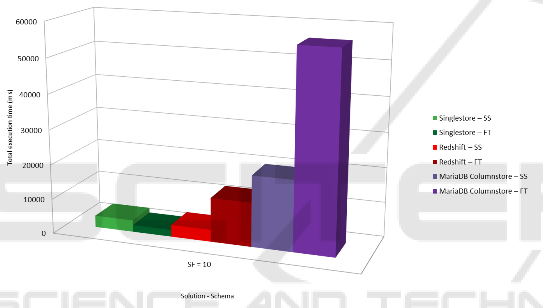

Figure 4: Total execution time (milliseconds) by scheme and solution; SF = 10.

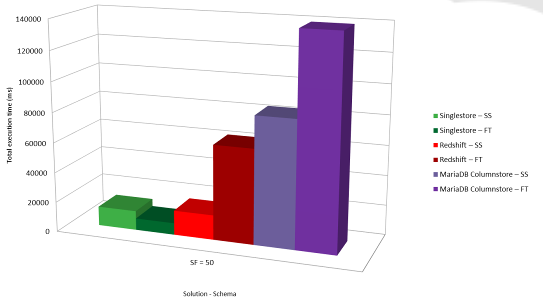

Figure 5: Total execution time (milliseconds) by scheme and solution; SF = 50.

In SF = 10, the overall performance of Singlestore (considering SS and FT) is 67% faster than Redshift, which is 78% faster than MariaDB Columnstore. In SF = 50, the overall performance of Singlestore is 73% faster than Redshift, which is 64% faster than MariaDB Columnstore. In a further analysis of the order of performance in figures 4 and 5, we can notice the clear difference in the execution process and in the architecture of Singlestore compared to Redshift and MariaDB Columnstore, with a clear optimization in SS compared to FT in the last two. This evidence suggests that, contrary to the common belief that denormalization is a guarantee of performance improvement, there are scenarios in which SS will perform better than FT, even with the necessity to perform join operations in a distributed DW.

6 CONCLUSION

Data modeling and performance tuning are the most difficult aspects of data management. It is especially challenging in the context of a distributed DW: Data is no longer centralized and limited to the company's OLTP systems, being now highly distributed, with different structures, and growing at an exponential rate, resulting in a variety of factors that may influence a DW's performance.

This work presented an evaluation and discussion of adequate data modeling strategies for data warehousing using three different DBaaS solutions: Singlestore, a hybrid store that favors HTAP and two columnar stores that favor OLAP: Amazon RedShift and MariaDB Columnstore. The SSB was used to evaluate the performance of a star schema and a fully denormalized schema in a distributed DW. Two scale factors were used: SF = 10, with 60 million rows (a volume where the data fit in memory both in SS and FT), and SF = 50, with 300 million rows (a volume where the data in FT do not fit in the total available memory of the cluster). The same cluster infrastructure (3 nodes and 16GB of RAM available in the nodes) was used across the solutions. For each solution, a detailed performance evaluation was performed to analyze the advantage of data denormalization in their design. Finally, we compared the overall performance of the DW to complete the entire workload in order to conclude which one delivers the most value.

Regarding the comparison between a DW built using a star schema and a DW built using a fully denormalized schema, this paper concludes that the denormalization of data is not a guarantee of performance improvement in a columnar distributed data warehouse. In the results achieved in the benchmark for Redshift and MariaDB Columnstore, SS outperformed FT respectively by 74% and 63%, meaning the star schema executes the workload considerably faster than the denormalized table for these solutions, even with the necessity to perform join operations. On the other hand, the results of the benchmark ex-

SCITEPRESS
SCIENCE AND TECHNOLOGY PUBLICATIONS

ecuted using Singlestore showed that FT completes the workload 42% faster than SS, meaning that denormalization, for this solution, can improve performance, even with the redundancy of data generated by the FT.

One can notice great disparities in the results of the experiment as a result of the difference in the architecture of the solutions. While Singlestore offers a single all-purpose solution focused on HTAP (transactional and analytical operations altogether), Redshift and MariaDB Columnstore are characterized as solutions focused specifically on OLAP. Surprisingly, our results show that a NewSQL HTAP solution, which would supposedly have an overhead of functions to handle both workloads, can have equal or superior performance in comparison to a solution specifically focused on OLAP workloads, given the NewSQL DW is built with a schema design using a fully denormalized table.

Regarding the comparison of performance between solutions, Singlestore appears highest in the rank, followed by Redshift and then by MariaDB Columnstore. Singlestore presents better results in terms of fastest total query execution time to complete the workload. Redshift, however, has better data compression and faster data loading time from .CSV files.

As future work, we intend to analyze in more detail the opposite scenario analyzed by this paper: instead of verifying the performance of data denormalization in the flat table, we intend to analyze the performance of OLAP queries in HTAP databases that claim no ETL process is needed, i.e., we can run analytics using the same normalized data schema designed for transactional workloads. We also intend to carry out experiments varying the number of nodes in the cluster and with a larger number of NewSQL DBaaS solutions.

DECLARATION OF COMPETING INTEREST

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

DATA AVAILABILITY

Data will be made available on request.

TRADEMARKS

All commercial products cited in this article are registered trademarks of their respective companies.

REFERENCES

Abadi, D. J.; Madden, S. R. . H. N. (2008). *Column-Stores vs. Row-Stores: How Different Are They Really?* Proceedings of the ACM SIGMOD Conference on Management of Data, 967–980.

Adamson, C. (2006). *Mastering data warehouse aggregates : solutions for star schema performance*. Proceedings of the 21st International Conference on Information Integration and Web-Based Applications e Services. Association for Computing Machinery. (iiWAS2019), p. 361–369., Wiley Pub, Indianapolis.

Almeida, R., Furtado, P., and Bernardino, J. (2015). Performance evaluation mysql innodb and microsoft sql server 2012 for decision support environments. pages 56–62.

Costa, E., Costa, C., and Santos, M. Y. (2017). Efficient big data modelling and organization for hadoop hive-based data warehouses. In Themistocleous, M. and Morabito, V., editors, *Information Systems*, pages 3–16, Cham. Springer International Publishing.

Dumbill, E. (2013). *Making sense of big data*, page 1–2.

Eduardo Pina, Filipe Sá, J. B. (2023). Newsql databases assessment: Cockroachdb, mariadb xpand, and voltdb. Future Internet.

G. Lawrence Sanders, S. S. (2002). *Denormalization effects on performance of relational database for data warehouse*. State University of New York at Buffalo ProQuest Dissertations Publishing, New York.

Geomar Schreiner, Ronan Knob, A. F. R. M. (2019). Uma análise de soluções newsql. 2019: ANAIS DA XV ESCOLA REGIONAL DE BANCO DE DADOS.

Grolinger, K; Higashino, W. T. A. C. M. (2013). Data management in cloud environments: Nosql and newsql data stores. Journal of Cloud Computing: Advances, Systems and Applications.

Idrissi, M. A. A. (2016). Database-as-a-service for big data: An overview. (IJACSA) International Journal of Advanced Computer Science and Applications, Vol.7.

Jaecksch, B.;Lehner. W;; Faerber, F. (2010). *A plan for OLAP*. Proceedings of the 13th International Conference on Extending Database Technology, Pages 681–686.

Kimball, R.; Ross, M. (2002). *The data warehouse toolkit : the complete guide to dimensional modeling*. Wiley, New York.

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

Krishnan, K. (2013). *Data Warehousing in the Age of Big Data*. Morgan Kaufmann Publishers Inc., San Francisco, CA.

Murazzo, M., Gómez, P., Rodríguez, N., and Medel, D. (2019). *Database NewSQL Performance Evaluation for Big Data in the Public Cloud*, pages 110–121.

Murphy, B. D. P. T. (1988). An architecture for a business and information system. IBM Systems Journal, Vol. 1.

Nambiar, R. O. and Poess, M. (2006). The making of tpc-ds. In *VLDB*, volume 6, pages 1049–1058.

Oliveira, J.; Bernardino, J. (2017). Newsql databases - memsql and voltdb experimental evaluation. KEOD. [S.l.: s.n.], p. 276–281.

O'Neil, P; O'Neil, B. C. X. (2009). Star schema benchmark. IEEE 9th International Conference on Research Challenges in Information Science (RCIS) pp. 480-485, doi: 10.1109/RCIS.2015.7128909.

Pandis, I. (2021). The evolution of amazon redshif. Proceedings of the VLDB Endowment.

Pavlo, A.; Aslett, M. (2017). *What's really new with newsql?* Association for Computing Machinery v. 45, n. 2, p. 45–55., New York, NY.

Prout, S.-P. W. J. V. Z. S. Y. L. J. C. E. B. E. H. R. W. R. G. N. S. A. (2022). Cloud-native transactions and analytics in singlestore. SIGMOD: Proceedings of the 2022 International Conference on Management of Data.

Rabelo Ferreira, F. (2023). Python scripts created for the experiment. .

S Ilić, S Ilić, I. M. D. M. (2022). A comparison of query execution speeds for large amounts of data using various dbms engines executing on selected ram and cpu configurations. volume 29.

Serlin, O; Sawyer, T. G. J. (1998). Tpc-h and tpc-ds. https://www.tpc.org.

Singlestore (Accessed in April, 2023). Singlestore pricing website. .

Snowflake (September, 2023). Snowflake: One platform, all your data, all your users. .

Stonebraker, M. (2011). Newsql: An alternative to nosql and old sql for new oltp apps. Communications of the ACM Blog.

SCITEPRESS SCIENCE AND TECHNOLOGY PUBLICATIONS