UNIVERSIDADE FEDERAL DE PERNAMBUCO
CENTRO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

ALESANCO ANDRADE AZEVEDO

Data Warehouse NewSQL: uma análise de desempenho explorando estratégias de armazenamento e distribuição

Recife

2021

ALESANCO ANDRADE AZEVEDO

Data Warehouse NewSQL: uma análise de desempenho explorando estratégias de armazenamento e distribuição

Trabalho apresentado ao Programa de Pós-graduação em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco, como requisito parcial para obtenção do grau de Mestre em Ciência da Computação.

Área de Concentração: Bancos de Dados

Orientador (a): Robson do Nascimento Fidalgo

Recife

2021

Catalogação na fonte
Bibliotecária Monick Raquel Silvestre da S. Portes, CRB4-1217

A994d Azevedo, Alesanco Andrade
Data warehouse newSQL: uma análise de desempenho explorando estratégias de armazenamento e distribuição / Alesanco Andrade Azevedo. – 2021.
77 f.: il., fig., tab.

Orientador: Robson do Nascimento Fidalgo.
Dissertação (Mestrado) – Universidade Federal de Pernambuco. CIn, Ciência da Computação, Recife, 2021.
Inclui referências e apêndices.

1. Banco de dados. 2. *Data warehouse*. 3. Desempenho. I. Fidalgo, Robson do Nascimento (orientador). II. Título.

025.04 CDD (23. ed.) UFPE - CCEN 2021 – 117

Alesanco Andrade Azevedo

“Data Warehouse NewSQL: uma análise de desempenho explorando estratégias de armazenamento e distribuição”

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Pernambuco, como requisito parcial para a obtenção do título de Mestre em Ciência da Computação.

Aprovado em: 26/02/2021.

BANCA EXAMINADORA

_____

Prof. Dr. Eduardo Antônio Guimarães Tavares
Centro de Informática/ UFPE

_____

Prof. Dr. Cláudio de Souza Baptista
Centro de Engenharia Elétrica e Informática,
Sistemas e Computação / UFCG

_____

Prof. Dr. Robson do Nascimento Fidalgo
Centro de Informática / UFPE
(Orientador)

AGRADECIMENTOS

Agradeço a minha esposa Renata e meu filho Francisco por tudo que abdicaram em prol dessa realização. Agradeço aos meus pais, Aliezio e Socorro, por terem me proporcionado bons caminhos de desenvolvimento pessoal. Agradeço a Luiz Carlos pelo companheirismo nessa jornada, me ajudando muitas vezes quando necessitava.

Agradeço a minha gestora Alice, por ter me apoiado e compreendido a importância desse momento para mim.

Agradeço também a Robério pelo acompanhamento minucioso, com sua troca constante de experiências acadêmicas. Sua ajuda contribuiu enormemente para a melhoria deste trabalho. Sou eternamente grato a ele.

Agradeço ao meu orientador Robson Fidalgo por ter aceitado me conduzir nesse longo trabalho. Agradeço por ter fornecido o hardware necessário para a realização deste estudo e pelo apoio motivacional e técnico, com paciência e disposição.

E finalmente agradeço ao grande amigo Will, que partiu recentemente. Will, fico com a última palavra que trocamos: "obrigado". Até mais meu irmão.

RESUMO

Data Warehouses (DWs) são bancos de dados projetados para favorecer o processamento analítico de grandes volumes de dados. Com o intuito de prover melhor desempenho de armazenamento e processamento analítico em DWs, sistemas de bancos de dados NewSQL surgem como uma alternativa promissora. Essa classe de banco de dados oferece facilidades para suportar escalabilidade horizontal, linguagem SQL e armazenamento principal em memória RAM. Existem estudos que analisam o desempenho de distribuições NewSQL em processamento transacional e analítico de dados, contudo, até onde sabemos, não há estudo que analise o impacto do uso de diferentes esquemas de dados, métodos de distribuição e formas de armazenamento no desempenho de um DW implementado em NewSQL. Dessa forma, usando diferentes volumes de dados, propomos uma análise de desempenho em DWs NewSQL comparando diferentes esquemas de dados (esquema estrela e tabela flat), formas de armazenamento (rowstore e columnstore) e métodos de distribuição (replicação e particionamento por round-robin ou por hash). Para alcançar esse objetivo, realizamos uma avaliação experimental de desempenho em DWs, utilizando o Star Schema Benchmark (SSB) e o Sistema Gerenciador de Banco de Dados (SGBD) MemSQL, em estrutura de cluster de 3 computadores. Para a avaliação experimental, utilizamos métricas de volume e desempenho de tempo em tarefas de carga e consultas de dados. A partir dos dados coletados, verificamos que o uso de tabelas flat, armazenamento columnstore e particionamento por chave hash gerou os melhores resultados no tempo médio de consultas, apresentando, contudo, desvantagens no tempo de carga e no volume de dados armazenado. Destacamos ainda que o uso de columnstore, realizado em disco, conseguiu obter melhores resultados em tarefas de consulta, quando comparado rowstore realizado em RAM, diante todos os cenários avaliados.

Palavras-chaves: Data Warehouse . NewSQL . Desempenho . Columnstore . Rowstore . SSB

ABSTRACT

Data Warehouses (DWs) are databases designed to favor the analytical processing of large volumes of data. In order to provide better storage performance and analytical processing in DWs, NewSQL database systems appear as a promising alternative. This class of database provides facilities to support scale-out, SQL language and main storage in RAM. There are studies that analyze the performance of NewSQL distributions in transactional and analytical data processing, however, as far as we know, there is no study that analyzes the impact of the use of different data schemes, distribution methods and forms of storage on the performance of a DW implemented in NewSQL. Thus, using different data volumes, we propose a performance analysis in NewSQL DWs comparing different data schemas (star schema and flat table), storage forms (rowstore and columnstore) and distribution methods (replication and partitioning by round-robin or hash). To achieve this goal, we performed an experimental performance evaluation on DWs, using the Star Schema Benchmark (SSB) and the MemSQL Database Manager System (SGBD), in a cluster structure of 3 computers. For the experimental evaluation, we used volume and time performance metrics in loading tasks and data queries. From the data collected, we verified that the use of flat tables, columnstore storage and hash key partitioning generated the best results in the average time of queries, presenting, however, disadvantages in the load time and in the volume of data stored. We also highlight that the use of columnstore, performed on disk, managed to obtain better results in query tasks, when compared to rowstore performed in RAM, given all the evaluated scenarios.

Keywords: Data Warehouse . NewSQL . Performance . Columnstore . Rowstore . SSB

LISTA DE FIGURAS

Figura 1 – Estratégias de particionamento *round-robin* (a), *range* (b) e *hash* (c) . . . 17
Figura 2 – Formas de armazenamento em MemSQL (Rowstore e Columnstore) 19
Figura 3 – Exemplos de modelagem física em MemSQL, com (a) distribuição por hash e armazenamento columnstore e (b) distribuição por round-robin e armazenamento rowstore 20
Figura 4 – Exemplo de esquemas de DW totalmente desnormalizado (a), parcialmente desnormalizado (b) e normalizado (c) 22
Figura 5 – Esquema do SSB com quantitativo de registros em cada tabela 23
Figura 6 – Etapas da pesquisa bibliográfica 27
Figura 7 – Média dos tempos de consulta 29
Figura 8 – Média de transações por segundo (a) e Latência média das transações (b) . 30
Figura 9 – Resultado das métricas 31
Figura 10 – Média dos tempos de cada carga de trabalho por SGBD NewSQL em milissegundos 32
Figura 11 – Média de throughput por SGBD NewSQL em operações por segundo . . . 32
Figura 12 – Métrica RPS: relação entre valor predito e valor real para os modelos gerados 34
Figura 13 – Métricas TRS, PRS e VPS: Pontuação R2 obtido com grid-search e 10-fold para características lineares e quadráticas com métodos RL e GBM (a) e com métodos AB e ERT (b) 34
Figura 14 – Cruzamento das configurações 39
Figura 15 – Etapas de execução do experimento 42
Figura 16 – Tempo ordenado de carga em cenários com esquema estrela (SF=10) . . . 45
Figura 17 – Gráfico relacionando volume de dados armazenado em disco e memória RAM - SF 1, 10 e 100 47
Figura 18 – Ranking de médias nas consultas Q1.1, Q2.1, Q3.1 e Q4.1, em SF=100 . . 52

LISTA DE QUADROS

Quadro 1 – Lista de consultas do SSB 24
Quadro 2 – Configuração dos cenários experimentados em cada SF (1, 1 e 100) (F - Tabela de Fatos, D - tabelas dimensionais e Flat - Tabela flat (ou tabela única)) 40
Quadro 3 – Tempo de carga em SF = 1, 10 e 100 44
Quadro 4 – Volume de dados armazenado em disco e RAM - SF 1, 10 e 100 46
Quadro 5 – Diferença no volume de dados armazenado entre esquema estrela e tabela flat, considerando ocupação em disco e em RAM - SF 1, 10 e 100 47
Quadro 6 – Diferença no volume de dados armazenado entre columnstore e rowstore, de acordo a forma de armazenamento na tabela de fatos e na tabela flat - SF 1, 10 e 100 48
Quadro 7 – Tempo médio de execução das 13 consultas do SSB, em SF = 1 49
Quadro 8 – Tempo médio de execução das 13 consultas do SSB, em SF = 10 50
Quadro 9 – Tempo médio de execução das 13 consultas do SSB, em SF = 100 50
Quadro 10 – Tempo médio das consultas Q1.1, Q2.1, Q3.1 e Q4.1, em SF 1, 10 e 100 51
Quadro 11 – Média das consultas Q1.1, Q2.1, Q3.1 e Q4.1, de acordo com o tipo de tabela e a forma de armazenamento, em SF1, 10 e 100 52
Quadro 12 – Desempenho médio na execução de consultas envolvendo 1 dimensão (Q1.1) e envolvendo 4 dimensões (Q4.1), em SF=100 53
Quadro 13 – Consumo de RAM durante a execução das consultas, nos SFs 1, 10 e 100 54
Quadro 14 – Consumo de RAM durante consultas (sumarizado) e diferença no consumo de RAM durante a execução de consultas 55

LISTA DE TABELAS

Tabela 1 – Exemplos de SGBDs NewSQL 18
Tabela 2 – Lista de trabalhos relacionados a esta pesquisa 28
Tabela 3 – Tempo médio de carga com SF=1 28
Tabela 4 – Avaliação consolidada dos trabalhos relacionados 35
Tabela 5 – Combinações de abordagens para o experimento 37
Tabela 6 – Comparação do presente trabalho com os trabalhos relacionados 60

LISTA DE ABREVIATURAS E SIGLAS

DDL	*Data Definition Language*
DW	*Data Warehouse*
ETL	*Extract, Transform and Loading*
HTAP	*Hybrid transactional/analytical processing*
OLAP	*Online Analytical Processing*
PK	*Primary Key*
SF	*Scale Fator*
SGBDs	Sistemas de Gerenciamento de Banco de Dados
SK	*Shard-Key*
SSB	*Star Schema Benchmark*

SUMÁRIO

1 INTRODUÇÃO 13
1.1 CONTEXTUALIZAÇÃO 13
1.2 PERGUNTA NORTEADORA À PESQUISA 14
1.3 OBJETIVOS 14
1.4 METODOLOGIA 15
1.5 ESTRUTURA DA DISSERTAÇÃO 15
2 REFERENCIAL TEÓRICO 16
2.1 NEWSQL 16
2.1.1 MemSQL (SingleStore) 18
2.2 DATA WAREHOUSE 21
2.3 STAR SCHEMA BENCHMARK - SSB 22
2.4 CONSIDERAÇÕES 24
3 REVISÃO DE LITERATURA 26
3.1 MÉTODO DE BUSCA E CRITÉRIOS DE ANÁLISE 26
3.2 TRABALHOS RELACIONADOS 27
3.2.1 NewSQL Databases - MemSQL and VoltDB Experimental Evaluation 28
3.2.2 NewSQL Through the Looking Glass 29
3.2.3 Performance Evaluation of NewSQL Databases 30
3.2.4 Performance Benchmarking of NewSQL Databases with Yahoo Cloud Serving Benchmark 31
3.2.5 Uma abordagem para modelagem de desempenho para sistemas de banco de dados NewSQL e comparação de modelos gerados com diferentes técnicas de aprendizado de máquina 33
3.3 ANÁLISE GERAL 35
3.4 CONSIDERAÇÕES 36
4 EXPERIMENTO 37
4.1 DEFINIÇÕES DO EXPERIMENTO E DO AMBIENTE DE AVALIAÇÃO . . 37
4.2 EXECUÇÃO DO EXPERIMENTO 41
4.3 CONSIDERAÇÕES 42
5 RESULTADOS 43

5.1 CARGA DE DADOS - TEMPO 43
5.2 CARGA DE DADOS - VOLUME DE DADOS 45
5.3 CONSULTAS - TEMPO MÉDIO 49
5.4 CONSULTAS - CONSUMO DE RAM 54
5.5 ANÁLISE GERAL 56
5.6 CONSIDERAÇÕES 58
6 CONSIDERAÇÕES FINAIS 59
6.1 LIMITAÇÕES 61
6.2 TRABALHOS FUTUROS 61
REFERÊNCIAS 62
APÊNDICE A – SCRIPTS EXECUTADOS NO EXPERIMENTO . 65
APÊNDICE B – DADOS DO TEMPO MÉDIO DE CONSULTAS . 70

1 INTRODUÇÃO

Neste capítulo, apresentamos a introdução desta dissertação, destacando o seu contexto (Seção 1.1). Prosseguimos com a pergunta norteadora de pesquisa (Seção 1.2) e os objetivos gerais e específicos (Seção 1.3). Por fim, apresentamos a metodologia de trabalho (Seção 1.4) e a estrutura geral do texto (Seção 1.5).

1.1 CONTEXTUALIZAÇÃO

Um *Data Warehouse* (DW) é um banco de dados histórico, não volátil e estruturado, usado no processo de tomada de decisão (INMON, 1997; KIMBALL; ROSS, 2002). DWs são construídos a partir de um processo chamado *Extract, Transform and Loading* (ETL), o qual obtém dados de diferentes fontes, transforma esses dados para tratar possíveis heterogeneidades e/ou adequá-los a um esquema pré-definido e carrega esses dados em um DW (PONNIAH, 2010). O esquema de dados de um DW é composto uma ou mais tabelas de fatos, que armazenam medidas do evento (e.g., quantidade de produtos vendidos), relacionadas a uma ou mais tabelas de dimensões, que armazenam dados descritivos (e.g., marca do produto). Além disso, DWs normalmente desnormalizam os dados de suas dimensões para melhorar o desempenho das consultas. Apesar desse benefício, a desnormalização das dimensões aumenta o volume de dados, exigindo maior capacidade de armazenamento no hardware empregado. Tradicionalmente, os DWs são implementados em banco de dados relacionais centralizados, contudo, existem estudos recentes que avaliaram alternativas a essa tecnologia de armazenamento, como por exemplo: NoSQL (BOUSSAHOUA; BOUSSAID; BENTAYEB, 2017; FERRO; FRAGOSO; FIDALGO, 2019) e NewSQL (OLIVEIRA; BERNARDINO, 2017).

Dentre as alternativas supracitadas, este trabalho irá focar no uso da tecnologia NewSQL, que é conceituada como uma classe de banco de dados projetada para estender os benefícios do modelo relacional com arquiteturas em cluster, promovendo ainda o armazenamento em memória, a escalabilidade horizontal (i.e., expansão computacional a partir de mais máquinas) e a alta disponibilidade. Os Sistemas de Gerenciamento de Banco de Dados (SGBDs) NewSQL também são caracterizados pelo armazenamento e processamento distribuído, utilização da linguagem SQL e manutenção das propriedades ACID (PAVLO; ASLETT, 2016). MemSQL, VoltDB e Google Spanner são exemplos de SGBDs NewSQL (DB-Engines, 2020), aos quais

apresentam diferentes métodos de distribuição e formas de armazenamento de dados. Sobre o armazenamento de dados, esse pode ser orientado por linhas (rowstore) ou por colunas (columnstore). Por sua vez, o particionamento de dados pode ser feito com base nas estratégias round-robin (i.e., distribuição sequencial), range (i.e., distribuição por uma faixa de valores) ou hash (i.e., distribuição por uma função hash). Há estudos que analisam o desempenho de bancos de dados NewSQL em sistemas transacionais (KAUR; SACHDEVA, 2017; SCHREINER et al., 2019) e analíticos (OLIVEIRA; BERNARDINO, 2017). No entanto, até onde sabemos, não há estudo que analise o desempenho no uso de diferentes esquemas de dados, métodos de distribuição e formas de armazenamento dos bancos de dados NewSQL, no contexto de DW (i.e., um DW NewSQL).

1.2 PERGUNTA NORTEADORA À PESQUISA

Considerando que diferentes abordagens tecnológicas suportadas por NewSQL podem influenciar no desempenho de uma solução de DW, formulamos a seguinte questão de pesquisa: Que impactos as diferentes combinações de esquemas de dados, métodos de distribuição e formas de armazenamento suportados por NewSQL podem provocar no desempenho de DWs?

O entendimento desses impactos visa fornecer um melhor aproveitamento dos recursos de hardware e no tempo de execução de operações de DWs implementados com NewSQL. Tais informações são úteis a pesquisadores que desejam implementar DWs em bancos de dados relacionais, utilizando hardware de baixo custo e manutenção de desempenho, mesmo em grandes volumes de dados.

1.3 OBJETIVOS

O objetivo geral deste trabalho é realizar uma avaliação experimental para analisar o impacto da tecnologia NewSQL no contexto de um ambiente de DW. Com base nisso, serão avaliados cenários que combinam diferentes abordagens de armazenamento (i.e., rowstore e columnstore), de distribuição (i.e., round-robin e hash) e de esquemas de dados (i.e., esquema estrela e tabela flat), em um ambiente controlado, provido por um benchmark específico para processamento analítico a partir de um DW. Para alcançar este objetivo geral, temos os seguintes objetivos específicos:

- Realizar estudo para se apropriar dos principais conceitos e trabalhos relacionados;
- Definir cenários de teste que abordem diferentes combinações de esquemas de dados, formas de armazenamento e métodos de distribuição de dados suportados por NewSQL;
- Implementar o ambiente experimental, para realizar carga de dados e execução de consultas em cada cenário de teste;
- Coletar, analisar e discutir as métricas obtidas no experimento;

1.4 METODOLOGIA

Iniciamos o estudo com uma revisão da literatura acerca de NewSQL, destacando trabalhos que realizam análises de desempenho, destacando suas lacunas. Em seguida, selecionamos o SGBD MemSQL para a realização de um estudo experimental, sendo este sistema instalado em ambiente computacional composto por 3 máquinas conectadas em cluster. No intuito de melhorar o controle dos cenários de testes, usamos o *Star Schema Benchmark* (SSB), por ser um benchmark bem conhecido na literatura, que disponibiliza a estrutura e os dados de um DW, bem como um conjunto de consultas que simulam operações OLAP. Em seguida, construímos 28 cenários de testes, combinando diferentes esquemas de dados (esquema estrela e tabela flat), métodos de distribuição (replicação de tabelas e particionamento por round-robin e por hash) e formas de armazenamento (rowstore e columnstore), para execução em 3 diferentes volumes de dados. Em cada um desses cenários, coletamos o tempo de carga, o volume de dados gerado, o tempo médio de execução de consultas e o consumo de memória RAM durante a execução das consultas. Analisamos e discutimos os resultados de cada métrica avaliada, apresentando, por fim, as conclusões do estudo.

1.5 ESTRUTURA DA DISSERTAÇÃO

Estruturamos este trabalho da seguinte forma: no Capítulo 2 apresentamos referenciais teóricos necessários ao entendimento desta pesquisa. No Capítulo 3, discutimos trabalhos acadêmicos que avaliam o desempenho de bancos de dados NewSQL. No Capítulo 4, propomos um experimento para comparar o desempenho de várias implementações de DW NewSQL. No Capítulo 5, apresentamos os resultados obtidos e realizamos análises sobre os mesmos. Por fim, no Capítulo 6, apresentamos as considerações finais deste estudo.

2 REFERENCIAL TEÓRICO

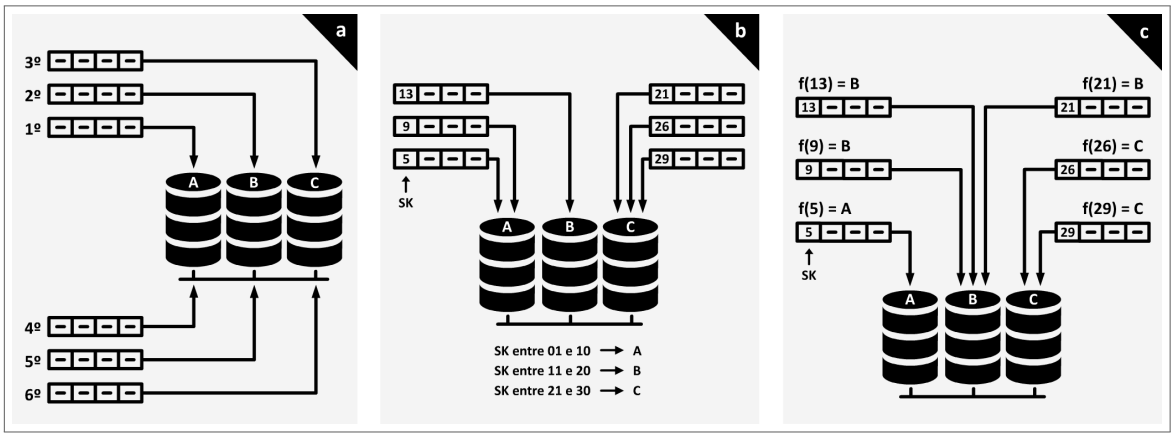
Apresentamos neste capítulo uma visão geral sobre o referencial teórico necessário à compreensão do presente trabalho. Neste sentido, abordamos os conceitos e características do NewSQL e do SGBD MemSQL (Seção 2.1). Na sequência, apresentamos conceitos sobre Data Warehouse (Seção 2.2). Depois, apresentamos o benchmark SSB (Seção 2.3). Por fim, realizamos as considerações finais deste capítulo (Seção 2.4).

2.1 NEWSQL

NewSQL é uma classe de banco de dados relacional que possui suporte à escalabilidade horizontal por meio do particionamento de tuplas em subconjuntos disjuntos, também chamados de partições ou shards (PAVLO; ASLETT, 2016; DUGGIRALA, 2018). Essas porções de dados são distribuídas entre os nós de um cluster com base em alguma estratégia de particionamento, entre as quais se destacam round-robin, range e hash (COSTA et al., 2015; DEWITT; GRAY, 1992). Na estratégia round-robin, tuplas de uma tabela são distribuídas de forma sequencial entre os nós, sem considerar qualquer valor de atributo, numa distribuição próxima à igualdade. Na estratégia de particionamento por range, um campo é definido como chave de particionamento (i.e., *Shard-Key* (SK)) e faixas de valores desse campo determinam onde as tuplas serão armazenadas fisicamente. A técnica de particionamento por hash também utiliza uma SK que define qual nó armazena uma tupla, no entanto, essa definição é realizada por meio de uma função hash. Tanto o particionamento por range, quanto por hash permitem agrupar, fisicamente, tuplas levando em consideração alguma similaridade entre os dados. A Figura 1 ilustra as estratégias round-robin, range e hash, abordando um cenário onde seis tuplas são distribuídas entre três nós (A, B, C) de um cluster.

A maioria dos bancos de dados NewSQL suporta o uso da memória RAM como meio de armazenamento principal, o que visa aumentar o desempenho na execução de consultas (DUGGIRALA, 2018). Além disso, alguns bancos de dados NewSQL realizam gerenciamento do consumo de memória RAM, promovendo o deslocamento de dados pouco utilizados (tuplas frias) para a memória secundária (PAVLO; ASLETT, 2016). Essa estratégia ajuda a prevenir possíveis gargalos no consumo de RAM e também auxilia quando o tamanho das bases de dados é superior ao limite de memória RAM instalada no cluster. VoltDB e MemSQL são

Figura 1 – Estratégias de particionamento *round-robin* (a), *range* (b) e *hash* (c)



Fonte: adaptado de (DEWITT; GRAY, 1992; COSTA et al., 2015).

exemplos de bancos de dados NewSQL que suportam nativamente o armazenamento em memória RAM.

As tabelas de um banco de dados NewSQL podem assumir duas formas de armazenamento físico de dados: baseada em linhas (rowstore) ou baseada em colunas (columnstore). Rowstore é a forma clássica de armazenamento em bancos de dados relacionais, que consiste em armazenar todos os campos de uma linha (i.e., uma tupla) em uma faixa contínua de endereçamento físico (ABADI; MADDEN; FERREIRA, 2006). No armazenamento columnstore, as colunas de uma tabela são separadas, ordenadas e armazenadas em faixas contínuas de endereçamento. Além disso, os dados das colunas são comprimidos, o que reduz o volume de dados. A criação de tabelas com armazenamento colunar passa pela escolha de um de seus campos para ser a chave columnstore da tabela e é com base nos valores dessa chave, que as tuplas são posicionadas nas partições de um determinado nó. A utilização de columnstore possibilita varredura de colunas de forma pontual, selecionando apenas colunas necessárias a cada consulta. VoltDB e NuoDB são alguns dos bancos de dados NewSQL que realizam armazenamento em rowstore, enquanto que MemSQL e TiDB são exemplos de NewSQL que suportam rowstore e columnstore.

Os bancos de dados NewSQL são classificados em *New Architecture*, *Transparent Clustering/Sharding* e *Database-as-a-Service* (DBaaS) (PAVLO; ASLETT, 2016). *New Architecture* é composta por bancos de dados construídos do zero para operar em ambiente distribuído. *Transparent clustering/sharding* consiste em bancos de dados intermediários entre a aplicação e um banco de dados não-distribuído, com a função de realizar a distribuição de dados entre máquinas de um cluster. E por fim, DBaaS são serviços de bancos de dados NewSQL virtualizados em nuvem, que simplificam etapas de instalação, otimização e replicação de ba-

ses. Na Tabela 1, apresentamos um conjunto de SGBDs NewSQL, dentre os mais conhecidos pela indústria, destacando algumas de suas características, bem como seu ranque, segundo o DB-Engines[1].

Tabela 1 – Exemplos de SGBDs NewSQL

SGBD	Forma de armazenamento	Estratégia de particionamento	Classe NewSQL
Cloud Spanner[2]	Rowstore	Hash	DBaaS
CockRoachDB[3]	Rowstore	Hash	New architecture
MemSQL[4]	Rowstore/Columnstore	Hash/Round-robin	New architecture/DBaaS
NuoDB[5]	Rowstore	Proprietária	New architecture
SAP HANA[6]	Rowstore/Columnstore	Hash/Range/Round-robin	New architecture/DBaaS
ScaleArc[7]	Rowstore	Não informado	Transparent sharding
TiDB[8]	Rowstore/Columnstore	Range/Hash	New architecture
VoltDB[9]	Rowstore	Consistent Hash	New architecture

Fonte: Elaborado pelo autor (2021).

Dentre os SGBDs NewSQL listados na Tabela 1, MemSQL, SAP HANA e TiDB se destacam por suportarem duas formas de armazenamento e duas estratégias de particionamento dos dados, permitindo maior variação na implementação de tabelas. Por diferencial entre eles, podemos destacar o MemSQL por suportar maior armazenamento em memória RAM, na versão de testes (com aproximadamente 128GB, ante os 32GB do SAP HANA) e por precisar de menor número de máquinas na formação de um cluster (2 máquinas, ante as 10 máquinas para um cluster com TiDB[10]). Na próxima Seção (2.1.1), detalhamos o SGBD MemSQL.

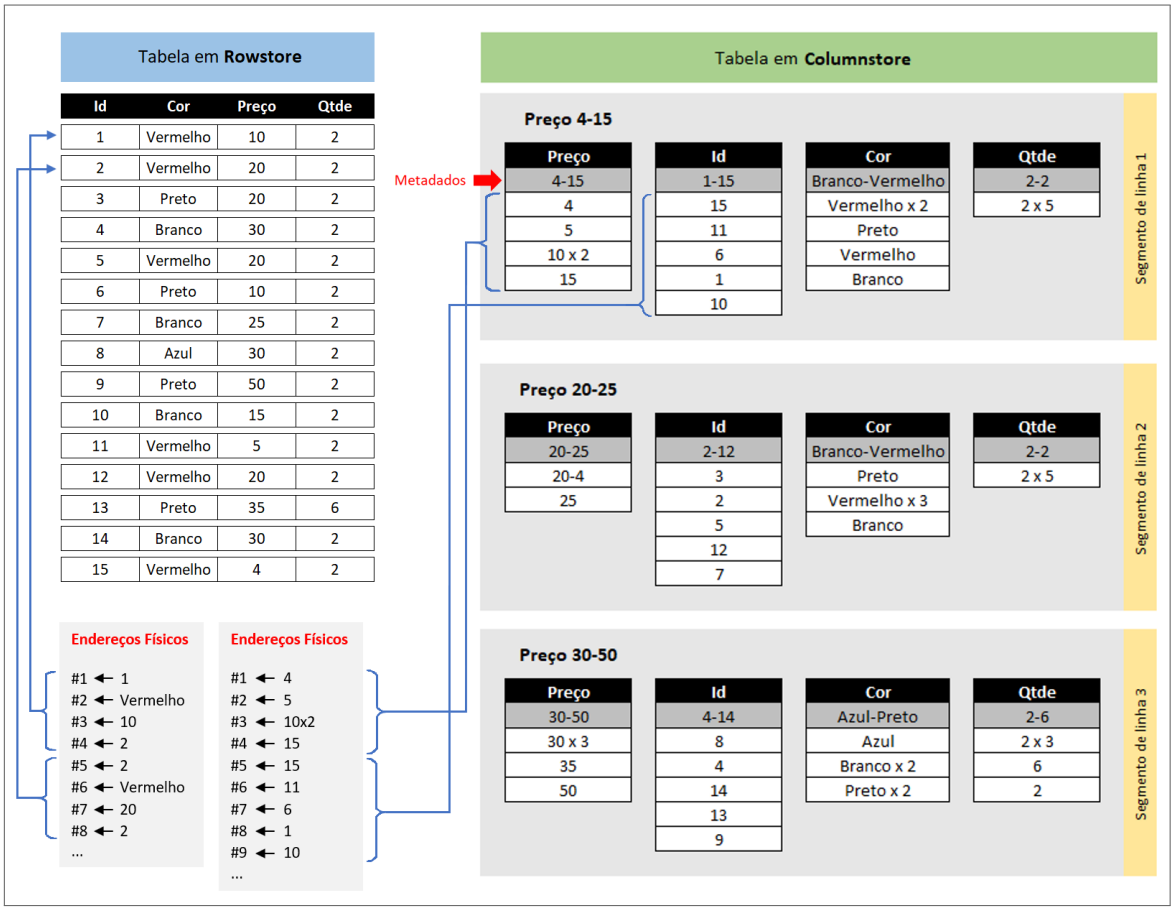
2.1.1 MemSQL (SingleStore)

MemSQL (recentemente renomeado para SingleStore) é um banco de dados relacional distribuído, classificado como NewSQL *new Architecture*, que é caracterizado pelo suporte aos armazenamentos rowstore e columnstore, além de distribuir dados por 3 métodos distintos. O armazenamento rowstore realizado em MemSQL é o mesmo dos bancos de dados relacionais, apresentando o diferencial de ser realizado diretamente em memória RAM. Já as tabelas no

[1] https://db-engines.com/en/ranking (01/2020)
[2] https://cloud.google.com/spanner
[3] https://www.cockroachlabs.com/product/
[4] https://www.singlestore.com/ (MemSQL mudou recentemente seu nome para SingleStore)
[5] https://nuodb.com/
[6] https://www.sap.com/brazil/products/hana.html
[7] https://www.devgraph.com/scalearc/
[8] https://pingcap.com/products/tidb/
[9] https://www.voltdb.com/
[10] https://docs.pingcap.com/tidb/stable/minimal-deployment-topology

formato columnstore têm seus dados armazenados em disco, e a partir deles, são gerados metadados que são armazenados em memória RAM. Esses metadados contêm informações intervalares dos dados em cada coluna (Figura 2, abaixo do nome de cada campo) e auxiliam na localização de valores. Os registros em columnstore são então agrupados em segmentos de linhas, dividindo grandes quantidades de dados em porções menores, a fim de contribuir na eliminação de segmentos ao realizar consultas. No exemplo da Figura 2, ilustramos dados de um mesmo nó, com 4 campos de uma tabela em rowstore e também em columnstore. No endereçamento físico desses dados, podemos perceber como os dados ficam dispostos, respeitando uma sequência baseada em linhas ou colunas. O campo "Preço" foi escolhido para ser a chave columnstore e a partir dele, tuplas são ordenadas e agrupadas em 3 segmentos de linha (preço de 4 a 15, de 20 a 25 e de 30 a 50). Neste exemplo, o limite é de 5 tuplas por segmento de linha, mas em aplicações reais, o limite é na casa dos milhões.

Figura 2 – Formas de armazenamento em MemSQL (Rowstore e Columnstore)

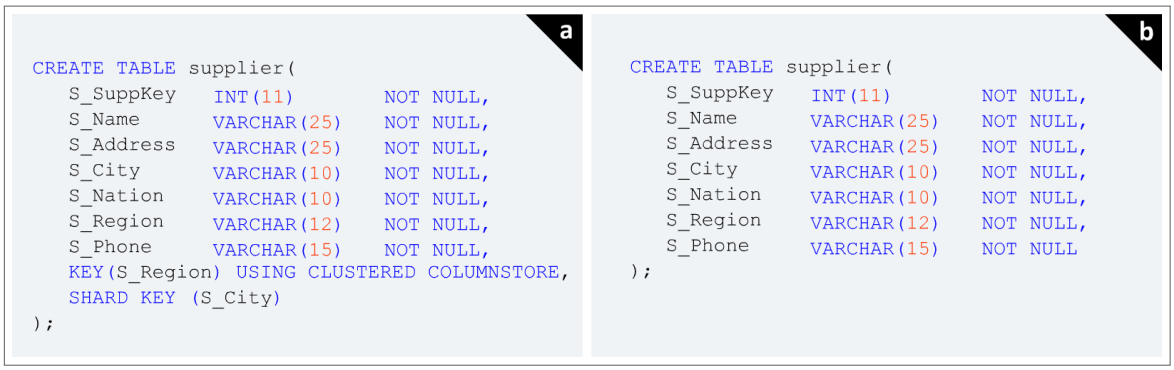


Fonte: adaptado de MemSQL (2020b).

MemSQL possibilita a escolha do método de distribuição de suas tabelas, podendo ser particionadas ou enviadas inteiramente para os nós. As partições das tabelas são distribuídas

de acordo com a estratégia hash ou round-robin. A estratégia hash é usada quando a tabela possui uma chave de particionamento (SK) ou *Primary Key* (PK). Por sua vez, a estratégia round-robin é usada quando a tabela não possui nenhuma dessas chaves (MemSQL, 2020). Além dessas duas estratégias, MemSQL ainda suporta a distribuição de todas as tuplas de uma tabela, para todos os nós folha de um cluster. Em outras palavras, é uma replicação de tabelas, que neste banco de dados é chamada de *reference table*. Em MemSQL, a definição dos métodos de distribuição e das formas de armazenamento empregados ocorre na própria definição da modelagem física de uma tabela. O exemplo da Figura 3 ilustra uma tabela com duas possíveis combinações de recursos: (a) distribuição por hash e armazenamento columnstore e (b) distribuição por round-robin e armazenamento rowstore.

Figura 3 – Exemplos de modelagem física em MemSQL, com (a) distribuição por hash e armazenamento columnstore e (b) distribuição por round-robin e armazenamento rowstore



Fonte: Elaborado pelo autor (2021).

Este banco de dados pode ser configurado na forma de um cluster, onde os nós podem assumir 3 papéis: agregador master, agregador ou folha (MEMSQL, 2020a). O agregador master gerencia os nós do cluster, além de coordenar a execução das operações de criação da estrutura do banco via comandos da *Data Definition Language* (DDL), como por exemplo, para criação de tabelas. Os nós agregadores armazenam metadados gerados com as informações de cada tabela (e.g., o intervalo de valores de uma partição), distribuindo também instruções SQL para os nós folha. Por fim, as folhas armazenam fatias dos dados e executam as instruções SQL emitidas pelos agregadores. Vale ressaltar que, quando o número conexões simultâneas é baixo e a quantidade de nós folha é pequena, os nós agregadores podem ser descartados, com o agregador master assumindo suas funções.

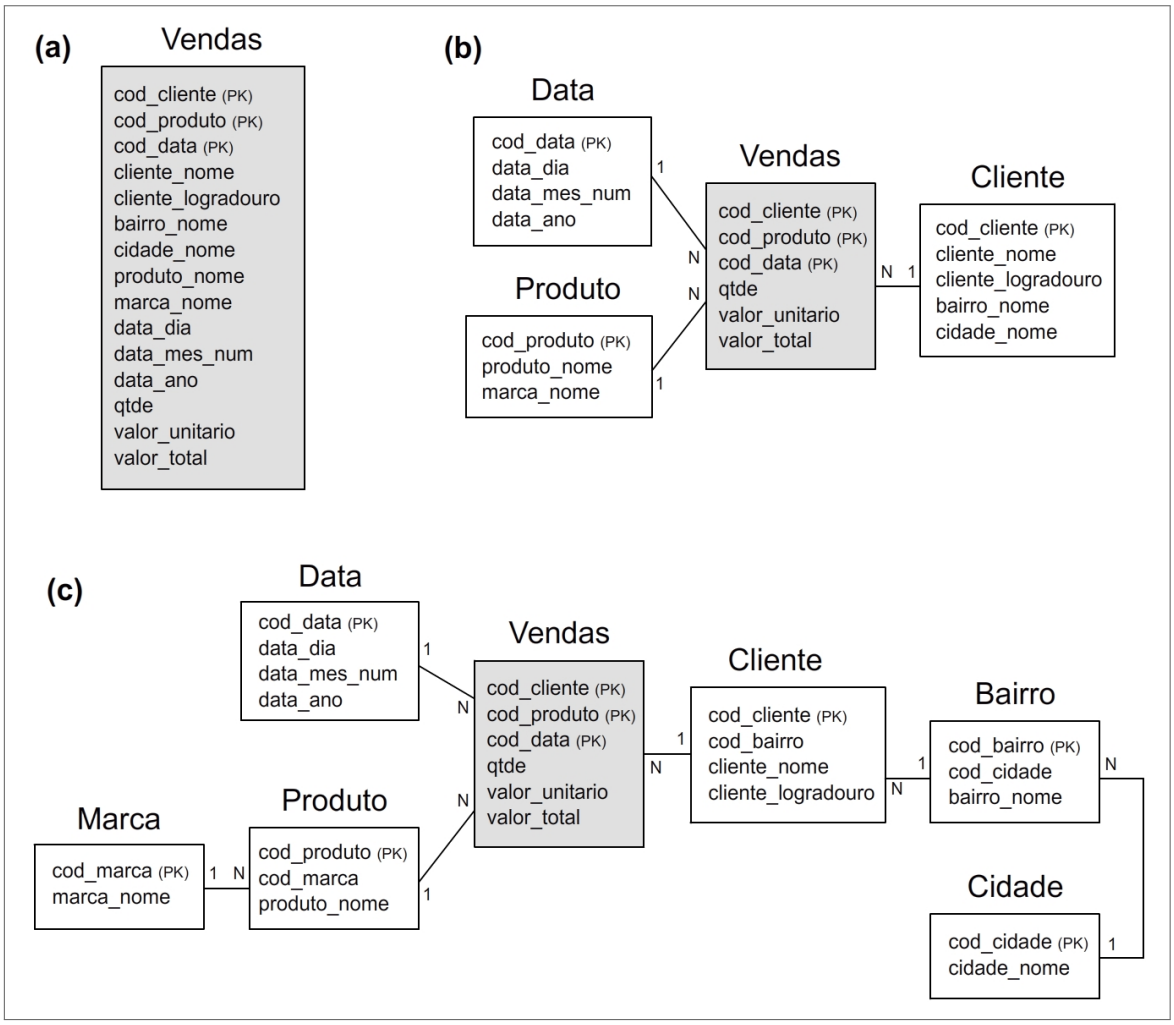
2.2 DATA WAREHOUSE

Um DW é um banco de dados histórico, consolidado e utilizado para processamento analítico de grandes volumes de dados (ou *Online Analytical Processing* (OLAP)), sendo projetado para realizar consultas que envolvam funções de agregação e operações de GROUP BY. Exemplos de consultas em DWs são: "quantos produtos de uma determinada marca foram vendidos por semana?" ou "qual a média mensal de frequência nas escolas municipais?". DWs são compostos por 2 tipos de tabelas: a tabela de fatos, para armazenar dados de eventos e a tabela de dimensão, que contém as características dos eventos.

O esquema de dados de um DW pode ser totalmente desnormalizado, parcialmente desnormalizado ou normalizado (ADAMSON, 2006; MURAZZA; NURWIDYANTORO, 2016; COSTA; SANTOS, 2018). No esquema totalmente desnormalizado (Figura 4a), os dados das tabelas de fatos e dimensões são armazenados em única tabela, resultando em grande nível de redundância, no entanto, elimina a necessidade de junções entre tabelas na realização de consultas. Em um esquema de dados parcialmente desnormalizado (Figura 4b), as tabelas de dimensão estão separadas da tabela de fatos, sendo as primeiras desnormalizadas e a segunda normalizada. Por fim, em um esquema normalizado (Figura 4c), não há redundância de dados, pois tanto as tabelas de dimensões quanto a tabela de fatos são normalizadas. Neste tipo de esquema, o volume de dados é menor, contudo, como os dados são estruturados em um número maior de tabelas, suas consultas utilizam um número maior de junções quando comparadas às duas abordagens anteriores. Exemplos de esquemas de dados que representam essas três abordagens são: tabela flat, esquema estrela e esquema floco de neve (GARANI; HELMER, 2012; COSTA; SANTOS, 2018).

Independente do esquema de dados utilizado, o povoamento do DW exige a Extração, Transformação e Carga de dados (ETL) para: acessar as fontes de dados, ajustar os dados extraídos e carregá-los no DW, respectivamente. Uma vez o DW construído, várias operações OLAP podem ser realizadas a partir dos seus dados. Entre estas, destacamos: *drill-down* - para migrar para um nível de agrupamento mais detalhado (e.g., visualizar as vendas por estado e depois por cidade); *roll-up* - sendo o oposto da operação anterior; *slice* - para projetar todos os dados de um ou mais campos (e.g., visualizar as vendas de todos os produtos por todos os meses) e *dice* - para selecionar apenas os dados que atendam a uma condição lógica (e.g., visualizar as vendas em janeiro e fevereiro dos produtos do tipo bebida) (CHAUDHURI; DAYAL, 1997).

Figura 4 – Exemplo de esquemas de DW totalmente desnormalizado (a), parcialmente desnormalizado (b) e normalizado (c)



Fonte: Elaborado pelo autor (2021)

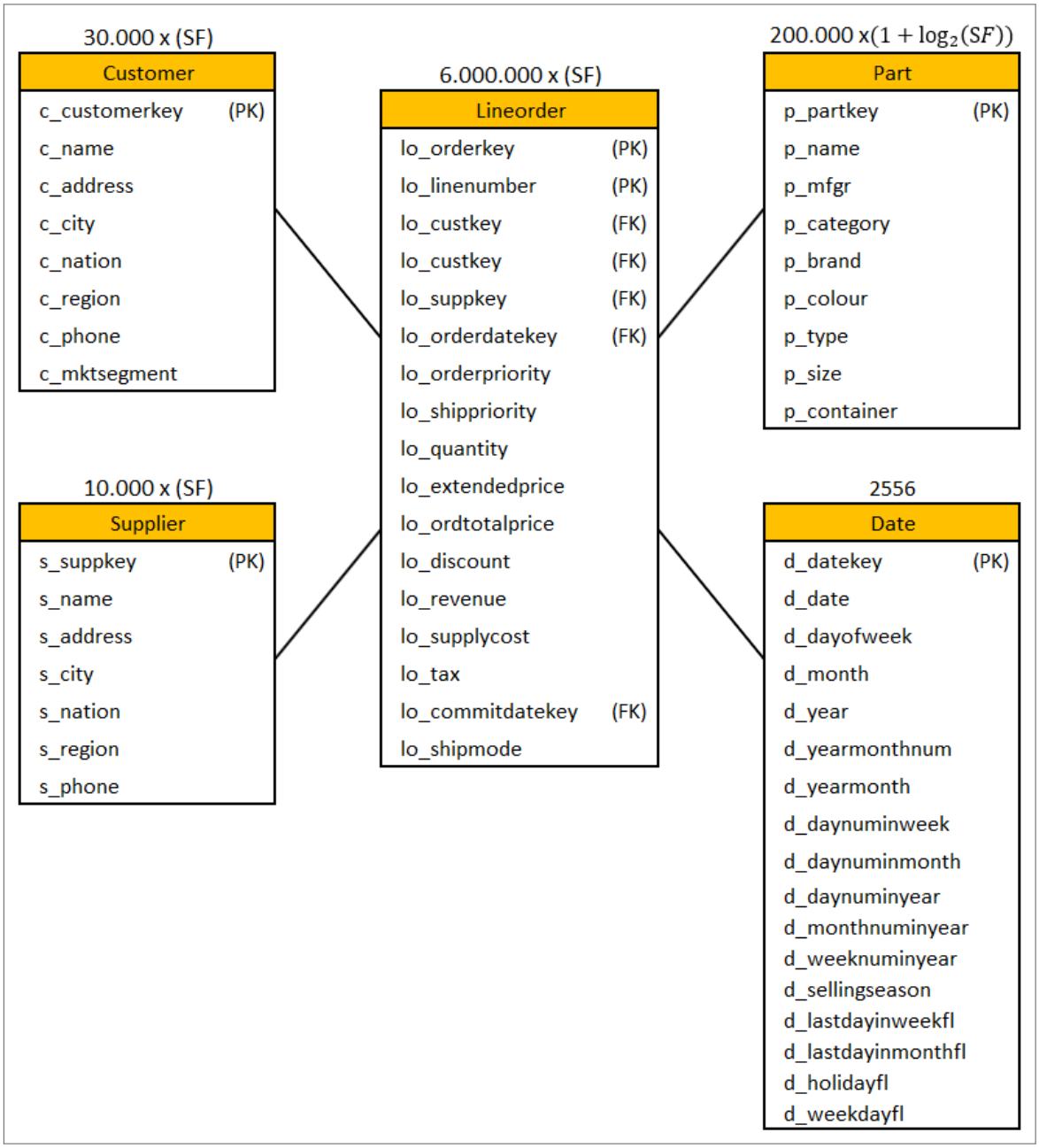
2.3 STAR SCHEMA BENCHMARK - SSB

Após levantamento sobre benchmarks para processamento analítico em DW, encontramos as seguintes propostas: TPC-H[11], TPC-DS[12] e SSB[13]. Nós escolhemos o SSB porque esse benchmark é o único baseado no esquema estrela (esquema mais utilizado em DWs). O SSB é composto por uma tabela de fatos (*Lineorder*) e quatro tabelas de dimensão (*Supplier*, *Customer*, *Part* e *Date*) (O'NEIL et al., 2009). Os dados do SSB podem ser obtidos em diferentes fatores de escala (ou *Scale Fator* (SF)), o que determina a quantidade de dados em suas tabelas. A tabela *Lineorder*, por exemplo, possui 6.000.000 registros em SF=1, sendo

[11] http://www.tpc.org/tpch/
[12] http://www.tpc.org/tpcds/
[13] https://github.com/electrum/ssb-dbgen

esta quantidade multiplicada por 10 e por 100, quando o SF é configurado em 10 e 100, respectivamente. As demais tabelas de dimensões também possuem diferentes quantidades de registros nos diferentes SF, com exceção da tabela *Date*, que possui 2556 registros em qualquer SF, representando um intervalo de 7 anos. A Figura 5 ilustra o esquema do SSB, indicando na parte superior das tabelas o cálculo para a quantidade de registros.

Figura 5 – Esquema do SSB com quantitativo de registros em cada tabela



Fonte: adaptado de (O'NEIL et al., 2009).

O SSB disponibiliza 13 consultas SQL que envolvem operações de *drill down*, *roll up*, *slice* e *dice*. Essas consultas são classificadas em quatro grupos que diferem entre si no número de tabelas e pela complexidade das consultas. O Quadro 1 apresenta as consultas, destacando os grupos, o código da consulta, a quantidade de dimensões e a consulta SQL.

Quadro 1 – Lista de consultas do SSB

Grupo	Código	Dimen.	Consulta
1	1.1	1	select sum(lo_extendedprice*lo_discount) as revenue from lineorder join dim_date on lo_orderdatekey = d_datekey where d_year = 1993 and lo_discount between 1 and 3 and lo_quantity < 25;
1	1.2	1	select sum(lo_extendedprice*lo_discount) as revenue from lineorder join dim_date on lo_orderdatekey = d_datekey where d_yearmonth = 199401 and lo_discount between 4 and 6 and lo_quantity between 26 and 35;
1	1.3	1	select sum(lo_extendedprice*lo_discount) as revenue from lineorder join dim_date on lo_orderdatekey = d_datekey where d_weeknuminyear = 6 and d_year = 1994 and lo_discount between 5 and 7 and lo_quantity between 26 and 35;
2	2.1	3	select sum(lo_revenue), d_year, p_brand from lineorder join dim_date on lo_orderdatekey = d_datekey join part on lo_partkey = p_partkey join supplier on lo_suppkey = s_suppkey where p_category = 'MFGR#12' and s_region = 'AMERICA' group by d_year, p_brand order by d_year, p_brand;
2	2.2	3	select sum(lo_revenue), d_year, p_brand from lineorder join dim_date on lo_orderdatekey = d_datekey join part on lo_partkey = p_partkey join supplier on lo_suppkey = s_suppkey where p_brand between 'MFGR#2221' and 'MFGR#2228' and s_region = 'ASIA' group by d_year, p_brand order by d_year, p_brand;
2	2.3	3	select sum(lo_revenue), d_year, p_brand from lineorder join dim_date on lo_orderdatekey = d_datekey join part on lo_partkey = p_partkey join supplier on lo_suppkey = s_suppkey where p_brand= 'MFGR#2239' and s_region = 'EUROPE' group by d_year, p_brand order by d_year, p_brand;
3	3.1	3	select c_nation, s_nation, d_year, sum(lo_revenue) as revenue from customer join lineorder on lo_custkey = c_customerkey join supplier on lo_suppkey = s_suppkey join dim_date on lo_orderdatekey = d_datekey where c_region = 'ASIA' and s_region = 'ASIA' and d_year >= 1992 and d_year <= 1997 group by c_nation, s_nation, d_year order by d_year asc, revenue desc;
3	3.2	3	select c_city, s_city, d_year, sum(lo_revenue) as revenue from customer join lineorder on lo_custkey = c_customerkey join supplier on lo_suppkey = s_suppkey join dim_date on lo_orderdatekey = d_datekey where c_nation = 'UNITED STATES' and s_nation = 'UNITED STATES' and d_year >= 1992 and d_year <= 1997 group by c_city, s_city, d_year order by d_year asc, revenue desc;
3	3.3	3	select c_city, s_city, d_year, sum(lo_revenue) as revenue from customer join lineorder on lo_custkey = c_customerkey join supplier on lo_suppkey = s_suppkey join dim_date on lo_orderdatekey = d_datekey where (c_city='UNITED KI1' or c_city='UNITED KI5') and (s_city='UNITED KI1' or s_city='UNITED KI5') and d_year >= 1992 and d_year <= 1997 group by c_city, s_city, d_year order by d_year asc, revenue desc;
3	3.4	3	select c_city, s_city, d_year, sum(lo_revenue) as revenue from customer join lineorder on lo_custkey = c_customerkey join supplier on lo_suppkey = s_suppkey join dim_date on lo_orderdatekey = d_datekey where (c_city='UNITED KI1' or c_city='UNITED KI5') and (s_city='UNITED KI1' or s_city='UNITED KI5') and d_yearmonth = 'Dec1997' group by c_city, s_city, d_year order by d_year asc, revenue desc;
4	4.1	4	select d_year, c_nation, sum(lo_revenue - lo_supplycost) as profit from lineorder join dim_date on lo_orderdatekey = d_datekey join customer on lo_custkey = c_customerkey join supplier on lo_suppkey = s_suppkey join part on lo_partkey = p_partkey where c_region = 'AMERICA' and s_region = 'AMERICA' and (p_mfgr = 'MFGR#1' or p_mfgr = 'MFGR#2') group by d_year, c_nation order by d_year, c_nation;
4	4.2	4	select d_year, s_nation, p_category, sum(lo_revenue - lo_supplycost) as profit from lineorder join dim_date on lo_orderdatekey = d_datekey join customer on lo_custkey = c_customerkey join supplier on lo_suppkey = s_suppkey join part on lo_partkey = p_partkey where c_region = 'AMERICA' and s_region = 'AMERICA' and (d_year = 1997 or d_year = 1998) and (p_mfgr = 'MFGR#1' or p_mfgr = 'MFGR#2') group by d_year, s_nation, p_category order by d_year, s_nation, p_category;
4	4.3	4	select d_year, s_city, p_brand, sum(lo_revenue - lo_supplycost) as profit from lineorder join dim_date on lo_orderdatekey = d_datekey join customer on lo_custkey = c_customerkey join supplier on lo_suppkey = s_suppkey join part on lo_partkey = p_partkey where s_nation = 'UNITED STATES' and (d_year = 1997 or d_year = 1998) and p_category = 'MFGR#14' group by d_year, s_city, p_brand order by d_year, s_city, p_brand;

Fonte: adaptado de (O'NEIL et al., 2009).

2.4 CONSIDERAÇÕES

Neste capítulo, introduzimos os conceitos e características básicas sobre os bancos de dados NewSQL, descrevendo os métodos de distribuição e as formas de armazenamento de

dados. Na sequência, apresentamos pontos de destaque do banco de dados MemSQL, trazendo detalhes de seu funcionamento. Após isso, apresentamos conceitos acerca de DWs, sua composição, seus esquemas de dados e operações de consultas mais comuns. Por fim, realizamos uma breve explanação sobre os conceitos envolvendo o benchmark SSB. No próximo capítulo apresentaremos os trabalhos relacionados a esta pesquisa.

3 REVISÃO DE LITERATURA

Neste capítulo, apresentamos trabalhos que analisam o desempenho de bancos de dados NewSQL. Inicialmente, descrevemos o método adotado para identificar estudos relacionados à nossa proposta e os critérios utilizados para compará-los (Seção 3.1). Na sequência, apresentamos os trabalhos encontrados na literatura, descrevendo os elementos testados e seus resultados (Seção 3.2). Em seguida, analisamos esses trabalhos de forma conjunta (Seção 3.3). Por fim, fazemos as considerações finais do capítulo (Seção 3.4).

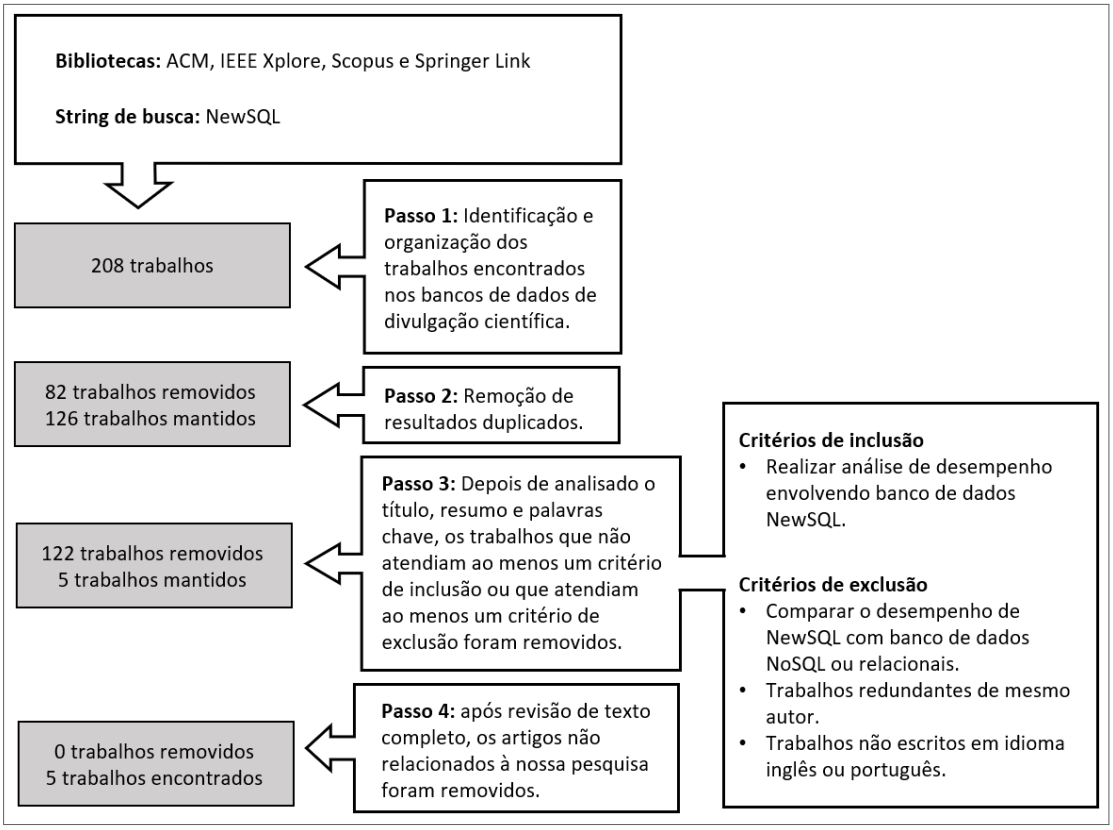
3.1 MÉTODO DE BUSCA E CRITÉRIOS DE ANÁLISE

Na Figura 6, apresentamos as etapas de identificação de trabalhos relacionados à nossa pesquisa. Inicialmente, definimos uma string de busca para ser executada nas bibliotecas ACM Digital Library, IEEE Xplore, Scopus e Springer Link. No passo 1, buscamos a string "NewSQL" nos títulos e resumos dos trabalhos, onde pudemos identificar 208 trabalhos. No passo 2, removemos os trabalhos duplicados. No passo 3, realizarmos análise do título, resumo e palavras chave dos trabalhos restantes, onde pudemos remover os estudos que satisfaziam ao menos um critério de exclusão ou que não satisfaziam ao critério de inclusão. No passo 4, realizamos a leitura completa dos trabalhos, onde selecionamos 5 pesquisas relacionadas.

Para comparar os trabalhos relacionados, adotamos critérios com base nos recursos experimentados pelos autores, que são:

- **Realização de testes em cluster:** armazenamento e processamento de bases de dados em cluster é uma característica central dos SGBDs NewSQL, conferindo bom desempenho mesmo diante grandes volumes de dados.
- **Realização de testes em DW:** como as bases de dados transacionais e analíticas apresentam diferenças de desempenho em operações de leitura e escrita de dados, consideramos importante apontar se os experimentos foram realizados com DW ou não.
- **Experimento com diferentes esquemas de dados:** o objetivo deste critério é destacar se os estudos realizaram variações na redundância de dados em NewSQL, observando seus possíveis impactos advindos de mudanças no esquema.

Figura 6 – Etapas da pesquisa bibliográfica



Fonte: Elaborado pelo autor (2021)

- **Testes com diferentes métodos de distribuição:** para destacar se os estudos abordaram ou não os métodos de distribuição suportados pelos SGBDs NewSQL e suas possíveis variações de desempenho.
- **Testes com diferentes formas de armazenamento:** para destacar se os estudos abordaram ou não as formas de armazenamento suportadas pelos SGBDs NewSQL.
- **Diferentes volumes de dados:** destacar se os estudos realizaram variação no volume de dados e observando suas possíveis alterações de desempenho.

3.2 TRABALHOS RELACIONADOS

A Tabela 2 apresenta os trabalhos selecionados após a realização das etapas ilustradas na Figura 6. Esses trabalhos são abordados nas próximas subsessões.

Tabela 2 – Lista de trabalhos relacionados a esta pesquisa

Nº	Título	Referência
1	NewSQL Databases - MemSQL and VoltDB Experimental Evaluation	Oliveira e Bernardino (2017)
2	NewSQL Through the Looking Glass	Schreiner et al. (2019)
3	Performance Evaluation of NewSQL Databases	Kaur e Sachdeva (2017)
4	Performance Benchmarking of NewSQL Databases with Yahoo Cloud Serving Benchmark	Astrova et al. (2021)
5	Uma abordagem para modelagem de desempenho para SGBDs NewSQL e comparação de modelos gerados com diferentes técnicas de aprendizado de máquina	Cá (2018)

Fonte: Elaborado pelo autor (2021)

3.2.1 NewSQL Databases - MemSQL and VoltDB Experimental Evaluation

Este trabalho apresenta uma análise de desempenho em DWs com os bancos de dados MemSQL e VoltDB. Para isso, utiliza o benchmark TPC-H, com SF=1 (1GB de dados), em um ambiente experimental composto por uma máquina com 40GB de RAM. O estudo apresenta o tempo de carga em cada banco de dados e seus resultados são apresentados na Tabela 3. O estudo também coleta os tempos médios de execução das consultas do TPC-H, ao qual são apresentados na Figura 7.

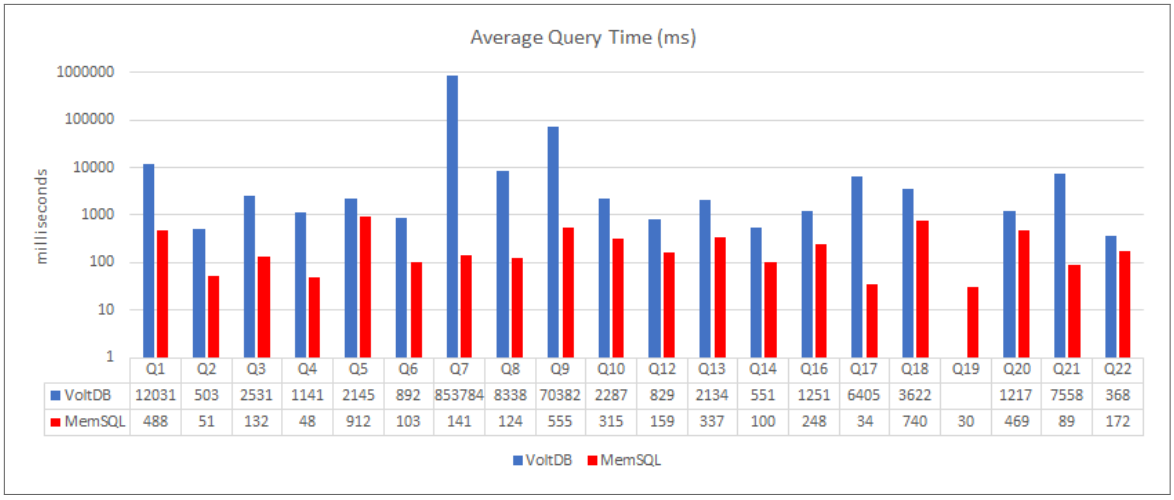
Tabela 3 – Tempo médio de carga com SF=1

Tabelas	VoltDB	MemSQL
Nation	0,32 seg	0,17 seg
Region	0,48 seg	0,18 seg
Part	9,64 seg	0,65 seg
Supplier	0,89 seg	0,35 seg
Partsupp	32,56 seg	0,86 seg
Customer	7,14 seg	0,49 seg
Orders	68,74 seg	2,90 seg
Lineitem	339,71 seg	33,58 seg

Fonte: adaptado de Oliveira e Bernardino (2017)

De acordo com seus resultados, MemSQL consegue melhor desempenho tanto no tempo de carga, quanto no tempo médio de consulta, com economia média de 92% do tempo gasto em relação a VoltDB. Segundo os autores, o motivo do melhor desempenho de MemSQL, na execução de consultas, reside num recurso que armazena resultados de buscas em memória

Figura 7 – Média dos tempos de consulta



	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Q12	Q13	Q14	Q16	Q17	Q18	Q19	Q20	Q21	Q22
VoltDB	12031	503	2531	1141	2145	892	853784	8338	70382	2287	829	2134	551	1251	6405	3622		1217	7558	368
MemSQL	488	51	132	48	912	103	141	124	555	315	159	337	100	248	34	740	30	469	89	172

Fonte: Oliveira e Bernardino (2017)

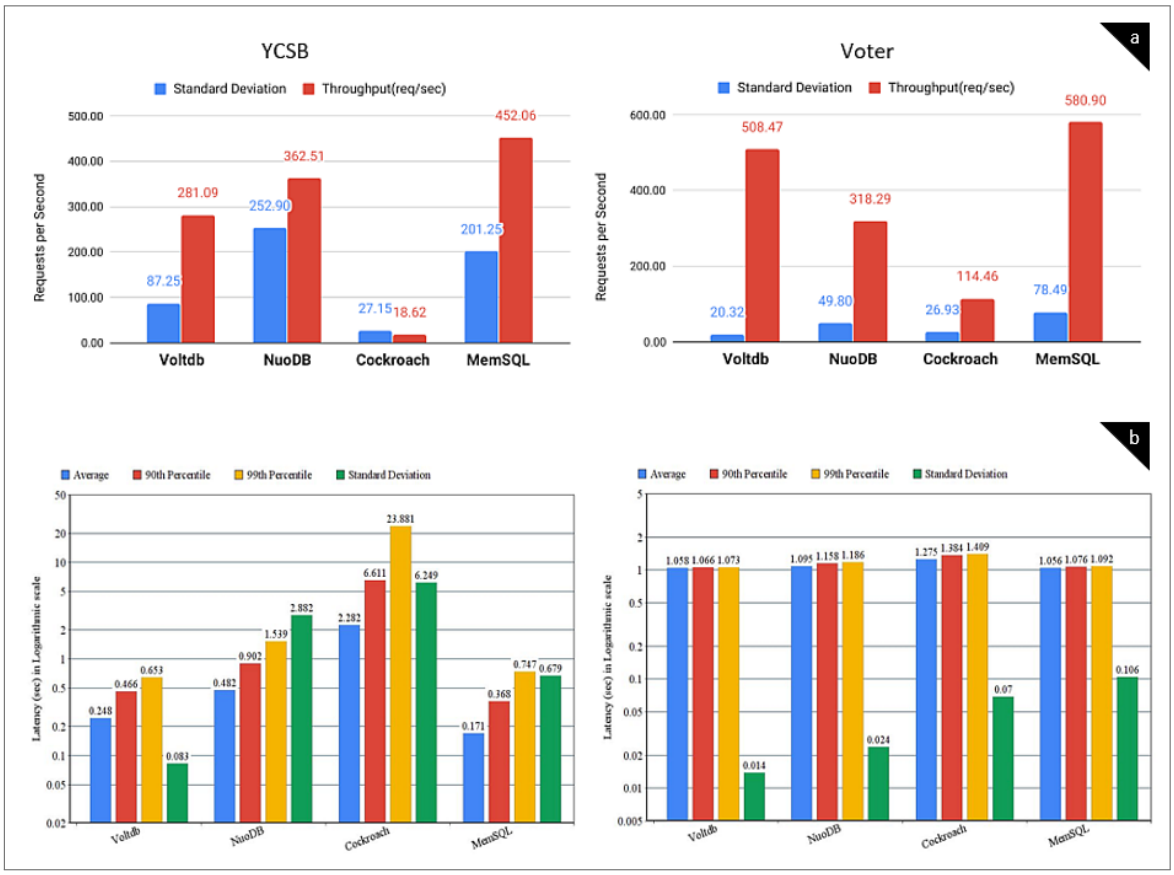
cache, oferecendo economia de tempo para re-execuções. Neste estudo, não há menções que se tenha investigado diferentes métodos de distribuição dos dados, formas de armazenamento, nem faz qualquer menção quanto a isso.

3.2.2 NewSQL Through the Looking Glass

A proposta de Schreiner et al. (2019) apresenta um comparativo de desempenho em bases de dados transacionais, diante 4 distribuições NewSQL: VoltDB, NuoDB, Cockroach e MemSQL. Nessa tarefa, utiliza os benchmarks YCSB e Voter, ambos da suíte OLTP-Bench, com SF=1000, em um ambiente experimental composto por 3 terminais físicos com 24GB de RAM no total. O estudo apresenta a média de transações por segundo e o tempo médio de latência em cada banco de dados (c.f., Figura 8).

Em geral, a experimentação nos dois benchmarks mostram que MemSQL obtém o melhor desempenho nas duas métricas avaliadas (i.e., média de transações por segundo e latência média das transações), com VoltDB e NuoDB se alternando numa segunda posição e Cockroach apresentando os piores resultados. Neste estudo, não há menções que se tenham variado os métodos de distribuição ou as formas de armazenamento dos dados, nem detalhes sobre isso.

Figura 8 – Média de transações por segundo (a) e Latência média das transações (b)



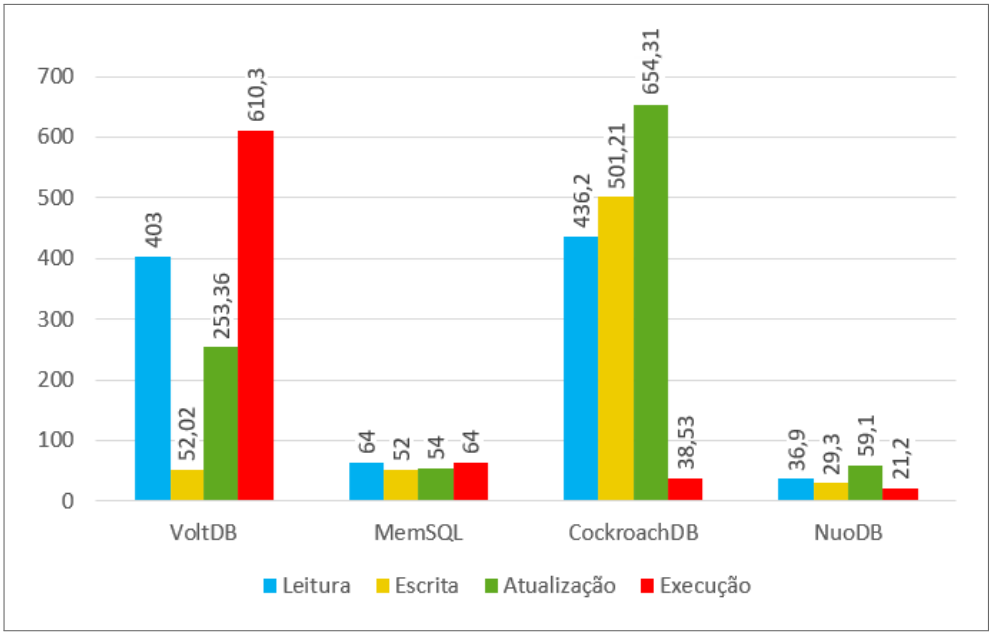
Fonte: adaptado de Schreiner et al. (2019)

3.2.3 Performance Evaluation of NewSQL Databases

O estudo realizado por Kaur e Sachdeva (2017) compara o desempenho de uma base de dados transacional, diante 4 bancos de dados NewSQL: VoltDB, NuoDB, Cockroach e MemSQL. Para isso, analisa os tempos de escrita, leitura e atualização dos dados, com os bancos de dados sendo instalados em um único terminal virtualizado com 4GB de RAM. Seus resultados são mostrados na Figura 9.

De acordo com seus resultados, NuoDB possui o melhor desempenho diante as métricas analisadas, seguido por MemSQL. CockroachDB apresentou os piores desempenhos em quase todos os testes, com exceção da métrica "Tempo de execução", ao qual não há maiores explicações. O trabalho não informa ainda qual a fonte de dados utilizada, nem se houve variação da quantidade de dados. Este é mais um estudo em que não há menções sobre os métodos de distribuição ou as formas de armazenamento aplicadas.

Figura 9 – Resultado das métricas



Fonte: adaptado de Kaur e Sachdeva (2017)

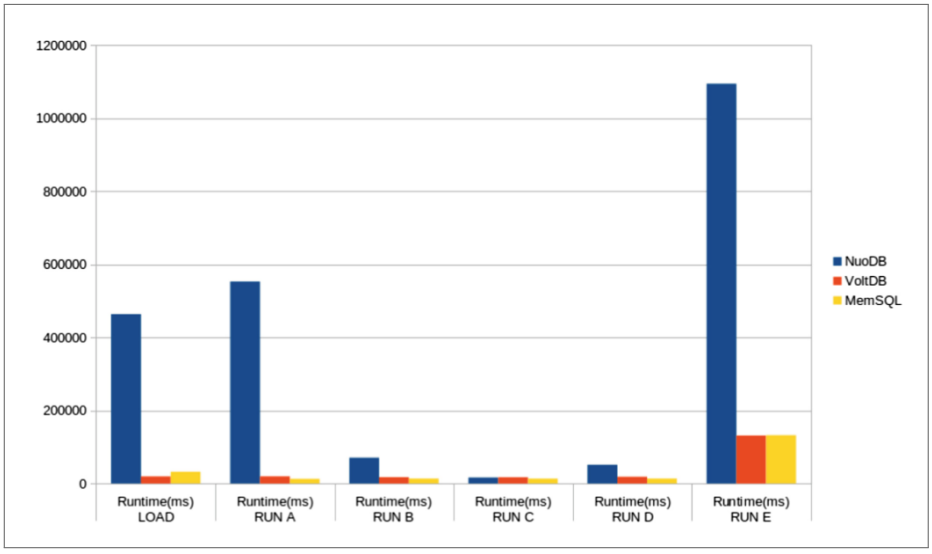
3.2.4 Performance Benchmarking of NewSQL Databases with Yahoo Cloud Serving Benchmark

O trabalho de Astrova et al. (2021) analisa o desempenho dos bancos de dados NuoDB, VoltDB e MemSQL, em bases de dados transacionais. Para a tarefa, utiliza o Yahoo Cloud Serving Benchmark (YCSB) com 1 milhão de registros (i.e., SF=1), testando operações de carga de dados e um conjunto de cargas de trabalho (A-E) que mesclam diferentes níveis de escrita e leitura, conforme listado a seguir.

- A - Atualizações pesadas, com uma mistura de leituras e gravações 50/50;
- B - Principalmente leituras, com uma mistura de 95/5 leituras e gravações;
- C - Somente leitura;
- D - Inserções de registros e leitura em seguida;
- E - Lê em intervalos curtos;

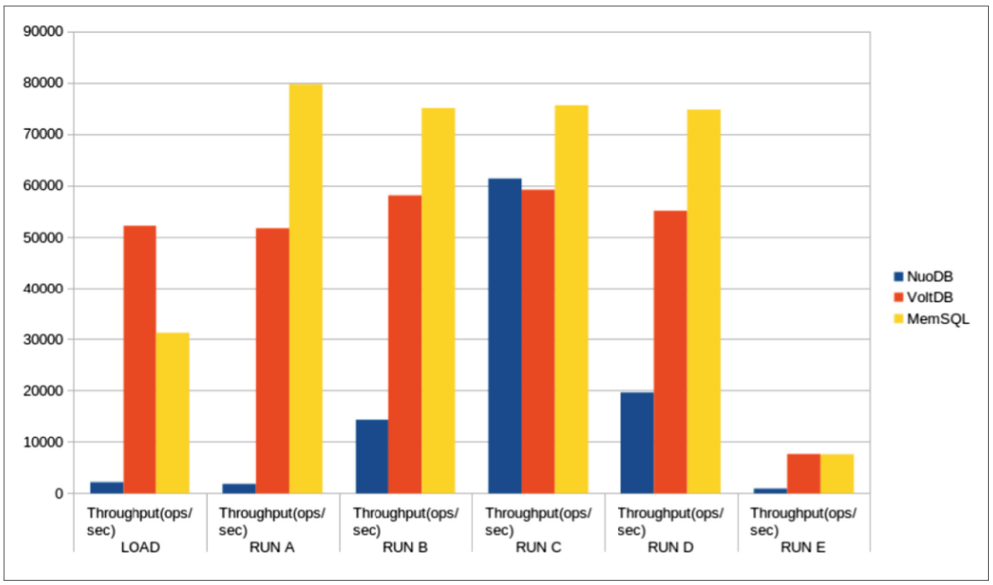
A experimentação foi realizada em um terminal, com 64GB de RAM e tem por métricas, o tempo médio e a média de *throughput* (quantidade de operações por segundo). Os resultados são ilustrados nas Figuras 10 e 11.

Figura 10 – Média dos tempos de cada carga de trabalho por SGBD NewSQL em milissegundos



Fonte: Astrova et al. (2021)

Figura 11 – Média de throughput por SGBD NewSQL em operações por segundo



Fonte: Astrova et al. (2021)

De acordo com os resultados deste estudo, a carga de dados em NuoDB levou 14x e 24x mais tempo que a carga de dados em MemSQL e em VoltDB, respectivamente. Na execução das 5 cargas de trabalho, também houve clara vantagem de MemSQL e VoltDB sob NuoDB. Entre MemSQL e VoltDB, o primeiro levou um pouco mais de tempo para carregar seus dados, entretanto, foi ligeiramente mais rápido na execução de consultas e atualizações. Essa vantagem de MemSQL na carga de trabalho é melhor observada na Figura 11, com praticamente todos seus gráficos apontando maior rendimento sob o VoltDB. NuoDB

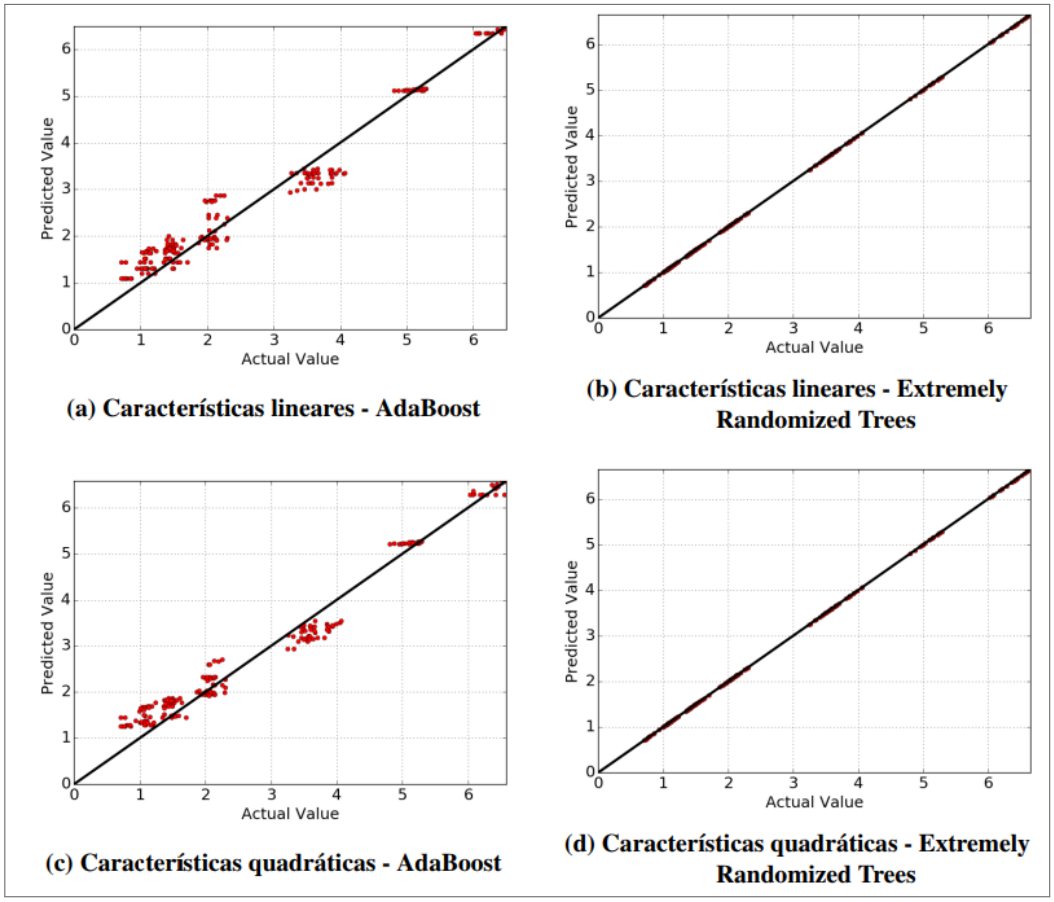
conseguiu equiparar o desempenho em cargas de trabalho com VoltDB e MemSQL, quando a tarefa envolveu a realização de apenas leituras (c.f. Figura 11C), podendo ser útil em contexto de análise de dados. Neste estudo, não há variação na quantidade de dados, nos métodos de distribuição, nem nas formas de armazenamento empregadas.

3.2.5 Uma abordagem para modelagem de desempenho para sistemas de banco de dados NewSQL e comparação de modelos gerados com diferentes técnicas de aprendizado de máquina

Diferentemente dos trabalhos anteriores, o objetivo deste é "propor uma solução de modelagem de desempenho para encontrar a técnica de aprendizado de máquina, que mais se adequada à tarefa de predição do comportamento de cargas de trabalho em SGBDs NewSQL". Nesta tarefa, utiliza o benchmark YCSB (com 360.000 tuplas) com 40 *threads* de clientes para as análises de desempenho em base de dados transacionais e ambiente experimental composto por um servidor com 64GB de RAM, testando 200 diferentes combinações de cargas de trabalho, baseadas em operações de leitura, escrita, atualização e varredura de intervalos. Para a modelagem de desempenho, utilizou as métricas: número de requisições executadas por segundo (vasão)(RPS), tempo de resposta médio das requisições executadas (TRS) e quantidade de violações ao serviço por segundo (VPS). De acordo com o autor, estas métricas permitem observar aspectos relativos à complexidade das consultas, concorrência e distribuição dos dados. Em aprendizagem de máquina, a comparação foi realizada com os métodos *AdaBoost* (AB), *Extremely Randomized Trees* (ETR), Regressão Linear (RL) e *Gradient Boosting Machine* (GBM), considerando a observância de características lineares (CL) e quadráticas (CQ). Seus resultados são exibidos através da métrica de erro R^2 com *grid-search*, onde valores próximos a 1 indicam maior acurácia dos métodos de predição. A Figura 12 exemplifica alguns de seus resultados e a Figura 13 exibe o desempenho de cada método de aprendizagem de máquina testado.

De acordo com seus resultados, o autor demonstra que o método AB obteve melhor capacidade preditiva com características quadráticas e que no método ETR não há diferença entre uso de características quadráticas ou lineares, recomendando este último por ser uma abordagem mais simples. Ao comparar esses dois métodos com os métodos RL e GBM, observou o autor que é mais vantajoso usar o método AB em relação ao RL, devido aos melhores resultados obtidos em AB e entre os métodos ETR e GBM, recomendou usar qualquer dos

Figura 12 – Métrica RPS: relação entre valor predito e valor real para os modelos gerados



Fonte: Cá (2018)

Figura 13 – Métricas TRS, PRS e VPS: Pontuação R2 obtido com grid-search e 10-fold para características lineares e quadráticas com métodos RL e GBM (a) e com métodos AB e ERT (b)

		TRS	RPS	VPS
RL	CL	0.74	0.63	0.94
	CQ	0.89	0.82	0.99
GBM	CL	0.98	0.98	0.98
	CQ	0.97	0.97	0.99

(a)

		TRS	RPS	VPS
AB	CL	0.91	0.93	0.98
	CQ	0.92	0.94	0.99
ERT	CL	0.97	0.98	0.99
	CQ	0.96	0.98	0.99

(b)

Fonte: Cá (2018)

métodos pela proximidade de resultados e ainda são mais eficientes em relação a RL e AB.

Neste estudo, não há indicação de quais recursos NewSQL foram utilizados para implementar as bases de dados dos experimentos e também não ficou claro o que significa a variação no tamanho do servidor, numa possível referência à escalabilidade vertical.

3.3 ANÁLISE GERAL

Após realizada a avaliação individual dos trabalhos, podemos observar seus destaques de forma conjunta na Tabela 4, ao qual indica o atendimento ou não em relação aos critérios da Seção 3.1.

Tabela 4 – Avaliação consolidada dos trabalhos relacionados

Critério	Oliveira e Bernardino (2017)	Schreiner et al. (2019)	Kaur e Sachdeva (2017)	Astrova et al. (2021)	Cá (2018)
Realizou testes em cluster?	Não	Sim	Não	Não	Não
Utilizou um DW?	Sim	Não	Não	Não	Não
Experimentou diferentes esquemas de dados?	Não	Não	Não	Não	Não
Testou diferentes métodos de distribuição?	Não	Não	Não	Não	Não
Avaliou várias formas de armazenamento?	Não	Não	Não	Não	Não
Testou diferentes volumes de dados?	Não	Não	Não	Não	Não

Fonte: Elaborado pelo autor (2021)

De acordo com a Tabela 4, apenas o trabalho de Schreiner et al. (2019) utilizou cluster computacional. Outro ponto de destaque é que apenas o trabalho de Oliveira e Bernardino (2017) avaliou NewSQL no contexto de DWs, mas sem revelar detalhes de recursos NewSQL empregados nas suas bases. Vale destacar que todas as pesquisas trabalharam com quantidades fixas de dados, não permitindo observar mudanças de comportamento ante o crescimento da base de dados. Além disso, nenhum dos trabalhos realizou variação no esquema de dados, no método de distribuição, nem na forma de armazenamento. Dessa forma, pretendemos analisar o desempenho de DWs em NewSQL, combinando diferentes esquemas de dados, métodos de distribuição, formas de armazenamento, gerando assim diferentes cenários. Além disso, propomos avaliar tais cenários a partir de diferentes volumes de dados, em ambiente de cluster computacional, o que difere dos demais estudos realizados.

3.4 CONSIDERAÇÕES

Neste capítulo, expomos critérios de avaliação para classificar os trabalhos selecionados na etapa de pesquisa bibliográfica. Apresentamos também a metodologia de pesquisa para selecionar trabalhos relacionados, promovendo a análise dos mesmos. Por fim, realizamos uma comparação entre os trabalhos para entender o que fora feito até então e observar possíveis lacunas de pesquisa. No próximo capítulo descrevemos a proposta de experimento deste estudo.

4 EXPERIMENTO

Neste capítulo, apresentamos a configuração do ambiente experimental, as configurações empregadas no SGBD MemSQL e as métricas coletadas (Seção 4.1). Na sequência, apresentamos os passos adotados para executar o experimento (Seção 4.2). Por fim, fazemos as considerações finais do capítulo (Seção 4.3).

4.1 DEFINIÇÕES DO EXPERIMENTO E DO AMBIENTE DE AVALIAÇÃO

Para realizar o armazenamento de dados e a execução de consultas em DWs, utilizamos o SGBD MemSQL, o qual foi instalado em um cluster formado por 3 máquinas conectadas em uma rede local, com interface de rede 1000BASE-T. Cada máquina configurada com processador Xeon E5410 2,33GHz com 8 threads, 48GB de memória RAM (frequência 2400MHz) e SSD de 240GB. Todas as máquinas possuíam sistema operacional CentOS 7, além do MemSQL 7.1.3 e Python 2.7.5 (para executar scripts do experimento e coletar os resultados). Na configuração do MemSQL, uma das máquinas assumiu o papel de agregador master, enquanto as outras duas foram configuradas como nós folha. Não foi definido nó agregador para esta experimentação, dada a quantidade reduzida de nós folha disponível e a melhor desempenho em testes preliminares de carga e consulta. A quantidade de memória RAM no cluster foi de 45000MB no agregador master e 41500MB em cada um dos nós folha, totalizando 128000MB.

No experimento, avaliamos o desempenho de DWs NewSQL combinando diferentes abordagens de esquemas, formas de armazenamento, métodos de particionamento e fatores de escala (SF) (c.f., Tabela 5). Diante desse contexto, avaliamos o desempenho de diferentes níveis de redundância de dados, recursos NewSQL (distribuição e armazenamento) e volume de dados.

Tabela 5 – Combinações de abordagens para o experimento

Fatores	Abordagens
Esquema	Estrela e Flat
Forma de armazenamento	Columnstore e Rowstore
Métodos de particionamento	Round-robin, Hash e Replicação
Fator de escala (SF)	1, 10 e 100

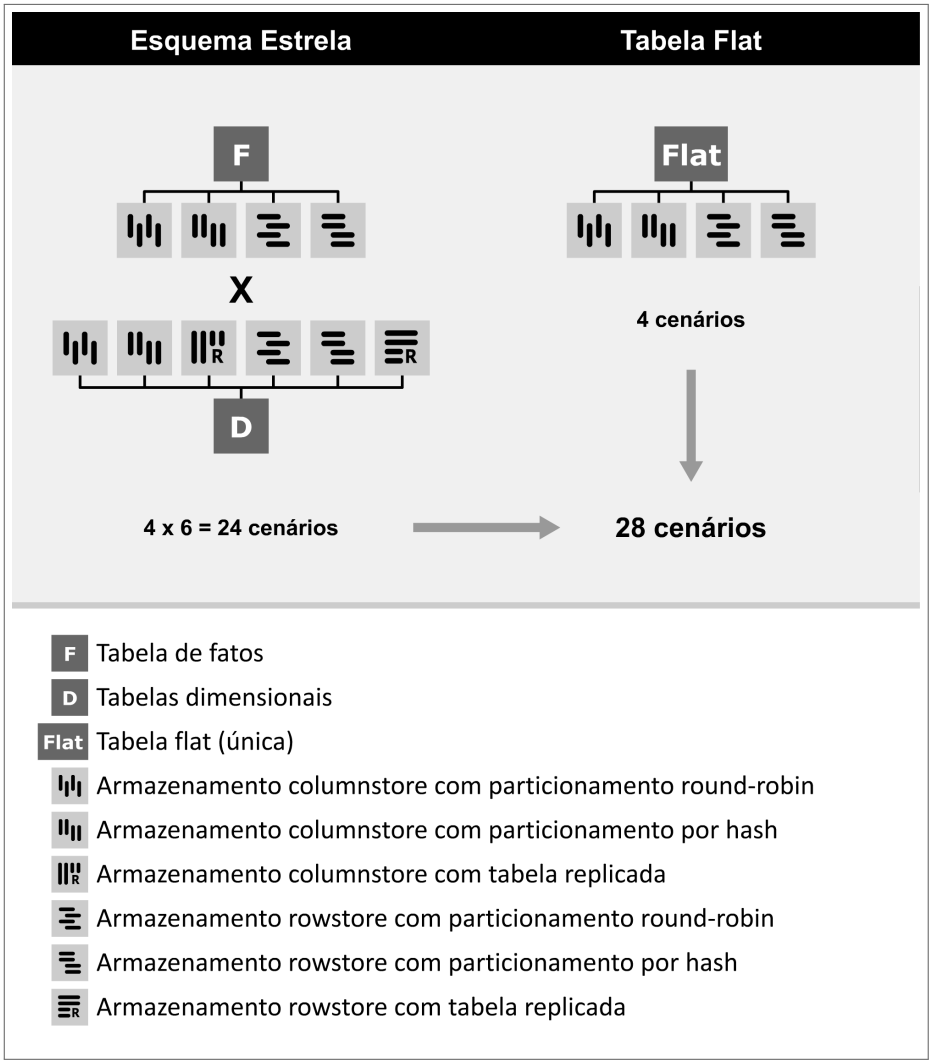
Fonte: Elaborado pelo autor (2021)

Para a tabela de fatos e o conjunto das tabelas de dimensões, adotamos independência nas configurações de armazenamento e distribuição, por apresentarem natureza descritiva distintas (i.e., tabela de fatos registrando ocorrência de eventos e tabelas de dimensão descrevendo tais eventos). Assim, as configurações de armazenamento e distribuição dadas a uma tabela de fatos podem, em um dado cenário, ser diferentes das configurações dadas às tabelas de dimensão. Por fim, não avaliamos a estratégia de particionamento *range*, por não haver suporte em MemSQL.

Na escolha da chave de particionamento hash (i.e., SK) e da chave columnstore em cada tabela, utilizamos o conceito de alta ou baixa seletividade de atributos (i.e., campos). A alta seletividade diz respeito à capacidade que um atributo tem de selecionar poucos registros em uma dada consulta. Por exemplo, o atributo "CPF", quando instanciado numa consulta (e.g., where cpf = "123456789012"), retorna apenas 1 registro da tabela de clientes. A baixa seletividade é justamente o contrário, sendo a capacidade que um campo tem de selecionar muitos registros em uma dada consulta. Por exemplo, o atributo "Sexo", quando instanciado numa consulta (e.g., where sexo = "F"), retorna grande quantidade de registros na mesma tabela de clientes. Dito isto, para a escolha da SK de cada tabela, optamos por campos com seletividade mediana, para atender 2 objetivos: balancear a quantidade de tuplas de cada nó e manter a proposta do particionamento por hash de agrupar tuplas com alguma similaridade de valores. Já para a escolha da chave columnstore, optamos por campos de baixa seletividade, visando aumentar o poder de compactação em campos com grande repetição de valores. A escolha dessas chaves pode envolver outros fatores (e.g., campos mais utilizados, hierarquias e combinação de chaves), no entanto, a investigação sobre os melhores campos para SK e chave columnstore não faz parte do escopo do presente trabalho. Por fim, decidimos não replicar tabelas de fatos, nem tabelas flat, por possuírem o maior volume de dados em cada cenário. Replicar grandes tabelas ocasionaria aumento substancial no volume de dados e comprometeria a escalabilidade horizontal do cluster.

A Figura 14 ilustra o cruzamento entre as características avaliadas no experimento, que resulta em 28 diferentes cenários de DW NewSQL. Os ícones utilizados nesta figura representam, graficamente, os conceitos de distribuição e armazenamento.

Figura 14 – Cruzamento das configurações



Fonte: Elaborado pelo autor (2021)

No Quadro 2, detalhamos as configurações de cada cenário (e.g., E01, E02, etc.). Tanto a Figura 14, quanto o Quadro 2 desconsideram o SF, por não representarem mudanças no design dos cenários. Cabe destacar, que mantemos as mesmas configurações de distribuição e armazenamento para as tabelas de dimensões, dentro de cada cenário. Por exemplo (c.f., Quadro 2), o cenário E07 tem todas as tabelas dimensionais armazenadas em columnstore, com distribuição por round-robin e o cenário E17 tem suas tabelas dimensionais armazenadas em rowstore, com distribuição por hash.

Quadro 2 – Configuração dos cenários experimentados em cada SF (1, 1 e 100) (F - Tabela de Fatos, D - tabelas dimensionais e Flat - Tabela flat (ou tabela única))

Cenário			R			R
E01	F , D					
E02	F	D				
E03	F		D			
E04	F			D		
E05	F				D	
E06	F					D
E07	D	F				
E08		F , D				
E09		F	D			
E10		F		D		
E11		F			D	
E12		F				D
E13	D			F		
E14		D		F		
E15			D	F		
E16				F , D		
E17				F	D	
E18				F		D
E19	D				F	
E20		D			F	
E21			D		F	
E22				D	F	
E23					F , D	
E24					F	D
E25	Flat					
E26		Flat				
E27				Flat		
E28					Flat	

Fonte: Elaborado pelo autor (2021)

Os DWs dos cenários E01 ao E24 utilizam o esquema estrela, enquanto que os DWs dos cenários E25 ao E28 utilizam tabela flat. Assim, identificamos a necessidade de realizar 84 experimentos individuais, que correspondem aos 28 cenários do Quadro 2 experimentados nos 3 SFs utilizados (i.e., 1, 10 e 100).

Durante a realização do experimento, coletamos as seguintes métricas para cada cenário de DW NewSQL: (i) o tempo da carga de dados, (ii) o volume de dados armazenado em disco e em memória RAM, (iii) o tempo médio de execução das 13 consultas do SSB e (iv) o consumo máximo de memória RAM durante a execução das consultas.

Como este estudo apresenta um caráter exploratório, optamos por utilização de elementos da estatística descritiva, com utilização da média dos resultados obtidos. Especificamente na métrica de tempo médio de consulta de dados, utilizamos a recomendação de Hogg (2013) de 30 amostras para o cálculo de média e coeficiente de variação máximo de 10%, que segundo Gomes (2000), representa baixa variabilidade amostral.

4.2 EXECUÇÃO DO EXPERIMENTO

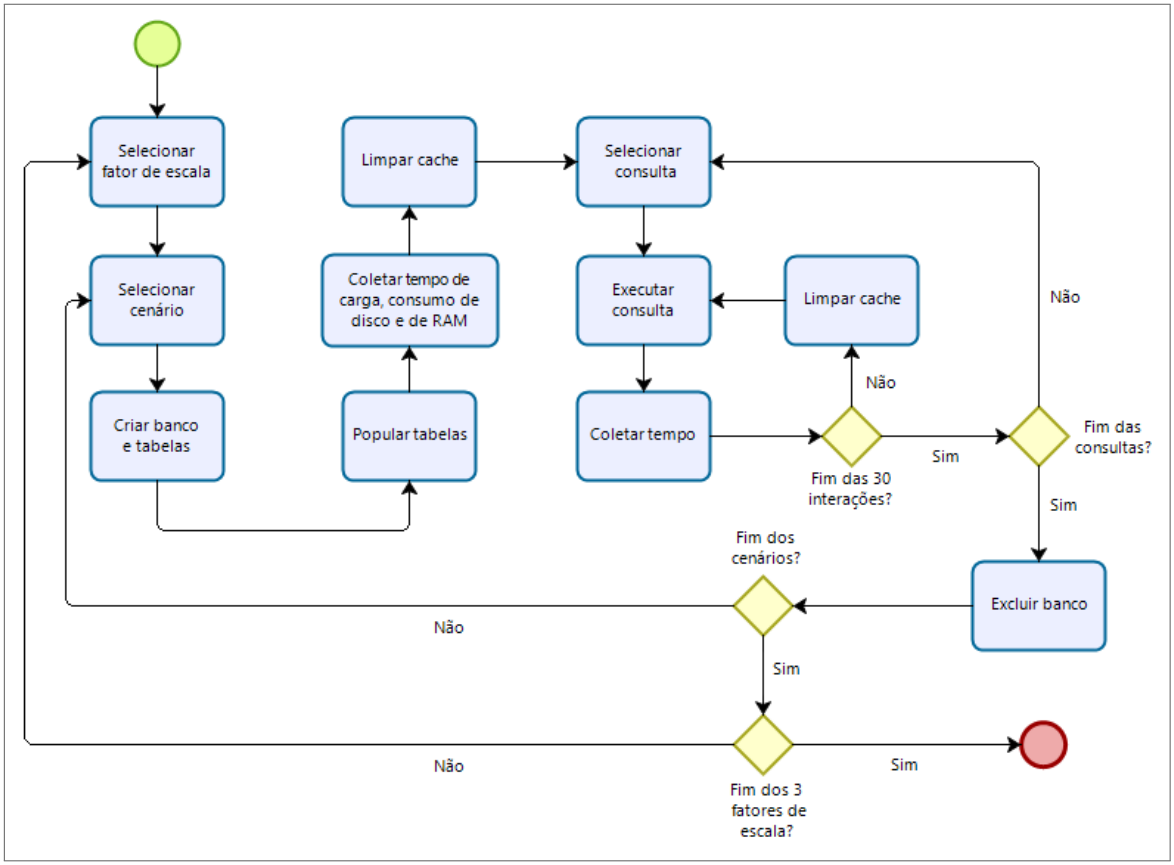
Inicialmente, utilizamos o software DBGEN[1] para gerar os dados do SSB nos SFs 1, 10, e 100. Como os dados gerados no DBGEN são estruturados de acordo com o esquema estrela, utilizamos o Pentaho Data Integration[2] para transformar esses dados e adequá-los para tabelas flat. Além disso, adaptamos as consultas do SSB (c.f., Quadro 1) para os cenários em tabela flat. As adaptações realizadas consistem em reescrever as consultas de modo a remover as junções entre tabelas e ajustar os nomes dos campos. Os códigos para implementação dos cenários podem ser obtidos no repositório online deste projeto[3]. Os códigos de execução das etapas do projeto também podem ser encontrados no Apêndice A.

Ilustramos na Figura 15, as etapas de execução do experimento com MemSQL. Como pode ser observado nesta figura, iniciamos pela seleção de um fator de escala e um dos 28 cenários. Após isso, criamos as tabelas obedecendo as configurações do cenário selecionado, que combina os diferentes esquemas de dados, métodos de distribuição e formas de armazenamento utilizadas (c.f., Quadro 2). Em seguida, populamos as tabelas de acordo com o fator de escala, e ao final disto, coletamos o tempo de carga e o volume de dados armazenado (disco e memória RAM). Após essas coletas, realizamos a limpeza de cache do sistema e do MemSQL, onde eliminamos dados temporários da etapa de carga. Na etapa de leitura de dados, executamos as 13 consultas do SSB, coletando o tempo médio de resposta para cada uma das consultas. Executamos 30 vezes cada consulta, efetuando limpeza de cache no banco de dados, entre essas execuções, para evitar contaminação nas coletas seguintes. Ao final das

[1] https://github.com/vadimtk/ssb-dbgen
[2] https://sourceforge.net/projects/pentaho/
[3] Projeto em https://github.com/alesancoml/testeDesempenhoMemsql/

30 execuções de uma consulta, calculamos a média e o desvio padrão. De forma paralela à execução das consultas, também coletamos o consumo de memória RAM nos nós do cluster, a cada 1 segundo. Concluída a etapa de leitura de dados, excluímos o DW do cluster a fim de preservar as mesmas condições de partida para cada experimentação de cenário. Todas as máquinas do cluster foram utilizadas exclusivamente para a realização dos experimentos. Ou seja, não houve interferências externas que pudessem prejudicar as métricas coletadas.

Figura 15 – Etapas de execução do experimento



Fonte: Elaborado pelo autor (2021)

4.3 CONSIDERAÇÕES

Neste capítulo, definimos a composição do ambiente experimental, as abordagens a serem avaliadas e quantitativo de dados experimentados. Detalhamos também o método para escolha de chaves de particionamento e de armazenamento, bem como a composição de 28 cenários para testes. Em seguida, definimos métricas de desempenho e estatística utilizada. Por fim, detalhamos os passos executados na experimentação dos cenários. No próximo capítulo, analisamos os resultados obtidos.

5 RESULTADOS

Neste capítulo, apresentamos e discutimos os resultados obtidos. Analisamos o tempo para a carga de dados (Seção 5.1) e o volume de dados armazenado em disco e em RAM (Seção 5.2) nos cenário testados. Na sequência, discutimos os resultados de tempo médio na execução de consultas (Seção 5.3) e do consumo de memória RAM durante as consultas (Seção 5.4). Após isso, os principais resultados de DWs em MemSQL são apresentados de forma conjunta (Seção 5.5). Por fim, realizamos as considerações finais (Seção 5.6).

5.1 CARGA DE DADOS - TEMPO

No Quadro 3, exibimos os tempos de carga de dados, para cada cenário, em SF 1, 10 e 100. É importante ressaltar que em SF=100, alguns dos cenários (i.e., E13-E24 e E27-E28) apresentaram erro durante a etapa de carga, por memória insuficiente. Isso ocorreu devido à quantidade de memória RAM necessária para realizar a carga de dados nesses cenários ter superado 83000 MB, a quantidade de memória RAM disponível nos nós folha neste experimento.

Conforme podemos ver no Quadro 3, a redundância de dados prejudicou o tempo de carga. Carregar dados em tabelas flat necessitou de 4 a 9 vezes mais tempo que no esquema estrela. Esse comportamento pode ser melhor observado quando comparados os tempos de carga nos cenários E25, E26, E27 e E28 (tabela flat), com os cenários E01, E08, E16 e E23 (esquema estrela), que compartilham as mesmas configurações de distribuição e armazenamento. Observamos também que o aumento na redundância por uso de replicação em tabelas dimensionais impactou no tempo de carga, com acréscimo de tempo de, aproximadamente, 24%, 10% e 4%, em SF=1, SF=10 e SF=100, respectivamente. Esse comportamento pode ser observado, por exemplo, quando comparamos os tempos de carga de E03 (que possui dimensões em tabelas replicadas) com os tempos de carga de E01 e E02 (que possuem dimensões em tabelas particionadas).

A respeito das formas de armazenamento, verificamos que cenários com tabela de fatos em rowstore levaram menos tempo para a carga de dados, em comparação a cenários com tabela de fatos em columnstore. Em SF=10 (maior SF com resultados em todos os cenários), cenários de tabela de fatos em rowstore (i.e., E13-E24) levaram, em média, 5,8% menos tempo de carga que os cenários de tabela de fatos em columnstore (E01-E12). Essa vantagem de

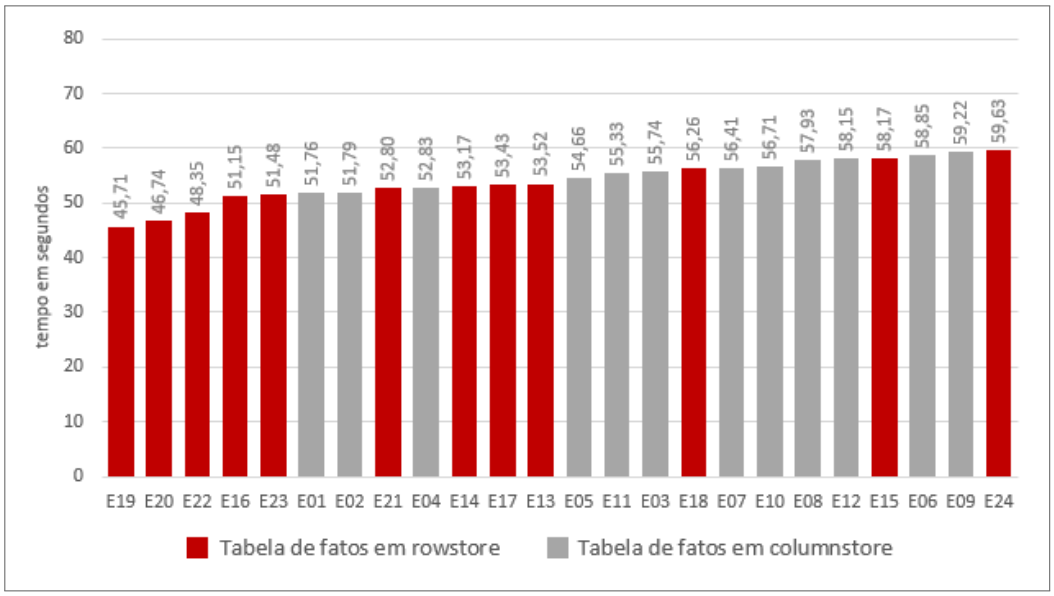
Quadro 3 – Tempo de carga em SF = 1, 10 e 100

Cenário							Tempo em segundos		
							SF1	SF10	SF100
E01	F , D						5,46	51,76	735,17
E02	F	D					5,58	51,79	720,28
E03	F		D				6,88	55,74	762,35
E04	F			D			5,41	52,83	719,06
E05	F				D		5,44	54,66	706,33
E06	F					D	6,44	58,85	747,81
E07	D	F					5,48	56,41	716,39
E08		F , D					5,60	57,93	722,44
E09		F	D				6,94	59,22	739,07
E10		F		D			5,53	56,71	715,25
E11		F			D		5,43	55,33	732,04
E12		F				D	6,64	58,15	740,39
E13	D			F			5,27	53,52	-
E14		D		F			5,43	53,17	-
E15			D	F			6,65	58,17	-
E16				F , D			5,23	51,15	-
E17				F	D		5,37	53,43	-
E18				F		D	6,35	56,26	-
E19	D				F		5,52	45,71	-
E20		D			F		5,49	46,74	-
E21			D		F		6,84	52,80	-
E22				D	F		5,45	48,35	-
E23					F , D		5,47	51,48	-
E24					F	D	6,78	59,63	-
E25	Flat						25,36	238,97	3588,88
E26		Flat					25,43	261,90	3728,29
E27				Flat			23,04	461,77	-
E28					Flat		22,89	414,76	-

Fonte: Elaborado pelo autor (2021)

rowstore sob columnstore, no tempo de carga, é facilmente observável quando ordenamos os resultados de SF=10 (c.f., Figura 16). Dentre os 12 melhores tempos de carga de dados, 9 deles tinham tabela de fatos com armazenamento em rowstore. Para tabelas dimensionais, não identificamos diferenças significativas nos tempos de carga entre cenários com dimensionais em columnstore ou em rowstore.

Figura 16 – Tempo ordenado de carga em cenários com esquema estrela (SF=10)



Fonte: Elaborado pelo autor (2021)

Entre cenários com tabela flat, não ficou claro se há vantagem no tempo de carga quando combinados com columnstore ou rowstore. Ao comparar os cenários E27 e E28 (flat em rowstore) com os cenários E25 e E26 (flat em columnstore), identificamos que os dois primeiros tiveram menor tempo de carga em SF=1, enquanto os dois últimos tiveram melhor tempo de carga em SF=10. Além disso, não identificamos diferenças significativas nos tempos de carga quando comparadas as estratégias de particionamento de dados round-robin e hash, para qualquer situação. Em resumo, temos:

- Para grandes massas de dados (e.g., SF=100), utilizar armazenamento rowstore (em RAM) com tabela de fatos ou com tabela flat pode ocasionar problemas na etapa de carga.
- Redundância em tabelas flat ou tabelas replicadas prejudica o tempo de carga.
- Cenários com tabela de fatos com rowstore foram, em média, 5,8% mais rápidos no tempo de carga em relação a cenários com tabela de fatos com columnstore.

5.2 CARGA DE DADOS - VOLUME DE DADOS

No Quadro 4, exibimos os resultados do volume de dados armazenado em disco (MB) e em memória RAM (MB), nos SFs 1, 10 e 100.

Quadro 4 – Volume de dados armazenado em disco e RAM - SF 1, 10 e 100

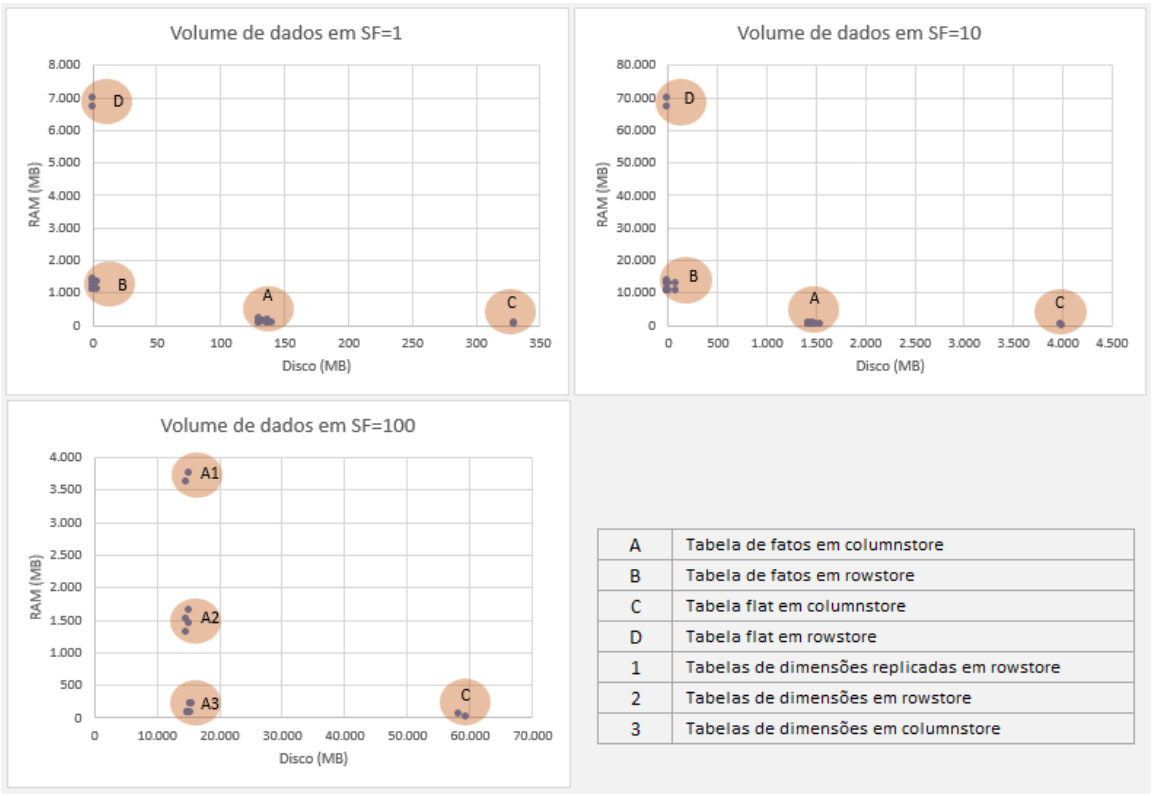
Cenário							Disco (MB)			RAM (MB)		
			R			R	SF1	SF10	SF100	SF1	SF10	SF100
E01	F, D						137	1467	15335	62	298	210
E02	F	D					137	1471	15330	63	194	210
E03	F		D				141	1543	15777	76	45	210
E04	F			D			137	1467	15102	62	298	1437
E05	F				D		137	1467	15102	71	341	1640
E06	F					D	137	1467	15102	163	808	3744
E07	D	F					130	1432	14917	92	331	76
E08		F, D					130	1436	14913	92	227	76
E09		F	D				134	1508	15360	106	77	76
E10		F		D			130	1432	14684	92	331	1304
E11		F			D		130	1432	14684	101	373	1507
E12		F				D	130	1432	14684	192	840	3611
E13	D			F			0	0	-	1094	10618	-
E14		D		F			0	4	-	1095	10514	-
E15			D	F			4	76	-	1108	10364	-
E16				F, D			0	0	-	1094	10618	-
E17				F	D		0	0	-	1103	10660	-
E18				F		D	0	0	-	1195	11127	-
E19	D				F		0	0	-	1333	13000	-
E20		D			F		0	4	-	1333	12895	-
E21			D		F		4	76	-	1346	12746	-
E22				D	F		0	0	-	1333	13000	-
E23					F, D		0	0	-	1341	13042	-
E24					F	D	0	0	-	1433	13509	-
E25	Flat						330	4005	59358	42	0	0
E26		Flat					330	3987	58263	60	49	55
E27				Flat			0	0	-	6717	67133	-
E28					Flat		0	0	-	6977	69729	-

Fonte: Elaborado pelo autor (2021)

Inicialmente, observamos que cenários com tabela de fatos ou tabela flat em columnstore (i.e., E01-E12 e E25-E26) alocaram seus dados majoritariamente em disco, em detrimento da utilização de memória RAM. O inverso ocorreu em cenários com tabela de fatos ou tabela flat em rowstore (i.e., E13-E24 e E27-E28), com uso significativo da memória RAM, em detrimento da utilização de disco no armazenamento de dados. A Figura 17 exibe os volumes de dados dos cenários, relacionando o armazenamento em disco e em RAM, nos SFs 1, 10 e 100. Nesta figura podemos ver que cenários com tabela de fatos ou flat em columnstore armazenaram seus dados em disco (i.e., grupos A e C da Figura 17) e cenários de grandes tabelas em rowstore armazenaram seus dados em RAM (i.e., grupos B e D da Figura 17). Note que, em SF=100, os cenários com maior volume em memória RAM foram exatamente aqueles com uso de tabelas de dimensões em rowstore, o que confirma a proposta de MemSQL de armazenar columnstore em disco e de rowstore em RAM (c.f., Seção 2.1.1).

De acordo com o Quadro 5, cenários em esquema estrela ocuparam menos espaço em disco que cenários de tabela flat, com diferença crescente de 59%, 63% e 74%, em SF=1, SF=10 e SF=100, respectivamente. Em relação ao armazenamento em memória RAM, cenários em

Figura 17 – Gráfico relacionando volume de dados armazenado em disco e memória RAM - SF 1, 10 e 100



Fonte: Elaborado pelo autor (2021)

esquema estrela também ocuparam menos espaço que cenários em tabelas flat, no entanto, a diferença fixa de 82% nos SFs, entre um esquema de dados e outro, é explicada pela ausência de compactação de dados em rowstore.

Quadro 5 – Diferença no volume de dados armazenado entre esquema estrela e tabela flat, considerando ocupação em disco e em RAM - SF 1, 10 e 100

			Média de armazenamento em disco (MB)			Diferença entre esquema estrela e tabela flat		
Armazenamento	Modelo de dados	Cenários	SF1	SF10	SF100	SF1	SF10	SF100
Columnstore	Esquema estrela	E01 ao E12	134	1.463	15.083	59%	63%	74%
	Tabela flat	E25 ao E26	330	3.996	58.811			

			Média de armazenamento em memória RAM (MB)			Diferença entre esquema estrela e tabela flat		
Armazenamento	Modelo de dados	Cenários	SF1	SF10	SF100	SF1	SF10	SF100
Rowstore	Esquema estrela	E13 ao E24	1.234	11.841	-	82%	82%	-
	Tabela flat	E27 ao E28	6.847	68.431	-			

Fonte: Elaborado pelo autor (2021)

De acordo com o Quadro 6, a compactação realizada no armazenamento columnstore reduziu, aproximadamente, 88% do volume de dados armazenado entre os cenários de esquema estrela e 94% do volume de dados armazenado entre os cenários de tabela flat, ambos em

SF=10 (i.e., o SF máximo comparável deste experimento).

Quadro 6 – Diferença no volume de dados armazenado entre columnstore e rowstore, de acordo a forma de armazenamento na tabela de fatos e na tabela flat - SF 1, 10 e 100

		Média do volume de dados (MB)			Diferença entre columnstore e rowstore		
Configuração	Cenários	SF1	SF10	SF100	SF1	SF10	SF100
Tabela de fatos em columnstore	E01 ao E12	134	1.463	15.083	89%	88%	-
Tabela de fatos em rowstore	E13 ao E24	1.234	11.841	-			
Tabela flat em columnstore	E25 ao E26	330	3.996	58.811	95%	94%	-
Tabela flat em rowstore	E27 ao E28	6.847	68.431	-			

Fonte: Elaborado pelo autor (2021)

Diante das estratégias de particionamento, em SF=100 percebemos que cenários com tabela de fatos em columnstore e particionamento com chave hash (i.e., E7 ao E12) apresentaram redução média de aproximadamente 2,7% no volume de dados armazenado quando comparados com cenários com tabela de fatos também em columnstore, mas com particionamento round-robin (i.e., E1 ao E06). O mesmo ocorreu em cenários de tabela flat em columnstore, com redução de 1,8% no volume de dados armazenado, quando particionado com chave hash (i.e., E26), ao invés de particionado com round-robin (i.e., E25). A redução no volume de dados armazenados em columnstore, em associação com o particionamento por chaves hash, ocorreu devido as tuplas com mesmo valor no campo chave estarem concentradas em nós específicos do cluster. Isso fez com que mais valores fossem eliminados por repetição, melhorando assim a compactação dos dados. Em relação às tabelas dimensionais, não identificamos diferenças significativas no volume de dados armazenado, em decorrência da estratégia de particionamento adotada. Em resumo, temos:

- O volume de dados armazenados em disco ou em memória RAM foi determinado pela forma de armazenamento (i.e., columnstore ou rowstore) da tabela de fatos e flat.
- Houve grande variação no volume de dados armazenado, em decorrência da quantidade de redundância introduzida e compactação de dados do columnstore.
- A compactação de dados presente no armazenamento columnstore reduziu em até 95% o volume de dados armazenado, quando comparado com o armazenamento rowstore.
- Com as estratégias de particionamento, o uso de chave hash proporcionou redução de até 2,7% no volume de dados armazenado em disco.

5.3 CONSULTAS - TEMPO MÉDIO

Os Quadros 7, 8 e 9 exibem a média das 13 consultas padrão do SSB, obtidas após 30 execuções individuais de cada consulta, em cada cenário, diante dos SFs 1, 10 e 100.

Quadro 7 – Tempo médio de execução das 13 consultas do SSB, em SF = 1

SF = 1														
Cenário	Q1.1	Q1.2	Q1.3	Q2.1	Q2.2	Q2.3	Q3.1	Q3.2	Q3.3	Q3.4	Q4.1	Q4.2	Q4.3	Total
E01	0,046	0,029	0,033	0,075	0,082	0,073	0,087	0,076	0,069	0,071	0,152	0,124	0,097	1,014
E02	0,045	0,029	0,033	0,091	0,102	0,069	0,080	0,072	0,067	0,065	0,148	0,124	0,091	1,016
E03	0,038	0,023	0,027	0,077	0,088	0,064	0,051	0,043	0,043	0,045	0,178	0,439	0,066	1,182
E04	0,044	0,028	0,032	0,070	0,083	0,075	0,070	0,058	0,057	0,059	0,140	0,122	0,088	0,926
E05	0,044	0,028	0,032	0,075	0,080	0,078	0,070	0,061	0,056	0,059	0,142	0,114	0,083	0,922
E06	0,038	0,023	0,027	0,252	0,239	0,228	0,089	0,039	0,039	0,037	0,128	0,094	0,049	1,282
E07	0,062	0,049	0,053	0,106	0,103	0,071	0,097	0,072	0,089	0,076	0,164	0,115	0,098	1,155
E08	0,061	0,048	0,052	0,101	0,101	0,075	0,097	0,076	0,083	0,068	0,164	0,130	0,091	1,147
E09	0,055	0,042	0,045	0,096	0,130	0,122	0,166	0,039	0,046	0,048	0,187	0,183	0,064	1,223
E10	0,060	0,048	0,050	0,083	0,075	0,072	0,088	0,059	0,075	0,058	0,154	0,117	0,084	1,023
E11	0,061	0,047	0,051	0,087	0,072	0,077	0,089	0,059	0,077	0,060	0,161	0,127	0,086	1,054
E12	0,055	0,041	0,044	0,245	0,248	0,237	0,059	0,040	0,047	0,049	0,162	0,082	0,049	1,358
E13	0,089	0,080	0,079	0,111	0,109	0,091	0,117	0,099	0,096	0,099	0,144	0,138	0,106	1,358
E14	0,140	0,129	0,129	0,108	0,109	0,092	0,118	0,095	0,091	0,091	0,141	0,128	0,111	1,482
E15	0,123	0,116	0,118	0,118	0,093	0,099	0,086	0,066	0,064	0,067	0,094	0,092	0,081	1,217
E16	0,139	0,130	0,131	0,120	0,102	0,091	0,105	0,084	0,084	0,084	0,149	0,131	0,108	1,458
E17	0,139	0,128	0,133	0,103	0,108	0,090	0,105	0,087	0,080	0,085	0,152	0,136	0,103	1,449
E18	0,122	0,117	0,117	0,264	0,246	0,247	0,168	0,058	0,058	0,062	0,164	0,117	0,082	1,822
E19	0,156	0,089	0,090	0,102	0,104	0,103	0,126	0,108	0,169	0,065	0,154	0,139	0,116	1,521
E20	0,154	0,086	0,091	0,115	0,104	0,102	0,141	0,115	0,174	0,062	0,148	0,144	0,102	1,538
E21	0,139	0,135	0,138	0,127	0,140	0,082	0,092	0,069	0,069	0,071	0,121	0,127	0,092	1,402
E22	0,156	0,086	0,091	0,111	0,105	0,114	0,119	0,102	0,157	0,053	0,159	0,133	0,106	1,492
E23	0,153	0,082	0,089	0,107	0,107	0,116	0,121	0,098	0,163	0,053	0,151	0,138	0,103	1,481
E24	0,141	0,138	0,139	0,186	0,263	0,252	0,089	0,076	0,073	0,075	0,176	0,127	0,058	1,793
E25	0,024	0,017	0,017	0,064	0,045	0,040	0,052	0,066	0,047	0,045	0,050	0,059	0,064	0,590
E26	0,025	0,018	0,019	0,072	0,054	0,037	0,050	0,068	0,045	0,042	0,047	0,054	0,066	0,597
E27	0,115	0,096	0,113	0,243	0,224	0,228	0,258	0,250	0,261	0,245	0,226	0,250	0,253	2,762
E28	0,121	0,093	0,116	0,250	0,230	0,241	0,269	0,254	0,273	0,256	0,234	0,264	0,267	2,868

Fonte: Elaborado pelo autor (2021)

Inicialmente, verificamos que as consultas Q1.2, Q1.3, Q3.2 e Q4.3, nos cenários E01-E12 e E25-E26 (i.e., cenários com tabela de fatos em columnstore e cenários com tabela flat em columnstore, respectivamente), não apresentaram aumento de seus tempos médios de consulta, mesmo com aumento nos fatores de escala. Apesar de termos executado a limpeza de dados no MemSQL, como indica sua documentação, os tempos sem aumento podem apontar para utilização de memória cache. Esse mesmo comportamento não foi observado nos cenários com tabela de fatos em rowstore (i.e., E13-E24 e E27-E28). Assim, para evitar inconsistência nas análises desta seção, optamos por utilizar os resultados de Q1.1, Q2.1, Q3.1 e Q4.1, precisamente onde não observamos tal problema. O Quadro 10 exibe os resultados dessas 4

Quadro 8 – Tempo médio de execução das 13 consultas do SSB, em SF = 10

SF = 10														
Cenário	Q1.1	Q1.2	Q1.3	Q2.1	Q2.2	Q2.3	Q3.1	Q3.2	Q3.3	Q3.4	Q4.1	Q4.2	Q4.3	Total
E01	0,157	0,027	0,032	0,162	0,144	0,178	0,191	0,070	0,147	0,075	0,281	0,257	0,097	1,818
E02	0,171	0,047	0,055	0,162	0,139	0,177	0,186	0,067	0,149	0,067	0,281	0,252	0,094	1,847
E03	0,142	0,029	0,033	0,210	0,280	0,146	0,141	0,040	0,081	0,057	0,367	0,311	0,064	1,901
E04	0,170	0,046	0,050	0,172	0,153	0,199	0,195	0,076	0,143	0,074	0,286	0,261	0,106	1,931
E05	0,173	0,041	0,050	0,172	0,173	0,209	0,192	0,073	0,147	0,071	0,292	0,286	0,105	1,984
E06	0,143	0,030	0,035	0,274	0,312	0,233	0,121	0,044	0,097	0,066	0,705	0,511	0,140	2,711
E07	0,159	0,045	0,049	0,167	0,153	0,190	0,206	0,072	0,126	0,080	0,315	0,256	0,092	1,910
E08	0,157	0,046	0,049	0,165	0,153	0,184	0,208	0,068	0,115	0,074	0,312	0,260	0,090	1,881
E09	0,166	0,066	0,068	0,171	0,136	0,120	0,162	0,040	0,101	0,056	0,375	0,221	0,073	1,755
E10	0,181	0,070	0,075	0,177	0,159	0,195	0,211	0,071	0,132	0,083	0,318	0,271	0,107	2,050
E11	0,154	0,042	0,031	0,176	0,181	0,213	0,198	0,071	0,127	0,080	0,326	0,282	0,107	1,988
E12	0,163	0,065	0,069	0,241	0,256	0,210	0,132	0,073	0,104	0,094	0,681	0,302	0,209	2,599
E13	0,564	0,549	0,541	0,317	0,307	0,307	0,407	0,321	0,303	0,307	0,428	0,396	0,340	5,087
E14	0,564	0,535	0,538	0,319	0,300	0,306	0,409	0,311	0,296	0,309	0,423	0,406	0,328	5,044
E15	0,635	0,534	0,531	0,319	0,404	0,296	0,385	0,303	0,286	0,296	0,474	0,426	0,296	5,185
E16	0,566	0,538	0,538	0,332	0,313	0,330	0,395	0,306	0,298	0,303	0,434	0,389	0,323	5,065
E17	0,565	0,542	0,541	0,345	0,327	0,320	0,392	0,308	0,305	0,315	0,439	0,407	0,342	5,148
E18	0,711	0,536	0,529	0,455	0,461	0,424	0,451	0,300	0,310	0,299	0,769	0,703	0,300	6,248
E19	0,459	0,377	0,385	0,384	0,361	0,358	0,472	0,370	0,677	0,091	0,565	0,363	0,311	5,173
E20	0,469	0,371	0,376	0,384	0,353	0,354	0,469	0,369	0,671	0,090	0,566	0,353	0,294	5,119
E21	0,448	0,367	0,371	0,394	0,348	0,347	0,415	0,347	0,345	0,348	0,515	0,436	0,368	5,049
E22	0,462	0,370	0,375	0,392	0,374	0,371	0,451	0,353	0,675	0,086	0,556	0,370	0,302	5,137
E23	0,452	0,371	0,377	0,387	0,380	0,368	0,463	0,355	0,674	0,087	0,569	0,384	0,309	5,176
E24	0,440	0,356	0,364	0,506	0,511	0,462	0,444	0,358	0,356	0,354	0,843	0,562	0,476	6,032
E25	0,095	0,026	0,019	0,088	0,079	0,064	0,074	0,098	0,061	0,050	0,092	0,068	0,108	0,922
E26	0,096	0,028	0,020	0,090	0,081	0,070	0,069	0,089	0,062	0,051	0,097	0,076	0,113	0,942
E27	0,557	0,539	0,562	1,265	1,331	1,254	1,307	1,177	1,340	1,297	1,245	1,234	1,213	14,321
E28	0,606	0,578	0,585	1,374	1,460	1,399	1,435	1,273	1,491	1,425	1,351	1,341	1,337	15,655

Fonte: Elaborado pelo autor (2021)

Quadro 9 – Tempo médio de execução das 13 consultas do SSB, em SF = 100

SF = 100														
Cenário	Q1.1	Q1.2	Q1.3	Q2.1	Q2.2	Q2.3	Q3.1	Q3.2	Q3.3	Q3.4	Q4.1	Q4.2	Q4.3	Total
E01	0,586	0,025	0,040	0,551	0,497	0,586	1,377	0,072	1,015	0,133	2,065	0,751	0,095	7,793
E02	0,590	0,025	0,041	0,544	0,484	0,579	1,362	0,069	1,002	0,127	2,057	0,749	0,099	7,728
E03	0,551	0,049	0,063	0,954	2,614	0,575	1,301	0,041	0,969	0,207	3,706	3,628	0,083	14,741
E04	0,571	0,085	0,100	0,543	0,489	0,589	1,374	0,151	1,130	0,210	2,099	0,770	0,156	8,267
E05	0,584	0,024	0,038	0,554	0,509	0,605	1,392	0,179	1,033	0,149	2,128	0,816	0,179	8,190
E06	0,553	0,049	0,065	1,148	2,762	0,762	3,723	0,413	1,202	0,214	4,043	4,563	0,147	19,644
E07	0,604	0,026	0,047	0,548	0,503	0,595	1,339	0,075	0,997	0,156	2,039	0,792	0,102	7,823
E08	0,602	0,025	0,046	0,535	0,484	0,590	1,338	0,071	0,969	0,148	2,033	0,789	0,101	7,731
E09	0,600	0,052	0,072	0,663	0,543	0,586	1,238	0,041	0,949	0,090	2,297	0,880	0,087	8,098
E10	0,604	0,024	0,044	0,568	0,495	0,608	1,392	0,131	1,085	0,223	2,076	0,821	0,156	8,227
E11	0,588	0,024	0,041	0,575	0,530	0,629	1,371	0,180	1,001	0,155	2,065	0,827	0,175	8,161
E12	0,596	0,052	0,070	0,851	0,793	0,755	1,397	0,231	1,161	0,254	2,741	1,332	0,154	10,387
E13 ao E24	-	-	-	-	-	-	-	-	-	-	-	-	-	-
E25	0,409	0,120	0,027	0,279	0,260	0,224	0,314	0,249	0,282	0,181	0,276	0,258	0,283	3,162
E26	0,424	0,121	0,034	0,284	0,274	0,229	0,320	0,258	0,291	0,179	0,285	0,271	0,290	3,260
E27 ao E28	-	-	-	-	-	-	-	-	-	-	-	-	-	-

Fonte: Elaborado pelo autor (2021)

consultas, em cada cenário, acompanhados dos tempos totais por SF. Já o Quadro 11 exibe esses mesmos resultados de forma agrupada.

Quadro 10 – Tempo médio das consultas Q1.1, Q2.1, Q3.1 e Q4.1, em SF 1, 10 e 100

Cenário	SF = 1					SF = 10					SF = 100				
	Q1.1	Q2.1	Q3.1	Q4.1	Total	Q1.1	Q2.1	Q3.1	Q4.1	Total	Q1.1	Q2.1	Q3.1	Q4.1	Total
E01	0,046	0,075	0,087	0,152	0,360	0,157	0,162	0,191	0,281	0,791	0,586	0,551	1,377	2,065	4,579
E02	0,045	0,091	0,080	0,148	0,364	0,171	0,162	0,186	0,281	0,800	0,590	0,544	1,362	2,057	4,553
E03	0,038	0,077	0,051	0,178	0,344	0,142	0,210	0,141	0,367	0,860	0,551	0,954	1,301	3,706	6,512
E04	0,044	0,070	0,070	0,140	0,324	0,170	0,172	0,195	0,286	0,823	0,571	0,543	1,374	2,099	4,587
E05	0,044	0,075	0,070	0,142	0,331	0,173	0,172	0,192	0,292	0,829	0,584	0,554	1,392	2,128	4,658
E06	0,038	0,252	0,089	0,128	0,507	0,143	0,274	0,121	0,705	1,243	0,553	1,148	3,723	4,043	9,467
E07	0,062	0,106	0,097	0,164	0,429	0,159	0,167	0,206	0,315	0,847	0,604	0,548	1,339	2,039	4,530
E08	0,061	0,101	0,097	0,164	0,423	0,157	0,165	0,208	0,312	0,842	0,602	0,535	1,338	2,033	4,508
E09	0,055	0,096	0,166	0,187	0,504	0,166	0,171	0,162	0,375	0,874	0,600	0,663	1,238	2,297	4,798
E10	0,060	0,083	0,088	0,154	0,385	0,181	0,177	0,211	0,318	0,887	0,604	0,568	1,392	2,076	4,640
E11	0,061	0,087	0,089	0,161	0,398	0,154	0,176	0,198	0,326	0,854	0,588	0,575	1,371	2,065	4,599
E12	0,055	0,245	0,059	0,162	0,521	0,163	0,241	0,132	0,681	1,217	0,596	0,851	1,397	2,741	5,585
E13	0,089	0,111	0,117	0,144	0,461	0,564	0,317	0,407	0,428	1,716	-	-	-	-	-
E14	0,140	0,108	0,118	0,141	0,507	0,564	0,319	0,409	0,423	1,715	-	-	-	-	-
E15	0,123	0,118	0,086	0,094	0,421	0,635	0,319	0,385	0,474	1,813	-	-	-	-	-
E16	0,139	0,120	0,105	0,149	0,513	0,566	0,332	0,395	0,434	1,727	-	-	-	-	-
E17	0,139	0,103	0,105	0,152	0,499	0,565	0,345	0,392	0,439	1,741	-	-	-	-	-
E18	0,122	0,264	0,168	0,164	0,718	0,711	0,455	0,451	0,769	2,386	-	-	-	-	-
E19	0,156	0,102	0,126	0,154	0,538	0,459	0,384	0,472	0,565	1,880	-	-	-	-	-
E20	0,154	0,115	0,141	0,148	0,558	0,469	0,384	0,469	0,566	1,888	-	-	-	-	-
E21	0,139	0,127	0,092	0,121	0,479	0,448	0,394	0,415	0,515	1,772	-	-	-	-	-
E22	0,156	0,111	0,119	0,159	0,545	0,462	0,392	0,451	0,556	1,861	-	-	-	-	-
E23	0,153	0,107	0,121	0,151	0,532	0,452	0,387	0,463	0,569	1,871	-	-	-	-	-
E24	0,141	0,186	0,089	0,176	0,592	0,440	0,506	0,444	0,843	2,233	-	-	-	-	-
E25	0,024	0,064	0,052	0,050	0,190	0,095	0,088	0,074	0,092	0,349	0,409	0,279	0,314	0,276	1,278
E26	0,025	0,072	0,050	0,047	0,194	0,096	0,090	0,069	0,097	0,352	0,424	0,284	0,320	0,285	1,313
E27	0,115	0,243	0,258	0,226	0,842	0,557	1,265	1,307	1,245	4,374	-	-	-	-	-
E28	0,121	0,250	0,269	0,234	0,874	0,606	1,374	1,435	1,351	4,766	-	-	-	-	-

Fonte: Elaborado pelo autor (2021)

De acordo com os Quadros 10 e 11, a execução de consultas em cenários com tabela de fatos ou flat em columnstore (i.e., E01 a E12 e E25 a E26, respectivamente) apresentou melhor desempenho que a execução de consultas de cenários equivalentes em rowstore (i.e., E13 a E24 e E27 a E28), com diferença de 52% em esquema estrela e de 92% em tabela flat, ambos em SF=10. No entanto, para tabelas de dimensões, verificamos que a utilização de armazenamento columnstore (i.e., E01, E02, E07 e E08) ou rowstore (i.e., E01, E02, E07 e E08) não apresentou diferença clara no desempenho de consultas, com variação de apenas 1,7% entre as formas de armazenamento. Assim, consideramos que o armazenamento columnstore, quando aplicado à tabela de fatos ou à tabela flat, melhorou o desempenho na execução de consultas. Podemos concluir também que, na realização de consultas, o armazenamento columnstore em disco

apresentou melhor desempenho que o armazenamento rowstore em memória RAM.

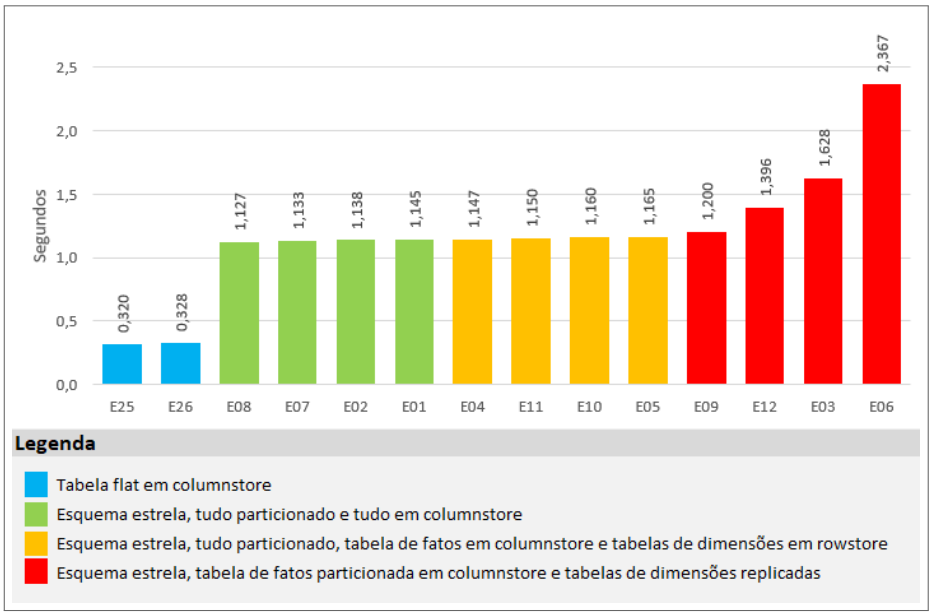
Quadro 11 – Média das consultas Q1.1, Q2.1, Q3.1 e Q4.1, de acordo com o tipo de tabela e a forma de armazenamento, em SF1, 10 e 100

Tipo de tabela	Forma de armazenamento	Cenários	Tempo médio (em segundos)			Redução de columnstore		
			SF1	SF10	SF100	SF1	SF10	SF100
Tabela de fatos	Columnstore	E01 ao E12	0,102	0,226	1,313	23%	52%	-
	Rowstore	E13 ao E24	0,133	0,471	-			
Tabela flat	Columnstore	E25 ao E26	0,048	0,088	0,324	78%	92%	-
	Rowstore	E27 ao E28	0,215	1,143	-			

Fonte: Elaborado pelo autor (2021)

A Figura 18 exibe os tempos médios de execução de consultas de forma ordenada, em SF=100, onde podemos verificar que os cenários com tabela flat tiveram ampla vantagem sob os cenários em esquema estrela, ambos em columnstore. Ao compararmos os cenários em tabela flat (i.e., E25 e E26) com os cenários em esquema estrela (sem uso de tabelas de dimensões replicadas), identificamos redução de, aproximadamente, 72% no tempo de execução de consultas.

Figura 18 – Ranking de médias nas consultas Q1.1, Q2.1, Q3.1 e Q4.1, em SF=100



Fonte: Elaborado pelo autor (2021)

A vantagem do uso de tabelas flat foi ainda maior quando as consultas envolviam maior número de dimensões (c.f., Quadro 12). Por exemplo, a vantagem do uso de tabelas flat em relação ao uso de esquema estrela, na execução da consulta Q1.1 (i.e., envolvendo apenas 1 tabela de dimensão), foi de 30% e na execução da consulta Q4.1 (i.e., envolvendo as 4 tabelas

de dimensões), essa vantagem foi de 86%. Assim, para a obtenção de alto desempenho na execução de consultas, o uso de tabela flat deve ser indicado.

Quadro 12 – Desempenho médio na execução de consultas envolvendo 1 dimensão (Q1.1) e envolvendo 4 dimensões (Q4.1), em SF=100

Cenário	SF = 100	
	Q1.1	Q4.1
E01	0,586	2,065
E02	0,590	2,057
E04	0,571	2,099
E05	0,584	2,128
E07	0,604	2,039
E08	0,602	2,033
E10	0,604	2,076
E11	0,588	2,065
Média	0,591	2,070
E25	0,409	0,276
E26	0,424	0,285
Média	0,417	0,281
Diferença	30%	86%

Fonte: Elaborado pelo autor (2021)

Com relação ao impacto que os métodos de distribuição exercem na execução de consultas, notamos que os cenários com replicação de tabelas de dimensões (i.e., E03, E06, E09 e E12) apresentaram perda de desempenho no tempo de execução de consultas, quando comparados aos cenários com distribuição por particionamento (c.f., Figura 18). Dessa forma, podemos concluir que o uso de tabelas replicadas, em DWs com MemSQL, deve ser evitado. Por fim, não identificamos diferenças de desempenho na execução de consultas, considerando as estratégias de particionamento round-robin e hash. Em resumo, para o tempo médio de execução de consultas, temos:

- Em DWs com MemSQL, o armazenamento columnstore (em disco) proporcionou melhor desempenho no tempo médio de execução de consultas, quando utilizado em grandes tabelas, apresentando redução de 52% em esquema estrela e de 92% em tabela flat. A utilização de columnstore em tabelas de dimensões também apresentou benefícios para o tempo de execução de consultas, apesar da diferença observada ser quase desprezível.
- Identificamos que cenários com tabela flat apresentaram redução de 72% no tempo médio de execução de consultas, em relação a cenários com esquema estrela, em MemSQL.

- Observamos melhorias de desempenho em cenários com tabelas de dimensões particionadas, em comparação aos cenários com tabelas de dimensões replicadas, mas não percebemos diferenças de desempenho em consultas entre as estratégias de particionamento round-robin e hash.

5.4 CONSULTAS - CONSUMO DE RAM

Analisar o consumo de memória RAM, durante a execução de consultas, atendeu dois objetivos: 1) saber se os tempos médios de consulta coletados foram afetados pelo espaço insuficiente na memória RAM e 2) propriamente investigar as diferenças deste consumo diante dos cenários experimentados. O Quadro 13, que exibe o somatório geral dos picos no consumo de RAM, obtidas nas 3 máquinas do cluster, durante a execução das consultas.

Quadro 13 – Consumo de RAM durante a execução das consultas, nos SFs 1, 10 e 100

Cenário							Memória RAM (MB)		
							SF1	SF10	SF100
E01	F, D						783	1203	8264
E02	F	D					332	1214	8349
E03	F		D				781	1007	8449
E04	F			D			319	1207	8388
E05	F				D		628	1194	8170
E06	F					D	581	1241	8359
E07	D	F					903	1621	8253
E08		F, D					389	1249	8037
E09		F	D				629	1447	8114
E10		F		D			475	1468	8002
E11		F			D		537	1083	7952
E12		F				D	429	1475	8108
E13	D			F			384	889	-
E14		D		F			351	993	-
E15			D	F			181	215	-
E16				F, D			428	1159	-
E17				F	D		307	412	-
E18				F		D	397	660	-
E19	D				F		410	1286	-
E20		D			F		401	855	-
E21			D		F		331	666	-
E22				D	F		673	814	-
E23					F, D		609	714	-
E24					F	D	290	697	-
E25	Flat						757	1163	8554
E26		Flat					526	937	8423
E27				Flat			150	245	-
E28					Flat		165	403	-

Fonte: Elaborado pelo autor (2021)

Ao considerarmos o Quadro 13 em conjunto com o Quadro 4 (i.e., volume de dados armazenado em disco e RAM), temos o total de memória RAM utilizada para as atividades de armazenamento e consultas de cada cenário. Assim, observamos que, entre todos os cenários avaliados, E27 e E28, em SF=10, apresentaram maior proximidade de atingir o limite de memória RAM disponível nos nós folha deste experimento (i.e., 83GB). Estes cenários utilizaram, aproximadamente, 67GB em memória RAM entre armazenamento e consultas de dados, representando ainda uma distância segura em relação ao limite disponível. Dessa forma, podemos afirmar que nenhum dos resultados de tempo médio de consultas foi afetado por limitação de memória RAM no cluster.

Podemos visualizar os resultados do Quadro 13 em agrupamentos, de acordo com a forma de armazenamento dada à tabela de fatos ou à tabela flat, conforme Quadro 14. De acordo com este quadro, cenários com grandes tabelas em rowstore apresentaram os menores consumos de memória RAM, durante a execução de consultas. Em SF=10, por exemplo, cenários com tabela de fatos em rowstore (i.e., E13 a E24) reduziram, em média, 39% do consumo de memória RAM durante a execução de consultas, quando comparados a cenários com tabela de fatos em columnstore (i.e., E01 a E12). Para cenários com tabela flat em rowstore (i.e., E27 e E28), a redução média foi de 69% em relação aos cenários com tabela flat em columnstore (i.e., E25 a E26). Apesar dessa constatação (de menor consumo de RAM durante a execução de consultas com cenários armazenados com rowstore), não podemos afirmar se isso possui relação com a forma armazenamento rowstore ou com a mídia de armazenamento utilizada (memória RAM), já que MemSQL não permite dissociar essas funcionalidades.

Quadro 14 – Consumo de RAM durante consultas (sumarizado) e diferença no consumo de RAM durante a execução de consultas

Tipo de tabela	Forma de armazenamento	Cenários	Memória RAM (MB)			Diferença entre columnstore e rowstore		
			SF1	SF10	SF100	SF1	SF10	SF100
Tabela de fatos	Columnstore	E01 ao E12	566	1284	8204			
	Rowstore	E13 ao E24	397	780	-	30%	39%	-
Tabela flat	Columnstore	E25 ao E26	641	1050	8488			
	Rowstore	E27 ao E28	158	324	-	75%	69%	-

Fonte: Elaborado pelo autor (2021)

Diante dos esquemas de dados (i.e., esquema estrela e tabela flat), dos métodos de distribuição (i.e., particionamento e replicação) e das estratégias de particionamento (i.e., round-robin e hash) empregadas em tabelas de fatos ou tabelas flat, não percebemos diferenças significativas, não permitindo maiores considerações. Sendo assim, em resumo temos:

- Nenhum dos cenários avaliados foi afetado por insuficiência de memória RAM no cluster, diante das atividades de armazenamento e consulta de dados. Essa observação reforça a validade dos resultados obtidos para o tempo médio de consultas.

- Utilizar bases em rowstore proporcionou redução no consumo de RAM, durante a execução de consultas. Em esquema estrela, houve redução de 39% e em tabela flat, a redução foi de 69%. Contudo, não podemos afirmar se isso possui relação com a forma armazenamento rowstore ou com a mídia de armazenamento utilizada (memória RAM).

- Diante dos modelos de dados, dos métodos de distribuição e das estratégias de particionamento, não foram percebidas diferenças significativas entre as opções testadas.

5.5 ANÁLISE GERAL

Analisando o conjunto das métricas apresentadas, podemos verificar que, o uso do armazenamento columnstore em DWs com MemSQL trouxe maiores benefícios em comparação ao uso do armazenamento rowstore. Apesar de ser 5,8% mais lento na carga dos dados em esquema estrela (em média), columnstore ofereceu melhor desempenho no tempo médio de consultas, com redução de aproximadamente 52% em esquema estrela e de 92% em tabelas flat. O uso de columnstore, em tabelas dimensionais, proporcionou economia de 1,7% no tempo médio de consultas, quando comparado com cenários de tabelas dimensionais em rowstore, no SF=100. Além disso, a compactação de dados, presente em columnstore, possibilitou redução de até 88% do volume de dados armazenados em esquema estrela e até 95% em cenários de tabela flat. Por fim, o uso de columnstore não provocou insuficiência de memória RAM no cluster, tal como ocorrera em cenários com tabela de fatos ou flat em rowstore (em RAM), em se tratando de maiores volumes de dados. Dessa forma, podemos concluir que, a opção pelo armazenamento columnstore deva ser adotada para todas as tabelas de um DW distribuído com MemSQL, mesmo realizado em disco.

Analisando os métodos de distribuição, constatamos que cenários com tabelas particionadas tiveram clara vantagem no tempo de carga, no volume de dados armazenado e no tempo médio de consultas, quando comparados aos cenários com replicação de tabelas. Considerando tais desvantagens, concluímos que o uso de replicação de tabelas, em DWs implementados com MemSQL, deva ser evitado. Dentre as estratégias de particionamento testadas, verificamos redução no volume de dados armazenados em disco de 2,8%, quando utilizamos particiona-

mento por chaves hash em conjunto com o armazenamento columnstore. No entanto, não percebemos qualquer vantagem entre a utilização dessas estratégias de particionamento, em relação ao tempo de carga e ao tempo médio de consultas.

Por fim, no que diz respeito aos esquemas de dados, percebemos que a utilização de tabelas flat proporcionou clara redução nos tempos médios de consultas (até 72%), quando comparados a cenários em esquema estrela. Apesar dessa vantagem significativa no tempo médio de consultas, devemos considerar os grandes aumentos no tempo de carga de dados e no volume de dados armazenados, em decorrência direta da eliminação de junções entre as tabelas do esquema estrela. Consideramos ainda que o desempenho em consultas nesses esquemas pode ser otimizado com a simples adição de máquinas ao cluster, o que aumenta a capacidade de processamento paralelo. Cabe analisar em cada situação, se é necessário ter o menor tempo de consultas possível em um DW implementado com NewSQL, utilizando tabelas flat. Em resumo, para a obtenção dos melhores desempenhos em DWs NewSQL, especificamente com MemSQL, indicamos:

- **Tempo de carga:** uso de esquema estrela, armazenamento rowstore e qualquer estratégia de particionamento (com a observação à garantia de memória RAM)
- **Armazenamento em disco:** uso de esquema estrela e particionamento por hash.
- **Armazenamento em RAM:** uso de esquema estrela.
- **Tempo de consultas:** uso de tabela flat, com armazenamento columnstore e qualquer estratégia de particionamento.
- **Consumo de RAM durante consultas:** uso do armazenamento rowstore na tabela de fatos ou tabela flat.

No contexto de DW implementado em MemSQL, identificamos como melhores cenários, levando em consideração todas as métricas avaliadas, o cenário E08 (i.e., todas as tabelas em columnstore e particionadas por hash) em esquema estrela e o cenário E25 (i.e., armazenado em columnstore e particionamento com round-robin) em tabela flat.

5.6 CONSIDERAÇÕES

Neste capítulo, apresentamos os resultados do experimento e realizamos análises para cada métrica avaliada. Isso permitiu a visualização de padrões no desempenho dos cenários, ao qual podemos gerar melhores orientações para a construção de DWs em NewSQL, mais precisamente com MemSQL. No próximo capítulo, apresentamos as considerações finais deste trabalho.

6 CONSIDERAÇÕES FINAIS

Neste trabalho, realizamos um estudo comparativo experimental sobre o desempenho DWs NewSQL, explorando diferentes métodos de distribuição, formas de armazenamento e esquema de dados, ao qual foram implementadas em uma das mais conhecidas distribuições NewSQL, o MemSQL.

No experimento, geramos cenários que combinavam: o uso de esquema estrela ou de tabela flat; o uso das formas de armazenamento rowstore e columnstore; o particionamento de dados usando round-robin e hash e também replicação de tabelas. Com o benchmark SSB, avaliamos o desempenho de cada um dos cenários propostos, diante das métricas de: tempo de carga de dados, volume de dados armazenados, tempo médio das consultas e consumo de memória durante as consultas. A experimentação ocorreu com 3 diferentes fatores de escala (1, 10 e 100), em ambiente de cluster computacional composto por 3 máquinas.

Por principais resultados desta pesquisa, verificamos que o uso do armazenamento columnstore trouxe maiores benefícios aos DWs, quando empregado em todas suas tabelas, com clara vantagem sob o armazenamento rowstore, diante do tempo médio de consultas e do volume de dados armazenado. A diferença no tempo médio de consultas foi de 52% em cenários de esquema estrela e de 92% em cenários de tabela flat. Verificamos também que a utilização de tabela flat (única) para DWs com MemSQL promoveu clara melhoria no tempo médio de consultas, apresentando economia de até 72% quando comparado com cenários em esquema estrela. Embora tenha promovido ganho expressivo no tempo de consultas, o uso de tabela flat apresentou considerável aumento no tempo de carga e no volume de dados de suas bases. Outra constatação foi que, tabelas dimensionais particionadas apresentaram melhor desempenho no tempo de carga de dados, no volume de dados armazenado e no tempo médio de consultas, quando comparadas com tabelas dimensionais replicadas. Por fim, observamos economia de 2,8% no volume de dados armazenado, em DWs distribuídos com chave hash em comparação a DWs distribuídos com round-robin. Não observamos relações entre essas estratégias de particionamento e os tempos de carga e de consulta de dados. De acordo com tais observações, pontuamos como bons caminhos na implementação de DWs em MemSQL, a utilização do armazenamento columnstore, da distribuição por particionamento com chave hash e esquema estrela ou tabela flat. Tabela flat, para os casos que buscam o menor tempo possível de consultas (com prejuízo no tempo de carga e no volume de dados gerado em disco)

e esquema estrela, para se obter bons tempos de carga e baixo volume de dados, apresentando desempenho ainda aceitável na execução de consultas.

Retomando o quadro de trabalhos relacionados da Seção 3.3, que exibe os recursos experimentados por cada autor, incluímos os dados deste trabalho para compor a Tabela 6.

Tabela 6 – Comparação do presente trabalho com os trabalhos relacionados

Critério	Oliveira e Bernardino (2017)	Schreiner et al. (2019)	Kaur e Sachdeva (2017)	Astrova et al. (2021)	Cá (2018)	Azevedo (2020)
Realizou testes em cluster?	Não	Sim	Não	Não	Não	Sim
Utilizou um DW?	Sim	Não	Não	Não	Não	Sim
Experimentou diferentes esquemas de dados?	Não	Não	Não	Não	Não	Sim
Testou diferentes métodos de distribuição	Não	Não	Não	Não	Não	Sim
Avaliou várias formas de armazenamento?	Não	Não	Não	Não	Não	Sim
Testou diferentes volumes de dados?	Não	Não	Não	Não	Não	Sim

Fonte: Elaborado pelo autor (2021)

Dentre os principais diferenciais deste trabalho, destacamos:

- A identificação de diferentes cenários para testes, combinando recursos presentes em bancos de dados NewSQL e diferentes modelos de dados para DW.
- A experimentação em cluster computacional, com diferentes volumes de dados. Isso nos permitiu um melhor entendimento sobre o desempenho dos recursos avaliados, considerando o aumento na quantidade de dados.
- Identificação de diferenças de desempenho entre as formas de armazenamento rowstore e columnstore, no contexto de DWs distribuídos em NewSQL.
- Identificação de diferenças de desempenho entre esquema estrela e tabela flat, no contexto de DWs distribuídos em NewSQL.

6.1 LIMITAÇÕES

A análise de desempenho neste trabalho possui algumas limitações:

- Não ter utilizado estatística inferencial que poderiam ter melhor referenciado as análises.
- Não ter explorado outras combinações de cenários para DWs NewSQL, como por exemplo a utilização pontual de tabelas replicadas (e.g., na tabela *date*) e de outros modelos de dados (e.g., esquema floco de neve ou tabela de fatos unificada com dimensão tempo).
- A quantidade de máquinas utilizadas, limitada pela versão de testes do MemSQL, não possibilitou maiores estudos sobre o desenho da arquitetura do cluster.

6.2 TRABALHOS FUTUROS

- Analisar o impacto de diferentes quantidades de máquinas no cluster e em diferentes papéis: realizar mudanças na quantidade de máquinas e em seus papéis permitirá entender o comportamento do desempenho a medida que os recursos ficam escaços ou abundantes. Isso é útil tanto para observar o comportamento da escalabilidade horizontal, quanto para ajustar o investimento em hardware a depender da quantidade de dados da base.
- Estudar mudanças nas chaves de particionamento e de armazenamento columnstore: entender quais fatores na escolha das chaves de particionamento e armazenamento podem melhorar o desempenho de consultas, bem como na quantidade de chaves a utilizar.
- Analisar o desempenho de DWs em diferentes bancos de dados NewSQL, explorando suas possibilidades tecnológicas.
- Analisar o desempenho de DWs NewSQL com ênfase em dados espaciais.
- Analisar o desempenho de DWs entre bancos de dados NoSQL e NewSQL.
- Estudar o desempenho de diferentes modelagens de bancos que atendam ao mesmo tempo necessidades transacionais e analíticas (*Hybrid transactional/analytical processing* (HTAP)).

REFERÊNCIAS

ABADI, D.; MADDEN, S.; FERREIRA, M. Integrating compression and execution in column-oriented database systems. In: *Proceedings of the 2006 ACM SIGMOD International Conference on Management of Data*. New York, NY, USA: Association for Computing Machinery, 2006. (SIGMOD '06), p. 671–682. ISBN 1595934340. Disponível em: <https://doi.org/10.1145/1142473.1142548>.

ADAMSON, C. *Mastering data warehouse aggregates : solutions for star schema performance*. Indianapolis, IN: Wiley Pub, 2006. ISBN 978-0-471-77709-0.

ASTROVA, I.; KOSCHEL, A.; WELLERMANN, N.; KLOSTERMEYER, P. Performance benchmarking of newsql databases with yahoo cloud serving benchmark. In: ARAI, K.; KAPOOR, S.; BHATIA, R. (Ed.). *Proceedings of the Future Technologies Conference (FTC) 2020, Volume 2*. Cham: Springer International Publishing, 2021. p. 271–281. ISBN 978-3-030-63089-8.

BOUSSAHOUA, M.; BOUSSAID, O.; BENTAYEB, F. Logical schema for data warehouse on column-oriented nosql databases. In: BENSLIMANE, D.; DAMIANI, E.; GROSKY, W. I.; HAMEURLAIN, A.; SHETH, A.; WAGNER, R. R. (Ed.). *Database and Expert Systems Applications*. Cham: Springer International Publishing, 2017. p. 247–256. ISBN 978-3-319-64471-4.

CHAUDHURI, S.; DAYAL, U. An overview of data warehousing and olap technology. *SIGMOD Rec.*, Association for Computing Machinery, New York, NY, USA, v. 26, n. 1, p. 65–74, mar. 1997. ISSN 0163-5808. Disponível em: <https://doi.org/10.1145/248603.248616>.

COSTA, C.; SANTOS, M. Y. Evaluating several design patterns and trends in big data warehousing systems. In: *Advanced Information Systems Engineering*. Springer International Publishing, 2018. p. 459–473. Disponível em: <https://doi.org/10.1007/978-3-319-91563-0_28>.

COSTA, C. H.; FILHO, J. V. B. M.; MAIA, P. H. M.; OLIVEIRA, F. C. M. B. Sharding by hash partitioning - a database scalability pattern to achieve evenly sharded database clusters. In: *Proceedings of the 17th International Conference on Enterprise Information Systems*. [S.l.]: SCITEPRESS - Science and and Technology Publications, 2015.

Cá, J. L. *Uma abordagem para modelagem de desempenho para sistemas de banco de dados NewSQL e comparação de modelos gerados com diferentes técnicas de aprendizado de máquina*. 59 p. Dissertação (Pós-Graduação em Ciência da Computação do Centro de Ciências e Tecnologia) — Universidade Estadual do Ceará, 2018. Disponível em: <http://siduece.uece.br/siduece/trabalhoAcademicoPublico.jsf?id=83564>.

DB-Engines. *DB-Engines Ranking*. 2020. <https://db-engines.com/en/ranking>. Accessed: 2020-06-06.

DEWITT, D.; GRAY, J. Parallel database systems: The future of high performance database systems. *Commun. ACM*, Association for Computing Machinery, New York, NY, USA, v. 35, n. 6, p. 85–98, jun. 1992. ISSN 0001-0782. Disponível em: <https://doi.org/10.1145/129888.129894>.

DUGGIRALA, S. *NewSQL Databases and Scalable In-Memory Analytics*. [S.l.: s.n.], 2018.

FERRO, M.; FRAGOSO, R.; FIDALGO, R. Document-oriented geospatial data warehouse: An experimental evaluation of SOLAP queries. In: *2019 IEEE 21st Conference on Business Informatics (CBI)*. IEEE, 2019. Disponível em: <https://doi.org/10.1109/cbi.2019.00013>.

GARANI, G.; HELMER, S. Integrating star and snowflake schemas in data warehouses. *International Journal of Data Warehousing and Mining*, v. 8, p. 22–40, 10 2012.

GOMES, F. P. *Curso de Estatística Experimental*. [S.l.]: FEALQ, 2000. 477 p. ISBN 9788571330559.

HOGG, R. *Probability and statistical inference*. Boston: Pearson, 2013. 202 p. ISBN 0-321-92327-8.

INMON, W. *Como construir o data warehouse*. Rio de Janeiro: Campus, 1997. ISBN 9788535201413. Disponível em: <https://books.google.com.br/books?id=pgVbywAACAAJ>.

KAUR, K.; SACHDEVA, M. Performance evaluation of newsql databases. In: . [S.l.: s.n.], 2017. p. 1–5.

KIMBALL, R.; ROSS, M. *The data warehouse toolkit : the complete guide to dimensional modeling*. New York: Wiley, 2002. ISBN 978-0471200246.

MemSQL. *Choosing a Shard Key*. 2020. <https://docs.singlestore.com/v7.3/guides/use-memsql/physical-schema-design/optimizing-table-data-structures/optimizing-table-data-structures/#choosing-a-shard-key>. Accessed: 2020-11-27.

MEMSQL. Distributed architecture. 2020. Disponível em: <https://docs.singlestore.com/v7.3/key-concepts-and-features/distributed-architecture/cluster-components/>.

MEMSQL. *How the Columnstore Works*. 2020. Disponível em: <https://docs.singlestore.com/v7.3/key-concepts-and-features/physical-schema-design/columnstore/how-the-columnstore-works/>.

MURAZZA, M. R.; NURWIDYANTORO, A. Cassandra and sql database comparison for near real-time twitter data warehouse. In: *2016 International Seminar on Intelligent Technology and Its Applications (ISITIA)*. [S.l.: s.n.], 2016. p. 195–200.

OLIVEIRA, J.; BERNARDINO, J. Newsql databases - memsql and voltdb experimental evaluation. In: *KEOD*. [S.l.: s.n.], 2017. p. 276–281.

O'NEIL, P.; O'NEIL, E.; CHEN, X.; REVILAK, S. The star schema benchmark and augmented fact table indexing. In: NAMBIAR, R.; POESS, M. (Ed.). *Performance Evaluation and Benchmarking*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009. p. 237–252. ISBN 978-3-642-10424-4.

PAVLO, A.; ASLETT, M. What's really new with newsql? *SIGMOD Rec.*, Association for Computing Machinery, New York, NY, USA, v. 45, n. 2, p. 45–55, set. 2016. ISSN 0163-5808. Disponível em: <https://doi.org/10.1145/3003665.3003674>.

PONNIAH, P. Data warehousing: a comprehensive guide for it professional. In: *2nd ed., New York: The McGraw-Hill Companies*. [S.l.: s.n.], 2010.

SCHREINER, G. A.; KNOB, R.; DUARTE, D.; VILAIN, P.; MELLO, R. d. S. Newsql through the looking glass. In: *Proceedings of the 21st International Conference on Information Integration and Web-Based Applications e Services*. New York, NY, USA: Association for Computing Machinery, 2019. (iiWAS2019), p. 361–369. ISBN 9781450371797. Disponível em: <https://doi.org/10.1145/3366030.3366080>.

APÊNDICE A – SCRIPTS EXECUTADOS NO EXPERIMENTO

```
#########################################################
# Script principal do experimento
# Autor: Alesanco Andrade Azevedo
# Arquivo: main.py
#########################################################

from sqlalchemy import Column, VARCHAR
from sqlalchemy.ext.declarative import declarative_base
from sqlalchemy import create_engine
from openpyxl import load_workbook
from datetime import datetime
import os, sys, time
import pandas as pd
import numpy as np

engine  = create_engine()
tables  = ['part','customer','supplier','dim_date','lineorder']
sf      = "sf1"
repeats = 30
cluster = " - 2 folhas - "
dir_experiments = "./experiments/"
hora = str(datetime.now())
stat1 = hora[0:10]+' - '+sf.upper()+cluster+'principal.csv'
stat2 = hora[0:10]+' - '+sf.upper()+cluster+'secundario.csv'
stat3 = hora[0:10]+' - '+sf.upper()+cluster+'zlixo.csv'

statistics1 = open(stat1,'a')
statistics2 = open(stat2,'a')
trash       = open(stat3,'a')
statistics1.write('date_time'+";"+'sf'+";"+'db'+";"+'part'+";"+'customer'+";"+'supplier'+";
"+'dim_date'+";"+'lineorder'+";"+'total_time'+";"+'ram_mb'+";"+'disk_mb'+";"+'Q1.1'+";"+'Q1.2'+
"+'Q1.3'+";"+'Q2.1'+";"+'Q2.2'+";"+'Q2.3'+";"+'Q3.1'+";"+'Q3.2'+";"+'Q3.3'+";"+'Q3.4'+";"+'Q4.1'+
"+'Q4.2'+";"+'Q4.3'+";"+"\n")
statistics2.write('date_time'+";"+'sf'+";"+'db'+";"+'query_id'+";")
trash.write('date_time'+";"+'sf'+";"+'db'+";"+'query_id'+";")
for idx, a in enumerate(range(repeats)):
    statistics2.write('T'+str(idx+1)+";")
    trash.write('T'+str(idx+1)+";")
statistics2.write("mean;std;coef_v;\n")
trash.write("mean;std;coef_v;\n")
statistics1.close()
statistics2.close()
trash.close()

def collect_resources(db):
    query = """ SELECT rowsegs.database_name, Memory_in_mb, Disk_in_mb FROM  (SELECT database_name,
 truncate(Round(Sum(memory_use / 1024 / 1024 ), 3),0) AS Memory_in_mb FROM
information_schema.table_statistics GROUP  BY database_name) rowsegs LEFT JOIN (SELECT
database_name, truncate(Round(Sum(compressed_size) / 1024 / 1024 , 3),0) AS Disk_in_mb FROM
information_schema.columnar_segments GROUP  BY database_name) colsegs ON rowsegs.database_name =
colsegs.database_name WHERE rowsegs.database_name = '%s';"""%(db)
    result = engine.execute(query)
    for res in result:
        try:
            return str(res[1]), str(res[2])
        except Exception:
            return 0,0

def create_db(experiment):
    try:
        sql = "%s%s"%(dir_experiments,experiment)
        file1 = open(sql,"r")
        query = ""
        for line in file1:
            query += line
        engine.execute(query)
        print "\nDatabase %s created."%(experiment[-6:-4])
        return 1
    except Exception:
```

```
        print "\nError in file %s."%(experiment)
        return 0

def load_db(db):
    hora = str(datetime.now())
    hora = hora[0:19]
    statistics1 = open(stat1,'a')
    statistics1.write(hora+";"+str(sf)+";"+str(db)+";")
    try:
        total_time = 0.0
        for table in tables:
            print "Insert data in table: %s" %(table),
            time1 = time.time()
            query = "LOAD DATA LOCAL INFILE './data/%s/%s.tbl' INTO TABLE %s.%s FIELDS TERMINATED
BY '|' LINES TERMINATED BY '|\r\n'" %(sf, table, db, table)
            engine.execute(query)
            time2 = time.time()
            parcial_time = time2 - time1
            total_time = total_time + float(parcial_time)
            parcial_time = "%.2f" %(parcial_time)
            parcial_time = parcial_time.replace(".",",")
            print parcial_time
            statistics1.write(parcial_time+";")
        print "######"
        total_time = "%.2f" %(total_time)
        total_time = str(total_time)
        total_time = total_time.replace(".",",")
        print "Total time:" , total_time
        print "######"
        statistics1.write(total_time+";")
        lengths = collect_resources(db)
        statistics1.write(str(lengths[0])+";"+str(lengths[1])+";")
        statistics1.close()
        rebalance = "REBALANCE PARTITIONS ON %s;" %(db)
        engine.execute(rebalance)
        os.system("sudo sysctl -w vm.drop_caches=3")
        return 1
    except Exception:
        statistics1.write("error;")
        print "\nError in load of %s."%(str(db))
        statistics1.close()
        os.system("sudo sysctl -w vm.drop_caches=3")
        return 0

def load_db_unique(db):
    hora = str(datetime.now())
    hora = hora[0:19]
    statistics1 = open(stat1,'a')
    statistics1.write(hora+";"+str(sf)+";"+str(db)+";-;-;-;-;")
    print "Insert data in table: lineorder",
    try:
        time1 = time.time()
        query = "LOAD DATA LOCAL INFILE './data/%s/unique.csv' INTO TABLE %s.lineorder FIELDS
TERMINATED BY '|' LINES TERMINATED BY '\n'" %(sf, db)
        engine.execute(query)
        time2 = time.time()
        parcial_time = time2 - time1
        parcial_time = "%.2f" %(parcial_time)
        parcial_time = parcial_time.replace(".",",")
        print parcial_time
        statistics1.write(parcial_time+";"+parcial_time+";")
        lengths = collect_resources(db)
        statistics1.write(str(lengths[0])+";"+str(lengths[1])+";")
        statistics1.close()
        os.system("sudo sysctl -w vm.drop_caches=3")
        return 1
    except Exception:
        statistics1.write("error;")
        print "\nError in load of %s."%(str(db))
```

```
        statistics1.close()
        os.system("sudo sysctl -w vm.drop_caches=3")
        return 0

def queries(db):
    query   = "USE %s"%(db)
    engine.execute(query)
    sheets  = []
    wb = ""
    if int(db[3:])<25:
        wb = load_workbook("queries.xlsx")
    else:
        wb = load_workbook("queries_unique.xlsx")
    names   = wb.sheetnames
    for name in names:
        sheets.append(name)
    #clear_caches(db)
    for sheet in sheets:
        ws = wb[sheet]
        for idx, line in enumerate(ws):
            if idx>0:
                query_id        = line[0].value
                query_content   = line[1].value
                statistics1     = open(stat1,'a')
                statistics2     = open(stat2,'a')
                try:
                    for k in range(10):
                        times = []
                        print query_id,"\t",
                        for a in range(repeats):
                            engine.execute("drop all from plancache;")
                            time1 = time.time()
                            engine.execute(query_content)
                            time2 = time.time()
                            time_query  = time2 - time1
                            time_query = "%.3f" %(time_query)
                            times.append(float(time_query))
                            if float(time_query)>100: #para economizar tempo de processamento com
modelos inuteis.
                                print "BREAK\t", time_query
                                break;
                        serie    =  pd.Series(times)
                        dp       = "%.3f" % (serie.std())
                        media    = "%.3f" % (serie.mean())
                        coef_var = "%.3f" % (serie.std() / serie.mean())
                        print "coef:", coef_var, "media:", media, "dp:", dp,
                        limiar = 0.10
                        if float(coef_var)<=limiar:
                            dp       = str(dp).replace(".",",")
                            media    = str(media).replace(".",",")
                            coef_var = str(coef_var).replace(".",",")

statistics2.write(str(datetime.now())+";"+str(sf)+";"+str(db)+";"+str(query_id)+";")
                            if len(times)>2: #para evitar preencher o arquivo statistics2 em casos
de problema no tempo (desnecessarios armazenar e bom para confirmar o problema).
                                for t in times:
                                    t = str(t).replace(".",",")
                                    statistics2.write(str(t)+";")
                                if len(times)<repeats:
                                    ajust = repeats - len(times)
                                    for i in range(ajust):
                                        statistics2.write(";")
                                statistics2.write(media+";"+dp+";"+coef_var+";\n")
                            else:
                                for t in times:
                                    t = str(t).replace(".",",")
                                    statistics2.write(str(t)+";")
                                statistics2.write(";\n")
                            time_mean  = float("%.3f"%(sum(times)/len(times)))
```

```
                                time_mean = str(time_mean)
                                time_mean = time_mean.replace(".",",")
                                statistics1.write(time_mean+";")
                                print "OK"
                                break;
                            else:
                                print "nao"
                                dp       = str(dp).replace(".",",")
                                media    = str(media).replace(".",",")
                                coef_var = str(coef_var).replace(".",",")
                                trash    = open(stat3,'a')

trash.write(str(datetime.now())+";"+str(sf)+";"+str(db)+";"+str(query_id)+";")
                                if len(times)>2:
                                    for t in times:
                                        t = str(t).replace(".",",")
                                        trash.write(str(t)+";")
                                    if len(times)<repeats:
                                        ajust = repeats - len(times)
                                        for i in range(ajust):
                                            trash.write(";")
                                    trash.write(media+";"+dp+";"+coef_var+";\n")
                                else:
                                    for t in times:
                                        t = str(t).replace(".",",")
                                        trash.write(str(t)+";")
                                    trash.write(";\n")
                                if k==9:
                                    statistics1.write("-;")
                                trash.close()
                        except Exception:
                            pass
                            statistics1.write("Error;")

statistics2.write(str(datetime.now())+";"+str(sf)+";"+str(db)+";"+str(query_id)+";")
                            for a in range(repeats+2):
                                statistics2.write("error;")
                            statistics2.write("\n")
                            print "Error in %s by query %s"%(str(db),str(query_id))
                        statistics1.close()
                        statistics2.close()
                        cleaner = os.system("sudo sysctl -w vm.drop_caches=3")
            clear_caches(db)
            statistics1 = open(stat1,'a')
            statistics1.write("\n")
            statistics1.close()


def remove(db):
    sql = "DROP DATABASE IF EXISTS %s"%(db)
    engine.execute(sql)
    print "Database %s removed."%(db)
    os.system("sudo sysctl -w vm.drop_caches=3")
    print "################################################"

def main():
    for i in range(1):
        for experiment in sorted(os.listdir(dir_experiments)):
            if "db" in experiment:
                db = "ssb%s"%(experiment[-6:-4])
                engine.execute("drop all from plancache;")
                sucess = create_db(experiment)
                if sucess:
                    if int(experiment[-6:-4])<25:
                        sucess = load_db(db)
                    else:
                        sucess = load_db_unique(db)
                    if sucess:
                        queries(db)
```

```
                remove(db)
                statistics1 = open(stat1,'a')
                statistics1.write("\n")
                statistics1.close()
                cleaner = os.system("sudo sysctl -w vm.drop_caches=3")
if __name__ == "__main__":
    main()


############################################################
# Script para leitura do consumo de memÃ³ria RAM
# Autor: Alesanco Andrade Azevedo
# Arquivo: memory.py
############################################################

import os, psutil, time
from datetime import datetime
from openpyxl import load_workbook

os.system("sudo sysctl -w vm.drop_caches=3")
memory = 'memory.csv'
file_memory = open(memory,'a')
file_memory.write('day;hour;ram_total_mb;ram_used_mb;perc;\n')
file_memory.close()

def calcMem():
    mem = psutil.virtual_memory()
    data = datetime.now()
    hora = str(data.hour)+":"+str(data.minute)+":"+str(data.second)
    dia  = str(data.day)+"-"+str(data.month)
    file_memory = open(memory,'a')
    percent = (str(mem.percent)).replace(".",",")
    total = ("%.2f" %(float(mem.total)/ (1024.0 ** 2))).replace(".",",")
    used  = ("%.2f" %(float(mem.total - mem.free) / (1024.0 ** 2))).replace(".",",")
    file_memory.write(dia+";"+hora+";"+total+";"+used+";"+percent+"%\n")
    file_memory.close()
    del dia, hora, total, used, percent, mem

while True:
    time.sleep(1)
    calcMem()
```

APÊNDICE B – DADOS DO TEMPO MÉDIO DE CONSULTAS

sf	cenario	consulta	desvio	coef_v	conf. 99% (n=30)	média
sf1	E01	Q1.1	0,003	0,067	0,044 ≤ μ ≤ 0,047	0,046
sf1	E01	Q1.2	0,001	0,030	0,028 ≤ μ ≤ 0,029	0,029
sf1	E01	Q1.3	0,001	0,040	0,032 ≤ μ ≤ 0,033	0,033
sf1	E01	Q2.1	0,003	0,046	0,073 ≤ μ ≤ 0,076	0,075
sf1	E01	Q2.2	0,010	0,120	0,076 ≤ μ ≤ 0,087	0,082
sf1	E01	Q2.3	0,005	0,073	0,070 ≤ μ ≤ 0,075	0,073
sf1	E01	Q3.1	0,012	0,139	0,080 ≤ μ ≤ 0,093	0,087
sf1	E01	Q3.2	0,012	0,156	0,069 ≤ μ ≤ 0,082	0,076
sf1	E01	Q3.3	0,005	0,072	0,066 ≤ μ ≤ 0,071	0,069
sf1	E01	Q3.4	0,007	0,094	0,067 ≤ μ ≤ 0,074	0,071
sf1	E01	Q4.1	0,012	0,078	0,145 ≤ μ ≤ 0,158	0,152
sf1	E01	Q4.2	0,014	0,113	0,116 ≤ μ ≤ 0,131	0,124
sf1	E01	Q4.3	0,013	0,131	0,090 ≤ μ ≤ 0,103	0,097
					Total	1,014
sf1	E02	Q1.1	0,002	0,038	0,044 ≤ μ ≤ 0,046	0,045
sf1	E02	Q1.2	0,001	0,034	0,028 ≤ μ ≤ 0,029	0,029
sf1	E02	Q1.3	0,001	0,029	0,032 ≤ μ ≤ 0,033	0,033
sf1	E02	Q2.1	0,009	0,098	0,086 ≤ μ ≤ 0,095	0,091
sf1	E02	Q2.2	0,009	0,090	0,097 ≤ μ ≤ 0,106	0,102
sf1	E02	Q2.3	0,004	0,060	0,067 ≤ μ ≤ 0,071	0,069
sf1	E02	Q3.1	0,004	0,056	0,077 ≤ μ ≤ 0,081	0,080
sf1	E02	Q3.2	0,011	0,155	0,066 ≤ μ ≤ 0,077	0,072
sf1	E02	Q3.3	0,006	0,087	0,063 ≤ μ ≤ 0,069	0,067
sf1	E02	Q3.4	0,004	0,065	0,062 ≤ μ ≤ 0,067	0,065
sf1	E02	Q4.1	0,014	0,096	0,140 ≤ μ ≤ 0,154	0,148
sf1	E02	Q4.2	0,014	0,111	0,116 ≤ μ ≤ 0,130	0,124
sf1	E02	Q4.3	0,012	0,131	0,084 ≤ μ ≤ 0,096	0,091
					Total	1,016
sf1	E03	Q1.1	0,002	0,045	0,036 ≤ μ ≤ 0,038	0,038
sf1	E03	Q1.2	0,001	0,042	0,022 ≤ μ ≤ 0,023	0,023
sf1	E03	Q1.3	0,001	0,038	0,026 ≤ μ ≤ 0,027	0,027
sf1	E03	Q2.1	0,003	0,035	0,075 ≤ μ ≤ 0,077	0,077
sf1	E03	Q2.2	0,003	0,030	0,086 ≤ μ ≤ 0,089	0,088
sf1	E03	Q2.3	0,002	0,035	0,063 ≤ μ ≤ 0,065	0,064
sf1	E03	Q3.1	0,001	0,017	0,050 ≤ μ ≤ 0,051	0,051
sf1	E03	Q3.2	0,001	0,022	0,042 ≤ μ ≤ 0,043	0,043
sf1	E03	Q3.3	0,003	0,059	0,041 ≤ μ ≤ 0,044	0,043
sf1	E03	Q3.4	0,002	0,042	0,044 ≤ μ ≤ 0,046	0,045
sf1	E03	Q4.1	0,009	0,049	0,173 ≤ μ ≤ 0,182	0,178
sf1	E03	Q4.2	0,006	0,014	0,435 ≤ μ ≤ 0,441	0,439
sf1	E03	Q4.3	0,003	0,040	0,064 ≤ μ ≤ 0,067	0,066
					Total	1,182
sf1	E04	Q1.1	0,002	0,034	0,043 ≤ μ ≤ 0,044	0,044
sf1	E04	Q1.2	0,001	0,025	0,028 ≤ μ ≤ 0,028	0,028
sf1	E04	Q1.3	0,001	0,035	0,031 ≤ μ ≤ 0,032	0,032
sf1	E04	Q2.1	0,004	0,060	0,066 ≤ μ ≤ 0,070	0,069
sf1	E04	Q2.2	0,009	0,113	0,077 ≤ μ ≤ 0,087	0,083
sf1	E04	Q2.3	0,007	0,092	0,071 ≤ μ ≤ 0,078	0,075
sf1	E04	Q3.1	0,005	0,071	0,067 ≤ μ ≤ 0,072	0,070
sf1	E04	Q3.2	0,003	0,045	0,056 ≤ μ ≤ 0,059	0,058
sf1	E04	Q3.3	0,006	0,097	0,054 ≤ μ ≤ 0,060	0,057
sf1	E04	Q3.4	0,003	0,050	0,057 ≤ μ ≤ 0,060	0,059
sf1	E04	Q4.1	0,005	0,037	0,137 ≤ μ ≤ 0,142	0,140
sf1	E04	Q4.2	0,011	0,092	0,116 ≤ μ ≤ 0,127	0,122
sf1	E04	Q4.3	0,010	0,116	0,082 ≤ μ ≤ 0,093	0,088
					Total	0,925
sf1	E05	Q1.1	0,002	0,042	0,043 ≤ μ ≤ 0,045	0,044
sf1	E05	Q1.2	0,001	0,034	0,027 ≤ μ ≤ 0,028	0,028
sf1	E05	Q1.3	0,001	0,037	0,030 ≤ μ ≤ 0,032	0,032
sf1	E05	Q2.1	0,004	0,053	0,072 ≤ μ ≤ 0,076	0,075
sf1	E05	Q2.2	0,004	0,046	0,077 ≤ μ ≤ 0,081	0,080
sf1	E05	Q2.3	0,003	0,045	0,075 ≤ μ ≤ 0,079	0,078
sf1	E05	Q3.1	0,005	0,074	0,067 ≤ μ ≤ 0,072	0,070
sf1	E05	Q3.2	0,004	0,065	0,058 ≤ μ ≤ 0,063	0,061
sf1	E05	Q3.3	0,003	0,049	0,054 ≤ μ ≤ 0,057	0,056
sf1	E05	Q3.4	0,003	0,057	0,057 ≤ μ ≤ 0,060	0,059
sf1	E05	Q4.1	0,006	0,042	0,139 ≤ μ ≤ 0,145	0,142
sf1	E05	Q4.2	0,005	0,042	0,111 ≤ μ ≤ 0,116	0,114
sf1	E05	Q4.3	0,004	0,054	0,080 ≤ μ ≤ 0,085	0,083
					Total	0,922

sf	cenario	consulta	desvio	coef_v	conf. 99% (n=30)	média
sf1	E06	Q1.1	0,002	0,045	0,036 ≤ μ ≤ 0,038	0,038
sf1	E06	Q1.2	0,001	0,042	0,022 ≤ μ ≤ 0,023	0,023
sf1	E06	Q1.3	0,001	0,027	0,026 ≤ μ ≤ 0,026	0,027
sf1	E06	Q2.1	0,010	0,038	0,247 ≤ μ ≤ 0,257	0,252
sf1	E06	Q2.2	0,002	0,009	0,237 ≤ μ ≤ 0,240	0,239
sf1	E06	Q2.3	0,010	0,046	0,223 ≤ μ ≤ 0,233	0,229
sf1	E06	Q3.1	0,003	0,032	0,087 ≤ μ ≤ 0,089	0,089
sf1	E06	Q3.2	0,001	0,033	0,038 ≤ μ ≤ 0,039	0,039
sf1	E06	Q3.3	0,002	0,050	0,038 ≤ μ ≤ 0,040	0,039
sf1	E06	Q3.4	0,001	0,033	0,036 ≤ μ ≤ 0,038	0,037
sf1	E06	Q4.1	0,002	0,019	0,126 ≤ μ ≤ 0,129	0,128
sf1	E06	Q4.2	0,004	0,040	0,091 ≤ μ ≤ 0,095	0,094
sf1	E06	Q4.3	0,003	0,061	0,047 ≤ μ ≤ 0,050	0,049
					Total	1,282
sf1	E07	Q1.1	0,002	0,029	0,061 ≤ μ ≤ 0,063	0,062
sf1	E07	Q1.2	0,002	0,034	0,048 ≤ μ ≤ 0,050	0,049
sf1	E07	Q1.3	0,002	0,035	0,051 ≤ μ ≤ 0,053	0,053
sf1	E07	Q2.1	0,008	0,077	0,101 ≤ μ ≤ 0,110	0,106
sf1	E07	Q2.2	0,008	0,076	0,099 ≤ μ ≤ 0,107	0,103
sf1	E07	Q2.3	0,005	0,074	0,068 ≤ μ ≤ 0,073	0,071
sf1	E07	Q3.1	0,008	0,088	0,092 ≤ μ ≤ 0,101	0,097
sf1	E07	Q3.2	0,002	0,034	0,070 ≤ μ ≤ 0,073	0,072
sf1	E07	Q3.3	0,008	0,094	0,084 ≤ μ ≤ 0,092	0,089
sf1	E07	Q3.4	0,011	0,148	0,070 ≤ μ ≤ 0,081	0,076
sf1	E07	Q4.1	0,011	0,069	0,158 ≤ μ ≤ 0,170	0,164
sf1	E07	Q4.2	0,005	0,048	0,112 ≤ μ ≤ 0,117	0,115
sf1	E07	Q4.3	0,012	0,125	0,092 ≤ μ ≤ 0,104	0,098
					Total	1,155
sf1	E08	Q1.1	0,002	0,032	0,059 ≤ μ ≤ 0,061	0,061
sf1	E08	Q1.2	0,002	0,043	0,046 ≤ μ ≤ 0,049	0,048
sf1	E08	Q1.3	0,002	0,042	0,051 ≤ μ ≤ 0,053	0,052
sf1	E08	Q2.1	0,009	0,093	0,095 ≤ μ ≤ 0,105	0,101
sf1	E08	Q2.2	0,009	0,085	0,096 ≤ μ ≤ 0,105	0,101
sf1	E08	Q2.3	0,009	0,115	0,070 ≤ μ ≤ 0,079	0,075
sf1	E08	Q3.1	0,006	0,066	0,093 ≤ μ ≤ 0,100	0,097
sf1	E08	Q3.2	0,010	0,130	0,070 ≤ μ ≤ 0,080	0,076
sf1	E08	Q3.3	0,006	0,069	0,080 ≤ μ ≤ 0,086	0,084
sf1	E08	Q3.4	0,004	0,064	0,066 ≤ μ ≤ 0,070	0,068
sf1	E08	Q4.1	0,013	0,076	0,157 ≤ μ ≤ 0,170	0,164
sf1	E08	Q4.2	0,012	0,092	0,123 ≤ μ ≤ 0,136	0,130
sf1	E08	Q4.3	0,011	0,125	0,085 ≤ μ ≤ 0,096	0,091
					Total	1,147
sf1	E09	Q1.1	0,002	0,036	0,054 ≤ μ ≤ 0,056	0,055
sf1	E09	Q1.2	0,002	0,041	0,041 ≤ μ ≤ 0,043	0,042
sf1	E09	Q1.3	0,002	0,045	0,043 ≤ μ ≤ 0,045	0,045
sf1	E09	Q2.1	0,008	0,082	0,091 ≤ μ ≤ 0,099	0,096
sf1	E09	Q2.2	0,010	0,079	0,124 ≤ μ ≤ 0,134	0,130
sf1	E09	Q2.3	0,009	0,075	0,117 ≤ μ ≤ 0,127	0,122
sf1	E09	Q3.1	0,002	0,011	0,164 ≤ μ ≤ 0,166	0,166
sf1	E09	Q3.2	0,002	0,043	0,038 ≤ μ ≤ 0,039	0,039
sf1	E09	Q3.3	0,003	0,063	0,044 ≤ μ ≤ 0,047	0,046
sf1	E09	Q3.4	0,001	0,023	0,047 ≤ μ ≤ 0,048	0,048
sf1	E09	Q4.1	0,012	0,062	0,181 ≤ μ ≤ 0,193	0,187
sf1	E09	Q4.2	0,011	0,062	0,177 ≤ μ ≤ 0,189	0,183
sf1	E09	Q4.3	0,010	0,159	0,058 ≤ μ ≤ 0,069	0,064
					Total	1,223
sf1	E10	Q1.1	0,002	0,035	0,059 ≤ μ ≤ 0,061	0,060
sf1	E10	Q1.2	0,002	0,048	0,046 ≤ μ ≤ 0,048	0,048
sf1	E10	Q1.3	0,002	0,044	0,048 ≤ μ ≤ 0,051	0,050
sf1	E10	Q2.1	0,002	0,027	0,081 ≤ μ ≤ 0,084	0,083
sf1	E10	Q2.2	0,007	0,089	0,071 ≤ μ ≤ 0,078	0,075
sf1	E10	Q2.3	0,008	0,106	0,067 ≤ μ ≤ 0,075	0,072
sf1	E10	Q3.1	0,004	0,046	0,086 ≤ μ ≤ 0,090	0,088
sf1	E10	Q3.2	0,003	0,052	0,057 ≤ μ ≤ 0,060	0,059
sf1	E10	Q3.3	0,003	0,043	0,073 ≤ μ ≤ 0,076	0,075
sf1	E10	Q3.4	0,003	0,051	0,056 ≤ μ ≤ 0,059	0,058
sf1	E10	Q4.1	0,005	0,031	0,151 ≤ μ ≤ 0,156	0,154
sf1	E10	Q4.2	0,006	0,055	0,113 ≤ μ ≤ 0,120	0,117
sf1	E10	Q4.3	0,010	0,121	0,078 ≤ μ ≤ 0,089	0,084
					Total	1,023

sf	cenario	consulta	desvio	coef_v	conf. 99% (n=30)	média
sf1	E11	Q1.1	0,001	0,020	0,060 ≤ μ ≤ 0,061	0,061
sf1	E11	Q1.2	0,002	0,037	0,045 ≤ μ ≤ 0,047	0,047
sf1	E11	Q1.3	0,002	0,044	0,049 ≤ μ ≤ 0,052	0,051
sf1	E11	Q2.1	0,004	0,047	0,084 ≤ μ ≤ 0,088	0,087
sf1	E11	Q2.2	0,006	0,089	0,068 ≤ μ ≤ 0,074	0,072
sf1	E11	Q2.3	0,006	0,081	0,073 ≤ μ ≤ 0,080	0,077
sf1	E11	Q3.1	0,004	0,049	0,086 ≤ μ ≤ 0,091	0,089
sf1	E11	Q3.2	0,003	0,055	0,057 ≤ μ ≤ 0,060	0,059
sf1	E11	Q3.3	0,005	0,068	0,074 ≤ μ ≤ 0,079	0,077
sf1	E11	Q3.4	0,003	0,049	0,058 ≤ μ ≤ 0,061	0,060
sf1	E11	Q4.1	0,005	0,034	0,158 ≤ μ ≤ 0,163	0,161
sf1	E11	Q4.2	0,009	0,074	0,121 ≤ μ ≤ 0,131	0,127
sf1	E11	Q4.3	0,010	0,117	0,081 ≤ μ ≤ 0,091	0,086
					Total	1,054
sf1	E12	Q1.1	0,002	0,043	0,054 ≤ μ ≤ 0,056	0,055
sf1	E12	Q1.2	0,002	0,037	0,040 ≤ μ ≤ 0,042	0,041
sf1	E12	Q1.3	0,002	0,036	0,043 ≤ μ ≤ 0,045	0,044
sf1	E12	Q2.1	0,010	0,043	0,239 ≤ μ ≤ 0,250	0,245
sf1	E12	Q2.2	0,012	0,049	0,241 ≤ μ ≤ 0,253	0,248
sf1	E12	Q2.3	0,016	0,069	0,229 ≤ μ ≤ 0,245	0,238
sf1	E12	Q3.1	0,004	0,067	0,057 ≤ μ ≤ 0,061	0,059
sf1	E12	Q3.2	0,002	0,038	0,039 ≤ μ ≤ 0,040	0,040
sf1	E12	Q3.3	0,003	0,067	0,045 ≤ μ ≤ 0,048	0,047
sf1	E12	Q3.4	0,002	0,040	0,048 ≤ μ ≤ 0,050	0,049
sf1	E12	Q4.1	0,010	0,063	0,157 ≤ μ ≤ 0,167	0,162
sf1	E12	Q4.2	0,009	0,114	0,077 ≤ μ ≤ 0,087	0,082
sf1	E12	Q4.3	0,003	0,059	0,047 ≤ μ ≤ 0,050	0,049
					Total	1,358
sf1	E13	Q1.1	0,002	0,021	0,088 ≤ μ ≤ 0,089	0,089
sf1	E13	Q1.2	0,003	0,034	0,078 ≤ μ ≤ 0,081	0,080
sf1	E13	Q1.3	0,002	0,029	0,077 ≤ μ ≤ 0,079	0,079
sf1	E13	Q2.1	0,009	0,078	0,106 ≤ μ ≤ 0,115	0,111
sf1	E13	Q2.2	0,009	0,087	0,103 ≤ μ ≤ 0,113	0,109
sf1	E13	Q2.3	0,005	0,051	0,088 ≤ μ ≤ 0,093	0,091
sf1	E13	Q3.1	0,008	0,067	0,112 ≤ μ ≤ 0,120	0,117
sf1	E13	Q3.2	0,007	0,072	0,095 ≤ μ ≤ 0,102	0,099
sf1	E13	Q3.3	0,009	0,098	0,090 ≤ μ ≤ 0,100	0,096
sf1	E13	Q3.4	0,003	0,035	0,097 ≤ μ ≤ 0,100	0,099
sf1	E13	Q4.1	0,008	0,055	0,139 ≤ μ ≤ 0,147	0,144
sf1	E13	Q4.2	0,012	0,090	0,131 ≤ μ ≤ 0,144	0,138
sf1	E13	Q4.3	0,007	0,067	0,102 ≤ μ ≤ 0,109	0,106
					Total	1,358
sf1	E14	Q1.1	0,006	0,042	0,136 ≤ μ ≤ 0,142	0,140
sf1	E14	Q1.2	0,006	0,044	0,126 ≤ μ ≤ 0,132	0,129
sf1	E14	Q1.3	0,005	0,041	0,126 ≤ μ ≤ 0,131	0,129
sf1	E14	Q2.1	0,008	0,076	0,103 ≤ μ ≤ 0,111	0,108
sf1	E14	Q2.2	0,009	0,083	0,104 ≤ μ ≤ 0,113	0,109
sf1	E14	Q2.3	0,007	0,076	0,088 ≤ μ ≤ 0,095	0,092
sf1	E14	Q3.1	0,011	0,094	0,112 ≤ μ ≤ 0,124	0,118
sf1	E14	Q3.2	0,007	0,072	0,091 ≤ μ ≤ 0,098	0,095
sf1	E14	Q3.3	0,011	0,117	0,085 ≤ μ ≤ 0,096	0,091
sf1	E14	Q3.4	0,004	0,039	0,088 ≤ μ ≤ 0,092	0,091
sf1	E14	Q4.1	0,008	0,056	0,136 ≤ μ ≤ 0,144	0,141
sf1	E14	Q4.2	0,010	0,081	0,122 ≤ μ ≤ 0,133	0,128
sf1	E14	Q4.3	0,010	0,088	0,106 ≤ μ ≤ 0,116	0,112
					Total	1,482
sf1	E15	Q1.1	0,004	0,032	0,121 ≤ μ ≤ 0,125	0,123
sf1	E15	Q1.2	0,004	0,038	0,113 ≤ μ ≤ 0,117	0,116
sf1	E15	Q1.3	0,005	0,041	0,115 ≤ μ ≤ 0,120	0,118
sf1	E15	Q2.1	0,005	0,041	0,115 ≤ μ ≤ 0,120	0,118
sf1	E15	Q2.2	0,003	0,031	0,091 ≤ μ ≤ 0,094	0,093
sf1	E15	Q2.3	0,004	0,042	0,096 ≤ μ ≤ 0,100	0,099
sf1	E15	Q3.1	0,003	0,031	0,084 ≤ μ ≤ 0,087	0,086
sf1	E15	Q3.2	0,003	0,039	0,064 ≤ μ ≤ 0,067	0,066
sf1	E15	Q3.3	0,003	0,048	0,062 ≤ μ ≤ 0,066	0,065
sf1	E15	Q3.4	0,003	0,038	0,065 ≤ μ ≤ 0,068	0,067
sf1	E15	Q4.1	0,002	0,023	0,092 ≤ μ ≤ 0,095	0,094
sf1	E15	Q4.2	0,003	0,030	0,090 ≤ μ ≤ 0,093	0,092
sf1	E15	Q4.3	0,001	0,015	0,080 ≤ μ ≤ 0,081	0,081
					Total	1,217

sf	cenario	consulta	desvio	coef_v	conf. 99% (n=30)	média
sf1	E16	Q1.1	0,005	0,036	0,136 ≤ μ ≤ 0,141	0,139
sf1	E16	Q1.2	0,004	0,033	0,128 ≤ μ ≤ 0,132	0,130
sf1	E16	Q1.3	0,005	0,034	0,128 ≤ μ ≤ 0,133	0,131
sf1	E16	Q2.1	0,009	0,072	0,115 ≤ μ ≤ 0,124	0,120
sf1	E16	Q2.2	0,010	0,102	0,096 ≤ μ ≤ 0,106	0,102
sf1	E16	Q2.3	0,002	0,021	0,089 ≤ μ ≤ 0,091	0,091
sf1	E16	Q3.1	0,003	0,026	0,103 ≤ μ ≤ 0,106	0,105
sf1	E16	Q3.2	0,003	0,035	0,082 ≤ μ ≤ 0,085	0,084
sf1	E16	Q3.3	0,010	0,118	0,079 ≤ μ ≤ 0,089	0,084
sf1	E16	Q3.4	0,003	0,040	0,082 ≤ μ ≤ 0,086	0,084
sf1	E16	Q4.1	0,011	0,071	0,143 ≤ μ ≤ 0,154	0,149
sf1	E16	Q4.2	0,010	0,077	0,125 ≤ μ ≤ 0,135	0,131
sf1	E16	Q4.3	0,010	0,094	0,102 ≤ μ ≤ 0,113	0,108
					Total	1,458
sf1	E17	Q1.1	0,005	0,034	0,136 ≤ μ ≤ 0,141	0,139
sf1	E17	Q1.2	0,005	0,037	0,125 ≤ μ ≤ 0,130	0,128
sf1	E17	Q1.3	0,004	0,033	0,130 ≤ μ ≤ 0,134	0,133
sf1	E17	Q2.1	0,009	0,086	0,098 ≤ μ ≤ 0,107	0,103
sf1	E17	Q2.2	0,009	0,080	0,103 ≤ μ ≤ 0,112	0,108
sf1	E17	Q2.3	0,003	0,032	0,088 ≤ μ ≤ 0,091	0,090
sf1	E17	Q3.1	0,004	0,037	0,102 ≤ μ ≤ 0,106	0,105
sf1	E17	Q3.2	0,003	0,036	0,085 ≤ μ ≤ 0,089	0,087
sf1	E17	Q3.3	0,006	0,078	0,076 ≤ μ ≤ 0,083	0,080
sf1	E17	Q3.4	0,005	0,061	0,082 ≤ μ ≤ 0,087	0,085
sf1	E17	Q4.1	0,011	0,070	0,146 ≤ μ ≤ 0,157	0,152
sf1	E17	Q4.2	0,010	0,076	0,130 ≤ μ ≤ 0,140	0,136
sf1	E17	Q4.3	0,005	0,044	0,100 ≤ μ ≤ 0,105	0,103
					Total	1,449
sf1	E18	Q1.1	0,004	0,033	0,120 ≤ μ ≤ 0,124	0,122
sf1	E18	Q1.2	0,004	0,031	0,115 ≤ μ ≤ 0,118	0,117
sf1	E18	Q1.3	0,005	0,041	0,114 ≤ μ ≤ 0,119	0,117
sf1	E18	Q2.1	0,010	0,038	0,259 ≤ μ ≤ 0,269	0,264
sf1	E18	Q2.2	0,011	0,044	0,240 ≤ μ ≤ 0,251	0,246
sf1	E18	Q2.3	0,011	0,044	0,241 ≤ μ ≤ 0,252	0,247
sf1	E18	Q3.1	0,001	0,007	0,167 ≤ μ ≤ 0,168	0,168
sf1	E18	Q3.2	0,001	0,024	0,057 ≤ μ ≤ 0,058	0,058
sf1	E18	Q3.3	0,003	0,047	0,056 ≤ μ ≤ 0,059	0,058
sf1	E18	Q3.4	0,002	0,035	0,061 ≤ μ ≤ 0,063	0,062
sf1	E18	Q4.1	0,010	0,060	0,159 ≤ μ ≤ 0,169	0,164
sf1	E18	Q4.2	0,010	0,088	0,111 ≤ μ ≤ 0,122	0,117
sf1	E18	Q4.3	0,004	0,054	0,079 ≤ μ ≤ 0,083	0,082
					Total	1,822
sf1	E19	Q1.1	0,007	0,045	0,152 ≤ μ ≤ 0,159	0,156
sf1	E19	Q1.2	0,005	0,055	0,086 ≤ μ ≤ 0,091	0,089
sf1	E19	Q1.3	0,004	0,049	0,088 ≤ μ ≤ 0,092	0,090
sf1	E19	Q2.1	0,003	0,031	0,100 ≤ μ ≤ 0,103	0,102
sf1	E19	Q2.2	0,010	0,095	0,098 ≤ μ ≤ 0,108	0,104
sf1	E19	Q2.3	0,005	0,045	0,100 ≤ μ ≤ 0,105	0,103
sf1	E19	Q3.1	0,005	0,038	0,123 ≤ μ ≤ 0,128	0,126
sf1	E19	Q3.2	0,007	0,063	0,104 ≤ μ ≤ 0,111	0,108
sf1	E19	Q3.3	0,006	0,038	0,165 ≤ μ ≤ 0,171	0,169
sf1	E19	Q3.4	0,003	0,043	0,063 ≤ μ ≤ 0,066	0,065
sf1	E19	Q4.1	0,013	0,082	0,147 ≤ μ ≤ 0,160	0,154
sf1	E19	Q4.2	0,012	0,084	0,133 ≤ μ ≤ 0,145	0,139
sf1	E19	Q4.3	0,011	0,097	0,109 ≤ μ ≤ 0,121	0,116
					Total	1,521
sf1	E20	Q1.1	0,012	0,079	0,147 ≤ μ ≤ 0,159	0,154
sf1	E20	Q1.2	0,004	0,048	0,083 ≤ μ ≤ 0,087	0,086
sf1	E20	Q1.3	0,004	0,044	0,089 ≤ μ ≤ 0,093	0,091
sf1	E20	Q2.1	0,005	0,044	0,112 ≤ μ ≤ 0,117	0,115
sf1	E20	Q2.2	0,010	0,094	0,099 ≤ μ ≤ 0,109	0,104
sf1	E20	Q2.3	0,003	0,033	0,099 ≤ μ ≤ 0,103	0,102
sf1	E20	Q3.1	0,013	0,093	0,134 ≤ μ ≤ 0,147	0,141
sf1	E20	Q3.2	0,011	0,096	0,108 ≤ μ ≤ 0,120	0,115
sf1	E20	Q3.3	0,012	0,067	0,167 ≤ μ ≤ 0,179	0,174
sf1	E20	Q3.4	0,003	0,052	0,060 ≤ μ ≤ 0,063	0,062
sf1	E20	Q4.1	0,005	0,034	0,145 ≤ μ ≤ 0,150	0,148
sf1	E20	Q4.2	0,013	0,092	0,137 ≤ μ ≤ 0,150	0,144
sf1	E20	Q4.3	0,003	0,031	0,100 ≤ μ ≤ 0,103	0,102
					Total	1,538

sf	cenario	consulta	desvio	coef_v	conf. 99% (n=30)	média
sf1	E21	Q1.1	0,007	0,047	0,135 ≤ μ ≤ 0,141	0,139
sf1	E21	Q1.2	0,006	0,042	0,132 ≤ μ ≤ 0,138	0,135
sf1	E21	Q1.3	0,005	0,035	0,135 ≤ μ ≤ 0,140	0,138
sf1	E21	Q2.1	0,005	0,039	0,124 ≤ μ ≤ 0,129	0,127
sf1	E21	Q2.2	0,011	0,077	0,134 ≤ μ ≤ 0,145	0,140
sf1	E21	Q2.3	0,008	0,099	0,077 ≤ μ ≤ 0,086	0,082
sf1	E21	Q3.1	0,004	0,038	0,089 ≤ μ ≤ 0,093	0,092
sf1	E21	Q3.2	0,005	0,073	0,066 ≤ μ ≤ 0,071	0,069
sf1	E21	Q3.3	0,002	0,031	0,068 ≤ μ ≤ 0,070	0,069
sf1	E21	Q3.4	0,002	0,028	0,069 ≤ μ ≤ 0,072	0,071
sf1	E21	Q4.1	0,010	0,087	0,115 ≤ μ ≤ 0,126	0,121
sf1	E21	Q4.2	0,002	0,018	0,126 ≤ μ ≤ 0,128	0,127
sf1	E21	Q4.3	0,010	0,109	0,087 ≤ μ ≤ 0,097	0,092
					Total	1,402
sf1	E22	Q1.1	0,006	0,040	0,152 ≤ μ ≤ 0,158	0,156
sf1	E22	Q1.2	0,004	0,049	0,083 ≤ μ ≤ 0,088	0,086
sf1	E22	Q1.3	0,004	0,042	0,088 ≤ μ ≤ 0,092	0,091
sf1	E22	Q2.1	0,010	0,090	0,105 ≤ μ ≤ 0,115	0,111
sf1	E22	Q2.2	0,009	0,087	0,100 ≤ μ ≤ 0,110	0,105
sf1	E22	Q2.3	0,011	0,098	0,108 ≤ μ ≤ 0,119	0,114
sf1	E22	Q3.1	0,004	0,033	0,117 ≤ μ ≤ 0,121	0,119
sf1	E22	Q3.2	0,012	0,114	0,096 ≤ μ ≤ 0,107	0,102
sf1	E22	Q3.3	0,004	0,028	0,154 ≤ μ ≤ 0,159	0,157
sf1	E22	Q3.4	0,003	0,048	0,051 ≤ μ ≤ 0,054	0,053
sf1	E22	Q4.1	0,011	0,067	0,153 ≤ μ ≤ 0,164	0,159
sf1	E22	Q4.2	0,011	0,079	0,128 ≤ μ ≤ 0,138	0,133
sf1	E22	Q4.3	0,010	0,096	0,101 ≤ μ ≤ 0,111	0,106
					Total	1,492
sf1	E23	Q1.1	0,006	0,042	0,149 ≤ μ ≤ 0,156	0,153
sf1	E23	Q1.2	0,004	0,051	0,079 ≤ μ ≤ 0,083	0,082
sf1	E23	Q1.3	0,004	0,043	0,086 ≤ μ ≤ 0,090	0,089
sf1	E23	Q2.1	0,005	0,047	0,104 ≤ μ ≤ 0,109	0,107
sf1	E23	Q2.2	0,010	0,091	0,102 ≤ μ ≤ 0,112	0,107
sf1	E23	Q2.3	0,011	0,099	0,110 ≤ μ ≤ 0,121	0,116
sf1	E23	Q3.1	0,005	0,037	0,118 ≤ μ ≤ 0,123	0,121
sf1	E23	Q3.2	0,004	0,044	0,096 ≤ μ ≤ 0,100	0,099
sf1	E23	Q3.3	0,007	0,041	0,159 ≤ μ ≤ 0,166	0,163
sf1	E23	Q3.4	0,002	0,040	0,051 ≤ μ ≤ 0,054	0,053
sf1	E23	Q4.1	0,004	0,029	0,148 ≤ μ ≤ 0,152	0,151
sf1	E23	Q4.2	0,010	0,073	0,133 ≤ μ ≤ 0,143	0,138
sf1	E23	Q4.3	0,003	0,030	0,101 ≤ μ ≤ 0,104	0,103
					Total	1,472
sf1	E24	Q1.1	0,006	0,044	0,137 ≤ μ ≤ 0,143	0,141
sf1	E24	Q1.2	0,005	0,034	0,135 ≤ μ ≤ 0,140	0,138
sf1	E24	Q1.3	0,004	0,030	0,136 ≤ μ ≤ 0,141	0,139
sf1	E24	Q2.1	0,005	0,026	0,183 ≤ μ ≤ 0,188	0,186
sf1	E24	Q2.2	0,009	0,036	0,258 ≤ μ ≤ 0,268	0,263
sf1	E24	Q2.3	0,010	0,039	0,246 ≤ μ ≤ 0,256	0,252
sf1	E24	Q3.1	0,002	0,019	0,088 ≤ μ ≤ 0,090	0,089
sf1	E24	Q3.2	0,003	0,036	0,074 ≤ μ ≤ 0,077	0,076
sf1	E24	Q3.3	0,002	0,031	0,071 ≤ μ ≤ 0,073	0,073
sf1	E24	Q3.4	0,002	0,027	0,074 ≤ μ ≤ 0,076	0,075
sf1	E24	Q4.1	0,010	0,059	0,170 ≤ μ ≤ 0,181	0,176
sf1	E24	Q4.2	0,010	0,076	0,122 ≤ μ ≤ 0,132	0,127
sf1	E24	Q4.3	0,001	0,019	0,057 ≤ μ ≤ 0,059	0,058
					Total	1,793
sf1	E25	Q1.1	0,002	0,070	0,022 ≤ μ ≤ 0,024	0,024
sf1	E25	Q1.2	0,001	0,036	0,016 ≤ μ ≤ 0,017	0,017
sf1	E25	Q1.3	0,001	0,053	0,016 ≤ μ ≤ 0,017	0,017
sf1	E25	Q2.1	0,009	0,148	0,059 ≤ μ ≤ 0,068	0,064
sf1	E25	Q2.2	0,005	0,121	0,042 ≤ μ ≤ 0,047	0,045
sf1	E25	Q2.3	0,004	0,087	0,038 ≤ μ ≤ 0,042	0,040
sf1	E25	Q3.1	0,006	0,114	0,048 ≤ μ ≤ 0,054	0,052
sf1	E25	Q3.2	0,006	0,098	0,062 ≤ μ ≤ 0,068	0,066
sf1	E25	Q3.3	0,002	0,040	0,046 ≤ μ ≤ 0,048	0,047
sf1	E25	Q3.4	0,002	0,038	0,044 ≤ μ ≤ 0,045	0,045
sf1	E25	Q4.1	0,004	0,078	0,048 ≤ μ ≤ 0,052	0,050
sf1	E25	Q4.2	0,008	0,128	0,055 ≤ μ ≤ 0,063	0,059
sf1	E25	Q4.3	0,006	0,097	0,060 ≤ μ ≤ 0,066	0,064
					Total	0,590

sf	cenario	consulta	desvio	coef_v	conf. 99% (n=30)	média
sf1	E26	Q1.1	0,001	0,043	0,024 ≤ μ ≤ 0,025	0,025
sf1	E26	Q1.2	0,001	0,044	0,017 ≤ μ ≤ 0,018	0,018
sf1	E26	Q1.3	0,001	0,041	0,018 ≤ μ ≤ 0,019	0,019
sf1	E26	Q2.1	0,010	0,135	0,066 ≤ μ ≤ 0,076	0,072
sf1	E26	Q2.2	0,005	0,090	0,051 ≤ μ ≤ 0,056	0,054
sf1	E26	Q2.3	0,003	0,088	0,035 ≤ μ ≤ 0,038	0,037
sf1	E26	Q3.1	0,007	0,134	0,046 ≤ μ ≤ 0,053	0,050
sf1	E26	Q3.2	0,008	0,120	0,063 ≤ μ ≤ 0,072	0,068
sf1	E26	Q3.3	0,002	0,046	0,044 ≤ μ ≤ 0,046	0,045
sf1	E26	Q3.4	0,002	0,045	0,040 ≤ μ ≤ 0,042	0,042
sf1	E26	Q4.1	0,005	0,104	0,044 ≤ μ ≤ 0,049	0,047
sf1	E26	Q4.2	0,006	0,110	0,051 ≤ μ ≤ 0,057	0,054
sf1	E26	Q4.3	0,007	0,113	0,062 ≤ μ ≤ 0,070	0,066
					Total	0,597
sf1	E27	Q1.1	0,007	0,064	0,111 ≤ μ ≤ 0,119	0,116
sf1	E27	Q1.2	0,005	0,048	0,093 ≤ μ ≤ 0,097	0,096
sf1	E27	Q1.3	0,008	0,069	0,109 ≤ μ ≤ 0,116	0,113
sf1	E27	Q2.1	0,009	0,039	0,238 ≤ μ ≤ 0,247	0,243
sf1	E27	Q2.2	0,015	0,067	0,216 ≤ μ ≤ 0,231	0,224
sf1	E27	Q2.3	0,006	0,026	0,225 ≤ μ ≤ 0,231	0,228
sf1	E27	Q3.1	0,008	0,033	0,253 ≤ μ ≤ 0,262	0,258
sf1	E27	Q3.2	0,007	0,030	0,245 ≤ μ ≤ 0,253	0,250
sf1	E27	Q3.3	0,009	0,034	0,256 ≤ μ ≤ 0,265	0,261
sf1	E27	Q3.4	0,006	0,024	0,241 ≤ μ ≤ 0,247	0,245
sf1	E27	Q4.1	0,016	0,070	0,217 ≤ μ ≤ 0,233	0,226
sf1	E27	Q4.2	0,008	0,031	0,245 ≤ μ ≤ 0,253	0,250
sf1	E27	Q4.3	0,010	0,041	0,247 ≤ μ ≤ 0,258	0,253
					Total	2,762
sf1	E28	Q1.1	0,009	0,075	0,116 ≤ μ ≤ 0,125	0,121
sf1	E28	Q1.2	0,002	0,025	0,091 ≤ μ ≤ 0,093	0,093
sf1	E28	Q1.3	0,007	0,060	0,112 ≤ μ ≤ 0,119	0,116
sf1	E28	Q2.1	0,008	0,031	0,245 ≤ μ ≤ 0,253	0,250
sf1	E28	Q2.2	0,025	0,110	0,217 ≤ μ ≤ 0,243	0,230
sf1	E28	Q2.3	0,007	0,031	0,237 ≤ μ ≤ 0,244	0,241
sf1	E28	Q3.1	0,009	0,035	0,263 ≤ μ ≤ 0,273	0,269
sf1	E28	Q3.2	0,008	0,033	0,249 ≤ μ ≤ 0,258	0,254
sf1	E28	Q3.3	0,009	0,034	0,268 ≤ μ ≤ 0,277	0,273
sf1	E28	Q3.4	0,007	0,028	0,252 ≤ μ ≤ 0,259	0,256
sf1	E28	Q4.1	0,021	0,089	0,223 ≤ μ ≤ 0,244	0,234
sf1	E28	Q4.2	0,009	0,034	0,259 ≤ μ ≤ 0,268	0,264
sf1	E28	Q4.3	0,006	0,021	0,264 ≤ μ ≤ 0,269	0,267
					Total	2,868

sf	cenario	consulta	desvio	coef_v	conf. 99% (n=30)	média
sf10	E01	Q1.1	0,013	0,081	0,150 ≤ μ ≤ 0,163	0,157
sf10	E01	Q1.2	0,001	0,033	0,026 ≤ μ ≤ 0,027	0,027
sf10	E01	Q1.3	0,001	0,042	0,031 ≤ μ ≤ 0,032	0,032
sf10	E01	Q2.1	0,010	0,062	0,157 ≤ μ ≤ 0,167	0,162
sf10	E01	Q2.2	0,011	0,077	0,138 ≤ μ ≤ 0,149	0,144
sf10	E01	Q2.3	0,007	0,037	0,174 ≤ μ ≤ 0,181	0,178
sf10	E01	Q3.1	0,015	0,079	0,183 ≤ μ ≤ 0,198	0,191
sf10	E01	Q3.2	0,005	0,069	0,067 ≤ μ ≤ 0,072	0,070
sf10	E01	Q3.3	0,008	0,051	0,143 ≤ μ ≤ 0,151	0,147
sf10	E01	Q3.4	0,006	0,076	0,072 ≤ μ ≤ 0,078	0,075
sf10	E01	Q4.1	0,021	0,075	0,270 ≤ μ ≤ 0,291	0,281
sf10	E01	Q4.2	0,015	0,059	0,248 ≤ μ ≤ 0,264	0,257
sf10	E01	Q4.3	0,013	0,131	0,090 ≤ μ ≤ 0,103	0,097
					Total	1,818
sf10	E02	Q1.1	0,006	0,036	0,168 ≤ μ ≤ 0,174	0,171
sf10	E02	Q1.2	0,006	0,135	0,044 ≤ μ ≤ 0,050	0,047
sf10	E02	Q1.3	0,007	0,121	0,051 ≤ μ ≤ 0,058	0,055
sf10	E02	Q2.1	0,011	0,068	0,156 ≤ μ ≤ 0,167	0,162
sf10	E02	Q2.2	0,011	0,079	0,133 ≤ μ ≤ 0,144	0,139
sf10	E02	Q2.3	0,009	0,051	0,172 ≤ μ ≤ 0,181	0,177
sf10	E02	Q3.1	0,006	0,033	0,182 ≤ μ ≤ 0,189	0,186
sf10	E02	Q3.2	0,005	0,068	0,065 ≤ μ ≤ 0,069	0,067
sf10	E02	Q3.3	0,010	0,070	0,143 ≤ μ ≤ 0,154	0,149
sf10	E02	Q3.4	0,004	0,064	0,064 ≤ μ ≤ 0,069	0,067
sf10	E02	Q4.1	0,019	0,069	0,270 ≤ μ ≤ 0,290	0,281
sf10	E02	Q4.2	0,014	0,056	0,245 ≤ μ ≤ 0,259	0,252
sf10	E02	Q4.3	0,011	0,122	0,088 ≤ μ ≤ 0,099	0,094
					Total	1,847
sf10	E03	Q1.1	0,008	0,060	0,137 ≤ μ ≤ 0,146	0,142
sf10	E03	Q1.2	0,001	0,045	0,028 ≤ μ ≤ 0,029	0,029
sf10	E03	Q1.3	0,001	0,027	0,032 ≤ μ ≤ 0,033	0,033
sf10	E03	Q2.1	0,010	0,047	0,204 ≤ μ ≤ 0,214	0,210
sf10	E03	Q2.2	0,014	0,049	0,272 ≤ μ ≤ 0,286	0,280
sf10	E03	Q2.3	0,003	0,018	0,145 ≤ μ ≤ 0,147	0,146
sf10	E03	Q3.1	0,001	0,010	0,140 ≤ μ ≤ 0,142	0,141
sf10	E03	Q3.2	0,001	0,024	0,039 ≤ μ ≤ 0,040	0,040
sf10	E03	Q3.3	0,002	0,024	0,079 ≤ μ ≤ 0,081	0,081
sf10	E03	Q3.4	0,002	0,029	0,056 ≤ μ ≤ 0,057	0,057
sf10	E03	Q4.1	0,025	0,068	0,354 ≤ μ ≤ 0,379	0,367
sf10	E03	Q4.2	0,023	0,074	0,299 ≤ μ ≤ 0,322	0,311
sf10	E03	Q4.3	0,002	0,028	0,063 ≤ μ ≤ 0,065	0,064
					Total	1,901
sf10	E04	Q1.1	0,005	0,032	0,167 ≤ μ ≤ 0,172	0,170
sf10	E04	Q1.2	0,006	0,127	0,043 ≤ μ ≤ 0,049	0,046
sf10	E04	Q1.3	0,007	0,138	0,046 ≤ μ ≤ 0,054	0,050
sf10	E04	Q2.1	0,008	0,049	0,167 ≤ μ ≤ 0,175	0,172
sf10	E04	Q2.2	0,004	0,029	0,150 ≤ μ ≤ 0,155	0,153
sf10	E04	Q2.3	0,009	0,045	0,194 ≤ μ ≤ 0,203	0,199
sf10	E04	Q3.1	0,012	0,064	0,188 ≤ μ ≤ 0,201	0,195
sf10	E04	Q3.2	0,009	0,124	0,070 ≤ μ ≤ 0,080	0,076
sf10	E04	Q3.3	0,005	0,037	0,140 ≤ μ ≤ 0,145	0,143
sf10	E04	Q3.4	0,003	0,042	0,072 ≤ μ ≤ 0,075	0,074
sf10	E04	Q4.1	0,016	0,055	0,277 ≤ μ ≤ 0,293	0,286
sf10	E04	Q4.2	0,010	0,040	0,255 ≤ μ ≤ 0,266	0,261
sf10	E04	Q4.3	0,008	0,078	0,102 ≤ μ ≤ 0,110	0,106
					Total	1,931
sf10	E05	Q1.1	0,008	0,048	0,168 ≤ μ ≤ 0,176	0,173
sf10	E05	Q1.2	0,006	0,151	0,038 ≤ μ ≤ 0,044	0,041
sf10	E05	Q1.3	0,007	0,135	0,046 ≤ μ ≤ 0,053	0,050
sf10	E05	Q2.1	0,008	0,046	0,167 ≤ μ ≤ 0,175	0,172
sf10	E05	Q2.2	0,008	0,044	0,169 ≤ μ ≤ 0,177	0,173
sf10	E05	Q2.3	0,008	0,036	0,205 ≤ μ ≤ 0,212	0,209
sf10	E05	Q3.1	0,008	0,042	0,187 ≤ μ ≤ 0,196	0,192
sf10	E05	Q3.2	0,003	0,041	0,071 ≤ μ ≤ 0,074	0,073
sf10	E05	Q3.3	0,011	0,075	0,141 ≤ μ ≤ 0,152	0,147
sf10	E05	Q3.4	0,003	0,044	0,069 ≤ μ ≤ 0,072	0,071
sf10	E05	Q4.1	0,014	0,048	0,284 ≤ μ ≤ 0,298	0,292
sf10	E05	Q4.2	0,009	0,033	0,280 ≤ μ ≤ 0,290	0,286
sf10	E05	Q4.3	0,010	0,092	0,099 ≤ μ ≤ 0,109	0,105
					Total	1,984

sf	cenario	consulta	desvio	coef_v	conf. 99% (n=30)	média
sf10	E06	Q1.1	0,006	0,044	0,140 ≤ μ ≤ 0,146	0,143
sf10	E06	Q1.2	0,001	0,033	0,029 ≤ μ ≤ 0,030	0,030
sf10	E06	Q1.3	0,001	0,029	0,034 ≤ μ ≤ 0,035	0,035
sf10	E06	Q2.1	0,011	0,039	0,268 ≤ μ ≤ 0,279	0,274
sf10	E06	Q2.2	0,014	0,044	0,304 ≤ μ ≤ 0,318	0,312
sf10	E06	Q2.3	0,010	0,044	0,227 ≤ μ ≤ 0,238	0,233
sf10	E06	Q3.1	0,002	0,016	0,120 ≤ μ ≤ 0,122	0,121
sf10	E06	Q3.2	0,002	0,038	0,043 ≤ μ ≤ 0,044	0,044
sf10	E06	Q3.3	0,002	0,024	0,095 ≤ μ ≤ 0,098	0,097
sf10	E06	Q3.4	0,002	0,028	0,064 ≤ μ ≤ 0,066	0,066
sf10	E06	Q4.1	0,020	0,028	0,695 ≤ μ ≤ 0,715	0,705
sf10	E06	Q4.2	0,023	0,045	0,499 ≤ μ ≤ 0,522	0,511
sf10	E06	Q4.3	0,021	0,151	0,129 ≤ μ ≤ 0,150	0,140
					Total	2,711
sf10	E07	Q1.1	0,014	0,090	0,151 ≤ μ ≤ 0,166	0,159
sf10	E07	Q1.2	0,004	0,088	0,042 ≤ μ ≤ 0,046	0,045
sf10	E07	Q1.3	0,002	0,049	0,047 ≤ μ ≤ 0,050	0,049
sf10	E07	Q2.1	0,012	0,074	0,160 ≤ μ ≤ 0,173	0,167
sf10	E07	Q2.2	0,011	0,074	0,147 ≤ μ ≤ 0,159	0,153
sf10	E07	Q2.3	0,014	0,074	0,182 ≤ μ ≤ 0,197	0,190
sf10	E07	Q3.1	0,016	0,076	0,197 ≤ μ ≤ 0,213	0,206
sf10	E07	Q3.2	0,004	0,055	0,069 ≤ μ ≤ 0,074	0,072
sf10	E07	Q3.3	0,012	0,097	0,119 ≤ μ ≤ 0,132	0,126
sf10	E07	Q3.4	0,010	0,126	0,075 ≤ μ ≤ 0,085	0,080
sf10	E07	Q4.1	0,024	0,077	0,302 ≤ μ ≤ 0,327	0,315
sf10	E07	Q4.2	0,012	0,045	0,249 ≤ μ ≤ 0,261	0,256
sf10	E07	Q4.3	0,013	0,140	0,084 ≤ μ ≤ 0,098	0,092
					Total	1,910
sf10	E08	Q1.1	0,013	0,082	0,150 ≤ μ ≤ 0,163	0,157
sf10	E08	Q1.2	0,003	0,069	0,044 ≤ μ ≤ 0,047	0,046
sf10	E08	Q1.3	0,003	0,055	0,047 ≤ μ ≤ 0,050	0,049
sf10	E08	Q2.1	0,013	0,077	0,158 ≤ μ ≤ 0,171	0,165
sf10	E08	Q2.2	0,010	0,068	0,147 ≤ μ ≤ 0,157	0,153
sf10	E08	Q2.3	0,011	0,061	0,177 ≤ μ ≤ 0,189	0,184
sf10	E08	Q3.1	0,015	0,071	0,200 ≤ μ ≤ 0,215	0,208
sf10	E08	Q3.2	0,006	0,087	0,065 ≤ μ ≤ 0,071	0,068
sf10	E08	Q3.3	0,010	0,089	0,109 ≤ μ ≤ 0,120	0,115
sf10	E08	Q3.4	0,009	0,119	0,069 ≤ μ ≤ 0,078	0,074
sf10	E08	Q4.1	0,017	0,054	0,303 ≤ μ ≤ 0,321	0,312
sf10	E08	Q4.2	0,014	0,055	0,252 ≤ μ ≤ 0,267	0,260
sf10	E08	Q4.3	0,013	0,143	0,083 ≤ μ ≤ 0,096	0,090
					Total	1,881
sf10	E09	Q1.1	0,013	0,078	0,158 ≤ μ ≤ 0,172	0,166
sf10	E09	Q1.2	0,003	0,039	0,064 ≤ μ ≤ 0,066	0,066
sf10	E09	Q1.3	0,003	0,047	0,066 ≤ μ ≤ 0,069	0,068
sf10	E09	Q2.1	0,008	0,046	0,167 ≤ μ ≤ 0,175	0,171
sf10	E09	Q2.2	0,010	0,073	0,130 ≤ μ ≤ 0,140	0,136
sf10	E09	Q2.3	0,009	0,078	0,115 ≤ μ ≤ 0,125	0,120
sf10	E09	Q3.1	0,003	0,021	0,160 ≤ μ ≤ 0,163	0,162
sf10	E09	Q3.2	0,002	0,038	0,038 ≤ μ ≤ 0,040	0,040
sf10	E09	Q3.3	0,002	0,021	0,099 ≤ μ ≤ 0,101	0,101
sf10	E09	Q3.4	0,004	0,076	0,054 ≤ μ ≤ 0,058	0,056
sf10	E09	Q4.1	0,019	0,050	0,365 ≤ μ ≤ 0,384	0,375
sf10	E09	Q4.2	0,012	0,056	0,214 ≤ μ ≤ 0,227	0,221
sf10	E09	Q4.3	0,010	0,141	0,068 ≤ μ ≤ 0,078	0,073
					Total	1,755
sf10	E10	Q1.1	0,014	0,079	0,174 ≤ μ ≤ 0,188	0,181
sf10	E10	Q1.2	0,003	0,042	0,068 ≤ μ ≤ 0,071	0,070
sf10	E10	Q1.3	0,005	0,071	0,072 ≤ μ ≤ 0,077	0,075
sf10	E10	Q2.1	0,009	0,052	0,172 ≤ μ ≤ 0,181	0,177
sf10	E10	Q2.2	0,009	0,054	0,154 ≤ μ ≤ 0,163	0,159
sf10	E10	Q2.3	0,008	0,040	0,191 ≤ μ ≤ 0,199	0,196
sf10	E10	Q3.1	0,011	0,053	0,205 ≤ μ ≤ 0,216	0,211
sf10	E10	Q3.2	0,002	0,032	0,069 ≤ μ ≤ 0,071	0,071
sf10	E10	Q3.3	0,010	0,075	0,127 ≤ μ ≤ 0,137	0,132
sf10	E10	Q3.4	0,004	0,046	0,081 ≤ μ ≤ 0,084	0,083
sf10	E10	Q4.1	0,020	0,061	0,307 ≤ μ ≤ 0,327	0,318
sf10	E10	Q4.2	0,013	0,046	0,264 ≤ μ ≤ 0,277	0,271
sf10	E10	Q4.3	0,008	0,078	0,102 ≤ μ ≤ 0,111	0,107
					Total	2,050

sf	cenario	consulta	desvio	coef_v	conf. 99% (n=30)	média
sf10	E11	Q1.1	0,015	0,097	0,146 ≤ μ ≤ 0,161	0,154
sf10	E11	Q1.2	0,004	0,094	0,040 ≤ μ ≤ 0,044	0,042
sf10	E11	Q1.3	0,001	0,040	0,030 ≤ μ ≤ 0,031	0,031
sf10	E11	Q2.1	0,009	0,051	0,171 ≤ μ ≤ 0,180	0,176
sf10	E11	Q2.2	0,011	0,060	0,175 ≤ μ ≤ 0,187	0,182
sf10	E11	Q2.3	0,011	0,050	0,207 ≤ μ ≤ 0,218	0,213
sf10	E11	Q3.1	0,008	0,040	0,193 ≤ μ ≤ 0,201	0,198
sf10	E11	Q3.2	0,003	0,044	0,069 ≤ μ ≤ 0,072	0,071
sf10	E11	Q3.3	0,007	0,057	0,122 ≤ μ ≤ 0,130	0,127
sf10	E11	Q3.4	0,008	0,095	0,076 ≤ μ ≤ 0,084	0,080
sf10	E11	Q4.1	0,018	0,056	0,316 ≤ μ ≤ 0,335	0,326
sf10	E11	Q4.2	0,012	0,043	0,275 ≤ μ ≤ 0,288	0,282
sf10	E11	Q4.3	0,009	0,083	0,102 ≤ μ ≤ 0,111	0,107
					Total	1,988
sf10	E12	Q1.1	0,013	0,078	0,156 ≤ μ ≤ 0,169	0,163
sf10	E12	Q1.2	0,003	0,049	0,063 ≤ μ ≤ 0,066	0,065
sf10	E12	Q1.3	0,003	0,043	0,067 ≤ μ ≤ 0,070	0,069
sf10	E12	Q2.1	0,018	0,073	0,231 ≤ μ ≤ 0,249	0,241
sf10	E12	Q2.2	0,012	0,048	0,249 ≤ μ ≤ 0,262	0,256
sf10	E12	Q2.3	0,014	0,065	0,202 ≤ μ ≤ 0,216	0,210
sf10	E12	Q3.1	0,003	0,020	0,130 ≤ μ ≤ 0,133	0,132
sf10	E12	Q3.2	0,002	0,022	0,071 ≤ μ ≤ 0,073	0,073
sf10	E12	Q3.3	0,003	0,028	0,102 ≤ μ ≤ 0,105	0,104
sf10	E12	Q3.4	0,003	0,035	0,092 ≤ μ ≤ 0,095	0,094
sf10	E12	Q4.1	0,013	0,019	0,674 ≤ μ ≤ 0,687	0,681
sf10	E12	Q4.2	0,023	0,076	0,290 ≤ μ ≤ 0,313	0,302
sf10	E12	Q4.3	0,015	0,071	0,201 ≤ μ ≤ 0,216	0,209
					Total	2,599
sf10	E13	Q1.1	0,005	0,009	0,561 ≤ μ ≤ 0,566	0,564
sf10	E13	Q1.2	0,019	0,034	0,539 ≤ μ ≤ 0,558	0,549
sf10	E13	Q1.3	0,011	0,021	0,534 ≤ μ ≤ 0,546	0,541
sf10	E13	Q2.1	0,005	0,015	0,315 ≤ μ ≤ 0,319	0,317
sf10	E13	Q2.2	0,011	0,035	0,301 ≤ μ ≤ 0,312	0,307
sf10	E13	Q2.3	0,006	0,021	0,303 ≤ μ ≤ 0,310	0,307
sf10	E13	Q3.1	0,028	0,068	0,393 ≤ μ ≤ 0,421	0,407
sf10	E13	Q3.2	0,010	0,031	0,316 ≤ μ ≤ 0,326	0,321
sf10	E13	Q3.3	0,009	0,031	0,297 ≤ μ ≤ 0,307	0,303
sf10	E13	Q3.4	0,005	0,016	0,304 ≤ μ ≤ 0,309	0,307
sf10	E13	Q4.1	0,019	0,044	0,418 ≤ μ ≤ 0,437	0,428
sf10	E13	Q4.2	0,016	0,040	0,388 ≤ μ ≤ 0,404	0,396
sf10	E13	Q4.3	0,019	0,056	0,330 ≤ μ ≤ 0,349	0,340
					Total	5,087
sf10	E14	Q1.1	0,010	0,017	0,558 ≤ μ ≤ 0,568	0,564
sf10	E14	Q1.2	0,013	0,025	0,528 ≤ μ ≤ 0,541	0,535
sf10	E14	Q1.3	0,013	0,025	0,530 ≤ μ ≤ 0,544	0,538
sf10	E14	Q2.1	0,007	0,023	0,315 ≤ μ ≤ 0,322	0,319
sf10	E14	Q2.2	0,007	0,023	0,296 ≤ μ ≤ 0,303	0,300
sf10	E14	Q2.3	0,004	0,013	0,303 ≤ μ ≤ 0,307	0,306
sf10	E14	Q3.1	0,017	0,041	0,400 ≤ μ ≤ 0,417	0,409
sf10	E14	Q3.2	0,013	0,043	0,304 ≤ μ ≤ 0,317	0,311
sf10	E14	Q3.3	0,010	0,034	0,291 ≤ μ ≤ 0,301	0,296
sf10	E14	Q3.4	0,006	0,021	0,305 ≤ μ ≤ 0,312	0,309
sf10	E14	Q4.1	0,018	0,043	0,413 ≤ μ ≤ 0,432	0,423
sf10	E14	Q4.2	0,020	0,048	0,395 ≤ μ ≤ 0,415	0,406
sf10	E14	Q4.3	0,016	0,048	0,319 ≤ μ ≤ 0,335	0,328
					Total	5,044
sf10	E15	Q1.1	0,004	0,006	0,632 ≤ μ ≤ 0,636	0,635
sf10	E15	Q1.2	0,005	0,010	0,530 ≤ μ ≤ 0,536	0,534
sf10	E15	Q1.3	0,005	0,009	0,528 ≤ μ ≤ 0,533	0,531
sf10	E15	Q2.1	0,014	0,044	0,312 ≤ μ ≤ 0,326	0,319
sf10	E15	Q2.2	0,017	0,042	0,394 ≤ μ ≤ 0,412	0,404
sf10	E15	Q2.3	0,013	0,044	0,289 ≤ μ ≤ 0,302	0,296
sf10	E15	Q3.1	0,011	0,029	0,378 ≤ μ ≤ 0,390	0,385
sf10	E15	Q3.2	0,015	0,051	0,295 ≤ μ ≤ 0,310	0,303
sf10	E15	Q3.3	0,011	0,040	0,279 ≤ μ ≤ 0,291	0,286
sf10	E15	Q3.4	0,012	0,040	0,290 ≤ μ ≤ 0,302	0,296
sf10	E15	Q4.1	0,019	0,041	0,464 ≤ μ ≤ 0,484	0,474
sf10	E15	Q4.2	0,018	0,042	0,416 ≤ μ ≤ 0,435	0,426
sf10	E15	Q4.3	0,002	0,007	0,295 ≤ μ ≤ 0,297	0,296
					Total	5,185

sf	cenario	consulta	desvio	coef_v	conf. 99% (n=30)	média
sf10	E16	Q1.1	0,011	0,020	0,560 ≤ μ ≤ 0,572	0,566
sf10	E16	Q1.2	0,005	0,010	0,534 ≤ μ ≤ 0,540	0,538
sf10	E16	Q1.3	0,006	0,011	0,535 ≤ μ ≤ 0,541	0,538
sf10	E16	Q2.1	0,017	0,051	0,323 ≤ μ ≤ 0,341	0,332
sf10	E16	Q2.2	0,011	0,035	0,307 ≤ μ ≤ 0,318	0,313
sf10	E16	Q2.3	0,021	0,065	0,318 ≤ μ ≤ 0,340	0,330
sf10	E16	Q3.1	0,022	0,056	0,383 ≤ μ ≤ 0,406	0,395
sf10	E16	Q3.2	0,004	0,013	0,303 ≤ μ ≤ 0,307	0,306
sf10	E16	Q3.3	0,007	0,022	0,294 ≤ μ ≤ 0,301	0,298
sf10	E16	Q3.4	0,003	0,011	0,301 ≤ μ ≤ 0,304	0,303
sf10	E16	Q4.1	0,021	0,049	0,422 ≤ μ ≤ 0,444	0,434
sf10	E16	Q4.2	0,010	0,026	0,383 ≤ μ ≤ 0,393	0,389
sf10	E16	Q4.3	0,006	0,019	0,320 ≤ μ ≤ 0,326	0,323
					Total	5,065
sf10	E17	Q1.1	0,005	0,008	0,562 ≤ μ ≤ 0,567	0,565
sf10	E17	Q1.2	0,008	0,015	0,537 ≤ μ ≤ 0,546	0,542
sf10	E17	Q1.3	0,007	0,013	0,537 ≤ μ ≤ 0,544	0,541
sf10	E17	Q2.1	0,021	0,060	0,334 ≤ μ ≤ 0,355	0,345
sf10	E17	Q2.2	0,016	0,048	0,318 ≤ μ ≤ 0,335	0,327
sf10	E17	Q2.3	0,005	0,015	0,317 ≤ μ ≤ 0,322	0,320
sf10	E17	Q3.1	0,016	0,042	0,383 ≤ μ ≤ 0,399	0,392
sf10	E17	Q3.2	0,005	0,017	0,305 ≤ μ ≤ 0,310	0,308
sf10	E17	Q3.3	0,011	0,036	0,299 ≤ μ ≤ 0,310	0,305
sf10	E17	Q3.4	0,018	0,056	0,305 ≤ μ ≤ 0,323	0,315
sf10	E17	Q4.1	0,020	0,047	0,428 ≤ μ ≤ 0,449	0,439
sf10	E17	Q4.2	0,010	0,023	0,401 ≤ μ ≤ 0,411	0,407
sf10	E17	Q4.3	0,016	0,047	0,333 ≤ μ ≤ 0,350	0,342
					Total	5,148
sf10	E18	Q1.1	0,004	0,006	0,708 ≤ μ ≤ 0,712	0,711
sf10	E18	Q1.2	0,007	0,014	0,532 ≤ μ ≤ 0,539	0,536
sf10	E18	Q1.3	0,003	0,007	0,526 ≤ μ ≤ 0,530	0,529
sf10	E18	Q2.1	0,010	0,022	0,449 ≤ μ ≤ 0,459	0,455
sf10	E18	Q2.2	0,010	0,022	0,455 ≤ μ ≤ 0,465	0,461
sf10	E18	Q2.3	0,011	0,026	0,418 ≤ μ ≤ 0,429	0,424
sf10	E18	Q3.1	0,021	0,047	0,440 ≤ μ ≤ 0,461	0,451
sf10	E18	Q3.2	0,013	0,042	0,293 ≤ μ ≤ 0,306	0,300
sf10	E18	Q3.3	0,013	0,041	0,303 ≤ μ ≤ 0,316	0,310
sf10	E18	Q3.4	0,014	0,046	0,291 ≤ μ ≤ 0,305	0,299
sf10	E18	Q4.1	0,025	0,033	0,756 ≤ μ ≤ 0,782	0,769
sf10	E18	Q4.2	0,030	0,042	0,687 ≤ μ ≤ 0,718	0,703
sf10	E18	Q4.3	0,013	0,044	0,292 ≤ μ ≤ 0,306	0,300
					Total	6,248
sf10	E19	Q1.1	0,009	0,019	0,454 ≤ μ ≤ 0,463	0,459
sf10	E19	Q1.2	0,005	0,014	0,374 ≤ μ ≤ 0,380	0,377
sf10	E19	Q1.3	0,007	0,017	0,381 ≤ μ ≤ 0,388	0,385
sf10	E19	Q2.1	0,010	0,026	0,379 ≤ μ ≤ 0,389	0,384
sf10	E19	Q2.2	0,005	0,014	0,358 ≤ μ ≤ 0,363	0,361
sf10	E19	Q2.3	0,005	0,014	0,355 ≤ μ ≤ 0,360	0,358
sf10	E19	Q3.1	0,022	0,046	0,460 ≤ μ ≤ 0,482	0,472
sf10	E19	Q3.2	0,009	0,024	0,365 ≤ μ ≤ 0,374	0,370
sf10	E19	Q3.3	0,015	0,022	0,668 ≤ μ ≤ 0,684	0,677
sf10	E19	Q3.4	0,008	0,091	0,086 ≤ μ ≤ 0,095	0,091
sf10	E19	Q4.1	0,011	0,019	0,559 ≤ μ ≤ 0,570	0,565
sf10	E19	Q4.2	0,022	0,062	0,351 ≤ μ ≤ 0,374	0,363
sf10	E19	Q4.3	0,020	0,063	0,301 ≤ μ ≤ 0,321	0,311
					Total	5,173
sf10	E20	Q1.1	0,021	0,046	0,458 ≤ μ ≤ 0,479	0,469
sf10	E20	Q1.2	0,006	0,017	0,367 ≤ μ ≤ 0,374	0,371
sf10	E20	Q1.3	0,006	0,016	0,373 ≤ μ ≤ 0,379	0,376
sf10	E20	Q2.1	0,017	0,044	0,375 ≤ μ ≤ 0,392	0,384
sf10	E20	Q2.2	0,005	0,015	0,350 ≤ μ ≤ 0,355	0,353
sf10	E20	Q2.3	0,005	0,015	0,351 ≤ μ ≤ 0,356	0,354
sf10	E20	Q3.1	0,020	0,042	0,458 ≤ μ ≤ 0,478	0,469
sf10	E20	Q3.2	0,014	0,038	0,361 ≤ μ ≤ 0,376	0,369
sf10	E20	Q3.3	0,013	0,020	0,664 ≤ μ ≤ 0,677	0,671
sf10	E20	Q3.4	0,010	0,116	0,084 ≤ μ ≤ 0,095	0,090
sf10	E20	Q4.1	0,018	0,031	0,557 ≤ μ ≤ 0,575	0,566
sf10	E20	Q4.2	0,019	0,053	0,343 ≤ μ ≤ 0,363	0,353
sf10	E20	Q4.3	0,025	0,084	0,281 ≤ μ ≤ 0,306	0,294
					Total	5,119

sf	cenario	consulta	desvio	coef_v	conf. 99% (n=30)	média
sf10	E21	Q1.1	0,004	0,009	0,445 ≤ μ ≤ 0,449	0,448
sf10	E21	Q1.2	0,004	0,012	0,364 ≤ μ ≤ 0,369	0,367
sf10	E21	Q1.3	0,006	0,016	0,368 ≤ μ ≤ 0,374	0,371
sf10	E21	Q2.1	0,016	0,040	0,386 ≤ μ ≤ 0,402	0,394
sf10	E21	Q2.2	0,003	0,009	0,346 ≤ μ ≤ 0,349	0,348
sf10	E21	Q2.3	0,016	0,045	0,338 ≤ μ ≤ 0,354	0,347
sf10	E21	Q3.1	0,020	0,048	0,404 ≤ μ ≤ 0,424	0,415
sf10	E21	Q3.2	0,016	0,047	0,338 ≤ μ ≤ 0,355	0,347
sf10	E21	Q3.3	0,012	0,034	0,338 ≤ μ ≤ 0,350	0,345
sf10	E21	Q3.4	0,015	0,044	0,339 ≤ μ ≤ 0,355	0,348
sf10	E21	Q4.1	0,016	0,031	0,507 ≤ μ ≤ 0,523	0,515
sf10	E21	Q4.2	0,020	0,045	0,426 ≤ μ ≤ 0,446	0,436
sf10	E21	Q4.3	0,017	0,047	0,359 ≤ μ ≤ 0,376	0,368
					Total	5,049
sf10	E22	Q1.1	0,021	0,045	0,451 ≤ μ ≤ 0,472	0,462
sf10	E22	Q1.2	0,004	0,010	0,368 ≤ μ ≤ 0,371	0,370
sf10	E22	Q1.3	0,004	0,012	0,373 ≤ μ ≤ 0,377	0,375
sf10	E22	Q2.1	0,018	0,047	0,382 ≤ μ ≤ 0,401	0,392
sf10	E22	Q2.2	0,017	0,045	0,365 ≤ μ ≤ 0,382	0,374
sf10	E22	Q2.3	0,018	0,048	0,361 ≤ μ ≤ 0,379	0,371
sf10	E22	Q3.1	0,021	0,047	0,440 ≤ μ ≤ 0,461	0,451
sf10	E22	Q3.2	0,004	0,011	0,350 ≤ μ ≤ 0,354	0,353
sf10	E22	Q3.3	0,018	0,026	0,665 ≤ μ ≤ 0,683	0,675
sf10	E22	Q3.4	0,008	0,090	0,082 ≤ μ ≤ 0,090	0,086
sf10	E22	Q4.1	0,010	0,018	0,551 ≤ μ ≤ 0,561	0,556
sf10	E22	Q4.2	0,018	0,048	0,360 ≤ μ ≤ 0,378	0,370
sf10	E22	Q4.3	0,023	0,075	0,290 ≤ μ ≤ 0,313	0,302
					Total	5,137
sf10	E23	Q1.1	0,006	0,013	0,449 ≤ μ ≤ 0,455	0,452
sf10	E23	Q1.2	0,004	0,012	0,368 ≤ μ ≤ 0,373	0,371
sf10	E23	Q1.3	0,003	0,009	0,375 ≤ μ ≤ 0,378	0,377
sf10	E23	Q2.1	0,014	0,035	0,379 ≤ μ ≤ 0,393	0,387
sf10	E23	Q2.2	0,019	0,050	0,370 ≤ μ ≤ 0,390	0,380
sf10	E23	Q2.3	0,005	0,014	0,365 ≤ μ ≤ 0,370	0,368
sf10	E23	Q3.1	0,017	0,037	0,454 ≤ μ ≤ 0,471	0,463
sf10	E23	Q3.2	0,005	0,013	0,352 ≤ μ ≤ 0,357	0,355
sf10	E23	Q3.3	0,011	0,016	0,668 ≤ μ ≤ 0,679	0,674
sf10	E23	Q3.4	0,007	0,082	0,083 ≤ μ ≤ 0,091	0,087
sf10	E23	Q4.1	0,015	0,026	0,561 ≤ μ ≤ 0,576	0,569
sf10	E23	Q4.2	0,024	0,062	0,371 ≤ μ ≤ 0,395	0,384
sf10	E23	Q4.3	0,023	0,076	0,297 ≤ μ ≤ 0,321	0,309
					Total	5,176
sf10	E24	Q1.1	0,004	0,009	0,438 ≤ μ ≤ 0,442	0,440
sf10	E24	Q1.2	0,004	0,010	0,354 ≤ μ ≤ 0,357	0,356
sf10	E24	Q1.3	0,004	0,010	0,362 ≤ μ ≤ 0,366	0,364
sf10	E24	Q2.1	0,009	0,019	0,501 ≤ μ ≤ 0,510	0,506
sf10	E24	Q2.2	0,005	0,009	0,509 ≤ μ ≤ 0,513	0,511
sf10	E24	Q2.3	0,009	0,020	0,457 ≤ μ ≤ 0,466	0,462
sf10	E24	Q3.1	0,009	0,021	0,438 ≤ μ ≤ 0,448	0,444
sf10	E24	Q3.2	0,017	0,047	0,349 ≤ μ ≤ 0,366	0,358
sf10	E24	Q3.3	0,014	0,040	0,348 ≤ μ ≤ 0,362	0,356
sf10	E24	Q3.4	0,014	0,039	0,347 ≤ μ ≤ 0,361	0,354
sf10	E24	Q4.1	0,012	0,014	0,837 ≤ μ ≤ 0,849	0,843
sf10	E24	Q4.2	0,023	0,040	0,550 ≤ μ ≤ 0,573	0,562
sf10	E24	Q4.3	0,009	0,019	0,471 ≤ μ ≤ 0,480	0,476
					Total	6,032
sf10	E25	Q1.1	0,004	0,039	0,092 ≤ μ ≤ 0,096	0,095
sf10	E25	Q1.2	0,001	0,040	0,025 ≤ μ ≤ 0,026	0,026
sf10	E25	Q1.3	0,001	0,042	0,018 ≤ μ ≤ 0,019	0,019
sf10	E25	Q2.1	0,007	0,078	0,084 ≤ μ ≤ 0,091	0,088
sf10	E25	Q2.2	0,004	0,048	0,077 ≤ μ ≤ 0,081	0,079
sf10	E25	Q2.3	0,004	0,058	0,062 ≤ μ ≤ 0,066	0,064
sf10	E25	Q3.1	0,008	0,110	0,070 ≤ μ ≤ 0,078	0,074
sf10	E25	Q3.2	0,008	0,077	0,093 ≤ μ ≤ 0,101	0,098
sf10	E25	Q3.3	0,003	0,051	0,059 ≤ μ ≤ 0,062	0,061
sf10	E25	Q3.4	0,002	0,042	0,048 ≤ μ ≤ 0,050	0,050
sf10	E25	Q4.1	0,003	0,033	0,090 ≤ μ ≤ 0,093	0,092
sf10	E25	Q4.2	0,008	0,117	0,063 ≤ μ ≤ 0,071	0,068
sf10	E25	Q4.3	0,008	0,073	0,103 ≤ μ ≤ 0,111	0,108
					Total	0,922

sf	cenario	consulta	desvio	coef_v	conf. 99% (n=30)	média
sf10	E26	Q1.1	0,002	0,025	0,094 ≤ μ ≤ 0,097	0,096
sf10	E26	Q1.2	0,001	0,030	0,027 ≤ μ ≤ 0,028	0,028
sf10	E26	Q1.3	0,001	0,052	0,019 ≤ μ ≤ 0,020	0,020
sf10	E26	Q2.1	0,004	0,049	0,087 ≤ μ ≤ 0,092	0,090
sf10	E26	Q2.2	0,003	0,039	0,079 ≤ μ ≤ 0,082	0,081
sf10	E26	Q2.3	0,002	0,032	0,069 ≤ μ ≤ 0,071	0,070
sf10	E26	Q3.1	0,004	0,060	0,066 ≤ μ ≤ 0,071	0,069
sf10	E26	Q3.2	0,008	0,093	0,085 ≤ μ ≤ 0,093	0,089
sf10	E26	Q3.3	0,002	0,037	0,061 ≤ μ ≤ 0,063	0,062
sf10	E26	Q3.4	0,002	0,036	0,049 ≤ μ ≤ 0,051	0,051
sf10	E26	Q4.1	0,004	0,046	0,094 ≤ μ ≤ 0,098	0,097
sf10	E26	Q4.2	0,008	0,109	0,071 ≤ μ ≤ 0,079	0,076
sf10	E26	Q4.3	0,010	0,089	0,107 ≤ μ ≤ 0,118	0,113
					Total	0,942
sf10	E27	Q1.1	0,020	0,036	0,547 ≤ μ ≤ 0,567	0,557
sf10	E27	Q1.2	0,014	0,026	0,532 ≤ μ ≤ 0,546	0,539
sf10	E27	Q1.3	0,021	0,037	0,551 ≤ μ ≤ 0,572	0,562
sf10	E27	Q2.1	0,038	0,030	1,246 ≤ μ ≤ 1,284	1,265
sf10	E27	Q2.2	0,046	0,035	1,307 ≤ μ ≤ 1,354	1,331
sf10	E27	Q2.3	0,034	0,027	1,236 ≤ μ ≤ 1,271	1,254
sf10	E27	Q3.1	0,028	0,022	1,292 ≤ μ ≤ 1,321	1,307
sf10	E27	Q3.2	0,041	0,035	1,156 ≤ μ ≤ 1,198	1,177
sf10	E27	Q3.3	0,008	0,006	1,335 ≤ μ ≤ 1,343	1,340
sf10	E27	Q3.4	0,030	0,023	1,282 ≤ μ ≤ 1,312	1,297
sf10	E27	Q4.1	0,005	0,004	1,242 ≤ μ ≤ 1,247	1,245
sf10	E27	Q4.2	0,007	0,006	1,230 ≤ μ ≤ 1,238	1,234
sf10	E27	Q4.3	0,007	0,006	1,209 ≤ μ ≤ 1,216	1,213
					Total	14,324
sf10	E28	Q1.1	0,021	0,035	0,594 ≤ μ ≤ 0,616	0,606
sf10	E28	Q1.2	0,017	0,030	0,569 ≤ μ ≤ 0,586	0,578
sf10	E28	Q1.3	0,011	0,019	0,578 ≤ μ ≤ 0,590	0,585
sf10	E28	Q2.1	0,022	0,016	1,362 ≤ μ ≤ 1,385	1,374
sf10	E28	Q2.2	0,045	0,031	1,437 ≤ μ ≤ 1,483	1,460
sf10	E28	Q2.3	0,017	0,012	1,390 ≤ μ ≤ 1,408	1,399
sf10	E28	Q3.1	0,036	0,025	1,416 ≤ μ ≤ 1,453	1,435
sf10	E28	Q3.2	0,024	0,019	1,260 ≤ μ ≤ 1,285	1,273
sf10	E28	Q3.3	0,020	0,013	1,480 ≤ μ ≤ 1,500	1,491
sf10	E28	Q3.4	0,017	0,012	1,416 ≤ μ ≤ 1,433	1,425
sf10	E28	Q4.1	0,009	0,007	1,346 ≤ μ ≤ 1,356	1,351
sf10	E28	Q4.2	0,004	0,003	1,338 ≤ μ ≤ 1,343	1,341
sf10	E28	Q4.3	0,005	0,004	1,334 ≤ μ ≤ 1,339	1,337
					Total	15,655

sf	cenario	consulta	desvio	coef_v	confiança 99%	média
sf100	E01	Q1.1	0,016	0,027	0,578 ≤ μ ≤ 0,594	0,586
sf100	E01	Q1.2	0,001	0,034	0,024 ≤ μ ≤ 0,025	0,025
sf100	E01	Q1.3	0,001	0,030	0,039 ≤ μ ≤ 0,040	0,040
sf100	E01	Q2.1	0,012	0,021	0,544 ≤ μ ≤ 0,556	0,551
sf100	E01	Q2.2	0,020	0,040	0,486 ≤ μ ≤ 0,506	0,497
sf100	E01	Q2.3	0,020	0,034	0,575 ≤ μ ≤ 0,596	0,586
sf100	E01	Q3.1	0,014	0,011	1,369 ≤ μ ≤ 1,383	1,377
sf100	E01	Q3.2	0,004	0,056	0,070 ≤ μ ≤ 0,074	0,072
sf100	E01	Q3.3	0,019	0,019	1,005 ≤ μ ≤ 1,025	1,015
sf100	E01	Q3.4	0,010	0,078	0,127 ≤ μ ≤ 0,137	0,133
sf100	E01	Q4.1	0,021	0,010	2,054 ≤ μ ≤ 2,076	2,065
sf100	E01	Q4.2	0,020	0,026	0,740 ≤ μ ≤ 0,760	0,751
sf100	E01	Q4.3	0,012	0,128	0,088 ≤ μ ≤ 0,101	0,095
					Total	7,793
sf100	E02	Q1.1	0,018	0,031	0,580 ≤ μ ≤ 0,599	0,590
sf100	E02	Q1.2	0,001	0,035	0,024 ≤ μ ≤ 0,025	0,025
sf100	E02	Q1.3	0,002	0,039	0,039 ≤ μ ≤ 0,041	0,041
sf100	E02	Q2.1	0,020	0,036	0,533 ≤ μ ≤ 0,554	0,544
sf100	E02	Q2.2	0,026	0,054	0,470 ≤ μ ≤ 0,497	0,484
sf100	E02	Q2.3	0,025	0,043	0,565 ≤ μ ≤ 0,591	0,579
sf100	E02	Q3.1	0,016	0,012	1,353 ≤ μ ≤ 1,369	1,362
sf100	E02	Q3.2	0,006	0,092	0,065 ≤ μ ≤ 0,072	0,069
sf100	E02	Q3.3	0,014	0,014	0,994 ≤ μ ≤ 1,008	1,002
sf100	E02	Q3.4	0,005	0,043	0,124 ≤ μ ≤ 0,129	0,127
sf100	E02	Q4.1	0,016	0,008	2,049 ≤ μ ≤ 2,065	2,057
sf100	E02	Q4.2	0,027	0,037	0,734 ≤ μ ≤ 0,763	0,749
sf100	E02	Q4.3	0,012	0,120	0,092 ≤ μ ≤ 0,104	0,099
					Total	7,728
sf100	E03	Q1.1	0,025	0,046	0,538 ≤ μ ≤ 0,564	0,551
sf100	E03	Q1.2	0,002	0,046	0,048 ≤ μ ≤ 0,050	0,049
sf100	E03	Q1.3	0,002	0,040	0,061 ≤ μ ≤ 0,064	0,063
sf100	E03	Q2.1	0,006	0,007	0,950 ≤ μ ≤ 0,957	0,954
sf100	E03	Q2.2	0,034	0,013	2,596 ≤ μ ≤ 2,631	2,614
sf100	E03	Q2.3	0,012	0,021	0,568 ≤ μ ≤ 0,581	0,575
sf100	E03	Q3.1	0,037	0,028	1,282 ≤ μ ≤ 1,319	1,301
sf100	E03	Q3.2	0,001	0,029	0,040 ≤ μ ≤ 0,041	0,041
sf100	E03	Q3.3	0,011	0,011	0,963 ≤ μ ≤ 0,974	0,969
sf100	E03	Q3.4	0,014	0,066	0,200 ≤ μ ≤ 0,214	0,207
sf100	E03	Q4.1	0,031	0,008	3,690 ≤ μ ≤ 3,721	3,706
sf100	E03	Q4.2	0,030	0,008	3,612 ≤ μ ≤ 3,642	3,628
sf100	E03	Q4.3	0,001	0,014	0,081 ≤ μ ≤ 0,083	0,083
					Total	14,741
sf100	E04	Q1.1	0,021	0,036	0,560 ≤ μ ≤ 0,581	0,571
sf100	E04	Q1.2	0,004	0,041	0,083 ≤ μ ≤ 0,086	0,085
sf100	E04	Q1.3	0,006	0,056	0,096 ≤ μ ≤ 0,102	0,100
sf100	E04	Q2.1	0,027	0,049	0,529 ≤ μ ≤ 0,556	0,543
sf100	E04	Q2.2	0,024	0,049	0,476 ≤ μ ≤ 0,500	0,489
sf100	E04	Q2.3	0,022	0,037	0,577 ≤ μ ≤ 0,599	0,589
sf100	E04	Q3.1	0,013	0,009	1,367 ≤ μ ≤ 1,380	1,374
sf100	E04	Q3.2	0,009	0,063	0,146 ≤ μ ≤ 0,155	0,151
sf100	E04	Q3.3	0,013	0,011	1,123 ≤ μ ≤ 1,136	1,130
sf100	E04	Q3.4	0,011	0,050	0,205 ≤ μ ≤ 0,215	0,210
sf100	E04	Q4.1	0,029	0,014	2,084 ≤ μ ≤ 2,113	2,099
sf100	E04	Q4.2	0,022	0,029	0,758 ≤ μ ≤ 0,781	0,770
sf100	E04	Q4.3	0,011	0,068	0,150 ≤ μ ≤ 0,161	0,156
					Total	8,267
sf100	E05	Q1.1	0,019	0,032	0,574 ≤ μ ≤ 0,593	0,584
sf100	E05	Q1.2	0,001	0,030	0,023 ≤ μ ≤ 0,023	0,024
sf100	E05	Q1.3	0,001	0,034	0,036 ≤ μ ≤ 0,038	0,038
sf100	E05	Q2.1	0,026	0,046	0,540 ≤ μ ≤ 0,566	0,554
sf100	E05	Q2.2	0,027	0,053	0,495 ≤ μ ≤ 0,523	0,509
sf100	E05	Q2.3	0,026	0,042	0,591 ≤ μ ≤ 0,618	0,605
sf100	E05	Q3.1	0,012	0,009	1,385 ≤ μ ≤ 1,398	1,392
sf100	E05	Q3.2	0,010	0,057	0,173 ≤ μ ≤ 0,184	0,179
sf100	E05	Q3.3	0,034	0,033	1,014 ≤ μ ≤ 1,050	1,033
sf100	E05	Q3.4	0,011	0,074	0,143 ≤ μ ≤ 0,154	0,149
sf100	E05	Q4.1	0,016	0,008	2,119 ≤ μ ≤ 2,135	2,128
sf100	E05	Q4.2	0,027	0,033	0,802 ≤ μ ≤ 0,829	0,816
sf100	E05	Q4.3	0,012	0,069	0,172 ≤ μ ≤ 0,185	0,179
					Total	8,190

sf	cenario	consulta	desvio	coef_v	confiança 99%	média
sf100	E06	Q1.1	0,027	0,048	0,538 ≤ μ ≤ 0,566	0,553
sf100	E06	Q1.2	0,002	0,049	0,047 ≤ μ ≤ 0,050	0,049
sf100	E06	Q1.3	0,005	0,071	0,062 ≤ μ ≤ 0,067	0,065
sf100	E06	Q2.1	0,020	0,018	1,137 ≤ μ ≤ 1,158	1,148
sf100	E06	Q2.2	0,029	0,010	2,747 ≤ μ ≤ 2,776	2,762
sf100	E06	Q2.3	0,008	0,011	0,758 ≤ μ ≤ 0,766	0,762
sf100	E06	Q3.1	0,029	0,008	3,708 ≤ μ ≤ 3,738	3,723
sf100	E06	Q3.2	0,016	0,040	0,404 ≤ μ ≤ 0,420	0,413
sf100	E06	Q3.3	0,008	0,007	1,197 ≤ μ ≤ 1,205	1,202
sf100	E06	Q3.4	0,020	0,094	0,203 ≤ μ ≤ 0,224	0,214
sf100	E06	Q4.1	0,043	0,011	4,021 ≤ μ ≤ 4,064	4,043
sf100	E06	Q4.2	0,048	0,010	4,538 ≤ μ ≤ 4,587	4,563
sf100	E06	Q4.3	0,005	0,033	0,144 ≤ μ ≤ 0,149	0,147
					Total	19,644
sf100	E07	Q1.1	0,028	0,046	0,589 ≤ μ ≤ 0,617	0,604
sf100	E07	Q1.2	0,001	0,042	0,025 ≤ μ ≤ 0,026	0,026
sf100	E07	Q1.3	0,001	0,022	0,046 ≤ μ ≤ 0,047	0,047
sf100	E07	Q2.1	0,025	0,045	0,535 ≤ μ ≤ 0,560	0,548
sf100	E07	Q2.2	0,023	0,046	0,491 ≤ μ ≤ 0,514	0,503
sf100	E07	Q2.3	0,020	0,033	0,585 ≤ μ ≤ 0,605	0,595
sf100	E07	Q3.1	0,024	0,018	1,326 ≤ μ ≤ 1,351	1,339
sf100	E07	Q3.2	0,006	0,080	0,072 ≤ μ ≤ 0,078	0,075
sf100	E07	Q3.3	0,025	0,025	0,984 ≤ μ ≤ 1,009	0,997
sf100	E07	Q3.4	0,011	0,069	0,150 ≤ μ ≤ 0,161	0,156
sf100	E07	Q4.1	0,029	0,014	2,024 ≤ μ ≤ 2,053	2,039
sf100	E07	Q4.2	0,031	0,040	0,775 ≤ μ ≤ 0,807	0,792
sf100	E07	Q4.3	0,013	0,128	0,095 ≤ μ ≤ 0,108	0,102
					Total	7,823
sf100	E08	Q1.1	0,027	0,044	0,588 ≤ μ ≤ 0,615	0,602
sf100	E08	Q1.2	0,001	0,028	0,025 ≤ μ ≤ 0,025	0,025
sf100	E08	Q1.3	0,001	0,031	0,045 ≤ μ ≤ 0,046	0,046
sf100	E08	Q2.1	0,025	0,047	0,522 ≤ μ ≤ 0,547	0,535
sf100	E08	Q2.2	0,020	0,041	0,473 ≤ μ ≤ 0,493	0,484
sf100	E08	Q2.3	0,017	0,030	0,581 ≤ μ ≤ 0,599	0,590
sf100	E08	Q3.1	0,028	0,021	1,323 ≤ μ ≤ 1,352	1,338
sf100	E08	Q3.2	0,007	0,097	0,067 ≤ μ ≤ 0,074	0,071
sf100	E08	Q3.3	0,028	0,029	0,954 ≤ μ ≤ 0,983	0,969
sf100	E08	Q3.4	0,010	0,069	0,142 ≤ μ ≤ 0,153	0,148
sf100	E08	Q4.1	0,036	0,018	2,014 ≤ μ ≤ 2,051	2,033
sf100	E08	Q4.2	0,032	0,040	0,772 ≤ μ ≤ 0,804	0,789
sf100	E08	Q4.3	0,013	0,130	0,094 ≤ μ ≤ 0,107	0,101
					Total	7,731
sf100	E09	Q1.1	0,024	0,040	0,587 ≤ μ ≤ 0,612	0,600
sf100	E09	Q1.2	0,002	0,043	0,050 ≤ μ ≤ 0,053	0,052
sf100	E09	Q1.3	0,003	0,042	0,070 ≤ μ ≤ 0,073	0,072
sf100	E09	Q2.1	0,013	0,020	0,656 ≤ μ ≤ 0,669	0,663
sf100	E09	Q2.2	0,022	0,041	0,531 ≤ μ ≤ 0,554	0,543
sf100	E09	Q2.3	0,014	0,024	0,578 ≤ μ ≤ 0,593	0,586
sf100	E09	Q3.1	0,017	0,014	1,229 ≤ μ ≤ 1,247	1,238
sf100	E09	Q3.2	0,001	0,031	0,040 ≤ μ ≤ 0,041	0,041
sf100	E09	Q3.3	0,022	0,023	0,937 ≤ μ ≤ 0,960	0,949
sf100	E09	Q3.4	0,004	0,041	0,088 ≤ μ ≤ 0,091	0,090
sf100	E09	Q4.1	0,030	0,013	2,281 ≤ μ ≤ 2,313	2,297
sf100	E09	Q4.2	0,016	0,018	0,872 ≤ μ ≤ 0,888	0,880
sf100	E09	Q4.3	0,001	0,014	0,086 ≤ μ ≤ 0,087	0,087
					Total	8,098
sf100	E10	Q1.1	0,029	0,047	0,589 ≤ μ ≤ 0,619	0,604
sf100	E10	Q1.2	0,001	0,032	0,023 ≤ μ ≤ 0,023	0,024
sf100	E10	Q1.3	0,001	0,027	0,043 ≤ μ ≤ 0,044	0,044
sf100	E10	Q2.1	0,020	0,036	0,557 ≤ μ ≤ 0,578	0,568
sf100	E10	Q2.2	0,022	0,045	0,483 ≤ μ ≤ 0,506	0,495
sf100	E10	Q2.3	0,024	0,039	0,595 ≤ μ ≤ 0,620	0,608
sf100	E10	Q3.1	0,055	0,040	1,363 ≤ μ ≤ 1,420	1,392
sf100	E10	Q3.2	0,009	0,070	0,125 ≤ μ ≤ 0,135	0,131
sf100	E10	Q3.3	0,016	0,015	1,076 ≤ μ ≤ 1,093	1,085
sf100	E10	Q3.4	0,011	0,049	0,216 ≤ μ ≤ 0,228	0,223
sf100	E10	Q4.1	0,024	0,011	2,063 ≤ μ ≤ 2,087	2,076
sf100	E10	Q4.2	0,029	0,036	0,805 ≤ μ ≤ 0,835	0,821
sf100	E10	Q4.3	0,008	0,049	0,152 ≤ μ ≤ 0,160	0,156
					Total	8,227

sf	cenario	consulta	desvio	coef_v	confiança 99%	média
sf100	E11	Q1.1	0,028	0,047	0,574 ≤ μ ≤0,602	0,588
sf100	E11	Q1.2	0,001	0,032	0,023 ≤ μ ≤0,023	0,024
sf100	E11	Q1.3	0,004	0,089	0,039 ≤ μ ≤0,043	0,041
sf100	E11	Q2.1	0,027	0,047	0,561 ≤ μ ≤0,588	0,575
sf100	E11	Q2.2	0,033	0,062	0,512 ≤ μ ≤0,546	0,530
sf100	E11	Q2.3	0,020	0,032	0,618 ≤ μ ≤0,639	0,629
sf100	E11	Q3.1	0,024	0,017	1,358 ≤ μ ≤1,383	1,371
sf100	E11	Q3.2	0,009	0,049	0,175 ≤ μ ≤0,184	0,180
sf100	E11	Q3.3	0,028	0,028	0,987 ≤ μ ≤1,015	1,001
sf100	E11	Q3.4	0,006	0,040	0,152 ≤ μ ≤0,158	0,155
sf100	E11	Q4.1	0,023	0,011	2,053 ≤ μ ≤2,077	2,065
sf100	E11	Q4.2	0,029	0,035	0,812 ≤ μ ≤0,841	0,827
sf100	E11	Q4.3	0,010	0,057	0,170 ≤ μ ≤0,180	0,175
					Total	8,161
sf100	E12	Q1.1	0,019	0,031	0,586 ≤ μ ≤0,605	0,596
sf100	E12	Q1.2	0,002	0,042	0,050 ≤ μ ≤0,052	0,052
sf100	E12	Q1.3	0,003	0,041	0,068 ≤ μ ≤0,071	0,070
sf100	E12	Q2.1	0,012	0,014	0,845 ≤ μ ≤0,857	0,851
sf100	E12	Q2.2	0,011	0,013	0,787 ≤ μ ≤0,798	0,793
sf100	E12	Q2.3	0,008	0,010	0,751 ≤ μ ≤0,759	0,755
sf100	E12	Q3.1	0,023	0,016	1,385 ≤ μ ≤1,409	1,397
sf100	E12	Q3.2	0,017	0,071	0,222 ≤ μ ≤0,239	0,231
sf100	E12	Q3.3	0,016	0,014	1,153 ≤ μ ≤1,169	1,161
sf100	E12	Q3.4	0,020	0,078	0,243 ≤ μ ≤0,263	0,254
sf100	E12	Q4.1	0,028	0,010	2,726 ≤ μ ≤2,754	2,741
sf100	E12	Q4.2	0,026	0,019	1,319 ≤ μ ≤1,345	1,332
sf100	E12	Q4.3	0,007	0,043	0,150 ≤ μ ≤0,157	0,154
					Total	10,387
sf100	E13 ao E24	-	-	-	-	-
					Total	-

sf	cenario	consulta	desvio	coef_v	confiança 99%	média
sf100	E25	Q1.1	0,014	0,034	0,401 ≤ μ ≤0,416	0,409
sf100	E25	Q1.2	0,013	0,109	0,112 ≤ μ ≤0,126	0,120
sf100	E25	Q1.3	0,002	0,060	0,026 ≤ μ ≤0,028	0,027
sf100	E25	Q2.1	0,007	0,025	0,275 ≤ μ ≤0,283	0,279
sf100	E25	Q2.2	0,010	0,038	0,254 ≤ μ ≤0,264	0,260
sf100	E25	Q2.3	0,032	0,141	0,208 ≤ μ ≤0,240	0,224
sf100	E25	Q3.1	0,009	0,029	0,309 ≤ μ ≤0,318	0,314
sf100	E25	Q3.2	0,010	0,040	0,244 ≤ μ ≤0,254	0,249
sf100	E25	Q3.3	0,007	0,027	0,277 ≤ μ ≤0,285	0,282
sf100	E25	Q3.4	0,005	0,030	0,178 ≤ μ ≤0,184	0,181
sf100	E25	Q4.1	0,007	0,025	0,272 ≤ μ ≤0,279	0,276
sf100	E25	Q4.2	0,008	0,031	0,253 ≤ μ ≤0,262	0,258
sf100	E25	Q4.3	0,011	0,040	0,277 ≤ μ ≤0,289	0,283
					Total	3,162
sf100	E26	Q1.1	0,020	0,047	0,413 ≤ μ ≤0,433	0,424
sf100	E26	Q1.2	0,015	0,125	0,112 ≤ μ ≤0,128	0,121
sf100	E26	Q1.3	0,002	0,049	0,033 ≤ μ ≤0,034	0,034
sf100	E26	Q2.1	0,010	0,037	0,278 ≤ μ ≤0,288	0,284
sf100	E26	Q2.2	0,016	0,058	0,266 ≤ μ ≤0,282	0,274
sf100	E26	Q2.3	0,028	0,122	0,214 ≤ μ ≤0,243	0,229
sf100	E26	Q3.1	0,011	0,033	0,314 ≤ μ ≤0,325	0,320
sf100	E26	Q3.2	0,012	0,048	0,251 ≤ μ ≤0,264	0,258
sf100	E26	Q3.3	0,006	0,021	0,288 ≤ μ ≤0,294	0,291
sf100	E26	Q3.4	0,006	0,033	0,176 ≤ μ ≤0,182	0,179
sf100	E26	Q4.1	0,005	0,019	0,281 ≤ μ ≤0,287	0,285
sf100	E26	Q4.2	0,007	0,026	0,267 ≤ μ ≤0,274	0,271
sf100	E26	Q4.3	0,009	0,032	0,285 ≤ μ ≤0,295	0,290
					Total	3,260
sf100	E27 ao E28	-	-	-	-	-
					Total	-