Um estudo de caso sobre uso de processamento de dados distribuídos através do citus

Vinícius Medeiros Wanderley

CENTRO DE INFORMÁTICA
UNIVERSIDADE FEDERAL DA PARAÍBA

João Pessoa, 2023

Vinícius Medeiros Wanderley

Um estudo de caso sobre uso de processamento de dados distribuídos através do citus

Monografia apresentada ao curso Ciência da Computação
do Centro de Informática, da Universidade Federal da Paraíba,
como requisito para a obtenção do grau de Bacharel em Ciência da Computação

Orientador: Prof. Dr. Raoni Kulesza
Coorientador: Prof. Dr. Marcelo Iury de Sousa Oliveira

Junho de 2023

Catalogação na publicação
Seção de Catalogação e Classificação

W245e Wanderley, Vinícius Medeiros.
Um estudo de caso sobre uso de processamento de dados distribuídos através do citus / Vinícius Medeiros Wanderley. - João Pessoa, 2023.
75 f. : il.

Orientação: Raoni Kulesza.
Coorientação: Marcelo Iury de Sousa Oliveira.
TCC (Graduação) - UFPB/CI.

1. Desempenho. 2. Notas fiscais eletrônicas. 3. SGBDs. 4. Processamento de dados. I. Kulesza, Raoni. II. Oliveira, Marcelo Iury de Sousa. III. Título.

UFPB/CI CDU 004.451.7:004.7

Elaborado por Michelle de Kássia Fonseca Barbosa - CRB-738

CENTRO DE INFORMÁTICA
UNIVERSIDADE FEDERAL DA PARAÍBA

Trabalho de Conclusão de Curso de Ciência da Computação intitulado ***Um estudo de caso sobre uso de processamento de dados distribuídos através do citus*** de autoria de Vinícius Medeiros Wanderley, aprovada pela banca examinadora constituída pelos seguintes professores:

Prof. Dr. Raoni Kulesza
UFPB

Prof. Dr. Marcelo Iury de Sousa Oliveira
UFPB

Prof. MSc. Hidelberg Oliveira Albuquerque
UFRPE

Coordenador(a) do Departamento Computação Científica
Leandro Carlos de Souza
CI/UFPB

João Pessoa, 22 de junho de 2023

Centro de Informática, Universidade Federal da Paraíba
Rua dos Escoteiros, Mangabeira VII, João Pessoa, Paraíba, Brasil CEP: 58058-600
Fone: +55 (83) 3216 7093 / Fax: +55 (83) 3216 7117

RESUMO

A partir do ano de 2007 foi implementado no Brasil o Sistema Público de Escrituração Digital, em empresas do setor privado e também em instituições públicas do governo, responsáveis pela fiscalização e controle dessas atividades como os fiscos municipais e estaduais e a própria Receita Federal. Nesta conjuntura, apesar da agilidade no monitoramento dos fiscos e economicidade, o grande volume de dados armazenados a partir das notas fiscais eletrônicas, impulsionou a necessidade do desenvolvimento de novas tecnologias para análise e monitoramento de modo eficiente. Desta forma, neste trabalho foi conduzido um estudo para avaliar o desempenho do tempo de execução de consultas no Citus em comparação com o PostgreSQL.

Palavras-chave: Desempenho, Notas fiscais eletrônicas, SGBDs.

ABSTRACT

Since 2007, the Public Digital Bookkeeping System was implemented in Brazil in private sector companies and also in public government institutions, which are responsible for the supervision and control of such activities as city and state tax authorities and the Federal Revenue Service itself. At this context, although the agility in the monitoring of tax authorities and economy, the large volume of data stored from electronic invoices has driven the demand for the development of new technologies for analysis and monitoring in an efficient manner. Thus, the present work conducted a study to evaluate the performance of query execution time in Citus in comparison with PostgreSQL.

Key-words: Performance, Electronic invoices, DBMSs.

LISTA DE FIGURAS

1 Fluxograma operacional da NF-e. 21

2 Diagrama simplificado dos grupos de informações da NF-e. 22

3 Descrição dos ambientes de experimentos. 29

4 Fluxograma para os cenários de execução. 33

5 Esquema de Consultas. 34

6 Gráfico para o tempo de execução da terceira execução, considerando consultas com índices *btree*, agregações (avg + count + sum) e amostra com 850 mil de notas. 36

7 Gráfico para o tempo de execução da terceira execução, considerando consultas com índices *btree*, agregações (avg + count + sum) e amostra com 10 milhões de notas 37

8 Gráfico para o tempo de execução da terceira execução, considerando consultas com índices *btree*, com métrica de registro completo e amostra com 850 mil notas. 38

9 Gráfico para o tempo de execução da terceira execução, considerando consultas com índices *btree*, com métrica de registro completo e amostra com 10 milhões de notas. 39

10 Gráfico para o tempo de execução da terceira execução, considerando consultas com índices *btree*, com métrica de seleção de coluna específica e amostra com 850 mil de notas. 40

11 Gráfico para o tempo de execução da terceira execução, considerando consultas com índices *btree*, com métrica de seleção de coluna específica e amostra com 10 milhões de notas. 41

12 Gráfico para o tempo de execução da terceira execução, considerando consultas com índices *hash*, agregações (avg + count + sum) e amostra com 850 mil de notas. 42

13 Gráfico para o tempo de execução da terceira execução, considerando consultas com índices *hash*, agregações (avg + count + sum) e amostra com 10 milhões de notas 43

14 Gráfico para o tempo de execução da terceira execução, considerando consultas com índices *hash*, com métrica de registro completo e amostra com 10 milhões de notas. 44

15 Gráfico para o tempo de execução da terceira execução, considerando consultas com índices *hash*, com métrica de registro completo e amostra com 850 mil de notas. 45

16 Gráfico para o tempo de execução da terceira execução, considerando consultas com índices *hash*, com métrica de seleção de coluna específica e amostra com 850 mil de notas. 46

17 Gráfico para o tempo de execução da terceira execução, considerando consultas com índices *hash*, com métrica de seleção de coluna específica e amostra com 10 milhões de notas. 47

18 Gráfico para o tempo de execução da terceira execução, considerando consultas sem índices, agregações (avg + count + sum) e amostra com 850 mil de notas. 48

19 Gráfico para o tempo de execução da terceira execução, considerando consultas sem índices, agregações (avg + count + sum) e amostra com 10 milhões de notas. 48

20 Gráfico para o tempo de execução da terceira execução, considerando consultas sem índices, com métrica de registro completo e amostra com 850 mil de notas. 49

21 Gráfico para o tempo de execução da terceira execução, considerando consultas sem índices, com métrica de registro completo e amostra com 10 milhões de notas. 50

22 Gráfico para o tempo de execução da terceira execução, considerando consultas sem índices, com métrica de seleção de coluna específica e amostra com 850 mil de notas. 51

23 Gráfico para o tempo de execução da terceira execução, considerando consultas sem índices, com métrica de seleção de coluna específica e amostra com 10 milhões de notas 52

LISTA DE TABELAS

1 Alocação da base de dados. 29

2 Informações Bases de Dados. 32

3 Consultas para agregações na base *small* utilizando o índice *btree*. 36

4 Consultas para agregações na base *big* utilizando o índice *btree*. 37

5 Consulta para recuperação de registro completo na base *small* utilizando o índice *btree*. 38

6 Consulta para recuperação de coluna específica na base *small* utilizando o índice *btree*. 40

7 Consultas para agregações na base *small* utilizando o índice *hash*. 42

8 Consultas para agregações na base *big* utilizando o índice *hash*. 43

9 Consulta para recuperação de registro completo na base *big* utilizando o índice *hash*. 44

10 Consultas para recuperação de registro completo na base *small* utilizando o índice *hash*. 45

11 Consultas para recuperação de coluna específica na base *small* utilizando o índice *hash*. 46

12 Consulta para recuperação de registro completo na base *small* sem índice. 49

13 Consultas para recuperação de registro completo na base *big* sem índice. . 50

14 Consulta para recuperação de coluna específica na base *small* sem índice. 51

15 Consulta para recuperação de coluna específica na base *big* sem índice. . . 52

16 Estatísticas descritivas para o tempo de execução, em milissegundos (ms), considerando o Citus com 3 *workers* e índice *btree* para amostra *small*. . . 70

17 Estatísticas descritivas para o tempo de execução, em milissegundos (ms), considerando o Citus com 6 *workers* e índice *btree* para amostra *small*. . . 70

18 Estatísticas descritivas para o tempo de execução, em milissegundos (ms), para índice *btree* e amostra *small*, considerando o Postgres com 1PC. . . . 70

19 Estatísticas descritivas para o tempo de execução, em milissegundos (ms), para índice *btree* e amostra *small*, considerando o Postgres com 2PC. . . . 70

20 Estatísticas descritivas para o tempo de execução, em milissegundos (ms), considerando o Citus com 3 *workers* e índice **btree** para amostra *big*. . . . 71

21 Estatísticas descritivas para o tempo de execução, em milissegundos (ms), considerando o Citus com 6 *workers* e índice **btree** para amostra *big*. . . . 71

22 Estatísticas descritivas para o tempo de execução, em milissegundos (ms), para índice *btree* e amostra *big*, considerando o Postgres com 1PC. 71

23 Estatísticas descritivas para o tempo de execução, em milissegundos (ms), para índice *btree* e amostra *big*, considerando o Postgres com 2PC. 71

24 Estatísticas descritivas para o tempo de execução, em milissegundos (ms), considerando o Citus com 3 *workers* e índice **btree** com métrica consulta completa para amostra *small*. 72

25 Estatísticas descritivas para o tempo de execução, em milissegundos (ms), considerando o Citus com 6 *workers* e índice **btree** com métrica consulta completa para amostra *small*. 72

26 Estatísticas descritivas para o tempo de execução, em milissegundos (ms), para índice *btree* para amostra *small*, considerando o Postgres com 1PC e métrica de consulta completa. 72

27 Estatísticas descritivas para o tempo de execução, em milissegundos (ms), para índice *btree* para amostra *small*, considerando o Postgres com 2PC e métrica de consulta completa. 73

28 Estatísticas descritivas para o tempo de execução, em milissegundos (ms), considerando o Citus com 3 *workers* e índice **btree** com métrica consulta completa para amostra *big*. 74

29 Estatísticas descritivas para o tempo de execução, em milissegundos (ms), considerando o Citus com 6 *workers* e índice **btree** com métrica consulta completa para amostra *big*. 74

30 Estatísticas descritivas para o tempo de execução, em milissegundos (ms), para índice *btree* para amostra *big*, considerando o Postgres com 1PC e métrica de regitro completo. 74

31 Estatísticas descritivas para o tempo de execução, em milissegundos (ms), para índice *btree* para amostra *big*, considerando o Postgres com 2PC e métrica de consulta completa. 75

LISTA DE ABREVIATURAS

SPED - Sistema Público de Escrituração Digital

ECD – Escrituração Contábil Digital

FCONT - Controle Fiscal Contábil de Transição

NF-e - Nota Fiscal Eletrônica

CT-e - Conhecimento de Transporte eletrônico

NFS-e - Nota Fiscal de Serviços Eletrônica

Sumário

1 INTRODUÇÃO **14**

1.1 Problema 14

1.2 Justificativa 14

1.3 Objetivos 16

1.3.1 Objetivos específicos 16

2 FUNDAMENTAÇÃO TEÓRICA **18**

2.1 Sistema Público de Escrituração Digital - SPED 18

2.1.1 Escrituração contábil digital - ECD 18

2.1.2 Escrituração contábil fiscal - ECF 19

2.1.3 eSocial 19

2.1.4 Nota Fiscal de Serviços Eletrônicos - NFSe 20

2.1.5 Nota fiscal eletrônica (NF-e) 20

2.1.6 Organização dos Dados da NF-e 21

2.2 Gerenciamento de banco de dados 22

2.3 Citus 24

2.4 Processamento de dados distribuídos 25

2.4.1 Armazenamento de dados em sistemas distribuídos 27

3 METODOLOGIA **28**

3.1 Ambiente dos Experimentos 28

3.2 Modelo de banco de dados 30

3.3 Implantação do modelo banco de dados 30

3.3.1 PostgreSQL 30

3.3.2 Citus 31

3.4 Base de Dados 32

3.5 Descrição dos Cenários 32

3.6 Métrica de Avaliação 33

4 RESULTADOS E DISCUSSÃO **35**

4.1 Buscas com índices *btree* 35

4.2 Buscas com índices *hash* 41

4.3 Buscas sem índices 47

5 CONSIDERAÇÕES FINAIS **53**

REFERÊNCIAS **54**

APÊNDICES 1 - Recorte das propriedades da fatonfe **56**

APÊNDICES 3 - Adaptação do Docker para o PostgreSQL **59**

APÊNDICES 3 - Adaptação do Docker para o Citus **60**

APÊNDICES 4 - Código para realização das Consultas. Referente a amostra pequena e grande, respectivamente **63**

APÊNDICES 5 - Estatísticas descritiva com índice *btree* com *queries* de agregação e tamanho amostral de 850 mil notas **70**

APÊNDICES 6 - Estatísticas descritiva com índice *btree* com *queries* de agregação e tamanho amostral de 10 milhões notas **71**

APÊNDICES 7 - Estatísticas descritiva com índice *btree* com *queries* de consulta completa e tamanho amostral de 850 mil notas **72**

APÊNDICES 8 - Estatísticas descritiva com índice *btree* com grupos *queries* de consulta completa e tamanho amostral de 10 milhões notas **74**

1 INTRODUÇÃO

No atual cenário da globalização o interesse em armazenar informações, desde a gestão de conhecimento ao fluxo de produção, tem crescido em todos os segmentos da sociedade, sobretudo em grandes organizações. Com isso, dada a demanda e em paralelo o grande volume de dados, cresceu a necessidade de otimização de Sistemas de Gerenciamento de Bancos de Dados (SGBD).

Os SGBDs objetivam o gerenciamento de grandes volumes de dados, em que envolve a definição de estruturas para o armazenamento das informações, oferecendo técnicas de manipulação destas informações. Destarte, visto o grande volume de dados, é necessário a utilização de sistemas eficientes e robustos para realização de consultas ao banco de dados, pois a execução de constantes consultas e alterações nos dados armazenados estão associadas ao desempenho dos SGBDs.

Desta forma, é relevante avaliar o processo de robustez dos SGBDs. Assim, este trabalho, a partir do método experimental objetiva investigar o uso do SGBD Citus para processamento e recuperação de dados de um volume considerável de NF-es. Em particular, procurou-se avaliar o desempenho e a performance do Citus para o processamento da base de dados em comparação ao PostgreSQL. À vista disso, ficou estabelecido o seguinte problema de pesquisa.

1.1 Problema

Diante da discussão precedente, e considerando os paradigmas ao realizar o processamento de dados distribuídos, busca-se responder:

Qual o SGBD com melhor performance de desempenho para consultas executadas, o Citus ou o PostgreSQL?

1.2 Justificativa

Com o avanço das tecnologias e com a quantidade de informações que são geradas diariamente, os Sistemas de Gerenciamento de Bancos de Dados (SGBDs) tornaram-se componentes essenciais no cotidiano da sociedade moderna. Os SGBDs são definidos na literatura como um conjunto de programas que objetivam o gerenciamento estrutural de um banco de dados e controlam o acesso aos dados armazenados. Desta forma, como destaca Santos et al. (2021), dadas suas características, os bancos de dados apresentam um conjunto de conceitos que podem ser usados para descrição de sua estrutura, dando origem a diversos modelos de banco de dados. Particularmente, para este trabalho serão considerados os modelos de dados relacional e orientados a documentos.

Os modelos relacionais apresentam como principal estrutura a relação entre os itens, ou seja, tabelas. Dentre alguns dos SGBDs que utilizam esse tipo de modelo estão o PostgreSQL, MySQL e o Microsoft SQL Server. Os modelos relacionais, desde sua criação, apresentam enorme relevância para o armazenamento de dados transacionais. Apesar disso, quando há necessidade de processar e armazenar um grande volume de dados, não necessariamente estruturados, estes tipos de SGBDs apresentam limitações (MENDONÇA, 2022).

Por outro lado, os modelos orientados a documentos armazenam dados no formato de coleções de documentos que podem ser em formatos semiestruturados como XML ou JSON, e outros. Desta forma, os documentos possuem auto descrição, além de uma estrutura de dados hierárquica em forma de árvore, que pode ser composta por mapas, valores escalares e coleções (SANTOS et al., 2021). Dentre alguns dos SGBDs que utilizam esse tipo de modelo estão o MongoDB, RedisJSON e o DynamoDB.

O PostgreSQL, criado em 1986 na Universidade da Califórnia em Berkeley, é um SGBD relacional com diversas extensões disponíveis online que integram dados específicos e suportam uma diversidade de propriedades, sendo utilizada como dimensionamento uma extensão de código aberto chamada Citus. A extensão Citus distribui dados e consultas em vários nós em um *cluster*, transformando o PostgreSQL em um SGBD distribuído, o que aprimora recursos como fragmentação, replicação, mecanismo SQL distribuído, tabelas de referência e distribuídas. Para mais, um *cluster* Citus de alto desempenho envolve considerar o modelo de dados, as ferramentas e a escolha dos recursos SQL a serem usados (GKAMAS et al., 2022).

De mais a mais, um conjunto massivo de dados, que nos últimos anos tem chamado atenção de governos, bem como de parte da comunidade científica, é o chamado Sistema Público de Escrituração Digital (SPED). Este é um sistema composto por uma série de subprojetos, quais são: SPED Contábil (ECD – Escrituração Contábil Digital), FCONT (Controle Fiscal Contábil de Transição), SPED Fiscal, Escrituração Fiscal Digital das Contribuições incidentes sobre a Receita (EFD-Contribuições), Nota Fiscal Eletrônica (NF-e), Conhecimento de Transporte eletrônico (CT-e) e Nota Fiscal de Serviços Eletrônica (NFS-e).

O SPED gera um grande volume de dados, que cresce de forma muito rápida, ocasionando o acúmulo de informações nos mais variados bancos de dados. De acordo com o Portal da Nota Fiscal Eletrônica (BRASIL, 2022), desde a sua implantação no ano de 2006, a quantidade de NF-e autorizada em 13 de dezembro de 2022 computava em $35,01$ bilhões, não consideradas NF-es canceladas ou degeneradas, enquanto que, ainda segundo o portal, a quantidade de emissores da NF-e era de $2,181$ milhões. Portanto, dado o volume de dados, surge a necessidade do desenvolvimento de modelos de tecnologias capazes de realizar o processamento de forma ágil.

Não obstante, além de auxiliar os órgãos de controle na fiscalização, as NF-es representam um desafio no contexto do gerenciamento de dados. Ainda que possua uma estrutura para emissão de notas fiscais unificada, o compartilhamento das notas fiscais entre a SEFAZ e os órgãos de controle ainda carece de SGBDs com uma arquitetura eficiente.

Para além disso, o volume e a variedade de formatos das NF-es portam muitas dificuldades para os SGBDs relacionais. Os esquemas de BD relacionais incluem a criação de múltiplas tabelas, com variadas informações, tornando o gerenciamento complexo. Em consideração a isso, na literatura pesquisada identificamos o trabalho de Santos et al. (2021), que propôs comparar modelos de dados distintos em SGBDs heterogêneos. Em seu trabalho, o autor implementou dois modelos de dados, um relacional, e o orientado à documento, em que foram analisados o desempenho e a performance de cada um deles, além do processamento da base de dados em cada modelo. Para tanto, em seu trabalho, o autor avaliou a performance comparando os SGBDs: MongoDB e PostgreSQL. Dessarte, neste trabalho avaliamos o desempenho em termos de tempo de execução entre armazenamentos de dados distribuídos escaláveis, utilizando o Citus e comparando com PostgreSQL em uma base de dados específica.

1.3 Objetivos

O objetivo geral do trabalho consiste em analisar o desempenho para recuperação e processamento de consultas sobre bases de NF-es, utilizando o Citus em comparação com o PostgreSQL.

1.3.1 Objetivos específicos

Para alcançarmos o objetivo geral são propostos os seguintes objetivos específicos:

1. Propor um modelo de banco de dados adequado;
2. Implementar o modelo de banco de dados através dos seguintes SGBDs: Citus e PostgreSQL;
3. Definir um conjunto de cenários e consultas que serão utilizados ao longo do estudo;
4. Conduzir um experimento de *análise de desempenho* com ênfase na funcionalidade de *sharding* (fragmentação) dos dados.

Dito isto, o trabalho está organizado em cinco capítulos, incluindo este de introdução. O Capítulo 2 foi destinado para apresentar a fundamentação teórica, destacando alguns

dos principais conceitos, suas características e as NF-e, que nortearão o desenvolvimento do trabalho. No Capítulo 3 são apresentados os aspectos metodológicos utilizados na construção da proposta. No Capítulo 4 são apresentados os resultados e discussões e o Capítulo é destinado para considerações finais. Ainda, é organizado um Apêndice para disponibilizar um recorte dos códigos utilizados ao longo do trabalho.

2 FUNDAMENTAÇÃO TEÓRICA

Nesta seção será apresentado o suporte teórico que sustentará o desenvolvimento deste trabalho.

2.1 Sistema Público de Escrituração Digital - SPED

Nos últimos anos, devido ao avanço tecnológico e a necessidade de melhorias nas administrações tributárias, foram implementadas uma série de mudanças. Dentre estas mudanças, destacamos a implantação do Sistemas Público de Escrituração Digital - SPED, instituído pelo Decreto nº 6.022, de 22 de janeiro de 2007, disponível em Brasil (2007). O SPED tem como principal objetivo integrar os fiscos por uma padronização e compartilhamento de informações contábeis e fiscais. Desta forma, Brasil (2007) define o SPED como "instrumento que unifica as atividades de recepção, validação, armazenamento e autenticação de livros e documentos que integram a escrituração comercial e fiscal dos empresários e das sociedades empresárias, mediante fluxo único, computadorizado, de informações".

Ademais, de acordo com Santos et al. (2021), o SPED apresenta como objetivos específicos:

- uniformizar as obrigações acessórias para os contribuintes, com o estabelecimento de transmissão única de distintas obrigações acessórias de diferentes órgãos fiscalizadores;
- tornar mais célere a identificação de ilícitos tributários, com a melhoria do controle dos processos, a rapidez no acesso às informações e a fiscalização mais efetiva das operações com o cruzamento de dados e auditoria eletrônica.

Diante disso, dentre os diversos benefícios do SPED para os contribuintes, de fato consiste na uniformização de informações prestadas aos diversos entes governamentais, na redução do envolvimento involuntário em práticas fraudulentas e na diminuição da burocracia improdutiva (MANOEL et al., 2011). Ainda, vale destacar que com a implantação do SPED tem-se a economia com papel e preservação do meio ambiente, a diminuição de tempo despendido para o registro, aceleração no processo de fiscalização e fortalecimento do controle, melhoria na qualidade da informação e disponibilidade de cópias autênticas para usos distintos e concomitantes.

2.1.1 Escrituração contábil digital - ECD

A escrituração contábil foi definida como sendo todos os registros contábeis de uma empresa em um determinado período, assim, as informações sejam úteis ao fisco, aos diretores e para uma melhor tomada de decisões.

A ECD tem por objetivo a substituição da escrituração manuscrita pela escrituração transmitida via arquivo, ou seja, corresponde à obrigação de transferir, em versão digital, os livros contábeis (MANOEL et al., 2011). Neste contexto, a ECD propõe a exemplificar e agilizar o acesso às informações das empresas em uma eventual fiscalização, sendo composta pelo Termo de abertura e encerramento, o Livro diário, o Livro razão, o Balanço patrimonial, a Demonstração do resultado do exercício e, as demais demonstrações auxiliares obrigatórias as empresas (MANOEL et al., 2011). Com a implementação da ECD, a empresa não necessita realizar a impressão de diários, balanços e outros demonstrativos, visto que este gera um arquivo digital padronizado, que é assinado através do certificado digital do contador e dos representantes legais da empresa perante a Junta Comercial.

2.1.2 Escrituração contábil fiscal - ECF

De acordo com Tsukamoto et al. (2019), a ECF teve o seu início no exercício de 2015, tendo sido instituída pela Instrução Normativa nº 1.422 de 2013. Tem como principal objetivo interligar os dados contábeis e fiscais relacionados a apuração do Imposto de Renda e da Contribuição Social, consistindo basicamente no transporte de todos os valores apurados nestes dois tipos de tributos específicos. Ademais, a ECF necessita ser validada através de assinatura digital do representante legal da empresa. O autor ainda destaca que os seguintes livros são contemplados nesta escrituração, o Registro de Entradas, o Registro de Saídas, o Registro de Inventário, o Registro de Apuração do Imposto sobre Produtos Industrializados - IPI e, o Registro de Apuração do ICMS (SANTOS et al., 2021).

2.1.3 eSocial

O Sistema de Escrituração Digital das Obrigações Fiscais, Previdenciárias e Trabalhistas eSocial foi instituído pelo Decreto n^0 8373/2014. O objetivo do sistema é permitir a comunicação direta dos empregadores com o Governo, de forma unificada, as informações relativas aos trabalhadores, como vínculos, contribuições previdenciárias, folha de pagamento, comunicações de acidente de trabalho, aviso prévio, escriturações fiscais e informações sobre o FGTS.

De acordo com Portal (2022), "*a implantação do eSocial viabilizará garantia aos diretos previdenciários e trabalhistas, racionalizará e simplificará o cumprimento de obrigações, eliminará a redundância nas informações prestadas pelas pessoas físicas e jurídicas, e aprimorará a qualidade das informações das relações de trabalho, previdenciárias e tributárias. A legislação prevê ainda tratamento diferenciado às micro e pequenas empresas*".

2.1.4 Nota Fiscal de Serviços Eletrônicos - NFSe

A Nota Fiscal de Serviços Eletrônica - NFSe, segundo Portal (2022), "*é um documento de existência digital, gerado e armazenado eletronicamente em Ambiente Nacional pela RFB, pela prefeitura ou por outra entidade conveniada, para documentar as operações de prestação de serviços*".

Desta forma, de acordo com o SPED, o projeto da NFSe tem como objetivo beneficiar as administrações tributárias, padronizando e melhorando a qualidade das informações, racionalizando os custos e gerando maior eficácia, bem como o aumento da competitividade das empresas brasileiras pela racionalização das obrigações acessórias (redução do custo-Brasil), em especial a dispensa da emissão e guarda de documentos em papel. O leitor interessado em maiores detalhes poderá consultar o portal do SPED, disponível em Portal (2022).

2.1.5 Nota fiscal eletrônica (NF-e)

A Nota fiscal eletrônica (NF-e), tal qual sua conceituação disponível no portal do SPED (PORTAL, 2022), é um projeto que "foi desenvolvido, de forma integrada, pelas Secretarias de Fazenda dos Estados e Receita Federal do Brasil, a partir da assinatura do Protocolo ENAT 03/2005, de 27/08/2005, que atribuiu ao Encontro Nacional de Coordenadores e Administradores Tributários Estaduais (ENCAT) a coordenação e a responsabilidade pelo desenvolvimento e implantação do Projeto NFe". Desta forma, para Portal (2022), o projeto das NF-es agrega os seguintes benefícios e vantagens às partes envolvidas:

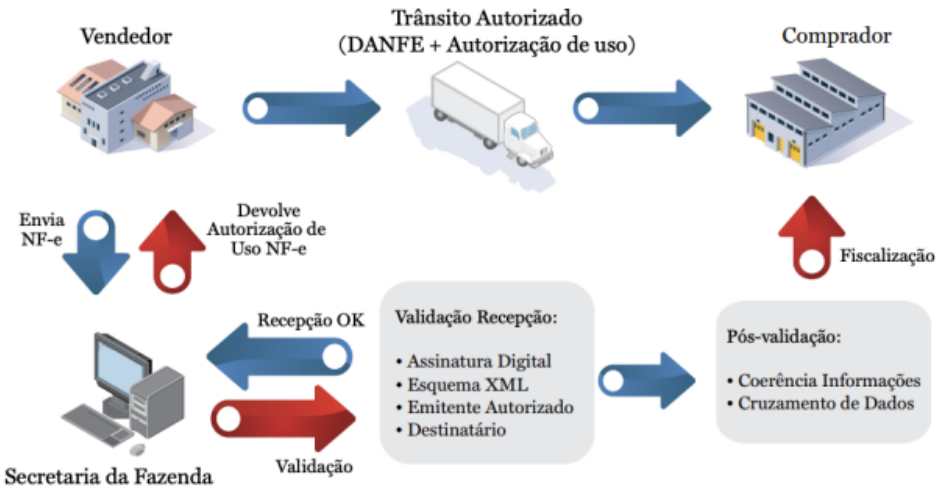
1. Aumento na confiabilidade da Nota Fiscal;
2. Melhoria no processo de controle fiscal, possibilitando um melhor intercâmbio e compartilhamento de informações entre os fiscos;
3. Redução de custos no processo de controle das notas fiscais capturadas pela fiscalização de mercadorias em trânsito;
4. Diminuição da sonegação e aumento da arrecadação;
5. Suporte aos projetos de escrituração eletrônica contábil e fiscal da Receita Federal e demais Secretarias de Fazendas Estaduais;
6. Fortalecimento da integração entre os fiscos, facilitando a fiscalização realizada pelas Administrações Tributárias devido ao compartilhamento das informações das NF-e;
7. Rapidez no acesso às informações;

8. Eliminação do papel;
9. Aumento da produtividade da auditoria através da eliminação dos passos para coleta dos arquivos;
10. Possibilidade do cruzamento eletrônico de informações.

O conceito de nota fiscal passou por constantes evoluções desde a década de 70 (SANTOS et al., 2021). Anteriormente as notas fiscais eram emitidas de forma manual ou através da máquina mecanográfica de escrever, evoluindo para os dias atuais, no qual sua emissão é realizada de forma eletrônica por meio de impressora, graças aos avanços tecnológicos (SANTOS et al., 2021). Portanto, a NF-e é um documento gerado e emitido de modo eletrônico, cujo objetivo é documentar as operações e prestações cujas validade jurídica é garantida pela assinatura digital do emitente juntamente com a autorização de uso pela administração tributária da unidade federada.

Desta feita, no trabalho Santos et al. (2021) é apresentado um fluxograma operacional quando da emissão da NF-e, ver Figura 1. A Figura 1 detalha o caminho seguido após a emissão da NF-e, quais são a geração de dados, o armazenamento, a determinação dos agentes e responsabilidades envolvidas, assim como dos aspectos relacionados à integridade e a contingência do processo.

Figura 1: Fluxograma operacional da NF-e.



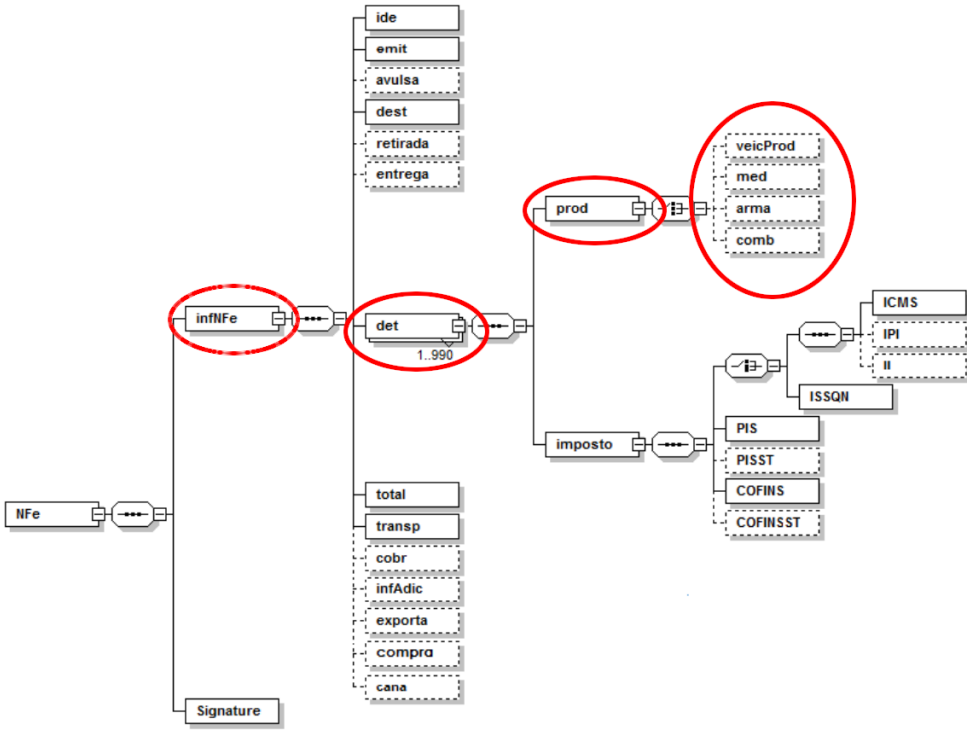
Fonte: Adaptada de Santos et al. (2021).

2.1.6 Organização dos Dados da NF-e

A Nota Fiscal Eletrônica NF-e é estruturada no formato *Extensible Markup Language* (XML), com hierarquia de campos, sendo importante seu entendimento para uma melhor

compreensão no processo de manipulação dos dados. Diante disso, no trabalho de Santos (2015), é apresentado um esquema simplificado das informações presentes na NF-e, como na Figura 2.

Figura 2: Diagrama simplificado dos grupos de informações da NF-e.



Fonte: Adaptada de Santos (2015).

O *Schema* XML é uma linguagem de marcação que define o conteúdo e organização do documento eletrônico. Como visto na Figura 2, o grupo de informações da NF-e se vincula em uma hierarquia com o grupo de detalhamento dos tipos de produtos e impostos da NF-e (det), os quais estão atrelados ao grupo de informações (prod) que são vinculados as informações sobre produtos específicos na NF-e.

Na versão 5.0 do manual de orientação do contribuinte Sefaz (2012, p.148), a partir da coluna campo, é detalhada os nomes padronizados dos mais de 400 campos possíveis de uma NF-e, como por exemplo, a IE, correspondente a inscrição estadual do contribuinte junto às secretarias de fazenda, sendo os grupos utilizados pelo *Schema* XML para diferenciar os campos Santos (2015).

2.2 Gerenciamento de banco de dados

Os bancos de dados e os Sistemas de Gerenciamento de Banco de Dados (SGBDs) se tornaram componentes essenciais no cotidiano da sociedade moderna. Assim, de acordo

com Ramakrishnan e Gehrke (2008), os SGBDs são uma ferramenta indispensável para gerenciar informações. No tocando ao conceito de banco de dados, segundo o autor ora citado, "*é uma coleção de dados que, tipicamente, descreve as atividades de uma ou mais organizações relacionadas*". Por outro lado, um SGBD é um software projetado para auxiliar a manutenção e utilização de vastos conjuntos de dados. Uma alternativa para não se usar um SGBD é armazenar os dados em arquivos e escrever código específico do aplicativo para gerenciá-los.

Similarmente, Santos et al. (2021) define o SGBD como uma coleção de programas que tem como função o gerenciamento da estrutura de um banco de dados e controle ao acesso dos dados armazenados. Para o autor, o SGBD serve como intermediário entre o usuário e o banco de dados. Ainda, o autor destaca que "sua estrutura é armazenada como um conjunto de arquivos e a única forma de se ter acesso a esses arquivos é por intermédio do SGBD".

Com isso, em uma perspectiva histórica, Ramakrishnan e Gehrke (2008), traz

> O primeiro SGBD de propósito geral, projetado por Charles Bachman, na General Electric, no início da década de 1960, foi chamado Depósito de Dados Integrados (Integrated Data Store). Ele constituiu a base do modelo de dados de rede, que foi padronizado pela Conference on Data Systems Languages (CODASYL) e influenciou bastante os sistemas de banco de dados na década de 1960. Bachman foi o primeiro a ser contemplado pelo Prêmio Turing da ACM (o equivalente ao Prêmio Nobel de Ciência da Computação) pelo trabalho na área de banco de dados; ele recebeu o prêmio em 1973.
>
> No final da década de 1960, a IBM desenvolveu o SGBD Sistema de Gerenciamento de Informação (IMS — Information Management System), ainda usado atualmente em diversas instalações. O IMS constituiu a base da estrutura de representação alternativa de dados, chamada modelo de dados hierárquicos. O sistema SABRE para reservas de passagens aéreas foi desenvolvido em conjunto pela American Airlines e pela IBM nessa mesma época, e permitiu que diversas pessoas acessassem os mesmos dados através de uma rede de computadores. Interessante observar que, atualmente, o mesmo sistema SABRE é utilizado para fornecer serviços populares de viagens baseados na Web, tais como o Travelocity (RAMAKRISHNAN; GEHRKE, 2008).

Assim, notadamente não é recente o emprego dos SGBDs, portanto, estudos indicam que sua utilização trazem inúmeras melhorias e/ou facilidades para gerenciamento de dados, dentre as quais Ramakrishnan e Gehrke (2008), destaca:

1. **Independência de Dados:** Os programas aplicativos não devem, idealmente, ser expostos aos detalhes de representação e armazenamento de dados. O SGBD provê uma visão abstrata dos dados que oculta tais detalhes.

2. **Acesso Eficiente aos Dados:** Um SGBD utiliza uma variedade de técnicas sofisticadas para armazenar e recuperar dados eficientemente. Este recurso é especialmente importante se os dados são armazenados em dispositivos de armazenamento externos.

3. **Integridade e Segurança dos Dados:** Se os dados são sempre acessados através do SGBD, ele pode forçar restrições de integridade. Por exemplo, antes de inserir informações sobre o salário de um funcionário, o SGBD pode verificar se o orçamento do departamento não está se excedendo. Além disso, ele pode forçar controles de acesso que governam quais dados estão visíveis a diferentes classes de usuários.

4. **Administração de Dados:** Quando diversos usuários compartilham dados, centralizar a administração dos dados pode oferecer melhorias significativas. Profissionais experientes que compreendem a natureza dos dados sendo gerenciados, e como os diferentes grupos de usuários os utilizam, podem ser responsáveis por organizar a representação dos dados para minimizar a redundância e para realizar as sintonizações finas do armazenamento dos dados para garantir uma eficiente recuperação.

5. **Acesso Concorrente e Recuperação de Falha:** Um SGBD planeja o acesso concorrente aos dados de maneira tal que os usuários podem achar que os dados estão sendo acessados por apenas um único usuário de cada vez. Além disso, o SGBD protege os usuários dos efeitos de falhas de sistema.

6. **Tempo Reduzido de Desenvolvimento de Aplicativo:** O SGBD suporta funções importantes que são comuns a vários aplicativos que acessam os dados no SGBD. Isso, em conjunto com uma interface de alto nível aos dados, facilita o desenvolvimento rápido de aplicativos. Os aplicativos de SGBD tendem a ser mais robustos do que os aplicativos similares independentes porque muitas tarefas importantes são tratadas pelo SGBD (e não precisam ser depuradas e testadas no aplicativo).

A importância de um SGBD entre as aplicações do usuário final e o banco de dados pode ser notada quando se analisa as melhorias que um SGBD oferece, como o fato dele permitir que os dados no banco sejam compartilhados por diversas aplicações e usuários (SANTOS et al., 2021). Outro benefício desse sistema se dá pelo fato dele ter a capacidade de integrar visualizações muito diferentes dos usuários sobre os dados em um único repositório, que engloba tudo. O autor ainda destaca que os dados constituem a matéria-prima, a partir da qual as informações são obtidas, é necessário um bom método para gerenciá-lo.

2.3 Citus

Citus é uma extensão do PostgreSQL distribuída para dados e consultas em um *cluster* de várias máquinas (DATA, 2022a). Com isso, o Citus disponibiliza novos recursos aos usuários, de modo que sejam mantida as ferramentas do PostgreSQL e, respaldando que (DATA, 2022a)

> O Citus escala o PostgreSQL horizontalmente em várias máquinas usando fragmentação e replicação. Seu mecanismo de consulta paraleliza as consultas SQL recebidas nesses servidores para permitir respostas humanas em tempo real (menos de um segundo) em grandes conjuntos de dados (DATA, 2022a).

O PostgreSQL, segundo a literatura, é um dos sistemas de gerenciamento de banco de dados de código aberto mais populares. Desta forma, o PostgreSQL é um sistema versátil e usado em diversas áreas como na indústria, na física de partículas e bancos de dados geoespaciais (CUBUKCU et al., 2021). O autor ainda destaca que dada sua extensibilidade, permite aos desenvolvedores adicionar novas funcionalidades de banco de dados sem se separar do projeto original. Como isso, para o autor muitas empresas aproveitaram a rica funcionalidade e o ecossistema do PostgreSQL para criar aplicativos avançados e bem-sucedidos, concluindo que foi criado uma demanda significativa para o PostgreSQL escalar além de um único servidor. Dito isto, ressalta-se que

> "o Citus é o primeiro banco de dados distribuído que oferece sua funcionalidade por meio das APIs de extensão do PostgreSQL. As APIs de extensão fornecem controle suficiente sobre o comportamento do PostgreSQL para integrar uma camada de fragmentação, um planejador e executor de consulta distribuído e transações distribuídas de forma transparente para o aplicativo. Ser uma extensão permite que o Citus mantenha a compatibilidade com os recursos e ferramentas mais recentes do PostgreSQL a um custo de engenharia insignificante. Além disso, o Citus distribui dados através de servidores PostgreSQL e envia consultas através do protocolo do PostgreSQL. Isso significa que o Citus pode utilizar todo o acesso a dados e recursos de armazenamento oferecidos pelos servidores PostgreSQL subjacentes, incluindo recursos avançados como JSONB, junções laterais, GiST índices, tipos de array e outras extensões"(CUBUKCU et al., 2021).

Ademais, tal qual é destaque no trabalho de Gkamas et al. (2022), a ferramenta Citus distribui dados e consultas em vários nós em um *cluster*, assim, o PostgreSQL é transformado em um SGBD distribuído, aprimorado com recursos como *sharding*, replicação, um mecanismo SQL distribuído, tabelas de referência e distribuídas. Assim, um *cluster* Citus de alto desempenho envolve considerar o modelo de dados, suas ferramentas e escolha dos recursos SQL a serem utilizadas. Além disso, o Citus não é suportado nativamente pelo PostgreSQL e requer a instalação de bibliotecas adicionais. Destarte, outras extensões úteis são PostgreSQL pg-stat-statements, que rastreia estatísticas sobre as consultas executadas por um banco de dados PostgreSQL, e PostGIS, que estende o PostgreSQL para lidar com dados espaciais e tipos de dados, também suportando objetos separados geograficamente.

2.4 Processamento de dados distribuídos

É fácil encontrar na literatura argumentos que definem um sistema de banco de dados distribuído (BDD) como uma relação de nós, cada qual podendo participar na

execução de transações que acessam dados em um ou mais nós. Desta forma, para Ramakrishnan e Gehrke (2008), em um sistema de banco de dados distribuído, os dados são armazenados em vários sites e, normalmente, cada site é gerenciado por um SGBD executado independentemente dos outros sites. Para o autor, "visão clássica de um sistema de banco de dados distribuído é que o sistema deve tornar o impacto da distribuição dos dados transparente".

Com isso, segundo o autor por hora citado, se os dados são distribuídos, mas todos os servidores executam o mesmo software de SGBD, temos um sistema de banco de dados distribuído homogêneo. Se diferentes sites são executados sob o controle de diferentes SGBDs, de forma basicamente autônoma, e são conectados que de algum modo permita o acesso aos dados a partir de vários sites, temos um sistema de banco de dados distribuído heterogêneo, também referido como sistema de múltiplos bancos de dados (multidatabase).

Desta feita, Ramakrishnan e Gehrke (2008), define três estratégias, como alternativas para separar funcionalidade entre diferentes processos relacionados ao SGBD, quais são:

1. **Cliente-servidor**: "O sistema cliente-servidor tem um ou mais processos clientes e um ou mais processos servidores, e um processo cliente pode enviar uma consulta para qualquer processo servidor. Os clientes são responsáveis por questões da interface com o usuário e os servidores gerenciam dados e executam transações. Assim, um processo cliente poderia ser executado em um computador pessoal e enviar consultas para um servidor sendo executado em um computador de grande porte";

2. **Servidor colaborador**: "A arquitetura cliente-servidor não permite que uma única consulta abranja vários servidores, pois o processo cliente teria de ser capaz de subdividir tal consulta nas subconsultas apropriadas, para serem executadas em diferentes sites e, depois, reunir as respostas das subconsultas. Portanto, o processo cliente seria muito complexo e seus recursos começariam a se sobrepor ao servidor; torna-se mais difícil distinguir entre clientes e servidores. A eliminação dessa distinção nos leva a uma alternativa para a arquitetura cliente-servidor: um sistema de servidor colaborador";

3. **Middleware**: "A arquitetura middleware é projetada para permitir que uma única consulta abranja vários servidores, sem exigir que todos os servidores de banco de dados sejam capazes de gerenciar tais estratégias de execução em vários sites. Ela é particularmente atraente ao se tentar integrar vários sistemas legados, cujos recursos básicos não podem ser estendidos".

2.4.1 Armazenamento de dados em sistemas distribuídos

Em um SGBD distribuído, as relações são armazenadas em vários sites. Assim, o acesso a uma relação armazenada em um site remoto acarreta custos de passagem de mensagem e, para reduzir essa sobrecarga, uma única relação pode ser particionada ou fragmentada entre vários sites, com os fragmentos armazenados nos sites onde são mais frequentemente acessados ou replicados em cada site onde a relação tem alta demanda (RAMAKRISHNAN; GEHRKE, 2008).

A fragmentação (sharding) consiste em subdividir uma relação em relações menores ou fragmentos e armazenar os fragmentos (em vez da relação em si), possivelmente em diferentes sites. Ademais, os fragmentos podem ser horizontais ou verticais (RAMAKRISHNAN; GEHRKE, 2008). Na fragmentação horizontal, cada fragmento consiste em um subconjunto de linhas da relação original, enquanto que na fragmentação vertical, cada fragmento consiste em um subconjunto de colunas da relação original. Diante disso, é possível realizar uma replicação, que significa armazenar várias cópias de uma relação ou fragmento de relação, sendo que uma relação inteira pode ser replicada em um ou mais sites. Analogamente, um ou mais fragmentos de uma relação podem ser replicados em outros sites.

3 METODOLOGIA

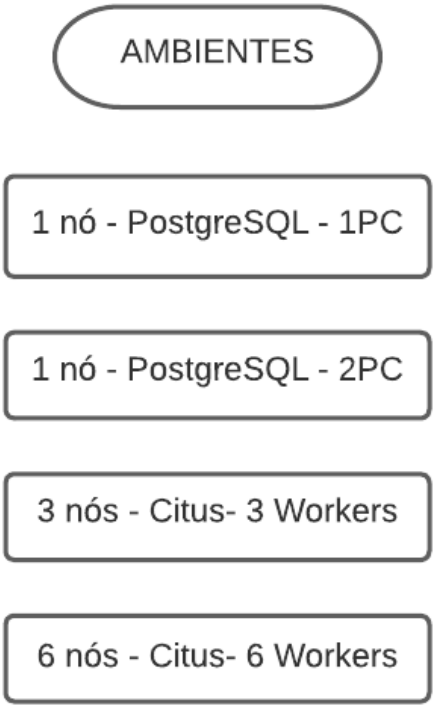
Neste trabalho, será realizada uma análise da performance do tempo de execução em consultas a um banco de dados contendo NF-es, utilizando o PostgreSQL e o Citus. Desta forma, pretende-se identificar em quais situações um SGBD se apresenta melhor que o outro, com relação a performance. Portanto, este Capítulo apresentará a metodologia utilizada para o desenvolvimento deste estudo, descrevendo os métodos e as técnicas empregadas. Na subseção que segue é apresentada a configuração do ambiente de experimentos.

3.1 Ambiente dos Experimentos

A proposta foi conduzida realizando uma *análise de desempenho* entre um *cluster* Citus com $3n$ nós, particularmente, consideramos $n = 1$ e 2, e dois tipos de configurações de máquinas rodando o PostgreSQL.

Para o caso do PostgreSQL, serão consideradas duas máquinas virtuais. Na primeira, serão atribuídos 2GB de RAM e 2 núcleos de CPU, que aqui chamaremos poder computacional (PC), enquanto que na segunda serão atribuídos 4GB de RAM e 4 núcleos de CPU, sendo nomeadas de 1PC e 2PC, respectivamente. Na Figura 3, são apresentadas as descrições dos ambientes dos experimentos. Mais detalhes sobre a configuração das máquinas virtuais são apresentadas na Tabela 3

Figura 3: Descrição dos ambientes de experimentos.



Fonte: Elaborada pelo Autor (2022).

Para realização dos experimentos, dado o ambiente de execução, foi definido o banco de dados bem como os pré-requisitos da máquina de hospedagem. Sendo considerada uma amostra anonimizada da base de dados NF-e de teste pertencente à Secretaria de Estado da Fazenda da Paraíba - SEFAZ/PB. A amostra não pode ser disponibilizada por questão de sigilo.

Tabela 1: Alocação da base de dados.

Número de máquinas	Sistema operacional	PC	CPU	Memória RAM
7	Debian GNU/Linux 11 (bullseye)	1	2 núcleos de um Xeon	
	x86 64 kernel 5.10.0-14-amd64		Gold 6330 a 1.995 GHz	2GB
1	Debian GNU/Linux 11 (bullseye)	2	4 núcleos de um Xeon	
	x86 64 kernel 5.10.0-14-amd64		Gold 6330 à 1.995 GHz	4GB

Nota: PC = Poder computacional.

Por conseguinte, as instâncias de SGBDs, foram executadas por meio de containers Docker com base na imagem oficial do *PostgreSQL* na versão 15.1. Para o caso dos experimentos com Citus, foi utilizada a versão 14.5 da base do postgres.

3.2 Modelo de banco de dados

Considerando o tipo comumente utilizado como documento fiscal na base de dados da SEFAZ/PB, uma tabela chamada *fatonfe* será criada para armazenar as NF-es. A estrutura da tabela, assim como a nomenclatura dos campos, foi elaborada a partir do leiaute[1] público da NF-e. O leiaute prevê um conjunto extenso de elementos XML, os quais podem ou não estar presentes em cada nota a depender das regras de preenchimento, da natureza da operação, do tipo de tributação incidente sobre o produto comercializado, entre outras dezenas de regras consolidadas no Manual de Orientação ao Contribuinte (BRASIL, 2022). Um recorte das propriedades da *fatonfe* é apresentado no Apêndice 1.

3.3 Implantação do modelo banco de dados

O objetivo desta Seção é ilustrar como é realizada a implantação dos SGDBs. Portanto, para o leitor interessado, é possível reproduzir os passos necessários para obter os resultados que serão descritos nas próximas Seções.

Para realizar a implantação, e como objetivamos transformar um pool de hosts Docker em um único host virtual, foi utilizado o gerenciador de container docker swarm e arquivos docker-compose. Isso se deu, além da simplicidade para realizar implementações com Swarm, também pela facilidade de gerenciar os SGDBs utilizando containers. Ademais, foram utilizados os 2 SGDBs distintos, o PostgreSQL e o Citus.

3.3.1 PostgreSQL

Considerando a exemplificação na documentação do dockerhub do próprio PostgreSQL, (HUB, 2022), foi utilizado para a implantação do PostgreSQL nas máquinas virtuais de teste da SEFAZ/PB o arquivo docker-compose disponível no Apêndice 2. Destaca-se que nesta etapa é utilizado um servidor único para os testes, portanto, não ocasionando nenhum impacto no ambiente real de produção das NF-es. No arquivo docker-compose está disponível o manual para realização da implantação de 1 nó contendo apenas o gerente (*manager*). Sua execução é feita pelo comando:

```
docker stack deploy -c <Path to a Compose file>
```

Além disso, também, a tabela fatonfe é criada no PostgreSQL como descrito no Apêndice 1. Com isso, o PostgreSQL está configurado e os testes podem ser realizados. É importante destacar que o procedimento de implantação descrito acima é o mesmo para ambas as máquinas virtuais destinadas para o PostgreSQL, isto é, tanto para a máquina com 1PC quanto de 2PC.

[1]Entrada: Estrutura XML com as notas fiscais enviadas.
Retorno: Estrutura XML contendo a mensagem do resultado da consulta do status do serviço.

3.3.2 Citus

Analogamente ao caso do PostgreSQL, a implantação do Citus se deu por meio da imagem oficial. Porém, foi necessário especificar as configurações relativas aos nós mestre, gerente e trabalhadores. Este é um passo necessário para quaisquer instalações do CitusData (2022b). O arquivo de configuração do Citus está disponível do Apêndice 3. Em particular, está especificado a configuração de um ambiente com 3 nós trabalhadores (*worker1*, *worker2* e *worker3*) e um 1 nó adicional contendo simultaneamente o gerente (*manager*) e o mestre (*master*). No caso da implantação de 6 nós trabalhadores, é adicionado mais 3 novos trabalhadores seguindo o mesmo padrão, ou seja, é necessário alterar o *service_name*, *container_name* e *deploy.placement.constraints*. Com isso, a execução da implantação é realizada pelo comando:

```
docker stack deploy -c <Path to a Compose file>
```

Por conseguinte, após a implantação do *cluster* Citus ter sido bem sucedida, é realizada a sincronização do nó *master* com os nós *workers*. A indicação do nó *master* no *cluster* é feita conectando no Citus, que está sendo executado no nó *master* utilizando o psql, e executando a consulta SQL com o comando:

```
SELECT citus_set_coordinator_host('coord.example.com', 5432);
```

Posteriormente, é conectado cada um dos nós *workers* ao nó *master*. Para tanto, serão executadas as consultas SQL no nó *master*:

```
1 SELECT * FROM citus_add_node('worker1', 5432);
2 SELECT * FROM citus_add_node('worker2', 5432);
3 SELECT * FROM citus_add_node('worker3', 5432);
```

À vista disso, faz-se necessário verificar se os nós estão se encontrando dentro do *cluster*, então, portanto, executa-se a consulta SQL no nó *master*:

```
SELECT * FROM master_get_active_worker_nodes();
```

Abaixo, destacamos um recorte, quando utilizado 3 nós *workers*, da saída da consulta acima.

```
1  node_name | node_port
2 -----------+-----------
```

```
 worker2   |      5432
 worker1   |      5432
 worker3   |      5432
(3 rows)
```

No mais, a tabela *fatonfe* é criada e distribuída entre os nós *workers*, aplicando por padrão *hash* na coluna *infprot_chnfe*. Sua execução é feita pela consulta SQL:

```
SELECT create_distributed_table('fatonfe', 'infprot_chnfe');
```

Ainda, a partir da consulta precedente, são criados os fragmentos (*shards*) nos nós *workers* utilizando os valores de configuração padrão do Citus: *citus.shard_count=32* e *citus.shard_replication_factor=1*, os quais representam a quantidade de fragmentos que devem ser criados em cada nó *worker* e quantas réplicas de cada fragmento deve ter existir, respectivamente. Feito isso, o *cluster* Citus está configurado e os testes poderão ser realizados.

3.4 Base de Dados

Foi criado um banco de dados de testes, tendo sido montada duas bases cada uma contendo, aproximadamente, 849 mil e 10 milhões de registros da tabela *fatonfe*, respectivamente. Para uma melhor visualização, estas informações foram organizadas na Tabela 2.

Tabela 2: Informações Bases de Dados.

Base de Dados - Tabela	Tamanho	Registros
Pequena - Fatonfe	0.8 GB	849.601
Grande - Fatonfe	11 GB	9.994.158

Fonte: Elaborada pelo autor (2022).

Destarte, os campos com informações sensíveis tais como chaves de acesso a notas fiscais, CPFs e CNPJs de emissores e destinatários, assinaturas digitais entre outros, foram anonimizados, preservando recorrências em todos os casos relevantes.

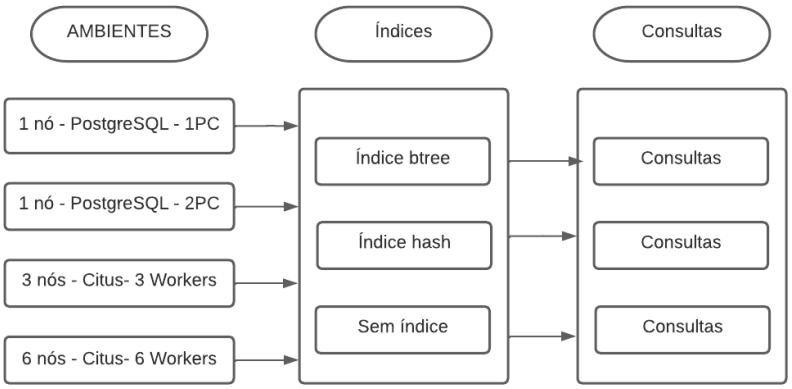
3.5 Descrição dos Cenários

Como cenários de execução foram utilizados: **Cenário 0:** 1 nó PostgreSQL com 1PC, **Cenário 1:** 1 nó PostgreSQL com 2PC, **Cenário 2:** 3 nós Citus com 3 *workers* e **Cenário 3:** 6 nós Citus com 6 *workers*. Estes cenário foram construídos com base nos

recursos disponíveis objetivando ter 2 grupos distintos de cenários, um de base (Cenários: 0 e 2) e outro com o dobro de poder computacional (Cenários: 1 e 3).

Para cada cenário, existem 3 subdivisões com base no índice utilizado, que são: sem índice, índice *btree* e índice *hash*. Para estas escolhas foi levado em consideração que o *btree* é o padrão de índice utilizado pelo PostgreSQL e o *hash* é padrão utilizado pelo Citus para distribuição da tabela pelo *cluster*. No total, foram observados 12 cenários, em cada cenário, foram executadas 60 ou 79 consultas (que serão descritas no próximo tópico), a depender da amostra utilizada. Desta forma, a Figura 4, representa o fluxograma para cada cenário.

Figura 4: Fluxograma para os cenários de execução.



Fonte: Elaborada pelo Autor (2022).

Objetivando contornar a interferência de otimizações por *caching*, cada consulta é executada três vezes. Contudo, desconsidera-se a primeira execução, permitindo, assim, que todas as consultas sejam igualmente beneficiadas por qualquer ganho de performance por *caching*. Para mais, como não constatou-se diferenças significativas no tempo de execução entre a segunda e terceira, foi considerado apenas o tempo da terceira e última execução.

3.6 Métrica de Avaliação

Como métricas de avaliação foram utilizadas a alocação de *shards* por nó e o tempo da terceira execução de cada consulta.

Posto isto, neste trabalho, para mensurar o desempenho dos SGBDs no contexto de notas fiscais foi avaliada a performance do PostgreSQL e do Citus com relação ao tempo de execução das consultas. Dessa maneira, a partir do embasamento teórico com a leitura dos trabalhos precedentes, foi possível estabelecer como métricas de implementação o registro completo, seleção de colunas e agregações, aos quais são definidos a seguir.

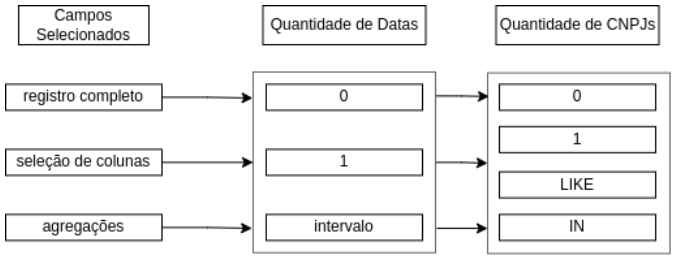
a) Registro Completo: Retorna todos os campos e atributos que pertencem a consulta;

b) Seleção de Colunas: Engloba apenas um subconjunto dos campos e atributos que pertencem a consulta.

c) Agregações: Retorna métricas de um subconjunto dos campos e atributos que pertencem a consulta.

Sendo assim, para cada uma das estratégias têm-se os filtros quantidade de datas e quantidade de CNPJ. Os filtros e consultas foram definidos levando em consideração as atividades habituais dos clientes que utilizam a base de dados da SEFAZ/PB. O esquema de consultas ilustrado na Figura 5.

Figura 5: Esquema de Consultas.



Fonte: Elaborada pelo Autor (2022).

O código completo das consultas, de acordo com o apresentado na lista precedente, está disponível do Apêndice 4.

4 RESULTADOS E DISCUSSÃO

Neste Capítulo, são apresentados os resultados experimentais acerca da análise de desempenho dos SGBD analisados neste trabalho.

Como apresentado no Capítulo anterior, a análise de desempenho se deu por meio da execução de consultas de recuperação de dados e coleta do tempo de execução destas consultas. Ademais, neste trabalho, a análise de desempenho em ambos SGBDs (Citus e PostgreSQL) considerou cenários com diferentes configurações de índices, sendo elas: uso de índices *btree*, uso de índices *hash* e nenhum índice sendo usado. Exceto a configuração de índice, ambos os SGBDs fizeram uso de configuração padrão. Isto é, não foram realizadas otimizações por meio de variáveis como tamanho de cache ou página.

Para cada cenário, foi elaborado um grupo específico de consultas, sendo eles uso de agregações (*avg + count + sum*), recuperação de registro completo (uma tupla com todas as suas colunas) e recuperação de registro com coluna específica (uma tupla com a coluna específica). Os cenários de análise de desempenho ainda consideraram dois tamanhos de bases de dados: 850 mil (*small*) e 10 milhões (*big*) de registros.

Dada a variabilidade do tempo (em milissegundos (ms)) de resposta em cada consulta, foram realizadas repetições das consultas, sendo, então, considerado apenas o tempo de execução da terceira e última execução uma vez que as primeiras execuções usualmente são usadas pelos SGBDs para formação de estatísticas e montagem de cache. Desta forma, em todo o texto, quando for mencionado tempo de execução estará sendo considerado o tempo de execução da terceira e última execução. Por fim, os resultados com as estatísticas descritivas são apresentados em forma de Tabelas no Apêndice 5.

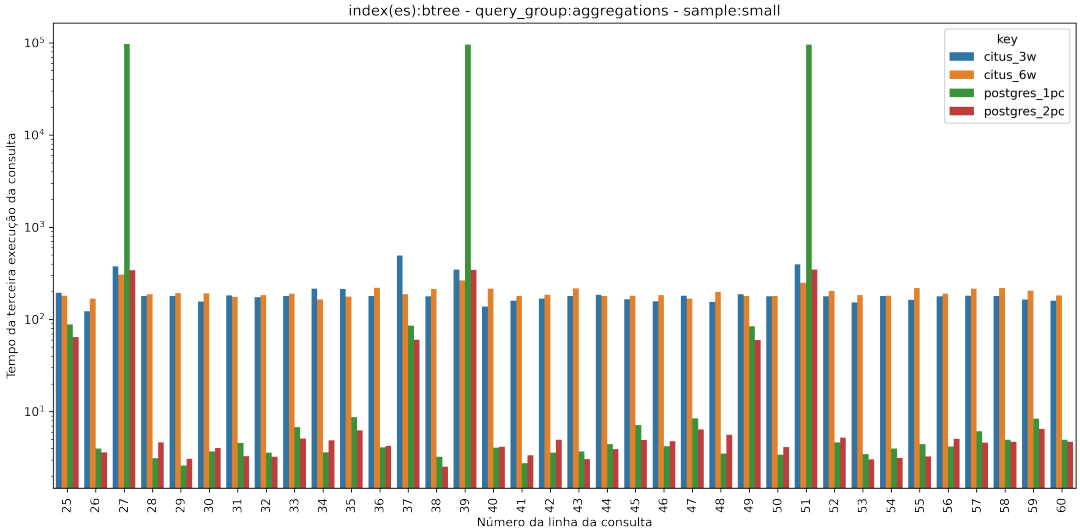
4.1 Buscas com índices *btree*

Nas Figuras 6 e 7, é ilustrado o tempo de execução para resposta das consultas para os cenários com o uso de índices *btree*. As consultas foram executadas em quatro ambientes, sendo dois para o Citus e dois para o PostgreSQL. No caso do Citus, foram considerados os ambientes com 3 e 6 *workers*. Para o PostgreSQL, foram consideradas uma máquina virtual com dois núcleos de CPU e 2 GB de RAM e outra máquina com quatro núcleos de CPU e 4 GB de RAM, sendo nomeadas de 1PC e 2PC, respectivamente.

Na Figura 6, são apresentados os resultados relacionados às consultas executadas sobre a base *small*, enquanto que na Figura 7 aqueles relacionados a base *big*. Diante dos resultados apresentados na Figura 6, notamos que, em 92% das consultas, o PostgreSQL apresentou melhores tempos de respostas, tanto para 1PC quanto para 2PC em comparação com o Citus. No entanto, destaca-se que nas consultas de número 27, 39 e 57 (para o PostgreSQL com 1PC), para o índice *btree sum*, *avg* e *count*, respectivamente, mostrados

na Tabela 3, o PostgreSQL apresentou tempo de execução atípico em comparação aos demais.

Figura 6: Gráfico para o tempo de execução da terceira execução, considerando consultas com índices *btree*, agregações (avg + count + sum) e amostra com 850 mil de notas.



Fonte: Elaborada pelo Autor (2023).

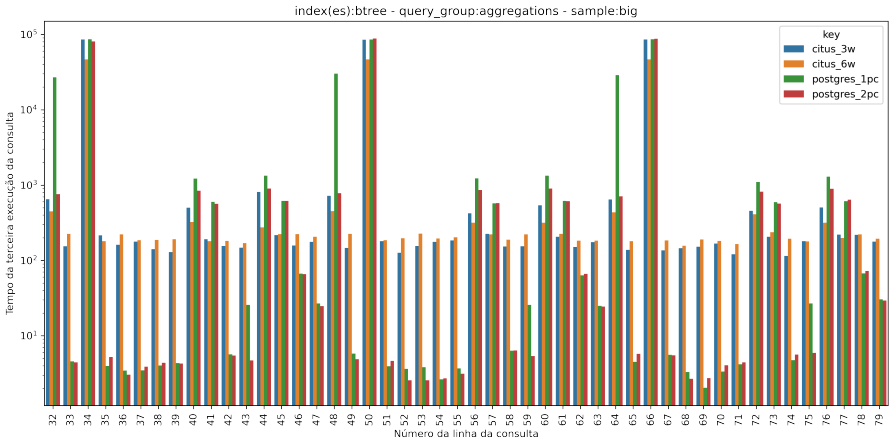
Tabela 3: Consultas para agregações na base *small* utilizando o índice *btree*.

Número	Consulta
27	SELECT sum(id_fatonfe) FROM fatonfe WHERE infNFe_emit_CNPJ LIKE '0411%';
39	SELECT avg(id_fatonfe) FROM fatonfe WHERE infNFe_emit_CNPJ LIKE '0411%';
57	SELECT count(id_fatonfe) FROM fatonfe WHERE infNFe_emit_CNPJ LIKE '0411%';

Fonte: Elaborada pelo Autor (2023).

A análise gráfica (Figura 7) corrobora para inferir que o PostgreSQL obteve desempenho 63% superior ao Citus considerando os cenários avaliados. Novamente, para a consulta de número 32 do PostgreSQL com 1PC, a consulta 34 para todos os casos, a consulta 48 do PostgreSQL com 1PC, a consulta 50 para todos os casos, a consulta 64 do PostgreSQL com 1PC, e a consulta 66 para todos os casos, para o índice *btree* sum, avg e count, destacados na Tabela 4, apresentaram resultados atípicos.

Figura 7: Gráfico para o tempo de execução da terceira execução, considerando consultas com índices *btree*, agregações (avg + count + sum) e amostra com 10 milhões de notas .



Fonte: Elaborada pelo Autor (2023).

Tabela 4: Consultas para agregações na base *big* utilizando o índice *btree*.

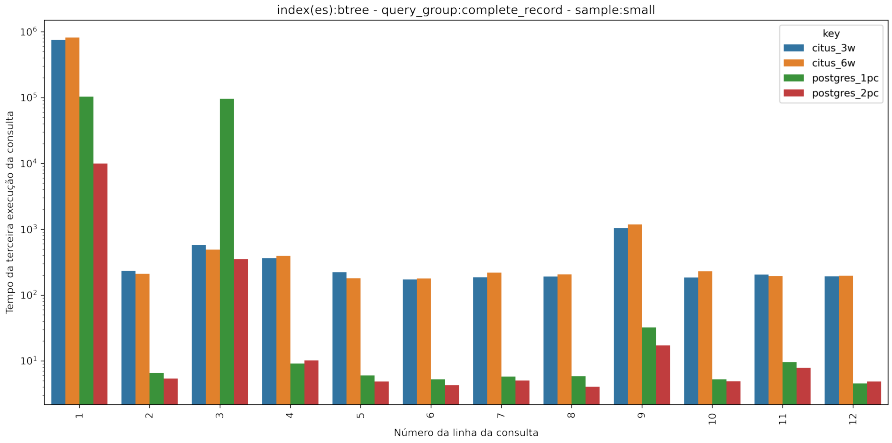
Número	Consulta
32	SELECT sum(id_fatonfe) FROM fatonfe;
34	SELECT sum(id_fatonfe) FROM fatonfe WHERE infNFe_emit_CNPJ LIKE '04116%';
48	SELECT avg(id_fatonfe) FROM fatonfe;
50	SELECT avg(id_fatonfe) FROM fatonfe WHERE infNFe_emit_CNPJ LIKE '04116%';
64	SELECT count(id_fatonfe) FROM fatonfe;
66	SELECT count(id_fatonfe) FROM fatonfe WHERE infNFe_emit_CNPJ LIKE '04116%';

Fonte: Elaborada pelo Autor (2023).

Nas Figuras 8 e 9, estão as representações para o tempo de execução da terceira execução, considerando os grupos de consultas com métrica de registro completo para as amostras com 850 mil e 10 milhões respectivamente. Para este caso, notamos que na primeira consulta (número 1), o tempo de execução para uma amostra com 850 mil é maior com relação a amostra com 10 milhões. Estes resultados eram esperados tendo em vista que para a amostra menor não são utilizados filtros, enquanto que para a amostra maior é aplicado o filtro de CNPJ. Para além disso, destacamos ainda que em ambas as amostras o PostgreSQL apresentou, na média, tempo de execução menor em comparação com o Citus. Por fim, destaca-se que a consulta de número 3 (para o PostgreSQL com 1PC), mostrada na Tabela 5, o PostgreSQL apresentou tempo de execução atípico em

comparação aos demais.

Figura 8: Gráfico para o tempo de execução da terceira execução, considerando consultas com índices *btree*, com métrica de registro completo e amostra com 850 **mil notas.**



Fonte: Elaborada pelo Autor (2023).

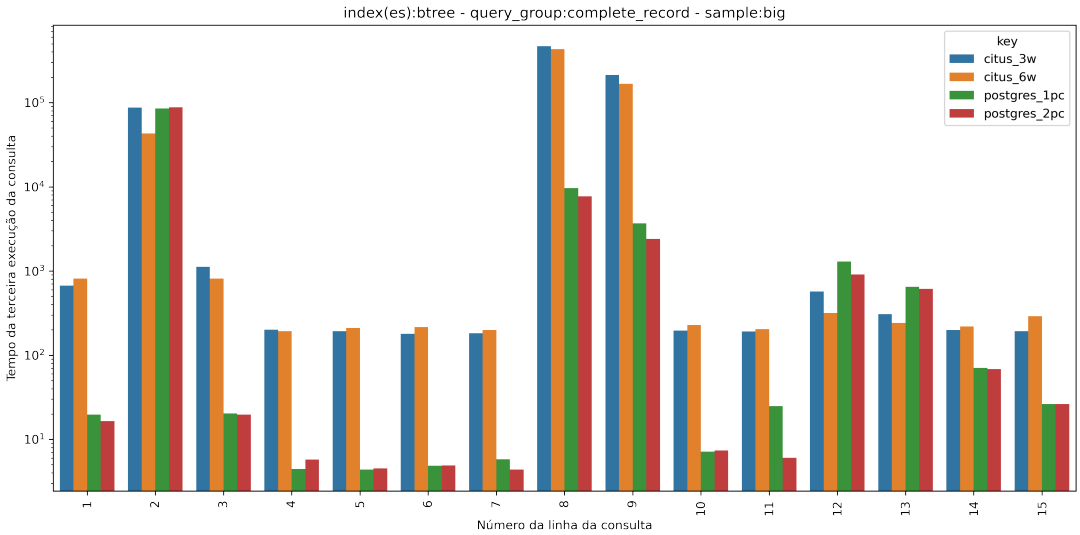
Tabela 5: Consulta para recuperação de registro completo na base *small* utilizando o índice *btree*.

Número	Consulta
3	SELECT * FROM fatonfe WHERE infNFe_emit_CNPJ LIKE '0411%';

Fonte: Elaborada pelo Autor (2023).

A Figura 9 apresenta os tempos de execução das consultas com índices *btree*, com métrica de registro completo e amostra com 10 milhões de notas. Comparando os valores mínimos, máximos e médios é possível inferir que os SGBDs PostgreSQL com 1PC e 2PC não apresentaram variações significativas, sendo portanto muito próximos. Por outro lado, para o caso do Citus com 3 e 6 *workers*, percebe-se que apresentaram desempenho ligeiramente distintos, sendo em média o Citus com 6 *workers* melhor.

Figura 9: Gráfico para o tempo de execução da terceira execução, considerando consultas com índices *btree*, com métrica de registro completo e amostra com 10 milhões de notas.

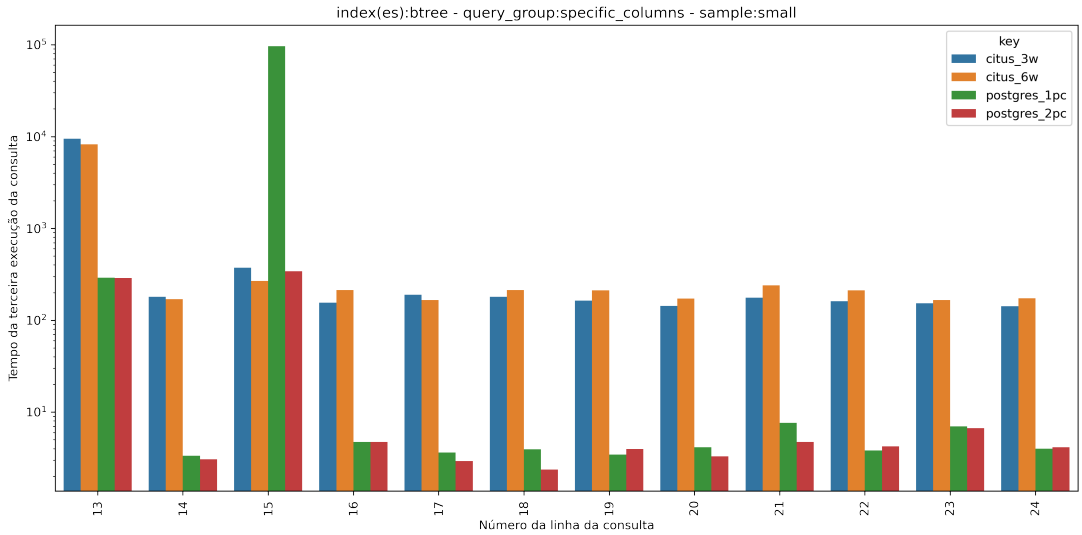


Fonte: Elaborada pelo Autor (2023).

Nas Figuras 10 e 11, estão representações para o tempo de execução da terceira execução, considerando os grupos de consultas com métrica de seleção de coluna específica. Neste caso, a Figura 10 evidencia um tempo de execução muito maior para PostgreSQL 1PC, quando submetido a consulta de número 15, destacada na Tabela 6.

Quando é considerado uma amostra maior, como apresentado na Figura 11, em geral, o PostgreSQL tanto 1PC quanto 2PC apresentou uma performance 81% superior que o Citus. Ademais, destaca-se que, neste caso, embora o Citus com 3 *workers* tenha apresentado valor máximo maior que o Citus com 6 *workers*, pode-se inferir que seu desempenho foi melhor, visto que apresentou um valor mínimo menor em média.

Figura 10: Gráfico para o tempo de execução da terceira execução, considerando consultas com índices *btree*, com métrica de seleção de coluna específica e amostra com 850 mil de notas.



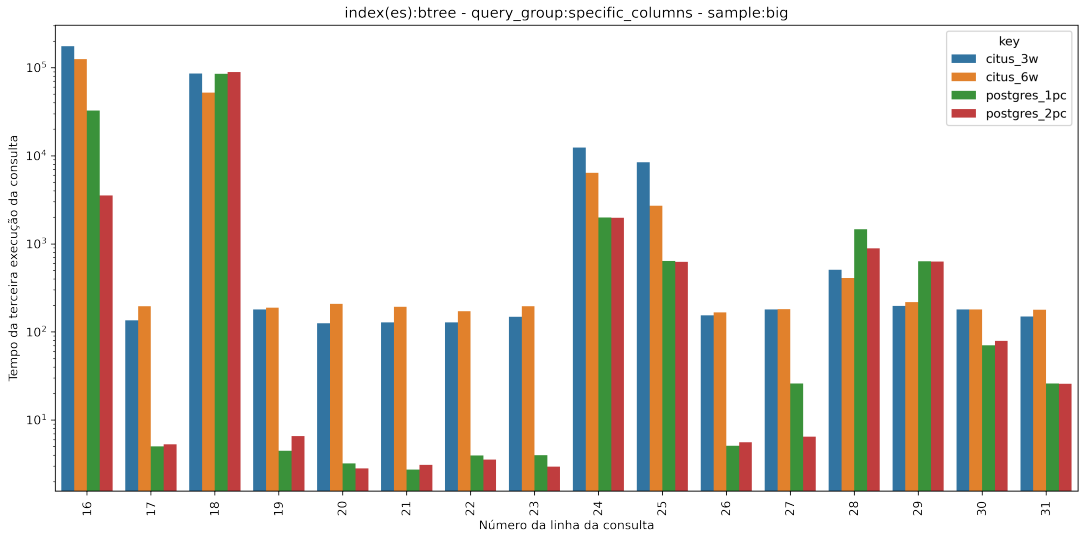
Fonte: Elaborada pelo Autor (2023).

Tabela 6: Consulta para recuperação de coluna específica na base *small* utilizando o índice *btree*.

Número	Consulta
15	SELECT id_fatonfe FROM fatonfe WHERE infNFe_emit_CNPJ LIKE '0411%';

Fonte: Elaborada pelo Autor (2023).

Figura 11: Gráfico para o tempo de execução da terceira execução, considerando consultas com índices *btree*, com métrica de seleção de coluna específica e amostra com 10 milhões de notas.



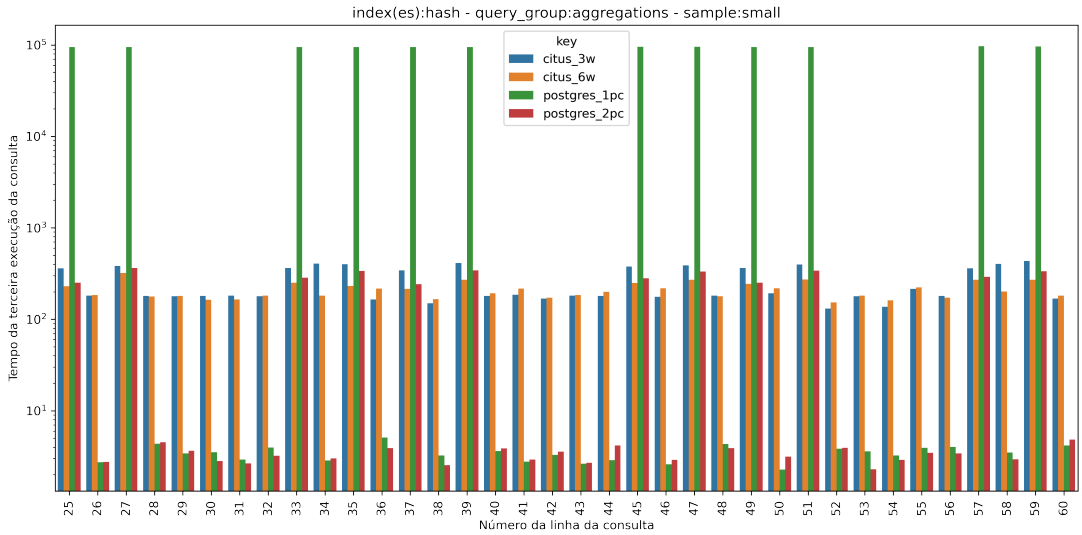
Fonte: Elaborada pelo Autor (2023).

4.2 Buscas com índices *hash*

Agora, as consultas são realizadas considerando índices *hash*. Nas Figuras 12 e 13, é apresentado o tempo de execução da terceira execução, considerando uma amostra com 850 mil e 10 milhões de notas, respectivamente, com consultas de agregações. Para o caso de 850 mil notas, notamos que em 12 consultas o PostgreSQL 1PC apresentou performance bastante inferior com relação aos demais, sendo ligeiramente equivalente ao PostgreSQL 2PC e 67% superior que o Citus, nos demais casos. Um subconjunto dessas 12 consultas, referentes a *sum*, está destacado na Tabela 7.

Para uma amostra com 10 milhões de notas, Figura 13, foram realizadas 78 consultas. Tendo em vista uma amostra maior, como esperado o tempo de execução é maior que no caso anterior. Novamente o PostgreSQL, tanto com 1PC quanto com 2PC, apresentou uma performance 63% superior em comparação com o Citus. Destacamos na Tabela 8, o subconjunto referente a *sum* com tempo de execução bastante superior em relação ao caso anterior. Embora não muito evidente pela análise gráfica, nota-se que, neste caso, o SGBD PostgreSQL com 1PC obteve melhor desempenho que todos os demais.

Figura 12: Gráfico para o tempo de execução da terceira execução, considerando consultas com índices *hash*, agregações (avg + count + sum) e amostra com 850 mil de notas.



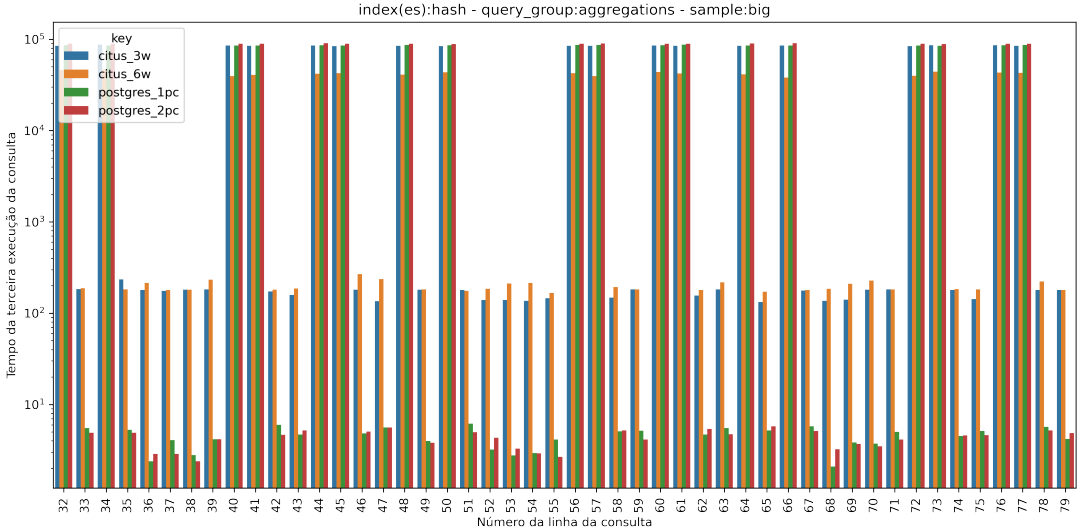
Fonte: Elaborada pelo Autor (2023).

Tabela 7: Consultas para agregações na base *small* utilizando o índice *hash*.

Número	Consulta
25	SELECT sum(id_fatonfe) FROM fatonfe;
27	SELECT sum(id_fatonfe) FROM fatonfe WHERE infNFe_emit_CNPJ LIKE '0411%';
33	SELECT sum(id_fatonfe) FROM fatonfe WHERE infNFe_ide_dhEmi BETWEEN '2022-01-01' AND '2022-06-01'
35	SELECT sum(id_fatonfe) FROM fatonfe WHERE infNFe_emit_CNPJ LIKE '0411%' AND infNFe_ide_dhEmi BETWEEN '2022-01-01' AND '2022-06-01'

Fonte: Elaborada pelo Autor (2023).

Figura 13: Gráfico para o tempo de execução da terceira execução, considerando consultas com índices *hash*, agregações (avg + count + sum) e amostra com 10 milhões de notas .



Fonte: Elaborada pelo Autor (2023).

Tabela 8: Consultas para agregações na base *big* utilizando o índice *hash*.

Número	Consulta
32	SELECT sum(id_fatonfe) FROM fatonfe;
34	SELECT sum(id_fatonfe) FROM fatonfe WHERE infNFe_emit_CNPJ LIKE '0411%';
40	SELECT sum(id_fatonfe) FROM fatonfe WHERE infNFe_ide_dhEmi BETWEEN '2022-03-01 17:06:27-03' AND '2022-04-04 10:15:00.000 -0300'
41	SELECT sum(id_fatonfe) FROM fatonfe WHERE infNFe_ide_dhEmi BETWEEN '2022-03-01 17:06:27-03' AND '2022-03-02 22:06:26.000 -0300'
44	SELECT sum(id_fatonfe) FROM fatonfe WHERE infNFe_emit_CNPJ LIKE '04116%' AND infNFe_ide_dhEmi BETWEEN '2022-03-01 17:06:27-03' AND '2022-04-04 10:15:00.000 -0300';
45	SELECT sum(id_fatonfe) FROM fatonfe WHERE infNFe_emit_CNPJ LIKE '04116%' AND infNFe_ide_dhEmi BETWEEN '2022-03-01 17:06:27-03' AND '2022-03-02 22:06:26.000 -0300';

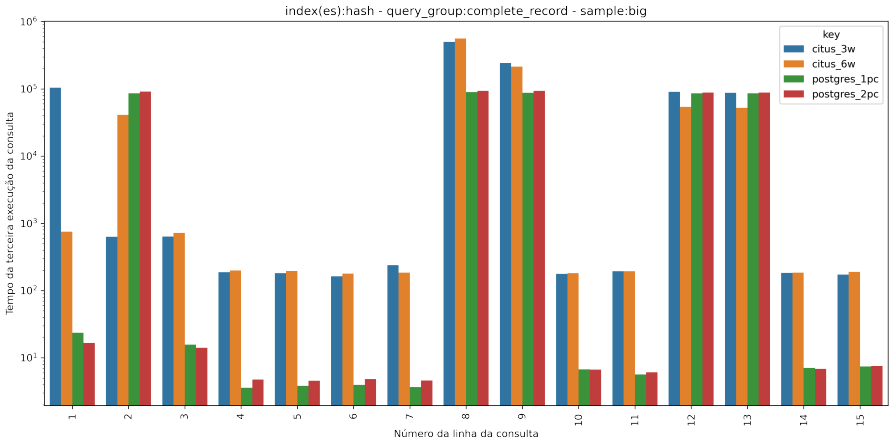
Fonte: Elaborada pelo Autor (2023).

Nas Figuras 15 e 14, são apresentadas as performances considerando o registro completo. Para este caso, nota-se que o Citus com 3 *workers* conseguiu uma performance melhor com relação ao PostgreSQL, para ambos os casos, quando realizada a consulta de

número 2, indicada na Tabela 9, para amostra com 10 milhões de notas. Apesar disso, para as demais consultas seu desempenho é bem inferior com relação ao PostgreSQL, tanto com 1PC quanto com 2PC.

Para amostra com 850 mil notas, Figura 15, as consultas 3, 9 e 11, indicadas na Tabela 10, apresentam-se discrepantes, quando considerado o PostgreSQL com 1PC. Apesar disso, de uma forma geral seu desempenho foi 92% superior ao Citus com 3 e 6 *workers*.

Figura 14: Gráfico para o tempo de execução da terceira execução, considerando consultas com índices *hash*, com métrica de registro completo e amostra com 10 milhões de notas.



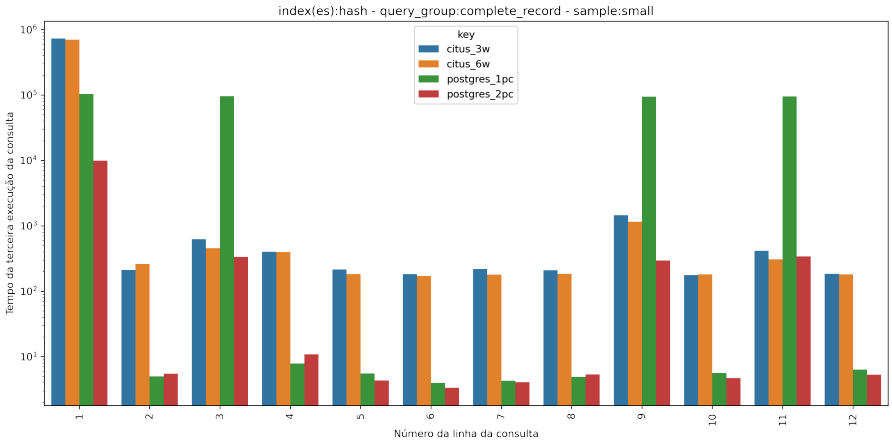
Fonte: Elaborada pelo Autor (2023).

Tabela 9: Consulta para recuperação de registro completo na base *big* utilizando o índice *hash*.

Número	Consulta
1	SELECT * FROM fatonfe WHERE infNFe_emit_CNPJ = '04116842000164';
2	SELECT * FROM fatonfe WHERE infNFe_emit_CNPJ LIKE '04116%';

Fonte: Elaborada pelo Autor (2023).

Figura 15: Gráfico para o tempo de execução da terceira execução, considerando consultas com índices *hash*, com métrica de registro completo e amostra com 850 mil de notas.



Fonte: Elaborada pelo Autor (2023).

Tabela 10: Consultas para recuperação de registro completo na base *small* utilizando o índice *hash*.

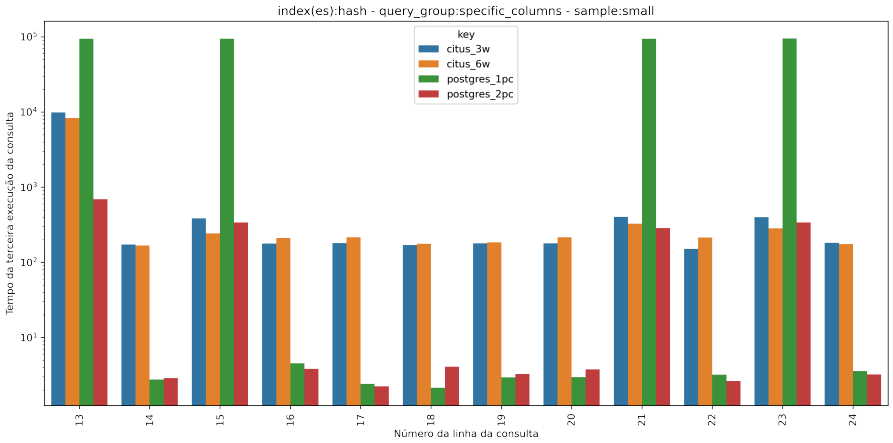
Número	Consulta
3	SELECT * FROM fatonfe WHERE infNFe_emit_CNPJ LIKE '0411%';
9	SELECT * FROM fatonfe WHERE infNFe_ide_dhEmi BETWEEN '2022-01-01' AND '2022-06-01';
11	SELECT * FROM fatonfe WHERE infNFe_emit_CNPJ LIKE '0411%' AND infNFe_ide_dhEmi BETWEEN '2022-01-01' AND '2022-06-01';

Fonte: Elaborada pelo Autor (2023).

Com relação às consultas por seleção de coluna específica, os resultados estão nas Figuras 16 e 17. Nota-se que para amostra com 850 mil, o Citus com 6 *workers* obteve uma performance ligeiramente melhor que o PostgreSQL para as consultas 15 e 23, indicadas na Tabela 11, sendo pior para as demais. Estes valores discrepantes influenciaram diretamente a média do desempenho.

Para uma amostra com 10 milhões de notas, a performance do Citus é pior em todos os casos. Apesar disso, devido a presença de consultas discrepantes, mesmo com valor mínimo cerca de aproximadamente 60 vezes maior que o PostgreSQL com 1PC e 2PC, em média, o Citus com 6 *workers* obteve o melhor desempenho.

Figura 16: Gráfico para o tempo de execução da terceira execução, considerando consultas com índices *hash*, com métrica de seleção de coluna específica e amostra com 850 mil de notas.



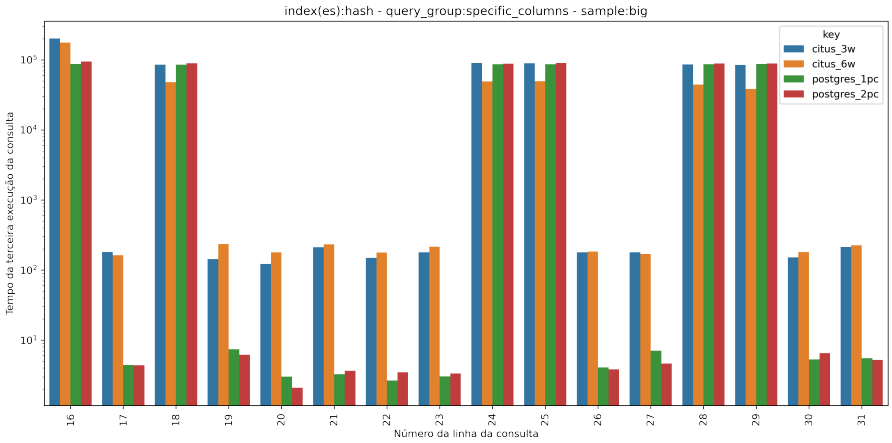
Fonte: Elaborada pelo Autor (2023).

Tabela 11: Consultas para recuperação de coluna específica na base *small* utilizando o índice *hash*.

Número	Consulta
13	SELECT id_fatonfe FROM fatonfe;
15	SELECT id_fatonfe FROM fatonfe WHERE infNFe_emit_CNPJ LIKE '0411%';
21	SELECT id_fatonfe FROM fatonfe WHERE infNFe_ide_dhEmi BETWEEN '2022-01-01' AND '2022-06-01';
23	SELECT id_fatonfe FROM fatonfe WHERE infNFe_emit_CNPJ LIKE '0411%' AND infNFe_ide_dhEmi BETWEEN '2022-01-01' AND '2022-06-01';

Fonte: Elaborada pelo Autor (2023).

Figura 17: Gráfico para o tempo de execução da terceira execução, considerando consultas com índices *hash*, com métrica de seleção de coluna específica e amostra com 10 milhões de notas.



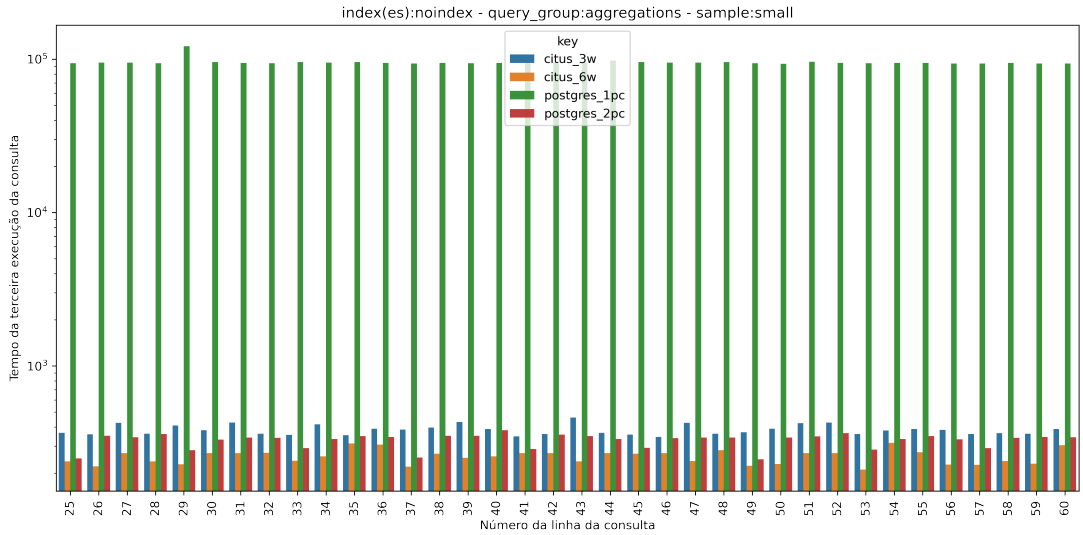
Fonte: Elaborada pelo Autor (2023).

4.3 Buscas sem índices

Nesta Seção, são apresentados os resultados quando as consultas são realizadas sem índices. Na Figura 18, são apresentados os resultados para amostra com 850 mil notas com agregações (avg + count + sum). Percebe-se que, neste caso, o tempo de execução do PostgreSQL com 1PC foi superior aos demais em 100% das consultas. Destaca-se que o Citus com 6 *workers* foi o melhor, obtendo o menor tempo de execução em todas as consultas. Uma possível explicação para esta performance é o fato de não se utilizar índices.

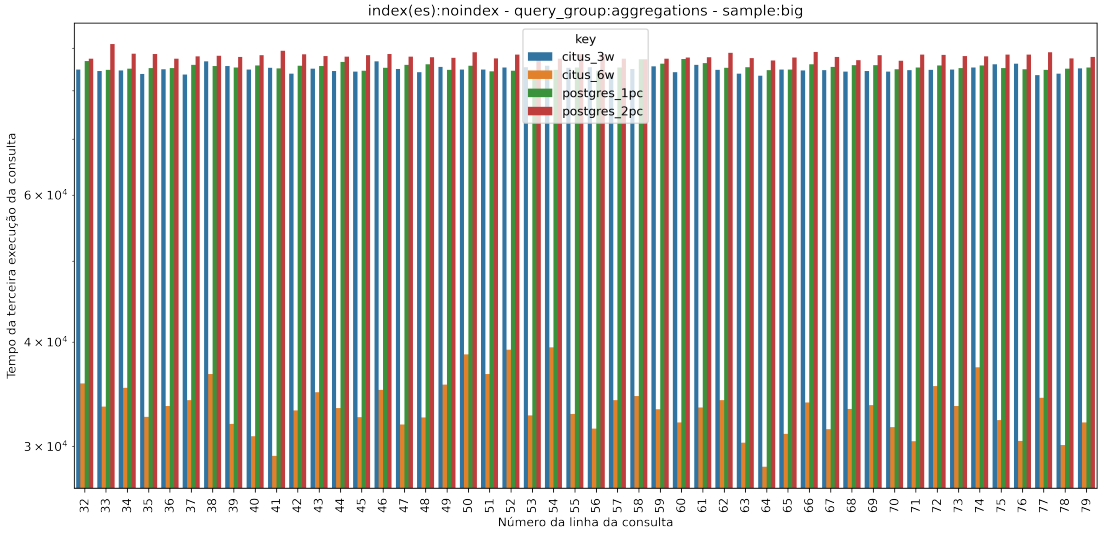
No caso da Figura 19, ainda com com agregações (avg + count + sum), porém com a base de dados de 10 milhões de notas. Para este caso, nota-se que os SGBDs Citus com 3 *workers*, PostgreSQL com 1PC e PostgreSQL com 2PC obtiveram tempo de execução superior a ordem de 6×10^4 em todas as consultas. Similarmente a base com 850 mil notas, o SGBD Citus com 6 *workers* obteve o melhor desempenho em todas as consultas. Outrossim, apesar do valor mínimo ser obtido pelo SGBD PostgreSQL com 1PC, em média o SGBD Citus com 6 *workers*, de fato foi melhor.

Figura 18: Gráfico para o tempo de execução da terceira execução, considerando consultas sem índices, agregações (avg + count + sum) e amostra com 850 mil de notas.



Fonte: Elaborada pelo Autor (2023).

Figura 19: Gráfico para o tempo de execução da terceira execução, considerando consultas sem índices, agregações (avg + count + sum) e amostra com 10 milhões de notas.



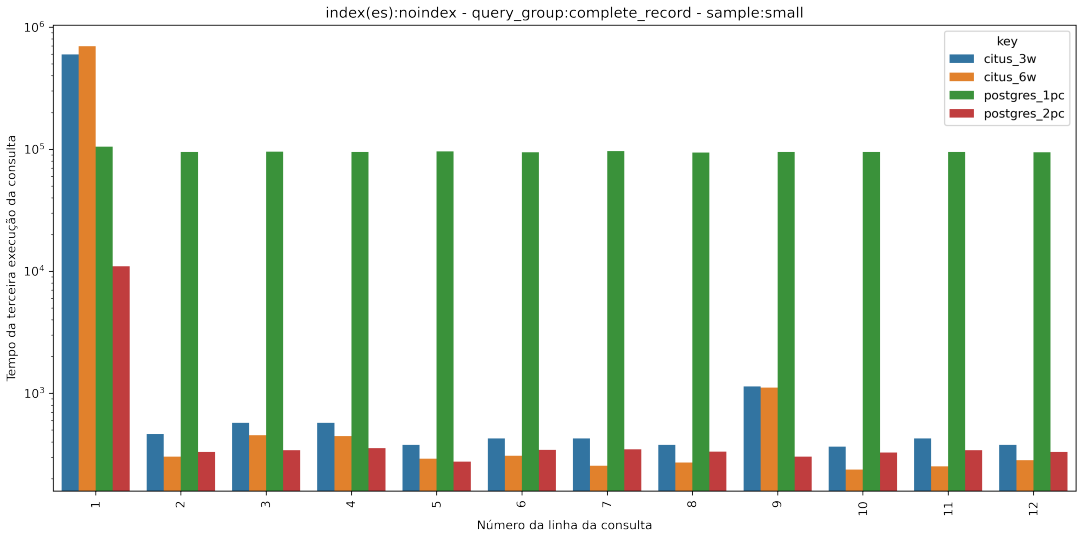
Fonte: Elaborada pelo Autor (2023).

Na Figura 20, as consultas são realizadas considerando a métrica do registro completo na base com 850 mil notas. Para este caso, percebe-se que o SGBD PostgreSQL

com 1PC obteve o pior desempenho em aproximadamente 92% das consultas, nestes casos, sendo muito discrepante com relação aos demais.Destaca-se ainda que o Citus com 6 *workers*, obteve o melhor desempenho em aproximadamente 58% das consultas. Apesar disso, em média, o SGBD PostgreSQL com 2PC obteve um melhor desempenho.A discrepância do Citus na consulta 1, indicada na Tabela 12, pode ser explicada por problemas de gargalos de I/O entre nós, tendo em vista que é uma consulta que não realiza filtragem de dados.

A Figura 21 ilustra os resultados para a base de dados com 10 milhões de notas, ainda com registros completos. Neste caso, nota-se que o Citus com 3 *workers*, PostgreSQL com 1PC e PostgreSQL com 2PC apresentaram tempo de execução aproximadamente constante na maioria das consultas, com resultados discrepantes nas consultas de números 8 e 9. A discrepância apresentada pelas consultas 8 e 9, descritas na Tabela 13, justifica-se por não existir fragmentação por data nem índice. Ademais, por não utilizar índices, novamente Citus com 6 *workers* apresentou o melhor desempenho em todas as consultas.

Figura 20: Gráfico para o tempo de execução da terceira execução, considerando consultas sem índices, com métrica de registro completo e amostra com 850 **mil de notas.**



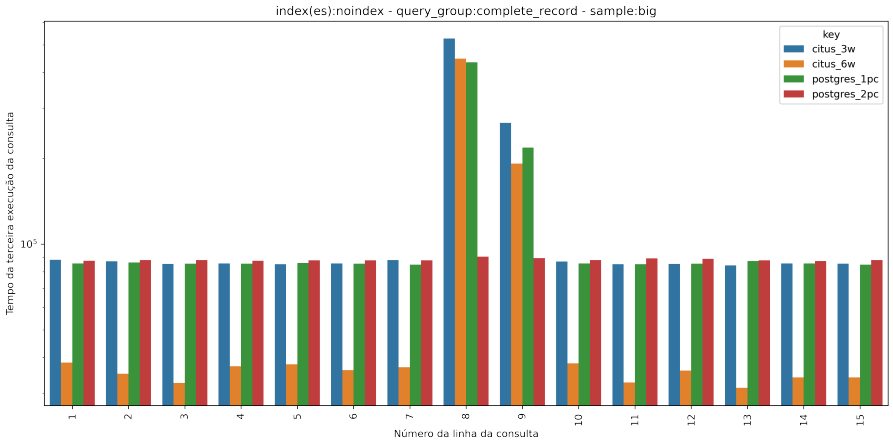
Fonte: Elaborada pelo Autor (2023).

Tabela 12: Consulta para recuperação de registro completo na base *small* sem índice.

Número	Consulta
1	SELECT * FROM fatonfe;

Fonte: Elaborada pelo Autor (2023).

Figura 21: Gráfico para o tempo de execução da terceira execução, considerando consultas sem índices, com métrica de registro completo e amostra com 10 milhões de notas.



Fonte: Elaborada pelo Autor (2023).

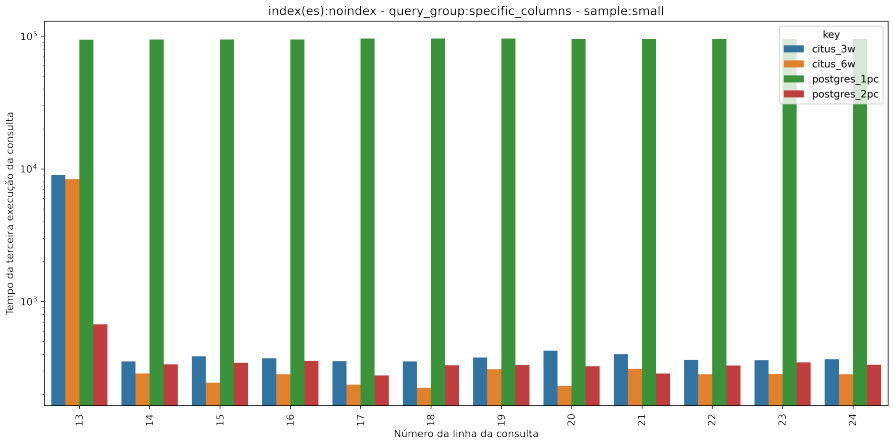
Tabela 13: Consultas para recuperação de registro completo na base *big* sem índice.

Número	Consulta
8	SELECT * FROM fatonfe WHERE infNFe_ide_dhEmi BETWEEN '2022-03-01 17:06:27-03' AND '2022-04-04 10:15:00.000 -0300';
9	SELECT * FROM fatonfe WHERE infNFe_ide_dhEmi BETWEEN '2022-03-01 17:06:27-03' AND '2022-03-02 22:06:26.000 -0300';

Fonte: Elaborada pelo Autor (2023).

Agora, são consideradas consultas por seleção de coluna específica da base de dados. Na Figura 22, os resultados são para a amostra com 850 mil notas. Notadamente, o PostgreSQL 1PC obteve um desempenho inferior em relação aos demais. O Citus com 6 *workers* obteve o melhor desempenho, cerca de 83% das consultas. Novamente a consulta de número 13 (Tabela 14) provavelmente apresentou problemas de gargalos de IO nos nós devido ao fato de ser uma consulta sem filtros.

Figura 22: Gráfico para o tempo de execução da terceira execução, considerando consultas sem índices, com métrica de seleção de coluna específica e amostra com 850 mil de notas.



Fonte: Elaborada pelo Autor (2023).

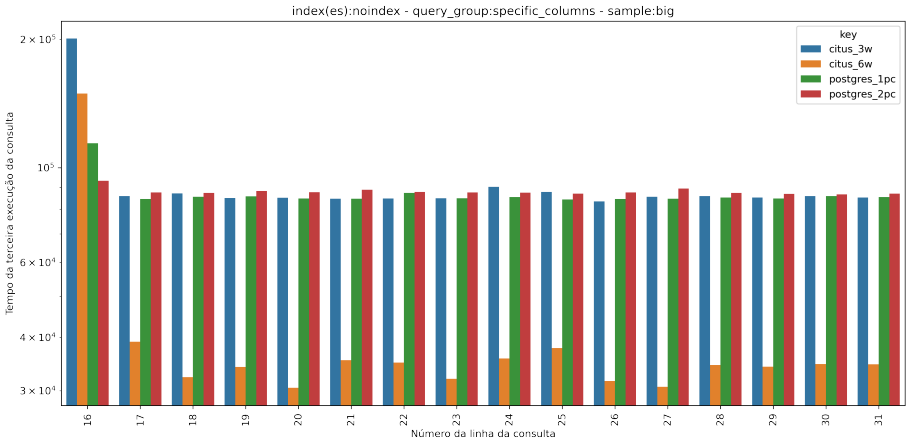
Tabela 14: Consulta para recuperação de coluna específica na base *small* sem índice.

Número	Consulta
13	SELECT id_fatonfe FROM fatonfe;

Fonte: Elaborada pelo Autor (2023).

Quando a base de dados é maior, com 10 milhões de notas, como na Figura 23, similarmente quando das consultas completas, o Citus com 3 *workers*, PostgreSQL 1PC e PostgreSQL 2PC apresentaram tempo de execução aproximadamente constante e, mais uma vez o Citus com 6 *workers* com melhor desempenho. Novamente a consulta de número 16 (Tabela 15) apresentou resultado atípico.

Figura 23: Gráfico para o tempo de execução da terceira execução, considerando consultas sem índices, com métrica de seleção de coluna específica e amostra com 10 milhões de notas .



Fonte: Elaborada pelo Autor (2023).

Tabela 15: Consulta para recuperação de coluna específica na base *big* sem índice.

Número	Consulta
16	SELECT id_fatonfe FROM fatonfe;

Fonte: Elaborada pelo Autor (2023).

Assim, os resultados mostraram que para conjuntos de dados pequenos a ferramenta Citus, embora não tenha apresentado melhores resultados que o PostgreSQL, apresentou desempenho eficiente com relação ao tempos de execução.

Destacamos que quando não foram aplicados índices nas consultas, o Citus apresentou desempenho 92,81% superior em relação ao PostgreSQL. Contudo, foram verificados problemas com gargalos de I/O devido ao fato da não aplicação de filtro.

Ademais, de um modo geral, as consultas no PostgreSQL foram 52,52% mais rápidas do que no Citus. Destaque para o desempenho superior do PostgreSQL em comparação com o Citus para os índices *btree*, em cerca de 79,86% das consultas, e *hash*, em cerca de 70,50% das consultas.

5 CONSIDERAÇÕES FINAIS

Este trabalho apresentou a análise de desempenho entre os sistemas de gerenciamento de bancos de dados PostgreSQL e o Citus, analisando as características de tempo de execução para consultas. Para tanto, os bancos de dados foram alimentados com uma base de dados composta de itens da NF-e obtidos a partir da Secretaria de Estado da Fazenda da Paraíba.

Os dois SGBDs foram implementados obedecendo suas particularidades estruturais de modelagem, contudo, sendo aplicados à mesma base de dados. Inicialmente um lote de 850 mil e, posteriormente, outro lote de 10 milhões de NF-es. A partir de então, foram realizados os experimentos objetivando avaliar a performance de cada um dos SGBDs, com relação às consultas realizadas, isto é, o tempo de resposta que cada SGBD tem no retorno das consultas.

Foram utilizados os recursos de índices *btree*, *hash* e sem índices. Com isso, foram considerados dois cenários de execução para ambos os SGBDs. Desta forma, foram criados os ambientes experimentais e realizados os testes de execução.

É imperioso destacar que esta pesquisa trouxe contribuições significativas para comunidade científica, tendo vista a disponibilidade de uma análise de performance de dois SGBD aplicados em uma base de dados da NF-es sendo, portanto, uma ferramenta para auxiliar o pesquisador na escolha do SGBD.

Outrossim, como trabalhos futuros pretendemos:

1. Aumentar o tamanho da base de testes de modo à aproximar-se do tamanho da base real (aproximadamente 15TB);
2. Executar um número maior de consultas;
3. Utilizar outras variações de quantidades de nós nos *clusters* Citus e tamanhos de máquinas virtuais PostgreSQL compatíveis;
4. Utilizar outras disposições de quantidade de fragmentos que devem ser criados em cada nó;
5. Testar outra ferramenta gerenciadora de *containers* como por exemplo *kubernetes* e comparar a implantação do modelo de banco de dados;
6. Projetar o experimento utilizando a mesma versão do PostgreSQL e Citus para reduzir qualquer tipo de ruído ocasionado pela diferença entre as versões do PostgreSQL;

7. Criar testes automatizados para o ambiente como um todo para agilizar o desenvolvimento. Por exemplo, verificar se os nós estão se comunicando corretamente, verificar se a tabela está distribuída corretamente entre os nós;

8. Gerar uma base de dados anonimizada de *benchmark* para documentos fiscais;

9. Utilizar outras métricas de avaliação além de tempo de resposta e alocação de *shards* por nó. Por exemplo, uso de CPU, memória RAM, internet IO, block IO, etc.

REFERÊNCIAS

BRASIL. Decreto nº 6.022, de 22 de janeiro de 2007. **Diário Oficial da República Federativa do Brasil**, Brasília, DF, 2007. Disponível em: ⟨http://www.planalto.gov.br/ccivil_03/_ato2007-2010/2007/decreto/d6022.htm⟩.

BRASIL, R. F. D. Portal da nota fiscal eletrônica: Informaçoes sobre estatísticas- quantidade de nf-e autorizadas desde a implantaçao em 2006. **Acessada em 13 de dezembro de 2022.**, 2022. Disponível em: ⟨https://www.nfe.fazenda.gov.br/portal/infoEstatisticas.aspx⟩.

CUBUKCU, U.; ERDOGAN, O.; PATHAK, S.; SANNAKKAYALA, S.; SLOT, M. Citus: Distributed postgresql for data-intensive applications. In: **Proceedings of the 2021 International Conference on Management of Data**. [S.l.: s.n.], 2021. p. 2490–2502.

DATA, C. **Citus documentation**. 2022.

DATA citus. docker-compose. v. 3, p. https://github.com/citusdata/docker/blob/master/docker–compose.yml, 2022.

GKAMAS, T.; KARAISKOS, V.; KONTOGIANNIS, S. Performance evaluation of distributed database strategies using docker as a service for industrial iot data: Application to industry 4.0. **Information**, MDPI, v. 13, n. 4, p. 190, 2022.

HUB, D. The postgresql object-relational database system provides reliability and data integrity. p. https://hub.docker.com/˙/postgres/, 2022.

MANOEL, V.; OLIVEIRA, C. G.; PEREIRA, A. L.; MATA, A. R. Escrituração contábil digital: consequências, benefícios e a evolução da profissão contábil. **Instituto de Ensino Superior de Londrina–INESUL, Londrina**, 2011.

MENDONÇA, A. d. R. B. **Avaliação experimental de uma arquitetura de microsserviços para o gerenciamento de notas fiscais eletrônicas**. Dissertação (Mestrado) — Universidade Federal de Pernambuco, 2022.

PORTAL, D. S. P. D. E. D. http://sped.rfb.gov.br/pagina/show/1328. **Acessada em 03 de novembro de 2022.**, 2022.

RAMAKRISHNAN, R.; GEHRKE, J. **Sistemas de gerenciamento de banco de dados-3**. [S.l.]: AMGH Editora, 2008.

SANTOS, D. F. P. et al. Um estudo comparativo sobre uso de modelos de dados para notas fiscais eletrônicas. Universidade Federal da Paraíba, 2021.

SANTOS, E. M. d. Uso de dados de documentos fiscais eletrônicos para o planejamento do transporte urbano de cargas. Universidade de Brasília, 2015.

SEFAZ. Manual de orienta¸c˜ao do contribuinte - padrões técnicos de comunicação versão 5.0. p. https://www.https://www.nfe.fazenda.gov.br/portal/exibirArquivo.aspx?conteudo=HO2V2012

TSUKAMOTO, V. H. S. et al. Sistema público de escrituração digital (sped)–ecf-escrituração contábil fiscal: lucro real, lucro presumido e imunes ou isentas. Universidade Federal de Mato Grosso, 2019.

APÊNDICES 1 - Recorte das propriedades da fatonfe

```
CREATE TABLE fatonfe(
    id_fatonfe BIGSERIAL NOT NULL,
    infProt_chNFe char(44) PRIMARY KEY,
    infNFe_sqn bigint NOT NULL,
    informix_stnfeletronica char(1) NOT NULL DEFAULT 'A',
    informix_dhconexao varchar NULL,
    informix_nriptransmissor varchar NULL,
    informix_nrportacon int4 NULL,
    infNFe_ide character varying,
    infNFe_ide_cUF integer,
    infNFe_ide_cNF character varying,
    infNFe_ide_natOp character varying,
    infNFe_ide_mod integer, infNFe_ide_serie character varying,
    infNFe_ide_nNF character varying,
    infNFe_ide_dhEmi timestamp with time zone,
    infNFe_ide_dhSaiEnt timestamp with time zone,
    infNFe_ide_tpNF integer,
    infNFe_ide_idDest integer,
    infNFe_ide_cMunFG character varying,
    infNFe_ide_tpImp integer,
    infNFe_ide_tpEmis integer,
    infNFe_ide_cDV integer,
    infNFe_ide_tpAmb integer,
    infNFe_ide_finNFe integer,
    infNFe_ide_indFinal integer,
    infNFe_ide_indPres integer,
    infNFe_ide_procEmi integer,
    infNFe_ide_verProc character varying,
    infNFe_emit_CNPJ char(14),
    infNFe_emit_CPF char(11),
    infNFe_emit_xNome character varying,
    infNFe_emit_xFant character varying,
    infNFe_emit_enderEmit_xLgr character varying,
    infNFe_emit_enderEmit_nro character varying,
    infNFe_emit_enderEmit_xCpl character varying,
    infNFe_emit_enderEmit_xBairro character varying,
    infNFe_emit_enderEmit_cMun character varying,
    infNFe_emit_enderEmit_xMun character varying,
    infNFe_emit_enderEmit_UF character varying,
    infNFe_emit_enderEmit_CEP character varying,
    infNFe_emit_enderEmit_cPais character varying,
    infNFe_emit_enderEmit_xPais character varying,
    infNFe_emit_enderEmit_fone character varying,
    infNFe_emit_IE character varying,
    infNFe_emit_IEST character varying,
```

```
    infNFe_emit_IM character varying,
    infNFe_emit_CNAE character varying,
    infNFe_emit_CRT character varying,
    infNFe_dest_CNPJ char(14),
    infNFe_dest_CPF char(11),
    infNFe_dest_idEstrangeiro character varying,
    infNFe_dest_xNome character varying,
    infNFe_dest_enderDest_xLgr character varying,
    infNFe_dest_enderDest_nro character varying,
    infNFe_dest_enderDest_xCpl character varying,
    infNFe_dest_enderDest_xBairro character varying,
    infNFe_dest_enderDest_cMun character varying,
    infNFe_dest_enderDest_xMun character varying,
    infNFe_dest_enderDest_UF character varying,
    infNFe_dest_enderDest_CEP character varying,
    infNFe_dest_enderDest_cPais character varying,
    infNFe_dest_enderDest_xPais character varying,
    infNFe_dest_enderDest_fone character varying,
    infNFe_dest_indIEDest character varying,
    infNFe_dest_IE character varying,
    infNFe_dest_ISUF character varying,
    infNFe_dest_IM character varying,
    infNFe_dest_email character varying,
    infNFe_total_ICMSTot_vBC numeric(16,2),
    infNFe_total_ICMSTot_vICMS numeric(16,2),
    infNFe_total_ICMSTot_vICMSDeson numeric(16,2),
    infNFe_total_ICMSTot_vFCPUFDest numeric(16,2),
    infNFe_total_ICMSTot_vICMSUFDest numeric(16,2),
    infNFe_total_ICMSTot_vICMSUFRemet numeric(16,2),
    infNFe_total_ICMSTot_vFCP numeric(16,2),
    infNFe_total_ICMSTot_vBCST numeric(16,2),
    infNFe_total_ICMSTot_vST numeric(16,2),
    infNFe_total_ICMSTot_vFCPST numeric(16,2),
    infNFe_total_ICMSTot_vFCPSTRet numeric(16,2),
    infNFe_total_ICMSTot_vProd numeric(16,2),
    infNFe_total_ICMSTot_vFrete numeric(16,2),
    infNFe_total_ICMSTot_vSeg numeric(16,2),
    infNFe_total_ICMSTot_vDesc numeric(16,2),
    infNFe_total_ICMSTot_vII numeric(16,2),
    infNFe_total_ICMSTot_vIPI numeric(16,2),
    infNFe_total_ICMSTot_vPIS numeric(16,2),
    infNFe_total_ICMSTot_vCOFINS numeric(16,2),
    infNFe_total_ICMSTot_vOutro numeric(16,2),
    infNFe_total_ICMSTot_vNF numeric(16,2),
    infNFe_total_ICMSTot_vTotTrib numeric(16,2),
    infNFe_transp_modfrete character varying(1),
    infNFe_transp_vagao character varying(20),
    infNFe_transp_balsa character varying(20),
```

```
    infNFe_transp_transporta_cnpj char(14),
    infNFe_transp_transporta_cpf char(11),
    infNFe_transp_transporta_xnome character varying(100),
    infNFe_transp_transporta_ie character varying,
    infNFe_transp_transporta_xender character varying(100),
    infNFe_transp_transporta_xmun character varying(100),
    infNFe_transp_transporta_uf character varying(2),
    infNFe_transp_rettransp_vserv numeric(16,2),
    infNFe_transp_rettransp_vbcret numeric(16,2),
    infNFe_transp_rettransp_picmsret numeric(16,2),
    infNFe_transp_rettransp_vicmsret numeric(16,2),
    infNFe_transp_rettransp_cfop character varying,
    infNFe_transp_rettransp_cmunfg character varying,
    infNFe_transp_veictransp_placa character varying(8),
    infNFe_transp_veictransp_uf character varying(2),
    infNFe_transp_reboque1_placa character varying(8),
    infNFe_transp_reboque1_uf character varying(2),
    infNFe_transp_reboque2_placa character varying(8),
    infNFe_transp_reboque2_uf character varying(2),
    infNFe_transp_reboque3_placa character varying(8),
    infNFe_transp_reboque3_uf character varying(2),
    infNFe_transp_reboque4_placa character varying(8),
    infNFe_transp_reboque4_uf character varying(2),
    infNFe_transp_reboque5_placa character varying(8),
    infNFe_transp_reboque5_uf character varying(2),
    infNFe_transp_vol1_nVol character varying(60),
    infNFe_transp_vol1_qVol numeric(16,2),
    infNFe_transp_vol1_esp character varying(60),
    infNFe_transp_vol1_marca character varying(60),
    infNFe_transp_vol1_pesoL numeric(16,2),
    infNFe_transp_vol1_pesoB numeric(16,2),
    infNFe_transp_vol1_lacres_nLacre character varying(60),
    infNFe_cobr_fat_nFat character varying(60),
    infNFe_cobr_fat_vOrig numeric(16,2),
    infNFe_cobr_fat_vDesc numeric(16,2),
    infNFe_cobr_fat_vLiq numeric(16,2),
    infNFe_pag_vTroco numeric(16,2),
    infNFe_infAdic_infAdFisco character varying,
    infNFe_infAdic_infCpl character varying,
    infNFe_infAdic_obsCont_xCampo character varying,
    infNFe_infAdic_obsCont_xTexto character varying
);
```

APÊNDICES 2 - Adaptação do Docker para o PostgreSQL

```
version: "3.7"
services:
  manager:
    image: postgres
    restart: always
    ports:
      - "5430:5432"
    volumes:
      - "/home/debian/controle/pgdata:/var/lib/postgresql/data"
      - "/home/debian/amostra:/amostra"
    environment:
      POSTGRES_PASSWORD: "Aiveid7n"
      TZ: "America/Recife"
    deploy:
      placement:
        constraints:
          - "node.hostname == sefaz-shard0"

```

APÊNDICES 3 - Adaptação do Docker para o Citus

```
version: "3.7"

networks:
  postgis_dist:
    driver: overlay

volumes:
  healthcheck-volume:

services:
  master:
    container_name: "${COMPOSE_PROJECT_NAME:-citus}_master"
    image: "citusdata/citus:11.1.2"
    ports: ["${COORDINATOR_EXTERNAL_PORT:-5432}:5432"]
    labels: ["com.citusdata.role=Master", "com.docker.compose.project=citus"]
    shm_size: '1gb'
    environment: &AUTH
      POSTGRES_USER: "datalakeuser"
      POSTGRES_PASSWORD: "EQuohG2i"
      PGUSER: "datalakeuser"
      PGPASSWORD: "EQuohG2i"
    deploy:
        placement:
          constraints:
            - "node.hostname == sefaz-shard0"
    networks:
      - postgis_dist

  manager:
    container_name: "${COMPOSE_PROJECT_NAME:-citus}_manager"
    image: "citusdata/membership-manager:0.3.0"
    labels: ["com.citusdata.role=Manager", "com.docker.compose.project=citus"]
    volumes:
      - "${DOCKER_SOCK:-/var/run/docker.sock}:/var/run/docker.sock"
      - healthcheck-volume:/healthcheck
    depends_on: [master]
    environment: *AUTH
    deploy:
        placement:
          constraints:
            - "node.hostname == sefaz-shard0"
    networks:
      - postgis_dist

  worker1:
```

```
    container_name: "${COMPOSE_PROJECT_NAME:-citus}_worker1"
    image: "citusdata/citus:11.1.2"
    labels: ["com.citusdata.role=Worker", "com.docker.compose.project=citus"]
    shm_size: '1gb'
    depends_on: [manager]
    environment: *AUTH
    ports:
      - "5481:5432"
    volumes:
      - healthcheck-volume:/healthcheck
      - "/home/debian/citus:/var/lib/postgresql/data"
    deploy:
        placement:
          constraints:
            - "node.hostname == sefaz-shard1"
    networks:
      - postgis_dist

  worker2:
    container_name: "${COMPOSE_PROJECT_NAME:-citus}_worker2"
    image: "citusdata/citus:11.1.2"
    labels: ["com.citusdata.role=Worker", "com.docker.compose.project=citus"]
    shm_size: '1gb'
    depends_on: [manager]
    environment: *AUTH
    ports:
      - "5482:5432"
    volumes:
      - healthcheck-volume:/healthcheck
      - "/home/debian/citus:/var/lib/postgresql/data"
    deploy:
        placement:
          constraints:
            - "node.hostname == sefaz-shard2"
    networks:
      - postgis_dist

  worker3:
    container_name: "${COMPOSE_PROJECT_NAME:-citus}_worker3"
    image: "citusdata/citus:11.1.2"
    labels: ["com.citusdata.role=Worker", "com.docker.compose.project=citus"]
    shm_size: '1gb'
    depends_on: [manager]
    environment: *AUTH
    ports:
      - "5483:5432"
    volumes:
      - healthcheck-volume:/healthcheck
```

```
      - "/home/debian/citus:/var/lib/postgresql/data"
    deploy:
        placement:
          constraints:
            - "node.hostname == sefaz-shard3"
    networks:
      - postgis_dist
```

APÊNDICES 4 - Código para realização das Consultas. Referente a amostra pequena e grande, respectivamente

```
SELECT * FROM fatonfe
SELECT * FROM fatonfe WHERE infNFe_emit_CNPJ = '04116842000164'
SELECT * FROM fatonfe WHERE infNFe_emit_CNPJ LIKE '0411%'
SELECT * FROM fatonfe WHERE infNFe_emit_CNPJ IN ('04111338000171', '04111497000176', '04112204000175', '04114549000168', '04114868000173', '04115363000123', '04115428000300', '04116210000109', '04116842000164', '04117715000180')
SELECT * FROM fatonfe WHERE infNFe_ide_dhEmi = '2022-02-03 08:49:26-03'
SELECT * FROM fatonfe WHERE infNFe_ide_dhEmi = '2022-02-03 08:49:26-03' AND infNFe_emit_CNPJ = '04116842000164'
SELECT * FROM fatonfe WHERE infNFe_ide_dhEmi = '2022-02-03 08:49:26-03' AND infNFe_emit_CNPJ LIKE '0411%'
SELECT * FROM fatonfe WHERE infNFe_ide_dhEmi = '2022-02-03 08:49:26-03' AND infNFe_emit_CNPJ IN ('04111338000171', '04111497000176', '04112204000175', '04114549000168', '04114868000173', '04115363000123', '04115428000300', '04116210000109', '04116842000164', '04117715000180')
SELECT * FROM fatonfe WHERE infNFe_ide_dhEmi BETWEEN '2022-01-01' AND '2022-06-01'
SELECT * FROM fatonfe WHERE infNFe_emit_CNPJ = '04116842000164' AND infNFe_ide_dhEmi BETWEEN '2022-01-01' AND '2022-06-01'
SELECT * FROM fatonfe WHERE infNFe_emit_CNPJ LIKE '0411%' AND infNFe_ide_dhEmi BETWEEN '2022-01-01' AND '2022-06-01'
SELECT * FROM fatonfe WHERE infNFe_emit_CNPJ IN ('04111338000171', '04111497000176', '04112204000175', '04114549000168', '04114868000173', '04115363000123', '04115428000300', '04116210000109', '04116842000164', '04117715000180') AND infNFe_ide_dhEmi BETWEEN '2022-01-01' AND '2022-06-01'
SELECT id_fatonfe FROM fatonfe
SELECT id_fatonfe FROM fatonfe WHERE infNFe_emit_CNPJ = '04116842000164'
SELECT id_fatonfe FROM fatonfe WHERE infNFe_emit_CNPJ LIKE '0411%'
SELECT id_fatonfe FROM fatonfe WHERE infNFe_emit_CNPJ IN ('04111338000171', '04111497000176', '04112204000175', '04114549000168', '04114868000173', '04115363000123', '04115428000300', '04116210000109', '04116842000164', '04117715000180')
SELECT id_fatonfe FROM fatonfe WHERE infNFe_ide_dhEmi = '2022-02-03 08:49:26-03'
SELECT id_fatonfe FROM fatonfe WHERE infNFe_ide_dhEmi = '2022-02-03 08:49:26-03' AND infNFe_emit_CNPJ = '04116842000164'
SELECT id_fatonfe FROM fatonfe WHERE infNFe_ide_dhEmi = '2022-02-03 08:49:26-03' AND infNFe_emit_CNPJ LIKE '0411%'
SELECT id_fatonfe FROM fatonfe WHERE infNFe_ide_dhEmi = '2022-02-03 08:49:26-03' AND infNFe_emit_CNPJ IN ('04111338000171', '04111497000176', '04112204000175', '04114549000168', '04114868000173', '04115363000123', '04115428000300', '04116210000109', '04116842000164', '04117715000180')
SELECT id_fatonfe FROM fatonfe WHERE infNFe_ide_dhEmi BETWEEN '2022-01-01' AND '2022-06-01'
SELECT id_fatonfe FROM fatonfe WHERE infNFe_emit_CNPJ = '04116842000164' AND infNFe_ide_dhEmi BETWEEN '2022-01-01' AND '2022-06-01'
```

```
SELECT id_fatonfe FROM fatonfe WHERE infNFe_emit_CNPJ LIKE '0411%' AND infNFe_ide_dhEmi BETWEEN '2022-01-01' AND '2022-06-01'
SELECT id_fatonfe FROM fatonfe WHERE infNFe_emit_CNPJ IN ('04111338000171', '04111497000176', '04112204000175', '04114549000168', '04114868000173', '04115363000123', '04115428000300', '04116210000109', '04116842000164', '04117715000180') AND infNFe_ide_dhEmi BETWEEN '2022-01-01' AND '2022-06-01'
SELECT sum(id_fatonfe) FROM fatonfe
SELECT sum(id_fatonfe) FROM fatonfe WHERE infNFe_emit_CNPJ = '04116842000164'
SELECT sum(id_fatonfe) FROM fatonfe WHERE infNFe_emit_CNPJ LIKE '0411%'
SELECT sum(id_fatonfe) FROM fatonfe WHERE infNFe_emit_CNPJ IN ('04111338000171', '04111497000176', '04112204000175', '04114549000168', '04114868000173', '04115363000123', '04115428000300', '04116210000109', '04116842000164', '04117715000180')
SELECT sum(id_fatonfe) FROM fatonfe WHERE infNFe_ide_dhEmi = '2022-02-03 08:49:26-03'
SELECT sum(id_fatonfe) FROM fatonfe WHERE infNFe_ide_dhEmi = '2022-02-03 08:49:26-03' AND infNFe_emit_CNPJ = '04116842000164'
SELECT sum(id_fatonfe) FROM fatonfe WHERE infNFe_ide_dhEmi = '2022-02-03 08:49:26-03' AND infNFe_emit_CNPJ LIKE '0411%'
SELECT sum(id_fatonfe) FROM fatonfe WHERE infNFe_ide_dhEmi = '2022-02-03 08:49:26-03' AND infNFe_emit_CNPJ IN ('04111338000171', '04111497000176', '04112204000175', '04114549000168', '04114868000173', '04115363000123', '04115428000300', '04116210000109', '04116842000164', '04117715000180')
SELECT sum(id_fatonfe) FROM fatonfe WHERE infNFe_ide_dhEmi BETWEEN '2022-01-01' AND '2022-06-01'
SELECT sum(id_fatonfe) FROM fatonfe WHERE infNFe_emit_CNPJ = '04116842000164' AND infNFe_ide_dhEmi BETWEEN '2022-01-01' AND '2022-06-01'
SELECT sum(id_fatonfe) FROM fatonfe WHERE infNFe_emit_CNPJ LIKE '0411%' AND infNFe_ide_dhEmi BETWEEN '2022-01-01' AND '2022-06-01'
SELECT sum(id_fatonfe) FROM fatonfe WHERE infNFe_emit_CNPJ IN ('04111338000171', '04111497000176', '04112204000175', '04114549000168', '04114868000173', '04115363000123', '04115428000300', '04116210000109', '04116842000164', '04117715000180') AND infNFe_ide_dhEmi BETWEEN '2022-01-01' AND '2022-06-01'
SELECT avg(id_fatonfe) FROM fatonfe
SELECT avg(id_fatonfe) FROM fatonfe WHERE infNFe_emit_CNPJ = '04116842000164'
SELECT avg(id_fatonfe) FROM fatonfe WHERE infNFe_emit_CNPJ LIKE '0411%'
SELECT avg(id_fatonfe) FROM fatonfe WHERE infNFe_emit_CNPJ IN ('04111338000171', '04111497000176', '04112204000175', '04114549000168', '04114868000173', '04115363000123', '04115428000300', '04116210000109', '04116842000164', '04117715000180')
SELECT avg(id_fatonfe) FROM fatonfe WHERE infNFe_ide_dhEmi = '2022-02-03 08:49:26-03'
SELECT avg(id_fatonfe) FROM fatonfe WHERE infNFe_ide_dhEmi = '2022-02-03 08:49:26-03' AND infNFe_emit_CNPJ = '04116842000164'
SELECT avg(id_fatonfe) FROM fatonfe WHERE infNFe_ide_dhEmi = '2022-02-03 08:49:26-03' AND infNFe_emit_CNPJ LIKE '0411%'
SELECT avg(id_fatonfe) FROM fatonfe WHERE infNFe_ide_dhEmi = '2022-02-03 08:49:26-03' AND infNFe_emit_CNPJ IN ('04111338000171', '04111497000176', '04112204000175', '04114549000168', '04114868000173', '04115363000123', '04115428000300', '04116210000109', '04116842000164', '04117715000180')
```

```
SELECT avg(id_fatonfe) FROM fatonfe WHERE infNFe_ide_dhEmi BETWEEN '2022-01-01' AND '2022-06-01'
SELECT avg(id_fatonfe) FROM fatonfe WHERE infNFe_emit_CNPJ = '04116842000164' AND infNFe_ide_dhEmi BETWEEN '2022-01-01' AND '2022-06-01'
SELECT avg(id_fatonfe) FROM fatonfe WHERE infNFe_emit_CNPJ LIKE '0411%' AND infNFe_ide_dhEmi BETWEEN '2022-01-01' AND '2022-06-01'
SELECT avg(id_fatonfe) FROM fatonfe WHERE infNFe_emit_CNPJ IN ('04111338000171', '04111497000176', '04112204000175', '04114549000168', '04114868000173', '04115363000123', '04115428000300', '04116210000109', '04116842000164', '04117715000180') AND infNFe_ide_dhEmi BETWEEN '2022-01-01' AND '2022-06-01'
SELECT count(id_fatonfe) FROM fatonfe
SELECT count(id_fatonfe) FROM fatonfe WHERE infNFe_emit_CNPJ = '04116842000164'
SELECT count(id_fatonfe) FROM fatonfe WHERE infNFe_emit_CNPJ LIKE '0411%'
SELECT count(id_fatonfe) FROM fatonfe WHERE infNFe_emit_CNPJ IN ('04111338000171', '04111497000176', '04112204000175', '04114549000168', '04114868000173', '04115363000123', '04115428000300', '04116210000109', '04116842000164', '04117715000180')
SELECT count(id_fatonfe) FROM fatonfe WHERE infNFe_ide_dhEmi = '2022-02-03 08:49:26-03'
SELECT count(id_fatonfe) FROM fatonfe WHERE infNFe_ide_dhEmi = '2022-02-03 08:49:26-03' AND infNFe_emit_CNPJ = '04116842000164'
SELECT count(id_fatonfe) FROM fatonfe WHERE infNFe_ide_dhEmi = '2022-02-03 08:49:26-03' AND infNFe_emit_CNPJ LIKE '0411%'
SELECT count(id_fatonfe) FROM fatonfe WHERE infNFe_ide_dhEmi = '2022-02-03 08:49:26-03' AND infNFe_emit_CNPJ IN ('04111338000171', '04111497000176', '04112204000175', '04114549000168', '04114868000173', '04115363000123', '04115428000300', '04116210000109', '04116842000164', '04117715000180')
SELECT count(id_fatonfe) FROM fatonfe WHERE infNFe_ide_dhEmi BETWEEN '2022-01-01' AND '2022-06-01'
SELECT count(id_fatonfe) FROM fatonfe WHERE infNFe_emit_CNPJ = '04116842000164' AND infNFe_ide_dhEmi BETWEEN '2022-01-01' AND '2022-06-01'
SELECT count(id_fatonfe) FROM fatonfe WHERE infNFe_emit_CNPJ LIKE '0411%' AND infNFe_ide_dhEmi BETWEEN '2022-01-01' AND '2022-06-01'
SELECT count(id_fatonfe) FROM fatonfe WHERE infNFe_emit_CNPJ IN ('04111338000171', '04111497000176', '04112204000175', '04114549000168', '04114868000173', '04115363000123', '04115428000300', '04116210000109', '04116842000164', '04117715000180') AND infNFe_ide_dhEmi BETWEEN '2022-01-01' AND '2022-06-01'
```

```
SELECT * FROM fatonfe WHERE infNFe_emit_CNPJ = '04116842000164';
SELECT * FROM fatonfe WHERE infNFe_emit_CNPJ LIKE '04116%';
SELECT * FROM fatonfe WHERE infNFe_emit_CNPJ IN ('04116159000127', '04116210000109', '04116475000107', '04116842000164');
SELECT * FROM fatonfe WHERE infNFe_ide_dhEmi = '2021-01-05 11:01:23.000 -0300';
SELECT * FROM fatonfe WHERE infNFe_ide_dhEmi = '2021-01-05 11:01:23.000 -0300' AND infNFe_emit_CNPJ = '04116842000164';
SELECT * FROM fatonfe WHERE infNFe_ide_dhEmi = '2021-01-05 11:01:23.000 -0300' AND infNFe_emit_CNPJ LIKE '04116%';
```

```
7  SELECT * FROM fatonfe WHERE infNFe_ide_dhEmi = '2021-01-05 11:01:23.000 -0300' AND
   ↪  infNFe_emit_CNPJ IN ('04116159000127', '04116210000109', '04116475000107',
   ↪  '04116842000164');
8  SELECT * FROM fatonfe WHERE infNFe_ide_dhEmi BETWEEN '2022-03-01 17:06:27-03' AND
   ↪  '2022-04-04 10:15:00.000 -0300';
9  SELECT * FROM fatonfe WHERE infNFe_ide_dhEmi BETWEEN '2022-03-01 17:06:27-03' AND
   ↪  '2022-03-02 22:06:26.000 -0300';
10 SELECT * FROM fatonfe WHERE infNFe_emit_CNPJ = '04116842000164' AND infNFe_ide_dhEmi
   ↪  BETWEEN '2022-03-01 17:06:27-03' AND '2022-04-04 10:15:00.000 -0300';
11 SELECT * FROM fatonfe WHERE infNFe_emit_CNPJ = '04116842000164' AND infNFe_ide_dhEmi
   ↪  BETWEEN '2022-03-01 17:06:27-03' AND '2022-03-02 22:06:26.000 -0300';
12 SELECT * FROM fatonfe WHERE infNFe_emit_CNPJ LIKE '04116%' AND infNFe_ide_dhEmi
   ↪  BETWEEN '2022-03-01 17:06:27-03' AND '2022-04-04 10:15:00.000 -0300';
13 SELECT * FROM fatonfe WHERE infNFe_emit_CNPJ LIKE '04116%' AND infNFe_ide_dhEmi
   ↪  BETWEEN '2022-03-01 17:06:27-03' AND '2022-03-02 22:06:26.000 -0300';
14 SELECT * FROM fatonfe WHERE infNFe_emit_CNPJ IN ('04116159000127', '04116210000109',
   ↪  '04116475000107', '04116842000164') AND infNFe_ide_dhEmi BETWEEN '2022-03-01
   ↪  17:06:27-03' AND '2022-04-04 10:15:00.000 -0300';
15 SELECT * FROM fatonfe WHERE infNFe_emit_CNPJ IN ('04116159000127', '04116210000109',
   ↪  '04116475000107', '04116842000164') AND infNFe_ide_dhEmi BETWEEN '2022-03-01
   ↪  17:06:27-03' AND '2022-03-02 22:06:26.000 -0300';
16 SELECT id_fatonfe FROM fatonfe;
17 SELECT id_fatonfe FROM fatonfe WHERE infNFe_emit_CNPJ = '04116842000164';
18 SELECT id_fatonfe FROM fatonfe WHERE infNFe_emit_CNPJ LIKE '04116%';
19 SELECT id_fatonfe FROM fatonfe WHERE infNFe_emit_CNPJ IN ('04116159000127',
   ↪  '04116210000109', '04116475000107', '04116842000164');
20 SELECT id_fatonfe FROM fatonfe WHERE infNFe_ide_dhEmi = '2021-01-05 11:01:23.000
   ↪  -0300';
21 SELECT id_fatonfe FROM fatonfe WHERE infNFe_ide_dhEmi = '2021-01-05 11:01:23.000
   ↪  -0300' AND infNFe_emit_CNPJ = '04116842000164';
22 SELECT id_fatonfe FROM fatonfe WHERE infNFe_ide_dhEmi = '2021-01-05 11:01:23.000
   ↪  -0300' AND infNFe_emit_CNPJ LIKE '04116%';
23 SELECT id_fatonfe FROM fatonfe WHERE infNFe_ide_dhEmi = '2021-01-05 11:01:23.000
   ↪  -0300' AND infNFe_emit_CNPJ IN ('04116159000127', '04116210000109',
   ↪  '04116475000107', '04116842000164');
24 SELECT id_fatonfe FROM fatonfe WHERE infNFe_ide_dhEmi BETWEEN '2022-03-01
   ↪  17:06:27-03' AND '2022-04-04 10:15:00.000 -0300';
25 SELECT id_fatonfe FROM fatonfe WHERE infNFe_ide_dhEmi BETWEEN '2022-03-01
   ↪  17:06:27-03' AND '2022-03-02 22:06:26.000 -0300';
26 SELECT id_fatonfe FROM fatonfe WHERE infNFe_emit_CNPJ = '04116842000164' AND
   ↪  infNFe_ide_dhEmi BETWEEN '2022-03-01 17:06:27-03' AND '2022-04-04 10:15:00.000
   ↪  -0300';
27 SELECT id_fatonfe FROM fatonfe WHERE infNFe_emit_CNPJ = '04116842000164' AND
   ↪  infNFe_ide_dhEmi BETWEEN '2022-03-01 17:06:27-03' AND '2022-03-02 22:06:26.000
   ↪  -0300';
28 SELECT id_fatonfe FROM fatonfe WHERE infNFe_emit_CNPJ LIKE '04116%' AND
   ↪  infNFe_ide_dhEmi BETWEEN '2022-03-01 17:06:27-03' AND '2022-04-04 10:15:00.000
   ↪  -0300';
```

```
29 SELECT id_fatonfe FROM fatonfe WHERE infNFe_emit_CNPJ LIKE '04116%' AND
   ↪  infNFe_ide_dhEmi BETWEEN '2022-03-01 17:06:27-03' AND '2022-03-02 22:06:26.000
   ↪ -0300';
30 SELECT id_fatonfe FROM fatonfe WHERE infNFe_emit_CNPJ IN ('04116159000127',
   ↪  '04116210000109', '04116475000107', '04116842000164') AND infNFe_ide_dhEmi
   ↪  BETWEEN '2022-03-01 17:06:27-03' AND '2022-04-04 10:15:00.000 -0300';
31 SELECT id_fatonfe FROM fatonfe WHERE infNFe_emit_CNPJ IN ('04116159000127',
   ↪  '04116210000109', '04116475000107', '04116842000164') AND infNFe_ide_dhEmi
   ↪  BETWEEN '2022-03-01 17:06:27-03' AND '2022-03-02 22:06:26.000 -0300';
32 SELECT sum(id_fatonfe) FROM fatonfe;
33 SELECT sum(id_fatonfe) FROM fatonfe WHERE infNFe_emit_CNPJ = '04116842000164';
34 SELECT sum(id_fatonfe) FROM fatonfe WHERE infNFe_emit_CNPJ LIKE '04116%';
35 SELECT sum(id_fatonfe) FROM fatonfe WHERE infNFe_emit_CNPJ IN ('04116159000127',
   ↪  '04116210000109', '04116475000107', '04116842000164');
36 SELECT sum(id_fatonfe) FROM fatonfe WHERE infNFe_ide_dhEmi = '2021-01-05 11:01:23.000
   ↪ -0300';
37 SELECT sum(id_fatonfe) FROM fatonfe WHERE infNFe_ide_dhEmi = '2021-01-05 11:01:23.000
   ↪ -0300' AND infNFe_emit_CNPJ = '04116842000164';
38 SELECT sum(id_fatonfe) FROM fatonfe WHERE infNFe_ide_dhEmi = '2021-01-05 11:01:23.000
   ↪ -0300' AND infNFe_emit_CNPJ LIKE '04116%';
39 SELECT sum(id_fatonfe) FROM fatonfe WHERE infNFe_ide_dhEmi = '2021-01-05 11:01:23.000
   ↪ -0300' AND infNFe_emit_CNPJ IN ('04116159000127', '04116210000109',
   ↪  '04116475000107', '04116842000164');
40 SELECT sum(id_fatonfe) FROM fatonfe WHERE infNFe_ide_dhEmi BETWEEN '2022-03-01
   ↪  17:06:27-03' AND '2022-04-04 10:15:00.000 -0300';
41 SELECT sum(id_fatonfe) FROM fatonfe WHERE infNFe_ide_dhEmi BETWEEN '2022-03-01
   ↪  17:06:27-03' AND '2022-03-02 22:06:26.000 -0300';
42 SELECT sum(id_fatonfe) FROM fatonfe WHERE infNFe_emit_CNPJ = '04116842000164' AND
   ↪  infNFe_ide_dhEmi BETWEEN '2022-03-01 17:06:27-03' AND '2022-04-04 10:15:00.000
   ↪ -0300';
43 SELECT sum(id_fatonfe) FROM fatonfe WHERE infNFe_emit_CNPJ = '04116842000164' AND
   ↪  infNFe_ide_dhEmi BETWEEN '2022-03-01 17:06:27-03' AND '2022-03-02 22:06:26.000
   ↪ -0300';
44 SELECT sum(id_fatonfe) FROM fatonfe WHERE infNFe_emit_CNPJ LIKE '04116%' AND
   ↪  infNFe_ide_dhEmi BETWEEN '2022-03-01 17:06:27-03' AND '2022-04-04 10:15:00.000
   ↪ -0300';
45 SELECT sum(id_fatonfe) FROM fatonfe WHERE infNFe_emit_CNPJ LIKE '04116%' AND
   ↪  infNFe_ide_dhEmi BETWEEN '2022-03-01 17:06:27-03' AND '2022-03-02 22:06:26.000
   ↪ -0300';
46 SELECT sum(id_fatonfe) FROM fatonfe WHERE infNFe_emit_CNPJ IN ('04116159000127',
   ↪  '04116210000109', '04116475000107', '04116842000164') AND infNFe_ide_dhEmi
   ↪  BETWEEN '2022-03-01 17:06:27-03' AND '2022-04-04 10:15:00.000 -0300';
47 SELECT sum(id_fatonfe) FROM fatonfe WHERE infNFe_emit_CNPJ IN ('04116159000127',
   ↪  '04116210000109', '04116475000107', '04116842000164') AND infNFe_ide_dhEmi
   ↪  BETWEEN '2022-03-01 17:06:27-03' AND '2022-03-02 22:06:26.000 -0300';
48 SELECT avg(id_fatonfe) FROM fatonfe;
49 SELECT avg(id_fatonfe) FROM fatonfe WHERE infNFe_emit_CNPJ = '04116842000164';
50 SELECT avg(id_fatonfe) FROM fatonfe WHERE infNFe_emit_CNPJ LIKE '04116%';
```

```
51 SELECT avg(id_fatonfe) FROM fatonfe WHERE infNFe_emit_CNPJ IN ('04116159000127',
   ↪  '04116210000109', '04116475000107', '04116842000164');
52 SELECT avg(id_fatonfe) FROM fatonfe WHERE infNFe_ide_dhEmi = '2021-01-05 11:01:23.000
   ↪ -0300';
53 SELECT avg(id_fatonfe) FROM fatonfe WHERE infNFe_ide_dhEmi = '2021-01-05 11:01:23.000
   ↪ -0300' AND infNFe_emit_CNPJ = '04116842000164';
54 SELECT avg(id_fatonfe) FROM fatonfe WHERE infNFe_ide_dhEmi = '2021-01-05 11:01:23.000
   ↪ -0300' AND infNFe_emit_CNPJ LIKE '04116%';
55 SELECT avg(id_fatonfe) FROM fatonfe WHERE infNFe_ide_dhEmi = '2021-01-05 11:01:23.000
   ↪ -0300' AND infNFe_emit_CNPJ IN ('04116159000127', '04116210000109',
   ↪  '04116475000107', '04116842000164');
56 SELECT avg(id_fatonfe) FROM fatonfe WHERE infNFe_ide_dhEmi BETWEEN '2022-03-01
   ↪  17:06:27-03' AND '2022-04-04 10:15:00.000 -0300';
57 SELECT avg(id_fatonfe) FROM fatonfe WHERE infNFe_ide_dhEmi BETWEEN '2022-03-01
   ↪  17:06:27-03' AND '2022-03-02 22:06:26.000 -0300';
58 SELECT avg(id_fatonfe) FROM fatonfe WHERE infNFe_emit_CNPJ = '04116842000164' AND
   ↪  infNFe_ide_dhEmi BETWEEN '2022-03-01 17:06:27-03' AND '2022-04-04 10:15:00.000
   ↪ -0300';
59 SELECT avg(id_fatonfe) FROM fatonfe WHERE infNFe_emit_CNPJ = '04116842000164' AND
   ↪  infNFe_ide_dhEmi BETWEEN '2022-03-01 17:06:27-03' AND '2022-03-02 22:06:26.000
   ↪ -0300';
60 SELECT avg(id_fatonfe) FROM fatonfe WHERE infNFe_emit_CNPJ LIKE '04116%' AND
   ↪  infNFe_ide_dhEmi BETWEEN '2022-03-01 17:06:27-03' AND '2022-04-04 10:15:00.000
   ↪ -0300';
61 SELECT avg(id_fatonfe) FROM fatonfe WHERE infNFe_emit_CNPJ LIKE '04116%' AND
   ↪  infNFe_ide_dhEmi BETWEEN '2022-03-01 17:06:27-03' AND '2022-03-02 22:06:26.000
   ↪ -0300';
62 SELECT avg(id_fatonfe) FROM fatonfe WHERE infNFe_emit_CNPJ IN ('04116159000127',
   ↪  '04116210000109', '04116475000107', '04116842000164') AND infNFe_ide_dhEmi
   ↪  BETWEEN '2022-03-01 17:06:27-03' AND '2022-04-04 10:15:00.000 -0300';
63 SELECT avg(id_fatonfe) FROM fatonfe WHERE infNFe_emit_CNPJ IN ('04116159000127',
   ↪  '04116210000109', '04116475000107', '04116842000164') AND infNFe_ide_dhEmi
   ↪  BETWEEN '2022-03-01 17:06:27-03' AND '2022-03-02 22:06:26.000 -0300';
64 SELECT count(id_fatonfe) FROM fatonfe;
65 SELECT count(id_fatonfe) FROM fatonfe WHERE infNFe_emit_CNPJ = '04116842000164';
66 SELECT count(id_fatonfe) FROM fatonfe WHERE infNFe_emit_CNPJ LIKE '04116%';
67 SELECT count(id_fatonfe) FROM fatonfe WHERE infNFe_emit_CNPJ IN ('04116159000127',
   ↪  '04116210000109', '04116475000107', '04116842000164');
68 SELECT count(id_fatonfe) FROM fatonfe WHERE infNFe_ide_dhEmi = '2021-01-05
   ↪  11:01:23.000 -0300';
69 SELECT count(id_fatonfe) FROM fatonfe WHERE infNFe_ide_dhEmi = '2021-01-05
   ↪  11:01:23.000 -0300' AND infNFe_emit_CNPJ = '04116842000164';
70 SELECT count(id_fatonfe) FROM fatonfe WHERE infNFe_ide_dhEmi = '2021-01-05
   ↪  11:01:23.000 -0300' AND infNFe_emit_CNPJ LIKE '04116%';
71 SELECT count(id_fatonfe) FROM fatonfe WHERE infNFe_ide_dhEmi = '2021-01-05
   ↪  11:01:23.000 -0300' AND infNFe_emit_CNPJ IN ('04116159000127', '04116210000109',
   ↪  '04116475000107', '04116842000164');
72 SELECT count(id_fatonfe) FROM fatonfe WHERE infNFe_ide_dhEmi BETWEEN '2022-03-01
   ↪  17:06:27-03' AND '2022-04-04 10:15:00.000 -0300';
```

```
73 SELECT count(id_fatonfe) FROM fatonfe WHERE infNFe_ide_dhEmi BETWEEN '2022-03-01
   ↪  17:06:27-03' AND '2022-03-02 22:06:26.000 -0300';
74 SELECT count(id_fatonfe) FROM fatonfe WHERE infNFe_emit_CNPJ = '04116842000164' AND
   ↪  infNFe_ide_dhEmi BETWEEN '2022-03-01 17:06:27-03' AND '2022-04-04 10:15:00.000
   ↪  -0300';
75 SELECT count(id_fatonfe) FROM fatonfe WHERE infNFe_emit_CNPJ = '04116842000164' AND
   ↪  infNFe_ide_dhEmi BETWEEN '2022-03-01 17:06:27-03' AND '2022-03-02 22:06:26.000
   ↪  -0300';
76 SELECT count(id_fatonfe) FROM fatonfe WHERE infNFe_emit_CNPJ LIKE '04116%' AND
   ↪  infNFe_ide_dhEmi BETWEEN '2022-03-01 17:06:27-03' AND '2022-04-04 10:15:00.000
   ↪  -0300';
77 SELECT count(id_fatonfe) FROM fatonfe WHERE infNFe_emit_CNPJ LIKE '04116%' AND
   ↪  infNFe_ide_dhEmi BETWEEN '2022-03-01 17:06:27-03' AND '2022-03-02 22:06:26.000
   ↪  -0300';
78 SELECT count(id_fatonfe) FROM fatonfe WHERE infNFe_emit_CNPJ IN ('04116159000127',
   ↪  '04116210000109', '04116475000107', '04116842000164') AND infNFe_ide_dhEmi
   ↪  BETWEEN '2022-03-01 17:06:27-03' AND '2022-04-04 10:15:00.000 -0300';
79 SELECT count(id_fatonfe) FROM fatonfe WHERE infNFe_emit_CNPJ IN ('04116159000127',
   ↪  '04116210000109', '04116475000107', '04116842000164') AND infNFe_ide_dhEmi
   ↪  BETWEEN '2022-03-01 17:06:27-03' AND '2022-03-02 22:06:26.000 -0300';
```

APÊNDICES 5 - Estatísticas descritiva com índice *btree* com *queries* de agregação e tamanho amostral de 850 mil notas

Tabela 16: Estatísticas descritivas para o tempo de execução, em milissegundos (ms), considerando o Citus com 3 *workers* e índice *btree* para amostra *small*.

	execution_time_1	execution_time_2	execution_time_3	execution_time_mean	execution_time_std
Count	36.0	36.0	36.0	36.0	36.0
Mean	202.443	186.778	198.913	196.045	23.293
Std	71.153	67.857	77.706	66.195	28.131
Min	132.886	124.195	122.963	153.383	0.451
Q_1	168.075	156.004	164.408	164.838	11.350
Q_2	179.285	176.2415	179.231	172.154	15.420
Q_3	196.829	179.908	183.257	185.383	25.612
Max	420.962	434.538	494.903	404.886	159.743

Tabela 17: Estatísticas descritivas para o tempo de execução, em milissegundos (ms), considerando o Citus com 6 *workers* e índice *btree* para amostra *small*.

	execution_time_1	execution_time_2	execution_time_3	execution_time_mean	execution_time_std
Count	36.0	36.0	36.0	36.0	36.0
Mean	199.176	197.061	197.859	198.032	17.214
Std	33.242	34.874	28.800	27.675	11.083
Min	169.954	157.836	164.896	169.332	0.274
Q_1	177.705	177.09	180.519	182.339	8.786
Q_2	187.340	182.181	187.994	192.470	16.198
Q_3	206.636	208.686	214.256	197.902	23.294
Max	307.082	306.564	306.669	306.772	52.249

Tabela 18: Estatísticas descritivas para o tempo de execução, em milissegundos (ms), para índice *btree* e amostra *small*, considerando o Postgres com 1PC.

	execution_time_1	execution_time_2	execution_time_3	execution_time_mean	execution_time_std
Count	36.0	36.0	36.0	36.0	36.0
Mean	7971.447	7948.838	8012.520	7977.602	53.353
Mtd	26765.904	26697.234	26911.732	26791.029	240.760
Min	2.673	3.057	2.594	2.775	0.123
Q_1	3.304	3.823	3.614	3.709	0.248
Q_2	4.222	4.042	4.324	3.994	0.506
Q_3	7.878	7.414	7.434	7.896	0.943
Max	96429.567	95826.247	96957.504	95988.269	1405.969

Tabela 19: Estatísticas descritivas para o tempo de execução, em milissegundos (ms), para índice *btree* e amostra *small*, considerando o Postgres com 2PC.

	execution_time_1	execution_time_2	execution_time_3	execution_time_mean	execution_time_std
Count	36.0	36.0	36.0	36.0	36.0
Mean	37.690	37.275	37.405	37.457	0.934
Std	95.677	94.203	95.094	94.985	1.238
Min	2.267	2.787	2.518	2.876	0.049
Q_1	3.7028	3.802	3.554	3.699	0.384
Q_2	4.245	4.319	4.719	4.359	0.527
Q_3	6.254	5.714	5.782	5.568	0.773
Max	349.977	345.565	346.652	346.502	5.476

APÊNDICES 6 - Estatísticas descritiva com índice *btree* com *queries* de agregação e tamanho amostral de 10 milhões notas

Tabela 20: Estatísticas descritivas para o tempo de execução, em milissegundos (ms), considerando o Citus com 3 *workers* e índice btree para amostra *big*.

	execution_time_1	execution_time_2	execution_time_3	execution_time_mean	execution_time_std
Count	48.0	48.0	48.0	48.0	48.0
Mean	6338.129	5576.767	5561.196	5825.364	470.492
Std	21192.947	20686.404	20786.820	20874.396	972.055
Min	141.542	128.406	114.654	137.737	4.260
Q_1	181.736	148.847	152.3195	170.752	20.164
Q_2	262.95	180.749	177.230	226.367	36.508
Q_3	627.584	298.289	274.253	381.069	213.876
Max	88040.686	86041.198	85656.078	86479.556	3924.231

Fonte: Elaborada pelo Autor (2022).

Tabela 21: Estatísticas descritivas para o tempo de execução, em milissegundos (ms), considerando o Citus com 6 *workers* e índice btree para amostra *big*.

	execution_time_1	execution_time_2	execution_time_3	execution_time_mean	execution_time_std
Count	48.0	48.0	48.0	48.0	48.0
Mean	3741.787	3332.589	3133.749	3402.708	381.151
Std	11772.114	12000.813	11362.094	11695.320	792.908
Min	171.049	169.612	156.057	175.599	1.074
Q_1	190.263	191.144	184.833	197.256	14.421
Q_2	228.762	205.311	204.692	214.525	23.514
Q_3	355.762	247.900	245.622	270.773	61.071
Max	49678.106	50025.405	46764.602	48771.722	3274.194

Tabela 22: Estatísticas descritivas para o tempo de execução, em milissegundos (ms), para índice *btree* e amostra *big*, considerando o Postgres com 1PC.

	execution_time_1	execution_time_2	execution_time_3	execution_time_mean	execution_time_std
Count	48.0	48.0	48.0	48.0	48.0
Mean	8066.157	7265.437	7396.219	7575.938	460.083
Std	22301.427	21416.604	21628.084	21757.727	1331.742
Min	2.503	2.94	2.029	3.052	0.304
Q_1	19.156	4.554	4.131	9.342	2.625
Q_2	142.740	25.473	26.260	71.404	13.375
Q_3	812.303	790.432	736.344	799.586	122.588
Max	87570.771	85606.528	86220.643	86349.789	6318.225

Tabela 23: Estatísticas descritivas para o tempo de execução, em milissegundos (ms), para índice *btree* e amostra *big*, considerando o Postgres com 2PC.

	execution_time_1	execution_time_2	execution_time_3	execution_time_mean	execution_time_std
Count	48.0	48.0	48.0	48.0	48.0
Mean	5948.920	5652.114	5574.952	5725.329	383.591
Std	20593.283	21121.767	20848.963	20834.671	1100.039
Min	2.993	2.809	2.556	3.412	0.262
Q_1	10.173	4.373	4.428	6.582	2.274
Q_2	71.426	15.469	15.316	34.112	11.635
Q_3	713.667	645.844	656.457	676.608	49.303
Max	84971.354	89193.143	88080.733	87415.077	6587.631

APÊNDICES 7 - Estatísticas descritiva com índice *btree* com *queries* de consulta completa e tamanho amostral de 850 mil notas

Tabela 24: Estatísticas descritivas para o tempo de execução, em milissegundos (ms), considerando o Citus com 3 *workers* e índice btree com métrica consulta completa para amostra *small*.

	execution_time_1	execution_time_2	execution_time_3	execution_time_mean	execution_time_std
Count	12.0	12.0	12.0	12.0	12.0
Mean	60196.118	62385.990	62647.998	61743.369	1385.852
Std	207315.459	214940.451	215894.749	212716.883	4689.704
Min	168.733	180.371	172.188	186.100	1.879
Q_1	185.329	202.038	190.588	192.327	16.830
Q_2	205.924	220.928	213.856	204.000	21.425
Q_3	482.398	480.747	417.929	460.358	51.918
Max	718510.556	744913.094	748205.429	737209.693	16277.382

Tabela 25: Estatísticas descritivas para o tempo de execução, em milissegundos (ms), considerando o Citus com 6 *workers* e índice btree com métrica consulta completa para amostra *small*.

	execution_time_1	execution_time_2	execution_time_3	execution_time_mean	execution_time_std
Count	12.0	12.0	12.0	12.0	12.0
Mean	74416.046	73387.798	68320.720	72041.521	3284.305
Std	256383.163	253065.631	235510.895	248319.886	11209.848
Min	197.09	177.418	178.485	188.373	4.438
Q_1	214.059	195.940	195.836	205.568	13.579
Q_2	221.253	228.040	214.0345	218.895	21.646
Q_3	603.283	417.804	418.518	504.017	36.276
Max	888541.24	876978.635	816167.711	860562.529	38879.279

Tabela 26: Estatísticas descritivas para o tempo de execução, em milissegundos (ms), para índice *btree* para amostra *small*, considerando o Postgres com 1PC e métrica de consulta completa.

	execution_time_1	execution_time_2	execution_time_3	execution_time_mean	execution_time_std
Count	12.0	12.0	12.0	12.0	12.0
Mean	16853.814	16374.076	16625.296	16617.729	247.429
Std	39262.910	38237.930	38843.256	38779.307	707.792
Min	3.994	4.246	4.538	4.743	0.138
Q_1	8.032	5.191	5.654	6.069	1.233
Q_2	37.157	6.810	6.284	16.094	18.242
Q_3	168.961	15.282	15.247	66.383	67.443
Max	105845.435	100886.923	103627.991	103453.450	2483.860

Tabela 27: Estatísticas descritivas para o tempo de execução, em milissegundos (ms), para índice *btree* para amostra *small*, considerando o Postgres com 2PC e métrica de consulta completa.

	execution_time_1	execution_time_2	execution_time_3	execution_time_mean	execution_time_std
Count	12.0	12.0	12.0	12.0	12.0
Mean	860.523	1494.346	862.604	1072.491	366.239
Std	2848.379	5044.535	2857.699	3583.439	1265.402
Min	4.088	4.096	4.029	4.071	0.036
Q_1	5.337	4.752	4.873	4.998	0.242
Q_2	5.985	5.456	5.2075	5.427	0.425
Q_3	14.509	12.854	11.966	13.110	1.256
Max	9900.074	17509.825	9931.569	12447.156	4384.428

APÊNDICES 8 - Estatísticas descritiva com índice *btree* com grupos *queries* de consulta completa e tamanho amostral de 10 milhões notas

Tabela 28: Estatísticas descritivas para o tempo de execução, em milissegundos (ms), considerando o Citus com 3 *workers* e índice btree com métrica consulta completa para amostra *big*.

	execution_time_1	execution_time_2	execution_time_3	execution_time_mean	execution_time_std
Count	15.0	15.0	15.0	15.0	15.0
Mean	50194.308	49219.889	51446.801	50286.999	1886.864
Std	127834.090	126064.624	128649.724	127451.383	4902.664
Min	185.235	169.51	178.644	183.826	3.162
Q_1	208.527	185.88	193.178	199.976	22.957
Q_2	334.782	225.437	199.801	255.863	41.547
Q_3	1011.593	706.156	893.961	870.570	227.404
Max	474937.099	467447.229	467377.71	469920.679	18753.394

Tabela 29: Estatísticas descritivas para o tempo de execução, em milissegundos (ms), considerando o Citus com 6 *workers* e índice btree com métrica consulta completa para amostra *big*.

	execution_time_1	execution_time_2	execution_time_3	execution_time_mean	execution_time_std
Count	12.0	12.0	12.0	12.0	12.0
Mean	74416.046	73387.798	68320.720	72041.521	3284.3049195257263
Std	256383.163	253065.631	235510.895	248319.886	11209.848
Min	197.09	177.418	178.485	188.373	4.438
Q_1	214.059	195.940	195.836	205.568	13.579
Q_2	221.253	228.040	214.034	218.895	21.646
Q_3	603.283	417.804	418.518	504.017	36.276
Max	888541.24	876978.635	816167.711	860562.529	38879.279

Tabela 30: Estatísticas descritivas para o tempo de execução, em milissegundos (ms), para índice *btree* para amostra *big*, considerando o Postgres com 1PC e métrica de regitro completo.

	execution_time_1	execution_time_2	execution_time_3	execution_time_mean	execution_time_std
Count	15.0	15.0	15.0	15.0	15.0
Mean	7293.755	6489.884	6709.314	6830.984	431.914
Std	22341.810	21740.647	21864.991	21963.432	1158.279
Min	3.994	4.52	4.361	4.869	0.126
Q_1	32.462	6.014	6.462	22.210	4.534
Q_2	68.402	25.343	24.894	67.207	15.308
Q_3	961.984	958.239	970.995	963.740	194.300
Max	86628.739	84728.945	85205.208	85520.964	4502.858

Tabela 31: Estatísticas descritivas para o tempo de execução, em milissegundos (ms), para índice *btree* para amostra *big*, considerando o Postgres com 2PC e métrica de consulta completa.

	execution_time_1	execution_time_2	execution_time_3	execution_time_mean	execution_time_std
Count	15.0	15.0	15.0	15.0	15.0
Mean	8814.779	6683.647	6661.373	7386.600	1248.609
Std	24143.238	22425.782	22625.160	22780.820	4513.859
Min	4.962	4.791	4.364	5.048	0.194
Q_1	27.611	6.324	5.881	16.852	2.319
Q_2	75.716	17.924	19.725	72.179	20.762
Q_3	790.098	745.1605	758.088	764.449	119.604
Max	88439.507	87322.589	88122.514	87961.537	17555.510