Universidade Federal da Paraíba
Centro de Informática
Programa de Pós-Graduação em Informática

Dissertação de Mestrado

Um estudo comparativo sobre uso de modelos de dados para notas fiscais eletrônicas

Daniel Fagner Pedroza Santos

João Pessoa - PB
Julho - 2021

Universidade Federal da Paraíba
Centro de Informática
Programa de Pós-Graduação em Informática

Um estudo comparativo sobre uso de modelos de dados para notas fiscais eletrônicas

Daniel Fagner Pedroza Santos

Dissertação submetida à Coordenação do Curso de Pós-Graduação em Informática da Universidade Federal da Paraíba como parte dos requisitos necessários para obtenção do grau de Mestre em Informática.

Área de Concentração: Computação Distribuída

Profa. Dra. Thaís Gaudencio do Rêgo
Orientadora

Prof. Dr. Raoni Kulesza
Coorientador

João Pessoa - PB
Julho - 2021

Catalogação na publicação
Seção de Catalogação e Classificação

S237e Santos, Daniel Fagner Pedroza.
Um estudo comparativo sobre uso de modelos de dados para notas fiscais eletrônicas / Daniel Fagner Pedroza Santos. - João Pessoa, 2021.
68 f.

Orientação: Thaís Gaudencio do Rêgo.
Coorientação: Raoni Kulesza.
Dissertação (Mestrado) - UFPB/CI.

1. Base de dados. 2. Big Data. 3. Nota fiscal - Eletrônica. 4. Escrituração digital. 5. Benchmark. I. Rêgo, Thaís Gaudencio do. II. Kulesza, Raoni. III. Título.

UFPB/BC CDU 004.65(043)

Elaborado por ANNA REGINA DA SILVA RIBEIRO - CRB-15/024

UNIVERSIDADE FEDERAL DA PARAÍBA
CENTRO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA

Ata da Sessão Pública de Defesa de Dissertação de Mestrado de Daniel Fagner Pedroza Santos, candidato ao título de Mestre em Informática na Área de Sistemas de Computação, realizada em 28 de julho de 2021.

Aos vinte e oito dias do mês de julho do ano de dois mil e vinte e um, às quatorze horas, por meio de videoconferência, reuniram-se os membros da Banca Examinadora constituída para julgar o trabalho do sr. Daniel Fagner Pedroza Santos, vinculado a esta Universidade sob a matrícula nº 20191000559, candidato ao grau de Mestre em Informática, na área de "Sistemas de Computação", na linha de pesquisa "Computação Distribuída", do Programa de Pós-Graduação em Informática, da Universidade Federal da Paraíba. A comissão examinadora foi composta pelos professores: Thaís Gaudencio do Rêgo (PPGI), Orientadora e Presidente da banca; Rostand Edson Oliveira Costa (PPGI), Examinador Interno; Raoni Kulesza (UFPB), Examinador Externo ao Programa e Coorientador; Telmo de Menezes e Silva Filho (UFPB), Examinador Externo ao Programa; Yuri de Almeida Malheiros Barbosa (UFPB), Examinador Externo ao Programa. Dando início aos trabalhos, a Presidente da Banca cumprimentou os presentes, comunicou aos mesmos a finalidade da reunião e passou a palavra ao candidato para que o mesmo fizesse a exposição oral do trabalho de dissertação intitulado: "Um estudo comparativo sobre uso de modelos de dados para notas fiscais eletrônicas". Concluída a exposição, o candidato foi arguido pela Banca Examinadora que emitiu o seguinte parecer: "***aprovado***". Do ocorrido, eu, Ruy Alberto Pisani Altafim, Coordenador do Programa de Pós-Graduação em Informática, lavrei a presente ata que vai assinada por mim e pelos membros da banca examinadora. João Pessoa, 28 de julho de 2021.

Prof. Dr. Ruy Alberto Pisani Altafim

Profa. Thaís Gaudencio do Rêgo
Orientadora (PPGI-UFPB)

Prof. Rostand Edson Oliveira Costa
Examinador Interno (PPGI-UFPB)

Prof. Raoni Kulesza
Examinador Interno (PPGI-UFPB) e Coorientador (UFPB)

Telmo de Menezes e Silva Filho
Examinador Externo ao Programa (UFPB)

Prof. Yuri de Almeida Malheiros Barbosa
Examinador Externo ao Programa (UFPB)

DEDICATÓRIA

Dedico esse trabalho a minha avó
materna Maria Bezerra Pedroza

AGRADECIMENTOS

Agradeço aos meus familiares, pais e irmãs por todo esforço e incentivo recebidos, que tornaram possível concluir esta etapa de minha vida. Dentre os familiares, deixo aqui um agradecimento especial para a minha avó materna, que foi minha maior incentivadora durante o tempo em que esteve presente.

Também gostaria de agradecer a minha namorada, Marina Moraes, que sempre me incentivou na jornada acadêmica e profissional, assim como para nossa cadelinha, Maggie, que foi minha fiel companheira durante as diversas noites de desenvolvimento desta pesquisa.

Agradeço a todos os professores dessa instituição, que me proporcionaram inúmeros conhecimentos profissionais e valores pessoais ao longo da minha jornada acadêmica.

Agradeço a minha orientadora, Thaís Gaudencio do Rêgo, e ao meu co-orientador, Raoni Kulesza, por todo seu apoio, incentivo e ajuda na realização desse trabalho, por suas orientações e, principalmente, por sua paciência.

Deixo, em especial, um grande agradecimento a todos os meus amigos.

RESUMO

O termo *Big Data* tem se popularizado cada vez mais, e consiste em um grande volume de dados e, por isso, necessita de atenção especial, como o uso de ferramentas não convencionais, que possuam a capacidade de armazenar, processar e integralizar todos as informações de uma enorme base de dados. É comum nos dias atuais, empresas e organizações adotarem essas tecnologias, visto que vivemos atualmente na era da informação e o volume de dados cresce de forma exponencial. Isso ocorre também no mundo contábil, principalmente a partir de 2007, quando foi implementado no Brasil, através do Governo Federal, 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. Isso levou ao uso constante de um grande volume de dados com notas fiscais eletrônicas entre esses órgãos, fazendo com que eles precisem ter um modelo de dados estruturado e robusto para o armazenamento, processamento e utilização por meio de consultas desses documentos fiscais eletrônicos. Este estudo analisou a performance de dois modelos de dados: relacional e orientado a documento, através da utilização do PostgreSQL e do MongoDB respectivamente, executado por um *benchmark* de consultas, utilizando um lote de 50 e 100 mil notas fiscais eletrônicas (NFes) e tendo-se como métrica o tempo de resposta. Obteve-se como resultado que o MongoDB possui uma taxa de eficiência no tempo de resposta das consultas superior ao do PostgreSQL, independente do operador analisado. Esta pesquisa trouxe duas contribuições principais: a possibilidade de mensurar o desempenho de dois modelos de dados, utilizando uma base de dados de NFes no âmbito da *Big Data*, assim como estabelecer, através de um software, um entendimento melhor dos campos de uma NF-e.

Palavras-chave: *Big Data*, bancos de dados, SQL, NoSQL, *benchmark*, escrituração digital, nota fiscal eletrônica

ABSTRACT

The term Big Data has become increasingly popular, and consists of a large volume of data and, therefore, needs special attention, such as the use of unconventional tools, which have the ability to store, process and integrate all information of a huge database. It is common these days, companies and organizations to adopt these technologies, as we currently live in the information age and the volume of data grows exponentially. This also occurs in the accounting world, especially from 2007, when the Public System of Digital Bookkeeping was implemented in Brazil, through the Federal Government, in private sector companies and also in public government institutions, responsible for the inspection and control of these activities such as municipal and state tax authorities and the Federal Revenue Service. This has led to the constant use of a large volume of data with electronic invoices among these agencies, making it necessary for them to have a structured and robust data model for the storage, processing and use through queries of these electronic fiscal documents. This study analyzed the performance of two data models: relational and document-oriented, using PostgreSQL and MongoDB respectively, executed by a query benchmark, using a batch of 50 and 100 thousand electronic invoices (NFes) and having up as a response time metric. It was obtained as a result that MongoDB has a higher efficiency rate in the query response time than PostgreSQL, regardless of the analyzed operator. This research brought two main contributions: the possibility of measuring the performance of two data models, using a database of NFes in the context of Big Data, as well as establishing, through software, a better understanding of the fields of an NF- and.

Keywords: *Big Data*, databases, SQL, NoSQL, *benchmark*, digital bookkeeping, electronic invoice

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

BSON	*Binary JSON*
CT-e	Conhecimento de Transporte eletrônico
FCONT	Controle Fiscal Contábil de Transição
ECD	Escrituração Contábil Digital
EFD	Escrituração Fiscal Digital ou Sped Fiscal
ETL	*Extract, Transform, Load* (Extração, Transformação, Carregamento)
EFD	Escrituração Fiscal Digital ou Sped Fiscal
XML	*Extensible Markup Language*
IBM	*International Business Machines Corporation*
JSON	*JavaScript Object Notation*
NF-e	Nota Fiscal Eletrônica
NF-s	Nota Fiscal de Serviços Eletrônica
OLAP	*Online Analytical Processing*
SPED	Sistema Público de Escrituração Digital
SGBD	Sistema de Gerenciamento de Banco de Dados
UoD	Universo do discurso
XSD	*XML Schema Definition*

SUMÁRIO

1 INTRODUÇÃO

O Sistema Público de Escrituração Digital (SPED) é um sistema composto por uma série de subprojetos: 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), que por sua vez geram um grande volume de dados, que cresce de forma rápida, levando ao acúmulo de informações nos mais variados bancos de dados, tornando necessário o desenvolvimento de tecnologias capazes de realizar o processamento de forma ágil e eficaz, sendo este um dos maiores desafios da área.

O SPED trata de um instrumento que une as atividades de recepção, armazenamento, validação e autenticação de livros e documentos integrantes da escrituração comercial e fiscal das organizações, sendo resultado do fluxo único e informatizado de informações [4]. Nele, as NF-e constituem um dos seus principais documentos, sendo utilizadas em larga escala, gerando, com isso, uma enorme quantidade de dados, que necessitam ser processados pelos órgãos fiscais competentes, como as Secretarias Estaduais da Fazenda e a própria Receita Federal. Para isso, torna-se necessário que os órgãos possuam modelos de dados e processamento de arquivos bem definidos, utilizando as melhores ferramentas e a melhor modelagem para suas atividades fins.

Um modelo de dados trata de uma coleção de conceitos que podem ser utilizados para descrever um conjunto de dados e operações para manipulá-los. Os modelos de dados podem ser classificados em duas dimensões: na primeira dimensão, são classificados em função da etapa de desenvolvimento do projeto em banco de dados, em que o modelo é utilizado. Por sua vez, a segunda dimensão classifica os modelos de dados quanto a sua flexibilidade e poder de expressão [5].

Este estudo propõe um comparativo entre os modelos de dados fazendo uso da base de dados das NF-e. Serão implementados dois modelos de dados: relacional, e orientado à documento, onde serão analisados o desempenho e a performance de cada um deles, bem como o processamento da base de dados em cada modelo.

1.1 HIPÓTESE DE PESQUISA

Neste estudo, teve-se como hipótese que o modelo de dados MongoDB teria uma performance de desempenho superior ao PostgreSQL, em virtude de seus modelos

estruturais, pois, ao conter coleções de documentos, as consultas executadas no MongoDB teriam um tempo de resposta menor.

1.2 OBJETIVO GERAL

Comparar a performance de desempenho dos modelos de dados relacional e orientado à documento, utilizando-se da base de dados de NF-e.

1.2.1 OBJETIVOS ESPECÍFICOS

- Avaliar, nos modelos de dados utilizados, o carregamento de dados, a modelagem e estrutura da base de dados, a performance de consulta e a visualização dos resultados.
- Definir um conjunto de consultas que será utilizado no estudo.
- Definir o modelo de dados que se adequa ao âmbito do *Big Data*, isto é, através de um caso de uso (validado utilizando uma base de dados real).

1.3 ESTRUTURA DA DISSERTAÇÃO

O restante deste texto está organizado da seguinte forma:

Capítulo 2: Apresenta a fundamentação teórica necessária para o desenvolvimento do trabalho e conceitos relacionados aos modelos de dados, suas características e as NF-e.

Capítulo 3: Apresenta os trabalhos relacionados, os principais pontos sobre os trabalhos no âmbito de *benchmark* de consultas e a tecnologia OLAP.

Capítulo 4: Mostra a proposta de implementação dos dois modelos de dados que receberão a mesma amostra de dados para análise, e que também estará detalhada aqui.

Capítulo 5: Apresenta os resultados esperados com relação ao comparativo entre os modelos de dados estudados: orientados à documentos e relacional.

Capítulo 6: Apresenta as considerações finais e os trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Nesta capítulo serão destacados conceitos fundamentais para o entendimento do trabalho, tais como Sistema Público de Escrituração Digital (SPED), em especial a Nota Fiscal Eletrônica - NF-e, modelos de bancos de dados e suas características e *Big Data*.

2.1 SPED - SISTEMA PÚBLICO DE ESCRITURAÇÃO DIGITAL

O SPED foi instituído pelo Decreto nº 6.022, de 22 de janeiro de 2007, pelo Governo Federal do Brasil, destacando-se por ser um avanço na relação entre o fisco e o contribuinte. Consiste na modernização da sistemática atual do cumprimento das obrigações acessórias efetuadas pelos contribuintes às administrações públicas e aos órgãos fiscalizadores [6].

O referido sistema tem como foco promover a integração entre os fiscos federais, estaduais e municipais, bem como integralizar todo o fluxo relativo às notas fiscais. Para que tenha validação jurídica, o sistema utiliza a certificação digital para fins de assinatura de documentos eletrônicos [7].

Conforme a Receita Federal do Brasil (RFB), o SPED tem como objetivos [6]:

- Promover a integração dos fiscos, mediante a padronização e compartilhamento das informações contábeis e fiscais, respeitadas as restrições legais;
- Racionalizar e uniformizar as obrigações acessórias para os contribuintes, com o estabelecimento de transmissão única de distintas obrigações acessórias de diferentes órgãos fiscalizadores, e;
- 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.

São sete os principais subprojetos que compõem o SPED (Figura 2.1), sendo estes arquivos eletrônicos que devem ser fornecidos ao fisco, a saber: ECD, EFD, CT-e, FCONT, NFS-e, EFD-Contribuições, NF-e.

Como cada subprojeto do SPED apresenta um especificidade, segue um detalhamento de cada um:

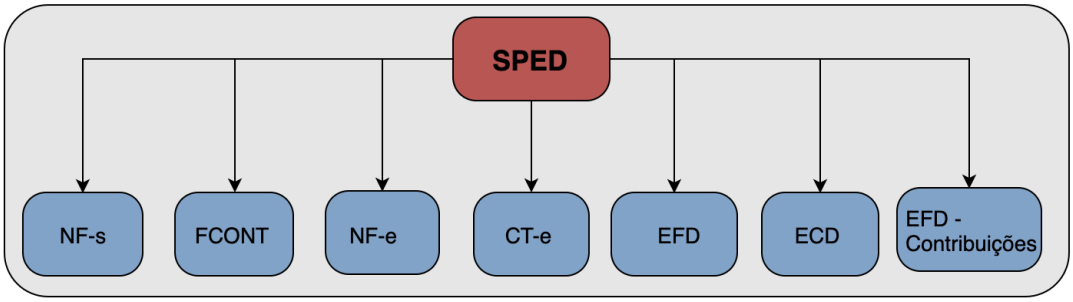

Figura 2.1 – Os subprojetos do SPED.
Fonte: Elaborada pelo Autor [8]

2.1.1 ESCRITURAÇÃO CONTÁBIL DIGITAL - ECD

A ECD tem por objetivo a substituição da escrituração em papel pela transmitida via arquivo digital, tornando obrigatória a transmissão neste formato ao fisco dos seguintes livros: I. livro Diário e seus auxiliares, se houver; II. livro-razão e seus auxiliares, se houver e III. livro de Balancetes Diários, Balanços e fichas de lançamento comprobatórias dos assentamentos neles transcritos [6].

Com a implementação desse sistema, 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. Em seguida, ocorre a validação por meio de um software disponibilizado pela Receita Federal do Brasil, Programa Validador Assinador (PVA), e, posteriormente, o envio para o sistema do SPED e para a Junta Comercial [9].

2.1.2 ESCRITURAÇÃO FISCAL DIGITAL OU SPED FISCAL - EFD

A EFD objetiva garantir o compartilhamento de informações relacionadas às escriturações fiscais e contábeis digitais em âmbito nacional. A sua criação se deu pela necessidade de substituir as diversas obrigações fiscais estaduais, passando a ser transmitidas digitalmente, assim, padronizando-as. A EFD necessita ser validada através de assinatura digital do representante legal da empresa. Os seguintes livros são contemplados nesta escrituração: I. Registro de Entradas; II. Registro de Saídas; III. Registro de Inventário; IV. Registro de Apuração do Imposto sobre Produtos Industrializados (IPI) e IV. Registro de Apuração do ICMS [7].

2.1.3 CONHECIMENTO DE TRANSPORTE ELETRÔNICO - CT-E

O CT-e objetiva documentar a prestação de serviços de transportes, de forma completamente digital, emitida e armazenada eletronicamente, substituindo seis tipos de documentos em papel, sendo eles: I. Conhecimento de Transporte Rodoviário de

Cargas, Modelo 8; II. Conhecimento de Transporte Aquaviário de Cargas, Modelo 9; III. Conhecimento Aéreo, Modelo 10; IV. Conhecimento de Transporte Ferroviário de Cargas, Modelo 11; V. Nota Fiscal de Serviço de Transporte Ferroviário de Cargas, Modelo 27 e VI. Nota Fiscal de Serviço de Transporte, Modelo 7, quando utilizada em transporte de cargas [9].

2.1.4 CONTROLE FISCAL CONTÁBIL DE TRANSIÇÃO - FCONT

O FCONT trata de um programa eletrônico que executa a escrituração das contas patrimoniais e de resultado. Objetiva reverter as repercussões tributárias provenientes dos lançamentos, que alteram o resultado para fins de apuração do lucro real e da base de cálculo da contribuição social sobre o lucro líquido (CSLL) [6].

2.1.5 NOTA FISCAL DE SERVIÇOS ELETRÔNICA - NFS-E

Documento de existência digital, gerado e armazenado eletronicamente pela Receita Federal do Brasil, pela gestão municipal ou por outra entidade conveniada, com o intuito de documentar as atividades de prestação de serviços, sendo emitida automaticamente através de serviços informatizados, disponibilizados a todos contribuintes. Pertence ao contribuinte a responsabilidade pelo fornecimento de dados corretos, posteriormente analisados e validados pela secretaria [6].

2.1.6 EFD - CONTRIBUIÇÕES - (PIS E COFINS)

A EFD é um arquivo digital utilizado por pessoas jurídicas de direito privado na escrituração da contribuição para o PIS/Pasep e da Cofins (tributos), nos regimes de apuração não-cumulativo e/ou cumulativo [6]. Documentos e operações representativos das receitas obtidas pela instituição, assim como custos, despesas, encargos e aquisições geradores de créditos da não cumulatividade, são a base do conjunto dessa escrituração [9].

2.1.7 NOTA FISCAL ELETRÔNICA - NF-E

A nota fiscal está em constante evolução desde a década de 1970, passando de uma emissão realizada de forma manual, através da máquina mecanográfica de escrever, até os dias atuais, onde, pela chegada dos computadores, teve sua emissão realizada de forma eletrônica por meio de impressora. Todo documento que tenha emissão e armazenamento de forma eletrônica, e possua apenas existência digital, é considerada NF-e, tendo esta o objetivo de documentar operações mercantis e prestações. A NF-e substitui tanto a Nota Fiscal, modelo 1 ou 1-A, como a Nota Fiscal de Produtor, Modelo 4. Assim, esse documento pode ser utilizado pelos contribuintes do Imposto sobre Produtos Industrializados (IPI)

ou Imposto sobre Operações Relativas à Circulação de Mercadorias e sobre a Prestação de Serviços de Transporte Interestadual e Intermunicipal e de Comunicação (ICMS) [9].

O fluxo operacional da NF-e pode ser observado na Figura 2.2 e ocorre no momento da emissão de uma NF-e. Algumas etapas são seguidas no momento dessa emissão, tais como a geração de dados, armazenamento, determinação dos agentes e responsabilidades envolvidas, assim como dos aspectos relacionados à integridade e a contingência do processo. A seguir tais etapas são descritas brevemente:

- O emissor do documento fiscal eletrônico (vendedor) gera e assina um arquivo digital, que contém os dados de transação comercial, transmitindo à Secretaria de Estado da Fazenda (SEFAZ) do seu Estado. O documento é recebido por esta secretaria para que possa ser validado. Uma das validações se dá através da assinatura digital, que garante a integridade e autoria do mesmo. Após a autorização de uso por parte da SEFAZ, o documento torna-se apto para a operação para o qual foi emitido e é armazenado no site da Receita Federal e da respectiva receita estadual [8].
- Após a geração do arquivo eletrônico, um documento auxiliar é impresso em papel para acompanhar a mercadoria em trânsito. Este documento auxiliar é chamado de Documento Auxiliar de Nota Fiscal Eletrônica – DANFE.
- O emitente da NF-e deve enviar o arquivo eletrônico até o recebedor da mercadoria ou serviço, que por sua vez, deve aceitar e fazer a devida contabilização.
- As fiscalizações tributárias, de posse das informações dos documentos fiscais eletrônicos, armazenam e cruzam os dados para detectar possíveis indícios de sonegação e, consequentemente, iniciar procedimentos como conferência física de cargas e auditorias em empresas. Trata-se de um processo sistemático de validação das informações.

O fluxo acima descrito é dinâmico e ocorre em ambiente web. Em caso de alguma interrupção nesse canal de comunicação, surge o processo denominado contingência: a NF-e é emitida no ambiente de serviços assíncronos, como pode ser observado na Figura 2.3, passando a fluir em modo não sincronizado. Posteriormente, é retomada na internet de modo a não interromper a logística interna das empresas [8].

2.1.8 ORGANIZAÇÃO DOS DADOS DA NF-E

A organização dos dados na NF-e se dá através do conceito de *Schema XML* com hierarquia de campos, sendo de grande valia que se compreenda toda a estrutura do arquivo para uma melhor orientação e manipulação dos dados. Abaixo segue uma sucinta explicação sobre tais conceitos [8].

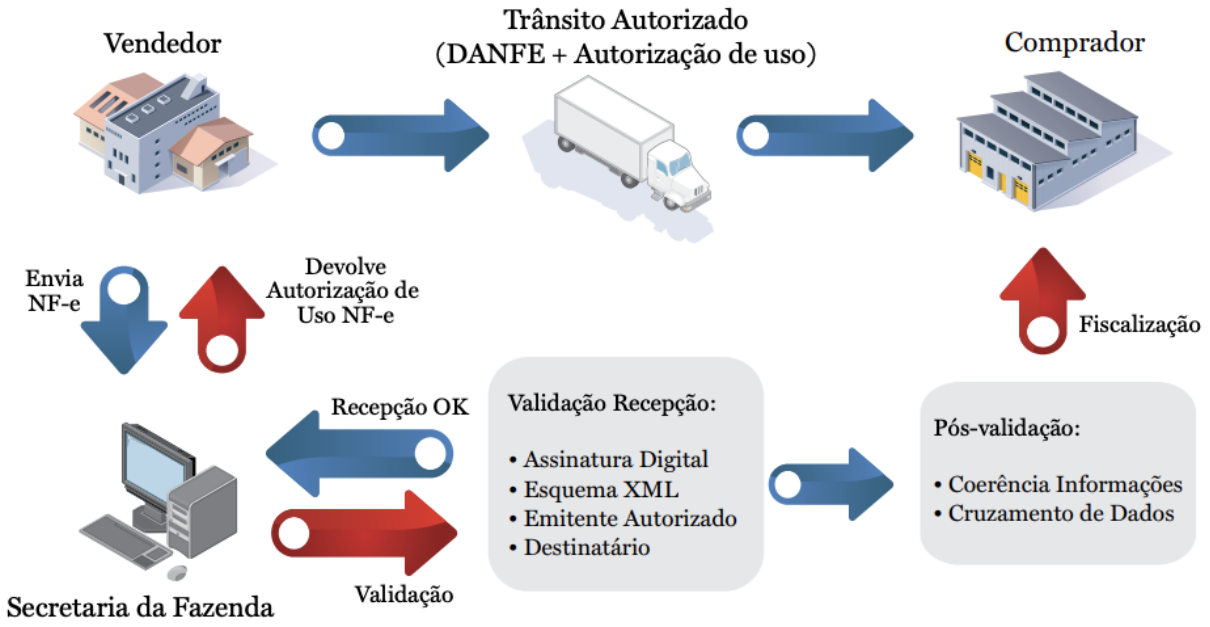

Figura 2.2 – Fluxo operacional da NF-e.
Fonte: Adaptada de [8]

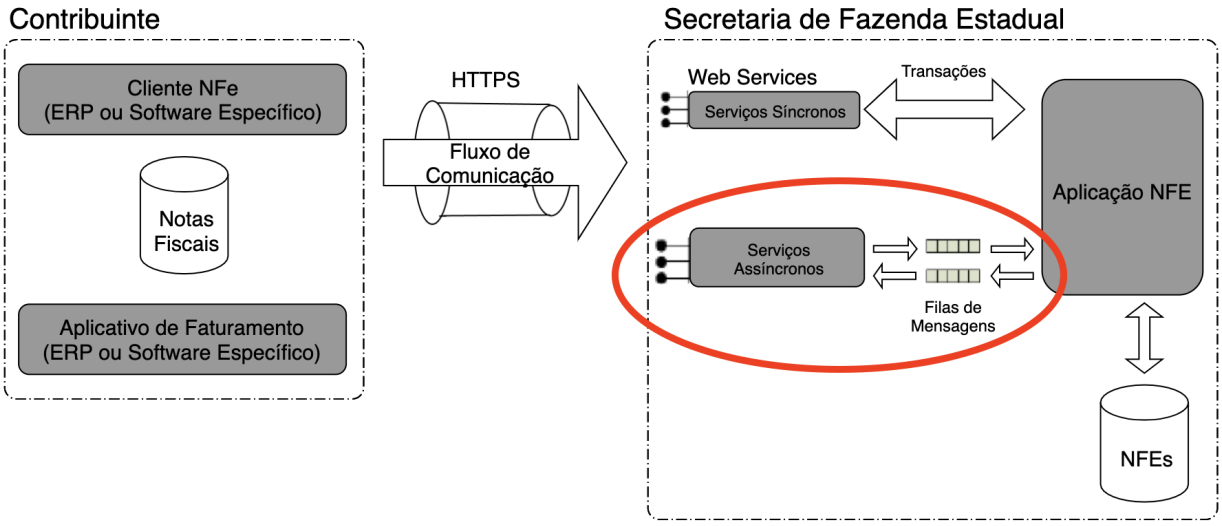

Figura 2.3 – Arquitetura de Comunicação da NF-e.
Fonte: Adaptada de [8]

- **Schema XML:** Trata-se da linguagem de marcação que define o conteúdo do documento eletrônico e a sua organização. Na Figura 2.4, nota-se que 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), que, por sua vez, está atrelado ao grupo de informações (prod), e este com as informações sobre produtos específicos na NF-e.

- **Coluna campo:** Possui os nomes padronizados dos diferentes campos, como por

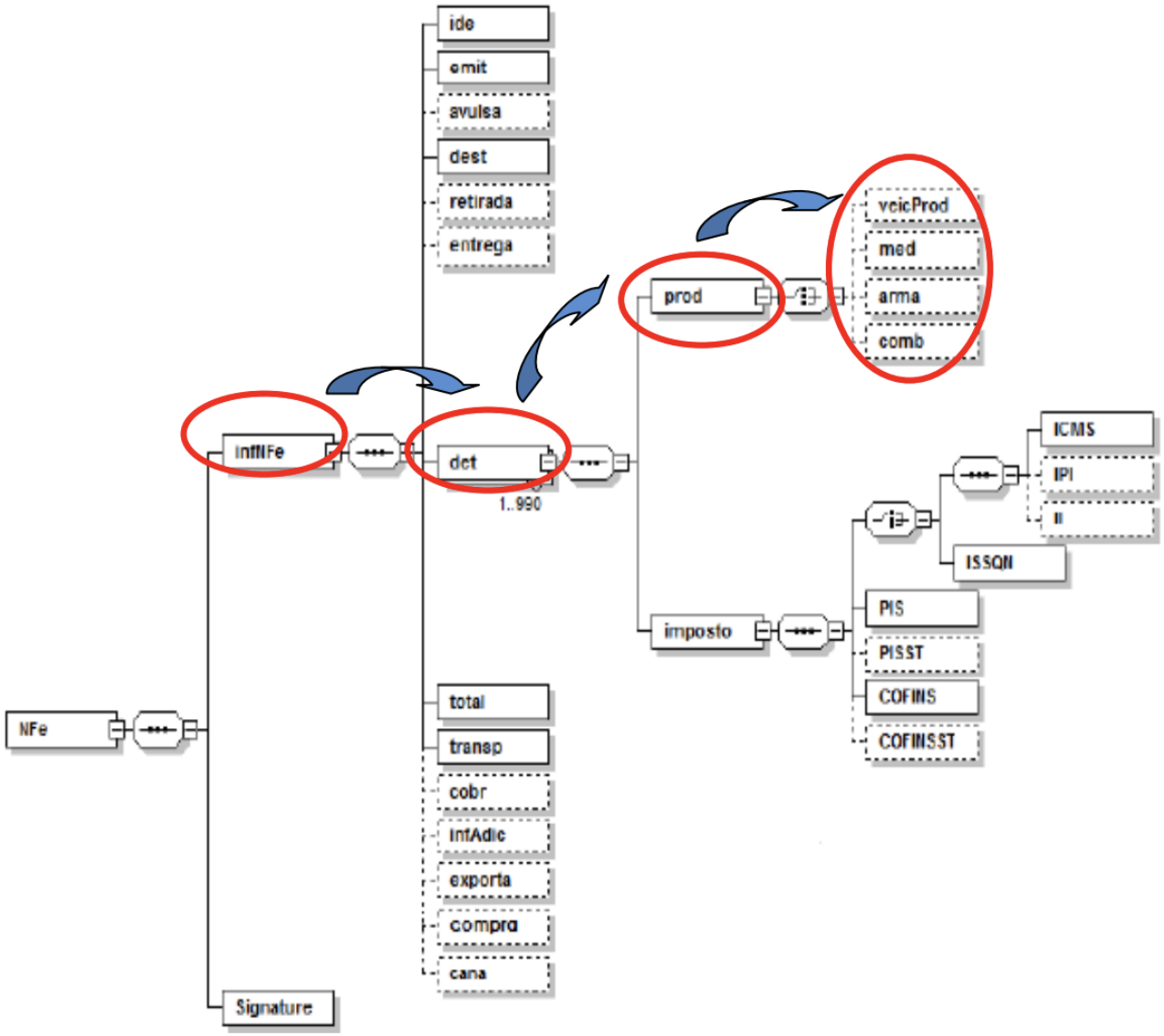

Figura 2.4 – Diagrama do *Schema XML* dos grupos de informações da NF-e.
Fonte: Adaptada de [8]

exemplo, a IE, correspondente a inscrição estadual do contribuinte junto às secretarias de fazenda. É através de qual grupo os campos pertencem no *Schema XML* que é realizada a diferenciação dos campos.

O armazenamento de dados, inclusive NFes, pode ocorrer nos mais variados modelos de dados, seja ele relacional, dimensional, orientado à documento ou orientado à coluna.

2.2 SISTEMA DE GERENCIAMENTO DE GERENCIAMENTO DE BANCO DE DADOS - SGBD

O SGBD trata de uma coleção de programas que tem como função o gerenciamento da estrutura de um banco de dados e controle ao acesso dos dados armazenados. O SGBD serve como intermediário entre o usuário e o banco de dados. 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. A Figura 2.6 esquematiza o funcionamento prático de um SGBD, evidenciando o fato deste fornecer ao usuário final (ou aplicativo) uma visualização única e integrada dos dados no banco. Os aplicativos enviam as solicitações que são recepcionadas pelo SGBD, que, por sua vez, as traduz nas operações complexas necessárias para atendê-las. Dessa maneira, observa-se que o SGBD oculta, dos aplicativos e usuários, a complexidade interna do banco de dados [10].

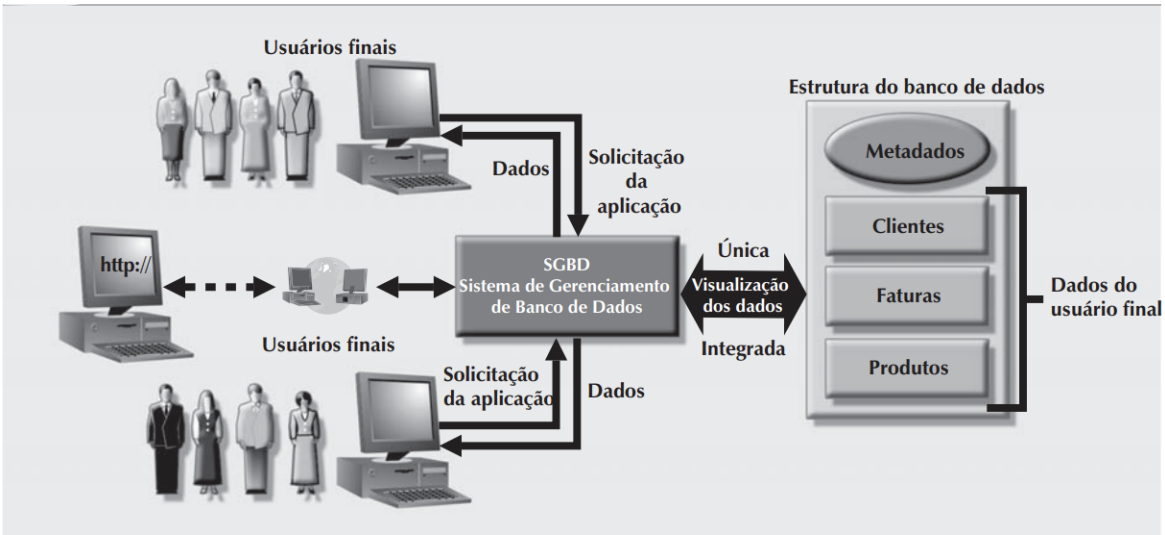

Figura 2.5 – Funcionamento do SGBD
Fonte: Adaptada de [10]

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. 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. Visto que os dados constituem a matéria-prima, a partir da qual as informações são obtidas, é necessário um bom método para gerenciá-lo. Dessa maneira, abaixo é descrito brevemente as melhorias que um SGBD fornece [10].

- Aprimoramento do compartilhamento de dados. O SGBD oferece a possibilidade de criação de um ambiente no qual os usuários sejam capazes de acessar os dados

em maior quantidade e de forma bem gerenciada. Isso possibilita que os usuários respondam de forma célere a mudanças em seu meio [10];

- Aprimoramento da segurança de dados. Se uma quantidade elevada de usuários acessarem os dados, os riscos e falhas de segurança serão mais propensos a ocorrerem. Assim, as corporações investem tempo, esforço e dinheiro para garantir que seus dados sejam usados da forma adequada. Neste aspecto, o SGBD oferece um modelo melhor, no qual é possível aplicar políticas de privacidade e segurança de dados [10], e;

- Melhoria na integração dos dados. O acesso de forma ampla aos dados bem gerenciados promove uma perspectiva de integração das operações da organização e uma visualização mais clara do panorama geral. A visualização de como as ações de um segmento da empresa afetam outros segmentos passa a ser facilitada [10].

Cada tipo de banco de dados possui um modelo de dados, os mais populares são : relacional, modelo dimensional, orientado à documento e orientado à coluna, que serão descritos nas próximas subseções.

2.2.1 MODELO DE DADOS

Os bancos de dados possuem uma característica fundamental que é a de permitir a abstração dos dados, ocultando detalhes do armazenamento de dados que são desnecessários para a maioria dos usuários. Um modelo de dados pode ser definido como um conjunto de conceitos que podem ser usados para descrever a estrutura de um banco de dados, fornecendo o significado necessário para permitir essa abstração. Por estrutura de um banco de dados, entendemos os tipos de dados, relacionamentos e restrições que devem suportá-los. Os modelos de dados representacionais ou de implementação são os mais usados nos SGBDs comerciais tradicionais [11]. Como objeto dessa pesquisa serão abordados dois modelos de dados: modelo relacional, e orientado à documento.

2.2.2 MODELO RELACIONAL

A relação (tabelas) é a principal estrutura fundamental deste modelo, onde uma relação é composta por um ou mais atributos (campos), que traduzem qual o tipo de dados será armazenado. Cada instância do esquema (linha) é chamada de tupla (registro). É importante frisar que esse modelo não possui caminhos pré-definidos para acessar os dados, como ocorre nos modelos que surgiram posteriormente a ele. Neste caso, o modelo relacional implementa estruturas de dados organizados em relacionamentos. A Figura 2.6 mostra tabelas sob o modelo relacional [1], enquanto a Figura 2.7 mostra o diagrama ER de um modelo relacional.

Cod_Cliente	Nome	Rua	Cidade
1	Pedro	A	São Paulo
2	Maria	B	Jundiai

Num_CC	Saldo
20121	1200
21582	1320
21352	652

Cod_Cliente	Num_CC
1	20121
2	21582
2	21352

Figura 2.6 – Tabelas do modelo relacional Cliente - Conta Corrente. Fonte: Adaptada de [1]

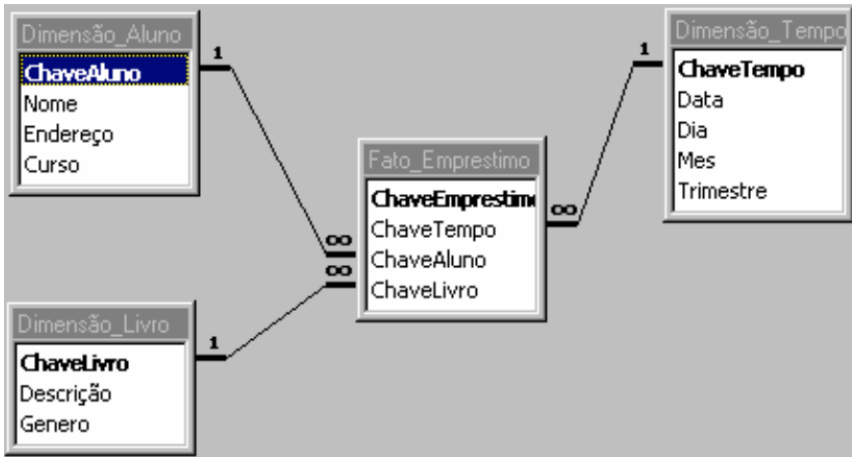

Figura 2.7 – Modelo Dimensional. Fonte: Adaptada de [2]

É importante ressaltar que esse tipo de modelagem possui, além do modelo estrela, o modelo floco de neve.

2.2.3 MODELO ESTRELA

Refere-se a uma estrutura básica de um modelo de dados multidimensional. A estrutura é composta por uma grande entidade central, que é chamado de fato (*fact table*), e por um conjunto de entidades de menor tamanho, chamadas de dimensões (*dimension tables*), dispostas em torno da entidade central, o que leva a um formato de estrela, como mostra a Figura 2.8. Neste modelo não acontece a normalização nas entidades de dimensões, o que ocorre é um relacionamento entre a entidade central fato e as dimensões por meio de ligações simples entre duas tabelas, em um relacionamento de um para muitos, no sentido da dimensão para o fato [12].

Figura 2.8 – Modelo Estrela. Fonte: Adaptada de [2]

2.2.4 MODELO FLOCO DE NEVE

O modelo floco de neve se assemelha ao modelo estrela, pois possui uma fato no centro da estrela e as dimensões a sua volta, formando assim as pontas dessa estrela. Neste modelo, acontece a terceira forma normal que trata da análise e refinamento da estrutura dos dados, fazendo com o que os mesmos sejam íntegros e exclusivos, objetivando repetições desnecessárias e possíveis sobrecargas no gerenciamento do modelo. Essa forma normal é aplicada sobre as entidades de dimensões, evitando assim redundância de dados. Pode-se dizer que tal modelo é o resultado da decomposição de uma ou mais dimensões que possuem hierarquia entre seus membros. Algumas vantagens deste modelo fazem com que os desenvolvedores o escolham, tais como, o seu fácil entendimento no âmbito de sistemas OLTP - *Online Transaction Processing*, por aplicar as formas normais e pelo fato de preservar a utilização de meios de armazenamento. A Figura 2.10 exemplifica esse tipo de modelo [12].

2.2.5 MODELO ORIENTADO À DOCUMENTO

Como o nome sugere, o principal conceito deste modelo é o documento. Neste modelo, o banco de dados armazena e recupera documentos que podem ser no formato *XML*, *JSON*, *BSON*, ou outros. Tais 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. Aqui os documentos que são armazenados possuem características parecidas entre eles, porém, não são necessariamente iguais. Neste caso, os bancos de dados de documentos armazenam documentos na parte do valor do armazenamento de chave-valor; isto pode ser interpretado como sendo um depósito de chave-valor, em que o

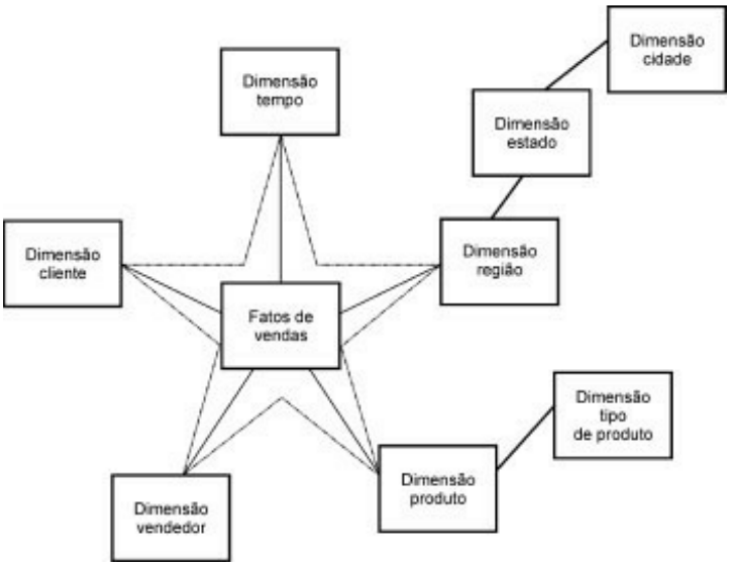

Figura 2.9 – Modelo Floco de Neve. Fonte: Adaptada de [2]

valor pode ser consultado [13].

Segundo [14], os bancos de dados orientados à documentos são compostos de estruturas de dados semiestruturadas, livre de esquema, que recebem o nome de documentos de maneira que retratem uma estrutura de dados poderosa, que pode representar de dados simples, até os mais complexos. Os documentos não possuem uma definição específica com relação a quantidade e tamanho dos campos, sendo assim, são em suma representados em notação de objeto *JavaScript* (*JSON*). É importante ressaltar que neste modelo a palavra documento denota um conjunto de dados semiestruturados e não bancos de dados com documentos, nem sistemas gerenciadores de conteúdo [15].

Os bancos de dados orientados à documentos são também conhecidos como bancos NoSQL *(Not Only SQL)*. Esse termo deriva do fato de ser mais do que apenas SQL, porém esse banco não se resume a apenas essa característica, de modo que o termo não seja o mais apropriado para se referir a este modelo. Vale ressaltar que o termo se tornou comum na comunidade e através da sua popularidade acabou sendo aceito. A próxima subseção trará detalhes a respeito dos bancos NoSQL.

2.2.6 NOSQL

De maneira comum, o NoSQL pode ser denominado como uma classe definida de banco de dados não-relacionais, que armazenam os dados de forma diferente da modelagem relacional, tendo surgido com o intuito de resolver algumas dificuldades presentes no modelo relacional. É preciso compreender que NoSQL não trata de um produto, mas sim de uma classe de produtos e conceitos, com o intuito de armazenar e manipular dados. A

principal diferença que pode ser destacada dos bancos de dados NoSQL para os relacionais são os modelos de dados sem um *schema* definido. Isto porque tais bancos podem ser classificados através de quatro grupos: chave-valor, orientado à documentos, orientado à colunas e baseados em grafos [16]. Não existe uma definição genericamente aceita, e tão pouco uma autoridade que possa fornecer uma, de forma que o que se faz é discutir características comuns em bancos de dados que sejam denominadas de NoSQL, tais como [13]:

- bancos de dados NoSQL não utilizam SQL;
- bancos de dados geralmente são projetos de código aberto;
- a maioria dos bancos de dados NoSQL é orientada pela necessidade de execução em *clusters*, e;
- Os bancos de dados NoSQL atuam sem um esquema, permitindo que sejam adicionados, livremente, campos aos registros do banco de dados, sem ter de definir primeiro quaisquer mudanças na estrutura.

2.2.7 MODELO ORIENTADO À COLUNA

Neste modelo, o armazenamento mantém cada coluna do banco de dados de forma separada, guardando contiguamente os valores de atributos pertencendo à mesma coluna de forma densa e comprimida. Tal maneira de armazenamento, embora beneficie a leitura dos dados, pode comprometer a escrita dos dados. De forma breve, pode-se destacar as três principais vantagens desse tipo de modelo [3]:

- compressão: possui a capacidade de organizar dados semelhantes de forma contígua;
- materialização: não é preciso processar tuplas desnecessárias, e;
- iteração: capacidade de analisar todos os valores de uma coluna com um número menor de instruções no processamento.

É necessário compreender que esse modelo é um pouco mais complexo do que o modelo chave-valor, sendo assim, muda-se o paradigma de orientação à registros ou tuplas para orientação à colunas. No modelo colunar, os dados são indexados por uma tripla (linha, coluna e *timestamp*), onde linhas e colunas são identificadas por chaves e o *timesatamp* proporciona diferenciar múltiplas versões de um mesmo dado.

É importante destacar outros dois conceitos com relação a este modelo, um é o conceito de família de colunas, utilizado com o objetivo de agrupar colunas que guardam

o mesmo tipo de dados, e outro, o conceito de que as operações de leitura e escrita são atômicas, isto é, todos os valores associados a uma linha são considerados na execução destas operações, independentemente das colunas que estão sendo lidas ou escritas [17]. A Figura 2.10 exemplifica um modelo de banco de dados colunar.

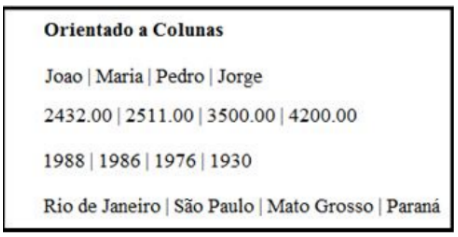

Figura 2.10 – Modelo Colunar. Fonte: Adaptada de [3]

2.2.8 MODELO DIMENSIONAL

Modelagem dimensional trata-se da técnica de projeto lógico de banco de dados, que é mais utilizada no desenvolvimento de *data warehouses*, porém esse modelo também pode ser utilizado em projetos de sistemas de informações operacionais. Tal modelo visa apresentar os dados em um formato que seja de fácil compreensão e que atenda a acessos com um alto desempenho [18].

Esse modelo, de forma diferente do modelo relacional, combina tabelas de armazenamento de dados históricos em séries temporais, indexados em chaves dimensionais, descritas em tabelas correspondentes, isto é, a organização desse modelo se dá por meio de tabelas de fato, que tratam das ocorrências de transações de negócios de uma empresa e tabelas de dimensões que possuem o papel descritivo nessas transações. Por sua vez, as tabelas de fatos contêm as medidas do negócio, além de possuírem dados decorrentes das tabelas de dimensões, o que garante a precisão do acesso aos dados, através de uma estrutura de chave completa, eliminando assim as pesquisas em tabelas e resultando em maior desempenho [2].

Por se tratar de uma técnica de concepção e visualização de um modelo de dados de um conjunto de medidas, que descrevem aspectos comuns de negócios, essa modelagem é utilizada especialmente para sumarizar e reestruturar dados e apresentá-los em visões que suportem a análise dos valores desses dados. Sendo assim, um modelo dimensional é formado por três elementos básicos [12]:

- Fatos: coleção de itens de dados, composta de dados de medidas e de contexto;
- Dimensões: elementos que participam de um fato, assuntos de negócios, e;
- Medidas: atributos numéricos que representam um fato. Avalia a performance de um indicador de negócios relativo às dimensões que participam desse fato (tabela fato).

2.2.9 INDEXAÇÃO EM SGBDS

Índice pode ser considerado como uma organização de dados que possibilita celeridade ao tempo de acesso às linhas de uma tabela. Assemelha-se ao conceito de índices em bancos de dados, o caso do índice remissivo, que é vastamente utilizado em livros, no qual os termos e definições buscados constantemente estão organizados em ordem alfabética, no final das obras. Assim, o leitor pode ir diretamente à página com o conteúdo desejo, mitigando o tempo da busca por tal informação/conceito. Da mesma forma que cabe ao autor prever os temas que os leitores irão provavelmente buscar, os índices trarão benefícios devem ser elencados previamente pelo administrador dos bancos de dados. [19] Os índices armazenam o valor de um atributo específico ou conjunto de atributos, sendo dados estruturados especiais, capazes de armazenar uma pequena porção de dados de uma tabela (casos dos bancos relacionais) ou de uma coleção (casos dos bancos não relacionais), fornecendo uma forma mais eficaz de realizar buscas. [19]

2.3 DATA WAREHOUSE

Data Warehousing fornece para as organizações uma sólida e concisa integração dos dados, permitindo uma análise gerencial estratégica de seus principais processos de negócio. Ele se preocupa em integrar e consolidar as informações de fontes internas, na maioria das vezes heterogêneas, e fontes externas, sumarizando, filtrando e limpando esses dados, preparando-os para análise e suporte à decisão. Algumas características dessa tecnologia podem ser destacadas, tais como [12]:

- Extração de dados de fontes heterogêneas (existentes ou externas);
- Transformação e integração dos dados antes de sua carga final;
- Normalmente requer máquina e suporte próprio;
- Visualização dos dados em diferentes níveis. Os dados do *Data Warehouse* podem ou não ser extraídos para um nível mais específico, os *Data Marts*, e a partir deste para um banco de dados individual;
- Utilização de ferramentas voltadas para acesso com diferentes níveis de apresentação, e;
- Dados somente são inseridos, não existindo atualização, ou melhor, *updates*.

2.3.1 OLAP - *ON-LINE ANALYTICAL PROCESSING*

Trata-se de um conceito de interface com o usuário, que tem como finalidade proporcionar uma análise profunda dos dados sob diversas perspectivas, mantendo uma

estrutura adequada e eficiente dos dados. Por se tratar de um processamento analítico online dos dados, as aplicações OLAP são capazes de solucionar problemas de síntese, análise e consolidação de dados. Como o objetivo da tecnologia OLAP é dar suporte ao usuário com relação a tomada de decisão, o mesmo realiza visualização de dados agregados e não em dados operacionais. Os dados são apresentados em termos de medidas e dimensão, com uma grande parcela das dimensões sendo hierárquica. Torna-se importante ressaltar que as ferramentas OLAP são capazes de navegar pelos dados de um *Data Warehouse*, possuindo uma estrutura adequada tanto para a realização de pesquisas, como para a apresentação de informações. [20]

A OLAP possui algumas características, podendo-se destacar algumas, como cubo, dimensão, hierarquia e membro. A primeira, cubo, especifica uma estrutura que facilita a análise dos dados, visto que os armazena em formato multidimensional. Já a dimensão, refere-se a uma unidade de análise, na qual os dados relacionados são reunidos. Todos os níveis de uma dimensão compõe uma hierarquia, que pode ser balanceada ou não. Por fim, membro pode ser considerado como um subconjunto de uma dimensão. [20]

Com o tempo, os SGBDS tradicionais foram impactados pelo grande volume de dados atual. Cada vez mais, exige-se mais processamento, surgindo então novas tecnologias que mudam a forma como se armazena e estrutura esse tipo de dados, o *Big Data*.

2.4 BIG DATA

Umas de primeiras definições de *Big Data* foi formalizada pelo então analista de dados Doug Laney, pertencente ao Gartner Group, que no começo dos anos 2000, definiu como sendo "ativos de alto volume, velocidade e variedade de informação, que exigem custo-benefício, de formas inovadoras, de processamento de informações para maior visibilidade e tomada de decisão". Pode-se observar, nessa definição, a presença daqueles que ficaram conhecidos como 3Vs (volume, velocidade e variedade). Entretanto, com o avanço das pesquisas, o conceito de *Big Data* evoluiu, passando a ser considerados 5Vs, devido ao acréscimo dos componentes veracidade e valor [21].

A seguir, segue uma breve definição de cada conceito relacionado ao *Big Data* [21]:

Volume: Toda a quantidade de informações diz respeito ao volume. Organizações e indivíduos espalhados pelo mundo, a todo instante, geram uma gama de dados dos mais variados tipos. Esse volume requer uma grande infraestrutura de armazenamento e processamento, o que só foi possível nos dias atuais devido ao surgimento de novas tecnologias.

Velocidade: Os dados, assim que são gerados, devem se movimentar com fluidez, o que requer que sejam tratados logo que possível. Dessa maneira, muitos dispositivos necessitam que tais informações ocorram em tempo real, ou próximo a isso.

Variedade: As informações criadas possuem os mais variados formatos, de dados estruturados, como os provenientes de banco de dados tradicionais, até os não estruturados, como dados de e-mail, áudios, vídeos, textos e transações financeiras.

Veracidade: Refere-se a necessidade de autenticidade dos dados, no momento da coleta. Contudo, ressalta-se que algumas fontes de dados podem não refletir a realidade, como as informações postadas em uma rede social, por exemplo.

Valor: Trata-se do ponto que mais possui destaque nos projetos de *Big Data*, pois, caso não seja possível extrair valor dos dados, ou seja, informações importantes para as análises da organização, nenhum dos conceitos anteriores fará sentido. Torna-se importante que a aplicação *Big Data* tenha definida, de forma clara, uma boa estratégia na extração de valor, a partir das informações estudadas, significando que metas e objetivos precisam ser bem definidos acerca do que se deseja obter e saber.

Big Data é a tecnologia responsável por lidar com um grande volume de informações, apresentadas nas mais variadas formas. Ela trouxe consigo mudanças no modo como os tipos de dados são armazenados, isto é, hoje é necessário compreender toda a arquitetura dos dados, bem como seus tipos, e forma de processamento para garantir eficiência no armazenamento, gerência e acesso a eles. Na seção seguinte, a arquitetura de dados será melhor discutida.

2.5 ARQUITETURA DOS DADOS

Toda a informação que se tem em formato de dados dentro de uma corporação é denominada como dados corporativos. Esses podem ser classificados, de forma básica, como: estruturados e não estruturados. Torna-se necessário dispor de todos os dados da corporação para que se obtenha a visão geral de como estes se relacionam e se encaixam na instituição, sendo esta temática pela qual a arquitetura de dados se encarrega [22].

2.5.1 DADOS ESTRUTURADOS

São dados de estrutura rígida, que possuem organização em bloco (entidades) ou tabelas, sendo agrupadas em associações e classes. Uma entidade pode ter os mesmos atributos e descrições, que podem ser aplicados para todas os demais de um grupo, assim, padronizando-os, contendo mesmo tamanho, formato e ordem. Estes dados são de natureza

repetitiva, previsível, quantificável e numérica, tendo o armazenamento realizado em SGBDs [23].

2.5.2 DADOS SEMIESTRUTURADOS

Em via de regra, esses dados não são armazenados em SGBD e possuem um elevado nível de heterogeneidade, o que não permite a sua generalização em qualquer que seja o tipo de estrutura. Como exemplo temos: *Extensible Markup Language* - XML, *Resource Description Framework* - RDF e *Web Ontology Language* - OWL [23].

2.5.3 DADOS NÃO ESTRUTURADOS

Esses dados não seguem regras e são imprevisíveis, obrigatoriamente não possuem um formato e, tão pouco, sequência. Atualmente, recebem um elevado grau de atenção, visto que estes são disseminados em larga escala por dispositivos móveis, redes sociais e outros dispositivos inteligentes, que rapidamente tornam-se capazes de criarem as mais diversas informações e em grande escala. Dessa maneira, tais dados são considerados como diversificados e não relacionados, tendo como exemplos imagens, textos, vídeos e outros.

2.6 ARQUITETURAS DE PROCESSAMENTO DE DADOS

Nesta seção iremos abordar as arquiteturas de processamento de dados, são elas: processamento de dados em lote, processamento de dados em tempo real e processamento de dados híbridos. Como ressaltado anteriormente, *Big Data* trata de um grande volume de dados, assim, com o avanço da tecnologia, esse quantitativo tende a aumentar em alta escala e passa a necessitar de um poder maior de processamento.

Para que essas informações sejam processadas, tornam-se necessárias arquiteturas estruturadas com a finalidade de oferecer bom desempenho. Dois tipos de arquiteturas podem ser utilizadas nesses casos, a saber: em lote (*batchs*) e em tempo real (*streaming*). O processamento em lote é uma possível solução para resolver conflitos de volume, contudo, o processamento de dados em tempo real vem sendo utilizado para resolver questões quanto a velocidade [24] [25] [26]. Para a finalidade desse estudo, apenas o processamento de dados em lote será apresentado.

2.6.1 PROCESSAMENTO DE DADOS EM LOTE

O processamento em lote é bastante difundido e, geralmente, vem sendo utilizado em aplicações nas quais os dados possuem grande volume e se encaixam em um determinado recorte de tempo. Essa arquitetura utiliza os dados que já se encontram armazenados nos sistemas, porém esse tipo de processamento não leva em conta as informações adicionadas

posteriormente ao processo. Tendo como principal característica a escalabilidade, esse método faz uso de um processamento distribuído paralelo, com a finalidade de lidar com o alto volume e assim, poder alcançar níveis elevados de escalabilidade [24] [25].

Por permitir os mais variados tipos de armazenamento de dados, sejam eles estruturados ou não, surgem alguns desafios quanto ao desenvolvimento de esquemas, fazendo com que todo o processamento de dados se torne de alto custo e gerem algumas dificuldades, como aponta [26].

Com o intuito de contornar estes problemas, e levando em consideração o processamento de dados em lote, foram desenvolvidos ao longo do tempo sistemas, arquiteturas e conceitos especificamente para analisar *Big Data*, os quais consistem de estruturas em camadas, idealizadas para processar e armazenar enormes volumes de dados. Estas plataformas, em conjunto com uma ferramenta capaz de processar e armazenar dados massivos, podem então suportar o processamento de uma grande quantidade de conjuntos de dados usando algoritmos distribuídos [26].

A tecnologia padrão utilizada para o processamento em lote é o *MapReduce*, a qual pode-se destacar algumas vantagens: permitir uma visão simplória e unida dos dados; ser essencialmente escalável; encobrir efetivamente a complexidade do programa de sistema distribuído, o que pode ser bem desafiador considerando a possibilidade de ocorrer falhas em hardwares, flutuações na qualidade de rede e heterogeneidade do equipamento [24].

Contudo, o *MapReduce* também se apresenta de forma limitada, visto que muitas tarefas de análise/mineração em softwares, ou aplicações que ocorrem em tempo real, precisam ser executadas iterativamente ou em várias etapas. O processamento em lote é um processamento confiável, entretanto, ressalta-se que os lotes podem levar mais tempos para serem concluídos, assim, não recomenda-se sua utilização em aplicações de baixa latência [24].

3 TRABALHOS RELACIONADOS

Neste capítulo, são abordados alguns dos principais trabalhos relacionados ao *benchmark* utilizando a tecnologia OLAP - *Online Analytical Processing* e suas etapas, isto é, são descritos como ocorre o processamento dos dados e os modelos de dados utilizados em cada trabalho. A Tabela 3.1 apresenta um resumo esquemático dos trabalhos relacionados, enfatizando os principais pontos de cada trabalho.

Em [27] foi apresentado um modelo de *benchmark* para o ecossistema de *Big Data*, com soluções para a construção do cubo OLAP, tirando proveito de técnicas de pré-agregação de dados para projetar modelos analíticos para *Big Data* OLAP. Nestes modelos, a informação de forma conceitual é organizada em cubos através de dimensões. O *benchmark* aqui proposto tinha como finalidade auxiliar os profissionais de designers de *Big Data* OLAP a decidirem o design de cubo mais adequado a suas tarefas. Para a validação do *benchmark* foi desenvolvido um estudo de caso, no qual tornou-se necessário criar uma implementação específica para uma determinada tecnologia, sendo selecionada a *Apache Kylin* como amostra de abordagem, por esta possuir capacidade de lidar com até dezenas de bilhões de tabelas fatos no tamanho das linhas, assim como pela sua eficácia em lidar com dimensões de cardinalidade e suporte para cenários OLAP em lote e em tempo real.

[28] apresentaram um modelo híbrido de *benchmark* para o processamento em tempo real em ambientes de BI (*Business Intelligence*). O *benchmark* híbrido proposto apresentava uma carga de trabalho mista, denominado como TPC-CH, que visava trabalhar simultaneamente as cargas de trabalhos únicos existentes: TPC-C, para OLTP e TPC-H, para OLAP. As duas cargas de trabalhos executadas por esse *benchmark* foram: uma carga de trabalho transacional baseada no processamento de entrada de pedido do TPC-C e um pacote de consulta OLAP, equivalente a TPC-H, correspondente neste banco de dados de vendas. Por ser oriundo dos dois *benchmarks* TPC mais utilizados, a proposta produziu resultados que podem ser comparáveis a sistemas híbridos e sistemas clássicos de carga de trabalho únicos.

O trabalho de [29] tinha como objetivo central traçar um comparativo entre consultas OLAP em banco de dados relacionais e gráficos, contendo a mesma amostra de dados. Sendo assim, o modelo relacional foi implementado usando o MySQL, enquanto o modelo gráfico foi efetivado por meio do Neo4j. Para fins de comparação de ambos os modelos, foram utilizadas três dimensões: o fato, o tempo de carregamento e a escalabilidade. A dimensão fato analisava se a operação poderia ser feita no modelo, os relatórios de

tempo de carregamento abordavam os desempenhos durante o carregamento de dados na memória, já os relatórios de escalabilidade relatavam os desempenhos do sistema, quando o tamanho dos dados aumentava. O estudo concluiu que as operações OLAP poderiam ser desenvolvidas em ambos os bancos de dados, porém o experimento mostrou que o Neo4j registrava melhor desempenho quando a reagregação de dados era necessária.

[30] realizou um comparativo entre MonetDB e PostgreSQL DBMS, utilizando-se do TPC-H como *benchmark*, tendo como objetivo indicar qual banco de dados possuía a melhor performance para gerenciar um *Data Warehouse* no acesso à informação. A comparação se deu da seguinte forma: os arquivos foram gerados e, em seguida, foi criado em cada SGBD um esquema correspondente às classes utilizadas, tanto para o ambiente floco de neve, quanto para o ambiente estrela. Dessa forma, cada SGBD foi executado sobre os dois ambientes propostos. Em cada ambiente se realizou o teste de desempenho, que consiste de duas operações: o teste de força e o teste de vazão. Assim, utilizou-se o tempo resultante de cada etapa para calcular o desempenho do SGBD. Ao final dos testes e analisando os resultados do *benchmark*, de uma maneira geral, o MonetDB foi o SGBD que apresentou melhor desempenho, tanto na execução isolada dos testes de força e vazão, quanto na execução final em relação ao PostgreSQL.

[31] apresentaram uma extensão para um *benchmark* popular, o *Star Schema* ou SSB, que levava em consideração os modelos NoSQL não relacionais. Nesse modelo, os dados eram gerados em diferentes formatos, evitando o pós-processamento de dados. Com o intuito de explorar a melhor forma do dimensionamento horizontal, os dados podem ser produzidos em um sistema de arquivos distribuído, removendo, portanto, os tamanhos de disco ou partição como limite para o conjunto de dados gerados. Assim, essa extensão de *benchmark* proposta foi projetada não apenas para bancos de dados relacionais, possuindo a capacidade de gerar dados para vários usos, incluindo diferentes sistemas NoSQL. O novo *benchmark* apresentado estendia a geração de dados e geração de consultas, solucionando assim problemas de dimensionamento existentes.

O estudo de [32] aborda a proposta de um novo *benchmark*, com o intuito de fornecer suporte à vários sistemas de armazenamento de dados, sendo capaz de lidar com grandes volumes, tanto em sistemas relacionais, como em sistemas não relacionais, tal qual o NoSQL. O esquema proposto reconhecia vários modelos de dados (topologias em floco de neve, estrela e plana) e vários formatos de dados (CSV, JSON, TBL, XML, entre outros). Como esta proposta ocorria dentro do ecossistema de *Big Data*, envolvia a geração de dados complexos caracterizados dentro da estrutura de "volume, variedade e velocidade"(3Vs). O esquema ainda proporcionava a geração de dados distribuída e paralela. A principal contribuição do *benchmark* proposto, denominado KoalaBench, foi abordar os problemas de sistemas de suporte a decisões baseados em grandes *Data Warehouses*

multidimensionais (*Big Data*).

O estudo de [33] traz uma especificação e análise experimental do *benchmark* OBAS - OLAP, com o objetivo de auxiliar os serviços de análises OLAP. O *benchmark* proposto era uma extensão do *Star Schema Benchmark* (SSB), cuja função é avaliar os serviços de análises e utilização de consultas escritas na linguagem MDX (*Multidimensional Expressions*). A especificação do *benchmark* mostrava as configurações de execução, como utilização de cache, localidade, paralelismo, serviços e catálogos. O estudo definiu os esquemas relacional e multidimensional correspondentes, e as consultas SQL (*Structured Query Language*) do SSB traduzidas para MDX, além de serem adicionadas novas consultas que avaliam funções OLAP na linguagem MDX. As métricas de avaliação da proposta foram: tempo de resposta, taxas de execução e confiabilidade. Uma ferramenta foi desenvolvida com a finalidade de realizar os testes de execução do OBAS, e assim analisar o desempenho e os serviços de análises OLAP, por meio de XMLA. Também foram realizados testes com base real, através de tradução de consultas e adaptação do esquema relacional e multidimensional, com a conversão dos dados. Torna-se importante ressaltar que essa adaptação pode ser aplicada a qualquer base de dados, necessitando apenas atender os requisitos mínimos para a criação do catálogo.

[34] apresenta uma abordagem que mapeia operações OLAP típicas para SPARQL e uma ferramenta denominada ASPG, para gerar automaticamente consultas OLAP de *Linked Data* do mundo real. Avaliou-se a ferramenta construída, ASPG, através de um *benchmark* denominado DBOB, elaborado a partir de um *endpoint online* da DBpedia, que foi usada para avaliar o tempo de processamento de consultas OLAP SPARQL. A ferramenta desenvolvida ASPG possui apenas um BGP (gerador de padrões gráficos básicos) e funções agregadas selecionadas anteriormente, enquanto as consultas do mundo real também poderiam empregar filtros e subconsultas. Como resultado, as consultas ASPG representavam apenas algumas necessidades analíticas básicas. O estudo deixa como contribuição a necessidade de realização de um trabalho futuro no qual a ASPG possa gerar vários BGPs e sub-consultas, cobrindo assim uma gama mais ampla de operações de análises.

A pesquisa de [35] apresenta todo o processo de determinação de métricas para mensurar resultados e a criação de um *microbenchmark* de consultas. Os experimentos nesse trabalho utilizam dados de trajetórias da Microsoft (T-Drive), assim, determina em quais situações cada SGBD apresenta melhor desempenho. Esses resultados são validados através do *microbanchmark*, em conjunto com as métricas estabelecidas pelo autor, que são: quantidade de operações de entrada e saída, uso de CPU e tempo de resposta de consulta. Como propostas futuras, o autor destaca executar o *microbenchmark* em um *dataset* com muitas tabelas correlacionadas; com o mesmo *dataset* utilizar índices espaciais;

realizar o trabalho utilizando os outros índices disponíveis para o PostgreSQL; re-executar os experimentos adicionando a métrica de uso de memória; realizar a análise comparativa com operações do tipo junção.

Tabela 3.1 – Resumo Esquemático dos Trabalhos Relacionados sobre *Benchmark* de consultas OLAP

Artigo	Domínio	Ferramentas	Tipo Processamento	*Benchmark*
[27]	*Big Data*	Apache Kylin, Hadoop HDFS, ETL, Data Warehouse, NoSQL, Kafka, Star Schema (Hive)	Lote, Tempo Real	Conjunto de métricas, de 30 consultas, um modelo de dados, foco em consultas analíticas de subsegundos para retomada ou agregação de dados com recursos de *Big Data*
[28]	Business Intelligence	ETL, MonetDB, VoltDB,	Lote, Tempo Real	Composto pelo esquema e transações TPC-C não modificados e uma versão adaptada das consultas TPC-H, ambos os *benchmarks* modelam negócios que "devem gerenciar, vender ou distribuir um produto ou serviço."
[29]	Bancos relacionais e gráficos	MySQL, Neo4j	Lote, Tempo Real	Operações OLAP foram realizadas usando consultas definidas em SQL para MySQL e em Cypher para Neo4j, três dimensões foram consideradas para a avaliação dos resultados
[30]	Data Warehouses	MonetDB, Drive JDBC, DBGen, QGen PostgreSQL	-	*Benchmark* TPC-H. Software DBgen para geração de dados, software Qgen para geração de consultas, ambos do TPC. Modelagem adpatada através do modelo floco de neve e estrela. Testes de forças e testes de vazão.
[31]	NoSQL	ETL, HBase, MongoDB, DBLoad	-	*Benchmark* sendo uma extensão do SSB. Softwares para a geração de dados DBGen e QGen para geração de consultas, DBLoad para migração dos dados.

[32]	*Big Data*	Hadoop HDFS, NoSQL, KoalaBench, Data Warehouse	Lote, Tempo Real	*KoalaBench* derivado do *benchmark* TPC-H, adaptado para suportar tecnologias de *Big Data* como NoSQL, gera dados consistentes com 3 modelos lógicos: modelo floco de neve, modelo estrela e modelo de dados simples. Gerador de consultas QGEN.
[33]	Bancos relacionais e multi-dimensional	MDX, XMLA	Lote, Tempo Real	Tem por objetivo avaliar alguns aspectos de desempenho dos serviços de análises, consultas MDX, três tipos de métricas de desempenho: tempo de resposta (*response time* – RT), taxa de execução ou vazão (*throughput*) e confiabilidade
[34]	SPARQL	ASPG	Lote, Tempo Real	Criação da ferramenta ASPG para avaliar consultas OLAP, validada por um *benchmark* chamado DBOB, capaz de gerar consultas não trivais, comparou as consultas do DBOB com as consultas OLAP4LD-SSB.
[35]	T-Drive	ASPG	Lote	Desenvolvido um *microbenchmark* com a finalidade de comparar o desempenho entre um modelo de dados NoSQL e um modelo de dados Relacional utilizando um conjunto de dados de trajetórias da Microsoft.

3.1 DISCUSSÃO

Com a revisão de literatura, foi possível extrair dos trabalhos, métricas de avaliação para esta pesquisa, além de ser possível estabelecer um conjunto de consultas para a base de testes do experimento e o embasamento para a utilização de um *benchmark* nesse trabalho. Contudo, dentre os trabalhos revisados, não foram identificados estudos que trabalhassem com o domínio do SPED, ou de algum documento em específico como, por exemplo, a nota fiscal eletrônica, objeto desta pesquisa, no qual compara-se dois modelos de dados. Evidencia-se então, que este estudo pode ser considerado como um tema interessante de exploração.

4 PROPOSTA DE TRABALHO

Neste capítulo serão apresentadas as etapas que compuseram o desenvolvimento da solução proposta neste trabalho.

4.1 AMBIENTE DOS EXPERIMENTOS

Um ambiente de trabalho foi preparado para a realização dos experimentos. Inicialmente, foi definida a base de dados que seria utilizada nesta pesquisa e os requisitos da máquina que hospedaria os bancos de dados. A base de dados selecionada foi a base de NF-e pertencente ao órgão da Secretaria de Estado da Fazenda da Paraíba - SEFAZ/PB, sendo esta escolhida por tratar de uma base confiável, em relação ao documentado estudado (NF-e). Ressalta-se que todos os dados utilizados nessa pesquisa encontravam-se anonimizados.

Foi adquirida uma máquina virtual e um disco virtual para a alocação da base de dados em ambos bancos de dados, na plataforma Azure da *Microsoft*. A configuração da máquina virtual trata-se: Sistema Operacional Linux Debian 10.10, com 1GB de memória, e processador Dual-Core. A Figura 4.1 mostra as propriedades da Máquina Virtual, enquanto a Figura 4.2 mostra as propriedades do Disco Virtual, que por sua vez conta com um armazenamento de 30GB. Todos os testes de consultas foram realizados da mesma forma, em ambos os modelos, por meio das configurações da Máquina Virtual acima citada.

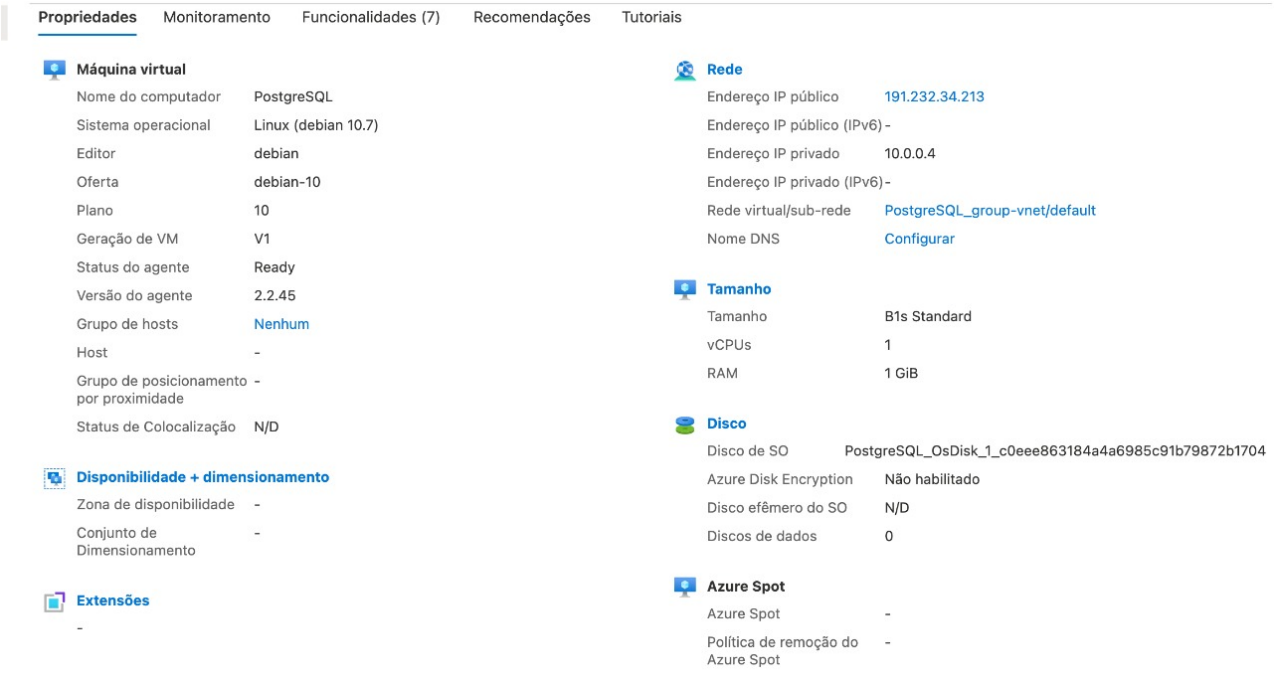

Figura 4.1 – Propriedades da Máquina Virtual
Fonte: Próprio Autor

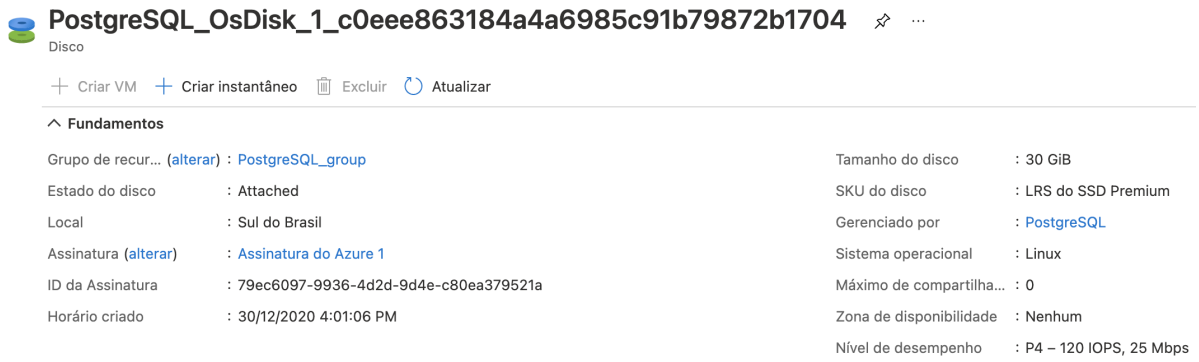

Figura 4.2 – Propriedades do Disco Virtual
Fonte: Próprio Autor

4.2 MÉTRICA DE AVALIAÇÃO

Trabalhos anteriores abordavam diferentes métricas de avaliação, em [33] foram utilizadas as seguintes métricas: tempo de resposta, taxas de execução e confiabilidade, já no estudo de [27], as métricas do *benchmkark* foram: tempo de construção, sucesso de construção, tamanho do Cubo, latência de consulta e cobertura de modelo. Dessa forma, tomando como base os trabalhos relacionados, nesta pesquisa, por se tratar de um comparativo de consultas entre modelos de dados, utilizou-se como métrica de avaliação o tempo de resposta das consultas, em cada modelo proposto, com o *benchmark*, que será mostrado a seguir.

4.3 BENCHMARK DE CONSULTA

Foi desenvolvido um *benchmark* de consulta com a finalidade de mensurar o desempenho dos SGBDs, juntamente com o auxílio da métrica definida na Subseção 4.2. Dessa maneira, três operações foram implementadas: *full scan*, *specific scan* e *exact match*.

- *Full Scan*: Refere-se a uma varredura completa, sendo responsável pelo comparativo em relação a consulta realizada em uma tabela, contemplando todos os campos e atributos que pertencem a mesma.
- *Specific Scan*: Refere-se a uma varredura específica, sendo responsável por retornar um atributo específico da tabela determinada na consulta.
- *Exact match*: Trata-se de uma combinação. Nesse caso, a consulta tem como objetivo final retornar atributos de uma mesma tabela, na qual a condição de combinação seja referente ao atributo.

4.4 SOFTWARE EXTRATOR DE DADOS DA NF-E

Foi desenvolvido um *software*, na linguagem de programação *Python*, que tem por objetivo realizar a leitura de um arquivo *.xsd*, que contém as informações da Nota Fiscal

Eletrônica. Foram extraídos os campos e os atributos necessários para a construção de um dicionário de dados. Ao final da execução, o programa exibiu um esquema com campos e colunas que especificam os dados extraídos automaticamente da NF-e. O objetivo desse esquema foi dar suporte para a criação da tabela de consultas, utilizada nos testes com os dois modelos de dados. O resultado final do arquivo gerado pelo programa pode ser visualizado no Apêndice 7 A.

Através do esquema fornecido pelo software, foi possível montar um modelo com um conjunto de 30 consultas que serviram como base para a realização dos testes no experimento. Esse conjunto de consultas foi baseado no trabalho de [27], que contemplava um total de 30 consultas, e do trabalho de [36], que contemplava 50 consultas. O conjunto de consultas selecionadas para este trabalho está descrito na Tabela 4.1, que também evidencia qual operador do *benchmark* a consulta representa.

Tabela 4.1 – Tabela de consultas do Benchmark

Número	Consulta	PostgreSQL	MongoDB	Operador
Categoria por campos:				
1	Consultar todos os campos de uma determinada tabela	SELECT * FROM nf.tb_nfe;	db.nfe.find( )	*Full Scan*
2	Consultar os campos por ordenação	SELECT * FROM nf.tb_nfe ORDER BY id;	db.nfe.find().sort(_id: 1)	*Full Scan*
3	Consultar o valor do campo imposto	SELECT valor FROM nf.tb_imposto;	db.nfe.find(, '0.produtos.imposto': 1)	*Specific Scan*
4	Consultar a natureza da operação da NFe	SELECT natureza_operacao FROM nf.tb_nfe;	db.nfe.find(,'0.natOp': 1)	*Specific Scan*
5	Consultar os campos com os valores do emitente	SELECT * FROM nf.tb_emitente;	db.nfe.find(,'0.emitente': 1)	*Full Scan*
6	Consultar o campo DANFE	SELECT * SELECT danfe FROM nf.tb_nfe;	db.nfe.find(,'0.chNFe': 1)	*Specific Scan*
7	Consultar os campos das alíquotas	SELECT aliquota FROM nf.tb_imposto;	db.nf.tb_imposto.find(,"aliquota": 1);	*Specific Scan*
8	Consultar os campos de uma tabela pelo id	SELECT id FROM nf.tb_nfe;	db.nfe.find(_id: 1)	*Full Scan*
Categoria por datas:				
9	Consultar data de saída da NFe	SELECT data_saida FROM nf.tb_nfe;	db.nfe.find(,'0.dhSaiEnt': 1))	*Specific Scan*
10	Consultar NFe por data (filtrar mais recentes e mais antigas)	SELECT * FROM nf.tb_ORDER BY data_emissao DESC;	db.nfe.find(). sort('0.dhEmi': 1)	*Full Scan*

Subcategorias por data:				
11	Consultar por dia	SELECT * FROM nf.tb_nfe WHERE data_saida::date = '2018-01-01'	db.nfe.find('0.dhEmi': '2020-09-19 T00:00:00-03:00')	*Exact Match*
12	Consultar por semana	SELECT * FROM nf.tb_nfe WHERE data_saida BETWEEN '2020/09/10' AND '2020/09/17'	db.nfe.find('0.dhEmi': $gte: 2020-09-10, $gte: 2020-09-17)	*Exact Match*
13	Consultar por mês	SELECT * FROM nf.tb_nfe WHERE data_saida BETWEEN '2020/09/10' AND '2020/10/10';	db.nfe.find('0.dhEmi': $gte: 2020-09-10, $gte: 2020-10-10)	Ex*act* *Match*
14	Consultar por ano	SELECT * FROM nf.tb_nfe WHERE data_saida BETWEEN '2020/01/01' AND '2021/01/01';	db.nfe.find('0.dhEmi': $gte: 2020-01-01, $gte: 2021-01-01)	*Exact Match*
Consultas específicas para as notas:				
15	Consultar o campo CNPJ da tabela emitente	SELECT cnpj FROM nf.tb_emitente;	db.nfe.find(,'0. emitente.CNPJ': 1)	*Specific Scan*
16	Consultar o campo produto da tabela produto	SELECT produto FROM nf.tb_produto;	db.nfe.find(,'0.produtos. prod. xProd': 1)	*Specific Scan*
17	Consultar o nome fantasia do emissor da Nfe	SELECT nome_fantasia FROM nf.tb_emitente;	db.nfe.find(,'0.emitente. xNome': 1)	*Specific Scan*
18	Consultar a quantidade do volume	SELECT quantidade FROM nf.tb_volume;	db.nfe.find(,'0.produtos. prod.qVol': 1)	*Specific Scan*
19	Consultar valor dos produtos	SELECT produtos FROM nf.tb_totais;	db.nfe.find(,'0.produtos. prod.vUnTrib': 1)	*Specific Scan*

20	Consultar a tag chave de cada nota	SELECT chave FROM nf.tb_nfe;	db.nfe.find(,'0.chNFe': 1)	*Specific Scan*
21	Consultar a versão da nota	SELECT versao_xml FROM nf.tb_nfe;	db.nfe.find(,'0.verProc': 1)	*Specific Scan*
22	Consultar a nota através de filtro de destino da nota (orgão estadual)	SELECT id_destino FROM nf.tb_nfe;	db.nfe.find(, '0.idDest': 1)	*Specific Scan*
23	Consultar Modelo do Documento Fiscal (55 ou 65)	SELECT modelo FROM nf.tb_nfe WHERE modelo = '55'; SELECT modelo FROM nf.tb_nfe WHERE modelo = '65';	db.nfe.find('0.mod': '55') , db.nfe.find('0.mod': '65')	*Extract Match*
24	Consultar Série do Documento Fiscal	SELECT serie FROM nf.tb_nfe;	db.nfe.find(,'0.serie': 1)	*Specific Scan*
25	Consultar Número do Documento Fiscal	SELECT numero FROM nf.tb_nfe;	db.nfe.find(,'0.cNF': 1)	*Specific Scan*
26	Consultar o código da unidade federativa da emissão da nota	SELECT cod_uf FROM nf.tb_nfe;	db.nfe.find(,'0.cUF': 1)	*Specific Scan*
27	Consultar a finalidade da nota	SELECT finalidade FROM nf.tb_nfe	db.nfe.find(,'0.tpNF': 1)	*Specific Scan*
28	Consultar o município do destinatário	SELECT municipio FROM nf.tb_destinatario;	db.nfe.find(, '0.destinatario. cMunF': 1)	*Specific Scan*
29	Consultar o munícipio do emissor	SELECT municipio FROM nf.tb_emitente	db.nfe.find(,'0.emitente. cMunF': 1)	*Specific Scan*
30	Consultar o ICMS do Município	SELECT municipio_icms FROM nf.tb_destinatario;	db.nf.tb_destinatario.find (, "municipio_icms": 1);	*Specific Scan*

4.5 BASE DE DADOS

Foi utilizado um arquivo de documento que continha em sua estrutura cerca de 200 mil NF-e em sequência. Contudo, em virtude da necessidade de submeter notas individuais na base de dados do PostgreSQL e do MongoDB, realizou-se a extração dessas notas, de maneira que as mesmas ficassem separadas uma por uma em arquivos no formato de *.xml*, sendo desenvolvido

um *script shell* para a realização desse procedimento. Considerando-se os trabalhos relacionados, definiu-se uma base contendo 50 mil e 100 mil notas, armazenando-as em cada um dos bancos de dados, relacional e NoSQL.

Após a realização desse processo, efetuou-se a implementação nos modelos de dados. Dessa maneira, para o modelo relacional, foi escolhido o SGBD PostgreSQL na Versão 12.4 e para o modelo NoSQL selecionou-se o SGBD MongoDB na Versão 4.4.3. O processo de implementação dos modelos, assim como o armazenamento dos dados nestes, será melhor explicitado a seguir.

4.6 IMPLEMENTAÇÃO DO MODELO DE DADOS RELACIONAL

Na implementação do modelo relacional, neste trabalho se fez necessário o desenvolvimento de um modelo Entidade-Relacionamento (ER) da NF-e. Este modelo descreveu as relações, as tabelas e os campos da NF-e, onde, posteriormente, tomando como base este modelo ER, foi desenvolvido um schema.sql, que possui toda a estrutura do banco de dados. Para que fosse possível o armazenamento das notas nos bancos de dados, foi desenvolvido um *script* que tem como objetivo efetuar o armazenamento das notas nos bancos de dados. A Figura 4.3 mostra o modelo ER desenvolvido para o modelo de dados relacional.

O modelo acima obedece as regras de normalização, segundo as quais os relacionamentos 1-1 teve somente um item, como evidencia-se na seguinte relação: a Tabela *tb_nfe* se relaciona com a *tb_totais*, de forma que cada NF-e terá um total, e cada total contempla uma única NF-e. O relacionamento 1-N, 1 para muitos, é observado para o estudo de caso deste trabalho, onde cada NF possui vários itens. No entanto, é importante ressaltar que cada item estava em uma relação única para cada NF-e, de forma que, ao ocorrer um relacionamento N-N, onde cada item estava relacionada a uma única NF-e. Quando houve relacionamento do tipo N-N, se fez necessária a criação de uma terceira tabela. Isso é evidenciado no caso da NF-e com Imposto, onde cada NF-e pode conter vários impostos e cada imposto está presente em várias NFes, tornando-se necessária a criação de uma tabela intermediária, como foi o caso da criação da Tabela *nf_imposto*.

4.7 MODELO DE DADOS ORIENTADO À DOCUMENTOS

A implementação do modelo de dados orientado a documentos se deu a partir de um SBGD NoSQL. Esta pesquisa desenvolveu a implementação no MongoDB e este SGBD, por estar isento de esquema, não exigiu que fosse criada uma estrutura prévia para a alocação de dados, fato que permite que a estrutura possa ser alterada de forma livre em execução. Neste experimento, foram alocadas todas as notas em uma só coleção, ou seja, cada nota passa a ser um documento dentro do MongoDB. Como o modelo do MongoDB não exige uma estrutura prévia para o armazenamento das NF-e, desenvolveu-se um *script.php* para a realização do armazenamento das notas no banco de dados.

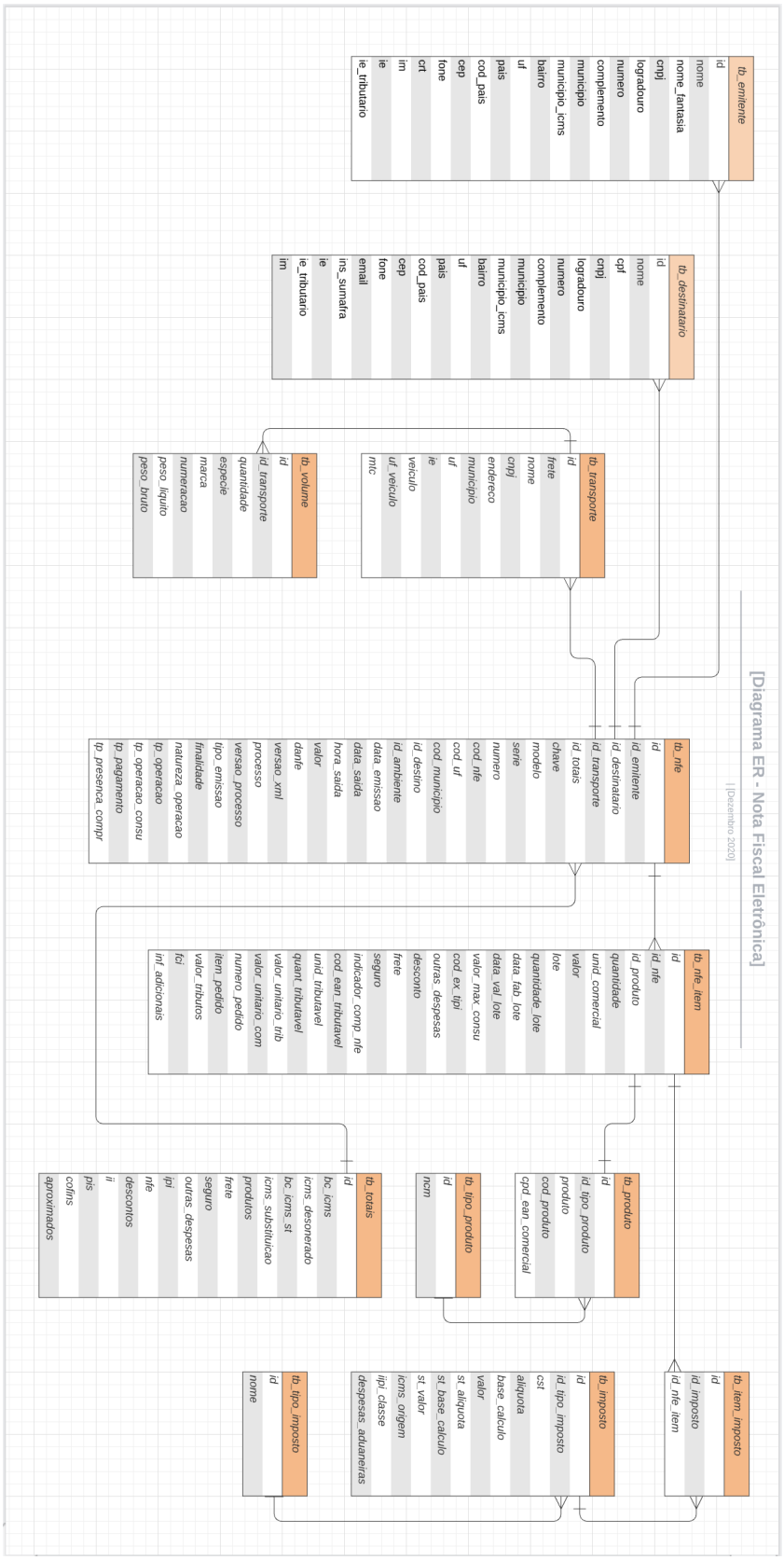

Figura 4.3 – Modelo ER da Nota Fiscal Eletrônica
Fonte: Próprio Autor

4.8 COLETA DOS RESULTADOS

Após a aplicação do *benchamrk*, os resultados foram coletados através de comandos específicos de cada SGBD, no qual os comandos foram executados localmente, no ambiente remoto da nuvem/VMs. Para coletar os dados do PostgreSQL, foi utilizado o comando *Explain Analyze*, que será descrito melhor abaixo:

- *Explain Analyze* - Este parâmetro não só fornece informação do que foi planejado, como executa, de fato, o comando. O tempo total decorrido gasto em cada nó do plano (em milissegundos) e o número total de linhas retornadas são adicionados ao que é apresentado. Torna-se útil para avaliar se as estimativas do planejador estão próximas ao resultado final.

  Nos próximos itens serão descritas as informações de retorno ao executar esse parâmetro.

- *Startup Cost* - O custo de partida estimado. O tempo gasto antes da varredura da saída poder começar como, por exemplo, o tempo para fazer a ordenação em um nó de ordenação.

- *Total Cost* - O custo total estimado.

- *Estimated Rows* - Número de linhas de saída estimado para este nó do plano.

- *Estimated Average Row Size* (bytes) - Largura média estimada (em bytes) das linhas de saída para este nó do plano.

No caso do MongoDB, foi utilizada a expressão *Explain executionStats*, que é um parâmetro que fornece dados sobre o plano de consulta. É retornado um documento descrevendo o processo e os índices utilizados para retornar a consulta. Nos itens a seguir serão descritas as informações fornecidas por esse operador. A Figura 4.4 exemplifica o comando *Explain executionStats* sendo executado, e retornando os dados.

- *queryPlanner.winningPlan.stage* - É exibido o *COLLSCAN*. Isso significa uma varredura de coleção.

- *Collection scans* - Neste caso, significa que o MongoDB teve que escanear documento por documento da coleção para obter os resultados.

- *executeStats.nReturned* - Mostra o número de documentos retornados, de acordo com a consulta.

- *executionTimeMillis* - Tempo total gasto em milissegundos para a execução da consulta.

- *executeStats.totalKeysExamined* - Indica o valor 0 ou 1, 0 para informar que a consulta está utilizando índice e 1, para informar caso ela não esteja usando.

- *executeStats.totalDocsExamined* - Indica o total de documentos que o MongoDB teve que digitalizar para informar o número correspondente de documentos, de acordo com a consulta realizada.

```
> db.nfe.find().explain("executionStats")
{
        "queryPlanner" : {
                "plannerVersion" : 1,
                "namespace" : "db.nfe",
                "indexFilterSet" : false,
                "parsedQuery" : {

                },
                "winningPlan" : {
                        "stage" : "COLLSCAN",
                        "direction" : "forward"
                },
                "rejectedPlans" : [ ]
        },
        "executionStats" : {
                "executionSuccess" : true,
                "nReturned" : 50000,
                "executionTimeMillis" : 1757,
                "totalKeysExamined" : 0,
                "totalDocsExamined" : 50000,
                "executionStages" : {
                        "stage" : "COLLSCAN",
                        "nReturned" : 50000,
                        "executionTimeMillisEstimate" : 1613,
                        "works" : 50002,
                        "advanced" : 50000,
                        "needTime" : 1,
                        "needYield" : 0,
                        "saveState" : 69,
                        "restoreState" : 69,
                        "isEOF" : 1,
                        "direction" : "forward",
                        "docsExamined" : 50000
                }
        },
        "serverInfo" : {
                "host" : "3e04b56c812e",
                "port" : 27017,
                "version" : "4.4.3",
                "gitVersion" : "913d6b62acfbb344dde1b116f4161360acd8fd13"
        },
        "ok" : 1
}
```

Figura 4.4 – Comando Explain executionStats sendo executado no MongoDB
Fonte: Próprio Autor

Os testes realizados no experimento ocorreram utilizando o recurso sem índice e com índice em ambos os SGBDs, tanto para a base de dados com 50 mil notas quanto para a base de dados com 100 mil notas.

5 RESULTADOS

Ao término dos experimentos, os dados foram coletados e as amostras resultantes serviram para a geração dos gráficos. Nesta pesquisa, inicialmente as consultas em ambos os bancos de dados (MongoDB e PostgreSQL) foram realizadas sem índice, com a sua utilização ocorrendo em seguida, tal como é observado na divisão dos resultados. Foi necessário realizar as operações de forma repetitiva, em virtude da variabilidade de tempo de resposta a uma mesma consulta, sendo assim necessário obter-se a média do tempo de resposta. Neste estudo, foram realizadas 100 execuções para cada banco de dados, sendo calculada a média dos resultados. Nas subseções a seguir, serão exibidos os gráficos mostrando os resultados gerados em cada SGBD, no qual o tempo de consulta foi realizado em em milissegundos para a sua execução.

5.1 BUSCAS SEM A UTILIZAÇÃO DE ÍNDICE

5.2 *FULL SCAN*

O *full scan* é um operador que realiza uma varredura completa, seja em uma base de dados, seja apenas em uma tabela ou em coleções de documentos, tal como o MongoDB. Assim, esse operador foi utilizado para realizar a varredura completa em duas tabelas do PostgreSQL, a saber: tabela nfe contendo 29 campos e tabela emitente com 19 campos.

5.2.1 UTILIZANDO UM LOTE DE 50 MIL NOTAS:

A Figura 5.1 exibe o *full scan*, através do *benchmark* proposto. Ao término de todas as execuções, foi calculada uma média dos resultados da métrica avaliada, ou seja, do tempo de resposta das consultas. O gráfico evidencia a diferença no desempenho e performance entre os dois SGBD avaliados, pois ao ser submetido a diferentes consultas (Tabela 4.1), o MongoDB obteve um tempo de resposta inferior ao PostgreSQL em todos os casos, com destaque para a Consulta 10 (consultar NFe por data), na qual o PostgreSQL apresentou um comportamento mais lento, comparado ao MongoDB, registrando uma diferença de 11.260 milissegundos.

Ressalta-se que essa busca realiza a leitura em cada bloco da tabela. O PostgreSQL leva desvantagem nesse quesito, pois possui uma alta taxa de blocos percorridos e, consequentemente, desses blocos serem gravados no disco, fato decorrente do SGBD buscar páginas simultâneas no disco de uma vez. Ocorre aqui um acesso ao disco que leva ao pior desempenho do PostgreSQL. A Tabela 5.1 mostra o tempo mínimo e o máximo de execução de cada SGBD.

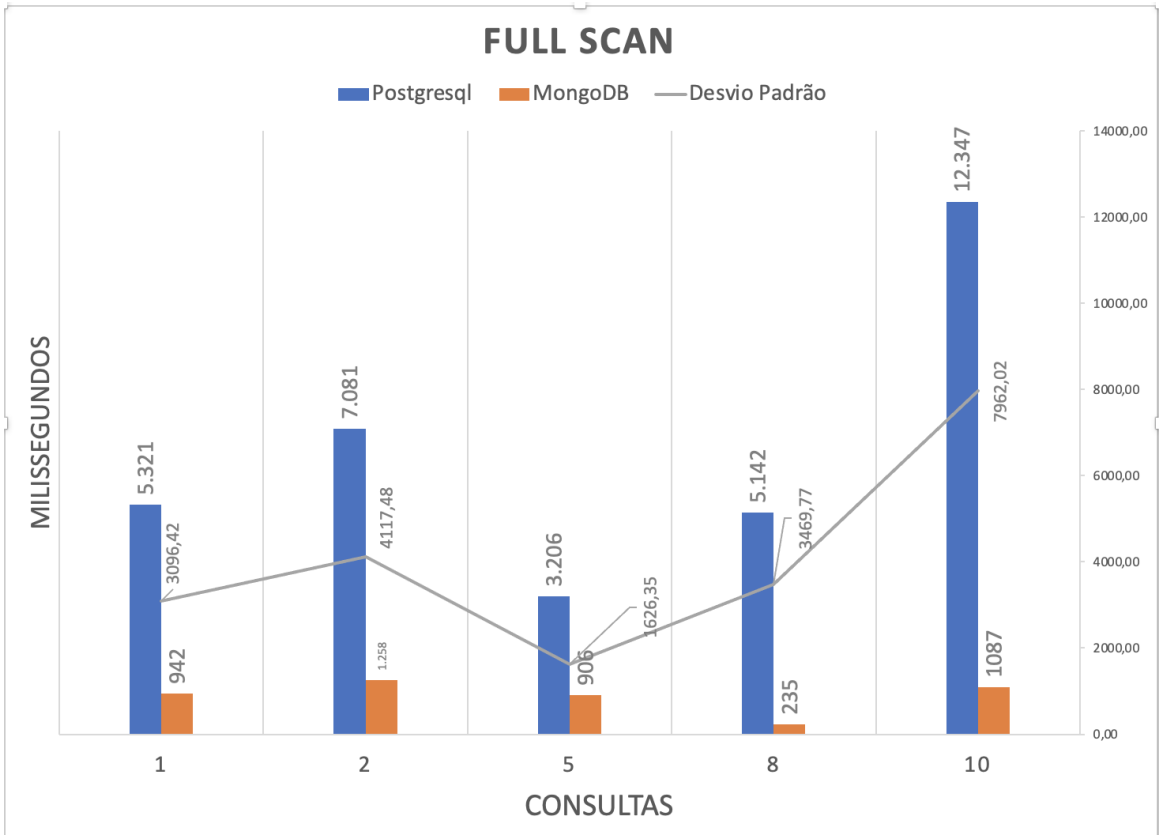

Figura 5.1 – Gráfico por Consulta do operador *Full Scan*, cuja numeração segue descrita na Tabela 4.1

Fonte: Próprio Autor

Tabela 5.1 – Tabela de mínimo e máximo do *Full Scan*

	PostgreSQL		MongoDB	
	Mínimo	Máximo	Mínimo	Máximo
Tempo de Execução (ms)	4263	19800	37	1338

Os dados evidenciados pela Tabela 5.1 reforçam o resultado de que o desempenho do MongoDB é superior ao do PostgreSQL, uma vez que o tempo mínimo de resposta do PostgreSQL é cerca de 115 vezes maior que o do outro SGBD, assim como o tempo máximo do MongoDB é cerca de 15 vezes menor que o do PostgreSQL. Os planos de consultas que representam o mínimo e o máximo nesse operador podem ser visualizados no Apêndice 7.1.

5.2.2 UTILIZANDO UM LOTE DE 100 MIL NOTAS:

A Figura 5.2 exibe o gráfico que mostra os resultados após a execução da base de dados contendo 100 mil notas fiscais eletrônicas. É possível notar que, mesmo com o dobro de notas fiscais eletrônicas na base de dados, o PostgreSQL tende a permanecer de forma mais lenta com relação ao MongoDB ao executar as consultas, como fica evidenciado na Consulta 10 (consultar

Tabela 5.2 – Tabela de mínimo e máximo do *Full Scan* com 100 mil notas fiscais eletrônicas

	PostgreSQL		MongoDB	
	Mínimo	Máximo	Mínimo	Máximo
Tempo de Execução (ms)	8427	40421	102	3692

NFe por data) na qual é registrada uma diferença de 24.000 milissegundos entre os SGBDs.

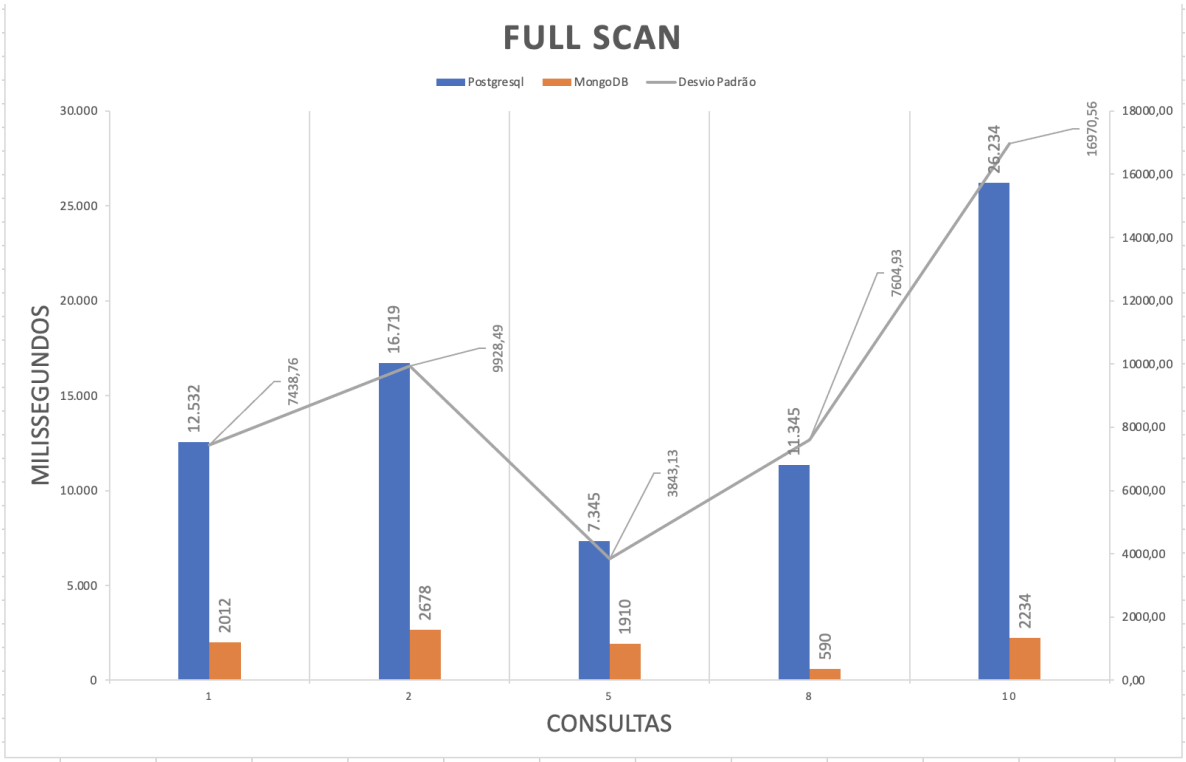

Figura 5.2 – Gráfico por Consulta do operador *Full Scan*, cuja numeração segue descrita na Tabela 4.1

Fonte: Próprio Autor

Os dados mostrados na Tabela 5.2 evidencia a eficácia do MongoDB com relação ao PostgreSQL, no qual o desempenho do PostgreSQL é cerca de 83 vezes maior que o do outro SGBD, bem como no o tempo máximo do MongoDB é cerca de 11 vezes menos que o do PostgreSQL.

5.3 *SPECIFIC SCAN*

O segundo operador a ser executado foi o *Specific Scan*, responsável por uma varredura específica, isto é, ele percorre toda a tabela com a finalidade de retornar apenas o campo buscado na consulta.

5.3.1 UTILIZANDO UM LOTE DE 50 MIL NOTAS:

Após as execuções da operação, foi calculada a média dos resultados e expostos no gráfico, evidenciado na Figura 5.3. Dessa maneira, observa-se que o desempenho do MongoDB é bem superior ao do PostgreSQL, como na Consulta 7, referente a consulta dos campos das alíquotas na tabela emitente, onde a diferença foi de 26.599 milissegundos.

Contudo, em outros casos, o desempenho torna-se quase que equivalente, tal como na Consulta 18 (consultar a quantidade do volume na tabela do volume), na qual apenas 208 milissegundos separam o desempenho de um SGBD do outro. Tal resultado pode ser atrelado ao fato de que, como não possui índice, o MongoDB tem sempre que percorrer a coleção por inteiro, o que acaba por ser custoso em alguns momentos, embora o PostgreSql tenha um processo semelhante com relação às tabelas.

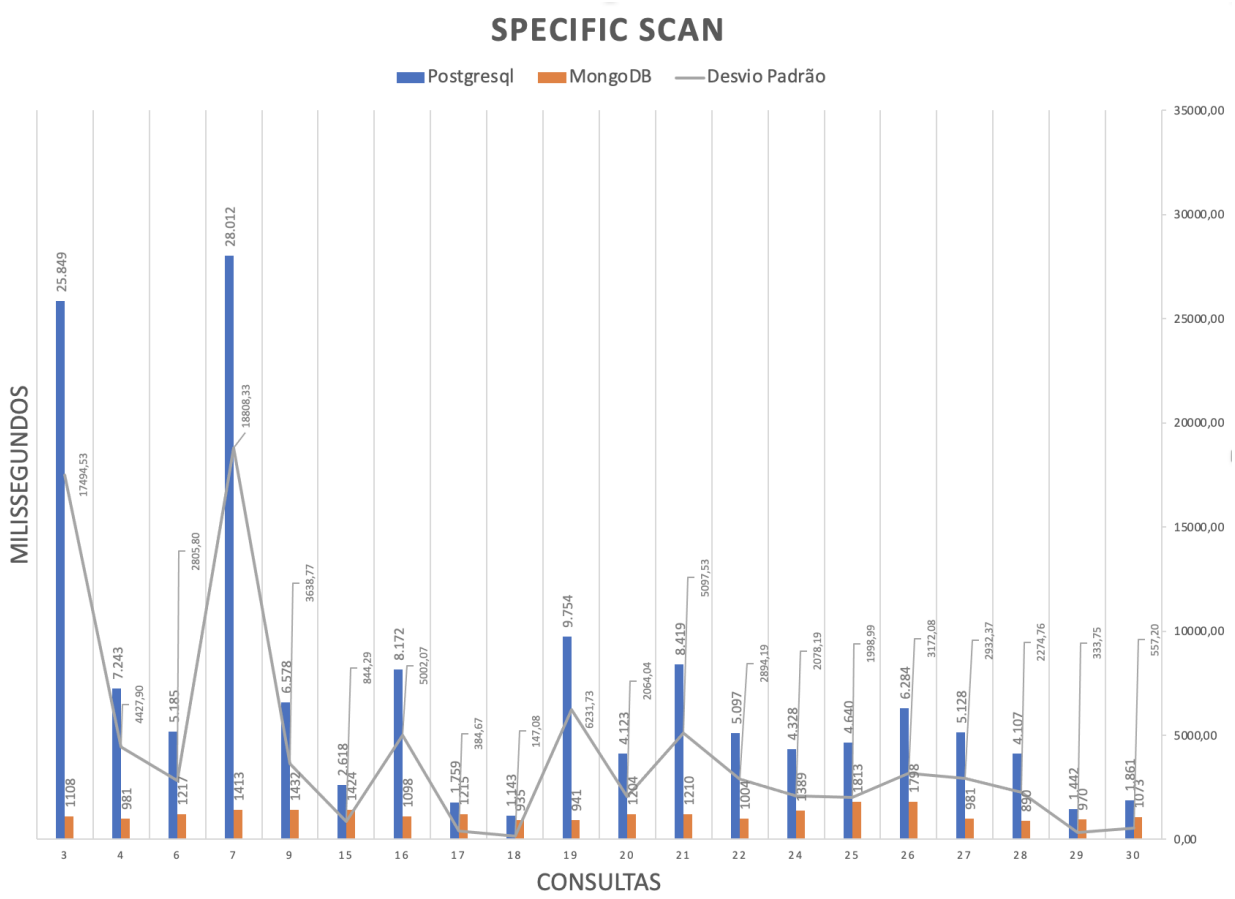

Figura 5.3 – Gráfico por Consulta do operador *Specific Scan*, cuja numeração segue descrita na Tabela 4.1

Fonte: Próprio Autor

A Tabela 5.3 mostra o mínimo e o máximo do tempo de execução em milissegundos de cada SGBD neste operador. Os planos de consultas que representam o mínimo e o máximo nesse operador podem ser visualizados no Apêndice 7.2.2.

A proximidade de desempenho entre os dois SGBD, em alguns pontos desse segundo

Tabela 5.3 – Tabela de mínimo e máximo do *Specific Scan*

	PostgreSQL		MongoDB	
	Mínimo	Máximo	Mínimo	Máximo
Tempo de Execução (ms)	1655	37849	700	1834

operador, torna-se mais evidente na Tabela 4.1, visto que o desempenho do MongoDB foi apenas duas vezes inferior ao do PostgreSQL, diferentemente de quando comparado com os dados do primeiro operador. Os planos de consultas que representam o mínimo e o máximo nesse operador podem ser visualizados no Apêndice 7.3.2.

5.3.2 UTILIZANDO UM LOTE DE 100 MIL NOTAS:

A Figura 5.4 exibe o gráfico que mostra os resultados após a execução da base de dados contendo 100 mil notas fiscais eletrônicas para o segundo operador, o *Specific Scan*, nesse caso é possível observar o desempenho do MongoDB sendo superior ao do PostgreSQL de forma acentuada, como evidenciado na Consulta 7 (Consultar os campos das alíquotas), onde a diferença foi de 54121 milissegundos. A Tabela 5.4 mostra os mínimos e os máximos do tempo de execução em milissegundos de cada SGBD nesse operador com uma carga de 100 mil notas fiscais eletrônicas.

Examinando a Tabela 5.4 é possível verificar que existe uma proximidade de desempenho com relação aos dois SBGDs, visto que o desempenho do MongoDB foi apenas duas vezes inferior ao do PostgreSQL.

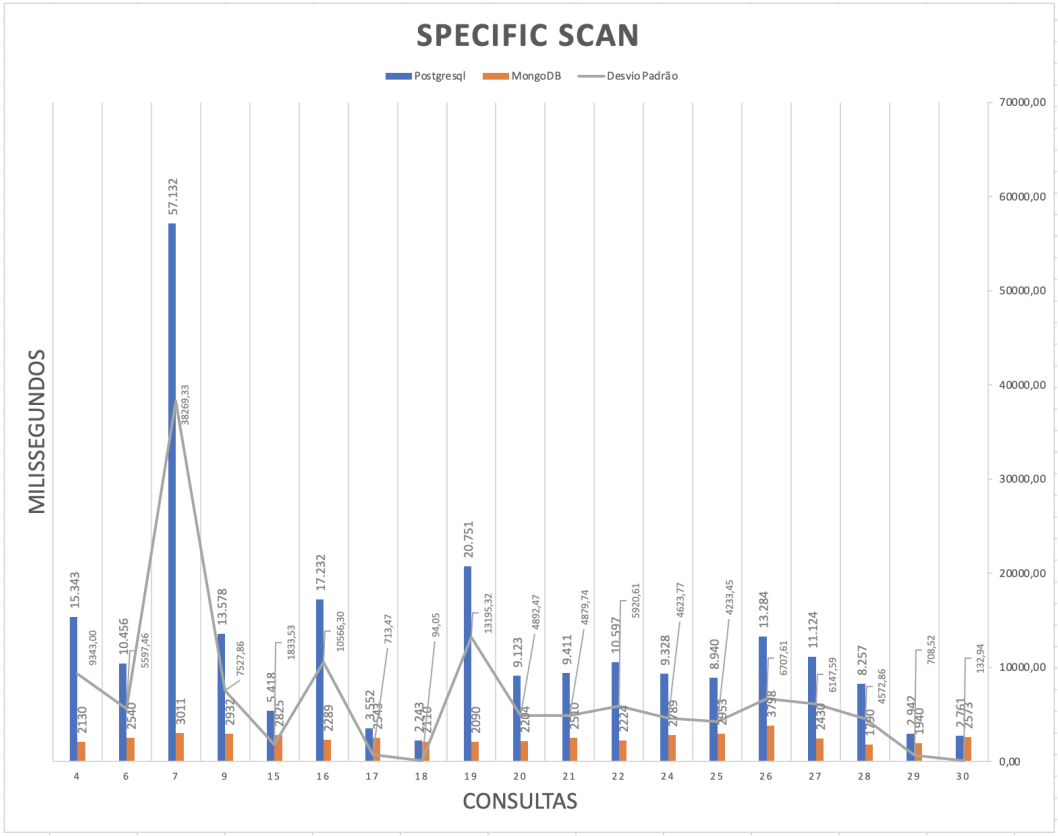

Figura 5.4 – Gráfico por Consulta do operador *Specific Scan*, cuja numeração segue descrita na Tabela 4.1

Fonte: Próprio Autor

Tabela 5.4 – Tabela de mínimo e máximo do *Specific Scan* com 100 mil notas fiscais eletrônicas

	PostgreSQL		MongoDB	
	Mínimo	Máximo	Mínimo	Máximo
Tempo de Execução (ms)	3221	75548	1442	3768

5.4 *EXACT MATCH*

O *Exact Match*, é um operador de combinação, com o qual é possível combinar atributos de uma tabela e retornar esses valores na consulta.

5.4.1 UTILIZANDO UM LOTE DE 50 MIL NOTAS:

Após a realização de toda a bateria de testes, foi calculada a média dos resultados e exibida no gráfico, evidenciados na Figura 5.5.

Nesse caso, semelhante ao que ocorreu no segundo operador, em algumas consultas a diferença de busca entre um e outro foi baixa, tal como na Consulta 14, acerca da busca das notas

Tabela 5.5 – Tabela de mínimo e máximo do Exact Match

	PostgreSQL		MongoDB	
	Mínimo	Máximo	Mínimo	Máximo
Tempo de Execução (ms)	2190	5527	1002	1580

fiscais por anos, em que a diferença foi de apenas 308 milissegundos. Isso deve ser decorrente da necessidade do MongoDB de percorrer toda a coleção de documentos (realiza um *Coll Scan*), retornar e capturar o documento que atende às condições. Contudo, em outras consultas, como na 23 (consultar o modelo do documento fiscal), o PostgreSQL utilizou 3.612 milissegundos a mais.

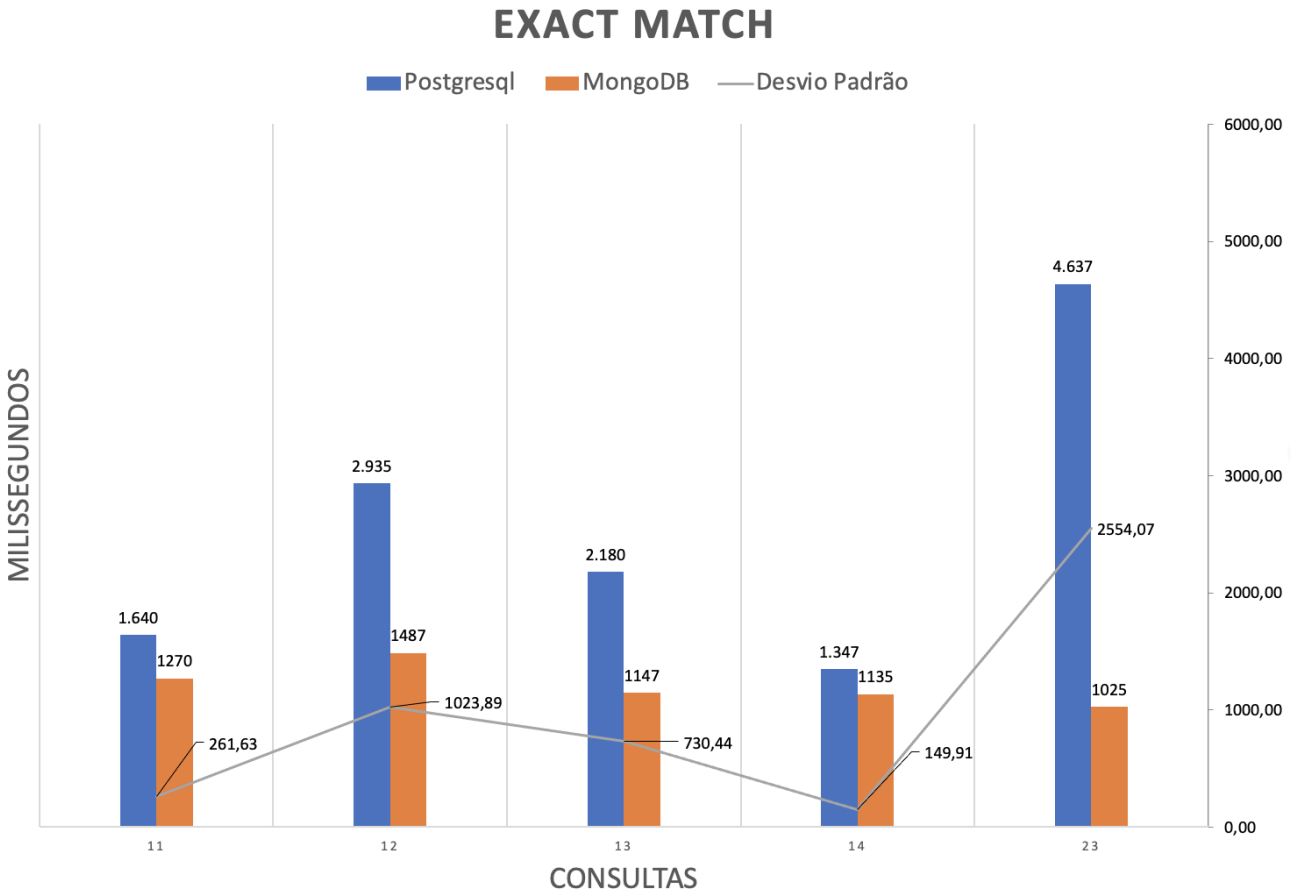

Figura 5.5 – Gráfico por Consulta do operador *Exact Match*, cuja numeração segue descrita na Tabela 4.1

Fonte: Próprio Autor

A Tabela 5.5 mostra o mínimo e o máximo do tempo de execução em milissegundos de cada SGBD no operador *Exact Match*. Os planos de consultas que representam o mínimo e o máximo nesse operador podem ser lidos no Apêndice 7.4.2 C.

Diante de tais dados, observa-se que o PostgreSQL teve um desempenho inferior ao

do MongoDB, tendo um tempo de resposta maior cerca de duas vezes, quando comparados os valores mínimos, e de cerca de três vezes, quando comparados os valores máximos.

5.4.2 UTILIZANDO UM LOTE DE 100 MIL NOTAS:

A Figura 5.6 exibe o terceiro operador, o *Exact Match* contendo em sua base 100 mil notas fiscais eletrônicas. Neste caso algumas consultas obtiveram resultados aproximados, destacando-se a Consulta 14 (Consultar notas fiscais por ano), onde a diferença ficou apenas em 403 milissegundos, porém, na Consulta 23 ocorreu uma maior discrepância de resposta (consultar o modelo do documento fiscal), na qual o PostgreSQL utilizou 7.607 milissegundos a mais.

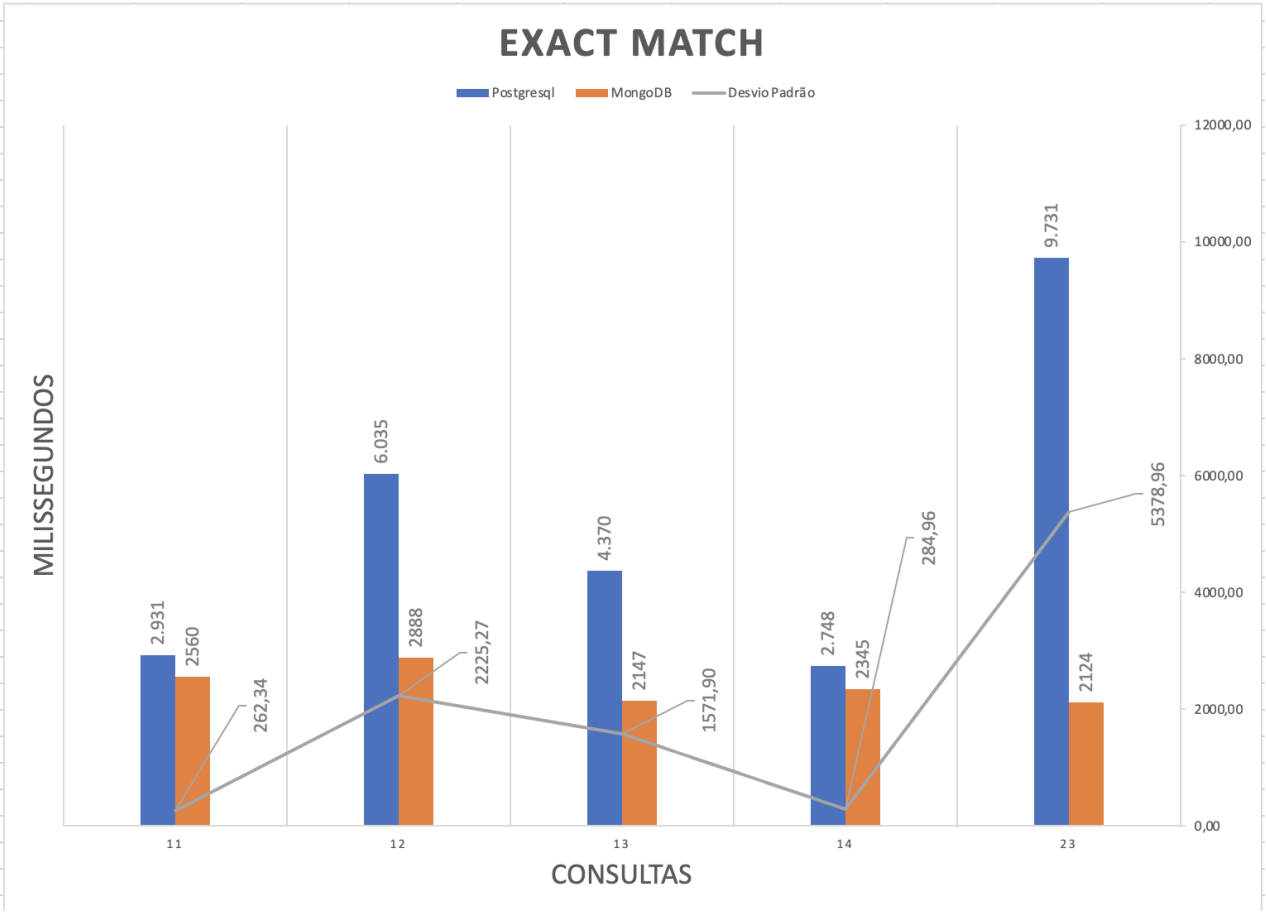

Figura 5.6 – Gráfico por Consulta do operador *Exact Match*, cuja numeração segue descrita na Tabela 4.1

Fonte: Próprio Autor

A Tabela 5.6 mostra o mínimo e o máximo do tempo de execução em milissegundos de cada SGBD no operador *Exact Match* com 100 mil notas fiscais eletrônicas.

Neste caso a tabela mostra que o desempenho do PostgreSQL foi inferior ao do MongoDB, visto que esse SGBD utilizou o dobro do tempo para retornar os resultados, quando comparados aos valores mínimos e o triplo, quando comparados aos valores máximos.

Tabela 5.6 – Tabela de mínimo e máximo do Exact Match com 100 mil notas fiscais eletrônicas

	PostgreSQL		MongoDB	
	Mínimo	Máximo	Mínimo	Máximo
Tempo de Execução (ms)	4560	12054	2307	3380

5.5 BUSCAS COM A UTILIZAÇÃO DE ÍNDICE

5.5.1 UTILIZANDO UM LOTE DE 50 MIL NOTAS:

A Figura 5.7 exibe as consultas executadas por meio de índice, uma forma de otimizar as consultas, gerando um melhor desempenho. Após a execução das buscas, foi calculada a média dos resultados, exibidas no gráfico a seguir.

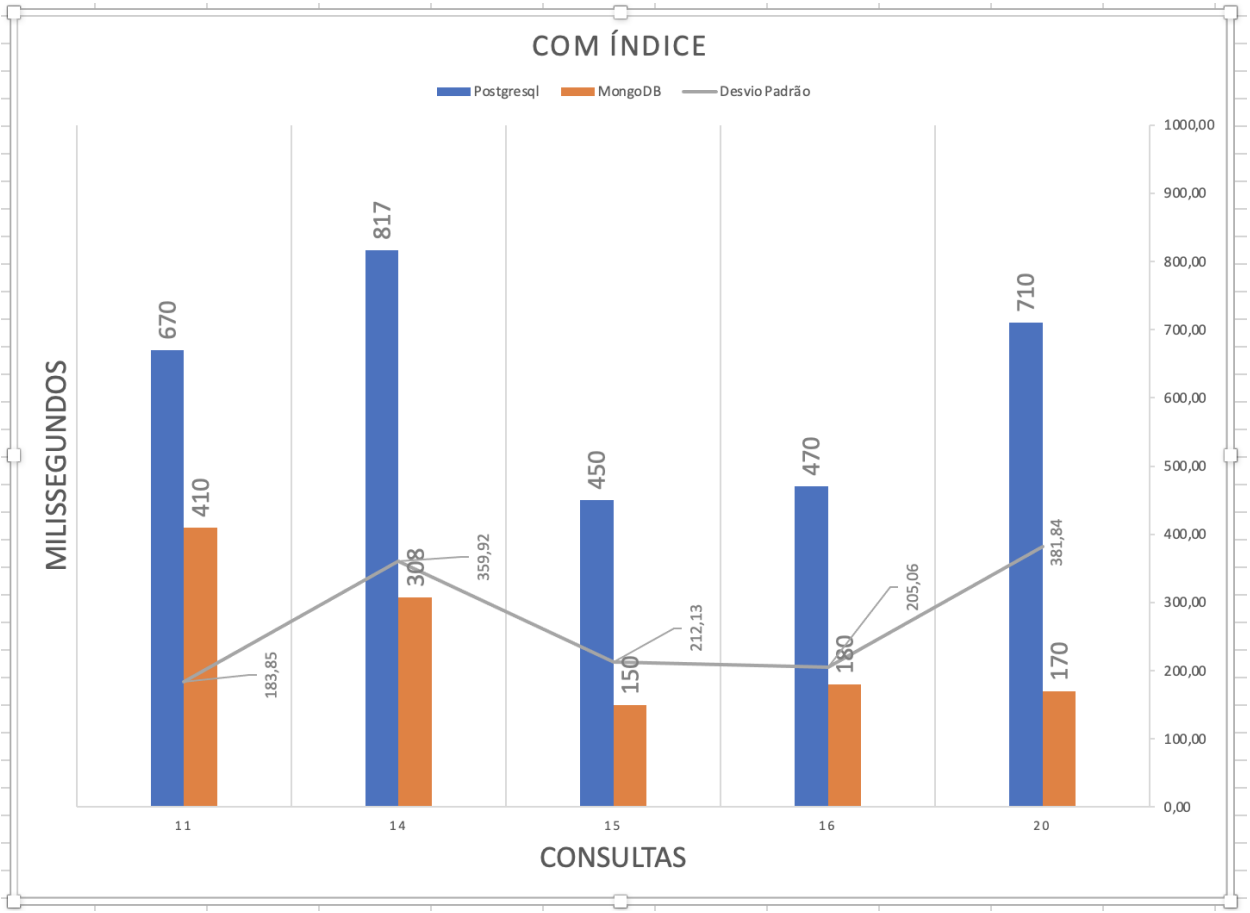

Figura 5.7 – Gráfico por Consulta com índice, cuja numeração segue descrita na Tabela 4.1

Fonte: Próprio Autor

A Tabela 5.7 mostra o mínimo e o máximo do tempo de execução em milissegundos de cada SGBD utilizando índice com 100 mil notas fiscais eletrônicas. Neste caso destacamos a

Tabela 5.7 – Tabela de mínimo e máximo com Índice

	PostgreSQL		MongoDB	
	Mínimo	Máximo	Mínimo	Máximo
Tempo de Execução (ms)	430	720	150	198

consulta 20 (Consultar a tag chave de cada nota) no qual o PostgreSQL foi mais lento do que o MongoDB registrando 540 milissegundos de diferença.

Diante de tais dados, observa-se que o PostgreSQL teve um desempenho inferior ao do MongoDB, tendo um tempo de resposta maior cerca de três vezes, quando comparados os valores mínimos, e de cerca de quatro vezes, quando comparados os valores máximos.

5.5.2 UTILIZANDO UM LOTE DE 100 MIL NOTAS:

A Figura 5.8 exibe as consultas executadas por meio de índice, com a base de dados carregada com 100 mil notas fiscais. É importante destacar que na Consulta 1 (consultar todos os campos de uma determinada) a diferença entre os SGBDs não foi tão significativa, com o PostgreSQL apresentando apenas uma lentidão de 460 milissegundos.

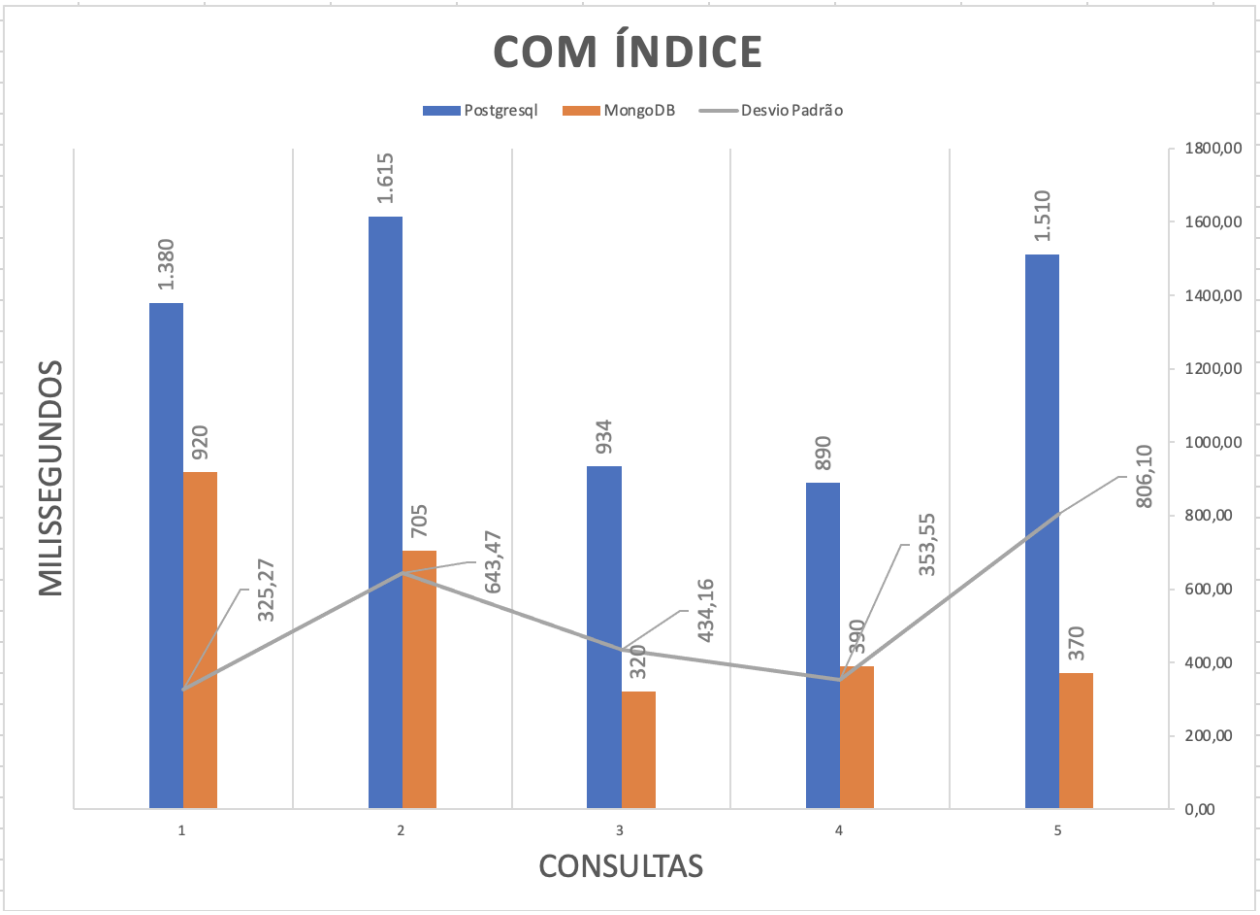

Figura 5.8 – Gráfico por Consulta com índice, cuja numeração segue descrita na Tabela 4.1

Fonte: Próprio Autor

Tabela 5.8 – Tabela de mínimo e máximo com Índice

	PostgreSQL		MongoDB	
	Mínimo	Máximo	Mínimo	Máximo
Tempo de Execução (ms)	1436	19405	760	3768

A Tabela 5.8 mostra o mínimo e o máximo do tempo de execução em milissegundos de cada SGBD, utilizando índice com 100 mil notas fiscais eletrônicas.

Podemos observar que o PostgreSQL teve um desempenho inferior ao do MongoDB, tendo um tempo de resposta maior cerca de duas vezes, quando comparados os valores mínimos, e de cerca de cinco vezes, quando comparados os valores máximos. Ao analisarmos os gráficos de todos os resultados, é possível identificar que as consultas que utilizam índices comparadas as que não utilizam, possuem um melhor desempenho durante as execuções de buscas. Isto ocorre devido ao fato dos índices serem estruturas cujo a função é fornecer celeridade no acesso às linhas de uma tabela.

Por fim, ao analisar os resultados obtidos nos três operadores acima descritos, observa-se que a hipótese inicial do estudo foi confirmada, com o MongoDB apresentando uma tempo de resposta baixo, independente do que é consultado no documento. O mesmo não ocorre com o PostgreSQL, no qual o tempo varia conforme o tipo de busca, custando um tempo muito maior nos casos que envolvem a consulta a um determinado atributo de uma tabela. Dessa maneira, pode-se considerar que o MongoDB é mais constante, pois não ocorrem tantas oscilações como no PostgreSQL.

6 CONSIDERAÇÕES FINAIS

Para obtenção de resultados que atendam atributos de qualidade relacionados à persistência de dados torna-se fundamental escolher uma tecnologia de armazenamento adequada para os dados de uma determinada aplicação. Nesse sentido, essa pesquisa analisou dois modelos de dados que são amplamente difundidos na comunidade, o modelo orientado a documentos (MongoDB) e o modelo relacional (PostgreSQL).

A implementação dos dois modelos de dados acima citados, se deu, inicialmente, obedecendo a estrutura de modelagem de cada um, sendo aplicados à mesma base de dados, inicialmente um lote de 50 mil e, posteriormente, outro lote de 100 mil NFes. Em seguida, foram realizados os experimentos com a finalidade de descobrir a eficiência de cada um dos modelos, com relação as consultas realizadas, isto é, o tempo de resposta que cada SGBD utiliza no retorno das consultas. Utilizando o recurso de índice pode-se notar que o PostgreSQL obteve um desempenho de melhor eficiência quando não utilizado o índice, o mesmo ocorreu com o MongoDB, isso evidenciou a melhor eficácia quando se utiliza o índice para a realização das consultas. Corroborando a hipótese inicialmente aventada, neste trabalho, os melhores resultados foram obtidos pelo modelo de dados orientados a documento, utilizando o MongoDB, em relação ao tempo de resposta das consultas em todos os operadores do *benchmark* avaliados.

Após a conclusão desta pesquisa, observa-se duas contribuições principais: a possibilidade de mensurar o desempenho de dois modelos de dados, utilizando uma base de dados de NFes no âmbito da *Big Data*, tratando-se assim de um de grande volumes de dados, assim como poder estabelecer, através de um software, um entendimento melhor dos campos de uma NF-e, permitindo ao utilizador a capacidade de elaborar uma metodologia mais eficaz de consultas, conforme o seu objetivo.

6.1 LIMITAÇÕES DO ESTUDO E TRABALHOS FUTUROS

Apesar dos avanços trazidos por este estudo, algumas limitações devem ser consideradas. Como trabalho futuro, sugere-se a implementação utilizando um volume maior de dados, assim como com outros modelos de dados, como por exemplo: modelo orientado a coluna, e modelo dimensional, para que seja possível uma nova análise desse mesmo ambiente e mais outros dois tipos de modelo. Além disso, sugere-se também a realização de novos testes utilizando algumas outras formas de consultas, avaliação sobre novas métricas. Contudo, considera-se que os resultados aqui apresentados podem ampliar o conhecimento dos profissionais que lidam diretamente com o gerenciamento dos SGBDS (MongoDB e PostgreSQL), subsidiando as escolhas destes quanto a melhor tecnologia a ser utilizada, conforme suas necessidades.

REFERÊNCIAS

[1] O. K. Takai, "Introducao a banco de dados," 2005.

[2] A. C. Centenaro, C. F. Druziani, Â. A. Sucolotti, and J. L. Todesco, "Modelando um data warehouse dimensional a partir de um modelo hierárquico."

[3] B. E. Soares and C. Boscarioli, "Modelo de banco de dados colunar: Características, aplicações e exemplos de sistemas," *Escola Regional de Banco de Dados–Sociedade Brasileira de Computação (IX ERBD–SBC), Camobiu*, 2013.

[4] A. C. d. Faria, J. R. Finatelli, C. Geron, and M. Romero, "Sped–sistema público de escrituração digital: Percepção dos contribuintes em relação os impactos da adoção do sped," *Recuperado em*, vol. 10, 2010.

[5] J. L. Filho and C. Iochpe, "Um estudo sobre modelos conceituais de dados para projeto de bancos de dados geográficos," *Revista IP-Informática Pública*, vol. 1, no. 2, pp. 37–90, 1999.

[6] R. F. Do Brasil, "Sistema Público de Escrituração Digital (SPED)," 2020. [Online]. Available: http://sped.rfb.gov.br/pagina/show/970

[7] C. R. Silva, "SISTEMA PÚBLICO DE ESCRITURAÇÃO DIGITAL (SPED): dificuldades e benefícios para contabilidades de Barreiras-BA no processo de transmissão das informações," pp. 1–28.

[8] E. M. d. Santos, "Uso de dados de documentos fiscais eletrônicos para o planejamento do transporte urbano de cargas," 2015.

[9] G. C. Nascimento, *Sped:(sistema público de escrituração digital) sem Armadilhas*. Editora Trevisan, 2014.

[10] P. Rob and C. Coronel, "Sistemas de banco de dados," *Projeto, implementação e*, 2011.

[11] R. Elmasri, S. B. Navathe, M. G. Pinheiro *et al.*, *Sistemas de banco de dados*. Pearson Addison Wesley São Paulo, 2005, vol. 6.

[12] F. N. R. Machado, *Tecnologia e projeto de Data Warehouse*. Saraiva Educação SA, 2004.

[13] P. J. Sadalage and M. Fowler, *NoSQL Essencial: Um guia conciso para o Mundo emergente da persistência poliglota*. Novatec Editora, 2019.

[14] B. Ritchie, *RavenDB high performance*. Packt Publishing Ltd, 2013.

[15] S. Tiwari, *Professional nosql*. John Wiley & Sons, 2011.

[16] D. A. P. Wanzeller, "Investigando o uso de bancos de dados orientados a documentos para gerenciar informações da administração pública," 2013.

[17] B. F. Lóscio, H. d. OLIVEIRA, and J. d. S. PONTES, "Nosql no desenvolvimento de aplicações web colaborativas," *VIII Simpósio Brasileiro de Sistemas Colaborativos*, vol. 10, no. 1, p. 11, 2011.

[18] D. C. Bruzarosco, A. V. Castoldi, and R. C. dos Santos Pacheco, "Criando data warehouse com o modelo dimensional," *Acta Scientiarum. Technology*, vol. 22, pp. 1389–1397, 2000.

[19] W. G. Pedrozo, "Tuning de índices em sistemas gerenciadores de banco de dados relacionais, utilizando sistemas classificadores."

[20] C. A. Anzanello, "Olap conceitos e utilização," *UFRGS-Instituto de Informática–Universidade Federal do Rio Grande do Sul*, 2007.

[21] F. N. R. Machado, *Big Data O Futuro dos Dados e Aplicações*. Saraiva Educação SA, 2018.

[22] W. Inmon, D. Linstedt, and M. Levins, *Data Architecture: A Primer for the Data Scientist: A Primer for the Data Scientist*. Elsevier Science, 2019. [Online]. Available: https://books.google.com.br/books?id=EZ-ZuwEACAAJ

[23] G. Li, B. C. Ooi, J. Feng, J. Wang, and L. Zhou, "Ease: an effective 3-in-1 keyword search method for unstructured, semi-structured and structured data," in *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, 2008, pp. 903–914.

[24] R. Casado and M. Younas, "Emerging trends and technologies in big data processing," *Concurrency and Computation: Practice and Experience*, vol. 27, no. 8, pp. 2078–2091, 2015.

[25] J. S. Van Der Veen, B. Van Der Waaij, E. Lazovik, W. Wijbrandi, and R. J. Meijer, "Dynamically scaling apache storm for the analysis of streaming data," *Proceedings - 2015 IEEE 1st International Conference on Big Data Computing Service and Applications, BigDataService 2015*, pp. 154–161, 2015.

[26] A. Jabbar, P. Akhtar, and S. Dani, "Real-time big data processing for instantaneous marketing decisions: A problematization approach," *Industrial Marketing Management*, no. September, pp. 0–1, 2019. [Online]. Available: https://doi.org/10.1016/j.indmarman.2019.09.001

[27] R. Tardío, A. Maté, and J. Trujillo, "A new big data benchmark for olap cube design using data pre-aggregation techniques," *Applied Sciences*, vol. 10, no. 23, p. 8674, 2020.

[28] F. Funke, A. Kemper, and T. Neumann, "Benchmarking hybrid oltp&olap database systems," *Datenbanksysteme für Business, Technologie und Web (BTW)*, 2011.

[29] A. Azzini, P. Ceravolo, and M. Colella, "Performances of olap operations in graph and relational databases," in *International Conference on Knowledge Management in Organizations*. Springer, 2019, pp. 282–293.

[30] L. Torres, G. R. Krüger, M. S. Oyamada, and C. Boscarioli, "Evaluating the impact of data modeling on olap applications using relacional and columnar dbms."

[31] M. Chevalier, M. El Malki, A. Kopliku, O. Teste, and R. Tournier, "Benchmark for olap on nosql technologies comparing nosql multidimensional data warehousing solutions," in *2015 IEEE 9th international conference on research challenges in information science (RCIS)*. IEEE, 2015, pp. 480–485.

[32] M. El Malki, A. Kopliku, E. Sabir, and O. Teste, "Benchmarking big data olap nosql databases," in *International Symposium on Ubiquitous Networking*. Springer, 2018, pp. 82–94.

[33] B. E. M. d. ALBUQUERQUE FILHO, "Obas: Um benchmark olap para serviços de análises," Master's thesis, Universidade Federal de Pernambuco, 2014.

[34] X. Wang, S. Staab, and T. Tiropanis, "Aspg: generating olap queries for sparql benchmarking," in *Joint International Semantic Technology Conference*. Springer, 2016, pp. 171–185.

[35] J. R. F. Lopes Júnior, "Postgresql versus mongodb: Um estudo comparativo baseado em benchmark de consultas," 2016.

[36] A. d. C. Oliveira Filho *et al.*, "Benchmark para métodos de consultas por palavras-chave a bancos de dados relacionais," 2018.

7 APÊNDICES

7.1 APÊNDICE A - RESULTADO GERADO PELO SOFTWARE

nome lógico	informação	descrição	observação
infNFe	infNFe-min_1-max_1	Informações da Nota Fiscal eletrônica	
infNFe_ide	infNFe_ide-min_1-max_1	Identificação da NF-e	
infNFe_ide_cUF	infNFe_ide_cUF-TCodUfIBGE-min_1-max_1	Código da UF do emitente do Documento Fiscal. Utilizar a Tabela do IBGE.	
infNFe_ide_cNF	infNFe_ide_cNF-min_1-max_1	Código numérico que compõe a Chave de Acesso. Número aleatório gerado pelo emitente para cada NF-e.	
infNFe_ide_natOp	infNFe_ide_natOp-min_1-max_1	Descrição da Natureza da Operação	
infNFe_ide_mod	infNFe_ide_mod-TMod-min_1-max_1	Código do modelo do Documento Fiscal. 55 = NF-e; 65 = NFC-e.	
infNFe_ide_serie	infNFe_ide_serie-TSerie-min_1-max_1	Série do Documento Fiscal	série normal 0-889
infNFe_ide_nNF	infNFe_ide_nNF-TNF-min_1-max_1	Número do Documento Fiscal	
infNFe_ide_dhEmi	infNFe_ide_dhEmi-TDateTimeUTC-min_1-max_1	Data e Hora de emissão do Documento Fiscal (AAAA-MM-DDThh:mm:ssTZD) ex.: 2012-09-01T13:00:00-03:00	
infNFe_ide_dhSaiEnt	infNFe_ide_dhSaiEnt-TDateTimeUTC-min_0-max_1	Data e Hora da saída ou de entrada da mercadoria / produto (AAAA-MM-DDThh:mm:ssTZD)	
infNFe_ide_tpNF	infNFe_ide_tpNF-min_1-max_1	Tipo do Documento Fiscal (0 - entrada; 1 - saída)	
infNFe_ide_idDest	infNFe_ide_idDest-min_1-max_1	Identificador de Local de destino da operação (1-Interna;2-Interestadual;3-Exterior)	
infNFe_ide_cMunFG	infNFe_ide_cMunFG-TCodMunIBGE-min_1-max_1	Código do Município de Ocorrência do Fato Gerador (utilizar a tabela do IBGE)	
infNFe_ide_tpImp	infNFe_ide_tpImp-min_1-max_1	Formato de impressão do DANFE (0-sem DANFE;1-DANFe Retrato; 2-DANFe Paisagem;3-DANFe Simplificado;	4-DANFe NFC-e;5-DANFe NFC-e em mer
infNFe_ide_tpEmis	infNFe_ide_tpEmis-min_1-max_1	Forma de emissão da NF-e	1 - Normal;
infNFe_ide_cDV	infNFe_ide_cDV-min_1-max_1	Dígito Verificador da Chave de Acesso da NF-e	
infNFe_ide_tpAmb	infNFe_ide_tpAmb-TAmb-min_1-max_1	Identificação do Ambiente:	1 - Produção
infNFe_ide_finNFe	infNFe_ide_finNFe-TFinNFe-min_1-max_1	Finalidade da emissão da NF-e:	1 - NFe normal
infNFe_ide_indFinal	infNFe_ide_indFinal-min_1-max_1	Indica operação com consumidor final (0-Não;1-Consumidor Final)	
infNFe_ide_indPres	infNFe_ide_indPres-min_1-max_1	Indicador de presença do comprador no estabelecimento comercial no momento da oepração	
infNFe_ide_indIntermed	infNFe_ide_indIntermed-min_0-max_1	IIndicador de intermediador/marketplace	
infNFe_ide_procEmi	infNFe_ide_procEmi-TProcEmi-min_1-max_1	Processo de emissão utilizado com a seguinte codificação:	0 - emissão de NF-e com aplicativo do co
infNFe_ide_verProc	infNFe_ide_verProc-min_1-max_1	versão do aplicativo utilizado no processo de	emissão
infNFe_ide_NFref	infNFe_ide_NFref-min_0-max_500	Grupo de informações da NF referenciada	
refNFe	refNFe-TChNFe-min_1-max_1	Chave de acesso das NF-e referenciadas. Chave de acesso compostas por Código da UF (tabela do IBGE) + AAMM da emissão + CNPJ do Emitente + modelo, série e número da NF-e Referenciada + Código Numérico + DV.	
refNF_cUF	refNF_cUF-TCodUfIBGE-min_1-max_1	Dados da NF modelo 1/1A referenciada ou NF modelo 2 referenciada Código da UF do emitente do Documento Fiscal. Utilizar a Tabela do IBGE.	
refNF_AAMM	refNF_AAMM-min_1-max_1	AAMM da emissão	
refNF_CNPJ	refNF_CNPJ-TCnpj-min_1-max_1	CNPJ do emitente do documento fiscal referenciado	
refNF_mod	refNF_mod-min_1-max_1	Código do modelo do Documento Fiscal. Utilizar 01 para NF modelo 1/1A e 02 para NF modelo 02	
refNF_serie	refNF_serie-TSerie-min_1-max_1	Série do Documento Fiscal, informar zero se inexistente	
refNF_nNF	refNF_nNF-TNF-min_1-max_1	Número do Documento Fiscal	
refNFP	refNFP-min_1-max_1	Grupo com as informações NF de produtor referenciada	
refNFP_cUF	refNFP_cUF-TCodUfIBGE-min_1-max_1	Código da UF do emitente do Documento Fiscal.Utilizar a Tabela do IBGE (Anexo IV - Tabela de UF, Município e País)	
refNFP_AAMM	refNFP_AAMM-min_1-max_1	AAMM da emissão da NF de produtor	
CNPJ	CNPJ-TCnpj-min_1-max_1	CNPJ do emitente da NF de produtor	
CPF	CPF-TCpf-min_1-max_1	CPF do emitente da NF de produtor	
refNFP_IE	refNFP_IE-TIeDest-min_1-max_1	IE do emitente da NF de Produtor	
refNFP_mod	refNFP_mod-min_1-max_1	Código do modelo do Documento Fiscal - utilizar 04 para NF da produtor ou 01 para NF Avulsa	
refNFP_serie	refNFP_serie-TSerie-min_1-max_1	Série do Documento Fiscal, informar zero se inexistentesérie	
refNFP_nNF	refNFP_nNF-TNF-min_1-max_1	Número do Documento Fiscal - 1 - 999999999	
refCTe	refCTe-TChNFe-min_1-max_1	Utilizar esta TAG para referenciar um CT-e emitido anteriormente, vinculada a NF-e atual	
refECF	refECF-min_1-max_1	Grupo do Cupom Fiscal vinculado à NF-e	
refECF_mod	refECF_mod-min_1-max_1	Código do modelo do Documento Fiscal	Preencher com "2B", quando se tratar de

Figura 7.1 – Arquivo de saída gerado pelo software
Fonte: Próprio Autor

7.2 APÊNDICE B - PLANOS DE CONSULTA DO FULL SCAN

7.2.1 POSTGRESQL

Consulta 8 - Seq Scan on tb_nfe (cost=0.00..894.03 rows=25603 width=4) (actual time=0.008..3.681 rows=50000 loops=1) Planning time: 0.050 ms Execution time: 4.263 ms (3 rows)

Consulta 10 - Sort (cost=14759.68..14823.69 rows=25603 width=1066) (actual time=17.920..21.892 rows=50000 loops=1) Sort Key: data_emissao Sort Method: external sort Disk: 4488kB -> Seq Scan on tb_nfe (cost=0.00..894.03 rows=25603 width=1066) (actual time=0.008..2.653 rows=50000 loops=1) Planning time: 0.131 ms Execution time: 19.800 ms (6 rows)

7.2.2 MONGODB

Consulta 5 - "queryPlanner": "plannerVersion": 1, "namespace": "db.nfe", "indexFilterSet": false, "parsedQuery":

, "winningPlan": "stage": "FETCH", "inputStage": "stage": "IXSCAN", "keyPattern": "_id": 1 , "indexName": "_id_", "isMultiKey": false, "multiKeyPaths": "_id": [ ] , "isUnique": true, "isSparse": false, "isPartial": false, "indexVersion": 2, "direction": "forward", "indexBounds": "_id": ["[MinKey, MaxKey]"] , "rejectedPlans": [ ] , "executionStats": "executionSuccess": true, "nReturned": 50000, "executionTimeMillis": 1338, "totalKeysExamined": 50000, "totalDocsExamined": 50000, "executionStages": "stage": "FETCH", "nReturned": 50000, "executionTimeMillisEstimate": 753, "works": 50001, "advanced": 50000, "needTime": 0, "needYield": 0, "saveState": 61, "restoreState": 61, "isEOF": 1, "docsExamined": 50000, "alreadyHasObj": 0, "inputStage": "stage": "IXSCAN", "nReturned": 50000, "executionTimeMillisEstimate": 137, "works": 50001, "advanced": 50000, "needTime": 0, "needYield": 0, "saveState": 61, "restoreState": 61, "isEOF": 1, "keyPattern": "_id": 1 , "indexName": "_id_", "isMultiKey": false, "multiKeyPaths": "_id": [ ] , "isUnique": true, "isSparse": false, "isPartial": false, "indexVersion": 2, "direction": "forward", "indexBounds": "_id": ["[MinKey, MaxKey]"] , "keysExamined": 50000, "seeks": 1, "dupsTested": 0, "dupsDropped": 0 , "serverInfo": "host": "3e04b56c812e", "port": 27017, "version": "4.4.3", "gitVersion": "913d6b62acfbb344dde1b116 "ok": 1

Consulta 8 - "queryPlanner": "plannerVersion": 1, "namespace": "db.nfe", "indexFilterSet": false, "parsedQuery": "_id": "$eq": 1 , "winningPlan": "stage": "IDHACK", "rejectedPlans": [ ] , "executionStats": "executionSuccess": true, "nReturned": 0, "executionTimeMillis": 37, "totalKeysExamined": 0, "totalDocsExamined": 0, "executionStages": "stage": "IDHACK", "nReturned": 0, "executionTimeMillisEstimate": 37, "works": 1, "advanced": 0, "needTime": 0, "needYield": 0, "saveState": 0, "restoreState": 0, "isEOF": 1, "keysExamined": 0, "docsExamined": 0 , "serverInfo": "host": "3e04b56c812e", "port": 27017, "version": "4.4.3", "gitVersion": "913d6b62acfbb344dde1b116f4161360acd8 "ok": 1

7.3 APÊNDICE C - PLANOS DE CONSULTA DO SPECIFIC SCAN

7.3.1 POSTGRESQL

Consulta 3 - Seq Scan on tb_imposto (cost=0.00..3871.51 rows=232851 width=2) (actual time=0.012..29.453 rows=232895 loops=1) Planning time: 0.049 ms Execution time: 37.849 ms (3 rows)

Consulta 29 - Seq Scan on tb_emitente (cost=0.00..96.53 rows=2553 width=11) (actual time=0.007..0.440 rows=2539 loops=1) Planning time: 0.033 ms Execution time: 1.642 ms (3 rows)

7.3.2 MONGODB

Consulta 15 - "queryPlanner": "plannerVersion": 1, "namespace": "db.nfe", "indexFilterSet": false, "parsedQuery":

, "winningPlan": "stage": "PROJECTION_DEFAULT", "transformBy": "0.emitente.CNPJ": 1 , "inputStage": "stage": "COLLSCAN", "direction": "forward"

, "rejectedPlans": [ ] , "executionStats": "executionSuccess": true, "nReturned": 50000, "executionTimeMillis": 1834, "totalKeysExamined": 0, "totalDocsExamined": 50000, "executionStages": "stage": "PROJECTION_DEFAULT", "nReturned": 50000, "executionTimeMillisEstimate": 1704, "works": 50002, "advanced": 50000, "needTime": 1, "needYield": 0, "saveState": 75, "restoreState": 75, "isEOF": 1, "transformBy": "0.emitente.CNPJ": 1 , "inputStage": "stage": "COLLSCAN", "nReturned": 50000, "executionTimeMillisEstimate": 1674, "works": 50002, "advanced": 50000, "needTime": 1, "needYield": 0, "saveState": 75, "restoreState": 75, "isEOF": 1, "direction": "forward", "docsExamined": 50000 , "serverInfo": "host": "3e04b56c812e", "port": 27017, "version": "4.4.3", "gitVersion": "913d6b62acfbb344dde1b116f4161360acd8fd13", "ok": 1

Consulta 28 - "queryPlanner": "plannerVersion": 1, "namespace": "db.nfe", "indexFilterSet": false, "parsedQuery":

, "winningPlan": "stage": "PROJECTION_DEFAULT", "transformBy": "0.destinatario.cMunF": 1 , "inputStage": "stage": "COLLSCAN", "direction": "forward"

, "rejectedPlans": [ ] , "executionStats": "executionSuccess": true, "nReturned": 50000, "executionTimeMillis": 700, "totalKeysExamined": 0, "totalDocsExamined": 50000, "executionStages": "stage": "PROJECTION_DEFAULT", "nReturned": 50000, "executionTimeMillisEstimate": 577, "works": 50002, "advanced": 50000, "needTime": 1, "needYield": 0, "saveState": 57, "restoreState": 57, "isEOF": 1, "transformBy": "0.destinatario.cMunF": 1 , "inputStage": "stage": "COLLSCAN", "nReturned": 50000, "executionTimeMillisEstimate": 542, "works": 50002, "advanced": 50000, "needTime": 1, "needYield": 0, "saveState": 57, "restoreState": 57, "isEOF": 1, "direction": "forward", "docsExamined": 50000 , "serverInfo": "host": "3e04b56c812e", "port": 27017, "version": "4.4.3",

"gitVersion": "913d6b62acfbb344dde1b116f4161360acd8fd13", "ok": 1

7.4 APÊNDICE D - PLANOS DE CONSULTA DO EXACT MATCH

7.4.1 POSTGRESQL

Consulta 11 - Seq Scan on tb_nfe (cost=0.00..958.04 rows=1 width=1066) (actual time=2.460..2.460 rows=0 loops=1) Filter: (data_saida = '2018-01-01'::date) Rows Removed by Filter: 50000 Planning time: 0.077 ms Execution time: 2.190 ms (5 rows)

Consulta 23 - Seq Scan on tb_nfe (cost=0.00..958.04 rows=16468 width=3) (actual time=0.010..4.941 rows=16287 loops=1) Filter: ((modelo)::text = '55'::text) Rows Removed by Filter: 9337 Planning time: 0.070 ms Execution time: 5.527 ms (5 rows)

7.4.2 MONGODB

Consulta 12 - "queryPlanner": "plannerVersion": 1, "namespace": "db.nfe", "indexFilterSet": false, "parsedQuery": "$and": ["0.dhEmi": "$lt": 1994 , "0.dhEmi": "$gte": 2001] , "winningPlan": "stage": "COLLSCAN", "filter": "$and": ["0.dhEmi": "$lt": 1994 , "0.dhEmi": "$gte": 2001] , "direction": "forward", "rejectedPlans": [ ] , "executionStats": "executionSuccess": true, "nReturned": 0, "executionTimeMillis": 1150, "totalKeysExamined": 0, "totalDocsExamined": 50000, "executionStages": "stage": "COLLSCAN", "filter": "$and": ["0.dhEmi": "$lt": 1994 , "0.dhEmi": "$gte": 2001] , "nReturned": 0, "executionTimeMillisEstimate": 1314, "works": 50002, "advanced": 0, "needTime": 50001, "needYield": 0, "saveState": 67, "restoreState": 67, "isEOF": 1, "direction": "forward", "docsExamined": 50000 , "serverInfo": "host": "3e04b56c812e", "port": 27017, "version": "4.4.3", "gitVersion": "913d6b62acfbb344dde1b116f4161360acd8fd13", "ok": 1

Consulta 13 - "queryPlanner": "plannerVersion": 1, "namespace": "db.nfe", "indexFilterSet": false, "parsedQuery": "$and": ["0.dhEmi": "$lt": 2000 , "0.dhEmi": "$gte": 2001] , "winningPlan": "stage": "COLLSCAN", "filter": "$and": ["0.dhEmi": "$lt": 2000 , "0.dhEmi": "$gte": 2001] , "direction": "forward", "rejectedPlans": [ ] , "executionStats": "executionSuccess": true, "nReturned": 0, "executionTimeMillis": 1102, "totalKeysExamined": 0, "totalDocsExamined": 50000, "executionStages": "stage": "COLLSCAN", "filter": "$and": ["0.dhEmi": "$lt": 2000 , "0.dhEmi": "$gte": 2001] , "nReturned": 0, "executionTimeMillisEstimate": 850, "works": 50002, "advanced": 0, "needTime": 50001, "needYield": 0, "saveState": 57, "restoreState": 57, "isEOF": 1, "direction": "forward", "docsExamined": 50000 , "serverInfo": "host": "3e04b56c812e", "port": 27017, "version": "4.4.3", "gitVersion": "913d6b62acfbb344dde1b116f4161360acd8fd13", "ok": 1