Análise comparativa de arquiteturas de bancos de dados distribuídos para documentos fiscais eletrônicos.

Jonathan Rodrigues de Almeida

CENTRO DE INFORMÁTICA
UNIVERSIDADE FEDERAL DA PARAÍBA

João Pessoa, 2023

Jonathan Rodrigues de Almeida

Análise comparativa de arquiteturas de bancos de dados distribuídos para documentos fiscais eletrônicos.

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

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

Dezembro de 2023

Catalogação na publicação
Seção de Catalogação e Classificação

A447a Almeida, Jonathan Rodrigues de.
Análise comparativa de arquiteturas de bancos de dados distribuídos para documentos fiscais eletrônicos. / Jonathan Rodrigues de Almeida. - João Pessoa, 2023.
59 f.

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

1. Bancos de dados distribuídos. 2. Benchmark. 3. Notas fiscais eletrônicas. I. Oliveira, Marcelo Iury De Sousa. II. Kulesza, Raoni. III. Título.

UFPB/CI CDU 004

Elaborado por Michelle de Kássia Fonseca Barbosa - CRB-738

CENTRO DE INFORMÁTICA
UNIVERSIDADE FEDERAL DA PARAÍBA

Trabalho de Conclusão de Curso de Ciência da Computaçãointitulado ***Análise comparativa de arquiteturas de bancos de dados distribuídos para documentos fiscais eletrônicos.*** de autoria de Jonathan Rodrigues de Almeida, aprovada pela banca examinadora constituída pelos seguintes professores:

Prof. Dr. Marcelo Iury De Sousa Oliveira
UFPB

Prof. Dr. Raoni Kulesza
UFPB

Prof. M.sc Hidelberg Oliveira Albuquerque
UFRPE

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

João Pessoa, 1 de dezembro de 2023

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

RESUMO

O grande volume de dados movimentados no manuseio de documentos fiscais digitais suscita a necessidade de otimização de seu armazenamento e acesso, sendo uma dessas formas de otimização, o particionamento e distribuição de bancos de dados. Neste estudo foi conduzida uma análise comparativa de arquiteturas de bancos de dados PostgreSQL distribuídos a fim de determinar o impacto da distribuição no desempenho de consulta desses bancos. Foi constatado que o custo de transmissão dos resultados supera os ganhos de performance por paralelização nas formas mais comuns de consulta.

Palavras-chave: Bancos de dados distribuídos, benchmark, notas fiscais eletrônicas.

ABSTRACT

The large volume of data moved when handling digital fiscal documents raises the need to optimize their storage and access, one of these forms of optimization being the partitioning and distribution of databases. In this study, a comparative analysis of distributed PostgreSQL database architectures was conducted in order to determine the impact of distribution on the query performance of these databases. It was found that the cost of transmitting results exceeds the performance gains due to parallelization in the most common forms of query.

Key-words: Distributed databases, benchmark, digital fiscal documents.

LISTA DE FIGURAS

Sumário

1 INTRODUÇÃO

A emissão e disponibilização de notas fiscais eletrônicas ocorre de forma integrada a nível nacional, e devido ao volume de dados em questão, demanda do sistema responsável (SPED) alta capacidade de processamento e armazenamento. Em consulta realizada em 22 de maio de 2022, o SPED contava com 32,46 bilhões de notas fiscais autorizadas[4], analisando as estatísticas de emissão entre 2016 e 2018, é possível estimar uma média aproximada de 196 milhões de NF-es de saída todos os meses.

Para facilitar a otimização e estabilidade do sistema, o ideal é que seus usuários minimizem transações computacionalmente caras. É o caso das análises das NF-es feitas pelas secretarias da fazenda dos estados, que são necessárias para auditoria e suporte a tomada de decisão nas administrações dos estados e municípios. A solução apropriada para esse cenário, seria a consolidação de uma base de dados analítica em âmbito estadual, alimentada regularmente pelos dados do SPED, evitando consultas diretas ao sistema transacional.

Mesmo considerando apenas dados regionais, um sistema assim ainda opera sobre um volume considerável de dados, que se acumulam continuamente e permanecem relevantes, apesar de cada vez menos acessados conforme envelhecem. Essas circunstâncias se assemelham ao cenário clássico em que se indicaria o particionamento de um banco, mas o impacto desse procedimento na performance de buscas varia drasticamente a depender das particularidades do sistema em questão.

1.1 Definição do Problema

Qual é o impacto na performance de consultas causado pela distribuição de um banco de dados PostgreSQL armazenando documentos fiscais digitais?

1.1.1 Objetivo geral

Investigar o impacto na performance de consultas causado por diferentes graus de distribuição de um banco de dados PostgreSQL armazenando documentos fiscais digitais.

1.1.2 Objetivos específicos

- Conceituar a NF-e nos modelos 55 e 65, apresentando sua origem e contexto atual na forma do sistema público de escrituração digital.
- Apresentar os conceitos e critérios envolvidos no particionamento e distribuição de um banco de dados.

- Avaliar a performance de consulta de um banco de dados analítico sob diferentes configurações de particionamento.
- Propor uma configuração de distribuição para um banco de documentos fiscais eletrônicos analítico de acordo com os resultados obtidos.

1.2 Estrutura da monografia

Os próximos capítulos cobrirão os seguintes tópicos:

Conceitos gerais: Apresenta conceitos necessários para a compreensão do trabalho, como as notas fiscais eletrônicas e o contexto em que se encontram, modelos de distribuição de bancos de dados analíticos e seu papel.

Metodologia: Especifica os critérios de avaliação dos cenários testados, assim como os detalhes do ambiente de testes.

Apresentação e análise dos resultados: Expõe os resultados obtidos na execução dos testes descritos no capítulo anterior.

Conclusões e trabalhos futuros: Aponta observações a respeito dos resultados, faz recomendações a respeito do sistema estudado e sugere trabalhos futuros.

2 CONCEITOS GERAIS

Neste capítulo, serão apresentados tópicos fundamentais para a compreensão do trabalho, passando pelos modelos de banco de dados comparados, conceitos de Big Data, os modelos de nota fiscal eletrônica abordados e o sistema do qual fazem parte.

2.1 Sistema Publico de Escrituração Digital

O SPED (Sistema Público de Escrituração Digital) é um dos passos no processo de informatização da relação entre o fisco e o contribuinte. Foi instituído pelo Decreto nº 6.022, de 22 de janeiro de 2007, e tem os objetivos de promover a integração dos fiscos, racionalizar e uniformizar as obrigações acessórias para os contribuintes, assim como acelerar a identificação de ilícitos tributários.

O sistema conta com um esforço conjunto das administrações tributárias nas três esferas governamentais para garantir a validade jurídica de documentos exclusivamente eletrônicos, assinados por meio de certificação digital, modernizando a sistemática anterior para o cumprimento das obrigações acessórias dos contribuintes às administrações tributárias e órgãos fiscalizadores.

Contribui para aprimorar e conferir um maior grau de legitimidade social aos meios de atendimento às obrigações tributárias acessórias exigidas pela legislação tributária, isso se dá através de parcerias fisco-empresas, que promovem a efetiva participação dos contribuintes na definição desses mecanismos e instrumentos, assim como no planejamento e identificação de soluções antecipadas no cumprimento das exigências requeridas pelas administrações tributárias. [3]
A composição atual do SPED conta com doze módulos, sendo eles:

- Conhecimento de Transporte Eletrônico (CT-e);
- Escrituração Fiscal Digital (EFD-Contribuições);
- Escrituração Fiscal Digital (EFD-ICMS e IPI);
- Escrituração Fiscal Digital das Retenções e Informações da Contribuição Previdenciária Substituída (EFD-Reinf);
- Escrituração Contábil Digital (ECD);
- Escrituração Contábil Fiscal (ECF);
- e-Financeira;
- eSocial;

- Manifesto Eletrônico de Documentos Fiscais (MDF-e);
- Nota Fiscal de Serviços Eletrônica (NFS-e);
- Nota Fiscal Eletrônica (NF-e);
- Nota Fiscal de Consumidor Eletrônica (NFC-e);

Para os fins deste estudo, serão abordados apenas os módulos NF-e e NFC-e.

2.1.1 NF-e (modelo 55) e NFC-e (modelo 65)

A Nota Fiscal Eletrônica (NF-e) e a Nota Fiscal de Consumidor Eletrônica (NFC-e) foram desenvolvidas de forma integrada, pelas Secretarias de Fazenda dos Estados e Secretaria da Receita Federal do Brasil, sob o projeto NF-e, coordenado pelo Encontro Nacional de Coordenadores e Administradores Tributários Estaduais (ENCAT), que foi responsável por seu desenvolvimento e implantação.

O projeto foi encerrado em dezembro de 2010, quando foi alcançado o seu objetivo de implantar um modelo nacional de documento fiscal eletrônico, substituindo a sistemática de emissão do documento fiscal em papel, de forma a simplificar as obrigações acessórias dos contribuintes e permitir, ao mesmo tempo, o acompanhamento em tempo real das operações comerciais pelo Fisco.

O manual de orientação ao contribuinte[5], disponível em [4] apresenta a seguinte definição para a NF-e modelo 55:

> Nota Fiscal Eletrônica (NF-e) é um documento de existência exclusivamente digital, emitido e armazenado eletronicamente, com o intuito de documentar uma operação de circulação de mercadorias ou prestação de serviços, nos campos de incidência do Imposto sobre Operações relativas à Circulação de Mercadorias e Prestação de Serviços de transporte Interestadual e Intermunicipal e de Comunicação (ICMS) e do Imposto Sobre Produtos Industrializados (IPI), cuja validade jurídica é garantida por duas condições necessárias: a assinatura digital do emitente e a autorização de Uso fornecida pela administração tributária do domicílio do contribuinte, que poderá ser utilizada em substituição:
>
> 1. à Nota Fiscal, modelo 1 ou 1-A;
> 2. à Nota Fiscal de Produtor, modelo 4.

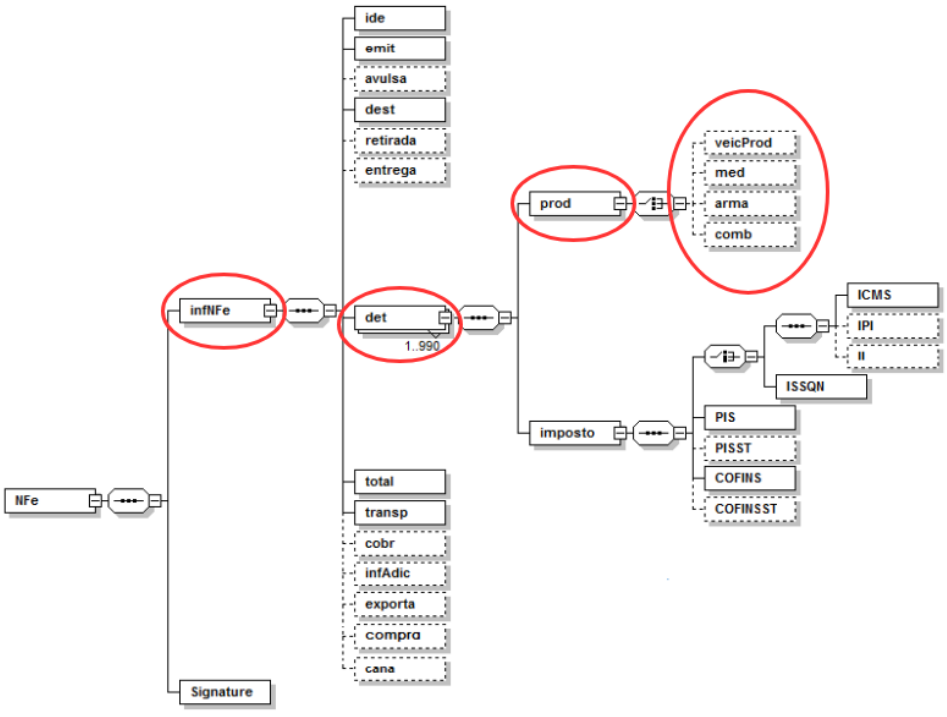
A Nota Fiscal de Consumidor Eletrônica (NFC-e), identificada como modelo 65, também é definida como documento emitido e armazenado eletronicamente, de existência apenas digital, cuja validade jurídica é garantida pela assinatura digital do emitente e autorização de uso pela administração tributária da unidade federada do contribuinte, mas enquanto o modelo 55 documenta operações de circulação de mercadorias e prestação de serviços, o modelo 65 trata de operações comerciais de venda presencial ou venda para entrega em domicílio a consumidor final (pessoa física ou jurídica) em operação interna e sem geração de crédito de ICMS ao adquirente. [3][5]
Os documentos que a NFC-e substitui, a critério da unidade federada, são:

1. A Nota Fiscal de Venda a Consumidor, modelo 2;
2. O Cupom Fiscal emitido por equipamento Emissor de Cupom Fiscal (ECF);
3. O Cupom Fiscal Eletrônico – SAT (CF-e-SAT).

2.1.2 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. [2]

Figura 1: Diagrama simplificado dos grupos de informações da NF-e.



Fonte: Adaptada de [2]

No schema disponível em [3] é utilizada linguagem de marcação XML para definir o conteúdo do documento eletrônico e a sua organização. Na figura 1, 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. [1]

Na versão 5.0 do manual de orientação do contribuinte [6], a partir da página 148, podemos encontrar uma tabela que detalha os nomes padronizados dos mais de 400 campos possíveis de uma NF-e, como por exemplo, a IE, correspondente a inscrição estadual do contribuinte junto às secretarias de fazenda. É através de qual grupo os campos pertencem no Schema XML que é realizada a diferenciação dos campos. [2]

2.2 Sistema Gerenciador de Banco de Dados

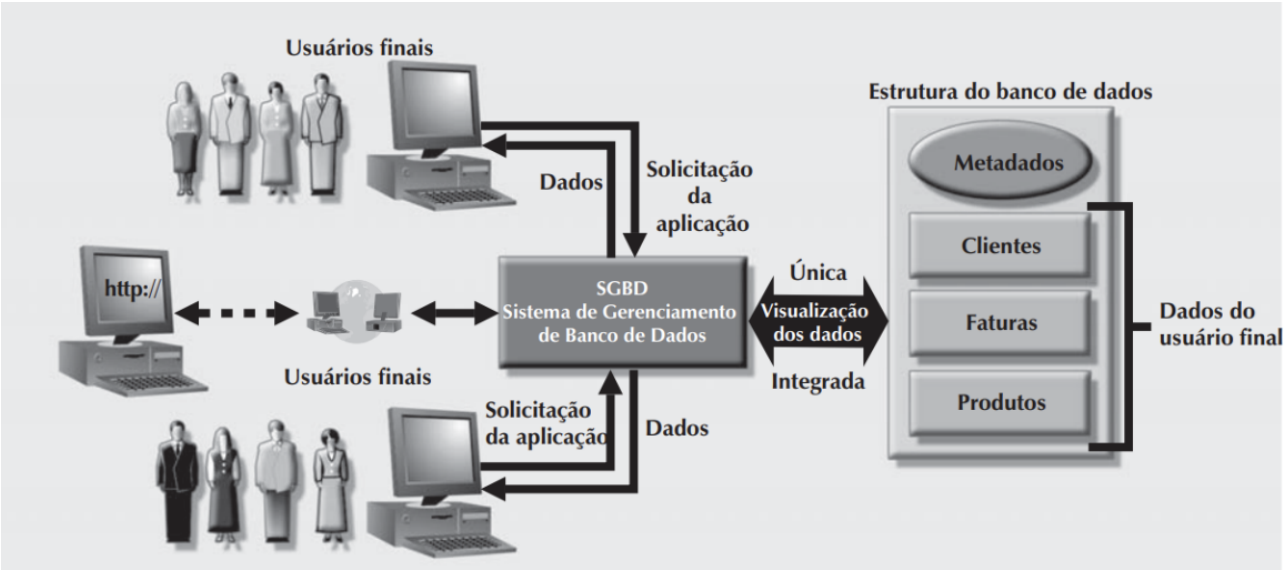
O SGBD (sistema gerenciador de banco de dados) se trata de uma coleção de programas que tem como função o gerenciamento da estrutura de um banco de dados e controle ao acesso dos dados armazenados. O banco de dados tem sua estrutura armazenada como um conjunto de arquivos, que só podem ser acessados por intermédio do SGBD, portanto, este serve como um intermediário entre o usuário e o banco de dados. A 2 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. [8][1]

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. [1]

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 [9].

Figura 2: Funcionamento do SGBD



Fonte: Adaptada de [8]

2.2.2 Indexação em SGBDs

Um índice com um dado atributo de uma relação, é uma estrutura de dados que torna eficiente a busca por tuplas que tenham um valor fixo para aquele atributo. Nós podemos pensar no índice como uma arvore de busca binária de pares chave-valor, onde uma chave (um dos valores que o atributo possa ter) é associada a um valor, que é o conjunto de localizações das tuplas que tenham a chave como componente para o atributo. Note que a chave para o índice pode ser qualquer atributo ou conjunto de atributos, e não precisa ser o atributo chave da relação sobre a qual o índice é construído. [17]

Índices do tipo hash são baseados em uma lista ordenada de valores hash extraídos de uma coluna chave. Esse valor aponta para um registro em uma tabela hash, que por sua vez, aponta para a posição da tupla na tabela em si. Este tipo de índice é recomendado para consultas rápidas baseadas em condições de igualdade. [8]

O índice de árvore balanceada ou B-tree, é uma estrutura de dados ordenada como uma árvore invertida. A árvore é armazenada de forma separada dos dados e as folhas de baixo nível contém as referências às tuplas da tabela. Seu auto-balanceamento garante que o acesso à qualquer registro leve aproximadamente a mesma quantidade de tempo. [8]

2.3 Arquitetura de 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. [20][1]

2.3.1 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. [1]

2.3.2 Dados Semiestruturados e Não Estruturados

Em via de regra, dados semiestruturados 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. [1]

Dados não estruturados 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. [1]

2.4 Big Data

O termo big data refere-se legitimamente a conjuntos de dados cujo tamanho está além da capacidade típica das ferramentas de software de bancos de dados tradicionais: capturar, armazenar, gerenciar e analisar. No ambiente atual, o tamanho dos conjuntos de dados que podem ser considerados como big data varia de terabytes até exabytes. A noção do que é big data depende do setor, de como os dados são usados, de quantos dados históricos estão envolvidos e de muitas outras características. Em 2011, o Gartner Group caracterizou big data em três aspectos: Volume, velocidade e variedade. Mais tarde outras características, como veracidade e valor, seriam adicionadas à definição. [9]

Volume: O volume de dados se refere ao tamanho total dos dados gerenciados

pelo sistema. Nota-se hoje uma cultura de geração e acúmulo de dados em escala massiva, cujo processamento e armazenamento requer estratégias especializadas.

Velocidade: A definição de big data vai além da dimensão do volume, incluindo os tipos e a frequência de dados que afetam as ferramentas tradicionais de gerenciamento de banco de dados. Pode-se descrever a velocidade como a taxa na qual os dados são criados, acumulados, ingeridos e processados.

Variedade: Big data inclui dados estruturados, semiestruturados e não estruturados em diferentes proporções com base no contexto, isso inclui desde os registros de um banco relacional até vídeos ou e-mails. Bancos de dados homogêneos não costumam caracterizar big data, mesmo que armazenem grandes quantidades de dados e sejam alimentados ou consumidos em alta velocidade.

Veracidade: O conceito de veracidade no contexto de big data está intimamente relacionada à confiança. Muitas fontes geram dados incertos, incompletos e imprecisos, tornando sua veracidade questionável, fazendo necessário algum grau de verificação dos dados durante o processo de carga. A veracidade tem dois recursos integrados: a credibilidade da fonte e a adequação dos dados ao público-alvo.

Valor: Big data não é uma aglomeração aleatória de dados, a informação obtida da sua análise deve gerar valor àqueles que a consultam. O papel dessa característica é destacar que os dados coletados devem ser relevantes.

3 METODOLOGIA

3.1 Ambiente de Testes

Os ambientes de teste foram montados conforme a disponibilidade de recursos para o trabalho, esses consistiram de sete máquinas virtuais configuradas em um cluster docker swarm e outra máquina virtual hospedando um dos bancos controle.

O cluster docker swarm contou com um manager e seis workers, todos com as seguintes configurações:

- 2 núcleos de um Intel Xeon Gold 6330 à 1.995GHz;
- 2GB de memória RAM;
- 500GB de armazenamento em disco rígido;
- Debian GNU/Linux 11 (bullseye) x86_64, kernel 5.10.0-14-amd64

A máquina com o banco controle possuía o dobro de recursos de um nó do cluster, tendo:

- 4 núcleos de um Intel Xeon Gold 6330 à 1.995GHz;
- 4GB de memória RAM;
- 100GB de armazenamento em disco rígido;
- Debian GNU/Linux 11 (bullseye) x86_64, kernel 5.10.0-14-amd64

Todas as instâncias de banco de dados utilizadas foram executadas em contêiner docker a partir da imagem oficial do postgres mais recente, referente à versão 15.1 do PostgreSQL.

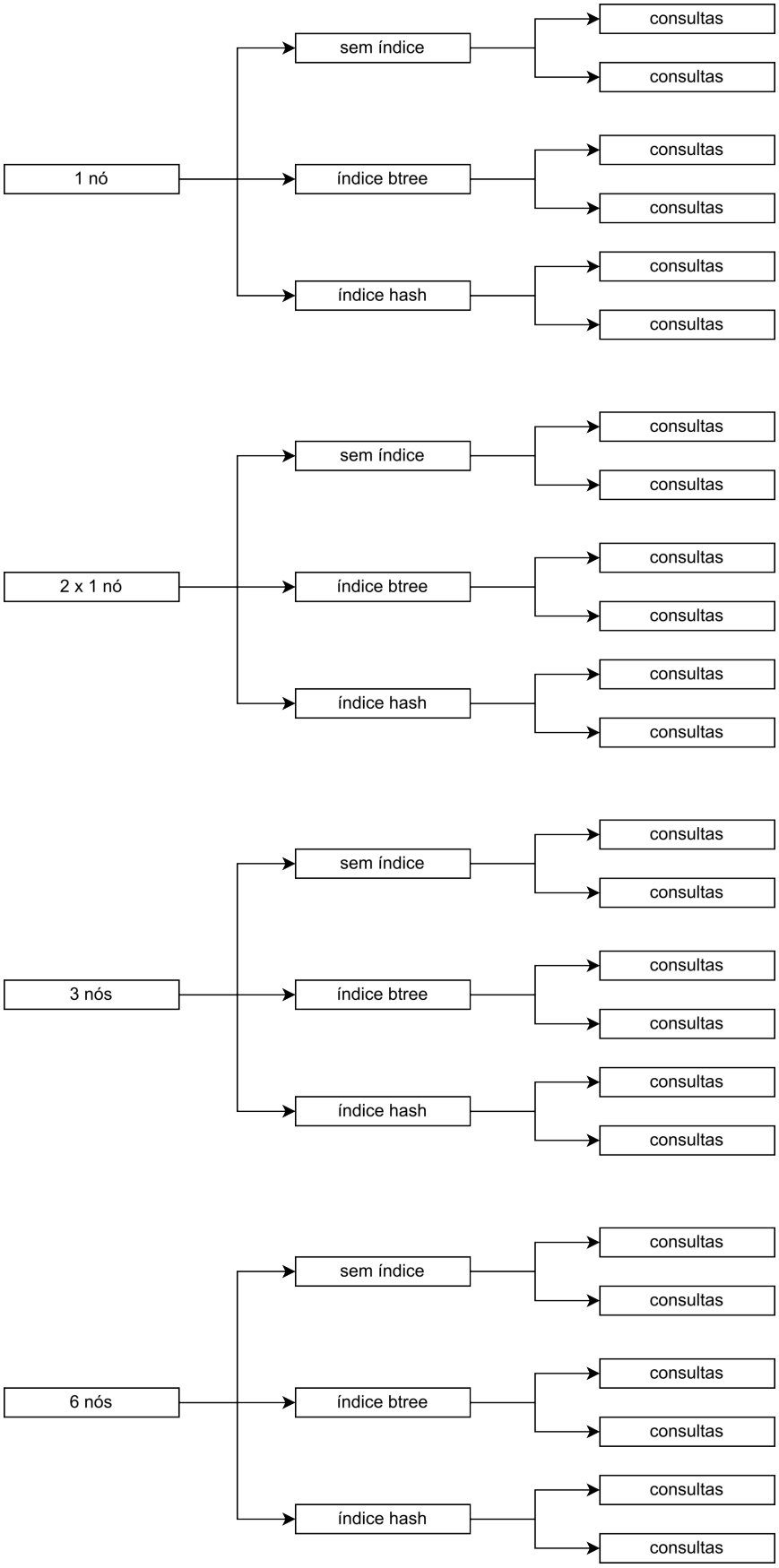
3.2 Cenários de Teste

Foram observados doze cenários de teste, com variações na quantidade de nós do cluster e tipo de índice utilizado, ilustrados no diagrama da figura 3.

Em cada cenário, foram executadas 79 consultas, com variações no dado a ser recuperado, em um filtro sobre o campo referente à data de emissão, e outro filtro referente ao CNPJ do emitente. A cada consulta foi atribuído um código, conforme ilustrado nas figuras 4 e 5.

O campo recuperado varia entre registro completo, campo específico e três funções de agregação sobre um campo específico, com o intuito de verificar o impacto do volume de dados contido no resultado, com o registro completo retornando um grande volume

Figura 3: Diagrama dos cenários de teste observados.



Fonte: Próprio autor

de dados, o registro específico apresentando um volume de dados reduzido, e as funções de agregação entregando apenas um valor. Para facilitar a visualização, o diagrama de consultas foi dividido em duas partes, com as consultas Q32 à Q80, referentes às funções de agregação, condensadas na figura 5, as consultas Q32 à Q47 utilizaram a função de agregação AVG, Q48 à Q63 utilizaram SUM, e Q64 à Q79 aplica a função COUNT. A figura 4 contém as demais consultas, com o intervalo de Q00 a Q15 retornando registros completos, e o intervalo de Q16 a Q31 contendo as consultas sobre campo específico id_nfe, que contem um código sequencial para identificar uma nota sem consultar o campo mais sensível infProt_chnfe, referente a chave da NF-e. As consultas completas podem ser encontradas no código 5 em anexo.

Para contornar a interferência de otimizações por caching, cada consulta foi executada três vezes, sendo as primeiras execuções desconsideradas, assim, qualquer ganho de performance por caching beneficia todas as consultas igualmente. A hipótese de que a primeira execução de cada consulta sofreria perda de performance foi confirmada após a execução dos testes preliminares, onde foram constatadas diferenças significativas entre a primeira e segunda execuções de cada consulta, e diferenças desprezíveis entre as duas últimas execuções.

Os valores em cada consulta foram ajustados de acordo com a amostra, para garantir a coleta de dados relevantes em alguns casos específicos, a exemplo da consulta por intervalo de data, que contempla cenários em que o intervalo está ou não contido nos limites de uma partição.

A performance das consultas foi medida pelo critério de tempo de execução, capturado através da opção "\timing"do cliente de linha de comandos do PostgreSQL.

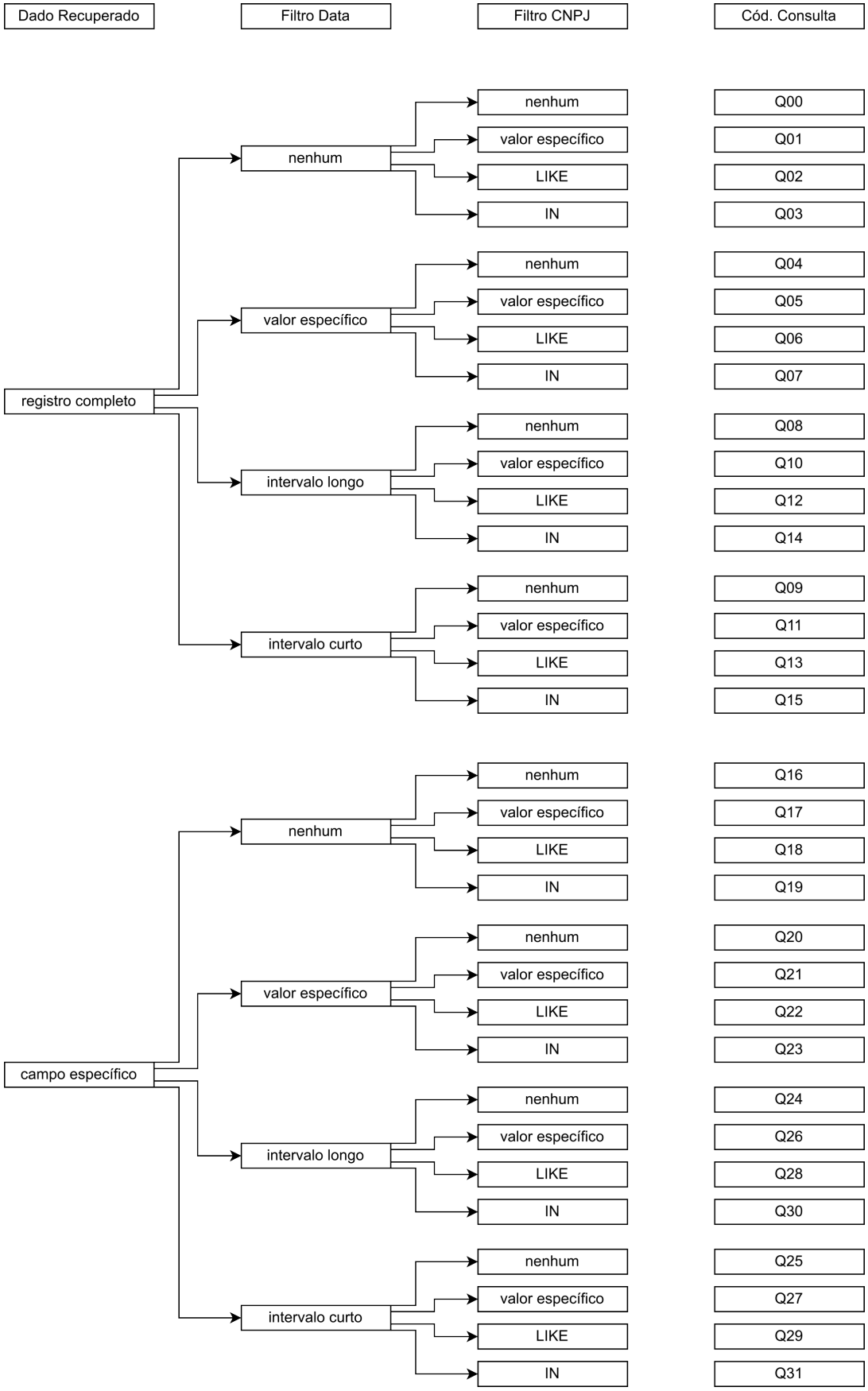
3.3 Amostra

Os testes foram conduzidos sobre uma amostra de notas fiscais eletrônicas modelo 55, contendo cerca de 10 milhões de registros, dispostos em intervalos balanceáveis à quantidade de nós utilizados. Todas os campos contendo informações sensíveis foram anonimizados, incluindo chaves de acesso a notas fiscais, CPFs e CNPJs de emissores e destinatários, assinaturas digitais, entre outros, preservando recorrências em todos os casos relevantes.

3.4 Banco

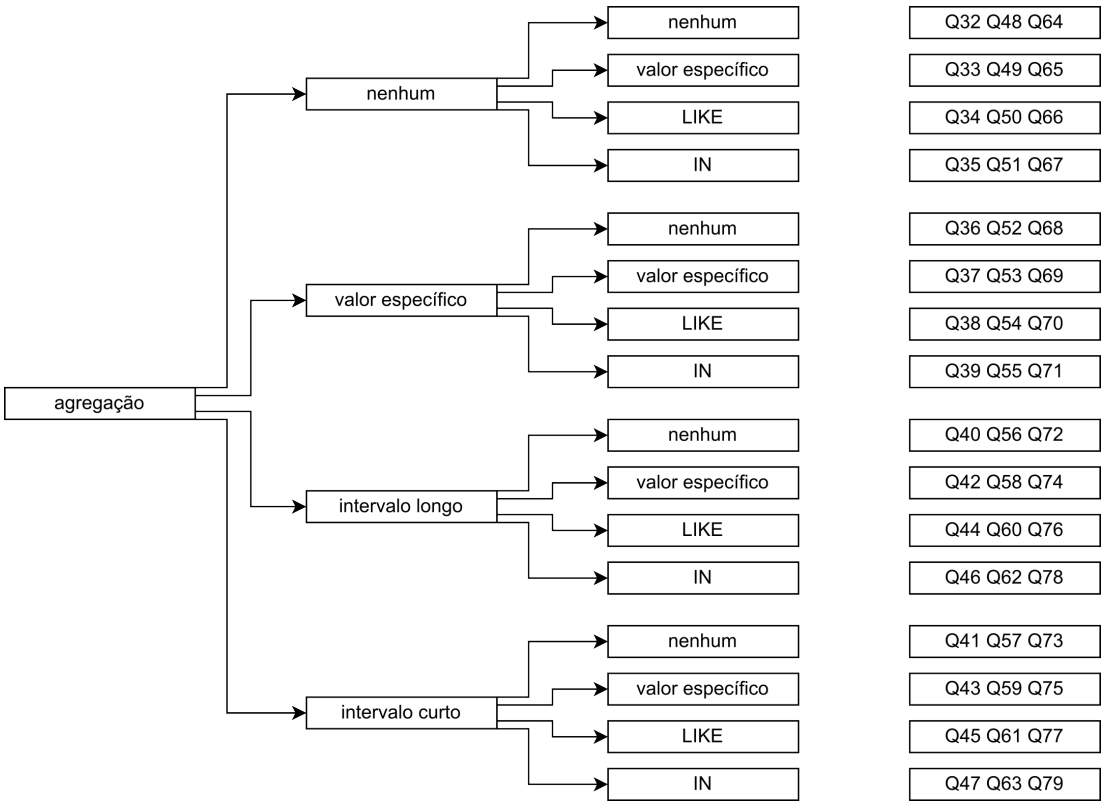
O banco foi modelado de forma horizontal, concentrando o máximo de informações do documento em uma única tabela (nfe), com exceção do grupo "det", referente ao detalhamento de produtos e serviços da nota, que por seu grande grau de variação em

Figura 4: Diagrama das consultas sem funções de agregação.



Fonte: Próprio autor

Figura 5: Diagrama das consultas com funções de agregação.



Fonte: Próprio autor

ocorrências (de 1 à 990 itens por nota), foi armazenado em uma tabela à parte (itemnfe). A estrutura da tabela, assim como a nomenclatura dos campos, foi elaborada a partir do leiaute público da NF-e [4].

A tabela nfe foi particionada por intervalos de data do campo infnfe_ide_dhemi, referente à data de emissão do documento, as partições foram balanceadas manualmente e atribuídas a foreign data wrappers em instâncias PostgreSQL executadas a partir de outros nós do cluster docker. Para a definição do banco, foram utilizados scripts SQL similares aos anexados como código 1, código 2, código 3 e código 4.

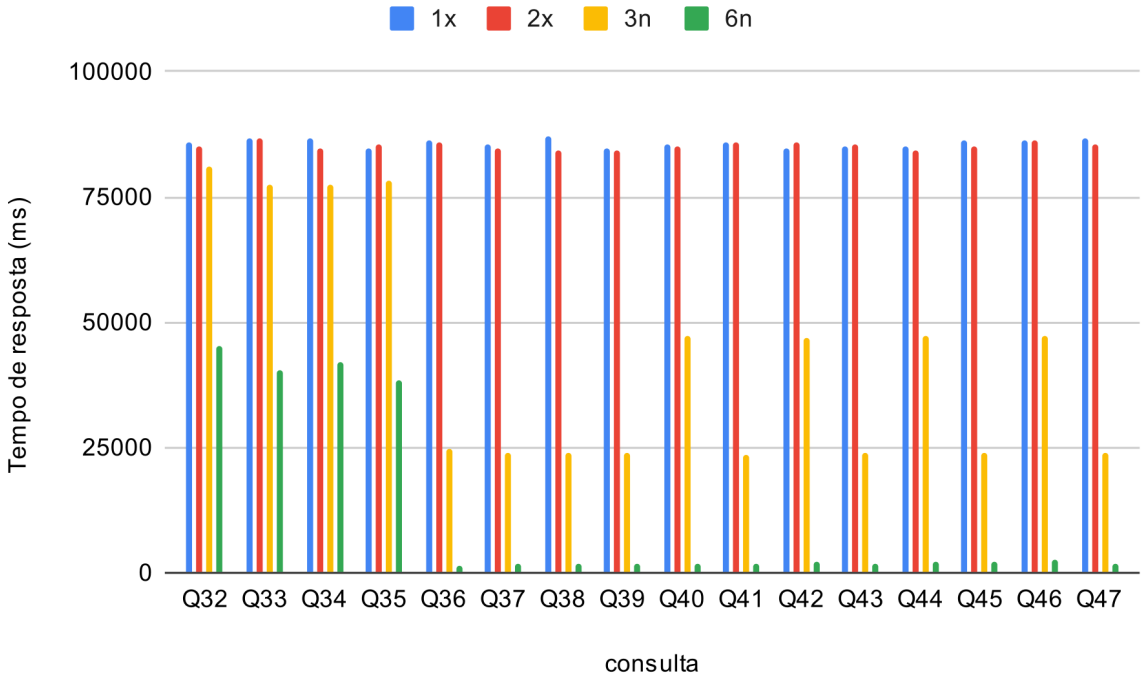
4 APRESENTAÇÃO E ANÁLISE DOS RESULTADOS

Neste capítulo serão apresentados os resultados obtidos, sob o critério de índice utilizado, começando com as consultas sobre campos não indexados, seguidos pelos campos com índice Hash e por fim campos com B-Tree.

Em todos os cenários observados, as consultas com função de agregação apresentaram resultados semelhantes, e assim, para fim de claridade, serão expostos apenas os valores da função de média.

À primeira vista, é possível notar que nos cenários sem índice, onde era necessária uma varredura completa da tabela, a distribuição do banco aumentou a performance de busca significativamente, com o banco de seis nós 6n obtendo ganhos de até 98,20% em relação ao banco monolítico escalado verticalmente 2x (Q36). Mesmo o banco de três nós 3n alcançou vantagem de 72,52% em relação a 2x considerando Q41.

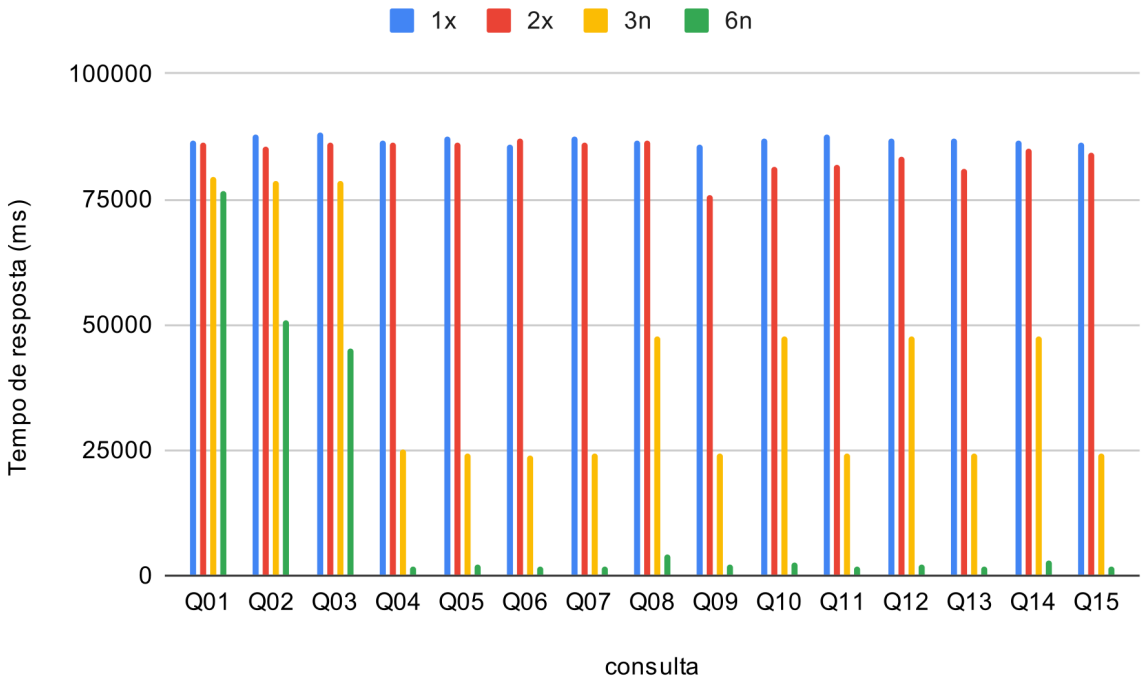
Figura 6: Tempo de resposta sem índice para consultas que recuperam a função AVG



Fonte: Próprio autor

Na maioria dos cenários sem índice, o tempo de resposta dos bancos monolíticos se manteve por volta de 85 segundos, equivalente ao tempo de varredura completa da tabela de forma sequencial, sem que o escalonamento vertical do banco 2x lhe ofereça vantagem sobre o banco 1x.

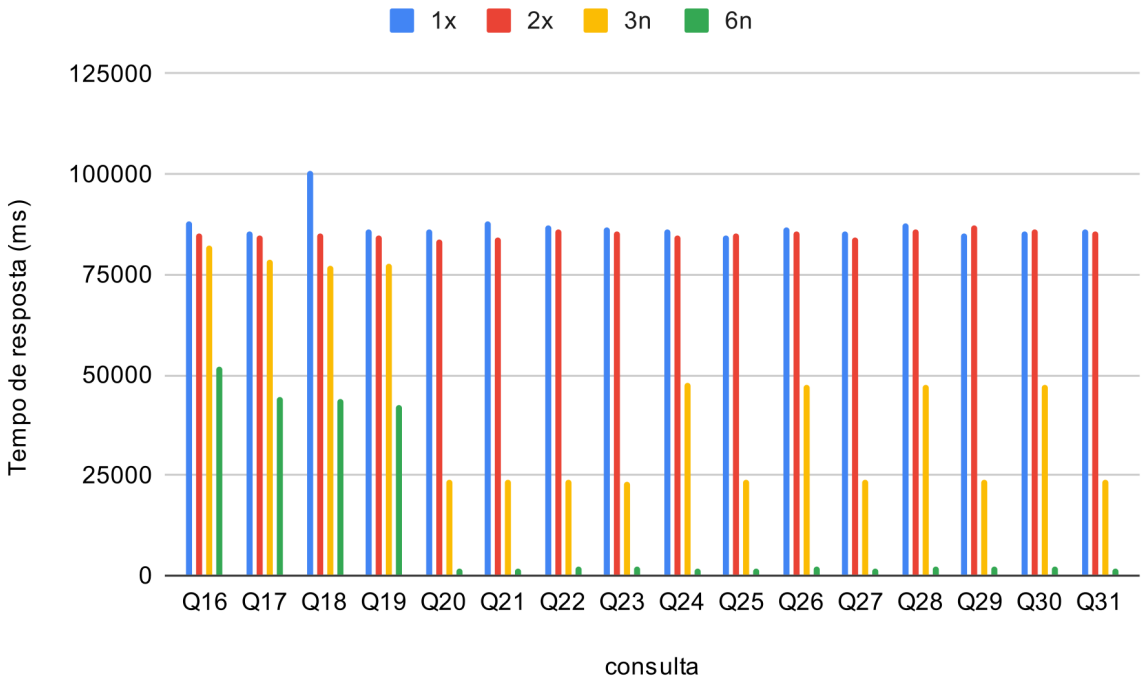
Figura 7: Tempo de resposta sem índice para consultas que recuperam registros completos



Fonte: Próprio autor

Enquanto os bancos monolíticos apresentaram performance uniforme, os bancos distribuídos demonstraram oscilações significativas a depender da consulta, apresentando ganhos menores em relação aos bancos controle nas consultas sem um filtro de data. Como exemplo, nas consultas Q16 à Q19 presentes no gráfico 8, 6n apresenta um ganho médio de performance de 46,23% em relação a 2x, enquanto em outros casos, como nos das consultas de Q20 à Q24 este valor é de 97,77%. O ponto em comum entre as consultas com ganho de performance reduzido é a ausência de um filtro sobre a chave de particionamento data, que não só aumenta o conjunto de dados a ser percorrido, como complexifica a etapa de agregação dos resultados de cada nó no manager.

Figura 8: Tempo de resposta sem índice para consultas que recuperam um campo específico



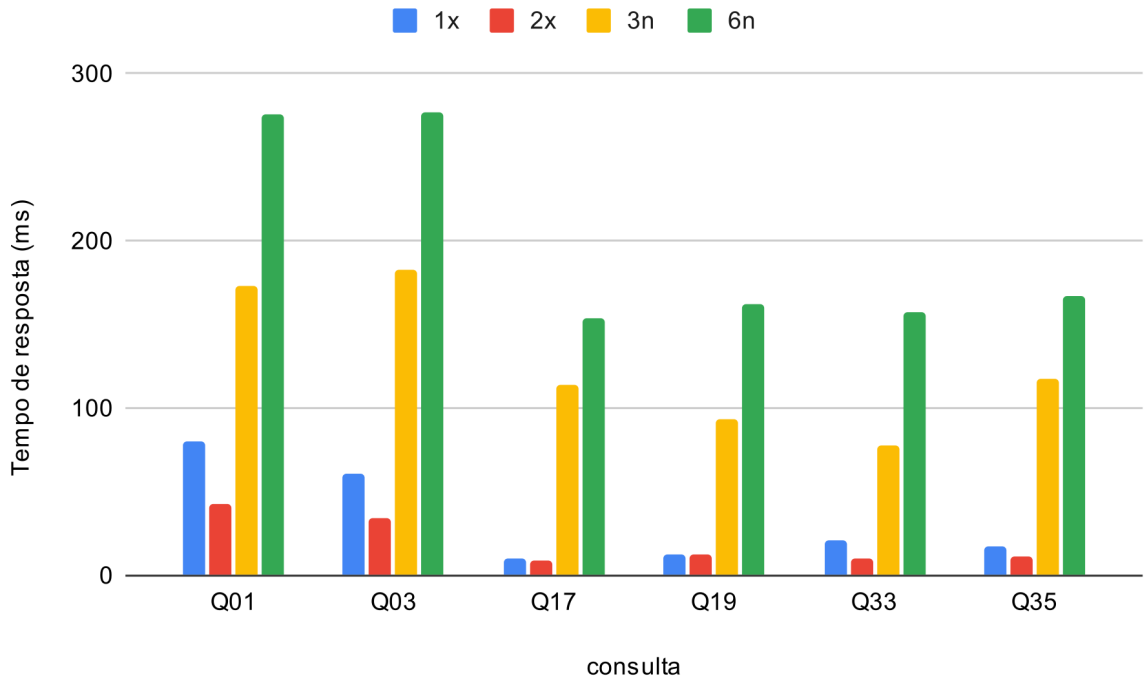
Fonte: Próprio autor

4.1 Buscas com índice Hash

Considerando a inaplicabilidade do índice do tipo hash à consultas baseadas em intervalos, os testes corresponderam às expectativas de que os resultados de consultas que usam apenas filtros de intervalo de valores, seriam similares aos das consultas sem índice algum.

Nas consultas em que o índice hash se faz relevante, foi observado que os bancos distribuídos tiveram performance inferior à dos bancos monolíticos. A perda de performance é particularmente grave nos cenários onde não há um filtro sobre o campo de data, que foi utilizado como chave de partição, nesses casos o banco x1 chegou a apresentar seus resultados mais de 11 vezes mais rápido que o banco 3n e mais de 15 vezes mais rápido que o banco 6n.

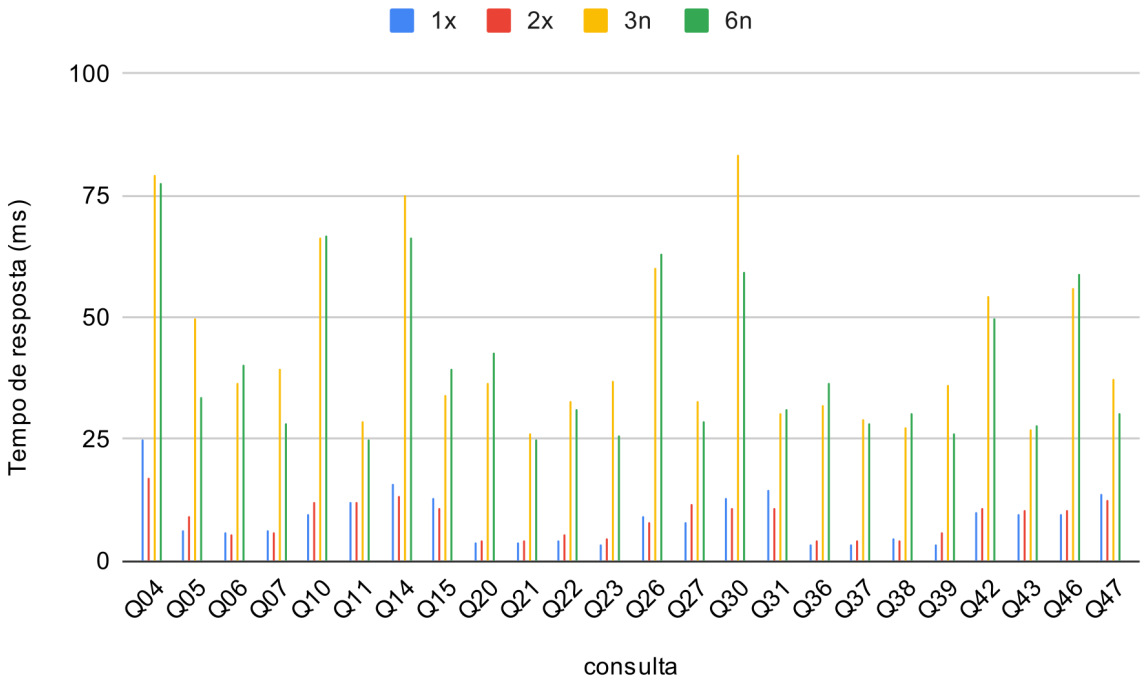
Figura 9: Tempo de resposta com índice hash para consultas efetivas sem um filtro no campo de data



Fonte: Próprio autor

Mesmo desconsiderando as consultas sem filtro na chave de partição apresentadas no gráfico 9, 6n ainda levou em média 5,6 vezes mais tempo que o banco 1x para executar cada uma das consultas restantes. O banco 3n apresentou um tempo de execução médio 6 vezes maior que o de 1x para as mesmas consultas.

Figura 10: Tempo de resposta com índice hash para consultas efetivas com um filtro no campo de data



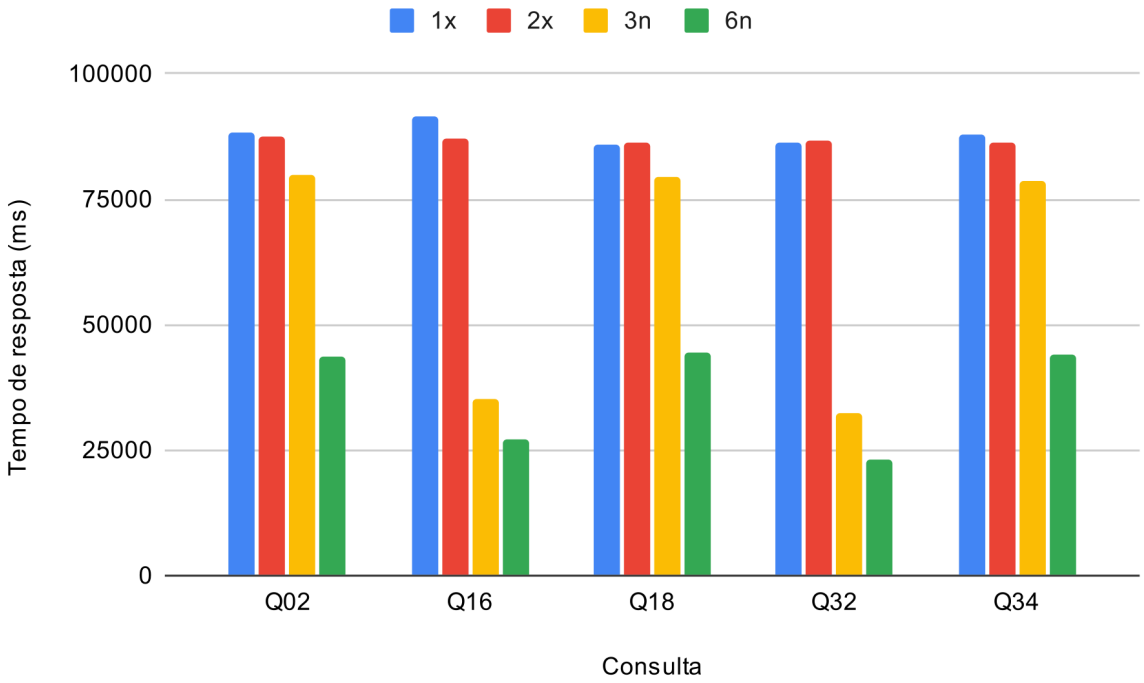
Fonte: Próprio autor

4.2 Buscas com índice B-Tree

Em todos os cenários anteriores, mesmo nos casos em que os bancos distribuídos apresentavam perda de desempenho proporcional significativa em relação aos bancos monolíticos, o valor absoluto do tempo de execução das consultas ainda podia ser considerado instantâneo nos casos desfavoráveis aos bancos distribuídos, enquanto os casos favoráveis agilizavam consultas particularmente problemáticas, em que o atraso no tempo de resposta era evidente ao usuário final, mesmo nos bancos monolíticos.

Este comportamento também pôde ser observado nos cenários com índice B-Tree, que apresentaram apenas cinco consultas consideradas lentas (gráfico 11), sendo elas as três variações de consultas onde não há filtro de data e o filtro de CNPJ utiliza o operador LIKE (Q02, Q18 e Q34), e as duas consultas onde não há filtro de data ou de CNPJ (Q16 e Q32). Seja o grande volume de dados do segundo conjunto ou a exigência de uma varredura sequencial do primeiro, a distribuição do banco trouxe ganho de performance por meio da paralelização, mesmo quando não foi possível extrair benefício da segmentação dos dados.

Figura 11: Tempo de resposta com índice B-Tree para consultas com varredura completa da tabela

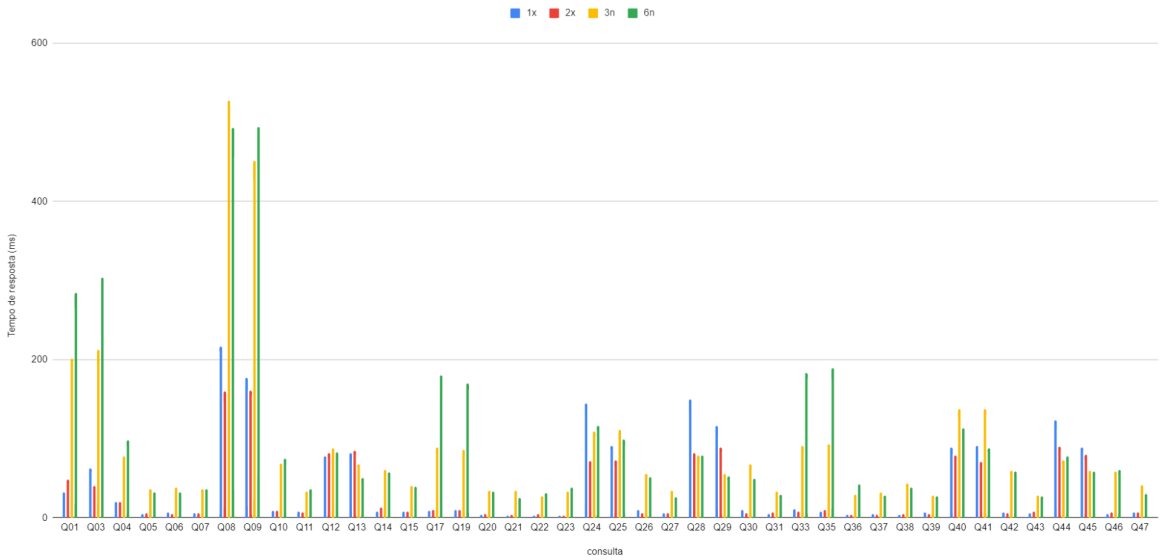


Fonte: Próprio autor

Dos 42 casos em que as consultas foram consideradas instantâneas (consultas sem varredura completa da tabela, dispostas no gráfico 12) e utilizaram um índice B-Tree, apenas 10 apresentaram um desempenho dos bancos distribuídos similar ou superior ao dos bancos monolíticos. Entre esses 10 casos, foi observado que os bancos distribuídos tiveram um ganho máximo de desempenho de 54,47% (Q29), enquanto nos casos em que ocorreu uma perda de desempenho nos bancos distribuídos, houve um cenário onde o banco monolítico 1x executou a consulta mais de 23 vezes mais rápido que o banco distribuído 6n (Q35).

Apesar disso, os benefícios da paralelização superam em muito suas desvantagens quando há a possibilidade de consultas com varredura completa da tabela. Em valores absolutos, mesmo a soma de todas as melhorias de desempenho de qualquer dos bancos monolíticos, ainda é uma fração da menor de suas perdas observada nas consultas com varredura completa da tabela.

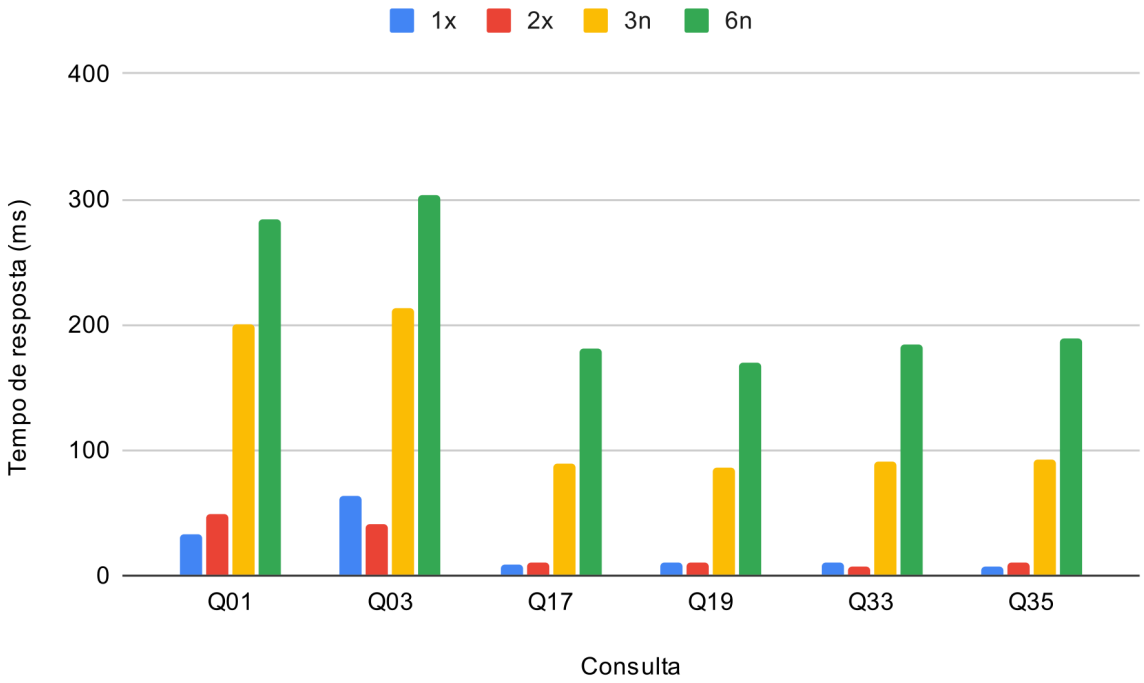
Figura 12: Tempo de resposta com índice B-Tree para consultas sem varredura completa da tabela



Fonte: Próprio autor

Os cenários sem filtro na chave de partição demonstraram um aumento gradual no desempenho dos bancos distribuídos em relação aos bancos monolíticos, de acordo com a exigência das consultas. As consultas do gráfico 11, que acessam todos os registros presentes no banco, apresentam os melhores resultados, enquanto as consultas delimitadas do gráfico 12 foram executadas, de forma distribuída, entre 7 e 24 vezes mais lentamente que nos bancos controle. Já nas consultas menos exigentes (Q17, Q19, Q33 e Q35), pôde ser observado que os bancos distribuídos apresentam tempo de resposta proporcional à quantidade de nós, o que sugere que a maior parte desse valor seja composto pelo custo de acesso às partições remotas, permitindo uma estimativa grosseira do "*overhead*"gerado para cada nó acessado de cerca de 30 ms.

Figura 13: Tempo de resposta com índice B-Tree para consultas sem varredura completa da tabela e sem um filtro para o campo de data

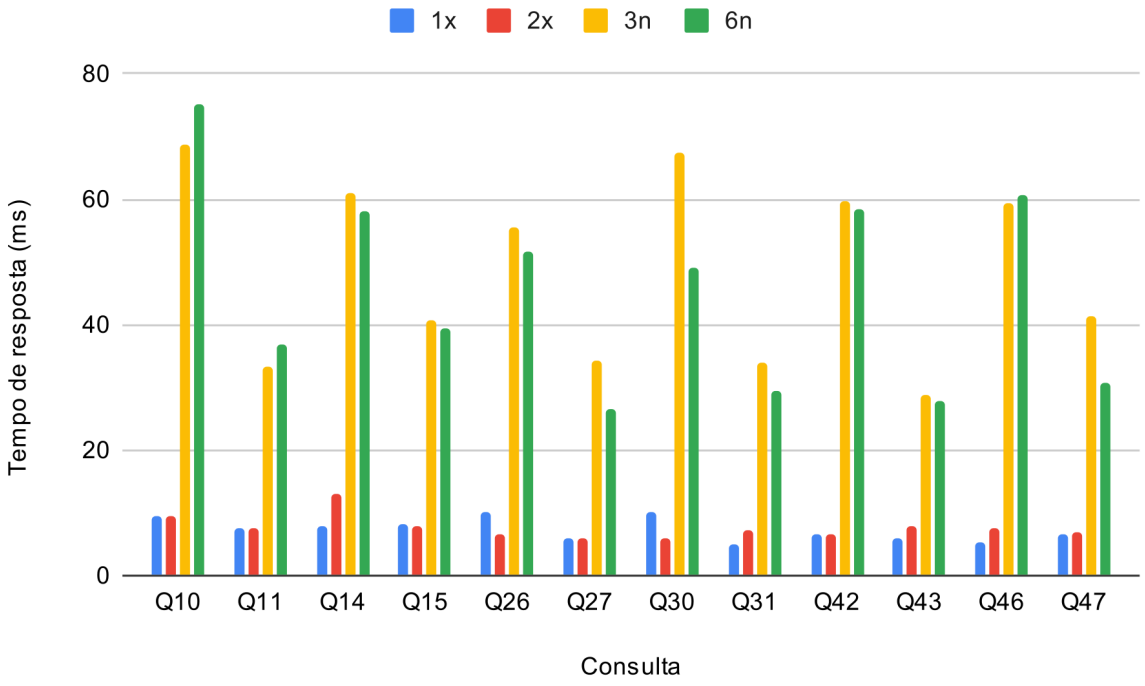


Fonte: Próprio autor

O gráfico 14 mostra consultas com filtros de conjunto de valores e valor específico no campo de CNPJ. Essas consultas abrangem dois intervalos de datas, sendo que as consultas pares representam um intervalo longo e exigem acesso a dois nós nos bancos de dados distribuídos, enquanto as consultas ímpares são limitadas a um único nó em um intervalo curto.

Graças às otimizações fornecidas pelo índice B-Tree e ao baixo volume de dados retornados pelas consultas, as diferenças de desempenho para os dois intervalos não foram muito significativas se considerado o tempo de acesso por nó observado anteriormente. As consultas limitadas a um único nó apresentaram entre 44% e 69% do tempo de resposta do intervalo longo, com atraso em relação aos bancos de controle variando entre 20 e 35 milissegundos.

Figura 14: Tempo de resposta com índice B-Tree para consultas com filtros de intervalo para o campo de data e filtros IN e de valor específico para o campo de CNPJ



Fonte: Próprio autor

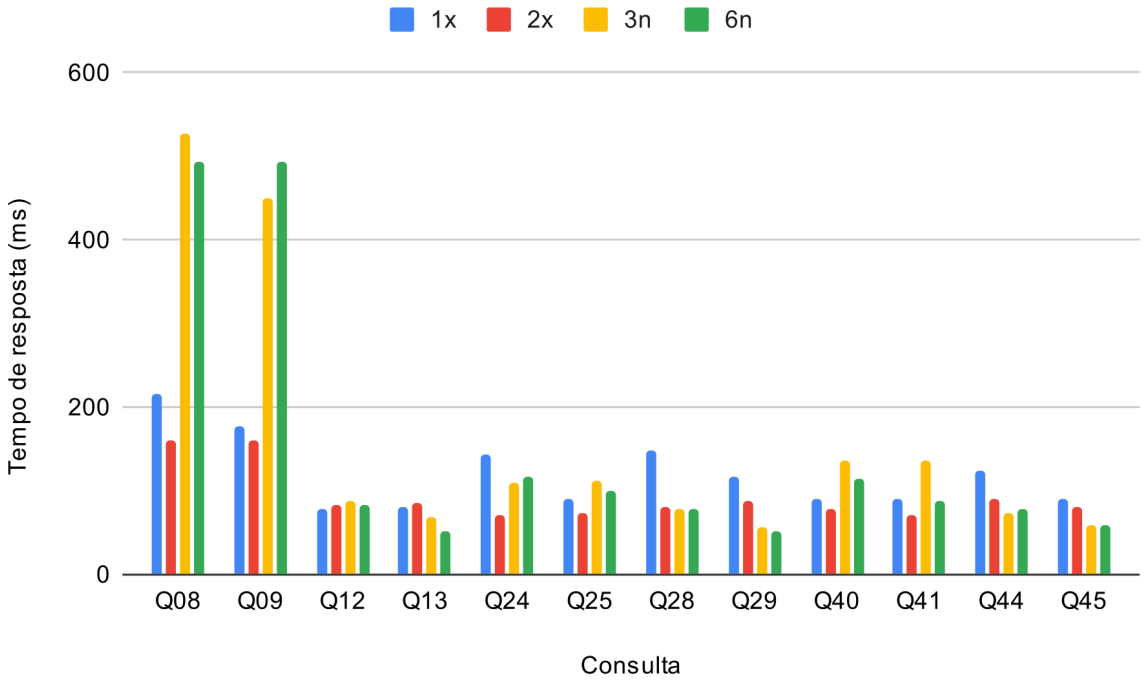
De forma similar ao 14, o gráfico 15 representa dois intervalos de datas, mas com os cenários onde o campo de CNPJ é filtrado pelo operador LIKE ou não é filtrado, permitindo demonstrar a diferença no custo da transmissão de um volume maior de dados, assim como os efeitos de um "*scan*"sequencial entre consultas delimitadas a um único nó ou dispersas em dois.

As consultas Q08 e Q09, que retornam todos os registros dos intervalos logo e curto sem nenhum filtro no campo CNPJ, se destacam das outras consultas no gráfico 15 devido a seus tempos de resposta altos (em média 5,65 vezes mais longos do que o tempo de execução das outras consultas). Isso mostra que a recuperação e transmissão de uma grande quantidade de dados têm um impacto significativo no desempenho dos bancos de dados remotos. O banco de dados 3N apresentou uma pequena perda de desempenho (14,43%) na consulta do intervalo longo Q08 em relação à consulta do intervalo curto Q09. Isso pode ser atribuído ao maior intervalo de particionamento de 3N em comparação com o banco de dados 6N, que teve um desempenho semelhante nos dois cenários. Isso também reforça que a maior parte da diminuição de desempenho nos bancos de dados distribuídos nessas consultas é causada pela transmissão dos resultados.

O filtro LIKE, utilizado nas consultas Q12, Q13, Q28, Q29, Q44 e Q45 do gráfico

15, ilustra como mesmo em situações menos extremas, consultas mais complexas tendem a beneficiar os bancos de dados distribuídos, reduzindo as desvantagens causadas pelos custos da comunicação entre os nós. Ao limitar o conjunto de dados em que a pesquisa sequencial requerida pelo filtro é executada, os bancos de dados distribuídos conseguem fornecer resultados em tempo semelhante ou até mais rápido do que os bancos de dados monolíticos, mesmo considerando o atraso dobrado causado pela transmissão de dados divididos entre dois nós.

Figura 15: Tempo de resposta com índice B-Tree para consultas com filtros de intervalo para o campo de data e filtros LIKE ou sem filtro para o campo de CNPJ



Fonte: Próprio autor

Em suma, os bancos distribuídos demonstraram ter utilidade em consultas pouco otimizadas ou filtradas por um campo não indexado, contudo, esses cenários são ativamente evitados em contextos reais, diminuindo a atratividade desse tipo de abordagem.

Também foi observado que consultas simples podem ser complexificadas, como o caso das funções de agregação que por padrão não recebem "*push down*" para os nós, exigindo a transmissão de um volume de dados maior que o necessário e tornando-as tão onerosas quanto as consultas que recuperam um campo específico.

Ao realizar consultas eficientes, o custo mínimo de comunicação entre os nós, apesar de baixo, se mostrou evidente, comprometendo a escalabilidade do sistema em cenários de alto volume de consultas sem varreduras sequenciais. Já em consultas mais exigentes,

mesmo quando há oportunidade de ganho de performance por paralelização, as vantagens são mitigadas pelo alto custo de transmissão dos resultados ao manager.

5 CONCLUSÕES E TRABALHOS FUTUROS

A fim de Investigar os efeitos da distribuição em um banco de dados PostgreSQL de documentos fiscais digitais, a pesquisa comparou a performance de quatro configurações de uma mesma base em três níveis de distribuição e diferentes graus de otimização de consultas, constatando impacto negativo na performance das bases distribuídas na maioria dos cenários mais relevantes.

O estudo gerou um retorno direto à sociedade, já que pôde informar o processo de tomada de decisões de projeto em ferramentas de apoio à processos de um órgão público, impactando tanto a qualidade destas, como sua demanda de recursos.

Entre as limitações da pesquisa, pode-se apontar a ausência de alguns importantes cenários de consulta, como JOINS, outros graus de acesso a nós específicos e às partições padrões, assim como o efeito de um particionamento local. Outros aspectos sujeitos a melhoria são a confiabilidade e reprodutibilidade dos testes, que dispuseram de hardware especializado e foram executados em ambiente sujeito à disputa de recursos de processamento, rede e armazenamento.

Trabalhos futuros poderiam elaborar uma amostra capaz de demonstrar o efeito de volumes específicos de dados, gerando um melhor panorama dos custos de transmissão dos resultados, ou explorar a viabilidade de outros modelos de particionamento, como a proposta de armazenamento local das partições e métodos de arquivamento de dados históricos.

REFERÊNCIAS

[1] SANTOS, Daniel Fagner Pedroza **Um estudo comparativo sobre uso de modelos de dados para notas fiscais eletrônicas**, 2021. Dissertação – Departamento de Informática, UFPB, João Pessoa/PB, 2021.

[2] SANTOS, Evandro Manzano dos. **Uso de dados de documentos fiscais eletrônicos para o planejamento do transporte urbano de cargas**, 2015. Tese – Departamento de Engenharia Civil e Ambiental, UnB, Brasília/DF, 2015.

[3] R. F. Do Brasil, "Sistema Público de Escrituração Digital (SPED)", 2022. Disponível em: <http://sped.rfb.gov.br/pasta/show/10>. Acesso em: 6 abr. 2022.

[4] R. F. Do Brasil, "Portal da Nota Fiscal Eletrônica", 2022. Disponível em: <http://sped.rfb.gov.br/pasta/show/10>. Acesso em: 6 abr. 2022.

[5] ENCAT, "Manual de Orientação ao Contribuinte - MOC - versão 7.0 - NF-e e NFC-e"2020. Disponível em: <https://www.nfe.fazenda.gov.br/portal/exibirArquivo.aspx?conteudo=LrBx7WT9PuA=>. Acesso em: 8 abr. 2022

[6] ENCAT, "Manual de Orientação do Contribuinte - Padrões Técnicos de Comunicação versão 5.0"2012. Disponível em: <https://www.nfe.fazenda.gov.br/portal/exibirArquivo.aspx?conteudo=HO2V%20vzhFBk=>. Acesso em: 8 abr. 2022

[7] BRAGHITTONI, R. **Business intelligence: implementar do jeito certo e a custo zero**. Casa do Código, 2017.

[8] CORONEL, C.; MORRIS, S.; ROB, P. **Database systems: design, implementation, and management**. 9th ed. ed. Australia; United States: Course Technology Cengage Learning, 2011.

[9] ELMASRI, R.; NAVATHE, S. B. **Sistemas de banco de dados**. 4. ed. São Paulo Pearson Addison Wesley, 2008.

[10] Takai, O. K.; Italiano, I.C.; Ferreira, J.E. **Introdução a banco de dados**. DCC-IME-USP, 2005

[11] KIMBALL, R. **The data warehouse lifecycle toolkit**. 2nd ed. Indianapolis, IN: Wiley Pub., 2008.

[12] KIMBALL, R.; ROSS, M. **The data warehouse toolkit: the definitive guide to dimensional modeling**. Third edition ed. Indianapolis, IN: John Wiley & Sons, Inc, 2013.

[13] VIDA, Edinilson da S.; ALVES, Nicolli S R.; FERREIRA, Rafael G C.; et al. **Data warehouse**. [Digite o Local da Editora]: Grupo A, 2021. *E-book.* ISBN 9786556901916. Disponível em: https://integrada.minhabiblioteca.com.br/#/books/9786556901916/. Acesso em: 07 abr. 2022.

[14] SADALAGE, P. J.; FOWLER, M. **NoSQL distilled: a brief guide to the emerging world of polyglot persistence**. Upper Saddle River, NJ: Addison-Wesley, 2013.

[15] RITCHIE, B. **RavenDB high performance: learn how to accelerate your application development by building scalable applications on the RavenDB document database**. Birmingham, UK: Packt Pub., 2013.

[16] TIWARI, S. **Professional NoSQL**. Indianapolis, Ind: Wiley, 2011.

[17] GARCÍA MOLINA, M.; ULLMAN, J. D.; WIDOM, J.; GARCIA-MOLINA, H. **Database systems: the complete book**. 2. edition, new international edition ed. Harlow: Pearson, 2014.

[18] INMON, W. H. **Building the data warehouse**. 4th ed ed. Indianapolis, Ind: Wiley, 2005.

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

[20] INMON, W. H.; LINST, D.; LEVINS, M. **Data architecture: a primer for the data scientist**. Second edition ed. London [England]; San Diego, CA: Academic Press, 2019.

[21] 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.

[22] 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.

ANEXO A – ANEXOS E APÊNDICES 1

Código 1: Definição da tabela fatoNFe em um dos nós

```
CREATE TABLE app.fatonfe_1(
    id_fatonfe BIGSERIAL NOT NULL,
    infProt_chNFe char(44) PRIMARY KEY,
    infNFe_sqn bigint NOT NULL,
    informix_stnfeletronica char(1) NOT NULL DEFAULT 'A',
    informix_dhconexao varchar NULL,
    informix_nriptransmissor varchar NULL,
    informix_nrportacon int4 NULL,
    infNFe_ide character varying,
    infNFe_ide_cUF integer,
    infNFe_ide_cNF character varying,
    infNFe_ide_natOp character varying,
    infNFe_ide_mod integer,
    infNFe_ide_serie character varying,
    infNFe_ide_nNF character varying,
    infNFe_ide_dhEmi timestamp with time zone,
    infNFe_ide_dhSaiEnt timestamp with time zone,
    infNFe_ide_tpNF integer,
    infNFe_ide_idDest integer,
    infNFe_ide_cMunFG character varying,
    infNFe_ide_tpImp integer,
    infNFe_ide_tpEmis integer,
    infNFe_ide_cDV integer,
    infNFe_ide_tpAmb integer,
    infNFe_ide_finNFe integer,
    infNFe_ide_indFinal integer,
    infNFe_ide_indPres integer,
    infNFe_ide_procEmi integer,
    infNFe_ide_verProc character varying,
    infNFe_emit_CNPJ char(14),
    infNFe_emit_CPF char(11),
    infNFe_emit_xNome character varying,
    infNFe_emit_xFant character varying,
    infNFe_emit_enderEmit_xLgr character varying,
    infNFe_emit_enderEmit_nro character varying,
    infNFe_emit_enderEmit_xCpl character varying,
    infNFe_emit_enderEmit_xBairro character varying,
    infNFe_emit_enderEmit_cMun character varying,
    infNFe_emit_enderEmit_xMun character varying,
    infNFe_emit_enderEmit_UF character varying,
    infNFe_emit_enderEmit_CEP character varying,
    infNFe_emit_enderEmit_cPais character varying,
    infNFe_emit_enderEmit_xPais character varying,
```

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

```
    infNFe_total_ICMSTot_vTotTrib numeric(16,2),
    infNFe_transp_modfrete character varying(1),
    infNFe_transp_vagao character varying(20),
    infNFe_transp_balsa character varying(20),
    infNFe_transp_transporta_cnpj char(14),
    infNFe_transp_transporta_cpf char(11),
    infNFe_transp_transporta_xnome character varying(100),
    infNFe_transp_transporta_ie character varying,
    infNFe_transp_transporta_xender character varying(100),
    infNFe_transp_transporta_xmun character varying(100),
    infNFe_transp_transporta_uf character varying(2),
    infNFe_transp_rettransp_vserv numeric(16,2),
    infNFe_transp_rettransp_vbcret numeric(16,2),
    infNFe_transp_rettransp_picmsret numeric(16,2),
    infNFe_transp_rettransp_vicmsret numeric(16,2),
    infNFe_transp_rettransp_cfop character varying,
    infNFe_transp_rettransp_cmunfg character varying,
    infNFe_transp_veictransp_placa character varying(8),
    infNFe_transp_veictransp_uf character varying(2),
    infNFe_transp_reboque1_placa character varying(8),
    infNFe_transp_reboque1_uf character varying(2),
    infNFe_transp_reboque2_placa character varying(8),
    infNFe_transp_reboque2_uf character varying(2),
    infNFe_transp_reboque3_placa character varying(8),
    infNFe_transp_reboque3_uf character varying(2),
    infNFe_transp_reboque4_placa character varying(8),
    infNFe_transp_reboque4_uf character varying(2),
    infNFe_transp_reboque5_placa character varying(8),
    infNFe_transp_reboque5_uf character varying(2),
    infNFe_transp_vol1_nVol character varying(60),
    infNFe_transp_vol1_qVol numeric(16,2),
    infNFe_transp_vol1_esp character varying(60),
    infNFe_transp_vol1_marca character varying(60),
    infNFe_transp_vol1_pesoL numeric(16,2),
    infNFe_transp_vol1_pesoB numeric(16,2),
    infNFe_transp_vol1_lacres_nLacre character varying(60),
    infNFe_cobr_fat_nFat character varying(60),
    infNFe_cobr_fat_vOrig numeric(16,2),
    infNFe_cobr_fat_vDesc numeric(16,2),
    infNFe_cobr_fat_vLiq numeric(16,2),
    infNFe_pag_vTroco numeric(16,2),
    infNFe_infAdic_infAdFisco character varying,
    infNFe_infAdic_infCpl character varying,
    infNFe_infAdic_obsCont_xCampo character varying,
    infNFe_infAdic_obsCont_xTexto character varying
);
```

Código 2: Configuração de acesso aos nós pelo manager

```
CREATE EXTENSION postgres_fdw;
GRANT USAGE ON FOREIGN DATA WRAPPER postgres_fdw to postgres;
GRANT USAGE ON FOREIGN DATA WRAPPER postgres_fdw to usuario;


CREATE SERVER shard01 FOREIGN DATA WRAPPER postgres_fdw
    OPTIONS (
        dbname 'datalake',
        host 'fdw-6n-btree_shard1',
        port '5491',
        async_capable 'true'
    );

--  ...
--  Definição dos servidores de shard02 - shard05

CREATE SERVER shard06 FOREIGN DATA WRAPPER postgres_fdw
    OPTIONS (
        dbname 'datalake',
        host 'fdw-6n-btree_shard6',
        port '5496',
        async_capable 'true'
    );



CREATE USER MAPPING for usuario SERVER shard01
    OPTIONS (user 'EDITADO', password 'EDITADO');

--  ...
--  Mapeamento de usuários dos servidores 02 - 05

CREATE USER MAPPING for usuario SERVER shard06
    OPTIONS (user 'EDITADO', password 'EDITADO');
```

Código 3: Definição da tabela NFe no nó manager

```
create table app.fatonfe(
  id_fatonfe bigserial not null,

    -- ...
  -- Campos omitidos estão presentes no apêndice [ - ]

    infnfe_infadic_obscont_xtexto character varying
) partition by range (infnfe_ide_dhemi);

```

```
10
11 CREATE FOREIGN TABLE app.fatonfe_1 PARTITION OF fatonfe
12     FOR VALUES FROM ('2021-01-01') TO ('2021-04-01')
13     SERVER shard01;
14
15 CREATE FOREIGN TABLE app.fatonfe_2 PARTITION OF fatonfe
16     FOR VALUES FROM ('2021-04-01') TO ('2021-07-01')
17     SERVER shard02;
18
19 CREATE FOREIGN TABLE app.fatonfe_3 PARTITION OF fatonfe
20     FOR VALUES FROM ('2021-07-01') TO ('2021-10-01')
21     SERVER shard03;
22
23 CREATE FOREIGN TABLE app.fatonfe_4 PARTITION OF fatonfe
24     FOR VALUES FROM ('2021-10-01') TO ('2022-01-01')
25     SERVER shard04;
26
27 CREATE FOREIGN TABLE app.fatonfe_5 PARTITION OF fatonfe
28     FOR VALUES FROM ('2022-01-01') TO ('2022-04-01')
29     SERVER shard05;
30
31 CREATE FOREIGN TABLE app.fatonfe_6 PARTITION OF fatonfe
32     FOR VALUES FROM ('2022-04-01') TO ('2022-07-01')
33     SERVER shard06;
34
35
36 CREATE TABLE fatonfe_default PARTITION OF fatonfe default;
```

Código 4: Definição da tabela fatoItemNFe em um dos nós

```
1  CREATE TABLE app.fatoitemnfe(
2    id_fatoitemnfe BIGSERIAL NOT NULL,
3    infProt_chNFe char(44) NOT NULL,
4    infNFe_det_nItem character varying NOT NULL,
5    infNFe_det_prod_cProd character varying,
6    infNFe_det_prod_cEAN character varying,
7    infNFe_det_prod_xProd character varying,
8    infNFe_det_prod_NCM character varying,
9    infNFe_det_prod_CFOP character varying,
10   infNFe_det_prod_uCom character varying,
11   infNFe_det_prod_qCom numeric(16,2),
12   infNFe_det_prod_vUnCom numeric(16,2),
13   infNFe_det_prod_vProd numeric(16,2),
14   infNFe_det_prod_cEANTrib character varying,
15   infNFe_det_prod_uTrib character varying,
16   infNFe_det_prod_qTrib numeric(16,2),
17   infNFe_det_prod_vUnTrib numeric(16,2),
18   infnfe_det_prod_vFrete numeric(16,2),
```

```
  infnfe_det_prod_vSeg numeric(16,2),
  infNFe_det_prod_vDesc numeric(16,2),
  infnfe_det_prod_vOutro numeric(16,2),
  infNFe_det_prod_indTot character varying,
  infnfe_det_prod_comb_cprodanp character varying,
  infnfe_det_prod_comb_descanp character varying,
  infnfe_det_prod_comb_pglp numeric(7,4),
  infnfe_det_prod_comb_pgnn numeric(7,4),
  infnfe_det_prod_comb_pgni numeric(7,4),
  infnfe_det_prod_comb_vpart character varying,
  infnfe_det_prod_comb_codif numeric,
  infnfe_det_prod_comb_qtemp numeric,
  infnfe_det_prod_comb_ufcons character varying(2),
  infnfe_det_prod_comb_cide_qbcprod numeric(16,2),
  infnfe_det_prod_comb_cide_valiqprod numeric(16,2),
  infnfe_det_prod_comb_cide_vcide numeric(16,2),
  infnfe_det_prod_comb_encerrante_nbico character varying,
  infnfe_det_prod_comb_encerrante_nbomba character varying,
  infnfe_det_prod_comb_encerrante_ntanque character varying,
  infnfe_det_prod_comb_encerrante_vencini numeric(16,2),
  infnfe_det_prod_comb_encerrante_vencfin numeric(16,2),
  infNFe_det_imposto_vTotTrib numeric(16,2),
  infNFe_det_imposto_ICMS_orig character varying(1),
  infNFe_det_imposto_ICMS_CST character varying(3),
  infNFe_det_imposto_ICMS_CSOSN character varying,
  infNFe_det_imposto_ICMS_vBCSTRet numeric(16,2),
  infNFe_det_imposto_ICMS_pST numeric(16,2),
  infNFe_det_imposto_ICMS_vICMSSubstituto numeric(16,2),
  infNFe_det_imposto_ICMS_vICMSSTRet numeric(16,2),
  infNFe_det_imposto_ICMS_vBCFCPSTRet numeric(16,2),
  infNFe_det_imposto_ICMS_pFCPSTRet numeric(16,2),
  infNFe_det_imposto_ICMS_vFCPSTRet numeric(16,2),
  infNFe_det_imposto_ICMS_vBCSTDest numeric(16,2),
  infNFe_det_imposto_ICMS_vICMSSTDest numeric(16,2),
  infNFe_det_imposto_ICMS_pRedBCEfet numeric(16,2),
  infNFe_det_imposto_ICMS_vBCEfet numeric(16,2),
  infNFe_det_imposto_ICMS_pICMSEfet numeric(16,2),
  infNFe_det_imposto_ICMS_vICMSEfet numeric(16,2),
  infNFe_det_imposto_ICMS_modBC character varying(1),
  infNFe_det_imposto_ICMS_pRedBC numeric(16,2),
  infNFe_det_imposto_ICMS_vBC numeric(16,2),
  infNFe_det_imposto_ICMS_pICMS numeric(16,2),
  infNFe_det_imposto_ICMS_vICMSOp numeric(16,2),
  infNFe_det_imposto_ICMS_pDif numeric(16,2),
  infNFe_det_imposto_ICMS_vICMSDif numeric(16,2),
  infNFe_det_imposto_ICMS_vICMS numeric(16,2),
  infNFe_det_imposto_ICMS_vBCFCP numeric(16,2),
```

```
  infNFe_det_imposto_ICMS_pFCP numeric(16,2),
  infNFe_det_imposto_ICMS_vFCP numeric(16,2),
  infNFe_det_imposto_ICMS_pFCPDif numeric(16,2),
  infNFe_det_imposto_ICMS_vFCPDif numeric(16,2),
  infNFe_det_imposto_ICMS_vFCPEfet numeric(16,2),
  infNFe_det_imposto_ICMS_modBCST character varying(1),
  infNFe_det_imposto_ICMS_pMVAST numeric(16,2),
  infNFe_det_imposto_ICMS_pRedBCST numeric(16,2),
  infNFe_det_imposto_ICMS_vBCST numeric(16,2),
  infNFe_det_imposto_ICMS_pICMSST numeric(16,2),
  infNFe_det_imposto_ICMS_vICMSST numeric(16,2),
  infNFe_det_imposto_ICMS_vBCFCPST numeric(16,2),
  infNFe_det_imposto_ICMS_pFCPST numeric(16,2),
  infNFe_det_imposto_ICMS_vFCPST numeric(16,2),
  infNFe_det_imposto_ICMS_vICMSDeson numeric(16,2),
  infNFe_det_imposto_ICMS_motDesICMS character varying(3),
  infNFe_det_imposto_ICMS_vICMSSTDeson numeric(16,2),
  infNFe_det_imposto_ICMS_motDesICMSST character varying(3),
  infNFe_det_imposto_ICMS_pBCOp numeric(16,2),
  infNFe_det_imposto_ICMS_UFST character varying(2),
  infNFe_det_imposto_ICMS_pCredSN numeric(16,2),
  infNFe_det_imposto_II_vBC numeric(16,2),
  infNFe_det_imposto_II_vDespAdu numeric(16,2),
  infNFe_det_imposto_II_vII numeric(16,2),
  infNFe_det_imposto_II_vIOF numeric(16,2),
  infNFe_det_imposto_ICMSUFDest_vBCUFDest numeric(16,2),
  infNFe_det_imposto_ICMSUFDest_vBCFCPUFDest numeric(16,2),
  infNFe_det_imposto_ICMSUFDest_pFCPUFDest numeric(16,2),
  infNFe_det_imposto_ICMSUFDest_pICMSUFDest numeric(16,2),
  infNFe_det_imposto_ICMSUFDest_pICMSInter character varying,
  infNFe_det_imposto_ICMSUFDest_pICMSInterPart numeric(16,2),
  infNFe_det_imposto_ICMSUFDest_vFCPUFDest numeric(16,2),
  infNFe_det_imposto_ICMSUFDest_vICMSUFDest numeric(16,2),
  infNFe_det_imposto_ICMSUFDest_vICMSUFRemet numeric(16,2),
  PRIMARY KEY (infProt_chNFe, infNFe_det_nItem)
);
```

Código 5: Lista de consultas executadas

```
-- Q01
SELECT * FROM app.fatonfe WHERE infNFe_emit_CNPJ = '00000000000010';
-- Q02
SELECT * FROM app.fatonfe WHERE infNFe_emit_CNPJ LIKE '0000000000%';
-- Q03
SELECT * FROM app.fatonfe WHERE infNFe_emit_CNPJ IN ('00000000000013', '
    00000000000012', '00000000000011', '00000000000010');
-- Q04
SELECT * FROM app.fatonfe WHERE infNFe_ide_dhEmi = '2021-12-09
```

```
    00:00:00.000';
-- Q05
SELECT * FROM app.fatonfe WHERE infNFe_ide_dhEmi = '2021-12-09
    00:00:00.000' AND infNFe_emit_CNPJ = '00000000000010';
-- Q06
SELECT * FROM app.fatonfe WHERE infNFe_ide_dhEmi = '2021-12-09
    00:00:00.000' AND infNFe_emit_CNPJ LIKE '0000000000%';
-- Q07
SELECT * FROM app.fatonfe WHERE infNFe_ide_dhEmi = '2021-12-09
    00:00:00.000' AND infNFe_emit_CNPJ IN ('00000000000013', '
    00000000000012', '00000000000011', '00000000000010');
-- Q08
SELECT * FROM app.fatonfe WHERE infNFe_ide_dhEmi BETWEEN '2021-12-08
    23:59:59.000 -0300' AND '2022-01-01 00:00:01.000 -0300';
-- Q09
SELECT * FROM app.fatonfe WHERE infNFe_ide_dhEmi BETWEEN '2021-12-08
    23:59:59.000 -0300' AND '2021-12-31 00:00:00.000 -0300';
-- Q10
SELECT * FROM app.fatonfe WHERE infNFe_emit_CNPJ = '00000000000010' AND
    infNFe_ide_dhEmi BETWEEN '2021-12-08 23:59:59.000 -0300' AND '
    2022-01-01 00:00:01.000 -0300';
-- Q11
SELECT * FROM app.fatonfe WHERE infNFe_emit_CNPJ = '00000000000010' AND
    infNFe_ide_dhEmi BETWEEN '2021-12-08 23:59:59.000 -0300' AND '
    2021-12-31 00:00:00.000 -0300';
-- Q12
SELECT * FROM app.fatonfe WHERE infNFe_emit_CNPJ LIKE '0000000000%' AND
    infNFe_ide_dhEmi BETWEEN '2021-12-08 23:59:59.000 -0300' AND '
    2022-01-01 00:00:01.000 -0300';
-- Q13
SELECT * FROM app.fatonfe WHERE infNFe_emit_CNPJ LIKE '0000000000%' AND
    infNFe_ide_dhEmi BETWEEN '2021-12-08 23:59:59.000 -0300' AND '
    2021-12-31 00:00:00.000 -0300';
-- Q14
SELECT * FROM app.fatonfe WHERE infNFe_emit_CNPJ IN ('00000000000013', '
    00000000000012', '00000000000011', '00000000000010') AND
    infNFe_ide_dhEmi BETWEEN '2021-12-08 23:59:59.000 -0300' AND '
    2022-01-01 00:00:01.000 -0300';
-- Q15
SELECT * FROM app.fatonfe WHERE infNFe_emit_CNPJ IN ('00000000000013', '
    00000000000012', '00000000000011', '00000000000010') AND
    infNFe_ide_dhEmi BETWEEN '2021-12-08 23:59:59.000 -0300' AND '
    2021-12-31 00:00:00.000 -0300';
-- Q16
SELECT id_fatonfe FROM app.fatonfe;
-- Q17
SELECT id_fatonfe FROM app.fatonfe WHERE infNFe_emit_CNPJ = '
```

```
    00000000000010';
35 -- Q18
36 SELECT id_fatonfe FROM app.fatonfe WHERE infNFe_emit_CNPJ LIKE '
    0000000000%';
37 -- Q19
38 SELECT id_fatonfe FROM app.fatonfe WHERE infNFe_emit_CNPJ IN ('
    00000000000013', '00000000000012', '00000000000011', '00000000000010'
    );
39 -- Q20
40 SELECT id_fatonfe FROM app.fatonfe WHERE infNFe_ide_dhEmi = '2021-12-09
    00:00:00.000';
41 -- Q21
42 SELECT id_fatonfe FROM app.fatonfe WHERE infNFe_ide_dhEmi = '2021-12-09
    00:00:00.000' AND infNFe_emit_CNPJ = '00000000000010';
43 -- Q22
44 SELECT id_fatonfe FROM app.fatonfe WHERE infNFe_ide_dhEmi = '2021-12-09
    00:00:00.000' AND infNFe_emit_CNPJ LIKE '0000000000%';
45 -- Q23
46 SELECT id_fatonfe FROM app.fatonfe WHERE infNFe_ide_dhEmi = '2021-12-09
    00:00:00.000' AND infNFe_emit_CNPJ IN ('00000000000013', '
    00000000000012', '00000000000011', '00000000000010');
47 -- Q24
48 SELECT id_fatonfe FROM app.fatonfe WHERE infNFe_ide_dhEmi BETWEEN '
    2021-12-08 23:59:59.000 -0300' AND '2022-01-01 00:00:01.000 -0300';
49 -- Q25
50 SELECT id_fatonfe FROM app.fatonfe WHERE infNFe_ide_dhEmi BETWEEN '
    2021-12-08 23:59:59.000 -0300' AND '2021-12-31 00:00:00.000 -0300';
51 -- Q26
52 SELECT id_fatonfe FROM app.fatonfe WHERE infNFe_emit_CNPJ = '
    00000000000010' AND infNFe_ide_dhEmi BETWEEN '2021-12-08 23:59:59.000
     -0300' AND '2022-01-01 00:00:01.000 -0300';
53 -- Q27
54 SELECT id_fatonfe FROM app.fatonfe WHERE infNFe_emit_CNPJ = '
    00000000000010' AND infNFe_ide_dhEmi BETWEEN '2021-12-08 23:59:59.000
     -0300' AND '2021-12-31 00:00:00.000 -0300';
55 -- Q28
56 SELECT id_fatonfe FROM app.fatonfe WHERE infNFe_emit_CNPJ LIKE '
    0000000000%' AND infNFe_ide_dhEmi BETWEEN '2021-12-08 23:59:59.000
    -0300' AND '2022-01-01 00:00:01.000 -0300';
57 -- Q29
58 SELECT id_fatonfe FROM app.fatonfe WHERE infNFe_emit_CNPJ LIKE '
    0000000000%' AND infNFe_ide_dhEmi BETWEEN '2021-12-08 23:59:59.000
    -0300' AND '2021-12-31 00:00:00.000 -0300';
59 -- Q30
60 SELECT id_fatonfe FROM app.fatonfe WHERE infNFe_emit_CNPJ IN ('
    00000000000013', '00000000000012', '00000000000011', '00000000000010'
    ) AND infNFe_ide_dhEmi BETWEEN '2021-12-08 23:59:59.000 -0300' AND '
```

```
    2022-01-01 00:00:01.000 -0300';
-- Q31
SELECT id_fatonfe FROM app.fatonfe WHERE infNFe_emit_CNPJ IN ('
    00000000000013', '00000000000012', '00000000000011', '00000000000010'
    ) AND infNFe_ide_dhEmi BETWEEN '2021-12-08 23:59:59.000 -0300' AND '
    2021-12-31 00:00:00.000 -0300';
-- Q32
SELECT avg(id_fatonfe) FROM app.fatonfe;
-- Q33
SELECT avg(id_fatonfe) FROM app.fatonfe WHERE infNFe_emit_CNPJ = '
    00000000000010';
-- Q34
SELECT avg(id_fatonfe) FROM app.fatonfe WHERE infNFe_emit_CNPJ LIKE '
    0000000000%';
-- Q35
SELECT avg(id_fatonfe) FROM app.fatonfe WHERE infNFe_emit_CNPJ IN ('
    00000000000013', '00000000000012', '00000000000011', '00000000000010'
    );
-- Q36
SELECT avg(id_fatonfe) FROM app.fatonfe WHERE infNFe_ide_dhEmi = '
    2021-12-09 00:00:00.000';
-- Q37
SELECT avg(id_fatonfe) FROM app.fatonfe WHERE infNFe_ide_dhEmi = '
    2021-12-09 00:00:00.000' AND infNFe_emit_CNPJ = '00000000000010';
-- Q38
SELECT avg(id_fatonfe) FROM app.fatonfe WHERE infNFe_ide_dhEmi = '
    2021-12-09 00:00:00.000' AND infNFe_emit_CNPJ LIKE '0000000000%';
-- Q39
SELECT avg(id_fatonfe) FROM app.fatonfe WHERE infNFe_ide_dhEmi = '
    2021-12-09 00:00:00.000' AND infNFe_emit_CNPJ IN ('00000000000013', '
    00000000000012', '00000000000011', '00000000000010');
-- Q40
SELECT avg(id_fatonfe) FROM app.fatonfe WHERE infNFe_ide_dhEmi BETWEEN '
    2021-12-08 23:59:59.000 -0300' AND '2022-01-01 00:00:01.000 -0300';
-- Q41
SELECT avg(id_fatonfe) FROM app.fatonfe WHERE infNFe_ide_dhEmi BETWEEN '
    2021-12-08 23:59:59.000 -0300' AND '2021-12-31 00:00:00.000 -0300';
-- Q42
SELECT avg(id_fatonfe) FROM app.fatonfe WHERE infNFe_emit_CNPJ = '
    00000000000010' AND infNFe_ide_dhEmi BETWEEN '2021-12-08 23:59:59.000
     -0300' AND '2022-01-01 00:00:01.000 -0300';
-- Q43
SELECT avg(id_fatonfe) FROM app.fatonfe WHERE infNFe_emit_CNPJ = '
    00000000000010' AND infNFe_ide_dhEmi BETWEEN '2021-12-08 23:59:59.000
     -0300' AND '2021-12-31 00:00:00.000 -0300';
-- Q44
SELECT avg(id_fatonfe) FROM app.fatonfe WHERE infNFe_emit_CNPJ LIKE '
```

```
    0000000000%' AND infNFe_ide_dhEmi BETWEEN '2021-12-08 23:59:59.000
    -0300' AND '2022-01-01 00:00:01.000 -0300';
-- Q45
SELECT avg(id_fatonfe) FROM app.fatonfe WHERE infNFe_emit_CNPJ LIKE '
    0000000000%' AND infNFe_ide_dhEmi BETWEEN '2021-12-08 23:59:59.000
    -0300' AND '2021-12-31 00:00:00.000 -0300';
-- Q46
SELECT avg(id_fatonfe) FROM app.fatonfe WHERE infNFe_emit_CNPJ IN ('
    00000000000013', '00000000000012', '00000000000011', '00000000000010'
    ) AND infNFe_ide_dhEmi BETWEEN '2021-12-08 23:59:59.000 -0300' AND '
    2022-01-01 00:00:01.000 -0300';
-- Q47
SELECT avg(id_fatonfe) FROM app.fatonfe WHERE infNFe_emit_CNPJ IN ('
    00000000000013', '00000000000012', '00000000000011', '00000000000010'
    ) AND infNFe_ide_dhEmi BETWEEN '2021-12-08 23:59:59.000 -0300' AND '
    2021-12-31 00:00:00.000 -0300';
-- Q48
SELECT sum(id_fatonfe) FROM app.fatonfe;
-- Q49
SELECT sum(id_fatonfe) FROM app.fatonfe WHERE infNFe_emit_CNPJ = '
    00000000000010';
-- Q50
SELECT sum(id_fatonfe) FROM app.fatonfe WHERE infNFe_emit_CNPJ LIKE '
    0000000000%';
-- Q51
SELECT sum(id_fatonfe) FROM app.fatonfe WHERE infNFe_emit_CNPJ IN ('
    00000000000013', '00000000000012', '00000000000011', '00000000000010'
    );
-- Q52
SELECT sum(id_fatonfe) FROM app.fatonfe WHERE infNFe_ide_dhEmi = '
    2021-12-09 00:00:00.000';
-- Q53
SELECT sum(id_fatonfe) FROM app.fatonfe WHERE infNFe_ide_dhEmi = '
    2021-12-09 00:00:00.000' AND infNFe_emit_CNPJ = '00000000000010';
-- Q54
SELECT sum(id_fatonfe) FROM app.fatonfe WHERE infNFe_ide_dhEmi = '
    2021-12-09 00:00:00.000' AND infNFe_emit_CNPJ LIKE '0000000000%';
-- Q55
SELECT sum(id_fatonfe) FROM app.fatonfe WHERE infNFe_ide_dhEmi = '
    2021-12-09 00:00:00.000' AND infNFe_emit_CNPJ IN ('00000000000013', '
    00000000000012', '00000000000011', '00000000000010');
-- Q56
SELECT sum(id_fatonfe) FROM app.fatonfe WHERE infNFe_ide_dhEmi BETWEEN '
    2021-12-08 23:59:59.000 -0300' AND '2022-01-01 00:00:01.000 -0300';
-- Q57
SELECT sum(id_fatonfe) FROM app.fatonfe WHERE infNFe_ide_dhEmi BETWEEN '
    2021-12-08 23:59:59.000 -0300' AND '2021-12-31 00:00:00.000 -0300';
```

```
115 -- Q58
116 SELECT sum(id_fatonfe) FROM app.fatonfe WHERE infNFe_emit_CNPJ = '
    00000000000010' AND infNFe_ide_dhEmi BETWEEN '2021-12-08 23:59:59.000
     -0300' AND '2022-01-01 00:00:01.000 -0300';
117 -- Q59
118 SELECT sum(id_fatonfe) FROM app.fatonfe WHERE infNFe_emit_CNPJ = '
    00000000000010' AND infNFe_ide_dhEmi BETWEEN '2021-12-08 23:59:59.000
     -0300' AND '2021-12-31 00:00:00.000 -0300';
119 -- Q60
120 SELECT sum(id_fatonfe) FROM app.fatonfe WHERE infNFe_emit_CNPJ LIKE '
    0000000000%' AND infNFe_ide_dhEmi BETWEEN '2021-12-08 23:59:59.000
    -0300' AND '2022-01-01 00:00:01.000 -0300';
121 -- Q61
122 SELECT sum(id_fatonfe) FROM app.fatonfe WHERE infNFe_emit_CNPJ LIKE '
    0000000000%' AND infNFe_ide_dhEmi BETWEEN '2021-12-08 23:59:59.000
    -0300' AND '2021-12-31 00:00:00.000 -0300';
123 -- Q62
124 SELECT sum(id_fatonfe) FROM app.fatonfe WHERE infNFe_emit_CNPJ IN ('
    00000000000013', '00000000000012', '00000000000011', '00000000000010'
    ) AND infNFe_ide_dhEmi BETWEEN '2021-12-08 23:59:59.000 -0300' AND '
    2022-01-01 00:00:01.000 -0300';
125 -- Q63
126 SELECT sum(id_fatonfe) FROM app.fatonfe WHERE infNFe_emit_CNPJ IN ('
    00000000000013', '00000000000012', '00000000000011', '00000000000010'
    ) AND infNFe_ide_dhEmi BETWEEN '2021-12-08 23:59:59.000 -0300' AND '
    2021-12-31 00:00:00.000 -0300';
127 -- Q64
128 SELECT count(id_fatonfe) FROM app.fatonfe;
129 -- Q65
130 SELECT count(id_fatonfe) FROM app.fatonfe WHERE infNFe_emit_CNPJ = '
    00000000000010';
131 -- Q66
132 SELECT count(id_fatonfe) FROM app.fatonfe WHERE infNFe_emit_CNPJ LIKE '
    0000000000%';
133 -- Q67
134 SELECT count(id_fatonfe) FROM app.fatonfe WHERE infNFe_emit_CNPJ IN ('
    00000000000013', '00000000000012', '00000000000011', '00000000000010'
    );
135 -- Q68
136 SELECT count(id_fatonfe) FROM app.fatonfe WHERE infNFe_ide_dhEmi = '
    2021-12-09 00:00:00.000';
137 -- Q69
138 SELECT count(id_fatonfe) FROM app.fatonfe WHERE infNFe_ide_dhEmi = '
    2021-12-09 00:00:00.000' AND infNFe_emit_CNPJ = '00000000000010';
139 -- Q70
140 SELECT count(id_fatonfe) FROM app.fatonfe WHERE infNFe_ide_dhEmi = '
    2021-12-09 00:00:00.000' AND infNFe_emit_CNPJ LIKE '0000000000%';
```

```sql
-- Q71
SELECT count(id_fatonfe) FROM app.fatonfe WHERE infNFe_ide_dhEmi = '2021-12-09 00:00:00.000' AND infNFe_emit_CNPJ IN ('00000000000013', '00000000000012', '00000000000011', '00000000000010');
-- Q72
SELECT count(id_fatonfe) FROM app.fatonfe WHERE infNFe_ide_dhEmi BETWEEN '2021-12-08 23:59:59.000 -0300' AND '2022-01-01 00:00:01.000 -0300';
-- Q73
SELECT count(id_fatonfe) FROM app.fatonfe WHERE infNFe_ide_dhEmi BETWEEN '2021-12-08 23:59:59.000 -0300' AND '2021-12-31 00:00:00.000 -0300';
-- Q74
SELECT count(id_fatonfe) FROM app.fatonfe WHERE infNFe_emit_CNPJ = '00000000000010' AND infNFe_ide_dhEmi BETWEEN '2021-12-08 23:59:59.000 -0300' AND '2022-01-01 00:00:01.000 -0300';
-- Q75
SELECT count(id_fatonfe) FROM app.fatonfe WHERE infNFe_emit_CNPJ = '00000000000010' AND infNFe_ide_dhEmi BETWEEN '2021-12-08 23:59:59.000 -0300' AND '2021-12-31 00:00:00.000 -0300';
-- Q76
SELECT count(id_fatonfe) FROM app.fatonfe WHERE infNFe_emit_CNPJ LIKE '0000000000%' AND infNFe_ide_dhEmi BETWEEN '2021-12-08 23:59:59.000 -0300' AND '2022-01-01 00:00:01.000 -0300';
-- Q77
SELECT count(id_fatonfe) FROM app.fatonfe WHERE infNFe_emit_CNPJ LIKE '0000000000%' AND infNFe_ide_dhEmi BETWEEN '2021-12-08 23:59:59.000 -0300' AND '2021-12-31 00:00:00.000 -0300';
-- Q78
SELECT count(id_fatonfe) FROM app.fatonfe WHERE infNFe_emit_CNPJ IN ('00000000000013', '00000000000012', '00000000000011', '00000000000010') AND infNFe_ide_dhEmi BETWEEN '2021-12-08 23:59:59.000 -0300' AND '2022-01-01 00:00:01.000 -0300';
-- Q79
SELECT count(id_fatonfe) FROM app.fatonfe WHERE infNFe_emit_CNPJ IN ('00000000000013', '00000000000012', '00000000000011', '00000000000010') AND infNFe_ide_dhEmi BETWEEN '2021-12-08 23:59:59.000 -0300' AND '2021-12-31 00:00:00.000 -0300';
```

Tabela 1: Tempos de resposta em milissegundos da última execução para cenários sem índice

Cód. Consulta	Máquina 1X	Máquina 2X	Máquina 3N	Máquina 6N
Q01	86864,794	86398,895	79356,316	76818,897
Q02	88066,597	85620,376	78484,238	50747,843
Q03	88329,453	86245,263	78566,135	45088,952
Q04	86651,512	86137,793	25009,589	1635,188
Q05	87432,522	86398,267	24125,077	1997,31
Q06	85746,436	87000,534	24089,859	1983,911
Q07	87340,372	86483,69	24247,988	1941,961
Q08	86549,783	86489,009	47514,926	4335,57
Q09	85825,911	75769,185	24496,717	2034,288
Q10	87252,959	81309,212	47723,827	2401,921
Q11	88079,15	81757,188	24200,151	1984,022
Q12	87112,201	83492,873	47765,491	2262,206
Q13	87189,39	81170,944	24359,29	1936,813
Q14	86795,328	84957,209	47603,664	3057,505
Q15	86221,831	84383,122	24235,052	1923,25
Q16	88104,994	85389,44	82092,102	52065,686
Q17	85515,329	84668,949	78744,748	44411,438
Q18	100736,126	85116,628	77279,785	43908,553
Q19	86311,977	84954,949	77810,968	42519,962
Q20	86075,46	83894,503	23704,12	1584,283
Q21	88114,294	84115,148	23875,203	1953,458
Q22	87336,724	86049,113	23835,318	2023,961
Q23	86852,488	85550,924	23583,318	2009,021
Q24	86337,943	84493,15	47824,52	1852,477
Q25	84843,847	85071,103	23772,13	1544,715
Q26	86700,108	85804,939	47380,296	2230,457
Q27	85710,552	84126,641	23912,308	1818,173
Q28	87539,588	86298,954	47387,572	2329,137

Continuado na próxima página

Tabela 1: **Tempos de resposta em milissegundos da última execução para cenários sem índice** (Continuado)

Cód. Consulta	Máquina 1X	Máquina 2X	Máquina 3N	Máquina 6N
Q29	85241,644	87359,072	23866,295	2115,381
Q30	85773,243	86269,082	47286,045	2176,926
Q31	86022,796	85896,52	23869,077	1945,022
Q32	85978,449	85095,798	81168,626	45191,416
Q33	86532,461	86495,755	77542,182	40383,697
Q34	86546,534	84837,077	77259,836	42197,41
Q35	84848,758	85392,991	78091,273	38427,914
Q36	86126,112	85956,006	24841,895	1549,775
Q37	85482,705	84730,445	23735,713	1971,505
Q38	86993,272	84079,07	24078,673	1989,839
Q39	84782,118	84202,456	24057,443	1903,291
Q40	85442,108	85250,839	47116,759	1896,118
Q41	85769,703	85802,323	23577,617	1666,619
Q42	84874,416	85822,574	47044,28	2342,145
Q43	85011,319	85551,982	23740,503	1917,549
Q44	85110,221	84134,376	47242,633	2285,851
Q45	86099,048	85261,325	24035,557	2024,348
Q46	86148,031	86440,708	47323,988	2528,613
Q47	86609,518	85454,195	23980,882	1936,012

Tabela 2: **Tempos de resposta em milissegundos da última execução para cenários com índice Hash**

Cód. Consulta	Máquina 1X	Máquina 2X	Máquina 3N	Máquina 6N
Q01	80,421	42,923	173,065	276,321
Q02	86531,877	87954,521	80403,204	75831,312
Q03	61,495	34,649	183,3	276,83
Q04	24,902	17,04	79,125	77,625
Q05	6,329	9,297	49,789	33,836
Q06	5,885	5,55	36,374	40,272
Q07	6,469	5,833	39,539	28,434
Q08	89269,121	85906,213	48110,475	4266,995
Q09	87873,128	77555,284	24518,345	1961,641
Q10	9,574	12,191	66,286	66,908
Q11	12,147	12,242	28,839	24,759
Q12	88070,325	81386,704	48258,385	1993,4
Q13	88010,57	82497,054	23702,149	1974,884
Q14	15,7	13,217	75,036	66,198
Q15	13,035	10,717	34,111	39,393
Q16	95118,978	84409,84	83141,213	47135,423
Q17	10,005	8,728	113,804	154,121
Q18	87483,436	82712,665	78253,378	41804,679
Q19	12,91	12,501	93,79	161,935
Q20	3,783	4,181	36,36	42,781
Q21	3,909	4,419	26,111	24,898
Q22	4,303	5,648	32,827	31,326
Q23	3,31	4,67	37,113	25,986
Q24	88188,981	83482,78	47888,817	1715,002
Q25	87432,07	83522,007	23731,524	1414,907
Q26	9,079	8,004	60,002	62,892
Q27	8,024	11,867	32,968	28,52
Q28	88364,216	83297,411	48062,999	2138,162

Continuado na próxima página

Tabela 2: **Tempos de resposta em milissegundos da última execução para cenários com índice Hash** (Continuado)

Cód. Consulta	Máquina 1X	Máquina 2X	Máquina 3N	Máquina 6N
Q29	90643,533	86209,631	23907,427	1841,506
Q30	12,742	10,928	83,426	59,481
Q31	14,628	11,051	30,509	31,043
Q32	87178,867	84645,729	83011,055	47864,034
Q33	21,061	10,178	78,115	157,824
Q34	89602,713	83555,606	78169,277	42164,528
Q35	17,906	11,013	117,571	167,128
Q36	3,547	4,269	32,18	36,768
Q37	3,394	4,387	29,14	28,39
Q38	4,671	4,434	27,284	30,303
Q39	3,412	5,783	36,218	26,283
Q40	89065,816	86120,694	48201,041	1835,965
Q41	89232,382	85214,367	23677,883	1514,014
Q42	9,849	10,925	54,302	49,693
Q43	9,744	10,567	26,997	27,759
Q44	88917,919	86341,329	48033,619	2127,092
Q45	89279,184	85637,428	24071,406	1859,245
Q46	9,555	10,366	55,924	59,015
Q47	13,802	12,367	37,196	30,397

Tabela 3: **Tempos de resposta em milissegundos da última execução para cenários com índice B-Tree**

Cód. Consulta	Máquina 1X	Máquina 2X	Máquina 3N	Máquina 6N
Q01	32,309	48,488	201,056	284,574
Q02	88103,156	87554,003	79987,899	43744,206
Q03	63,068	40,967	212,553	303,803
Q04	20,273	20,265	78,129	98,574
Q05	5,527	6,441	36,715	32,192
Q06	7,002	5,114	38,898	32,836
Q07	5,578	5,985	36,311	36,18
Q08	216,384	159,579	527,177	492,816
Q09	176,911	160,919	451,114	494,404
Q10	9,375	9,525	68,587	75,085
Q11	7,709	7,527	33,368	36,863
Q12	78,231	82,281	88,263	82,662
Q13	81,815	85,206	67,691	51,086
Q14	7,724	13,043	60,841	58,152
Q15	8,065	8,005	40,684	39,566
Q16	91491,132	87098,49	35020,77	27143,783
Q17	9,532	10,142	89,057	180,254
Q18	85850,692	86121,882	79362,892	44347,575
Q19	10,276	10,267	85,855	170,022
Q20	4,184	4,994	34,071	33,7
Q21	2,888	4,469	34,519	25,324
Q22	3,383	4,903	27,493	31,797
Q23	3,403	3,139	33,096	38,425
Q24	144,648	71,954	109,585	116,601
Q25	90,791	72,558	111,009	99,618
Q26	10,002	6,567	55,482	51,691
Q27	5,949	5,83	34,167	26,659
Q28	149,361	81,711	78,658	79,047

Continuado na próxima página

Tabela 3: **Tempos de resposta em milissegundos da última execução para cenários com índice B-Tree** (Continuado)

Cód. Consulta	Máquina 1X	Máquina 2X	Máquina 3N	Máquina 6N
Q29	116,326	88,682	55,924	52,969
Q30	10,212	5,868	67,543	49,106
Q31	5,086	7,078	33,835	29,321
Q32	86477,81	86873,674	32467,839	23278,619
Q33	11,069	7,963	90,949	183,56
Q34	88089,323	86403,727	78708,728	44032,539
Q35	7,599	10,403	93,008	189,669
Q36	4,408	3,72	28,931	42,601
Q37	4,563	4,2	32,539	28,106
Q38	3,914	4,948	43,309	38,303
Q39	7,532	4,914	28,669	27,738
Q40	89,474	78,818	137,468	113,817
Q41	91,459	70,9	137,234	88,186
Q42	6,679	6,525	59,592	58,551
Q43	5,987	7,767	28,642	27,687
Q44	123,665	90,286	72,655	78,375
Q45	89,408	80,219	60,061	58,457
Q46	5,373	7,523	59,218	60,511
Q47	6,656	6,793	41,365	30,714