Universidade Federal de Pernambuco
Centro de Informática

Graduação em Ciência da Computação

Análise dos Impactos Após Otimização no Desempenho do Banco de Dados da Ferramenta JuMP - CNJ

André Luiz Pereira da Silva (alps2@cin.ufpe.br)

Trabalho de Graduação

Recife - PE
Março/2024

Universidade Federal de Pernambuco
Centro de Informática

André Luiz Pereira da Silva (alps2@cin.ufpe.br)

Análise dos Impactos Após Otimização no Desempenho do Banco de Dados da Ferramenta JuMP - CNJ

Trabalho apresentado ao Programa de Graduação em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco como requisito parcial para obtenção do grau de Bacharel em Ciência da Computação.

Orientador: *Ricardo Massa (rmfl@cin.ufpe.br)*
Co-orientador: *Thiago de Sousa Araújo (tsa2@cin.ufpe.br)*

Recife - PE
Março/2024

Ficha de identificação da obra elaborada pelo autor,
através do programa de geração automática do SIB/UFPE

Silva, André Luiz Pereira da.
Análise dos impactos após otimização no desempenho do banco de dados da ferramenta JuMP - CNJ / André Luiz Pereira da Silva. - Recife, 2024.
39 p. : il., tab.

Orientador(a): Ricardo Massa Ferreira Lima
Cooorientador(a): Thiago de Souza Araújo
Trabalho de Conclusão de Curso (Graduação) - Universidade Federal de Pernambuco, Centro de Informática, Ciências da Computação - Bacharelado, 2024.
Inclui referências.

1. Otimização de Consultas SQL. 2. PostgreSQL. 3. Experimentação. 4. Engenharia de Software. I. Lima, Ricardo Massa Ferreira. (Orientação). II. Araújo, Thiago de Souza. (Coorientação). IV. Título.

000 CDD (22.ed.)

André Luiz Pereira da Silva

Análise dos Impactos Após Otimização no Desempenho do Banco de Dados da Ferramenta JuMP - CNJ

Trabalho de Conclusão de Curso apresentado à Coordenação do Curso de Bacharelado em Ciência da Computação, da Universidade Federal de Pernambuco, como parte dos requisitos à obtenção do grau de Bacharel em Ciência da Computação.

Data de Aprovação: 18/03/2024

Nota: _____

BANCA EXAMINADORA:

_____
Prof. Ricardo Massa Ferreira Lima (Orientador)
Centro de Informática - UFPE

_____
Prof. Márcio Lopes Cornélio (1º Titular)
Centro de Informática - UFPE

RECIFE
2024

Agradecimentos

Inicio os agradecimentos com uma menção ao orientador deste trabalho, o professor Ricardo Massa, assim como o co-orientador, Thiago Araújo, que guiaram-me durante o processo de produção científica e forneceram-me as ferramentas necessárias para produzir um trabalho de valor. Espero que tenha conseguido atender às suas expectativas.

Aos colegas de trabalho que tive ao decorrer do meu tempo no JuMP - CNJ, agradeço por acolherem-me mesmo com pouca experiência e permitirem-me amadurecer profissionalmente. Excepcionalmente, menciono Danilo Carmo, o Tech Lead da equipe, como alguém fora da curva em todos os sentidos positivos. Muito do que eu sou capaz de fazer hoje, tecnicamente falando, é por conta da sua mentoria.

Aos colegas de trabalho atuais da Nina, agradeço pela compreensão e paciência nos momentos onde precisei priorizar a conclusão deste trabalho.

Aos colegas do Maracatronics, agradeço por terem me deixado experimentar como é fazer parte de um time de robótica, um sonho de criança. Mesmo que nossos caminhos tenham divergido, guardo com carinho as memórias forjadas em meio às adversidades.

Ao Google e ao Stack Overflow, agradeço por tornarem a minha profissão mais fácil!

Ao meu psicólogo, Silvio, agradeço por me munir com as ferramentas necessárias para entender e lidar com os meus sentimentos e abrir meus olhos para acreditar em possibilidades num momento de profunda desesperança.

Ao professor especialista e amigo, Caio Dias, agradeço pelo tempo dedicado à sugerir melhorias na escrita do trabalho.

Aos amigos e colegas do 90 Graus, aos quais tive o prazer de reconectar recentemente, obrigado por terem aturado aquele eu do passado, que naqueles tempos ainda havia de aprender a ser gente.

Aos amigos do Santa Emília que ainda estão presentes na minha vida, agradeço pelo convívio nos tempos de escola e por terem continuado a suportar as minhas ladainhas mesmo quando não éramos mais obrigados a dividir o mesmo espaço diariamente. Com vocês, eu tive a oportunidade de amadurecer, enxergar a vida por outros ângulos e expandir meus horizontes.

Aos amigos da UFPE, obrigado por apoiarem-me nos árduos anos de estudo que passamos juntos. Aos que graduaram-se antes de mim: Finalmente os alcancei! Aos que ainda estão na estrada: Perseverem. Existe glória no fim dessa luta.

Aos diversos amigos que mantenho contato pela internet, de todos os lugares do Brasil, sou grato pelas madrugadas de descontração e por sempre acolherem-me quando eu aparecia para falar exclusivamente dos contratempos da vida, incluindo as dificuldades de realizar este trabalho.

Aos meus melhores amigos, da TDS, agradeço pelos laços profundos e boas memórias

que pudemos fazer, mesmo com o empecilho da distância. Saibam que mesmo que não nos vejamos com a mesma frequência dos tempos de juventude despreocupada, ainda penso em cada um como uma parte integral do meu ser e acredito que o nosso grupo é mais do que a soma das partes.

Dos meus grandes amigos, gostaria de destacar um: Lucas Araújo. No livro "Frankenstein", de Mary Shelley, o cientista que dá nome ao livro por vezes toca no assunto da necessidade de se ter um Amigo, alguém que nos acorrente à realidade e à sanidade nos momentos que estamos perigosamente perto de deixá-las para trás. Alguém que verdadeiramente nos entenda e a quem possamos compartilhar os pensamentos mais sombrios sem medo de julgamentos. Essas passagens fixaram-se em minha memória pois não conseguia parar de traçar paralelos com a nossa amizade. Para mim, você é esse Amigo. Tenho medo de imaginar a pessoa que seria hoje se não houvesse sido influenciado por você.

Finalmente, agradeço à toda a minha família: Gilson, minha Titia Déia, Melinha, falecidos avós, primos e primas distantes e todos os outros que assim como esses se fizeram presentes durante a minha trajetória, mesmo que sem nenhuma ligação genética.

O agradecimento mais especial de todos dedico à minha mãe, Ângela Lima, carinhosamente apelidada de Si. Estou em débito eterno com você por ter plantado as sementes da curiosidade que acabaram por germinar nesta mente inquisitiva que possuo hoje e por conceder-me o raro privilégio de manter um foco integral nos estudos por tantos anos.

O caminho do conhecimento é árduo e frustrante, mas ao fim de mais uma etapa formativa do meu saber, percebo que não me contentaria com nenhuma outra forma de ser, que não esta.

“A educação tem raízes amargas, mas os seus frutos são doces.”
—ARISTÓTELES

Resumo

Este trabalho introduz os elementos pertinentes à Mineração de Processos, contextualizando-os para a esfera jurídica e apresenta como a ferramenta JuMP - CNJ lida com o armazenamento dos dados de processos e movimentos jurídicos que abastecem as funcionalidades da aplicação, assim como os desafios enfrentados pela equipe de desenvolvimento para gerir um volume sempre crescente de informação. A partir dos requisitos funcionais de duas das funcionalidades mais relevantes da plataforma, coletados junto à equipe, são implementadas diversas melhorias às consultas SQL utilizadas, norteadas por guias de otimização e com foco na redução do tempo para obtenção dos resultados. Os impactos dessas mudanças são avaliados por meio de uma metodologia de experimentação própria para engenharia de software e os resultados são analisados estatisticamente. Embora os resultados positivos possam servir de precedente para mais otimizações no JuMP - CNJ, não é possível ignorar o *trade-off* entre tempo de execução e custo de armazenamento no contexto de bancos de dados, de forma que caso deseje-se extrapolar as medidas aqui aplicadas em outros contextos, um estudo dos requisitos e características da aplicação serão essenciais para evitar que os custos acabem sendo maiores que os ganhos.

Palavras-chave: Otimização de Consultas SQL, PostgreSQL, Experimentação, Engenharia de Software

Abstract

This work introduces Process Mining's pertinent elements, contextualizing them to the legal sphere and presents how the tool JuMP - CNJ deals with the storage of data from legal processes and movements that serve as the fuel to the application functionalities, as well as the challenges faced by the development team in managing an ever-increasing volume of information. From the functional requirements of two of the most relevant functionalities of the platform, collected among the team, many improvements were implemented to the SQL queries used by them, led by optimization guides and focused in reducing the time needed to receive the results. The impacts from these changes are evaluated by a proper software engineering experimentation methodology and the results are statistically analyzed. Even though the positive results can lead the way to more optimizations at JuMP - CNJ, it's impossible to ignore the trade-off between execution time and storage cost in the context of databases, so that in the case that one wishes to extrapolate the measures applied here in other contexts, studying the requirements and specifics of the application will prove essential in order to avoid costs that outweight the gains.

Keywords: SQL Query Optimization, PostgreSQL, Experimentation, Software Engineering

Sumário

Lista de Figuras

Lista de Tabelas

CAPÍTULO 1

Considerações Iniciais

1.1 Bancos de Dados e Consultas SQL

De forma genérica, o nome *Base de Dados* é utilizado para referenciar qualquer coleção de dados relacionados que representa algum aspecto do mundo real e agrega dados logicamente coerentes e com algum significado inerente. Bases de dados são desenhadas, construídas e populadas com dados para propósitos específicos, do interesse de algum grupo de usuários ou aplicações [13].

Mais especificamente, *Bancos de Dados* geralmente descrevem Bases de Dados estruturadas e organizadas, acopladas a um *Sistema Gerenciador de Bancos de Dados* (SGBD), softwares projetados para facilitar sua criação, manipulação e administração, fornecendo uma interface para os usuários e garantindo ferramentas de manipulação de dados eficientes [13].

Existem tipos diferentes de SGBD's, porém esta investigação tratará de um Sistema de Informação específico, detalhado mais à frente, que utiliza um SGBD com modelo relacional. Ele é baseado em relações, tabelas bidimensionais compostas por linhas e colunas. Cada tabela contém registros (linhas) que representam entidades e atributos (colunas) que representam características dessas entidades. Elas podem ser relacionadas à outras tabelas através de chaves estrangeiras, que as ligam com base em um campo comum. Isso permite a criação de consultas complexas e a integridade referencial entre os dados.

Para interagir com Bancos de Dados relacionais, é possível utilizar a *Structured Query Language* (SQL) para encadear instruções e recuperar informações específicas. Exemplos incluem consultas simples como "selecionar todos os dados de uma tabela", ou mais complexas que envolvem junções, filtragens e ordenações.

Esses conceitos formam a base para o design, a consulta e a manipulação eficientes de bancos de dados, sendo essenciais para o funcionamento de sistemas de informação modernos.

1.2 Mineração de Processos

A área da Mineração de Processos surge como um conjunto de ferramentas valiosas para proporcionar insights fundamentados por dados e impulsionar melhorias nos processos executados por instituições e organizações. Essa área amplia as abordagens voltadas para modelos de processos e mineração de dados. As técnicas e ferramentas dentro desse campo têm a capacidade de descobrir e monitorar os processos reais, frequentemente divergentes dos modelos de processos esperados. O conhecimento adquirido durante essas etapas é crucial para identificar gargalos operacionais e inconsistências nas estruturas descobertas [17].

Para a aplicação efetiva dessa tecnologia, é imperativo que os dados estejam organizados no formato de "Log de Eventos". Embora os dados de eventos sejam comuns nos sistemas de informação modernos, muitas vezes carecem de uma estrutura bem definida e universal. A seguir, são destacadas características essenciais que os "Logs de Eventos"precisam obedecer [17]:

- Identificador de caso (*case id*): Distingue e identifica os casos.
- Identificador de evento (*event id*): Diferencia os eventos pertencentes a um caso.
- Atributos do evento: Incluem diversos atributos, como *timestamps* que indicam data e hora do ocorrido, os recursos ou funcionários responsáveis, custos operacionais, entre outros.
- Ordenação dos eventos: Os eventos precisam ser ordenados, geralmente seguindo a ordem de ocorrência.
- Instância de evento pertence a apenas um caso: Garante que uma instância de evento seja associada exclusivamente a um caso.

Esse formato facilita a interpretação dos dados de maneira orientada a eventos. Os "casos"relacionam eventos para formar uma instância de processo quando agrupados, enquanto as "classes de evento"diferenciam as diversas atividades que podem ocorrer dentro dessas instâncias de processo [18].

1.3 Mineração de Processos no Contexto Judiciário

O artigo "Process Mining-Enabled Jurimetrics: Analysis of a Brazilian Court's Judicial Performance in Business Law Processing"argumenta a necessidade de utilizar ferramentas digitais para promover mudanças positivas no contexto jurídico. Conclui que as inovações das técnicas de mineração de processos provam-se poderosas aliadas, capazes de revelar conjuntos de métricas de diagnóstico, insights sobre as raízes das ineficiências e ideias para melhorias que dificilmente seriam descobertas sem uma abordagem orientada aos processos [16].

A mineração de processos pode ser vista como uma ferramenta analítica naturalmente adequada para o contexto jurídico, devido à natureza procedural inerente dos processos jurídicos. Na Tabela 1.1, observa-se como os conceitos de mineração de processos traduzem para o cotidiano das unidades jurídicas [16].

1.4 JuMP - CNJ

O JuMP - CNJ é um projeto de inovação desenvolvido pelo V-Lab, laboratório vinculado à Universidade Federal de Pernambuco, em parceria com o Conselho Nacional de Justiça (CNJ). O CNJ é responsável por aprimorar o trabalho realizado pelo setor judiciário brasileiro [2].

Mineração de Processos	Contexto Judiciário.
Atividade	Nome ou identificador de um movimento procedural que gera progresso em um processo jurídico. Exemplos incluem atividades desempenhadas por magistrados ou servidores, como decisões, despachos e julgamentos.
Evento	Ocorrência de um movimento procedural.
Timestamp	Data de ocorrência do movimento procedural.
Caso	Processo jurídico.
Identificador do caso	Identificador do processo jurídico.
Atributos do evento	Atributos gerais relacionados ao movimento procedural.
Atributos do caso	Atributos gerais relacionados ao processo jurídico.

Tabela 1.1 Mineração de Processos no Contexto Jurídico

O principal objetivo do projeto é aplicar técnicas de Mineração de Processos aos dados transacionais que regem os processos jurídicos nos tribunais brasileiros.

O JuMP utiliza dados do *data lake* Codex, propriedade do CNJ. Esse sistema consolida as bases de dados processuais dos tribunais brasileiros e disponibiliza essas informações para várias aplicações por meio de uma API [3].

Analisar efetivamente esses processos judiciais representa um desafio significativo devido ao grande volume de dados. O JuMP já conta com mais de 23 milhões de processos internamente carregados, representando menos de 20% do total no Codex, que tende a crescer diariamente, como mostra a Figura 1.1.

Para armazenar o montante de informações provenientes do Codex, o JuMP utiliza uma instância remota de banco de dados PostgreSQL fornecido pelo setor de infraestrutura do CNJ. O modelo lógico, que define as tabelas, suas características e relacionamentos, evoluiu organicamente com a aplicação, passando por revisões constantes para garantir conformidade com as regras de normalização [10]. No entanto, a aplicação sofre com os impactos negativos no desempenho originados pelo excesso de normalização, especialmente ao executar operações de *Join* entre várias tabelas [14].

Dada a maturidade da arquitetura do JuMP, é relevante avaliar a camada de gerenciamento de dados em busca de pontos de melhoria, especialmente ao lidar com grandes volumes de dados sobre unidades judiciais inteiras.

Figura 1.1 *Snapshot* das estatísticas de carregamento do JuMP em 05/06/2023.

1.4.1 Modelo Lógico do Banco de Dados

A aplicação organiza dados sobre os processos jurídicos de cada unidade judicial em duas tabelas distintas, exclusivas para cada unidade. Seguindo a convenção de nomenclatura ”processos_xxxx” e ”movimentos_xxxx”, onde ”xxxx” é substituído pelo número identificador da unidade. A Figura 1.2 oferece uma representação visual dos segmentos de modelo lógico relevantes para este trabalho.

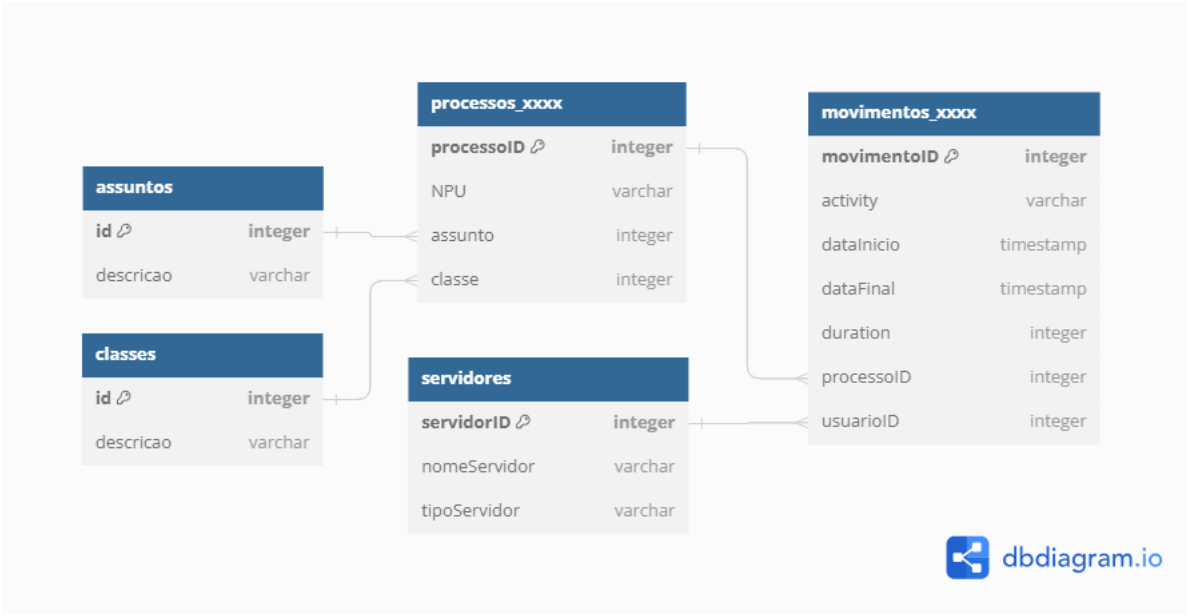

Figura 1.2 Modelo lógico parcial do banco de dados da ferramenta JuMP - CNJ.

- Tabela de processos: cada linha representa um processo da unidade judicial, sendo que, na linguagem de mineração de processos, um processo jurídico equivale a um *caso*.

- Tabela de movimentos: cada linha representa um movimento procedural que impulsionou o progresso em um dos processos da unidade judicial. Cada um está vinculado a apenas um processo. No contexto jurídico, exemplos de movimentações incluem atividades realizadas por magistrados ou servidores, como *decisões*, *despachos* e *julgamentos*. Na linguagem de mineração de processos, representam uma *classe de evento* ou *atividade*.

1.4.2 Fluxograma

Fluxogramas são representações visuais derivadas dos *Logs de Eventos* que auxiliam na visualização dos processos descobertos e suas características. Uma captura de tela dessa ferramenta pode ser vista na Figura 1.3. O JuMP - CNJ oferece a produção e manipulação desses fluxogramas como uma de suas principais funcionalidades. Requisitos para os dados utilizados em sua construção incluem todos os movimentos desempenhados em uma unidade jurídica, contendo:

- Nome do movimento.
- Data/hora de início do movimento.
- Data/hora de término do movimento.
- Duração do movimento.
- Identificador único do usuário responsável por realizar o movimento.
- Nome do usuário responsável por realizar o movimento.
- Tipo do usuário responsável por realizar o movimento.
- Identificador único do processo no qual o movimento foi realizado.
- Identificador único, junto ao CNJ, do processo no qual o movimento foi realizado (NPU).
- Classe do processo no qual o movimento foi realizada.
- Assunto do processo no qual o movimento foi realizada.
- Filtro de processos para recuperar apenas aqueles iniciados após uma certa data.
- Ordenação dos dados por identificador único do processo e data/hora de término dos movimentos.

A consulta utilizada para recuperar esses dados é apresentada no Código 1.1. A filtragem na data ”01/01/1900” está fixa aqui para forçar um pior caso, mas, na prática, a data é inserida na consulta de maneira programática. A definição de "Pior caso", no contexto de uma filtragem, significa uma filtragem com alto grau de seletividade, que não retém muitos registros da relação original [7]. Uma data como ”01/01/1900” garante que após essa filtragem, 100% dos registros de uma relação passem para as próximas etapas da consulta.

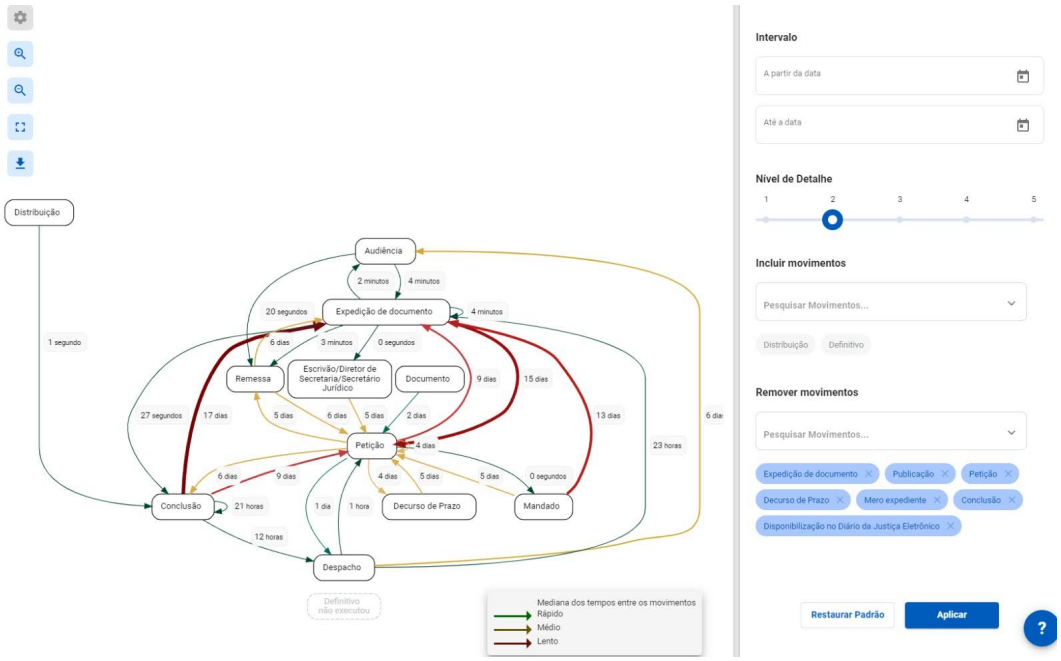

Figura 1.3 Captura de tela da funcionalidade Fluxograma do JuMP - CNJ.

Código 1.1 Consulta Utilizada na Função Fluxograma.

```
SELECT
proc."NPU",
proc."processoID",
subquery.earliest_date,
c."descricao" AS "classe",
a."descricao" AS "assunto",
mov."movimentoID"
mov."activity",
mov."dataInicio",
mov."dataFinal",
mov."usuarioID",
mov."duration",
COALESCE(s."nomeServidor", 'N/I') AS "nomeServidor",
COALESCE(s."tipoServidor", 'N/I') AS "tipoServidor"
FROM processos_xxxx proc
JOIN classes c ON proc.classe = c.id
JOIN assuntos a ON proc.assunto = a.id
JOIN movimentos_xxxx mov
ON proc."processoID" = mov."processoID"
LEFT OUTER JOIN servidores s
ON mov."usuarioID" = s."servidorID"
JOIN (
```

```
    SELECT
    mov."processoID" AS "processoID",
    min(mov."dataInicio") AS earliest_date
    FROM movimentos_xxxx mov
    GROUP BY mov."processoID"
) AS subquery ON proc."processoID" = anon_1."processoID"
WHERE subquery.earliest_date >= '1900-01-01'
ORDER BY proc."processoID", mov."dataFinal" ASC;
```

1.4.3 Corregedoria

Essa funcionalidade permite a comparação simultânea de múltiplas unidades jurídicas para análise de métricas de desempenho definidas pelo CNJ em uma faixa de tempo específica. Uma captura de tela dessa ferramenta pode ser vista na Figura 1.4. Embora possuam requisitos distintos, a mesma consulta utilizada para gerar o fluxograma é empregada aqui com múltiplas consultas encadeadas em sequência, uma para cada unidade jurídica selecionada pelos usuários para comparação. Os requisitos para cada movimento retornado são:

- Nome do movimento.
- Data/hora de início do movimento.
- Data/hora de término do movimento.
- Duração do movimento.
- Identificador único do processo no qual o movimento foi realizado.
- Identificador único, junto ao CNJ, do processo no qual o movimento foi realizado (NPU).
- Classe do processo no qual o movimento foi realizado.
- Filtro de processos para recuperar apenas aqueles iniciados após uma certa data.
- Ordenação dos dados por identificador único do processo e data/hora de término dos movimentos.

1.5 Otimização de Consultas SQL

A otimização é uma etapa essencial no processo de desenvolvimento de software, apesar da abordagem comum de "fazer funcionar primeiro e otimizar depois"por alguns desenvolvedores [7].

Para ser efetivamente utilizável, os resultados das consultas SQL precisam estar disponíveis em um prazo razoável para o consumidor desses dados, seja ele humano, máquina ou outro

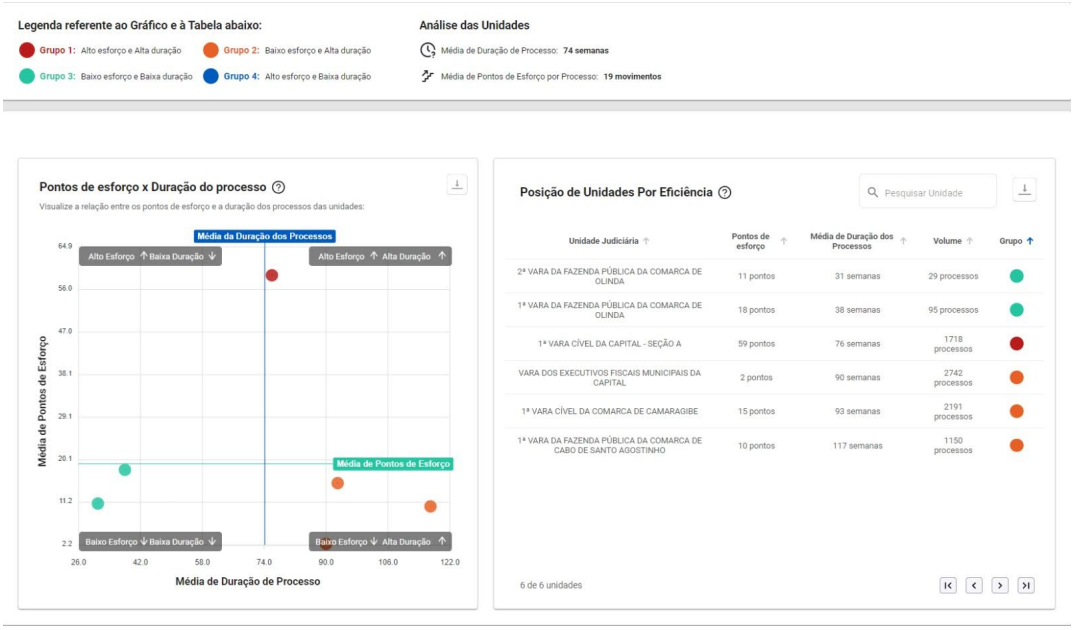

Figura 1.4 Captura de tela da funcionalidade Corregedoria do JuMP - CNJ.

sistema de gerenciamento de bancos de dados [9]. Durante a construção dessas consultas, é possível que ineficiências evitáveis sejam introduzidas aos sistemas, oriundas de uma falta de atenção às particularidades de suas execuções [7]. Além disso, considerando que o tempo de execução das consultas SQL muitas vezes compõe os requisitos não funcionais de um sistema, a otimização torna-se crucial para atender às expectativas por sistemas "mais rápidos, baratos, amigáveis e melhores", expressas por usuários e engenheiros de software [1].

A capacidade do Sistema de Informação, como o JuMP - CNJ, de cumprir seu papel e proporcionar uma experiência agradável aos usuários está diretamente relacionada ao desempenho e agilidade de suas funções. Nesses pontos, a otimização das consultas SQL adota um papel fundamental.

Para realizar as otimizações, serão analisados os requisitos apresentados, gerando consultas personalizadas para cada um. As mudanças também podem ser propostas no modelo lógico das relações presentes no banco de dados. O objetivo será a redução do tempo de execução e serão seguidos guias da literatura que fornecem instruções e orientações para alcançá-lo [7]. Este estudo não visa apresentar ou discutir esses guias, mas sim utilizá-los para gerar novas versões dos artefatos apresentados, medindo o impacto nos tempos de execução e custos de armazenamento. As mudanças aplicadas serão listadas na seção de experimentação como base dos tratamentos.

1.6 Objetivos do Estudo

Essa investigação possui como objetivo geral: medir o impacto de guias de otimização no tempo de recuperação de dados e custo de armazenamento da ferramenta JuMP - CNJ. Para

alcançá-lo, são declarados os seguintes objetivos específicos:

- Investigar e definir os objetivos das consultas mais volumosas da ferramenta JuMP - CNJ.
- Propor e aplicar mudanças nas consultas SQL, baseadas em técnicas comprovadas de afinação, visando uma otimização do custo temporal.
- Comparar o tempo de execução e custo em memória dos artefatos originais e otimizados.

CAPÍTULO 2

Metodologia

A metodologia escolhida para este estudo é a de experimentação, com o seguinte escopo: analisar consultas SQL com o propósito de determinar os impactos de seguir guias de otimização em relação aos tempos de execução e custo espacial, da perspectiva dos desenvolvedores da ferramenta de mineração de processos JuMP - CNJ. Para a execução correta de um experimento, é necessário definir antecipadamente alguns elementos [19]:

- Variáveis independentes: todas as variáveis que podem ser manipuladas e afetam um processo.
- Variáveis dependentes: variáveis medidas para estudar o impacto das mudanças realizadas sob as variáveis independentes.
- Objetos: artefatos sobre os quais os tratamentos são aplicados.
- Fatores: sub-conjunto das variáveis independentes que foram escolhidas para variação durante a experimentação.
- Tratamentos: instâncias ou valores particulares que podem ser adotados pelos fatores.

2.1 Design de Experimento

O experimento será conduzido por meio da aplicação de dois tratamentos em um único fator. Como o objetivo é medir o impacto das novas consultas, o design escolhido é o de *Comparação Pareada*, conforme definido no livro "Experimentation in software engineering" [19]. Para analisar os dados de forma estatística, será utilizado o *Teste U de Mann-Whitney*, com significância estatística de 0.05. Se a diferença estatística entre os dados gerados após os dois tratamentos for menor que a significância estatística, a hipótese nula não poderá ser rejeitada. Um gráfico de densidade e um de caixa serão construídos com os valores coletados durante a experimentação para atestar se a tendência das mudanças é negativa ou positiva em relação ao tempo de execução das consultas. Para comparar os impactos no custo de armazenamento, será usado um gráfico de barras.

Variáveis independentes:

- *Hardware* onde os dados são armazenados.
- *Hardware* onde os comandos SQL são processados.

- Comandos SQL quem compõe as consultas.
- Atributos das relações (índices, colunas).
- Quantidade de registros nas relações.
- Sistema Gerenciador de Banco de Dados.

Variáveis dependentes:

- Tempo de execução das consultas SQL.
- Custo de memória das relações.

Objetos deste experimento:

- Consulta SQL da funcionalidade de Fluxograma.
- Consulta SQL da funcionalidade de Corregedoria.
- Relações do banco de dados, geralmente representadas por tabelas, estruturas organizadas para armazenar dados de maneira tabular, com linhas e colunas, onde cada linha em uma tabela representa uma entrada ou registro único, enquanto as colunas representam atributos específicos desses registros.

Fatores deste experimento:

- Comandos SQL (Structured Query Language) que compõem as consultas. São instruções utilizadas para interagir com bancos de dados relacionais. Essas instruções permitem a criação, manipulação e consulta de dados armazenados em um banco de dados.
- Atributos das relações, como colunas, os atributos individuais ou campos em uma tabela que armazenam dados específicos, como nomes, datas e números, e índices, estruturas que melhoram a velocidade de recuperação de dados. Eles são criados em uma ou mais colunas e agilizam as operações de busca [15].
- Quantidade de registros nas relações. Refere-se ao número de entradas ou linhas em uma tabela, que representam uma instância única de dados. O total de registros de uma tabela pode variar pois novos dados podem ser adicionados (por meio dos comandos SQL INSERT) e removidos (por meio dos comandos SQL DELETE).

Tratamentos deste experimento:

- Consultas e relações originais (nenhum tratamento).
- Consultas e relações após a afinação.

2.2 Planejamento da Experimentação

2.2.1 Seleção de Contexto

Os testes serão conduzidos no ambiente de *staging* da aplicação, que tem um alto grau de representatividade, pois foi gerado de modo a refletir o estado e dados presentes no ambiente de produção.

As consultas serão realizadas através do PGAdmin, um administrador de bancos de dados. Ele conta com ferramentas nativas para execução e visualização dos resultados das consultas [11].

2.2.2 Formulação de Hipóteses

$$h_0 : \mu_{j,o} = \mu_{j,opt}\, j \in \{Fluxograma, Corregedoria\}$$

$$h_1 : \mu_{j,o} > \mu_{j,opt}\, j \in \{Fluxograma, Corregedoria\}$$

$\mu_{Fluxograma,o}$: Tempo de execução médio nas consultas de Fluxograma originais.
$\mu_{Fluxograma,opt}$: Tempo de execução médio nas consultas de Fluxograma otimizadas.
$\mu_{Corregedoria,o}$: Tempo de execução médio nas consultas de corregedoria originais.
$\mu_{Corregedoria,opt}$: Tempo de execução médio nas consultas de corregedoria otimizadas.

2.2.3 Seleção de Amostras

As consultas serão executadas sobre diferentes unidades judiciais, selecionadas a cada decil em relação à quantidade de processos armazenados no banco, abrangendo exemplos com volumes variados. Essa abordagem possibilita maior generalização dos resultados coletados ao medir o impacto em diferentes concentrações de dados. Nas Tabelas 2.1 e 2.2, estão listadas as unidades selecionadas e o número de processos presentes em cada uma.

Além disso, antes do início da experimentação, os dados foram replicados para tabelas dedicadas ao experimento, evitando perturbações na ferramenta ou no experimento. As tabelas de experimentação possuem todos os dados e características das tabelas originais.

2.3 Instrumentação

Para evitar a influência de perturbações pontuais, as consultas serão executadas e seus resultados registrados 100 vezes consecutivas, permitindo que os resultados possam ser generalizados com maior confiabilidade.

Para realizar a coleta de dados desse experimento, foram desenvolvidas algumas funções e tabelas para disparar as consultas múltiplas vezes, calcular e armazenar os objetos de observação da experimentação, de forma automática [12].

Para medir o tempo médio de execução das consultas, será utilizada a função presente no Código 2.1, capaz de executar uma consulta passada como parâmetro um número arbitrário

Identificador	Unidade	Tribunal	Quantidade de processos registrados
44508	PRESIDÊNCIA	Tribunal Regional Federal da 4ª Região	375.052
8380	JUIZADO ESPECIAL CÍVEL DE TRÂNSITO DE ARACAJU	Tribunal de Justiça do Estado de Sergipe	43.615
18104	NOVA IGUACU-MESQUITA IV JUIZADO ESPECIAL CIVEL	Tribunal de Justiça do Estado do Rio de Janeiro	29.533
17384	10º JUIZADO ESPECIAL CÍVEL E DAS RELAÇÕES DE CONSUMO DA CAPITAL	Tribunal de Justiça do Estado de Pernambuco	23.583
14013	6ª VARA FEDERAL DE CAMPO GRANDE	Tribunal Regional Federal da 3ª Região	20.340
3843	VARA CÍVEL - FORO REGIONAL PARTENON - PORTO ALEGRE/RS	Tribunal de Justiça do Estado do Rio Grande do Sul	18.128

Tabela 2.1 Unidades selecionadas para Experimentação

Identificador	Unidade	Tribunal	Quantidade de processos registrados
7433	TANGARÁ DA SERRA - VARA ESPECIALIZADA DOS JUIZADOS ESPECIAIS	Tribunal de Justiça do Estado de Mato Grosso	16.769
14009	5ª VARA FEDERAL DE SÃO JOSÉ DO RIO PRETO	Tribunal Regional Federal da 3ª Região	15.199
18228	2ª VARA CÍVEL E CRIMINAL DE SIMÃO DIAS	Tribunal de Justiça do Estado de Sergipe	14.086
13325	4ª VARA CÍVEL DE CACOAL	Tribunal de Justiça do Estado de Rondônia	12.789
18007	1ª VARA CÍVEL DA CAPITAL - SEÇÃO B	Tribunal de Justiça do Estado de Pernambuco	3.716

Tabela 2.2 Unidades selecionadas para Experimentação

de vezes, capturar o tempo que cada uma tomou e armazená-los numa tabela específica para armazenar os resultados dos testes, criada por meio do comando presente no Código 2.2.

Código 2.1 Função para Cálculo e Armazenamento do Tempo de Execução.

```
CREATE OR REPLACE FUNCTION record_execution_time(
    num_executions INT,
    unidade INT,
    modulo TEXT,
    comment TEXT,
    query_text TEXT
) RETURNS VOID AS
$$
DECLARE
    l_start_time TIMESTAMP;
    l_end_time TIMESTAMP;
    l_execution_time INTERVAL;
BEGIN
    FOR counter IN 1..num_executions LOOP
        -- Start the timer
        l_start_time := clock_timestamp();

        -- Execute the query
        EXECUTE query_text;

        -- Stop the timer
        l_end_time := clock_timestamp();

        -- Calculate the execution time
        l_execution_time := l_end_time - l_start_time;

        -- Insert the execution time into the table
        INSERT INTO query_execution_times (
                unidade, modulo, comment,
                query_execution_time)
        VALUES (unidade, modulo, comment, l_execution_time);
    END LOOP;
END;
$$
LANGUAGE plpgsql;
```

Código 2.2 Função para Criar Tabela que Armazena Tempos de Execução.

```
CREATE TABLE query_execution_times (
    unidade INT,
    modulo TEXT,
```

```
    comment TEXT,
    query_execution_time INTERVAL
);
```

Uma opção que foi brevemente considerada, porém descartada, é a utilização da configuração *log_duration* para registrar os tempos de execução de todas as consultas realizadas com sucesso no banco. Esse é um comportamento do PostgreSQL que precisa ser conscientemente ativado, caso o administrador do banco assim queira. Infelizmente, as credenciais concedidas à equipe do JuMP, pelo CNJ, não possuem privilégios suficientes para alterar seu estado [5].

Para armazenar o custo espacial das tabelas envolvidas, é possível utilizar o comando *PG_RELATION_SIZE*, que retorna a quantidade de memória ocupada por uma entidade do banco de dados, em bytes, junto ao comando *PG_SIZE_PRETTY*, que formata o resultado da função anterior para MB, GB ou TB quando apropriado [4]. Para auxiliar, será utilizada a função presente no Código 2.3, que recebe como parâmetro o número identificador de uma unidade judiciária, calcula o custo total para armazenar os dados da unidade e registra os valores resultantes numa tabela específica para esse propósito, criada por meio do comando presente no Código 2.4,

Código 2.3 Função para Cálculo e Armazenamento do Custo de Memória.

```
CREATE OR REPLACE FUNCTION record_storage_size(
    unidade INT,
    comment VARCHAR
)
RETURNS VOID AS
$$
DECLARE
    unidade_size NUMERIC;
    unidade_size_in_mb TEXT;
BEGIN
    -- Calculate size in bytes
        IF (comment = 'optimized') THEN
            SELECT (
                PG_RELATION_SIZE(
                    'movimentos_' || unidade
                ) +
                PG_RELATION_SIZE(
                    'processos_' || unidade
                ) +
                PG_RELATION_SIZE(
                    'mat_view_fluxograma_' || unidade
                ) +
                PG_RELATION_SIZE(
                'mat_view_corregedoria_' || unidade))
            INTO unidade_size;
```

```
        ELSE
            SELECT (
                PG_RELATION_SIZE(
                    'movimentos_' || unidade
                ) +
                PG_RELATION_SIZE(
                    'processos_' || unidade))
            INTO unidade_size;
        END IF;

    -- Size in MegaBytes
    SELECT
    PG_SIZE_PRETTY(unidade_size)
    INTO unidade_size_in_mb;

    INSERT INTO unidade_storage_size (
            unidade,
            comment,
            total_size,
            total_size_in_mb)
    VALUES (
            unidade,
            comment,
            unidade_size,
            unidade_size_in_mb);
END;
$$
LANGUAGE plpgsql;
```

Código 2.4 Função para Criar Tabela que Armazena Custos de Memória.

```
CREATE TABLE unidade_storage_size(
    unidade INT,
    comment VARCHAR,
    total_size NUMERIC,
    total_size_in_mb TEXT
);
```

CAPÍTULO 3

Experimentos

Em relação à otimização das consultas SQL, é fundamental distinguir entre dois tipos principais [7]:

- Consultas Curtas: apenas algumas linhas (máximo de 10% das tabelas mais largas) são necessárias para produzir o resultado. Esse limite pode variar dependendo do caso.

- Consultas Longas: praticamente todas as linhas da tabela contribuem para o resultado. Além disso, as consultas longas podem ser incrementais ou não. As incrementais retornam apenas os dados que foram inseridos ou atualizados desde a última execução da consulta. As não-incrementais buscam sempre todos os dados.

Essa distinção é crucial, pois as estratégias de otimização variam de acordo com o tipo de consulta.

Dado o caráter dinâmico das funcionalidades, que permitem filtragens com graus de seletividade variados, dependendo das intenções dos usuários, categorizar as consultas pode ser desafiador. No entanto, ao considerar o pior caso (sem filtragem aplicada), é possível enquadrá-las como consultas longas, especificamente do tipo não-incremental, já que todos os dados das tabelas precisam ser acessados para produzir um resultado.

Os passos adotados para otimizar as consultas foram:

1. Adequação dos dados recuperados pela consulta aos requisitos de cada funcionalidade: Durante a otimização, a consulta de corregedoria, originada da consulta de fluxograma, teve diversas cláusulas e colunas desnecessárias removidas, reduzindo a complexidade da consulta.

2. Execução dos *joins* mais restritivos primeiro: Iniciar a consulta da perspectiva da tabela de movimentos e fazer o primeiro *join* com a tabela de processos ajudou a reduzir os resultados indesejados ao eliminar processos sem nenhum movimento.

3. Inclusão de uma nova coluna ”dataPrimeiroMovimento” para armazenar a data de início do primeiro movimento registrado para aquele processo, eliminando a necessidade de subconsulta e *join* subsequente. O comando utilizado pode ser visto no Código 3.1.

4. Adição de *index* na coluna ”dataPrimeiroMovimento”: A introdução de um *index* nessa coluna pode acelerar a consulta, especialmente quando há filtragem nesse campo. O comando utilizado pode ser visto no Código 3.2.

5. Criação de *materialized view*: Essa técnica envolve a criação de uma nova tabela com o resultado de uma consulta, acelerando as buscas de dados. Contudo, *materialized views* replicam dados, trazendo custos de armazenamento redundante.

Os comandos para criação das *materialized views* de cada funcionalidade podem ser vistos nos Códigos 3.3 e 3.5. Após implementadas todas as mudanças, as consultas ficam significativamente menos verbosas, como pode ser observado nos Códigos 3.4 e 3.6. O custo de armazenamento desse novo objeto será somado ao custo de armazenamento das tabelas originais para medir o impacto dessa medida no tamanho do banco.

Código 3.1 Criação da Coluna "dataPrimeiroMovimento".

```
UPDATE processos_xxxx
SET "dataPrimeiroMovimento" = min_subquery.earliest_date,
FROM (
    SELECT
    mov."processoID" AS "processoID",
    min(mov."dataFinal") AS earliest_date,
    FROM movimentos_xxxx mov
    GROUP BY mov."processoID") min_subquery
WHERE processos_xxxx."processoID" = min_subquery."processoID"
```

Código 3.2 Criação do *Index* na Nova Coluna "dataPrimeiroMovimento".

```
CREATE INDEX idx_dataPrimeiroMovimento
ON processos_xxxx("dataPrimeiroMovimento");
```

Código 3.3 Criação da *Materialized View* de Fluxograma.

```
CREATE MATERIALIZED VIEW mat_view_fluxograma_xxxx AS
SELECT
proc."NPU",
proc."processoID",
proc."dataPrimeiroMovimento",
c."descricao" AS "classe",
a."descricao" AS "assunto",
mov."activity",
mov."dataInicio",
mov."dataFinal",
mov."usuarioID",
mov."duration",
COALESCE(s."nomeServidor", 'N/I') AS "nomeServidor",
COALESCE(s."tipoServidor", 'N/I') AS "tipoServidor"
FROM movimentos_xxxx mov
JOIN processos_xxxx proc
ON mov."processoID" = proc."processoID"
JOIN classes c ON proc.classe = c.id
```

```
JOIN assuntos a ON proc.assunto = a.id
LEFT OUTER JOIN servidores s
ON mov."usuarioID" = s."servidorID"
ORDER BY proc."processoID", mov."dataFinal" ASC;
```

Código 3.4 Nova Consulta para Fluxograma

```
SELECT *
FROM mat_view_fluxograma_xxxx mat_view
WHERE mat_view."dataPrimeiroMovimento" >= '1900-01-01';
```

Código 3.5 Criação da *Materialized View* de Corregedoria.

```
CREATE MATERIALIZED VIEW mat_view_corregedoria_xxxx AS
SELECT
mov.activity,
mov."dataInicio",
mov."dataFinal",
mov.duration,
mov."processoID",
proc."NPU",
proc."dataPrimeiroMovimento",
classes.descricao AS classe
FROM movimentos_xxxx mov
JOIN processos_xxxx proc
ON mov."processoID" = proc."processoID"
JOIN classes ON proc.classe = classes.id
ORDER BY proc."processoID", mov."dataFinal" ASC;
  :
```

Código 3.6 Nova Consulta para Corregedoria.

```
SELECT *
FROM mat_view_corregedoria_xxxx mat_view
WHERE mat_view."dataPrimeiroMovimento" >= '1900-01-01';
```

CAPÍTULO 4

Resultados

4.1 Teste de Hipótese

Para testar as hipóteses, foi assumida a não-normalidade dos dados, em razão da natureza das consultas impactadas pelos diferentes volumes de dados nas unidades jurídicas. O teste não-paramétrico escolhido foi o Teste U de Mann-Whitney, com um nível de significância de 0.05.

Os resultados dos P-Valores estão na Tabela 4.1. Com ambos os P-Valores menores que 0.05, rejeitamos as hipóteses nulas, indicando que as medições de tempo de execução das consultas otimizadas são significativamente diferentes das medições das consultas originais.

Como ambos os P-Valores são menores que o nível de significância de 0.05, ambas as hipóteses nulas podem ser rejeitadas, demonstrando que as medições de tempo de execução das consultas otimizadas são significativamente diferentes das medições aferidas a partir das consultas originais. Na próxima seção será discutida a natureza dessa diferença.

4.2 Gráficos Comparativos dos Resultados

Os gráficos de densidade comparando a distribuição dos tempos de execução antes e após a otimização das consultas de Fluxograma e Corregedoria estão nas Figuras 4.1 e 4.2, respectivamente.

As curvas dos tratamentos originais são semelhantes, pois a consulta de Corregedoria original é idêntica à de Fluxograma. As curvas dos tratamentos otimizados são similares, mas não idênticas, devido às diferenças nas *materialized views*.

Em todas as curvas, observa-se um comportamento recorrente: uma curva mais alta seguida de uma mais baixa. Esses picos em consultas que levaram mais tempo são provenientes das unidades jurídicas que possuem mais registros para serem processados, naturalmente acarretando em mais tempo de execução.

As consultas otimizadas mostram uma melhoria significativa, com uma curva aguda entre

	Fluxograma	Corregedoria
Valor-P	5,044739930247448e-34	3,9939242464420235e-34
Veredito	Hipótese nula rejeitada	Hipótese nula rejeitada

Tabela 4.1 Resultados dos Testes U de Mann-Whitney

0 e 1 segundos. indicando que grande parte está concentrada entre esses valores, enquanto as originais exibem uma concentração entre 0 e 3 segundos, com outra notável entre 6 e 8 segundos.

Na Figura 4.3, é possível notar as diferenças entre os tempos de execução registrados para cada tratamento por meio de um Diagrama de Caixas, ou Boxplot, uma ferramenta visual eficaz para representar e comparar distribuições de valores numéricos entre diferentes grupos. Ele exibe de forma concisa diversas estatísticas descritivas, como a mediana, quartis, e possíveis valores atípicos.

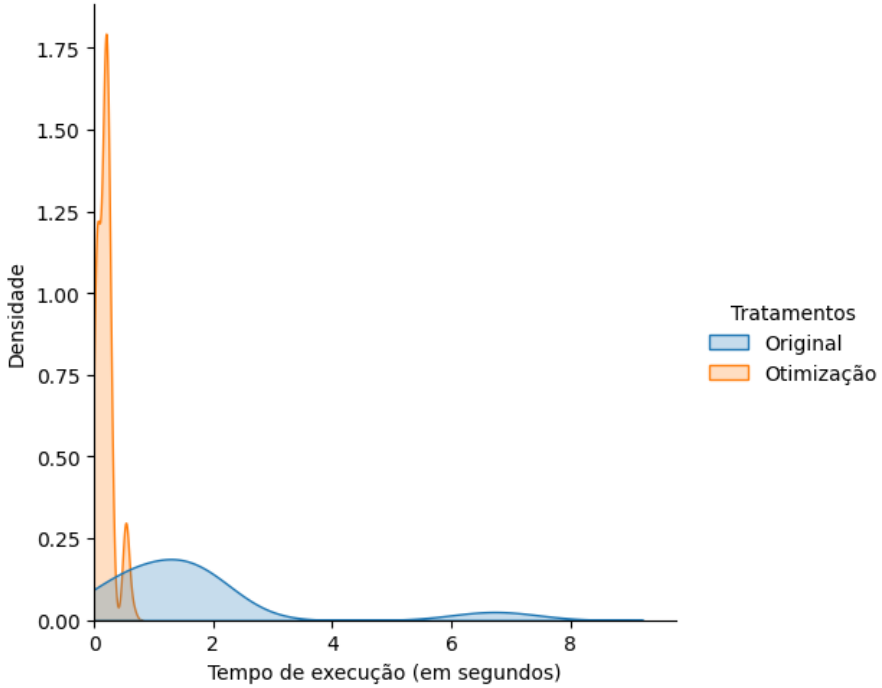

Figura 4.1 Comparação entre os tratamentos nas consultas de Fluxograma.

4.2.1 Tempos de Execução Médios

A diferença entre os tempos médios de execução para cada unidade, antes e depois da otimização, está nas Tabelas 4.2 e 4.3. Os desvios-padrões podem ser conferido na Tabela 4.4.

4.2.2 Custo de Armazenamento

O aumento expressivo no desempenho tem um custo. A Figura 4.4 mostra a comparação entre os custos de armazenamento antes e depois das otimizações. O espaço em MB necessário triplicou devido à introdução das *materialized views* e da nova coluna.

Em resumo, a otimização resultou em melhorias significativas no tempo de execução das consultas, mas isso veio à custa de um aumento substancial no espaço de armazenamento necessário. A decisão de implementar essas otimizações dependerá da importância relativa do

Unidade	Consulta original	Consulta otimizada
3843	1,543	0,188
7433	1,998	0,260
8380	1,701	0,215
13325	1,283	0,170
14009	0,396	0,045
14013	0,294	0,033
17384	1,325	0,200
18007	0,356	0,040
18104	1,909	0,241
18228	0,877	0,115
44508	6,738	0,544

Tabela 4.2 Tempos de execução médios em segundos das consultas de Fluxograma

Unidade	Consulta original	Consulta otimizada
3843	1,558	0,195
7433	2,025	0,254
8380	1,811	0,225
13325	1,273	0,182
14009	0,399	0,046
14013	0,303	0,033
17384	1,313	0,160
18007	0,351	0,038
18104	1,958	0,246
18228	0,867	0,118
44508	6,776	0,546

Tabela 4.3 Tempos de execução médios em segundos das consultas de Corregedoria

Módulo	Consulta original	Consulta otimizada
Fluxograma	1,708	0,138
Corregedoria	1,72	0,138

Tabela 4.4 Desvios-padrões dos tempos de execução em segundos.

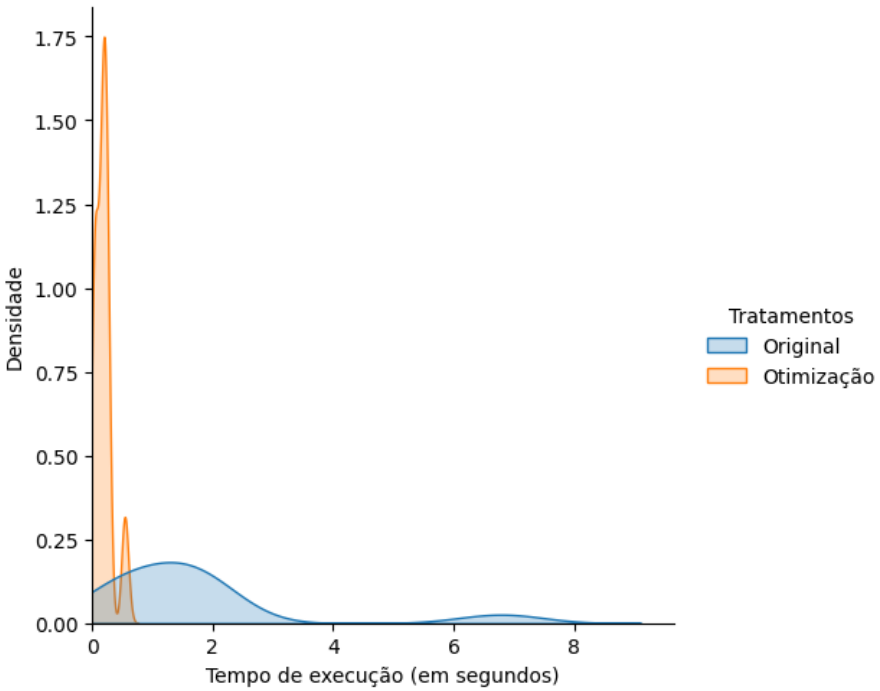

Figura 4.2 Comparação entre os tratamentos nas consultas de Corregedoria.

desempenho e dos custos de armazenamento para a aplicação específica.

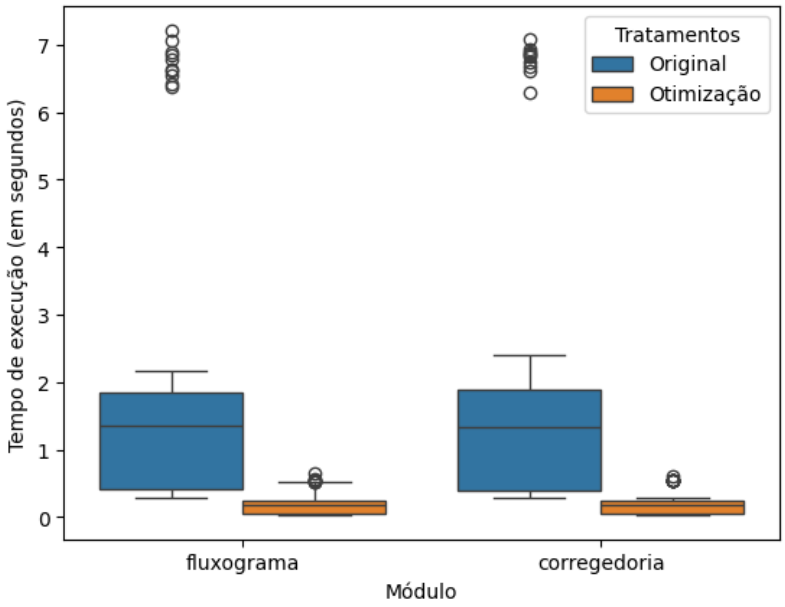

Figura 4.3 Diagramas de caixa das consultas originais e otimizadas.

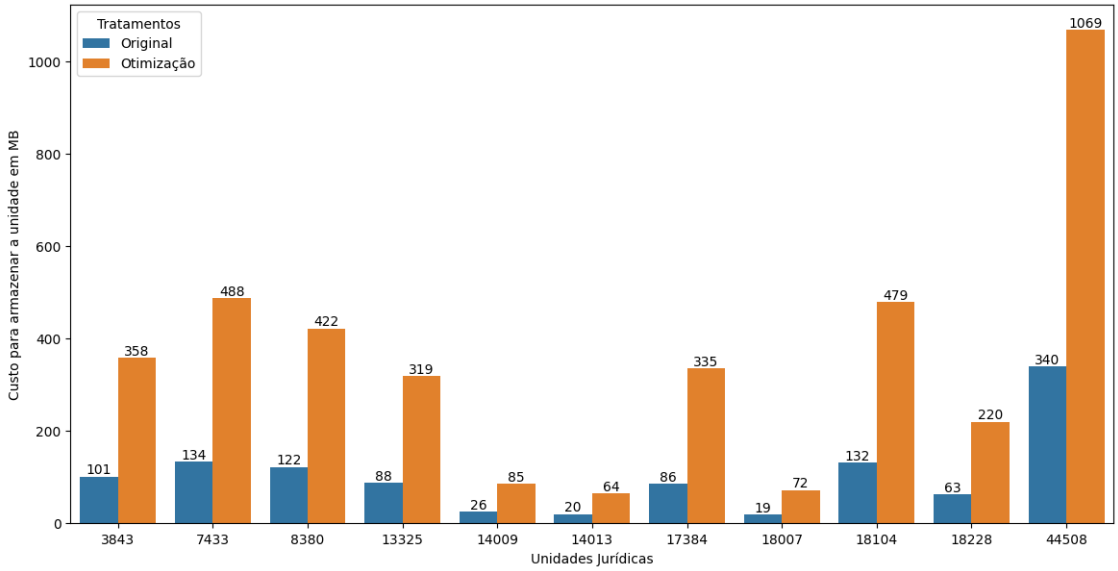

Figura 4.4 Comparação entre os custos de armazenamento em MB das unidades judiciárias, antes e depois das otimizações.

CAPÍTULO 5

Considerações Finais e Limitações

O estudo revelou uma melhora significativa no tempo de execução médio das consultas após a otimização. Contudo, esse ganho acompanha um aumento considerável no custo de armazenamento dos dados. Além disso, a implementação de *materialized views* requer uma manutenção regular para garantir que os dados estejam atualizados. A frequência dessa manutenção dependerá dos requisitos específicos da aplicação. Entretanto, a similaridade entre as *materialized views* resultantes sugere que uma única *materialized view* poderia ser explorada para ambas as consultas, mitigando o impacto no custo de armazenamento.

Para análise de dados eficientes, é ideal que a maior quantidade possível de dados seja integrada, constituindo o conceito de *Big Data*, caracterizado por volumes significativos de informação distribuída entre locais de armazenamento diferentes (frequentemente de maneira geográfica), em vários formatos e com mudanças frequentes. O JuMP - CNJ, em relação aos sistemas regentes do CNJ, assinala alguns desses pontos, principalmente o volume e distribuição dos dados. Uma abordagem clássica e potencialmente menos onerosa para análise de informações com essas qualidades seria a refatoração completa do modelo lógico do banco de dados, visando características de um *data warehouse* (DW). DW's podem ser compreendidos como uma coleção de tecnologias que permitem trabalhadores com conhecimento de regras de negócio (executivos, gerentes e analistas) tomar decisões melhores e mais rapidamente. Sistemas orientados à transações como o PostgreSQL são inapropriados para as demandas de suporte à decisão e redes de alta velocidade não resolvem o problema da acessibilidade de informações geograficamente distribuídas. O armazenamento de dados em DW's é uma estratégia importante para integração de fontes de informação distintas e o desenvolvimento de sistemas analíticos (em contrapartida aos tradicionais sistemas transacionais) [8]. O JuMP - CNJ já dá passos nessa direção ao concentrar os dados de diversas unidades jurídicas espalhadas pelo país em um banco único, de maneira uniformizada. O modelo conceitual do JuMP - CNJ pode ser utilizado do jeito que está para a composição de *data marts* (DM), que são essencialmente pequenos DW's, contendo apenas um sub-conjunto do DW completo. Eles podem ser implementados por meio de bancos de dados relacionais e geralmente são acessíveis apenas para um departamento da instituição [8]. Na prática, cada unidade jurídica consumiria de seu próprio DM, de forma muito semelhante ao que acontece atualmente. Essa escolha de design continuaria a permitir a segmentação de dados por unidade jurídica, reduzindo o tempo de respostas de consultas locais por conta do menor volume de dados e possibilitando a distribuição da carga de trabalho entre máquinas diferentes. A novidade seria a existência do DW global, com um maior volume de dados e já contendo o resultados dos processos de extração e transformação custosos que buscamos evitar, a fim de não os repetir com frequência. Como a ferramenta está num avançado estado de produção, seria possível fazer uso das regras de negócios e documentação

de funcionalidades que já estão bem definidas. As propostas apresentadas nesta investigação, no fim, revelaram um artefato muito semelhante ao conceito de modelo unificado, característico de DW's [20]. Um estudo de caso voltado para a remodelagem completa da camada de banco de dados como um *data warehouse* poderia trazer resultados muito positivos para o tempo de execução das consultas, com *trade-off's* mais amigáveis. O processo de design de um desses contém os seus próprios desafios e incertezas. Uma das metodologias propostas [20] exige:

1. Fase de conceituação: os usuários evidenciam que tipo de análises serão feitas e seus requisitos não funcionais e essas ideias desestruturadas são traduzidas para algum modelo conceitual.

2. Design lógico: o modelo conceitual gerado na etapa anterior é utilizado para definição de um esquema lógico de um modelo unificado dos dados.

3. Design físico: Finalmente, o esquema lógico pode ser mapeado e implementado num esquema de banco de dados.

A pesquisa focou apenas em duas funcionalidades específicas da ferramenta JuMP - CNJ, por sua relevância e frequência de uso. A falta de exemplos de consultas curtas e a seleção específica dessas funcionalidades podem limitar a generalização dos resultados para outras partes da aplicação. O objetivo nunca foi otimizar todas as consultas da aplicação, mas sim comprovar a possibilidade de fornecer um serviço mais ágil e eficácia dos guias de otimização neste caso específico, assim como medir a dimensão das melhorias nos resultados. Dito isso, a ferramenta conta com muitas outras consultas que, em retrospecto, poderiam ter sido melhores exemplos para os tratamentos por possuírem requisitos verdadeiramente distintos. Em especial, destaque-se a falta de pelo menos um exemplo de consulta curta para que fossem medidos os impactos das otimizações típicas desse perfil.

Ademais, o processo de otimização é um ato contínuo no desenvolvimento de um sistema. Não existe uma abordagem única que sempre gerará o melhor resultado possível. Na realidade, até mesmo as decisões que podem parecer óbvias dependem de diversas variáveis, exigindo que os desenvolvedores exerçam cautela e analisem um caso de cada vez. No contexto de bancos de dados, muitas medidas que visam otimizar a recuperação dos registros, como o próprio ato de criar *indexes* em colunas, frequentemente resultam em encargos nas demais operações, de tal modo que deve haver uma preocupação constante em analisar se os ganhos justificam os custos [15].

Um trabalho futuro pode expandir a cobertura dos tratamentos para mais consultas SQL no JuMP - CNJ, explorando um novo conjunto de melhorias e otimizações para consultas curtas, assim como a continuação da medição e atestamento de seus impactos.

A vastidão e precisão dos dados pode ser uma ótima fonte de alimentação para modelos de Inteligência Artificial. Embora seja necessário voltar para a etapa a fase de idealização e levantamento de requisitos, uma metodologia de gerenciamento de projetos como CRISP-DM poderia ser capaz de revelar dores e interesses do usuários solucionáveis com técnicas de aprendizagem de máquina e os dados consumidos pela aplicação [6].

Para concluir, a última sugestão é acerca da avaliação contínua do desempenho da ferramenta. Para identificar problemas, quando visíveis, ou até analisar a tendência com o passar

do tempo e prever possíveis problemas, seria interessante um módulo de testes automatizados que mede, além do funcionamento correto da aplicação, o tempo decorrido para obtenção dos resultados e os recursos de processamento e armazenamento utilizados. Como qualquer funcionalidade, a implementação desse módulo de observabilidade exigiria dos desenvolvedores planejamento e tempo dedicado à sua conclusão. Porém, os dados gerados por essa medida seriam muito valiosos em investigações semelhantes a essa no futuro. Os resultados poderiam ser utilizados para validação de alterações realizadas sob a plataforma numa escala de tempo mais vasta que a que foi possível utilizar nessa pesquisa.

Referências

[1] Lawrence Chung, Brian A. Nixon, Eric Yu, and John Mylopoulos. *Introduction to Non-Functional Requirements in Software Engineering*, pages 1–9. Springer US, Boston, MA, 2000.

[2] Conselho Nacional de Justiça. Cnj - quem somos. Acessado em 14/06/2023.

[3] Conselho Nacional de Justiça. Plataforma codex - portal cnj. Acessado em 14/06/2023.

[4] The PostgreSQL Global Development Group. Postgresql - admin functions, 2022. Acessado em 16/06/2023.

[5] The PostgreSQL Global Development Group. Postgresql - error reporting and logging, 2022. Acessado em 16/06/2023.

[6] Muhammad Tamiramin Hayat Suhendar and Yani Widyani. Machine learning application development guidelines using crisp-dm and scrum concept. In *2023 IEEE International Conference on Data and Software Engineering (ICoDSE)*, pages 168–173, 2023.

[7] Henrietta Dombrovskaya, Boris Novikov and Anna Bailliekova. *PostgreSQL Query Optimization: The Ultimate Guide to Building Efficient Queries*. Apress Berkeley, CA, 2021.

[8] Matthias Jarke, Maurizio Lenzerini, Yannis Vassiliou, and Panos Vassiliadis. *Data Warehouse Practice: An Overview*, pages 1–14. Springer Berlin Heidelberg, Berlin, Heidelberg, 2003.

[9] Jr. Michael L. Rupley. Introduction to query processing and optimization. Indiana University at South Bend, 2008.

[10] Microsoft. Descrição das noções básicas de normalização do banco de dados. Acessado em 14/06/2023.

[11] The pgAdmin Development Team. Pgadmin - query tool, 2023. Acessado em 05/10/2023.

[12] GILVANO PIONTKOSKI. Comparativo de desempenho de consultas sql entre banco de dados em memÓria ram e em ssd. *2018 7th Brazilian Conference on Intelligent Systems (BRACIS)*, pages 200–205, 2018.

[13] Shamkant B. Navathe Ramez Elmasri. *Fundamentals of Database Systems*. Pearson, 2015.

[14] G.L. Sanders and Seungkyoon Shin. Denormalization effects on performance of rdbms. In *Proceedings of the 34th Annual Hawaii International Conference on System Sciences*, pages 9 pp.–, 2001.

[15] Seema Sultana and Sunanda Dixit. Indexes in postgresql. In *2017 International Conference on Innovative Mechanisms for Industry Applications (ICIMIA)*, pages 512–515, 2017.

[16] Adriana Jacoto Unger, José Francisco dos Santos Neto, Marcelo Fantinato, Sarajane Marques Peres, Julio Trecenti, and Renata Hirota. Process mining-enabled jurimetrics: Analysis of a brazilian court's judicial performance in the business law processing. In *Proceedings of the Eighteenth International Conference on Artificial Intelligence and Law*, ICAIL '21, page 240–244, New York, NY, USA, 2021. Association for Computing Machinery.

[17] Wil M. P. van der Aalst. *Process Mining: Discovery, Conformance and Enhancement of Business Processes*. Springer International Publishing, Den Dolech 2, 5612 AZ Eindhoven, The Netherlands, 2011.

[18] Maikel L. van Eck, Xixi Lu, Sander J. J. Leemans, and Wil M. P. van der Aalst. Pm²: A process mining project methodology. In Jelena Zdravkovic, Marite Kirikova, and Paul Johannesson, editors, *Advanced Information Systems Engineering*, pages 297–313, Cham, 2015. Springer International Publishing.

[19] Claes Wohlin, Per Runeson, Martin Höst, Magnus C. Ohlsson, Björn Regnell, and Anders Wesslén. *Experimentation in Software Engineering*. Springer, 2012.

[20] Wang Zhijuan, Wei Hongchang, and Wu Xuefang. A data warehouse design method. In *2012 International Conference on Computer Science and Service System*, pages 2063–2066, 2012.