UNIVERSIDADE DE SÃO PAULO

Instituto de Ciências Matemáticas e de Computação

Armazenamento e Processamento de Dados de Comércio Internacional Usando TimescaleDB

Bruno Del Monde

São Carlos – SP

Armazenamento e Processamento de Dados de Comércio Internacional Usando TimescaleDB

Bruno Del Monde

Orientador: **Prof. Dr. Elaine Parros Machado de Sousa**

Monografia final de conclusão de curso apresentada ao Instituto de Ciências Matemáticas e de Computação – ICMC-USP, como requisito parcial para obtenção do título de Bacharel em Ciências de Computação.
Área de Concentração: Bases de Dados

USP – São Carlos
Junho de 2021

Monde, Bruno Del
Armazenamento e Processamento de Dados de Comércio Internacional Usando TimescaleDB / Bruno Del Monde. - São Carlos - SP, 2021.
38 p.; 29,7 cm.

Orientador: Elaine Parros Machado de Sousa.
Monografia (Graduação) - Instituto de Ciências Matemáticas e de Computação (ICMC/USP), São Carlos - SP, 2021.

1. Banco de dados. 2. TimescaleDB. 3. Séries temporais. 4. PostgreSQL. 5. Comércio internacional. 6. ECI. I. Sousa, Elaine Parros Machado de. II. Instituto de Ciências Matemáticas e de Computação (ICMC/USP). III. Título.

RESUMO

MONDE, B. D.. **Armazenamento e Processamento de Dados de Comércio Internacional Usando TimescaleDB**. 2021. 38 f. Monografia (Graduação) – Instituto de Ciências Matemáticas e de Computação (ICMC/USP), São Carlos – SP.

Este trabalho descreve a criação de um banco de dados relacional otimizado com técnicas de tratamento de séries temporais para armazenamento de dados de comércio internacional, disponibilizados pela ONU através do UN Comtrade. A solução proposta, implementada com o SGBD PostgreSQL e a extensão TimescaleDB, utiliza particionamento de tabelas temporais para garantir agrupamento de dados relacionados, localidade espacial e divisão do tamanho de índices de forma a otimizar padrões de acesso bem localizados temporalmente. Também foram implementadas operações de cálculo de medidas de interesse: similidade de séries e medidas de complexidade econômicas tanto geográficas quanto de produtos, com suporte a cálculo em diversas granularidades geográficas (países, continentes e grupos arbitrários de países), de produtos (agrupamentos do sistema classificação de produtos HS) e intervalos temporais. Os resultados dos experimentos revelaram que o particionamento das tabelas promoveu um ganho de performance, em relação a uma tabela convencional, de cerca de 34% em consultas filtradas por um intervalo de 3 anos e mesmo para consultas envolvendo uma alta parcela dos anos da tabela houve ganho de performance.

Palavras-chave: Banco de dados, TimescaleDB, Séries temporais, PostgreSQL, Comércio internacional, ECI.

ABSTRACT

MONDE, B. D.. **Armazenamento e Processamento de Dados de Comércio Internacional Usando TimescaleDB**. 2021. 38 f. Monografia (Graduação) – Instituto de Ciências Matemáticas e de Computação (ICMC/USP), São Carlos – SP.

This thesis defines the creation of a relational database optimized with time series treatment techniques for storing international trade data, made available by the UN through UN Comtrade. The proposed solution, implemented with PostgreSQL DBMS and the TimescaleDB extension, uses temporal table partitioning to ensure grouping of related data, spatial locality, and index size division in order to optimize well-located in time access patterns. Measures of interest calculate operations were also implemented: series similarity and geographic and product complexity measures, with support for calculation in various geographic granularities (countries, continents and arbitrary groups of countries), of products (groupings of HS product classification system) and time intervals. The results of the experiments revealed that the partitioning of the tables promoted a performance gain, compared to a conventional table, of about 34 % in queries filtered over a 3-year interval and even for queries involving a high portion of the table's years, there was a performance gain.

Keywords: Database, TimescaleDB, Time series, PostgreSQL, International trade, ECI.

LISTA DE ABREVIATURAS E SIGLAS

BEC ..... *Broad Economic Categories*
ECI ..... *Economic Complexity Index*
GBDI .... Grupo de Bases de Dados e Imagens
HS ..... *Harmonized System*
MER ..... Modelo Entidade Relacionamento
PCI ..... *Product Complexity Index*
PIC ..... *Position-Independent Code*
SGBD .... Sistema Gerenciador de Banco de Dados
SITC ..... *Standard International Trade Classification*
SLRU .... *Simple Least Recently Used*
SPI ..... *Server Programming Interface*

SUMÁRIO

1 INTRODUÇÃO 11
1.1 Motivação e Contextualização 11
1.1.1 Origem e importância econômica dos dados *11*
1.1.2 Conceitos fundamentais *12*
1.2 Objetivos 12
1.3 Organização 13

2 MÉTODOS, TÉCNICAS E TECNOLOGIAS UTILIZADAS 15
2.1 Medidas 15
2.2 Séries temporais e medidas de similaridade 16
2.3 PostgreSQL e TimescaleDB 16
2.4 Linguagem C++ 18
2.4.1 Interface PostgreSQL *18*
2.4.2 OpenMP *19*
2.5 Docker 19

3 DESENVOLVIMENTO 21
3.1 O Problema 21
3.2 Atividades Realizadas 21
3.2.1 Criação do esquema da base de dados *21*
3.2.2 Implementação TimescaleDB *24*
3.2.3 Índices *24*
3.2.4 Operações *25*
3.2.5 Implementações alternativas *26*
3.2.6 Testes experimentais *26*
3.3 Resultados 27
3.4 Dificuldades e Limitações 29

4 CONCLUSÃO 31
4.1 O projeto 31
4.2 Principais influências da graduação 31
4.3 Análise da graduação do autor 32

REFERÊNCIAS 35

ANEXO A RESULTADOS DOS TESTES 37

Capítulo 1

INTRODUÇÃO

1.1 Motivação e Contextualização

Na seção subseção 1.1.1 é descrito o conjunto de dados que será utilizado de base para o projeto. Também é explicada a importância econômica do estudo desse conjunto de dados que motivou o desenvolvimento do presente trabalho. Na seção 1.1.2 são introduzidos conceitos importantes que serão tratados nesse trabalho.

1.1.1 Origem e importância econômica dos dados

A economia internacional tem como objeto de estudo as transações comerciais internacionais, os organismos internacionais que regulam o mercado global, os blocos econômicos, assim como as normas e acordos que regulam essa atividade. O crescimento da globalização financeira reduz a importância das fronteiras do ponto de vista econômico e torna cada vez mais importante o entendimento das relações econômicas internacionais na medida que decisões micro e macroeconômicas são condicionadas ao cenário internacional.

O UN Comtrade[1] é o maior repositório atual de dados de comércio internacional. Com relatórios fornecidos por mais de 170 países/áreas, ele acumula desde 1962 mais de 40 bilhões de registros de transações disponibilizadas publicamente através da internet. Os registros são publicados por mês e ano, consistindo[2] em período, país autor do registro, país parceiro, o período, sentido do fluxo da mercadoria (importação ou exportação do país autor), o valor em dólar, a quantidade (se aplicável), o peso (se aplicável) e o código do produto em diferentes convenções de classificação de produtor (*Standard International Trade Classification* (SITC)[3], *Harmonized System* (HS)[4] e *Broad Economic Categories* (BEC)[5]). Os registros são emitidos por tanto o país importador quanto o exportador e podem inclusive apresentar valores diferentes dado a data em que cada país registrou a transação, o modo como é incluído o custo de transporte e o registro do destino final do produto que pode não ser conhecido pelo exportador.

1 <https://comtrade.un.org/>
2 <https://comtrade.un.org/data/>
3 <https://unstats.un.org/unsd/trade/sitcrev4.htm>
4 <https://unstats.un.org/unsd/tradekb/Knowledgebase/50018/Harmonized-Commodity-Description-and-Coding-Systems-HS>
5 <https://unstats.un.org/unsd/tradekb/Knowledgebase/50089/Classification-by-Broad-Economic-Categories-Rev4?Keywords=Broad+Economic+Categories>

Métricas obtidas a partir desses dados contribuem para o planejamento de políticas econômicas tanto internacionais quanto em nível nacional que afetam a economia interna e a condução de políticas econômicas nacionais. Dentre as questões afetadas pelo balanço de transações internacionais estão na macroeconomia o planejamento do Banco Central sobre taxas de câmbio e política de juros, e na microeconomia o planejamento de diversas áreas e atividades individualmente.

1.1.2 Conceitos fundamentais

Para esse trabalho, dentre os sistemas de classificação de produtos disponibilizados pelo UN Comtrade será usado o HS. O sistema permite que a representação de produtos como um código seja feita de modo uniformizado por qualquer país ou entidade. O código possui 6 dígitos numéricos que podem ser divididos em 3 pares de dígitos representando categorias hierárquicas, por exemplo 09: café, chá, mate e pimentas, 09.02: chá, aromatizado ou não, 09.02.10: chá-verde, não fermentado.

O *Economic Complexity Index* (ECI) é uma medida econômica escalar que relaciona uma geografia, no caso estudado países, com suas produções, no caso estudado exportação de produtos. Essa medida é importante para entender a economia de um país como um todo, pois é calculada unicamente por meio da análise das produções do país, sendo agnóstica em relação a quais fatores foram responsáveis por elas. O índice se relaciona com o grau de sofisticação, a diversidade e resiliência da economia, com o quanto um país possui baixa dependência e tem uma produção de alta demanda. A medida análoga calculada sobre os produtos é o *Product Complexity Index* (PCI) (HIDALGO, 2021).

1.2 Objetivos

O Grupo de Bases de Dados e Imagens (GBDI) do ICMC vem realizando pesquisas voltadas à análise dos dados disponibilizados pelo UN Comtrade. Esse projeto se propõe a criação uma base de dados no PostgreSQL[6], com a extensão para séries temporais TimescaleDB[7] para armazenar os registros publicados pelo UN Comtrade de modo estruturado e otimizado, que possa ser utilizada em trabalhos de pesquisa desenvolvidos por integrantes do GBDI. Além disso, a base deverá conter indicadores econômicos diversos, como PIB e ECI, e tabelas auxiliares necessárias para as operações de interesse descritas no próximo parágrafo, além de índices para recuperação eficiente de dados. Esse trabalho possibilitará os pesquisadores do GBDI explorarem com mais eficiência os dados armazenados.

Outro objetivo é a implementação de operações de interesse para trabalhos relacionados, como cálculo de ECI, de PCI e similaridade entre séries temporais. As operações devem ser

[6] <https://www.postgresql.org/>
[7] <https://www.timescale.com/>

implementadas tendo em vista diferentes níveis de granularidade geográfica, podendo contemplar países, continentes, blocos econômicos e permitir agrupamento arbitrário de países e níveis de agregação de produtos, considerando os agrupamentos do HS pela utilização de um número reduzido de pares de dígitos.

1.3 Organização

No capítulo 2 são descritas as operações e medidas implementadas, descreve a extensão TimescaleDB e as demais tecnologias utilizadas. No capítulo 3 é apresentada a estrutura da base da dados criada assim como a arquitetura escolhida, incluindo testes experimentais que comparam a abordagem utilizada com outras duas alternativas. O capítulo 4 discute os resultados finais do projeto e por fim apresenta um avaliação do curso de Bacharelado em Ciência de Computação feito pelo autor.

Capítulo 2

MÉTODOS, TÉCNICAS E TECNOLOGIAS UTILIZADAS

Nesse capítulo são explicados os métodos pelos quais são implementadas as operações de interesse e posteriormente são introduzidas as tecnologias que foram usadas nesse projeto, com foco na explicação do TimescaleDB.

2.1 Medidas

ECI e PCI

Para o cálculo do ECI consideramos uma matriz de exportação tal que X_{cp} é o valor do produto p exportado pelo país c. X deve ser normalizada para permitir comparação entre países com tamanhos e quantidades de exportação muito diferentes, assim como entre produtos com preços e quantidade de exportação não comparáveis, resultando na matriz R (Equação 2.1), chamada de matriz de especialização (HIDALGO, 2021).

$$R_{cp} = \frac{X_{cp} \sum_{ij} X_{ij}}{\sum_i X_{ip} \sum_j X_{cj}} \tag{2.1}$$

Essa normalização representa a divisão do valor real de exportação pelo valor esperado considerando o volume total de exportações do país e produto, logo, se o valor do elemento for maior que 1, então, pode-se dizer que o país produz mais do produto que seria esperado e portanto, é especializado naquele produto. Com isso, a matriz de especialização R_{cp} é então convertida em uma matriz binária de especialização, chamada de M (Equação 2.2).

$$M_{cp} = \begin{cases} 1 & \text{se } Rcp >= 1 \\ 0 & \text{se } Rcp < 1 \end{cases} \tag{2.2}$$

Com os valores de M é então construída uma matriz assimétrica de proximidade entre os países (Equação 2.3). Cada valor da matriz representa a semelhança entre os países correspondentes à linha e à coluna e é assimétrica porque é normalizada pelo somatório da linha de

M.

$$\widetilde{M}_{cc'} = \sum_p \frac{M_{cp} M_{c'p}}{\sum_i M_{ci} \sum_j M_{jp}} \tag{2.3}$$

Tendo então a matriz $\widetilde{M}$ que tem a característica de ser estocástica, o cálculo do ECI é feito através do cálculo do segundo autovetor com autovalor 1, sendo que o primeiro autovetor é composto de 1s, o segundo autovetor é o que mais carrega informação sobre o sistema. Esse autovetor, definido como K_c (Equação 2.4), corresponde ao vetor dos valores do ECI para os países correspondentes a cada linha.

$$K_c = \widetilde{M}_{cc'} K_{c'} \tag{2.4}$$

O cálculo do vetor K_p contendo os valores de PCI para cada produto é trivial, pois pode ser definido como a média do ECI de todos os países que são especializados em determinado produto como mostrado na Equação 2.5.

$$K_p = \frac{\sum_c M_{cp} K_c}{\sum_i M_{ip}} \tag{2.5}$$

2.2 Séries temporais e medidas de similaridade

Séries temporais são conjuntos de medidas de valores reais distribuídas por intervalos regulares de tempo (MITSA, 2010). As medidas de similaridade entre séries são importantes para o entendimento das séries pois permitem a comparação de séries diferentes.

No contexto desse trabalho as séries temporais são identificadas pelo valor de exportação do país c para o país c' de um determinado produto p. Portando, podendo ser representadas por $S = (c, c', p)$ conforme ilustrado na Figura 1, enquanto um valor da série pode ser representado por S_y, sendo y o ano. Será implementada a distância euclidiana como medida de similaridade entre séries. Dado o intervalo y_1 a y_2, obtém-se a distância euclidiana entre duas séries S_1 e S_2 de acordo com a Equação 2.6.

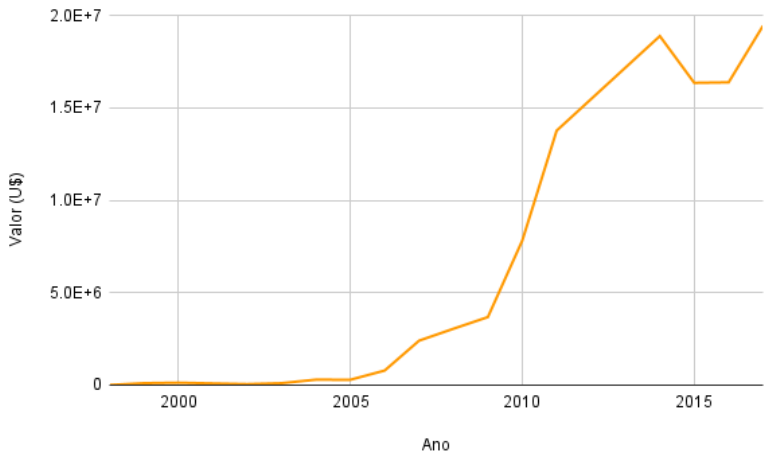
$$D(S_1, S_2, y_1, y_2) = \sqrt{\sum_{y_1 \to y_2} (S_{1y} - S_{2y})^2} \tag{2.6}$$

2.3 PostgreSQL e TimescaleDB

PostgreSQL[1] é um Sistema Gerenciador de Banco de Dados (SGBD) Open Source objeto-relacional, isto é, implementa grande parte do padrão SQL com adição de funcionalidades

[1] <https://www.postgresql.org/>

Figura 1 – Ilustração de série temporal do conjunto de dados: exportação de HS 841239 da China para os Estados Unidos entre 1998 e 2017.



Fonte: Elaborada pelo autor.

pertencentes ao paradigma de orientação a objetos. TimescaleDB[2] é uma extensão ao PostgreSQL que implementa funcionalidades relacionadas a séries temporais.

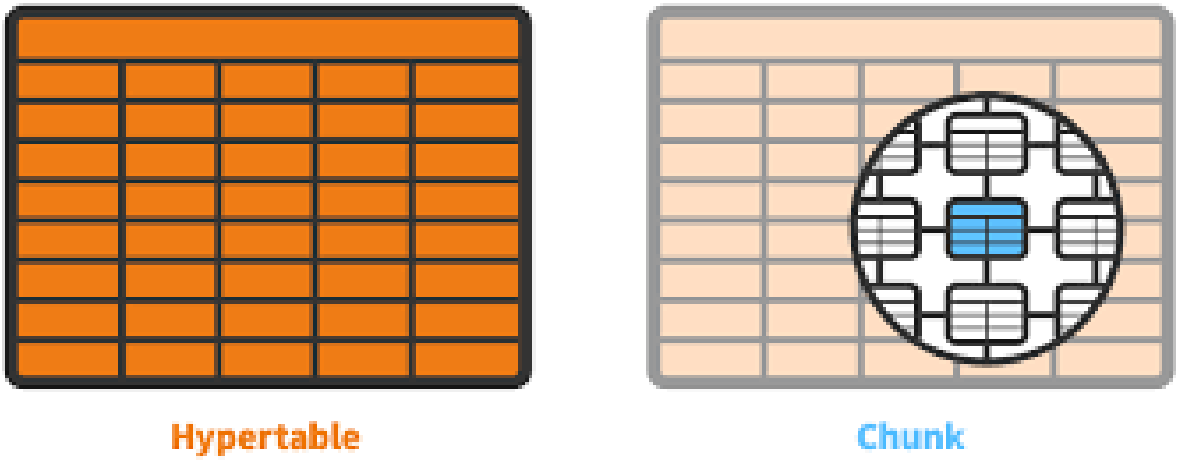
A estrutura usada pelo Timescale é chamada de *hypertable* e criada através do comando `create_hypertable(relation, time_column_name)`, que converte uma tabela existente em uma *hypertable* definindo a coluna referente ao dado temporal. Do ponto de vista do usuário, uma *hypertable* funciona como uma tabela PostgreSQL padrão, podendo ser consultada, ter dados inseridos, alterados ou removidos e podendo receber alterações estruturais (*ALTER TABLE*). Uma *hypertable* tem a interface de uma tabela PostgreSQL padrão, mas como ilustrado na Figura 2 é implementada como um conjunto de *chunks*, cada qual retendo as tuplas pertencentes a um intervalo da série temporal. [3]

Um *chunk* é uma tabela que herda da *hypertable* através dos recursos de orientação a objetos do PostgreSQL[4] e por padrão é criado no esquema *_timescaledb_internal*. Cada *chunk* retém os registros de um certo intervalo de tempo que pode ser definido pelo usuário. Uma consulta efetuada na *hypertable* envolve efetuar sub-consultas nos *chunks*, o que beneficia consultas com restrições na coluna temporal porque permite que somente um subconjunto dos *chunks* seja processado. Os tipos de consulta que mais se beneficiam disso são *scans* sequenciais (inclusive às vezes substituindo o que numa tabela única seria feito através do uso extensivo de um índice com acessos esparsos a disco) e a redução do tamanho de índices.

Ao contrário de tabelas particionadas somente por orientação a objeto do PostgreSQL os *chunks* são gerenciados pelo TimescaleDB, incluindo o cache e mecanismo de consulta. Com a vantagem dos *chunks* não possuírem sobreposição e serem logicamente ordenados é possível

[2] <https://www.timescale.com/>
[3] <https://docs.timescale.com/timescaledb/latest/overview/core-concepts/hypertables-and-chunks/>
[4] <https://www.postgresql.org/docs/current/ddl-partitioning.html#DDL-PARTITIONING-USING-INHERITANCE>

Figura 2 – Ilustração de *hypertable*, com visualização dos *chunks*.



Fonte – <https://legacy-docs.timescale.com/v1.7/introduction/architecture>

resolver algumas das desvantagens do método tradicional de particionamento por herança. Por exemplo, deixa de ser necessário fazer a exclusão de sub-consultas através de, durante o planejamento, verificação exaustiva de *check constrains* previamente adicionadas às sub-tabelas e torna-se possível inclusive que a exclusão seja feita em tempo de execução utilizando resultados de consultas alinhadas e funções não marcadas como *IMMUTABLE*.

2.4 Linguagem C++

As operações de cálculo de índices de complexidade foram implementadas em linguagem C++ tendo em vista o alto custo relativo dessas operações e a necessidade de estruturas de dados complexas. O PostgreSQL permite a definição de funções em linguagens compatíveis com C que são carregadas dinamicamente ao servidor após a compilação em uma biblioteca dinâmica no formato *Position-Independent Code* (PIC) e inclusão, na base de dados, da definição SQL da função com adição da localização do arquivo e o nome da função. Esse método dispensa a necessidade da criação de novos processos e transferência de dados, evitando o *overhead* da criação e comunicação com um novo processo.

2.4.1 Interface PostgreSQL

A interface de programação de funções em C para o PostgreSQL disponibiliza a convenção de chamada versão-1[5] para a passagem de argumentos e retorno de funções. Para interação da função com o banco de dados é utilizada a *Server Programming Interface* (SPI). Seguindo as recomendações da documentação[6] o código utiliza a diretiva `extern "C"` para manter uma

[5] <https://www.postgresql.org/docs/13/xfunc-c.html#id-1.8.3.13.7>
[6] <https://www.postgresql.org/docs/13/xfunc-c.html#EXTEND-CPP>

interface de linkagem compatível com o padrão C e o parâmetro de compilação *-fno-exceptions* para evitar exceções, que são uma fonte de erro caso não sejam tratadas antes de passar pela interface com o servidor compilado em C.

2.4.2 OpenMP

OpemMP[7] é uma API de programação paralela utilizada para otimização das operações implementadas na base. A otimização foi feita no modelo multi-threading com o intuito de permanecer eficiente em um leque maior de ambientes de implantação.

2.5 Docker

Docker[8] é uma tecnologia de virtualização com base em contêineres que possibilita a criação de uma máquina virtual no nível de sistema operacional como camada de compatibilidade entre o banco de dados e a máquina host. A base foi implementada dentro de um contêiner Docker[9] disponibilizado no Docker Hub pelos desenvolvedores do TimescaleDB com o SGBD e a extenção já pré-configurados para permitir maior portabilidade independentemente do servidor aonde ela seja implementada promovendo flexibilidade para a implantação em diferentes servidores ou em nuvem.

[7] <https://www.openmp.org/>
[8] <https://www.docker.com/>
[9] <https://hub.docker.com/r/timescale/timescaledb>

Capítulo 3

DESENVOLVIMENTO

3.1 O Problema

O UN Comtrade disponibiliza dados estruturados de transações do comércio internacional. A proposta deste trabalho é criar uma base de dados para armazenar as séries temporais que são geradas a partir desses dados, de modo a possibilitar recuperação e processamento eficientes voltados a operações sobre séries temporais. A estrutura deve otimizar consultas realizadas nos registros de um único ou de um conjunto reduzido de anos.

Ademais, deseja-se o suporte ao processamento desses dados em diferentes granularidades: continentes, blocos econômicos e agrupamentos arbitrários. Devem ser implementadas operações de cálculo de medidas de interesse como ICE, PCI e cálculo de similaridade entre séries temporais.

3.2 Atividades Realizadas

O desenvolvimento deste trabalho foi realizado em diferentes etapas. Primeiro foi criado um modelo conceitual utilizando Modelo Entidade Relacionamento (MER), a partir da semântica desenvolvida foi realizada a modelação lógica com Modelo Relacional (ELMASRI; NAVATHE, 2007). Então as relações foram implementadas, utilizando TimescaleDB para relações que contém séries temporais. Foram criados índices de apoio as consultas e implementadas as operações de interesse propostas. Por fim, foram implementadas soluções alternativas e realizados testes experimentais com objetivo de comparar a performance das diferentes soluções.

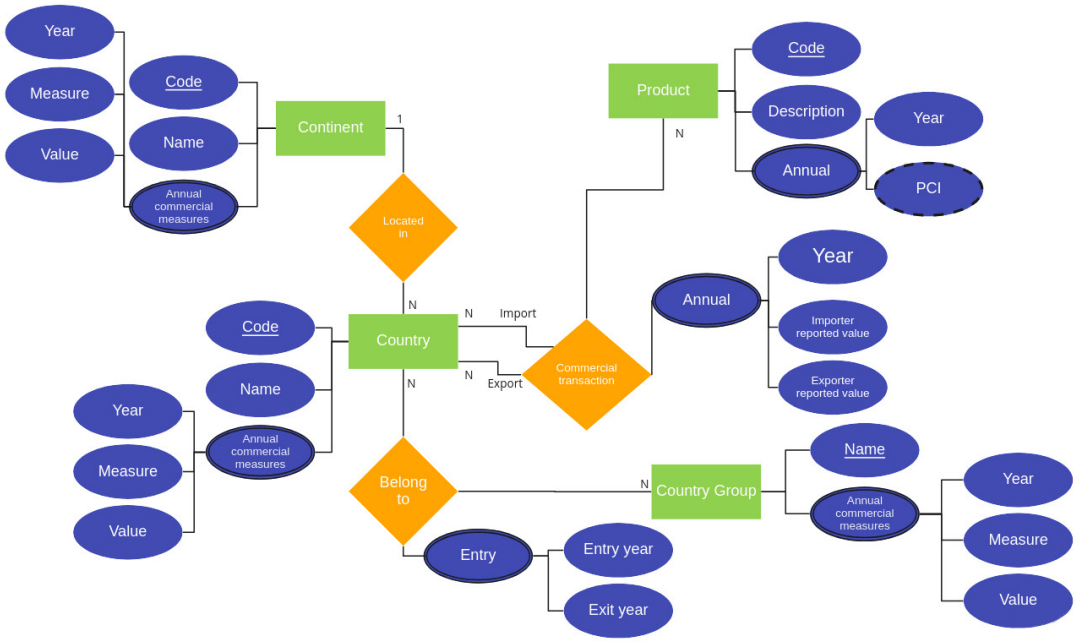
3.2.1 Criação do esquema da base de dados

O esquema conceitual da base de dados, criado usando o MER, é apresentado na Figura 3. A partir da semântica representada nesse esquema, foi desenvolvido o esquema lógico, em Modelo Relacional, a ser criado no PostgreSQL. As relações (tabelas) são apresentadas na Figura 4.

Durante essa etapa do desenvolvimento, foi tomada a primeira decisão importante da solução proposta: modelar a transação comercial como um relacionamento envolvendo país importador, país exportador e produto, integrando dados que são fornecidos em registros

separados na base de dados do UN Comtrade. Com isso, é esperado diminuir o número de tuplas, juntando o valor fornecido pelo país importador e pelo país exportador em uma única tupla. Além disso, essa estrutura é mais semântica na representação do sistema e agrupa informações relacionadas.

Figura 3 – Esquema conceitual



Fonte: Elaborada pelo autor.

As medidas de interesse diversas (PIB, população, ...), representadas no esquema como *Annual Commercial Mesuares*, são reportadas por país e os valores referentes a granularidades maiores são obtidos por agregação. Portanto, foi decidido implementar uma tabela para registros nacionais e obter as granularidades maiores através de *views* e consultas de agregação.

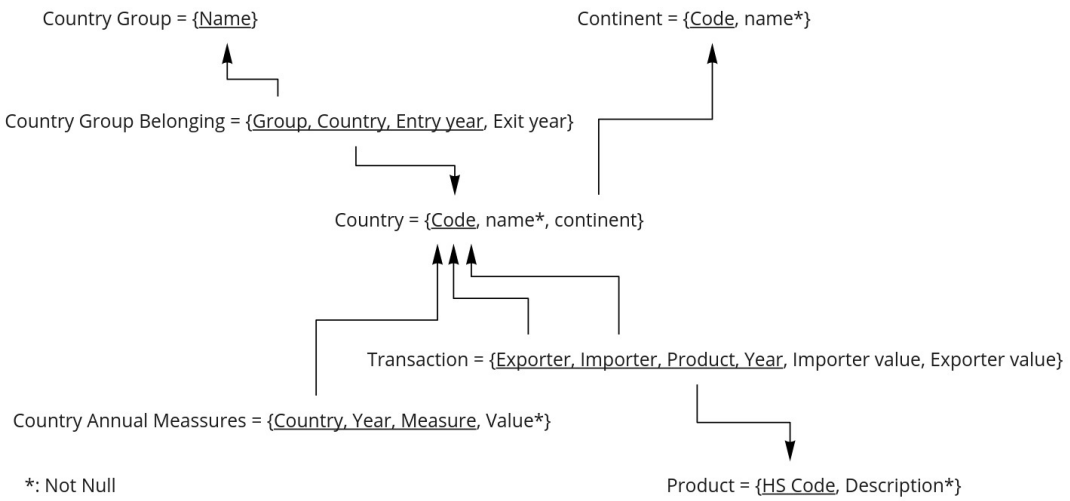
A relação *country* tem como *Primary Key* os códigos dos países de acordo com a ISO 3166-1 Alpha-3[1] e também contém os respectivos nomes. Essa relação é referenciada por *Foreign Keys* de diversas outras tabelas.

A relação *country_annual_measure* armazena uma série temporal que agrupa diversas medidas de interesse de cada ano para cada país. A relação foi criada no formato contendo *measure* e *value* para haver mais flexibilidade no seu uso e novas medidas poderem ser acrescentadas posteriormente sem necessitar a reescrita de toda a tabela.

A relação *continents* contém um código único para cada continente de modo a possibilitar, com a chave estrangeira de *country*, a agregação continental em consultas envolvendo os países. É possível tanto calcular medidas de *country_annual_measure* agregadas (ex. o PIB de um continente), como realizar uma das operações implementadas (ex. o cálculo do ICE do continente).

[1] <https://www.iso.org/glossary-for-iso-3166.html>

Figura 4 – Modelo lógico



Fonte: Elaborada pelo autor.

A relação *country_group* representa grupos de países utilizados da mesma maneira que a relação *continents* para consultas de agregação. Entretanto, essa relação é mais flexível, pois os grupos podem ser definidos arbitrariamente, contendo tanto blocos econômicos quanto alianças comerciais e outras divisões lógicas que podem ser incluídas conforme a necessidade. O pertencimento de um país a um grupo de países é uma relação N para N não estática, isto é, acontece por períodos determinados de tempo.

A relação *country_group_belonging* contém os registros de pertencimento de um país a um grupo de países. Essa relação, além do nome do grupo e do código do país, contém o ano de entrada do país no grupo e pode conter o ano de saída, caso o país tenha se retirado do grupo, ou o valor NULL, caso contrário. Através dessa relação é possível modelar a entrada e saída de países a grupos em anos específicos.

A relação *product* contém os códigos dos produtos no sistema HS e a descrição dos produtos representados por cada código. Essa relação contém aproximadamente 5300 produtos definidos pelo sistema HS. Todos os pares de dígitos são armazenados no mesmo campo, o que não implica em problemas de granularização dado a possibilidade de extrair a parte necessária com a função `left(string, n)`[2].

A relação *transaction* contém os dados do UN Comtrade convertidos ao formato proposto, tem um número de tuplas na ordem de centenas de milhões. Portanto, é a relação mais custosa da base de dados além do principal alvo de consultas. Cada tupla contém o ano, o país exportador, o país importador, o produto e os valores reportados por ambos os países na transação.

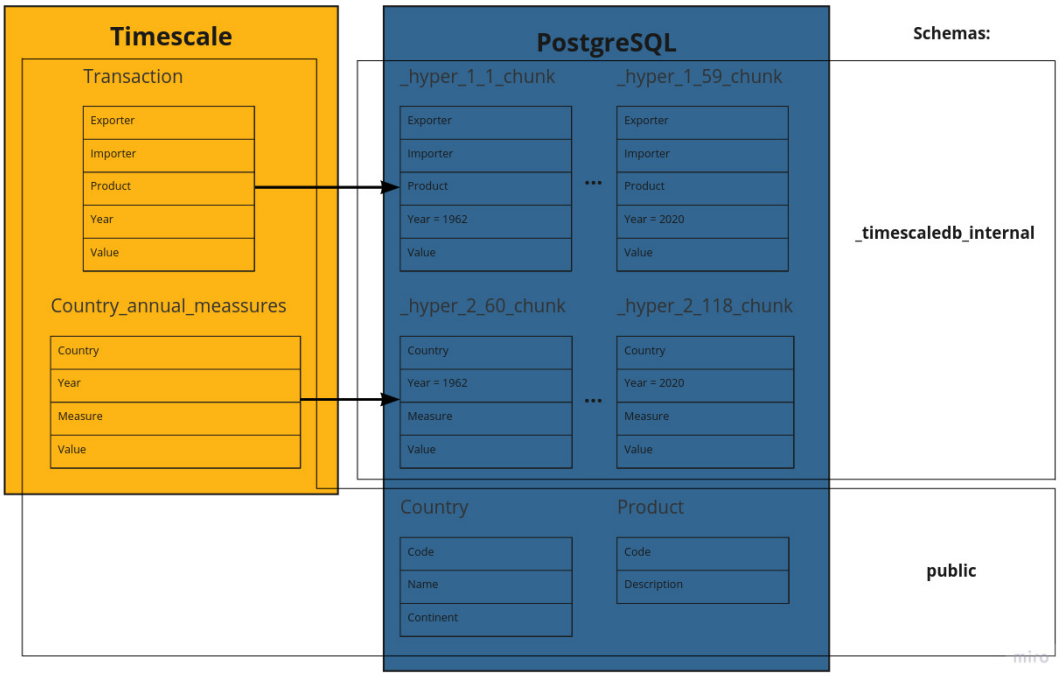
[2] <https://www.postgresqltutorial.com/postgresql-left/>

3.2.2 Implementação TimescaleDB

A solução proposta consiste na implementação das relações que contém séries temporais (*transaction* e *country_annual_measure*) como *hypertables* enquanto as demais relações são implementadas como tabelas convencionais. A relação *country_group_belonging* também contém dados temporais. Entretanto, não apresenta quantidade significativa de registros que compense a partição por *hypertable* e possui um padrão de acesso com inigualdade no campo *Entry year*, dado que para saber se um pertencimento existe em um ano, deve-se verificar se ele foi estabelecido anteriormente e não foi encerrado. Esse padrão não favorece a restrição da consulta em um número reduzido de *chunks* e provavelmente tornaria a consulta mais custosa.

Na figura 5 é ilustrada a implementação das *hypertables* da solução desenvolvida, juntamente com duas das tabelas convencionais. Os *chunks* foram configurados para utilizar a coluna de tipo inteiro *Year* de cada tabela como coluna temporal e terem um intervalo de valor 1.

Figura 5 – Implementação TimescaleDB



Fonte: Elaborada pelo autor.

3.2.3 Índices

Cada relação possui um índice que implementa sua *Primary Key* criado automaticamente pelo PostgreSQL, no momento da definição da mesma. Nessa seção são descritos os índices pertencentes à relação *transaction*, que requer uma estrutura de indexação mais complexa.

O índice *transaction_pkey* implementa a *Primary Key (Exporter, Importer, Product, Year)*. O campo *Year* faz parte do índice por ser obrigatória a presença da coluna temporal em qualquer

índice *UNIQUE*. Como cada *chunk* possui somente registros de um único ano, *Year* é o último elemento do índice por não possuir potencial de filtragem.

O índice *transaction_importer_product_idx* complementa o índice *transaction_pkey* por permitir indexações que contenham restrição em *Importer*, mas não em *Exporter*.

O índice *transaction_product_exporter_idx* complementa os índices anteriores por permitir indexações somente em *Product* e indexações combinadas de *Product* com *Exporter*, sem envolver *Importer*.

Com esse conjunto de índices é possível indexar qualquer subconjunto do subconjunto de atributos com capacidade filtrante da *Primary Key (Exporter, Importer, Product)*. Por essa ser uma tabela em que os registros são inseridos em lote somente uma vez por *chunk* e não são modificados, todos os índices *B-tree* foram construídos com o parâmetro *fillfactor* = 100, o que configura páginas da árvore 100% utilizadas, sem necessitar de uma reserva para maleabilidade de crescimento da árvore. Nota-se também que a indexação de agregações de *Product* é suportada, pois índices *B-tree* do PostgreSQL permitem indexação pela parte inicial de strings [3].

3.2.4 Operações

A interface desenvolvida para cálculo de medidas de complexidade conta com 12 funções criadas por meio das combinações de um prefixo e um sufixo no formato `$geography_$measure()`. Podendo `$geography` ser `contries`, `continents`, `groups` ou `common`. E `$measure` ser `eci`, `pci` ou `eci_pci`.

As funções recebem os seguintes argumentos: `(text[], start_year integer, end_year integer, hs_digit_pairs integer)`. O primeiro é um vetor que contém os países, ou nomes das agregações de acordo com o prefixo, que devem ser considerados para o cálculo da métrica. Os argumentos `start_year e end_year` definem o intervalo para o qual a métrica deve ser agregada temporalmente. Caso omitido, o argumento `end_year` recebe valor 0 e a métrica é calculada apenas para o ano `start_year`. O argumento `hs_digit_pairs` define quantos pares de dígitos do HS serão considerados, caso omitido recebe o valor 3, caso receba um valor diferente de 1, 2 ou 3 a função retorna um erro.

Especialmente para as funções com o prefixo `common` o primeiro argumento tem o tipo `cgroup[]`, sendo `cgroup` um tipo composto para representar grupos de países, definido na base de dados como `(name text, members text[])`. Desse modo é possível calcular as medidas para grupos de países definidos arbitrariamente.

A função retorna um *SET* de tuplas com o formato `(name text, value double precision)`. Especialmente para funções com o sufixo `eci_pci` o retorno consiste em uma tupla contendo dois vetores, o primeiro vetor contém tuplas correspondentes saída do ECI e o segundo tuplas referentes a saída do PCI.

[3] <https://www.postgresql.org/docs/13/indexes-types.html>

A distância entre séries é calculada com a função `euclidean_distance(country_1 text, country_2 text, start_year integer, end_year integer, hs_code varchar(6))`. Os argumentos `start_year` e `end_year` podem ser omitidos, sendo que nesse caso eles recebem o valor zero e não é definido limite inferir/superior para as séries. O argumento `hs_code` contém qualquer número de pares de dígitos de um código HS, sendo os valores de todos os produtos representados por um código com pares de dígito faltando acrescentados como dimensões no cálculo da distância. Caso omitido o argumento `hs_code` recebe uma *string* vazia, calculando a distância para as dimensões representadas por todos os produtos. A função também aceita argumentos do tipo *text[]* para receber grupos de países, nesse caso é realizado a soma das respectivas dimensões de cada grupo de países antes do cálculo da distância.

3.2.5 Implementações alternativas

Foram implementadas duas soluções alternativas para a tabela *transaction* com o intuito de avaliar a performance da solução proposta (TimescaleDB) em testes experimentais.

A primeira solução alternativa consiste em uma tabela convencional sem particionamento. Para essa solução o índice *transaction_pkey* teve a ordem dos campos alterada para *(Year, Exporter, Importer, Product)* para proporcionar indexação por *Year* e *Exporter*, ação que é realizada na *hypertable* por exclusão de *chunks*. A necessidade de indexação por *Year* e *Exporter* se deve a maneira como testes serão realizados, conforme descrito na seção 3.2.6.

A segunda solução alternativa consiste em uma tabela particionada utilizando um método nativo do PostgreSQL chamado particionamento declarativo[4]. Esse particionamento é similar ao particionamento TimescaleDB, torna a tabela uma abstração e armazena os dados em sub-tabelas chamadas partições, mas é implementado pelo PostgreSQL na declaração das tabelas, sem o uso de herança de tabelas. O particionamento foi realizado pela coluna *Year* por listagem, cada sub-tabela foi criada listando os valores de *Year* que ela deve conter, no caso o respectivo ano.

3.2.6 Testes experimentais

Os testes foram executados localmente em um sistema Debian GNU/Linux 10 (buster) com kernel 4.19.0-16-amd64 dentro de um contêiner Docker 20.10.7[5] utilizando PostgreSQL 13.2 e Timescale 2.3.0. O sistema é equipado por um processador Intel(R) Core(TM) i5-7200U CPU @ 2.50GHz de 4 núcleos e 8 GB de RAM. O objetivo dos testes realizados é avaliar a performance da solução proposta, por meio de medidas de tempo e tentativas de acesso a *cache*, em comparação com a performance das soluções alternativas. As medidas foram obtidas pelo comando *EXPLAIN*[6] com as opções *ANALYZE* e *BUFFERS*.

[4] <https://www.postgresql.org/docs/current/ddl-partitioning.html#DDL-PARTITIONING-DECLARATIVE>
[5] <https://hub.docker.com/r/timescale/timescaledb>
[6] <https://www.postgresql.org/docs/current/sql-explain.html>

Os testes foram realizados sobre um subconjunto dos dados do UN Comtrade que inclui dados distribuídos por 18 anos com, após a conversão de formato dos registros, 2.3 a 4.1 milhões de transações (tuplas) por ano, sendo a média 3.5 milhões e o total 62.7 milhões. O tamanho total dos dados armazenados é 3.99 Gb.

Foram realizados dois tipos de testes com variação de um parâmetro diferente em cada. O tipo 1 consistiu em consultas sobre 3 anos das relações variando-se a porcentagem das tuplas recuperadas. Foram realizados 10 configurações desse tipo a partir da recuperação de 10% das tuplas, incrementando-se 10% por vez até ser atingido 100%. O tipo 2 consistiu na recuperação fixa de 10% das tuplas, incrementando a quantidade de anos gradativamente. Foram utilizadas 10 configurações desse tipo de teste recuperando os dados com números de anos variando em intervalos regulares entre 1 e 18.

Para cada configuração diferente foram realizados 10 testes em cada tabela analisada. Os testes foram realizados sequencialmente com intervalos de 5 segundos antes do início da testagem de uma dada configuração para uma dada tabela e intervalos de 1 segundo entre cada teste. Pois foi observado que esses intervalos permitiam ao sistema operacional e ao SGBD estabilizarem o gerenciamento de recursos latente.

Para a recuperação de uma porcentagem fixa da tabela foi utilizada a filtragem *WHERE exporter IN ($SET_OF_COUNTRIES)*, de modo que *$SET_OF_COUNTRIES* contenha uma combinação de códigos de países referente à parcela correta da tabela. Isso resultou em um *overhead* crescente para cada configuração do primeiro tipo, mas fiel ao padrão mais comum atual de consulta a tabela. Para recuperação de 100% das tuplas essa clausula não foi inserida, o que resultou em tempos menores de busca para a última configuração do primeiro tipo.

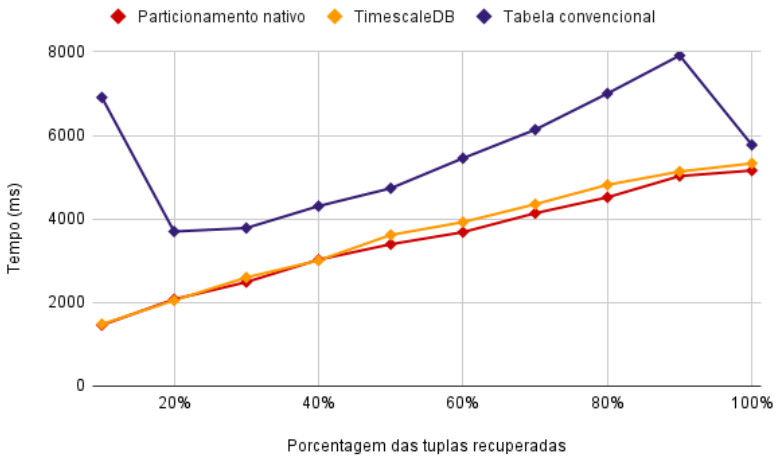
3.3 Resultados

Tabelas pertencentes aos testes e figuras relativas às medidas de cache então disponíveis no Apêndice A.

Os experimentos do tipo 1 ilustrados na Figura 6 apresentaram uma taxa de ganho de desempenho de 33.92% da *hypertable* em relação à tabela convencional, enquanto apresentaram uma taxa de 3.45% de perda de desempenho em relação à tabela com o particionamento nativo PostgreSQL. Como ilustrado na Figura 8, foi observado um alto uso de *cache* por parte das tabelas particionadas, 79.72% pela *hypertable*, 90.01% pela particionada nativamente, ao lado de só 4.58% pela tabela convencional.

Os experimentos do tipo 2 ilustrados na Figura 7 apresentaram um ganho médio de 35.70% de desempenho em relação à tabela convencional e apresentaram um ganho de 2.71% de desempenho em relação à tabela com particionamento nativo PostgreSQL. Como ilustrado na Figura 9, foi observado um alto uso de *cache* pelas tabelas particionadas para até 7 *chunks*/partições

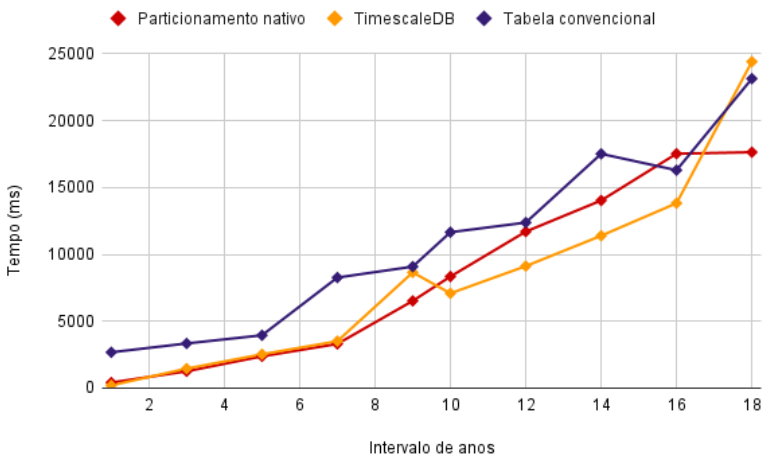
Figura 6 – Medidas de tempo dos testes tipo 1



Fonte: Elaborada pelo autor.

e quase nenhum uso para quantidades superiores, a tabela convencional quase não realizou uso de *cache*. As requisições por páginas de disco se relacionaram linearmente nas tabelas particionadas com o número de anos, enquanto foram constantes para a tabela convencional.

Figura 7 – Medidas de tempo dos testes tipo 2



Fonte: Elaborada pelo autor.

O comportamento observado do *cache* pode ser explicado pelo algoritmo usado pelo PostgreSQL[7]: *Simple Least Recently Used* (SLRU). É um algoritmo que, quando ocorre uma falha de cache, substitui a página menos recentemente acessada (PASCHOS; IOSIFIDIS; CAIRE, 2020). É um algoritmo barato de executar e que funciona bem quando existem páginas acessadas tão frequentemente que elas raramente são a última a ter sido usada e saem do *cache*. Entretanto, nos testes que executamos, as páginas são acessadas uniformemente e em uma ordem fixa. Por

[7] <https://www.postgresql.org/docs/13/monitoring-stats.html#MONITORING-PG-STAT-SLRU-VIEW>

conta disso, se o *cache* é maior que o número de páginas acessadas, a taxa de acerto é elevada (erros somente no primeiro teste da configuração), mas se o *cache* é menor que o número de páginas acessadas, a taxa de acerto se torna quase nula pela natureza cíclica de acessos retirar cada página do *cache* antes que ela seja usada novamente.

Sobre as medidas de *cache* coletadas é importante notar que elas se resumem ao uso de *cache* interno ao PostgreSQL e que grande parte dos arquivos fica carregada em memória primária por *cache* do sistema operacional.

É observado um tempo de execução bem alto na primeira configuração do teste tipo 1 para a tabela convencional (6908 ms), que se deve ao carregamento da tabela do disco, enquanto as outras duas relações carregaram somente 3 *chunks*/partições.

As consultas na tabela convencional foram realizadas através do *scan* sequencial da tabela, enquanto as consultas nas tabelas particionadas foram realizadas com a técnica *bitmap index scan*. Essa técnica consiste na leitura do índice para a criação de uma estrutura *bitmap* que contém os endereços das tuplas filtradas. Essa estrutura é ordenada de acordo com o endereço na tabela, e portanto, é utilizada para os acessos a disco serem realizados com maior ordenação física, reduzindo tempo de *seeking* de sistemas HDD.

3.4 Dificuldades e Limitações

Um dos desafios foi tratar a generalidade de consultas e usos da base de dados. Por se tratar de um banco de dados que tem como objetivo ser utilizado não só pelos projetos acontecendo atualmente no GDBI, mas também em futuros projetos que podem contar com abordagens e padrões de acesso diferentes, foi necessário durante todo o projeto atentar-se criar suporte para usos diversos.

Uma das maiores dificuldades pontuais foi quanto ao tratamento de registros de transações com códigos não especificados pela ISO 3166-1 Alpha-3. Estão presentes nos dados códigos entre XXA e XXI, reservados pela ISO para serem definidos por usuário conforme necessidade. Entre esses, os registros contendo os códigos XXA a XXC definidos na ISO/IEC 7501-1 como passaporte de pessoas sem nacionalidade e refugiados, não apresentam unicidade nos registros de transações.

Para lidar com a falta de unicidade foram tentadas estratégias como particionamento da tabela[8] e unicidade garantida por um índice parcial não incluindo esses códigos[9], que foram abandonadas respectivamente por incompatibilidade com o TimescaleDB e por anulação da utilidade do índice. Após discussão com integrantes do GBDI que utilizam os dados, foi decidido agregar os valores dos registros sem unicidade.

[8] <https://www.postgresql.org/docs/13/ddl-partitioning.html>
[9] <https://www.postgresql.org/docs/13/indexes-partial.html>

Capítulo 4

CONCLUSÃO

4.1 O projeto

O presente trabalho realizou a criação de uma base de dados voltada para o armazenamento de grandes quantidades de dados temporais otimizada com uma extensão para esse propósito em um SGBD open source amplamente utilizado e implementação de operações de interesse com granularidade flexível. O banco pode ser utilizado como estrutura base de pesquisa por pesquisadores do GBDI para temas como séries temporais e economia internacional. Os objetivos apresentados na seção 1.2 foram cumpridos. Os resultados indicam que a solução proposta, utilizando o TimescaleDB, apresenta melhor desempenho, considerando tempo de processamento de consultas, se comparada à abordagem baseadas em tabela convencionais, mas é comparável ao método de particionamento nativo do PostgreSQL.

4.2 Principais influências da graduação

Durante o desenvolvimento do trabalho foram usados diversos conhecimentos técnicos e métodos de resolução de problemas experienciados previamente ao decorrer da graduação. Entre os aprendizados que foram úteis durante o desenvolvimento do projeto destacam-se:

- Bases de Dados: forneceu a maioria dos conceitos explorados durante o projeto, como modelagem de banco de dados, familiaridade com SQL, bons hábitos na utilização de bases de dados e outras funcionalidades de um SGBD.
- Laboratório de Física I: trouxe experiência de análise de experimentos, método científico e organização de relatório científico.
- Desenvolvimento de Código Otimizado: contribuiu principalmente com uma base forte de planejamento de experimento, que foi de muita utilidade durante o projeto. Também permitiu uma visão geral das possibilidades de otimização da parte implementada em C++.
- Introdução a Ciência de Computação I, Programação Concorrente e Projeto e Desenvolvimento de Software de Sistema: foram requisitos essenciais para vários aspectos do desenvolvimento das operações implementadas em C++. Cada matéria forneceu respectivamente: a programação tendo em vista os tipos e estruturas de memória complexos da

API do PostgreSQL, o contato com programação paralelizada de diferentes formas que permitiu uma escolha bem ponderada pelo método multi-threading e a familiaridade com o funcionamento e criação de bibliotecas dinâmicas, que auxiliou o entendimento da API.

4.3 Análise da graduação do autor

Durante minha graduação eu sempre acompanhei bem as matérias e o período ideal, embora perceba agora que meu comprometimento caiu sutilmente com o passar dos anos, mesmo assim, sempre estando a altura das cobranças exigidas. Algo que marcou profundamente minha passagem pela universidade foram os grupos e eventos de extensão em que tive a oportunidade de aprender profundamente sobre temas de meu interesse, visualizar conhecimentos do curso aplicados no mundo real, conhecer pessoas que enriqueceram minha experiência acadêmica e passar por experiências de desenvolvimento de projetos, cursos, palestras, órgãos discentes e contato com a estrutura burocrática da universidade. Percebo que tenho alguns *débitos técnicos* de coisas que deveria ter aprendido, mas passei pelas disciplinas sem ter dado a merecida atenção, normalmente por necessidade de equilibrar meu tempo com outras disciplinas e atividades, mas sei que adquiri uma visão geral sólida da área da computação e vários conhecimentos aprofundados em áreas de interesse. Vendo meu percurso, fico feliz com os resultados alcançados e tenho poucos arrependimentos, como não ter feito projeto em intercâmbio e não ter feito IC em áreas nas quais descobri muito tarde meu interesse.

Os professores com quem tive aula possuíam um excelente domínio das disciplinas e, em geral, estiveram sempre solícitos à resolução de dúvidas. O nível de didática é algo que variou muito, com professores excelentes que eu passei o dia refletindo sobre o conteúdo passado em sala a professores que simplesmente não conseguia focar na aula. Em geral, os professores de computação estiveram animados em lecionar, enquanto com os de matemática isso era mais variável. Alguns professores foram marcantes na graduação por estarem sempre incentivando a sala, fazendo a matéria parecer possível de ser aprendida e sempre se certificando de que o conteúdo estava sendo realmente aprendido e não só passado. Isso foi possível através de uma boa relação e abertura de comunicação com a classe. Outro ponto que percebi em alguns professores foi o teste de técnicas de ensino e avaliação constante, tentando melhorar a cada oferecimento da disciplina, algo que foi excelente de presenciar também.

O ICMC oferece uma infraestrutura excelente para a formação de profissionais da ciência da computação, não só o espaço físico do instituto é muito bem planejado, possibilitando tanto lugares tranquilos para estudo só ou em grupo, como também propicia o contato e integração entre os estudantes dos diversos cursos, promovendo uma troca de conhecimentos e mútua ajuda nos estudos com o conhecimento complementar da visão de outros cursos de exatas, coisa que não vejo com intensidade em outros institutos. As salas de aula e os laboratórios são bem equipados e confortáveis, proporcionando um ambiente favorável para as aulas e estudo. A

biblioteca é muito bem provida de obras além de ser onde se localizam muitos dos espaços referidos anteriormente. Os funcionários das secretarias são sempre dispostos a ajudar com as questões requeridas. O que poderia ser incrementado é uma maior disponibilidade de ferramentas de uso específico das disciplinas dos cursos do instituto nos computadores, o que às vezes faz falta, e mais estabilidade nas redes wifi.

REFERÊNCIAS

ELMASRI, R.; NAVATHE, H. B. Fundamentals of database systems. **Boston: Pearson/Addison Wesley**, 2007. Citado na página 21.

HIDALGO, C. A. Economic complexity theory and applications. 2021. Disponível em: <https://www.nature.com/articles/s42254-020-00275-1>. Citado 2 vezes nas páginas 12 e 15.

MITSA, T. Temporal data mining. **CRC Press**, 2010. Citado na página 16.

PASCHOS, G. S.; IOSIFIDIS, G.; CAIRE, G. Cache optimization models and algorithms. 2020. Disponível em: <http://dx.doi.org/10.1561/0100000104>. Citado na página 28.

ANEXO A

RESULTADOS DOS TESTES

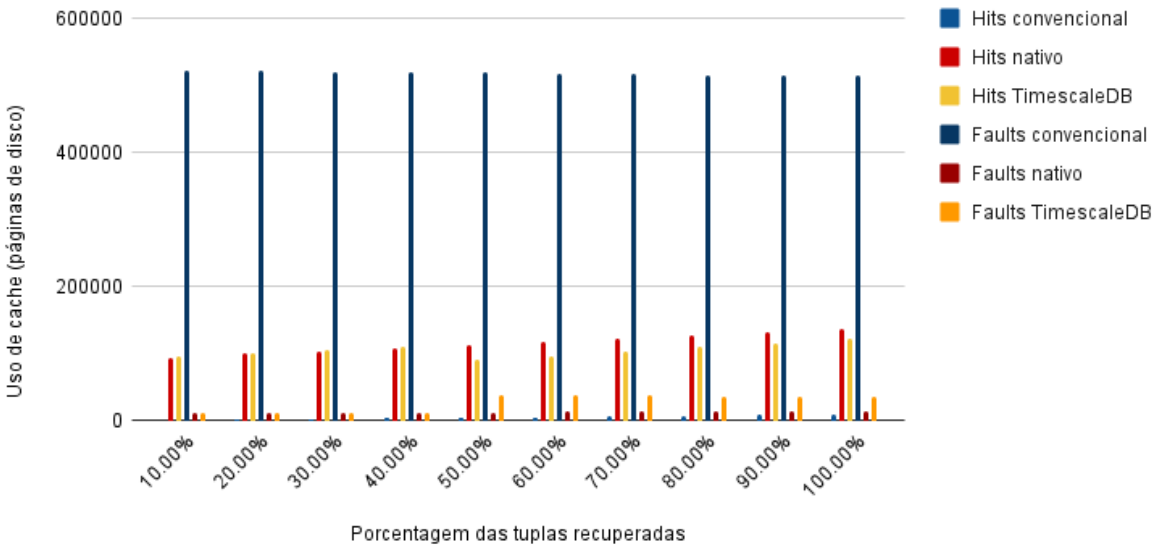
Tabela 1 – Melhora relativa da *hypertable* (TimescaleDB) nos teste de tipo 1, para tempo de processamento.

Porcentagem de tuplas	Tabela convencional	Particionamento nativo
10%	78.54%	-1.57%
20%	44.57%	1.29%
30%	31.25%	-4.51%
40%	30.18%	0.60%
50%	23.59%	-6.55%
60%	28.04%	-6.53%
70%	29.06%	-5.21%
80%	31.22%	-6.62%
90%	35.10%	-2.15%
100%	7.65%	-3.31%
Média	33.92%	-3.45%

Tabela 2 – Melhora relativa da *hypertable* (TimescaleDB) nos teste de tipo 2, para tempo de processamento.

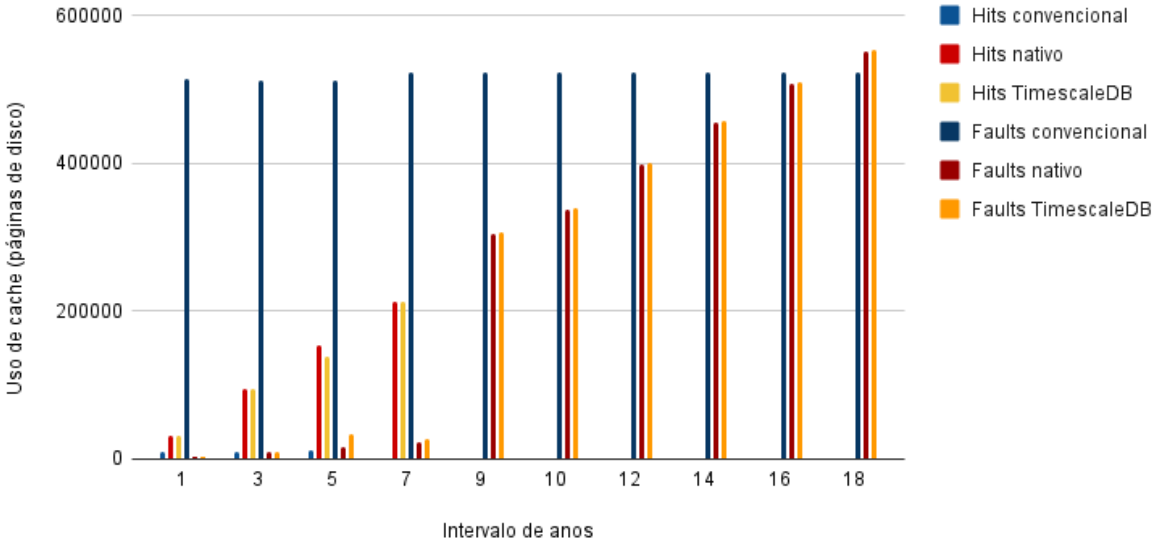
Anos	Tabela convencional	Particionamento nativo
1 ano	92.28%	50.06%
3 anos	56.32%	-16.12%
5 anos	35.92%	-6.57%
7 anos	57.57%	-6.19%
9 anos	4.74%	-32.72%
10 anos	39.22%	15.08%
12 anos	26.31%	22.06%
14 anos	34.96%	18.79%
16 anos	15.15%	21.12%
18 anos	-5.49%	-38.38%
Média	35.70%	2.71%

Figura 8 – Medidas de cache dos testes tipo 1



Fonte: Elaborada pelo autor.

Figura 9 – Medidas de cache dos testes tipo 2



Fonte: Elaborada pelo autor.