Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Otimização de Esquemas NoSQL Orientado a Documentos: Avaliação Baseada em Métricas e Algoritmo VNS

Harley Vera Olivera

Tese apresentada como requisito parcial para
conclusão do Doutorado em Informática

Orientadora
Prof.a Dr.a Maristela Terto de Holanda

Brasília
2025

Ficha Catalográfica de Teses e Dissertações

Está página existe apenas para indicar onde a ficha catalográfica gerada para dissertações de mestrado e teses de doutorado defendidas na UnB. A Biblioteca Central é responsável pela ficha, mais informações nos sítios:

http://www.bce.unb.br
http://www.bce.unb.br/elaboracao-de-fichas-catalograficas-de-teses-e-dissertacoes

Esta página não deve ser inclusa na versão final do texto.

Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Otimização de Esquemas NoSQL Orientado a Documentos: Avaliação Baseada em Métricas e Algoritmo VNS

Harley Vera Olivera

Tese apresentada como requisito parcial para
conclusão do Doutorado em Informática

Prof.a Dr.a Maristela Terto de Holanda (Orientadora)
CIC/UnB

Prof. Dr. Daniel Cardoso Moraes de Oliveira
Universidade Federal Fluminense

Prof.a Dr.a Edna Dias Canedo
Universidade de Brasília

Prof. Dr. Ronaldo dos Santos Mello
Universidade Federal de Santa Catarina

Prof. Dr. Rodrigo Bonifácio Almeida
Coordenador do Programa de Pós-graduação em Informática

Brasília, 30 de maio de 2025

Dedicatória

A Nadia, Ruth, Lia Paola, Ivanna, Segundo e Hurley, por serem as pessoas mais importantes da minha vida.

Agradecimentos

Agradeço, em primeiro lugar, à Prof.ª Dra. Maristela Terto de Holanda, cuja orientação dedicada e incentivo constante acompanharam todo o meu percurso acadêmico, desde o mestrado até a conclusão deste doutorado. Sua visão crítica, aliada a uma generosidade intelectual incomparável, não apenas orientou meu trabalho, mas também moldou minha formação como pesquisador. Nos momentos de dificuldade, seus conselhos certeiros foram fundamentais para redefinir rumos; nas conquistas, seu entusiasmo renovou minha confiança. Mais do que mentora, foi exemplo de ética, perseverança e paixão pela ciência, oferecendo oportunidades de crescimento que ultrapassaram os limites da sala de aula, seja em projetos, seminários ou interlocuções que ampliaram meu horizonte acadêmico.

Sou também profundamente grato ao corpo docente do Programa de Pós-Graduação em Informática da Universidade de Brasília (PPGI/UnB). Seus ensinamentos rigorosos, as discussões instigantes em sala de aula e a abertura para o diálogo científico foram decisivos para a construção do meu pensamento crítico e para o aprimoramento das habilidades de pesquisa que sustentam esta tese.

Por fim, agradeço aos colegas Helard Becerra, José Soncco, Pedro Garcia, Ruben Cruz, Edwin Alvarez, Gabriela Zuñiga e Gerar Quispe pela camaradagem e pelo apoio nos momentos decisivos, bem como pela constante troca de conhecimentos que tornou esta jornada mais leve e enriquecedora.

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES).

Resumo

Em bancos de dados NoSQL orientados a documentos, a definição do esquema influencia diretamente o armazenamento, a velocidade das consultas e a escalabilidade do sistema. No entanto, determinar a configuração ideal constitui um desafio significativo devido ao elevado número de combinações possíveis entre coleções e suas relações. Cada uma dessas combinações precisa ser avaliada individualmente para mensurar seu impacto sobre o desempenho geral do sistema, o que torna inviável uma análise exaustiva devido ao alto custo computacional e à complexidade envolvida. Neste trabalho, implementou-se o algoritmo metaheurístico VNS (*Variable Neighborhood Search*) para identificar soluções eficientes a partir de um conjunto de consultas e uma configuração inicial. Para tanto, definiram-se métricas de avaliação que quantificam a qualidade dos esquemas, considerando os relacionamentos referenciados e aninhados com uma ponderação específica. Integradas como função objetivo no algoritmo, essas métricas permitem uma avaliação mais ampla, concentrando-se nos aspectos estruturais dos esquemas. Operações de perturbação específicas foram projetadas para explorar eficientemente o espaço de busca, diversificando as soluções e prevenindo a convergência para mínimos locais. Assim, o algoritmo analisa diferentes estruturas, otimiza a complexidade do esquema e garante suporte completo às consultas definidas, alcançando uma solução eficaz no nível lógico de modelagem. Diferentemente dos trabalhos relacionados focados na representação conceitual e na adaptação de notações como UML e ER, este trabalho propõe uma abordagem automatizada de otimização lógica com métricas quantitativas e meta-heurísticas, eliminando a necessidade de avaliação física e facilitando o uso por projetistas com variados níveis de experiência.

Palavras-chave: Nosql, orientado a documentos, métricas de avaliação, coeficientes de ponderação, algoritmos metaheurísticos, VNS, otmização esquemas.

Abstract

In document-oriented NoSQL databases, schema definition directly impacts storage, query speed, and system scalability. However, identifying the optimal configuration poses a significant challenge due to the many possible combinations between collections and their relationships. Each of these combinations must be individually evaluated to measure their impact on overall system performance, making an exhaustive analysis infeasible due to high computational costs and inherent complexity. In this study, the metaheuristic algorithm VNS (*Variable Neighborhood Search*) was implemented to identify efficient solutions based on a set of queries and an initial configuration. To achieve this, evaluation metrics were defined to quantify schema quality, considering referenced and embedded relationships with specific weightings. Integrated as the objective function of the algorithm, these metrics enable a more comprehensive evaluation, focusing primarily on the structural aspects of schemas. Specific perturbation operations were designed to explore the search space effectively, diversify solutions, and prevent convergence to local minima. Consequently, the algorithm examines various structures, optimizes schema complexity, and ensures full support for the defined queries, thus achieving an effective solution at the logical modeling level. Unlike related works focused on conceptual representation and the adaptation of notations such as UML and ER, this study proposes an automated approach to logical optimization using quantitative metrics and metaheuristics, eliminating the need for physical evaluation and making the method accessible to designers with varying levels of experience.

Keywords: NoSQL, document-oriented, evaluation metrics, weighting coefficients, metaheuristic algorithms, VNS, schema optimization.

Sumário

1 Introdução **1**

1.1 Problema de Pesquisa 2
1.2 Justificativa 3
1.3 Contribuição Original 3
1.4 Objetivo Geral 4
1.4.1 Objetivos específicos 4
1.5 Metodologia 4
1.6 Organização do Trabalho 5

2 Estado da Arte **7**

2.1 Modelagem em Banco de Dados NoSQL 7
2.2 Coeficientes de Ponderação 12
2.3 Métricas de Avaliação 14
2.4 Geração de Esquemas 16
2.5 Considerações Finais 17

3 Visualização de Esquemas **19**

3.1 Notação Gráfica 19
3.1.1 Representação de coleções e atributos 21
3.1.2 Relacionamentos e cardinalidades 21
3.2 Considerações Finais 24

4 Coeficientes de Ponderação **26**

4.1 Contextualização 26
4.2 Ambiente Experimental 27
4.2.1 Arquitetura de Implantação 28
4.3 Banco de Dados Sintético 29
4.4 Datasets 30
4.5 Modelos de Regressão Múltipla 33
4.5.1 Construção dos Modelos 34

4.5.2 Treinamento e Validação . . . 35
4.6 Determinação dos Coeficientes de Ponderação para Relacionamentos . . . 36
4.6.1 Processo de Obtenção . . . 36
4.7 Determinação dos Coeficientes de Ponderação para Atributos Simples e Complexos . . . 38
4.8 Cenário de Validação . . . 40
4.9 Considerações Finais . . . 43

5 Métricas de Avaliação 44
5.1 Definições Formais . . . 44
5.2 Métrica Completude . . . 47
5.2.1 Funcionamento . . . 48
5.2.2 Exemplo . . . 48
5.2.3 Interpretação dos Resultados . . . 49
5.3 Métrica Padrão de Acesso . . . 50
5.3.1 Funcionamento . . . 51
5.3.2 Exemplo . . . 51
5.3.3 Interpretação dos Resultados . . . 52
5.4 Métrica Custo de Recuperação . . . 52
5.4.1 Funcionamento . . . 53
5.4.2 Exemplo . . . 53
5.4.3 Interpretação de Resultados . . . 54
5.5 Métrica Redundância . . . 55
5.5.1 Funcionamento . . . 55
5.5.2 Exemplo . . . 55
5.5.3 Interpretação dos Resultados . . . 56
5.6 Cenários de Validação . . . 56
5.6.1 Primeiro Cenário de Avaliação . . . 56
5.6.2 Segundo Cenário de Avaliação . . . 61
5.7 Considerações Finais . . . 67

6 Otimização Heurística: Determinação da Melhor Solução Encontrada 69
6.1 Algoritmo `VNS` . . . 69
6.1.1 Pseudocódigo . . . 71
6.1.2 Regras de Validação . . . 76
6.1.3 Métricas de Avaliação . . . 78
6.1.4 Estrategias de Perturbação . . . 79
6.2 Cenários de Validação . . . 86

6.2.1 Primeiro cenário 86
6.2.2 Segundo Cenário 92
6.3 Limitações 99
6.4 Ameaças 99
6.5 Considerações Finais 100

7 Conclusões **101**
7.1 Resultados Acadêmicos 102
7.2 Trabalhos Futuros 103

Lista de Figuras

2.1 Processo de mapeamento sistemático utilizado. 8
2.2 Níveis de representação por categorias NoSQL. 9

3.1 Exemplo esquema em formato JSON. 20
3.2 Notação gráfica para representar coleções. 21
3.3 Notação gráfica para representar atributos nas coleções. 22
3.4 Notação gráfica de relacionamentos por referência. 22
3.5 Notação gráfica de relacionamentos por aninhamento. 22
3.6 Notação gráfica aplicada ao *dataset* ENEM. 23
3.7 Notação gráfica aplicada a um sistema de software para revistas científicas. 24

4.1 Processo metodológico para obtenção de coeficientes de ponderação. 27
4.2 Arquitetura centralizada 28
4.3 Arquitetura distribuída. 28
4.4 Esquema com coleções referenciadas. 29
4.5 Esquema com coleções aninhadas. 30
4.6 Processo de geração dos 16 *datasets*. 31
4.7 Execução dos modelos de regressão para obter as médias $\overline{ref}$ e $\overline{emb}$. 37
4.8 Processo para obter as médias de tempo de resposta para atributos simples e complexos. 39
4.9 Processo para obter os coeficientes de ponderação no cenário de validação. 40

5.1 Esquema e_1 composto de duas coleções 45
5.2 Esquema e_1 e e_2 composto de três coleções A, B e C. 46
5.3 Conjunto de esquemas 46
5.4 Esquema e_1 com atributos. 54
5.5 Esquemas para o primeiro cenário (link para os esquemas). 57
5.6 Esquemas para o segundo cenário (link para os esquemas). 63

6.1 Diagrama de fluxo do algoritmo VNS para obtenção da solução mais ótima. 71
6.2 Arquitetura do algoritmos VNS. 72

6.3 Representação das consultas. 73
6.4 Representação gráfica e matricial do esquema e_1. 74
6.5 Representação gráfica e matricial do esquema e_2 75
6.6 Representação gráfica e matricial do esquema e_3 75
6.7 Representação da solução inicial. 76
6.8 Error referenciando uma coleção aninhada. 77
6.9 Solução $\mathtt{E}_6$ composta de três esquemas 87
6.10 Consultas e esquemas referentes à solução inicial do cenário 1, apresentados em formato codificado. 88
6.11 Antes e depois da solução otimizada. 89
6.12 Comparação da solução antes e depois da otimização. 90
6.13 Diagramas de convergência das soluções do cenário 1. 91
6.14 Diagramas de convergência das soluções do cenário 1. 92
6.15 Solução $\mathtt{E}_1$ do cenário de validação 2. 94
6.16 Representação matricial dos quatro esquemas da solução $\mathtt{E}_1$. 94
6.17 Consultas e esquemas referentes à solução inicial do cenário 2, apresentados em formato codificado. 95
6.18 Antes e depois da solução otimizada cenário 2. 96
6.19 Comparação visual da solução antes e depois da otimização cenário 2. . . . 96
6.20 Diagramas de convergência das soluções do cenário 2. 98

Lista de Tabelas

2.1 Modelos e formatos usados por categoria e níveis de representação. 9
2.2 Trabalhos acadêmicos detalhados pelos modelos utilizados nos níveis de representação. 10
2.3 Estudos selecionados sobre modelagem de dados em bancos `NoSQL`, por contexto. 12
2.4 Visão geral das propostas existentes para otimizar esquemas. 18

4.1 Equipamentos computacionais físicos e na nuvem utilizados. 27
4.2 Distribuição das arquiteturas implementadas. 29
4.3 Estrutura do *dataset* gerado. 33
4.4 Modelos de regressão múltipla obtidos no Eq_1 para arquitetura centralizada. 35
4.5 Modelos de regressão múltipla obtidos no Eq_1 para arquitetura distribuída. 35
4.6 Modelos de regressão múltipla obtidos no Eq_2 para arquitetura distribuída. 36
4.7 Modelos de regressão múltipla obtidos no Eq_3 para arquitetura distribuída. 36
4.8 Coeficientes de ponderação em modo centralizado no equipamento Eq_1. . . 37
4.9 Coeficientes de ponderação em modo distribuído no equipamento Eq_1. . . . 38
4.10 Coeficientes de ponderação em modo distribuído no equipamento Eq_2. . . . 38
4.11 Coeficientes de ponderação em modo distribuído no equipamento Eq_3. . . . 38
4.12 Coeficientes de ponderação obtidos nos modos centralizado e distribuído. . 40
4.13 Coeficientes de ponderação para o caso de validação no modo centralizado no equipamento Eq_1. 41
4.14 Coeficientes de ponderação para o caso de validação no modo distribuído no equipamento Eq_1. 41
4.15 Coeficientes de ponderação para o caso de validação no modo distribuído no equipamento Eq_2. 41
4.16 Coeficientes de ponderação para o caso de validação no modo distribuído no equipamento Eq_3. 42
4.17 Comparação de coeficientes do primeiro experimento com o caso de validação. 42

5.1 Consultas definidas envolvendo quatro coleções. 45

5.2 Consulta definida envolvendo duas coleções. 45
5.3 Consultas definidas envolvendo três coleções. 46
5.4 Verificação da existência das coleções de q_1 nos esquemas e_1 e e_2. 49
5.5 Verificação da existência de relacionamentos que atendam as coleções de q_1 nos esquemas e_1 e e_2. 49
5.6 Verificação da existência de relacionamentos que atendam as coleções de q_1 nos esquemas e_1 e e_2. 49
5.7 Tabela de análise padrão de acesso. 51
5.8 Tabela de análise padrão de acesso. 53
5.9 Tabela de análise função $contarAtr()$. 54
5.10 Tabela de análise de redundância. 56
5.11 Consultas definidas para o primeiro cenário. 57
5.12 Análise de $existeColecao(\mathtt{E}, q_i)$ e $existeRelacionamento(\mathtt{E}, q_i)$ para cada consulta nos nove esquemas. 58
5.13 Resultados da métrica completude para as nove soluções. 58
5.14 Análise de *padrão de acesso* primeiro cenário. 59
5.15 Análise de *custo de recuperação* primeiro cenário. 60
5.16 Análise de coleções repetidas em cada esquema. 60
5.17 Resultados final das métricas no primeiro cenário. 61
5.18 Consultas definidas para o segundo cenário. 64
5.19 Análise de $existeColecao(\mathtt{E}, q_i)$ e $existeRelacionamento(\mathtt{E}, q_i)$ para cada consulta nos quatro esquemas. 64
5.20 Resultados da métrica completude para o segundo cenário. 64
5.21 Análise do *padrão de acesso* no segundo cenário. 65
5.22 Análise de *custo de recuperação* segundo cenário. 66
5.23 Análise da *redundância*. 66
5.24 Resultados final das métricas no segundo cenário. 67

6.1 Operações de perturbação usadas na atividade *perturbar solução*. 71
6.2 Consultas definidas para o primeiro cenário. 87
6.3 Comparação das métricas entre a solução original e a otimizada. 89
6.4 Comparação das métricas entre as soluções avaliadas. 90
6.5 Consultas definidas para o segundo cenário. 93
6.6 Comparação das métricas entre a solução original e a otimizada no cenário 2. 97
6.7 Comparação das métricas entre as soluções avaliadas no cenário 2. 97

Capítulo 1

Introdução

A modelagem de banco de dados constitui uma das etapas fundamentais no desenvolvimento de sistemas de informação, pois permite compreender e abstrair a complexidade inerente ao domínio do problema (Simsion & Witt, 2004). No contexto de bancos de dados relacionais, o processo de modelagem é realizado em três níveis: conceitual, lógico e físico, utilizando-se linguagens de modelagem como Entidade-Relacionamento (ER) ou *Unified Modeling Language* (UML), de acordo com o nível de abstração requerido (Elmasri, 2008). No nível conceitual, não se exige conhecimento técnico sobre o sistema gerenciador de banco de dados (SGBD), uma vez que o foco está na representação semântica dos dados. No nível lógico, é necessário um conhecimento intermediário, dado que envolve a estruturação dos dados conforme o modelo relacional. Já no nível físico, a modelagem requer um conhecimento do SGBD específico, sendo recomendável experiência prévia na administração e otimização de bancos de dados.

Em bancos de dados NoSQL, ainda não há padrões amplamente estabelecidos para a modelagem; contudo, diversos contextos foram propostos, abrangendo os níveis conceitual, lógico e físico e utilizando-se ferramentas como ER, UML, *Extensible Markup Language* (XML), *XML Metadata Interchange* (XMI), *JavaScript Object Notation* (JSON), *Web Ontology Language* (OWL) e *Formal Concept Analysis* (FCA) (Vera-Olivera & et. al., 2021). Além disso, na modelagem de esquemas orientados a documentos, devem ser consideradas novas características, tais como a definição da quantidade de esquemas necessários conforme as consultas propostas; a escolha entre relacionamentos referenciados ou aninhados; e a seleção de atributos simples ou compostos. Essas decisões impactam diretamente o desempenho do banco de dados (Gómez et al., 2016; Reis et al., 2018; Vera-Olivera et al., 2023). Ademais, a experiência do projetista desempenha também papel determinante na otimização dessas escolhas técnicas.

No entanto, determinar a melhor solução de esquemas para um problema específico em bancos de dados NoSQL não é trivial, dada a multiplicidade de possibilidades decorrentes

do número de coleções e das relações entre elas, tais como referenciadas, aninhadas ou independentes (Vera-Olivera & Holanda, 2024). Para avaliar a eficiência dos esquemas propostos, é necessário verificar o atendimento às consultas predefinidas e analisar sua estrutura por meio de operações `CRUD` (*Create, Read, Update, Delete*), que permitem identificar os impactos no desempenho operacional (Győrödi et al., 2022) (Llano-Ríos et al., 2020) (Andor et al., 2019) (Carvalho et al., 2023).

A otimização de esquemas em bancos de dados NoSQL ocorre em diferentes contextos, como transformação, migração, geração de esquemas e otimização multicritério (Hewasinghage et al., 2023; Vera-Olivera & et. al., 2021). Na transformação, esquemas são convertidos de modelos conceituais (`ER` ou `UML`) para lógicos ou físicos, com validação por operações `CRUD`, embora a exploração do espaço de busca seja limitada, comprometendo a eficiência (Bansal et al., 2023; Muhammad & Azizah, 2022; Saha & Sachdeva, 2024). Na migração, a otimização adapta dados a novas cargas de trabalho, utilizando técnicas como programação linear inteira ou abstração temporal para reduzir custos, mas também sofre com restrições na exploração do espaço de busca (Wakuta et al., 2023).

Na geração de esquemas, métodos automatizados, como *Document Database Schema Recommender* (DBSR) (Reniers et al., 2020) e *Dynamic Schema Proposition* (DSP) (A. A. Imam et al., 2020), processam modelos de dados e consultas, aplicando heurísticas para criar esquemas otimizados com menor redundância e maior eficiência. Já na otimização multicritério, algoritmos heurísticos, como os propostos por (Hewasinghage et al., 2023), utilizam modelos canônicos e buscas locais guiadas por funções de perda, mas enfrentam limitações devido à dependência de transformações fixas, avaliação em níveis lógico e físico que exige conhecimento técnico, e falta de exploração dinâmica do espaço de busca, restringindo a diversidade das soluções.

1.1 Problema de Pesquisa

O problema de pesquisa concentra-se na otimização de esquemas em bancos de dados NoSQL orientados a documentos, visando atender às consultas definidas, ao mesmo tempo em que se minimizam o padrão de acesso e a redundância. Essa otimização considera as configurações possíveis exclusivamente no nível lógico, sem a necessidade de avaliação por operações `CRUD` ou de conhecimentos técnicos relacionados ao nível físico.

O problema é agravado pelo crescimento não linear do espaço de busca, que se expande com o aumento de coleções e seus relacionamentos. Em cenários práticos, métodos tradicionais tornam-se inviáveis, destacando a vantagem de algoritmos meta-heurísticos (Naka & Guliashki, 2021). Para a utilização de algoritmos meta-heurísticos, três pilares são essenciais (Mladenović & Hansen, 1997):

- Representação da solução (esquemas e consultas);
- Função objetivo (Métricas de avaliação);
- Operadores *shaking*

1.2 Justificativa

A relevância deste estudo é amplificada pela migração de banco de dados para a nuvem, que evidenciou ineficiências decorrentes de esquemas mal estruturados, como redundâncias, alta latência e custos operacionais elevados (Abadi et al., 2022). A abordagem proposta enfrenta esses desafios ao operar exclusivamente no nível lógico, utilizando métricas de avaliação e operações de perturbação para otimização. A adoção de métricas de avaliação permite quantificar objetivamente aspectos críticos do esquema, como redundância e padrão de acesso, viabilizando comparações entre diferentes alternativas de modelagem. Já as operações de perturbação são fundamentais para explorar múltiplas regiões do espaço de soluções, evitando estagnações em mínimos locais e promovendo a descoberta de esquemas mais eficientes. Essa estratégia elimina a necessidade de operações `CRUD` no nível físico e reduz a dependência de conhecimentos especializados em `SGBD`, democratizando o acesso a projetistas com diferentes níveis de experiência.

A ausência de padrões consolidados em `NoSQL` exige soluções que integrem rigor teórico e exploração algorítmica. Ao focar no nível lógico, este trabalho oferece um método replicável e acessível, apto a gerar esquemas otimizados mesmo em contextos dinâmicos, onde cargas de trabalho e requisitos evoluem constantemente. Essa flexibilidade é crítica em ambientes de nuvem, onde a escalabilidade e a eficiência operacional são prioritárias.

1.3 Contribuição Original

Este trabalho apresenta como contribuição original o desenvolvimento de um algoritmo meta-heurístico baseado em `VNS` que realiza a otimização de esquemas em bancos de dados `NoSQL` orientados a documentos exclusivamente no nível lógico, eliminando a necessidade de avaliação por experimentos de leitura através de *benchmarks* e reduzindo a dependência de conhecimento especializado no nível físico. O diferencial desta proposta reside na integração de métricas quantitativas e coeficientes de ponderação. Neste contexto, os coeficientes de ponderação são valores atribuídos aos diferentes tipos de relacionamentos entre coleções — referenciados e aninhados — com o objetivo de quantificar seu impacto no desempenho do esquema. Esses coeficientes influenciam diretamente no cálculo das mé-

tricas, permitindo ajustar a importância relativa de cada tipo de relacionamento durante a busca por configurações mais eficientes.

1.4 Objetivo Geral

O objetivo geral desta tese de doutorado é desenvolver um algoritmo meta-heurístico para otimizar esquemas em bancos de dados `NoSQL` orientados a documentos, garantindo o atendimento às consultas definidas, com minimização do padrão de acesso e da redundância, por meio de uma abordagem no nível lógico. O algoritmo deverá incorporar métricas de avaliação que permitam quantificar objetivamente a qualidade dos esquemas e utilizar operações de perturbação para explorar eficientemente o espaço de busca e evitar mínimos locais. Este algoritmo deve explorar de forma eficiente o espaço de busca, sem depender de operações `CRUD` nem de conhecimento técnico avançado do nível físico do SGBD.

1.4.1 Objetivos específicos

Os objetivos específicos definidos são:

- Estabelecer coeficientes de ponderação que atribuam valores numéricos representativos aos tipos de relacionamentos referenciados e aninhados de modo a quantificar o custo relativo de cada tipo de relação;
- Desenvolver métricas quantitativas para avaliar a qualidade dos esquemas no nível lógico;
- Adaptar e implementar o algoritmo de busca heurística para otimizar esquemas, integrando as métricas de avaliação, coeficientes de ponderação e operadores de perturbação

1.5 Metodologia

A metodologia deste trabalho foi estruturada em cinco etapas principais, descritas a seguir:

- **Mapeamento Sistemático da Literatura**: Realizou-se um mapeamento sistemático com base nas diretrizes de (Petersen et al., 2015) e (Kitchenham et al., 2010) para revisar o estado da arte. Foram investigados quatro temas centrais: modelagem em bancos de dados `NoSQL`, coeficientes de ponderação, métricas de avaliação e otimização heurística. Esta etapa permitiu identificar lacunas e fundamentar as etapas subsequentes;

- **Obtenção de Coeficientes de Ponderação**: Foram gerados bancos de dados sintéticos para representar o comportamento de coleções relacionadas por referência e por aninhamento. A partir desses bancos, criou-se *datasets* com os tempos de recuperação de consultas para coleções referenciadas e aninhadas. Esses *datasets* foram utilizados para treinar modelos de regressão múltipla, que generalizaram o comportamento das respostas às consultas. Os modelos resultantes foram empregados para derivar os coeficientes de ponderação;

- **Proposta de Métricas de Avaliação**: Foram desenvolvidas métricas de avaliação para quantificar o desempenho dos esquemas em relação às consultas e à estrutura dos esquemas. Nesta etapa, os coeficientes de ponderação foram utilizados para orientar a escolha entre relacionamentos referenciados e aninhados, considerando o impacto no desempenho;

- **Otimização com Algoritmo *Variable Neighborhood Search* (`VNS`)**: Utilizou-se o algoritmo `VNS` para otimizar um conjunto de esquemas, integrando os coeficientes de ponderação e as métricas de avaliação previamente definidas. O objetivo foi atender completamente às consultas, minimizando métricas relacionadas ao padrão de acesso e à redundância nos esquemas;

- **Validação em Cenários**: Por fim, foram definidos dois cenários de validação para demonstrar a eficácia do algoritmo proposto. Esses cenários permitiram avaliar o desempenho dos esquemas otimizados em condições práticas, verificando a aplicabilidade e a robustez da abordagem

1.6 Organização do Trabalho

Este trabalho está organizado nos seguintes capítulos, cada um abordando aspectos fundamentais da pesquisa realizada.

No Capítulo 2, é apresentado o estado da arte sobre bancos de dados NoSQL orientados a documentos. São discutidos a modelagem de esquemas, a avaliação de coeficientes de ponderação, o uso de métricas estruturais para avaliação de esquemas e as abordagens existentes para a geração automática de esquemas otimizados, destacando as limitações e lacunas na literatura atual.

No Capítulo 3, é apresentada uma notação gráfica específica para a visualização de esquemas em bancos de dados NoSQL. A notação diferencia coleções, atributos simples e complexos, além de representar relacionamentos referenciados e aninhados, permitindo melhor compreensão das estruturas e das cardinalidades envolvidas.

No Capítulo 4, é descrito o processo de determinação dos coeficientes de ponderação para relacionamentos referenciados e aninhados. Utilizando um ambiente experimental com arquiteturas centralizadas e distribuídas, são construídos modelos de regressão múltipla a partir de bancos de dados sintéticos, permitindo quantificar o impacto estrutural dos diferentes tipos de relacionamento.

No Capítulo 5, são apresentadas quatro métricas desenvolvidas para a avaliação de esquemas: completude, padrão de acesso, custo de recuperação e redundância. Essas métricas foram integradas à avaliação no nível lógico, dispensando a necessidade de operações CRUD, e foram validadas em cenários experimentais.

No Capítulo 6, é detalhada a implementação do algoritmo `VNS` para a otimização de esquemas. O capítulo inclui o pseudocódigo, as regras de validação, as estratégias de perturbação aplicadas e a validação experimental em dois cenários distintos, comprovando a eficácia da abordagem para encontrar soluções otimizadas.

Finalmente, no Capítulo 7, são apresentadas as conclusões da pesquisa, abordando o cumprimento dos objetivos propostos, as contribuições obtidas, as limitações identificadas e as direções sugeridas para trabalhos futuros na otimização de esquemas em bancos de dados NoSQL orientados a documentos.

Capítulo 2

Estado da Arte

Este capítulo apresenta o estado da arte da pesquisa sobre bancos de dados NoSQL orientados a documentos, organizado em quatro seções complementares. A primeira seção realiza uma análise abrangente de modelagem por meio de um mapeamento sistemático. Em seguida na segunda seção, discute-se a avaliação de coeficientes de ponderação para relações e atributos, comparando abordagens de documentos referenciados e aninhados em termos de desempenho. A terceira seção analisa estudos dedicados à avaliação de esquemas, empregando métricas que mensuram a complexidade em diferentes níveis de representação. Por fim, a quarta seção examina estratégias de otimização de esquemas.

2.1 Modelagem em Banco de Dados NoSQL

A modelagem de dados desempenha um papel fundamental no desenvolvimento de bancos de dados, e sua aplicação em bancos de dados `NoSQL` ainda carece de um padrão consolidado. Com essa finalidade foi realizada uma revisão sistemática, baseada nos protocolos de (Petersen et al., 2015) e (Kitchenham et al., 2010), para responder a três questões principais de pesquisa: (1) Quais níveis de representação são utilizados na modelagem de bancos de dados `NoSQL`? (2) Quais modelos são empregados em cada nível de representação? (3) Em quais contextos ocorre o processo de modelagem? A metodologia adotada, ilustrada na Figura 2.1, organiza o processo de mapeamento em duas etapas: planejamento e execução. No planejamento, são definidos o escopo, os objetivos, as questões de pesquisa, fontes de busca, estratégias de pesquisa, e os critérios de inclusão e exclusão. Durante a execução, a busca de literatura é realizada, os estudos relevantes são selecionados de acordo com os critérios, as informações são extraídas e classificadas, a análise e síntese dos resultados são feitas. Para operacionalizar o protocolo, foi formulada uma estratégia de busca estruturada com base nas questões de pesquisa, incluindo termos como "*model*", "*NoSQL database*", "*document*", "*graph*", "*column*" e "*key-value*", complemen-

tada por uma análise semântica com a ferramenta `VOSviewer`. A *string* de busca utilizada foi:

((design OR model OR "data structure" OR "relational data") AND nosql) OR (modeling AND (Key-value OR column OR document OR graph) AND (nosql OR database OR schema)

Essa *string* foi ajustada conforme as restrições das bases *Scopus* e *Web of Science*, selecionadas por sua abrangência em ciência da computação. Os critérios de inclusão foram: IC-1) trabalhos acadêmicos que tratassem diretamente da modelagem de dados em bancos NoSQL como tema principal; IC-2) artigos completos publicados em conferências ou periódicos. Por outro lado, os critérios de exclusão compreenderam: EC-1) artigos não disponíveis online; EC-2) trabalhos não redigidos em inglês; EC-3) artigos curtos ou resumos estendidos; EC-4) revisões de literatura como surveys ou mapeamentos sistemáticos; EC-5) estudos fora das áreas de computação, engenharia ou sistemas de informação; EC-6) publicações sem revisão por pares; EC-7) trabalhos publicados fora do intervalo entre 2008 e 2019. A aplicação rigorosa desse protocolo permitiu a seleção de 54 estudos primários.

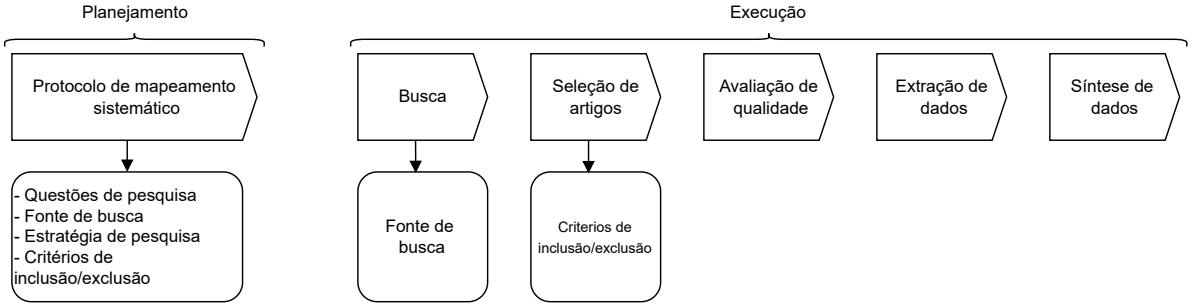


Figura 2.1: Processo de mapeamento sistemático utilizado.

A análise dos 54 artigos primários selecionados revelou que, em relação aos níveis de representação, os níveis conceitual e lógico receberam mais atenção do que o nível físico. A Figura 2.2 apresenta os resultados detalhados por categoria de banco de dados NoSQL.

No que se refere à segunda questão de pesquisa, verificou-se que os modelos `UML` e `ER`, bem como notações adaptadas derivadas desses paradigmas, foram amplamente empregados na modelagem de bancos de dados `NoSQL`. Além disso, formatos como `JSON`, `XML` e `XMI` foram utilizados para a definição de esquemas nos três níveis de representação. A Tabela 2.1 sintetiza os resultados obtidos.

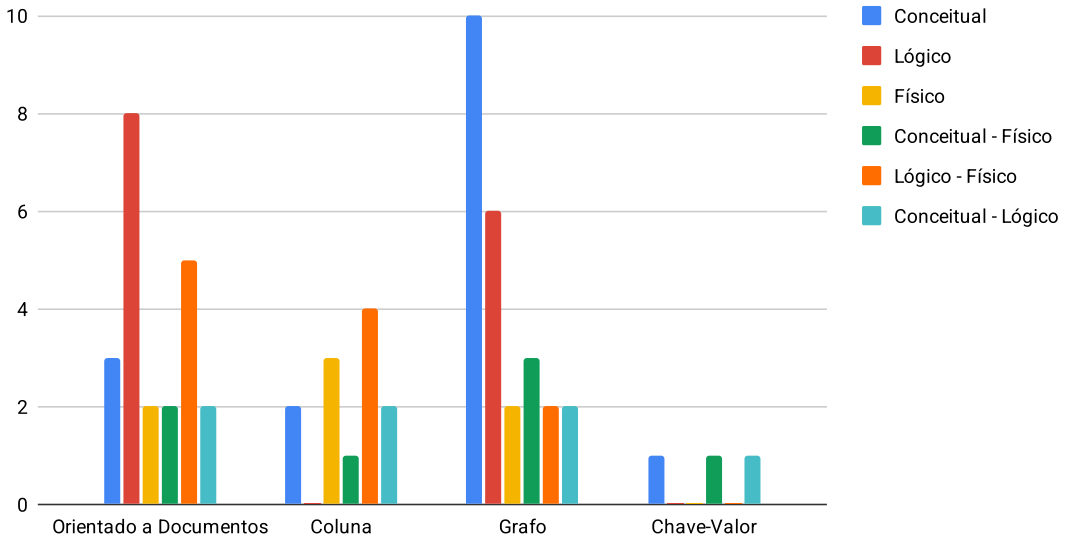


Figura 2.2: Níveis de representação por categorias NoSQL.

Tabela 2.1: Modelos e formatos usados por categoria e níveis de representação.

Tipo banco de dados	Conceitual	Logico	físico
Orientado a documentos	Novas notações baseadas em ER e UML, UML, OWL, FCA	JSON, XMI, Novas notações baseadas em UML, Modelo relacional	JSON, modelo generico, modelo MongoDB
Grafo	EER, UML, modelo generico, RDF, O-ER, Novas notações baseadas em ER, ER, OWL	JSON, modelo generico, XMI, Novas notações baseadas em ER, EER, and UML	JSON, XML, modelo Neo4j, modelo generico, modelo OrientDB
Chave-valor	Novas notações baseadas em ER, OWL, UML	JSON	modelo Oracle NoSQL
Coluna	UML, Novas notações baseadas em ER, OWL	JSON, Novas notações baseadas em UML, modelo generico, XML, modelo relacional, XMI	JSON, modelo generico, XML, modelo Cassandra, modelo HBase, modelo Amazon DynamoDB

A Tabela 2.2 apresenta uma visão geral dos artigos mapeados junto com o tipo de banco de dados NoSQL e os modelos usados nos níveis conceitual, lógico e físico.

Para a terceira questão de pesquisa, foram identificados novos contextos de modelagem, incluindo avaliação de desempenho, processos de migração e geração de esquemas. Adicionalmente, novas características relevantes para a modelagem de bancos de dados NoSQL foram evidenciadas, como a quantidade de registros por entidade, operações CRUD e requisitos do sistema (disponibilidade, consistência e escalabilidade). A Tabela 2.3 apresenta os artigos classificados nos diferentes contextos identificados. O *benchmark* é utilizado para avaliar o desempenho de diferentes abordagens de modelagem sob cargas específicas, enquanto a avaliação abrange análises qualitativas e quantitativas que consideram critérios como expressividade e manutenibilidade. As diretrizes de projeto fornecem recomendações práticas para orientar decisões de modelagem, especialmente em relação à desnormalização e à escolha entre referências e aninhamento. O contexto de migração refere-se à transição de modelos e dados entre bancos relacionais e bancos NoSQL. Essa

Tabela 2.2: Trabalhos acadêmicos detalhados pelos modelos utilizados nos níveis de representação.

Artigo	**Tipo NoSQL**	**Conceitual**	**Lógico**	**Físico**
(A. Imam et al., 2019)	Documentos	—	—	JSON
(Akintoye et al., 2019)	Documentos, Grafo	—	—	JSON
(Martins de Sousa & del Val Cura, 2018)	Grafo	ER modificado	ER modificado	—
(Vágner, 2018)	Grafo	EER	—	Neo4j
(Abdelhedi, Ait Brahim, & Zurfluh, 2018)	Coluna	—	—	Cassandra/HBase
(Nogueira et al., 2018)	Documentos	—	—	JSON
(Hamouda & Zainol, 2018)	Documentos	—	UML modificado	—
(Abdelhedi, Brahim, et al., 2018)	Documentos, Grafo, Coluna	—	Genérico	Genérico
(A. A. Imam & Basri, 2018)	Documentos	—	UML modificado	—
(Suárez-Otero et al., 2018)	Coluna	ERD	UML modificado	—
(Van Erven et al., 2018)	Grafo	UML	—	—
(Angles, 2018)	Grafo	Genérico	—	—
(Chiș-Rațiu & Buchmann, 2018)	Grafo	RDF	—	—
(Villa et al., 2018)	Grafo	O-ER	—	—
(A. Imam et al., 2017)	Documentos	UML modificado	—	—
(Roy-Hubara et al., 2017)	Grafo	UML	—	—
(Zhang, 2017)	Grafo	Nova notação	—	—
(M. J. Mior et al., 2017)	Coluna	UML	—	Cassandra
(Banerjee & Sarkar, 2017)	Documentos, Grafo, Coluna, KV	ER modificado	JSON	—
(Shin et al., 2017)	Documentos	UML	UML modificado	—
(Chillón et al., 2017)	—	UML	—	—
(Orel et al., 2017)	Grafo	—	—	Neo4j
(Pokorný, 2016)	Grafo	ER	—	—
(Bermbach et al., 2016)	Coluna	ER	—	—
(Lima & Mello, 2016)	Documentos	—	UML modificado	—
(Banerjee & Sarkar, 2016)	Documentos, Grafo, Coluna, KV	OWL	—	—
(Abdelhedi et al., 2016)	Coluna	—	XML	—
(Varga et al., 2016)	Documentos	FCA	—	—
(M. Zhao et al., 2016)	Grafo	Nova notação	—	—
(De Lima & Dos Santos Mello, 2015)	Documentos	—	UML modificado	—
(Vera & et. al., 2015)	Documentos	UML modificado	—	—
(X. Li et al., 2014)	—	—	UML modificado	JSON
(M. Mior, 2014)	Coluna	—	—	Cassandra/HBase
(Y. Li et al., 2014)	Coluna	—	—	XML
(Sedlmeier & Gogolla, 2014)	Grafo	—	UML modificado	—
(Yoo et al., 2014)	Grafo	Nova notação	—	—
(G. Zhao et al., 2013)	Documentos, Grafo	—	Relacional	—
(Kaur & Rani, 2013)	Documentos	—	UML modificado	—
(Vajk et al., 2013)	Coluna	—	—	DynamoDB
(Schram & Anderson, 2012)	Coluna	—	OWN	—
(Santos & Costa, 2016)	Coluna	—	Relacional	—
(Hewasinghage et al., 2018)	Documentos	—	HeRM	—
(Santisteban & Ticona-Herrera, 2018)	Grafo	—	ER modificado	—
(De Virgilio et al., 2014)	Grafo	O-ER	—	—
(Daniel et al., 2016)	Grafo	—	—	Neo4j/OrientDB
(Abdelhedi, Brahim, et al., 2017)	Documentos, Grafo, Coluna	—	UML	MongoDB/Neo4j/Cassandra
(Abdelhedi, Ait Brahim, et al., 2017)	Documentos, Grafo, Coluna	—	XMI	MongoDB/Neo4j/Cassandra
(A. Imam et al., 2018)	Documentos	—	—	MongoDB
(Shoval, 2018)	Grafo	—	UML	—
(Roy-Hubara et al., 2018)	Grafo	—	UML	—
(Reniers et al., 2018)	Documentos	—	Nova notação	—
(la Vega et al., 2018)	Documentos, Coluna	—	UML	JSON
(Varga et al., 2018)	Grafo	ERD	—	XML
(Bugiotti et al., 2014)	Chave-valor	UML	—	Oracle NoSQL

transição exige a adaptação de esquemas, consultas e operações para preservar a semântica e a integridade dos dados. O contexto de ontologias refere-se ao uso de representações semânticas, como `RDF` e `OWL`, para modelar dados em bancos NoSQL de forma conceitual e interoperável. A transformação de esquemas refere-se à conversão de modelos de dados entre diferentes níveis de abstração ou entre diferentes formatos de representação como `ER`. O contexto orientado a consultas foca na modelagem de dados com base nos padrões de acesso e nas operações de leitura e escrita mais frequentes. O contexto de geração de esquemas refere-se à criação automática ou assistida de estruturas de dados (como `JSON`, `XML` ou `XMI`) a partir de modelos conceituais ou lógicos.

Tabela 2.3: Estudos selecionados sobre modelagem de dados em bancos NoSQL, por contexto.

Contexto	**Documentos**
Benchmark	(Nogueira et al., 2018)
Avaliação	(Roy-Hubara et al., 2018)
Diretrizes	(Akintoye et al., 2019), (Angles, 2018), (Banerjee & Sarkar, 2016), (Bermbach et al., 2016), (Bugiotti et al., 2014), (Chillón et al., 2017), (Chiș-Rațiu & Buchmann, 2018), (De Virgilio et al., 2014), (Hewasinghage et al., 2018), (A. Imam et al., 2017), (A. A. Imam & Basri, 2018), (Kaur & Rani, 2013), (Pokorný, 2016), (Santisteban & Ticona-Herrera, 2018), (Shin et al., 2017), (Suárez-Otero et al., 2018), (Van Erven et al., 2018), (Varga et al., 2016), (Varga et al., 2018), (Vera & et. al., 2015), (Vágner, 2018), (G. Zhao et al., 2013), (M. Zhao et al., 2016)
Migração	(Hamouda & Zainol, 2018)
Ontologia	(Banerjee & Sarkar, 2016)
Transformação de Esquemas	(Abdelhedi, Ait Brahim, et al., 2017), (Abdelhedi, Ait Brahim, & Zurfluh, 2018), (Abdelhedi et al., 2016), (Abdelhedi, Brahim, et al., 2017), (Abdelhedi, Brahim, et al., 2018), (Banerjee & Sarkar, 2017), (Daniel et al., 2016), (la Vega et al., 2018), (De Lima & Dos Santos Mello, 2015), (Y. Li et al., 2014), (Lima & Mello, 2016), (Martins de Sousa & del Val Cura, 2018), (M. Mior, 2014), (Orel et al., 2017), (Reniers et al., 2018), (Roy-Hubara et al., 2017), (Santos & Costa, 2016), (Schram & Anderson, 2012), (Shoval, 2018), (Vajk et al., 2013), (Villa et al., 2018), (Yoo et al., 2014), (Zhang, 2017)
Orientado a Consultas	(X. Li et al., 2014)
Geração de Esquemas	(A. Imam et al., 2019), (A. Imam et al., 2018), (M. J. Mior et al., 2017)

2.2 Coeficientes de Ponderação

Os coeficientes de ponderação são valores numéricos atribuídos a diferentes elementos de um sistema, com o objetivo de refletir seu impacto sobre um determinado resultado. No contexto da modelagem e otimização de esquemas em bancos de dados NoSQL orientados a documentos, os coeficientes de ponderação permitem quantificar de forma diferenciada os custos e benefícios associados a diversas configurações de esquemas, como o uso de relacionamentos referenciados ou aninhados.

Seguindo a mesma metodologia empregada na seção anterior, a análise dos trabalhos relacionados aos coeficientes de ponderação foi conduzida com base no protocolo de mapeamento sistemático da literatura descrito por Petersen et al. (2015) e Kitchenham et al. (2010), conforme ilustrado na Figura 2.1. Esse processo foi estruturado em duas etapas principais: planejamento e execução.

Na fase de planejamento, foram definidas as seguintes questões de pesquisa: (1) Quais estudos abordam diretamente ou indiretamente o uso de coeficientes de ponderação na modelagem de relacionamentos entre coleções em bancos de dados NoSQL? (2) Quais métricas ou abordagens quantitativas são utilizadas para comparar o desempenho entre modelos de dados referenciados e aninhados? (3) Em quais contextos metodológicos (avaliação de desempenho, transformação de esquemas, análise estrutural, etc.) ocorrem essas comparações?

Com base nessas perguntas, elaborou-se uma cadeia de busca combinando os termos "*NoSQL*", "*document database*", "*embedding*", "*referencing*", "*nested model*", "*performance*", "*join*", "*query optimization*", "*data modeling*" e "*weight coefficient*". A expressão genérica utilizada foi:

((NoSQL AND "document database" AND embedding OR referencing OR "nested model")) AND (performance OR join OR "weight coefficient" OR "data modeling")

Esse protocolo permitiu identificar uma lacuna significativa na literatura sobre comparação quantitativa entre modelos de dados em bancos NoSQL. Embora existam alguns trabalhos relacionados, como os de (Shah et al., 2022), (Gómez et al., 2021), (la Vega et al., 2020), (Hewasinghage et al., 2021) e (Kuszera et al., 2020), o estudo sobre a comparação quantitativa entre modelos de dados referenciados e aninhados permanece em grande parte incipiente. Um dos poucos estudos que aborda diretamente esse tema é o de Vera-Olivera et al. (2023), que propôs um índice de comparação para a recuperação de dados envolvendo esses tipos de relacionamentos.

O trabalho de Shah et al. (2022) propôs esquemas alternativos utilizando padrões específicos de estruturação em bancos de dados orientados a documentos. Eles realizaram uma análise detalhada sobre os critérios de armazenamento e o desempenho de consultas com e sem indexação, além de examinar operações de junção em nível de aplicação. Embora tenham contribuído para a otimização de consultas, o estudo não aborda a criação de um índice comparativo entre os diferentes modelos de dados.

Em um estudo paralelo, Gómez et al. (2021) desenvolveram o projeto SCORUS, que utiliza automação para gerar alternativas de estruturas e avaliar métricas estruturais em bancos de dados. Sua contribuição está na análise da complexidade estrutural, mas a comparação de desempenho entre abordagens de modelagem referenciada e aninhada não foi o foco principal de sua pesquisa.

Outro exemplo relevante é o trabalho de la Vega et al. (2020), que apresentaram o Mortadelo, um processo automatizado para a geração de bancos de dados NoSQL a partir de modelos conceituais. Embora o Mortadelo permita a criação de implementações automáticas, a avaliação do sistema não inclui métricas específicas para comparar a eficiência de diferentes esquemas de dados.

No estudo de Hewasinghage et al. (2021), os autores discutem a falta de *frameworks* padronizados para a otimização de dados e consultas em bancos de dados NoSQL. Embora tenham proposto um modelo de custo genérico para armazenamento e consulta, sua pesquisa não aborda diretamente a comparação entre modelos de dados aninhados e referenciados.

Em contraste, o trabalho de Vera-Olivera et al. (2023) representa uma das abordagens mais próximas da análise de coeficientes de ponderação, ao desenvolver um índice de comparação de desempenho entre modelos referenciados e aninhados em bancos de dados NoSQL orientados a documentos. Nesse estudo, foi introduzido o conceito do Índice de Desempenho (InD), uma métrica projetada para medir a eficiência relativa desses modelos sob diferentes configurações e cargas de trabalho.

Os resultados obtidos por Vera-Olivera et al. (2023) indicaram que, em média, o acesso a documentos aninhados é 2,20 vezes mais rápido com cache ativado e 1,74 vezes mais rápido com cache desativado no modo replica set. No modo standalone, o desempenho foi 1,70 vezes superior com cache ativado e 1,24 vezes sem cache. Esses resultados foram obtidos a partir de modelos sintéticos validados com datasets reais, oferecendo uma métrica prática para orientar decisões no design de esquemas de bancos de dados NoSQL.

2.3 Métricas de Avaliação

De forma similar, foi aplicado um protocolo de mapeamento sistemático para avaliar as métricas utilizadas na análise de esquemas em bancos de dados NoSQL orientados a documentos. O objetivo foi mapear o estado da arte e identificar as abordagens existentes que propõem indicadores quantitativos para comparar esquemas sob diferentes aspectos funcionais e estruturais. A aplicação rigorosa deste protocolo permitiu também estabelecer conexões diretas entre essas métricas e fatores de ponderação que influenciam a eficácia de um esquema.

Foram definidas as seguintes questões de pesquisa para orientar o levantamento: (1) Quais métricas têm sido propostas para avaliar esquemas em bancos de dados NoSQL orientados a documentos? (2) Quais aspectos dos esquemas são considerados por essas métricas (ex.: estrutura, desempenho de consulta, redundância, completude)? (3) Que abordagens metodológicas são utilizadas para validar ou aplicar essas métricas na prática?

A estratégia de busca foi construída com base nessas questões, integrando termos técnicos relacionados aos objetivos do mapeamento. A *string* de busca genérica utilizada foi:

NoSQL AND “document database” AND “schema evaluation” AND (metric OR performance OR structure OR completeness OR redundancy OR “query pattern” OR “access cost”)

Os trabalhos de (Kuszera et al., 2020), (Gómez et al., 2021) e (Vera-Olivera & Holanda, 2024) estabeleceram bases importantes nesta área, embora apresentem abordagens distintas e limitações específicas.

O estudo de Kuszera et al. (2020) concentra-se na avaliação de esquemas com base na capacidade de responder a consultas predefinidas. Utilizam grafos acíclicos dirigidos para modelar as consultas e os esquemas no formato `JSON`, avaliando caminhos e subcaminhos para determinar a cobertura das consultas. A métrica proposta atribui pontuações aos esquemas com base em sua capacidade de suportar as consultas, mas não aborda aspectos estruturais dos esquemas nem considera atributos complexos ou relacionamentos aninhados. Essa limitação deixa uma lacuna na análise integral dos esquemas.

Por outro lado, o trabalho de Gómez et al. (2021) adota uma abordagem estrutural, avaliando a qualidade do esquema em termos de consumo de memória, redundância e custo de navegação. Utilizam diagramas `UML` para gerar esquemas potenciais no formato `JSON` e avaliam sua complexidade por meio de métricas como profundidade e amplitude dos documentos, bem como taxas de referência. No entanto, este estudo não leva em consideração a capacidade do esquema para responder a consultas específicas, o que limita sua aplicabilidade em contextos onde as consultas são um fator crítico.

O trabalho de Vera-Olivera and Holanda (2024), aborda essas limitações ao propor um modelo de avaliação integral. São introduzidas quatro métricas principais: completude, padrão de acesso, custo de recuperação e redundância. Essas métricas consideram tanto a estrutura do esquema quanto a capacidade de responder a consultas específicas. Além disso, o modelo permite a avaliação simultânea de múltiplos esquemas e emprega coeficientes de ponderação para refletir a influência relativa de atributos e relacionamentos, tanto referenciados quanto aninhados.

Nos estudos de caso apresentados por Vera-Olivera and Holanda (2024), a proposta é validada ao demonstrar que os esquemas mais complexos tendem a obter pontuações mais altas, enquanto os esquemas mais simples apresentam pontuações mais baixas. Isso evidencia a capacidade do modelo para capturar a complexidade estrutural e a adequação funcional dos esquemas, proporcionando uma solução mais holística em comparação com as abordagens anteriores.

2.4 Geração de Esquemas

De forma consistente com as seções anteriores, a análise sobre geração automatizada de esquemas para bancos de dados NoSQL adota a mesma metodologia de Revisão Sistemática da Literatura detalhada na Figura 2.1. As seguintes questões de pesquisa foram definidas para guiar a análise: (1) Quais métodos têm sido propostos para a geração automática de esquemas em bancos de dados NoSQL? (2) Quais técnicas são utilizadas para adaptar os esquemas às cargas de trabalho e padrões de acesso? (3) Quais são os critérios ou métricas utilizadas para validar ou comparar esses métodos? A *string* de busca aplicada combinou termos relacionados aos objetivos do estudo, estruturada da seguinte forma:

(NoSQL AND "schema generation" OR "automatic design") AND ('workload-driven OR "schema optimization" OR "document database" OR "heuristic algorithm")

O desenho automatizado de esquemas em bancos de dados NoSQL tem sido amplamente estudado para otimizar o desempenho em sistemas que lidam com grandes volumes de dados distribuídos. Diversas abordagens têm tratado dos desafios específicos desses ambientes, destacando metodologias para bancos de dados orientados a documentos e baseados em colunas amplas.

Mozaffari and Nazemi (2023) propõem uma abordagem para bancos de dados NoSQL baseados em colunas amplas, utilizando um algoritmo "*workload-driven*" que analisa padrões de acesso de leitura e escrita. Seu método emprega técnicas como desnormalização, normalização parcial e fusão de tabelas, adaptando dinamicamente os esquemas em resposta a mudanças nas cargas de trabalho. Esta proposta melhora a eficiência em consultas e reduz custos de armazenamento, estabelecendo bases aplicáveis também a bancos de dados orientados a documentos devido às suas estruturas flexíveis.

Em um contexto diferente, A. A. Imam et al. (2020) introduzem o modelo *Dynamic Schema Proposition* (DSP) focado em bancos de dados NoSQL. Sua estratégia utiliza programação inteira binária e novas notações de cardinalidade para gerar esquemas otimizados, priorizando eficiência em operações CRUD. Embora o DSP se destaque em sistemas com dados estáticos e padrões previsíveis, sua abordagem determinística pode limitar a adaptabilidade a ambientes dinâmicos em comparação com algoritmos heurísticos.

A. A. Imam et al. (2018) desenvolvem algoritmos heurísticos para bancos de dados orientados a documentos, considerando parâmetros como operações CRUD, consistência, disponibilidade e segurança. Sua metodologia gera esquemas iniciais equilibrados entre relações incorporadas e referenciadas, otimizando escalabilidade e eficiência operacional.

Bansal et al. (2023) apresentam um modelo automatizado para bancos de dados orientados a documentos, como MongoDB, baseado em cargas de trabalho. Sua abordagem transforma esquemas conceituais em representações físicas otimizadas por meio de gráficos de consulta e etiquetas, melhorando o desempenho e simplificando a administração de da-

dos. Este modelo prioriza a eficiência prática em relação aos algoritmos metaheurísticos, tornando-o ideal para implementações rápidas.

Finalmente, Abdelhedi et al. (2021) desenvolvem o método *ToNoSQLSchema*, baseado na Arquitetura Dirigida por Modelos (MDA). Seu processo utiliza transformações formais em QVT (*Query/View/Transformation*) para extrair esquemas estruturados de bancos de dados orientados a documentos sem esquema predefinido. Aplicado em contextos médicos e jurídicos, essa abordagem se destaca por sua precisão e flexibilidade, facilitando a consulta e gestão de dados complexos.

2.5 Considerações Finais

Este capítulo evidenciou que a modelagem de bancos de dados `NoSQL` ainda é um campo em evolução, marcado pela predominância dos níveis conceitual e lógico sobre o físico, com ampla utilização de modelos `UML` e `ER` adaptados. Os processos de modelagem estão acontecendo sobre contextos como otimização de desempenho, migração de dados e automação de esquemas.

Sobre comparação de desempenho entre relacionamentos referenciados e aninhados, especificamente no custo de recuperação de dados, a literatura indica que os relacionamentos aninhados apresentam melhor eficiência. Contudo, não há evidências quantitativas que demonstrem a magnitude dessa vantagem em relação aos relacionamentos referenciados. A determinação dessa diferença de desempenho poderia embasar o desenvolvimento de métricas para avaliação de esquemas, contribuindo para decisões mais fundamentadas no projeto de banco de dados.

No que diz respeito a métricas, a literatura demonstra que alguns trabalhos dedicam-se exclusivamente a análise de consultas, enquanto outros focam na estrutura do esquema, sem abordar ambos os aspectos de forma integrada. Além disso, atributos simples e complexos não são adequadamente explorados nos estudos existentes. Por fim, verifica-se que soluções para um determinado problema podem envolver um ou múltiplos esquemas, uma abordagem que ainda não foi suficientemente investigada em pesquisas anteriores.

A Tabela 2.4 resume a classificação dos trabalhos analisados sobre otimização de bancos de dados orientados a documentos, organizados em três categorias principais: desenho de esquemas, otimização de esquemas e abordagens híbridas (desenho-otimização). A literatura revela que as abordagens mais recorrentes concentram-se na análise de operações `CRUD` para refinar a estrutura de esquemas em níveis lógico e físico. No entanto, essa metodologia mostra-se limitada para explorar o espaço de soluções possíveis, impossibilitando a identificação de esquemas ótimos para um cenário determinado. Embora existam propostas baseadas em modelos determinísticos e heurísticos, estas frequentemente negligenciam

métricas críticas que avaliam simultaneamente a qualidade estrutural dos esquemas e seu desempenho em nível lógico. Adicionalmente, há uma lacuna prática relevante: a maioria dos usuários (como desenvolvedores de aplicações) não possui conhecimento especializado sobre otimizações em nível físico (ex.: armazenamento, indexação). Consequentemente, suas intervenções restringem-se ao nível lógico, onde alterações são viáveis sem compreensão profunda da implementação física.

Tabela 2.4: Visão geral das propostas existentes para otimizar esquemas.

Trabalho	Enfoque	Entrada	Métricas	Baseado em	Algoritmo baseado em
Chen et al., 2022	desenho/otimização	`UML`, frequência consultas	custo consulta, armazenamento	*workload-driven*	regras
Reniers et al., 2020	desenho/otimização	`ER`, operações join, frequência consultas	custo consulta, armazenamento indices, frequência acesso	*workload-driven*	regras
De Lima and dos Santos Mello, 2015	desenho/otimização	`EER`, frequência consultas e volume	frequência acesso, volume dados	*workload-driven*	regras
la Vega et al., 2020	desenho	`ER`, `UML`, frequência consultas	—	*model-driven*	regras
Hewasinghage et al., 2020	otimização	Modelos, frequência consultas	custo armazenamento, custo consulta	*cost-based*	regras
Roy-Hubara et al., 2023	desenho	`UML`, requisitos funcionais	tempo consultas	requisitos	regras
Kuszera et al., 2022	otimização	modelo relacional, frequência consultas, espaço disco	tempo consultas, espaço em disco redundância	*workload-driven*	regras
Hewasinghage et al., 2023	otimização	objetivos otimização, frequência consultas	custo consultas, espaço em disco profundidade documentos	*workload-driven*	meta-heurística
Proposta atual	otimização	consultas, esquemas	completude, padrão acesso e redundância	—	meta-heurística

Capítulo 3

Visualização de Esquemas

Desde que o trabalho aborda o desenho de esquemas em bancos de dados NoSQL orientados a documentos, é necessário contar com uma ferramenta que permita visualizar corretamente um esquema, incluindo suas coleções, atributos, relacionamentos e suas cardinalidades. Assim, este capítulo introduz a notação gráfica para a visualização de esquemas de banco de dados, proporcionando uma representação das estruturas e interconexões presentes nos esquemas definida em (Vera et al., 2015).

3.1 Notação Gráfica

Na metodologia clássica de banco de dados, existem métodos de desenho de esquemas bem estabelecidos, como os diagramas ER e os diagramas UML, que permitem modelar a estrutura e as relações dos dados de forma clara e padronizada. Essas ferramentas têm sido amplamente adotadas e são consideradas essenciais no desenho de sistemas relacionais, facilitando a compreensão das estruturas e dependências dos dados.

Por outro lado, no contexto dos bancos de dados NoSQL orientados a documentos, essas ferramentas não estão bem definidas nem padronizadas. Bancos de dados NoSQL, como MongoDB e CouchDB, geralmente representam dados nos formatos JSON (*JavaScript Object Notation*), organizados em pares de chave-valor. Este formato permite uma grande flexibilidade e suportam dados heterogêneos sem a necessidade de esquemas rígidos. Contudo, esses formatos não facilitam a visualização das relações entre documentos, nem das cardinalidades, e tampouco diferenciam de forma clara entre atributos simples e complexos. Isso limita a compreensão do esquema e torna o desenho de sistemas NoSQL mais desafiador.

Na Figura 3.1, identificam-se principalmente duas coleções: “emprego” e categorias de pessoas como “adulto” e “crianca”. Contudo, as cardinalidades entre essas coleções não são de fácil interpretação, e embora a relação seja estabelecida por aninhamento,

torna-se desafiador distinguir precisamente onde começa e termina cada coleção. Além dessas, observam-se outras possíveis coleções, como “emprego_deteccao_pessoas”, mas não se sabe com exatidão se estas foram intencionalmente projetadas como coleções pelos modeladores de banco de dados. Adicionalmente, há a presença de atributos simples e complexos que também são pouco distinguíveis em meio ao código JSON da figura, dificultando a identificação clara da estrutura e dos tipos de dados. Essa falta de clareza na representação compromete a compreensão da organização dos dados e das relações entre as entidades envolvidas.


```
{
 "emprego": {
  "emprego_deteccao_pessoas": {
   "conteudo": {
    "categorias": {
     "adulto": {
      "criancas": [],
      "nome": "adulto",
      "color": "#733AFB",
      "id": "categoria47"
     },
     "crianca": {
      "criancas": [],
      "nome": "crianca",
      "color": "#8274c9",
      "id": "categoria48"
     }
    },
    "entrada": "radio"
   },
   "instrucao": "pessoas",
   "ehcrianca": false,
   "ferramentas": ["semantica"]
  }
 }
}
```


Figura 3.1: Exemplo esquema em formato JSON.

Diante dessa limitação, usamos uma notação gráfica de esquemas especificamente voltada para bancos de dados NoSQL orientados a documentos. Essa notação baseia-se em princípios de modelagem lógica do UML, introduzindo uma representação gráfica que facilita a visualização das relações, cardinalidades e tipos de atributos. Ao propor uma notação padrão de visualização, o estudo visa simplificar a interpretação dos dados e fornecer uma base visual que auxilie administradores e desenvolvedores na construção e

manutenção desses sistemas. Nesta notação, são abordados aspectos como a representação de coleções, relacionamentos (referenciados ou aninhados), atributos simples e complexos e cardinalidades.

3.1.1 Representação de coleções e atributos

Uma coleção corresponde a um agrupamento lógico de documentos, onde cada registro pode possuir uma estrutura distinta, atuando como um contêiner flexível para dados relacionados. Essa organização é visualizada na Figura 3.2, que exemplifica como documentos heterogêneos são agrupados em uma mesma coleção.

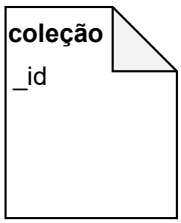


Figura 3.2: Notação gráfica para representar coleções.

Os atributos de um documento podem ser classificados em simples e complexos. Atributos simples armazenam valores atômicos, como números, strings ou datas, sendo representados diretamente no documento, sem a necessidade de estrutura adicional. Já os atributos complexos contêm coleções de dados ou outras estruturas, como vetores ou objetos aninhados, permitindo a representação de informações hierárquicas. Na Figura 3.3, os atributos simples e complexos são representados utilizando a notação chave-valor, na qual a chave corresponde ao nome do atributo e o valor indica o tipo de dado. Os atributos simples são apresentados diretamente, com o tipo de dado especificado (por exemplo, *string*, número, data, etc.). Por sua vez, os atributos complexos são identificados pela notação entre colchetes "[ ]", podendo ser vetores ou objetos (estruturas compostas por pares chave-valor aninhados). Essa notação visa fornecer uma visualização clara dos dados, facilitando a compreensão das relações e hierarquias entre os atributos em cada documento.

3.1.2 Relacionamentos e cardinalidades

Em bancos de dados `NoSQL` orientados a documentos, as relações entre coleções podem ser configuradas de duas formas: por referência ou por aninhamento. A Figura 3.4 apresenta o relacionamento por referência entre seis coleções. Na proposta, as referências são representadas por uma linha simples, e a direção da referência é indicado pelo *_id* do documento referenciado dentro da coleção que faz a referência. Entre a linha simples é

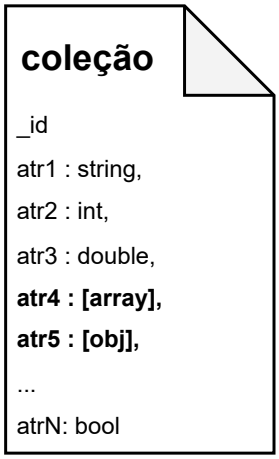


Figura 3.3: Notação gráfica para representar atributos nas coleções.

mostrado o tipo de cardinalidade entre duas coleções podendo tomar valores entre: 1..1, 1..N e N..M. A interpretação correta da notação é a seguinte: "A Coleção B referencia Coleção A com cardinalidade 1..1", ou ainda, "A Coleção C referencia Coleção D com cardinalidade 1..N" e "As coleções E e F referenciam-se mutuamente, apresentando uma cardinalidade de N..M".

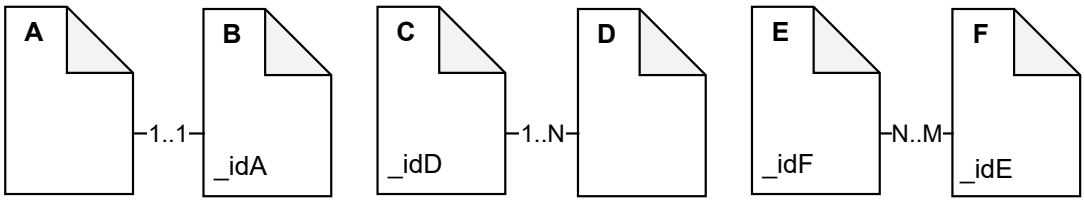


Figura 3.4: Notação gráfica de relacionamentos por referência.

Da mesma forma, a Figura 3.5 ilustra as relações por aninhamento entre duas coleções. Nesta figura, observa-se que a Coleção B está aninhada dentro da Coleção A, com uma cardinalidade de 1..1, conforme indicado no canto superior direito da Coleção B. A interpretação correta da notação é a seguinte: "A Coleção A aninha a Coleção B com cardinalidade 1..1".

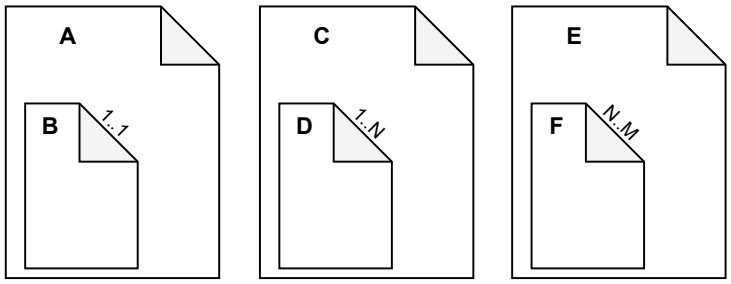


Figura 3.5: Notação gráfica de relacionamentos por aninhamento.

Um exemplo da aplicação da notação gráfica é apresentado na Figura 3.6, utilizando o *dataset* do `ENEM` (Exame Nacional do Ensino Médio) do ano 2021 como estudo de caso. O `ENEM` é uma avaliação educacional aplicada anualmente pelo `INEP`, vinculado ao `MEC`, que tem como objetivo avaliar o desempenho dos estudantes do ensino médio em diferentes áreas do conhecimento e verificar a proficiência em redação. Os dados contidos no *dataset* são relativos a: dados da escola, dados do questionário socioeconômico e dados da prova. Na Figura 3.6 são apresentadas duas coleções: "ALUNO" e "PROVA", ambas com atributos simples. A coleção "ALUNO" está relacionada à coleção "PROVA" por meio de um relacionamento por referencia, com cardinalidade 1..1.

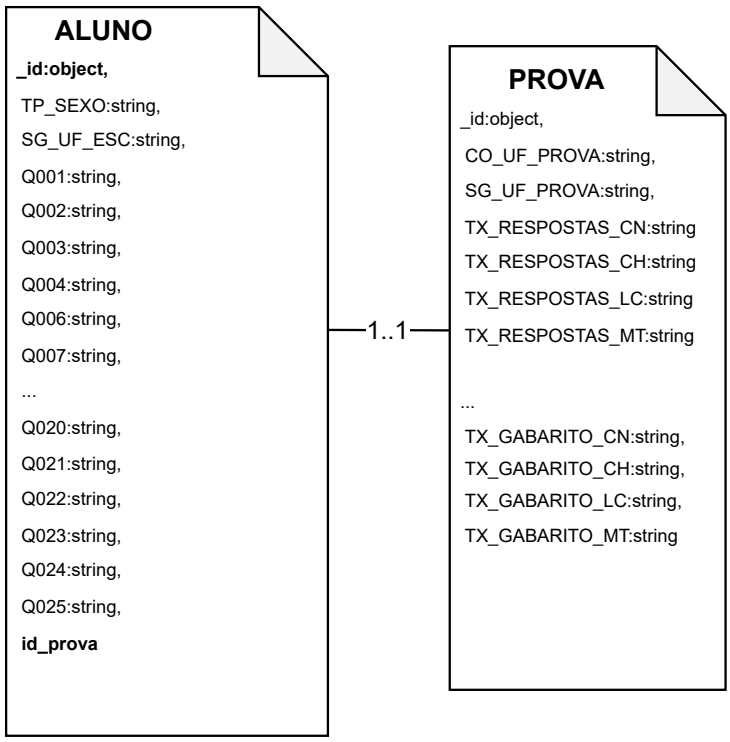


Figura 3.6: Notação gráfica aplicada ao *dataset* ENEM.

Outro exemplo de aplicação da notação gráfica é apresentada na Figura 3.7. Nesta figura apresenta-se um sistema de software para revistas científicas. Neste desenho de esquema, é possível identificar claramente quatro coleções: *Author*, *Article*, *Comments* e *Journal*. Além disso, observa-se que existe um único atributo complexo, denominado idAuthors, na coleção *Article*. Também é evidente como as quatro coleções estão relacionadas entre si por meio de duas relações por referência e uma por aninhamento, com suas respectivas cardinalidades claramente definidas.

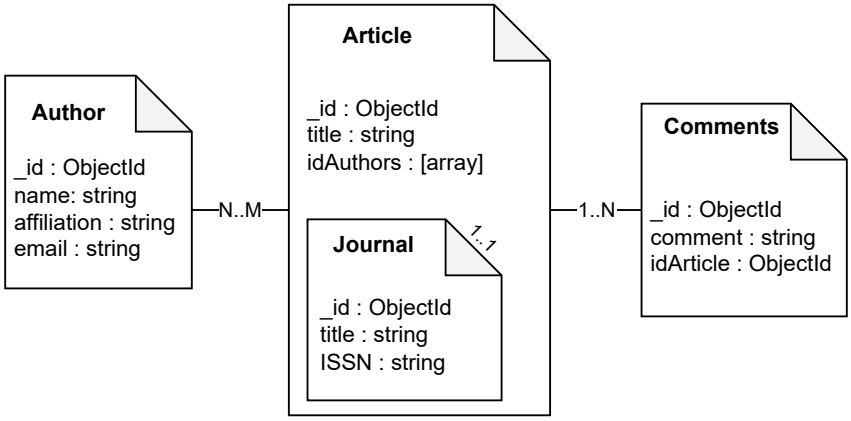


Figura 3.7: Notação gráfica aplicada a um sistema de software para revistas científicas.

3.2 Considerações Finais

A modelagem de dados é um elemento essencial no desenvolvimento de sistemas de informação, pois fornece uma estrutura organizada para armazenar, gerenciar e recuperar dados de forma eficiente (Simsion & Witt, 2004). Assim como um mapa ou diagrama facilita a compreensão de um território complexo, a modelagem permite visualizar e entender o problema mediante representações gráficas que mostram entidades, relacionamentos e regras de negócio de maneira clara e não ambígua. No entanto, em bancos de dados NoSQL, onde não há um padrão de modelagem aceito pela comunidade acadêmica e empresaria, os usuários frequentemente adotam estruturas flexíveis como documentos `JSON` ou `XML` sem uma abordagem sistemática, o que pode comprometer os princípios da modelagem ao não visualizar adequadamente o contexto do problema, os relacionamentos entre coleções, as cardinalidades ou os tipos de atributos. Essa falta de estrutura pode levar a redundâncias, inconsistências e dificuldades para manter a integridade dos dados em cenários complexos. A notação gráfica deste capítulo aborda essas lacunas na etapa de modelagem lógica, oferecendo uma metodologia que torna explícitos os relacionamentos e restrições entre coleções mesmo em bancos `NoSQL`, facilita a compreensão visual do problema mediante diagramas adaptados a modelos não relacionais, e define limites claros para o domínio dos dados, evitando armazenamento desorganizado ou semântica ambígua. Dessa forma, mesmo em ambientes flexíveis, é possível alcançar a clareza dos modelos relacionais tradicionais, garantindo que a qualidade dos dados não seja sacrificada pela ausência de esquemas rígidos. Diferentemente da nossa notação proposta, que prioriza a padronização visual e a clareza estrutural, De Lima and dos Santos Mello (2015) concentram-se na eficiência de acesso e no ajuste dos esquemas conforme a carga de trabalho da aplicação. Assim, enquanto nossa proposta fornece uma notação orientada à compreensão e documentação, De Lima and dos Santos Mello (2015) entregam uma

modelagem voltada à otimização operacional, embora ambas as contribuições avancem significativamente no desenvolvimento de notações especializadas para ambientes NoSQL. A notação gráfica apresentada neste capítulo é adotada como padrão nos demais capítulos deste trabalho.

Capítulo 4

Coeficientes de Ponderação

Este capítulo descreve a obtenção de coeficientes de ponderação para relacionamentos de referência e aninhamento, aplicando modelos de regressão múltipla a bases de dados sintéticas em cenários distribuídos e centralizados. Este capítulo estrutura-se em sete seções: na Seção 4.1 contextualização da obtenção dos coeficientes de ponderação; na Seção 4.2 é descrito o ambiente experimental; na Seção 4.3 é caracterizado o banco de dados sintético gerado; na Seção 4.4 a metodologia de geração dos *datasets* é apresentado; na Seção 4.5 é especificado os modelos de regressão múltipla empregados; na Seção 4.6 é apresentado o processo de obtenção dos coeficientes de ponderação; e na Seção 4.8 o cenário de validação adotado é mostrado.

4.1 Contextualização

Em bancos de dados NoSQL orientados a documentos, os relacionamentos entre coleções e, por extensão, as estratégias de recuperação de dados manifestam-se de duas formas principais: por referência ou por aninhamento. Embora se reconheça que o acesso a informações em estruturas aninhadas tende a ser mais ágil, ainda não se comprovaram, de forma quantitativa, os ganhos de desempenho em relação às coleções referenciadas. Por isso, torna-se essencial definir coeficientes de ponderação que representem adequadamente cada tipo de vínculo, permitindo avaliar com maior precisão o impacto dessas escolhas no tempo de resposta do sistema.

O processo metodológico aplicado para a determinação dos coeficientes é mostrado na Figura 4.1 e consiste em quatro etapas. Na primeira etapa, foram criados bancos de dados sintéticos nos modos centralizados e distribuídos em diferentes equipamentos de computação. Em seguida, conjuntos de dados foram gerados a partir de consultas feitas nos bancos de dados. Na etapa seguinte, construímos modelos de regressão múltipla a partir dos conjuntos de dados. Finalmente, na quarta etapa, obtivemos coeficientes de

ponderação associados aos documentos referenciados e aninhados usando os modelos de regressão múltipla.

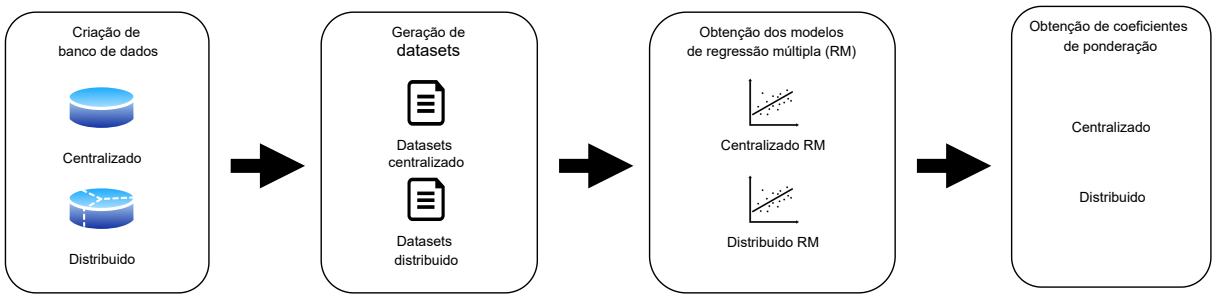


Figura 4.1: Processo metodológico para obtenção de coeficientes de ponderação.

4.2 Ambiente Experimental

O ambiente experimental consistiu em uma infraestrutura híbrida, compreendendo equipamentos físicos e instâncias virtuais em nuvem. A configuração detalhada dos recursos computacionais está apresentada na Tabela 4.1, enquanto os principais aspectos são discutidos a seguir.

Tabela 4.1: Equipamentos computacionais físicos e na nuvem utilizados.

Equipamento	CPU	RAM	Armazenamento	S.O.	Tipo	Quantidade
Eq_1	Intel Xeon 32 núcleos	128 GB	HDD 1,82 TB	Ubuntu 22.04	físico	1
Eq_2	Intel Core i7 16 núcleos	16 GB	HDD 500 GB	Ubuntu 22.04	físico	3
Eq_3	Intel Cascade Lake 2 núcleos	4 GB	HDD 250 GB	Ubuntu 20.04	nuvem	3

No ambiente físico, foi empregado um servidor identificado como Eq_1, com as seguintes características: CPU Intel Xeon de 32 núcleos, 128 GB de RAM, e armazenamento HDD de 1,82 TB, rodando o sistema operacional Ubuntu 22.04. Além disso, foram utilizadas três estações de trabalho, identificadas como Eq_2, cada uma equipada com um processador Intel Core i7 de 16 núcleos, 16 GB de memória RAM e armazenamento HDD de 500 GB, também com sistema operacional Ubuntu 22.04.

Por outro lado, no ambiente virtual, foram utilizados três equipamentos na nuvem, identificados como Eq_3. Cada um deles possui processador *Intel Cascade Lake* de dois núcleos, 4 GB de memória RAM e armazenamento HDD de 250 GB. Esses equipamentos operam com o sistema operacional Ubuntu 22.04 e foram implementados na plataforma *Google Cloud*, permitindo a execução de experimentos em ambientes distribuídos.

4.2.1 Arquitetura de Implantação

As arquiteturas de implantação utilizadas neste capítulo foram baseadas nas configurações centralizadas (*standalone*) e distribuídas (*replica set*) do `MongoDB`, com o objetivo de avaliar o desempenho em diferentes cenários operacionais. No modo centralizado (Figura 4.2), o banco de dados é configurado em uma única instância, centralizando todas as operações de leitura e escrita em um único nó, o que permite uma análise controlada do impacto das consultas sobre os recursos físicos disponíveis. Já no modo distribuído (Figura 4.3), o `MongoDB` foi configurado para operar em um ambiente distribuído, composto por um nó primário e dois nós secundários. Essa configuração habilita a replicação de dados, melhorando a resiliência e a disponibilidade do sistema, além de permitir a execução de leituras em nós secundários para reduzir a carga no nó primário.

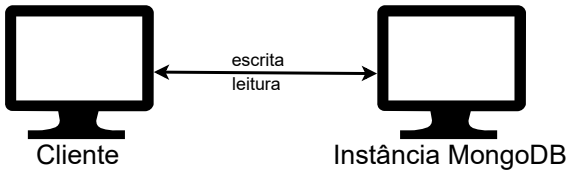


Figura 4.2: Arquitetura centralizada

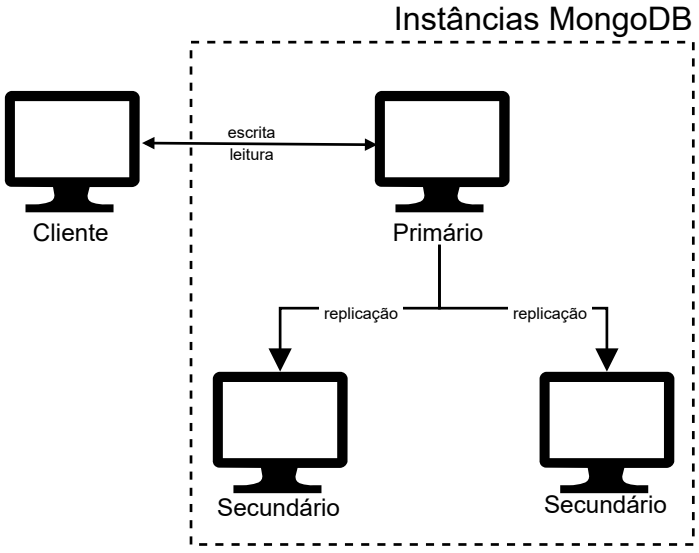


Figura 4.3: Arquitetura distribuída.

O sistema de banco de dados usado foi o `MongoDB` 6.0 com as configurações padrões de instalação. Em particular, as funcionalidades *read concern* e *write concern* foram executadas com suas configurações padrão, sendo "local" para *read concern* e "acknowledge" para *write concern*. Adicionalmente, o modo de preferência de leitura (*read preference mode*) foi estabelecido com a configuração padrão "primary".

Finalmente, no equipamento Eq_1, foram implementadas as arquiteturas centralizada e distribuída. Para a arquitetura distribuída, especificamente, foi utilizado o Docker na versão 23.0 para virtualizar três instâncias do `MongoDB`, configuradas de forma a simular um ambiente distribuído. Adicionalmente, a arquitetura distribuída também foi implementada nos equipamentos Eq_2 e Eq_3, garantindo a avaliação do desempenho em diferentes configurações físicas e virtuais.

A Tabela 4.2 apresenta um resumo detalhado da distribuição das arquiteturas implementadas nos diferentes equipamentos utilizados durante os experimentos.

Tabela 4.2: Distribuição das arquiteturas implementadas.

Equipamento	Centralizado	Distribuído
Eq_1	X	X
Eq_2		X
Eq_3		X

4.3 Banco de Dados Sintético

Nesta subseção, será apresentado o processo de desenvolvimento do banco de dados sintético, criado com o objetivo de possibilitar a execução de consultas específicas voltadas à análise de desempenho. A Figura 4.4 apresenta o esquema do banco de dados baseado em referência, enquanto a Figura 4.5 ilustra o esquema aninhado. Para isso, foram desenvolvidos dois esquemas distintos envolvendo duas coleções, denominadas A e B. No primeiro esquema, as duas coleções estão relacionadas por referência, indicada na coleção A pelo atributo b_Id. Essa relação apresenta uma cardinalidade de 1 para 1, garantindo que cada documento na coleção A esteja associado a exatamente um documento na coleção B. Por outro lado, no segundo esquema, foi implementado um esquema aninhado em que a coleção A contém a coleção B como um documento aninhado, também com cardinalidade de 1 para 1.

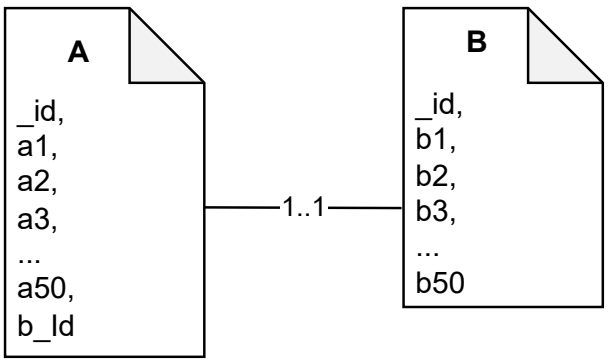


Figura 4.4: Esquema com coleções referenciadas.

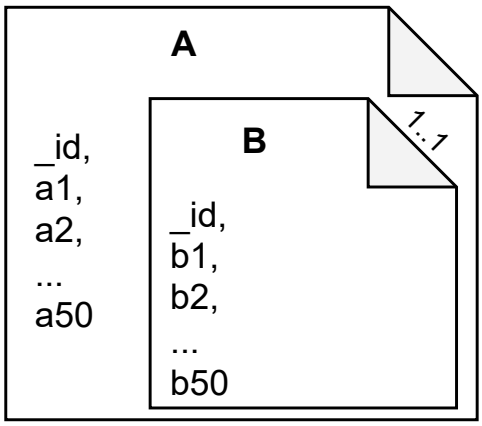


Figura 4.5: Esquema com coleções aninhadas.

Ambas coleções possuem um identificador único no atributo *_id*, e os demais atributos das coleções foram gerados de forma aleatória, com valores do tipo *string* e comprimento variando entre 10 e 50 caracteres. Esses atributos foram projetados para simular cenários reais de consulta.

Os dois esquemas foram implementados nas arquiteturas centralizada e distribuída, utilizando os equipamentos descritos na Tabela 4.1, conforme a distribuição apresentada na Tabela 4.2. Como apresentado o equipamento Eq_1 foi utilizado de modo centralizado e distribuído e os equipamentos Eq_2 e Eq_3 de forma distribuído. Esses esquemas foram configurados no banco de dados `MongoDB` 6.0 e populados com 10 milhões de documentos.

4.4 Datasets

Para realizar uma avaliação abrangente e comparável do desempenho de bancos de dados `NoSQL`, é fundamental dispor de um conjunto de dados que generalize comportamentos típicos desses sistemas. O uso de *datasets* reais, embora possa refletir cenários específicos, geralmente está vinculado a aplicações particulares e limita a possibilidade de replicar experimentos em diferentes contextos. Por esse motivo, optou-se pela construção de um *dataset* baseado no banco de dados sintético, cuidadosamente projetado para abstrair padrões de uso comuns e representar estruturas e operações genéricas encontradas em bancos de dados `NoSQL`.

Os *datasets* foram criados com base no banco de dados sintético previamente implementado, considerando os dois esquemas de dados (referenciado e aninhado) e os cenários de configuração centralizado e distribuído. Essa abordagem foi essencial para garantir a coleta de dados estruturados e consistentes, permitindo uma análise detalhada do desempenho das consultas em diversas configurações e topologias.

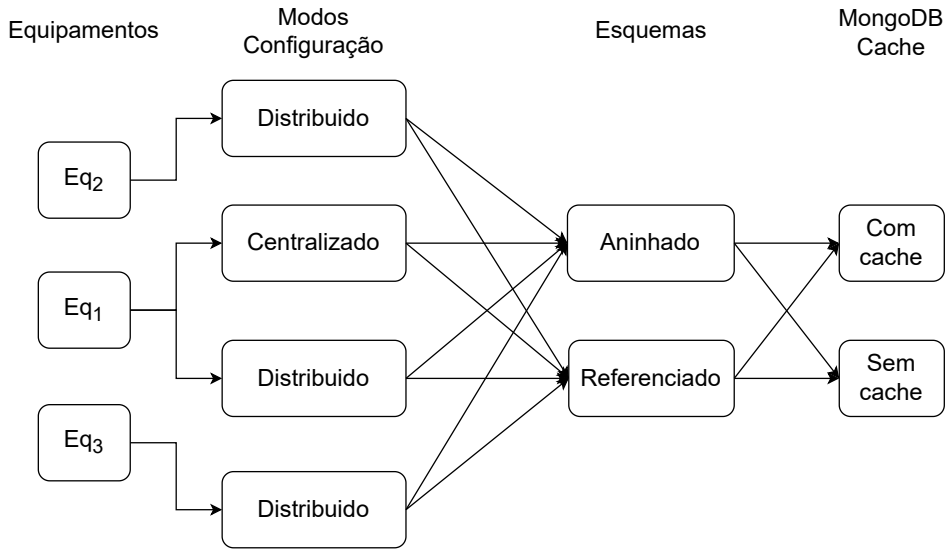


Figura 4.6: Processo de geração dos 16 *datasets*.

Para cada consulta, foram definidos múltiplos cenários de consultas, começando com 200.000 documentos e aumentando progressivamente até 5 milhões. Em cada cenário, o número de atributos por documento variou de 5 a 50, permitindo avaliar como a quantidade de informações impacta o desempenho das consultas. A consulta preparada para os experimentos, denominada Q_1, foi projetada para recuperar a mesma quantidade de atributos dos documentos A e B.

Código 4.1: Consulta Q_1 implementada no esquema referenciado e aninhado

```
# Modelo Referenciado
consulta = [
    {'$limit': n},  # Limitar a quantidade de documentos a obter
        .
    # {'$sample': {'size': n}},  # Obter n documentos aleatorios
        da colecao A
    {
        '$lookup': {
            'from': 'B',
            'localField': 'id_B',
            'foreignField': '_id',
            'as': 'documentosB'
        }
    },
    {
```

```
        '$project': {
            '_id': 1,  # Incluir o campo _id da colecao A
            **proyeccion_a,  # Selecionar os atributos de A de
               forma dinamica
            'documentosB': {
                '_id': '$documentosB._id',  # Incluir o campo
                   _id da colecao B
                **proyeccion_b  # Selecionar os atributos de B
                   de forma dinamica
            }
        }
    }
]

# Modelo Aninhado
consulta = [
    {'$limit': n},
    # {'$sample': {'size': n}},
    {'$project': {
        '_id': 1,  # Incluir o _id de A
        **{f'a{i}': f'$a{i}' for i in range(1, x+1)},
        'B': {
            '_id': '$B._id',
            **{f'b{i}': f'$B.b{i}' for i in range(1, y+1)}
        }
    }}
]
```

No modelo referenciado, a consulta Q_1 foi implementada utilizando o *pipeline* de agregação do `MongoDB`, um *framework* para processamento de dados que transforma documentos por meio de uma sequência de operações. Especificamente, empregou-se o método *aggregate* com três estágios principais: (1) *lookup*, que realiza junções entre coleções; (2) *project*, que seleciona campos específicos; e (3) *limit*, que restringe o número de documentos retornados. No modelo aninhado, a mesma consulta Q_1 pode ser simplificada para apenas os estágios *project* e *limit*, aproveitando a estrutura dos documentos aninhados que já incorporam os relacionamentos. Ambas implementações foram executadas nas arquiteturas centralizada e distribuída, utilizando a infraestrutura descrita na Tabela 4.1.

Para assegurar uma análise abrangente, as consultas foram executadas com e sem o

cache do `MongoDB`. O processo de geração dos *datasets*, ilustrado na Figura 4.6, resultou em dados cuja estrutura, apresentada na Tabela 4.3, é composta por três colunas principais:

- Número documentos: representa a quantidade de documentos recuperados por consulta;
- Número atributos: indica a quantidade de atributos por documento incluídos no resultado;
- Tempo resposta: registra o tempo necessário para executar cada consulta

Tabela 4.3: Estrutura do *dataset* gerado.

Número Documentos	Número atributos	Tempo resposta
200 000	5	t_1
200 000	10	t_{11}
...	...	...
200 000	50	t_{91}
400 000	5	t_{101}
400 000	10	t_{111}
...	...	...
400 000	50	t_{191}
...	...	...
5 000 000	5	t_{2401}
5 000 000	10	t_{2411}
...	...	...
5 000 000	50	t_{2500}

Todas as consultas foram executadas com e sem a influência do cache, assegurando a coleta de dados que refletem o desempenho em diversas condições operacionais e configurações de hardware. Essa metodologia permitiu capturar, de maneira abrangente, as variações de desempenho em cada cenário avaliado.

4.5 Modelos de Regressão Múltipla

Os modelos de regressão múltipla (scikit-learn, 2024) foram desenvolvidos com o objetivo de analisar o comportamento do tempo de resposta das consultas em função de variáveis independentes, como o número de documentos recuperados e a quantidade de atributos por documento. Modelos de regressão múltipla demonstram utilidade na predição, especialmente quando os dados apresentam relações lineares e a quantidade disponível do

dataset é limitada (James et al., 2013). Esta abordagem permitiu a criação de modelos estatísticos que representam o desempenho dos dois esquemas de modelagem de dados (referenciado e aninhado) em diferentes cenários de configuração.

4.5.1 Construção dos Modelos

Os modelos de regressão múltipla foram construídos com base nos *datasets* gerados, descritos na Seção 4.4, e foram ajustados separadamente para cada uma das combinações de fatores:

- Modelos de dados: referenciado e aninhado;
- Arquiteturas: centralizada e distribuída;
- Estados de cache: com e sem;
- Equipamentos: Eq_1, Eq_2, e Eq_3

Cada modelo foi treinado considerando duas variáveis independentes:

- Número de documentos recuperados (X_1): indica a quantidade de documentos incluídos na consulta;
- Número de atributos por documento (X_2): representa a complexidade das consultas em termos de informações solicitadas (atributos)

A variável dependente (Y) foi definida como o tempo de resposta das consultas, medido em milissegundos. A relação entre essas variáveis foi modelada pela equação geral da regressão múltipla:

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \epsilon$$

onde:

Y : Tempo de resposta das consultas

β_0 : Intercepto

β_1 : Coeficiente associado ao número de documentos recuperados

β_2 : Coeficiente associado ao número de atributos por documento

ϵ : Termo de erro

4.5.2 Treinamento e Validação

O processo de geração dos modelos de regressão múltipla foi conduzido com base no mesmo procedimento utilizado para estruturar os experimentos descritos na Figura 4.6. Como resultado dos 16 *datasets* gerados nesse processo, foram criados 16 modelos de regressão múltipla, cada um correspondendo a uma combinação específica de arquitetura, equipamento, esquema de modelagem de dados e estado do cache.

Os modelos foram treinados utilizando os *datasets* correspondentes a cada combinação de fatores. O ajuste dos modelos foi avaliado pela métrica $R^2 score$ para entender o quão bem o modelo explica a variabilidade nos dados. O *dataset* foi dividido em 70% para treinamento e 30% teste.

A ferramenta utilizada para a obtenção dos modelos foi o $Scikit - learn$ (scikit-learn, 2024), com os parâmetros $fit\_intercept$, $copy\_X$, n_jobs e *positive* configurados com seus valores padrão. O parâmetro $fit\_intercept$ garantiu que o modelo calculasse o intercepto durante o ajuste, enquanto $copy\_X$ assegurou que os dados originais não fossem alterados durante o processamento. O parâmetro n_jobs, ao permanecer em seu valor padrão, utilizou um único núcleo de processamento para a execução do modelo, e o parâmetro *positive* permitiu a geração de coeficientes positivos, embora não tenha sido aplicado neste caso específico devido à configuração padrão.

A Tabela 4.4 apresenta os resultados dos modelos de regressão múltipla obtidos para o equipamento Eq_1 no modo centralizado. Esses resultados destacam os coeficientes ajustados, o intercepto, e os valores do R^2 tanto para o conjunto de treinamento quanto para o de teste. De forma semelhante, as Tabelas 4.5, 4.6 e 4.7 apresentam os modelos de regressão múltipla gerados para o modo *replica set* nos ambientes Eq_1, Eq_2 e Eq_3.

Tabela 4.4: Modelos de regressão múltipla obtidos no Eq_1 para arquitetura centralizada.

Esquema	**Cache**	β_1	β_2	β_0	R^2 **Treinamento**	R^2 **Teste**
Referenciado	Com cache	1.02472334	0.21763937	-3.017720709513224	0.99	0.99
	Sem cache	0.99698879	0.14871953	-2.551136584402331	0.99	0.99
Aninhado	Com cache	1.04980047	-0.00863899	-3.1112693442386834	0.97	0.97
	Sem cache	1.0035651	-0.01104212	-2.477332503137232	0.99	0.99

Tabela 4.5: Modelos de regressão múltipla obtidos no Eq_1 para arquitetura distribuída.

Esquema	**Cache**	β_1	β_2	β_0	R^2 **Treinamento**	R^2 **Teste**
Referenciado	Com cache	1.02265971	0.19742946	-2.9585879035633367	0.97	0.97
	Sem cache	1.029068	0.20535015	-3.017058594187399	0.99	0.99
Aninhado	Com cache	1.05163906	-0.01738922	-3.130097427002466	0.97	0.97
	Sem cache	1.01645005	-0.01997622	-2.9198769723630726	0.98	0.98

Tabela 4.6: Modelos de regressão múltipla obtidos no Eq_2 para arquitetura distribuída.

Esquema	**Cache**	β_1	β_2	β_0	R^2 **Treinamento**	R^2 **Teste**
Referenciado	Com cache	1.04713869	0.20608861	-3.1017870609811835	0.97	0.97
	Sem cache	1.02776791	0.19461247	-2.978403725711111	0.98	0.98
Aninhado	Com cache	1.01856333	-0.03465397	-2.9172136708137346	0.97	0.97
	Sem cache	1.0129133	-0.00998741	-2.903627613427707	0.97	0.96

Tabela 4.7: Modelos de regressão múltipla obtidos no Eq_3 para arquitetura distribuída.

Esquema	**Cache**	β_1	β_2	β_0	R^2 **Treinamento**	R^2 **Teste**
Referenciado	Com cache	2.51495792	0.07972045	-11.301507836259296	0.84	0.84
	Sem cache	1.89948341	0.0667366	-7.50971984432025	0.74	0.73
Aninhado	Com cache	2.45338698	-0.03287923	-11.112852953364289	0.82	0.83
	Sem cache	1.72818948	-0.01729218	-6.564202672819475	0.67	0.67

4.6 Determinação dos Coeficientes de Ponderação para Relacionamentos

Nesta seção, descreve-se o processo de determinação dos coeficientes de ponderação, os quais foram calculados mediante aplicação dos modelos de regressão múltipla apresentados anteriormente. Estes coeficientes quantificam a intensidade relativa das relações por referência e por aninhamento em bancos de dados documentais. Os coeficientes obtidos variam em uma escala de 0 a 1.

4.6.1 Processo de Obtenção

As fórmulas utilizadas para obter os coeficientes de ponderação são apresentadas a seguir:

$$coef_{ref} = \frac{ref}{ref + emb} \tag{4.1}$$

$$coef_{emb} = \frac{emb}{ref + emb} \tag{4.2}$$

Onde:

- $coef_{ref}$: Coeficiente de ponderação para relações por referência;
- $coef_{emb}$: Coeficiente de ponderação para relações por aninhamento;
- *ref*: Média dos tempos de resposta calculados para as relações por referência;
- *emb*: Média dos tempos de resposta calculados para as relações por aninhamento

No entanto, os valores de *ref* e *emb*, que correspondem às médias dos tempos de resposta dos relacionamentos por referência e por aninhamento, respectivamente, precisam ser calculados previamente. A Figura 4.7 apresenta o processo detalhado para a obtenção desses valores, descrito a seguir:

1. Foram preparadas consultas Q, variando aleatoriamente o número de documentos "d" (de 1 a 10 milhões) e o número de atributos "a" por documento (de 1 a 50);

2. Cada consulta foi processada pelos modelos de regressão múltipla para gerar tempos de resposta estimados para os esquemas referenciado (ref) e aninhado (emb);

3. Após 1000 iterações para cada configuração experimental (centralizado, distribuído e equipamentos Eq_1, Eq_2 e Eq_3), foram calculadas as médias dos tempos de resposta para cada tipo de relação, gerando os valores de $\overline{ref}$ e $\overline{emb}$

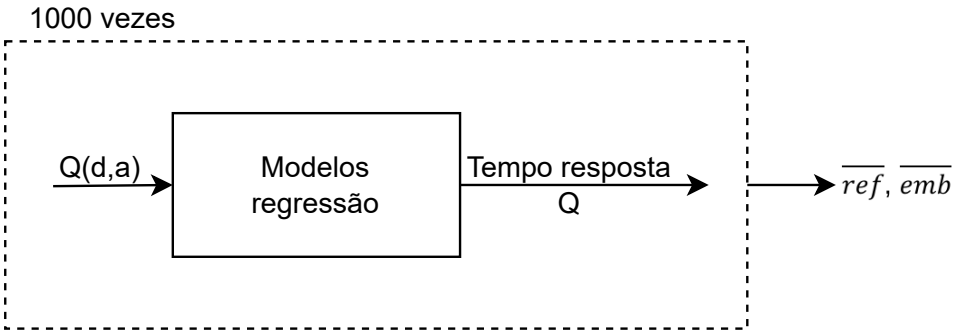


Figura 4.7: Execução dos modelos de regressão para obter as médias $\overline{ref}$ e $\overline{emb}$.

Os valores de $\overline{ref}$ e $\overline{emb}$ foram, então, inseridos nas Fórmulas 4.1 e 4.2 apresentadas anteriormente para calcular os coeficientes de ponderação ($coef_{ref}$, $coef_{emb}$), que representam a intensidade relativa de cada tipo de relação no desempenho do banco de dados.

A Tabela 4.8 detalha os coeficientes obtidos no modo centralizado no dispositivo Eq_1. Com a memória cache ativada, o coeficiente de referência é de 0.51, enquanto o coeficiente de aninhamento assume o valor de 0.49. Quando a memória cache está desativada, os coeficientes permanecem os mesmos. Esses resultados indicam um leve favorecimento das relações por referência nesse modo de configuração.

Tabela 4.8: Coeficientes de ponderação em modo centralizado no equipamento Eq_1.

	Com cache	Sem cache
$coef_{ref}$	0.51	0.51
$coef_{emb}$	0.49	0.49

As Tabelas 4.9, 4.10 e 4.11 mostram os coeficientes para o modo distribuído nos dispositivos Eq_1, Eq_2, e Eq_3. Em geral, os coeficientes de referência são consistentemente superiores aos coeficientes de aninhamento em todas as configurações.

Tabela 4.9: Coeficientes de ponderação em modo distribuído no equipamento Eq_1.

	Com cache	Sem cache
$coef_{ref}$	0.52	0.52
$coef_{emb}$	0.48	0.48

Tabela 4.10: Coeficientes de ponderação em modo distribuído no equipamento Eq_2.

	Com cache	Sem cache
$coef_{ref}$	0.52	0.52
$coef_{emb}$	0.48	0.48

Tabela 4.11: Coeficientes de ponderação em modo distribuído no equipamento Eq_3.

	Com cache	Sem cache
$coef_{ref}$	0.52	0.51
$coef_{emb}$	0.48	0.49

Os resultados indicam que as relações por referência apresentam maior impacto no desempenho, independentemente da configuração do sistema (centralizado ou distribuído) ou do equipamento utilizado. Essa tendência é evidente tanto com a memória cache ativada quanto desativada. Embora as diferenças entre os coeficientes sejam sutis, elas podem ter implicações significativas no desempenho geral do modelo, especialmente ao decidir entre o uso de relações por referência ou aninhamento em cenários específicos.

4.7 Determinação dos Coeficientes de Ponderação para Atributos Simples e Complexos

Para determinar os coeficientes de ponderação dos atributos simples e complexos, seguiu-se a metodologia apresentada na Figura 4.8. Foi utilizado o mesmo banco de dados sintético descrito na Seção 4.3, implementado nos equipamentos Eq_1 no modo centralizado e Eq_2 no modo distribuído. Uma consulta Q_2 foi preparada para selecionar aleatoriamente um único atributo de um esquema, medir o tempo de resposta para a recuperação desse atributo e registrar as informações no dataset correspondente. Esse dataset armazena o tipo de atributo, o tipo de dado e o tempo de recuperação, utilizando a estrutura "*attrType*, *dataType*, *time*". Com base nos datasets obtidos tanto no modo centralizado

quanto no modo distribuído, foram calculadas as médias dos tempos de recuperação para atributos simples e complexos.

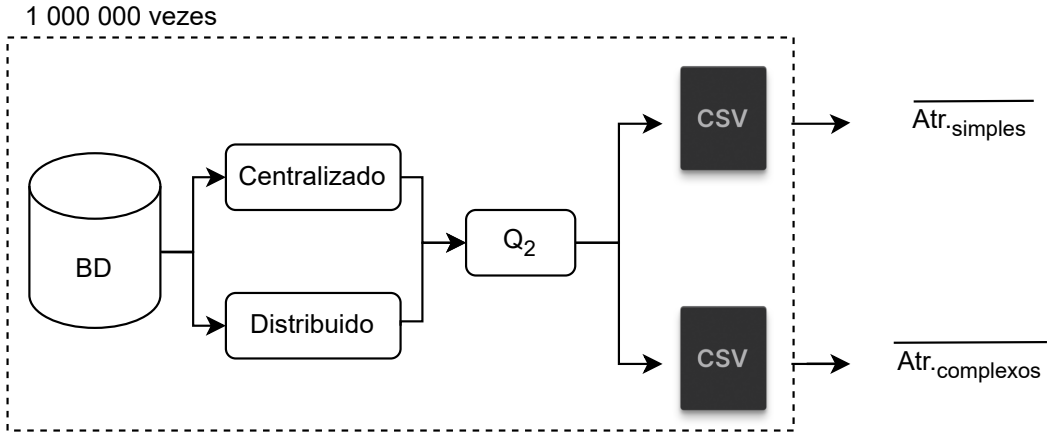


Figura 4.8: Processo para obter as médias de tempo de resposta para atributos simples e complexos.

Assim, com as médias $\overline{Atr.simples}$ e $\overline{Atr.complexos}$, foram determinados os coeficientes de ponderação para atributos simples ($coef_s$) e complexos ($coef_c$), conforme as Equações 4.3 e 4.4.

$$coef_s = \frac{\overline{Atr._{simples}}}{\overline{Atr._{simples}} + \overline{Atr._{complexos}}} \tag{4.3}$$

$$coef_c = \frac{\overline{Atr._{complexos}}}{\overline{Atr._{simples}} + \overline{Atr._{complexos}}} \tag{4.4}$$

A Tabela 4.12 apresenta os coeficientes de ponderação obtidos para atributos simples e complexos em dois modos operacionais: centralizado e distribuído, avaliados nas condições com e sem cache. Os resultados mostram que, no modo centralizado, os coeficientes mantêm valores muito próximos entre si, indicando baixa variação tanto para atributos simples (0,51 com cache e 0,51 sem cache) quanto para atributos complexos (0,49 com cache e 0,49 sem cache). Por outro lado, no modo distribuído, observa-se uma maior variação entre os coeficientes obtidos, especialmente para atributos simples (0,53 com cache e 0,49 sem cache) e complexos (0,51 com cache e 0,47 sem cache). Este resultado sugere que, em ambientes distribuídos, o cache exerce um impacto mais significativo, promovendo diferenças notáveis no comportamento de recuperação de dados conforme a configuração adotada. Além disso, a variação acentuada nos coeficientes distribuídos pode ser atribuída à latência e à dispersão física dos dados entre múltiplos nós, o que potencializa o benefício do cache no acesso eficiente às informações.

Tabela 4.12: Coeficientes de ponderação obtidos nos modos centralizado e distribuído.

	Simples		Complexos	
	Com Cache	Sem Cache	Com Cache	Sem Cache
centralizado	0.51	0.51	0.49	0.49
distribuído	0.53	0.49	0.51	0.47

De forma geral, os atributos simples mantêm um peso médio superior (0,51) em comparação aos atributos complexos (0,49), conforme estabelecido nos critérios iniciais. Estes valores médios serão usados em capítulos posteriores como uma forma de ponderar os atributos.

4.8 Cenário de Validação

A Figura 4.9 apresenta o processo de obtenção dos coeficientes de ponderação no cenário de validação. Para avaliar a aplicabilidade dos coeficientes de ponderação obtidos pelos modelos de regressão múltipla, foi desenvolvida uma nova base de dados sintética, seguindo os critérios estabelecidos na Seção 4.3. Essa base de dados foi estruturada em dois esquemas principais: um esquema com relações referenciadas e outro com relações aninhadas. Cada documento foi projetado com 50 atributos simples do tipo *string*, com tamanhos variando entre 10 e 50 caracteres.

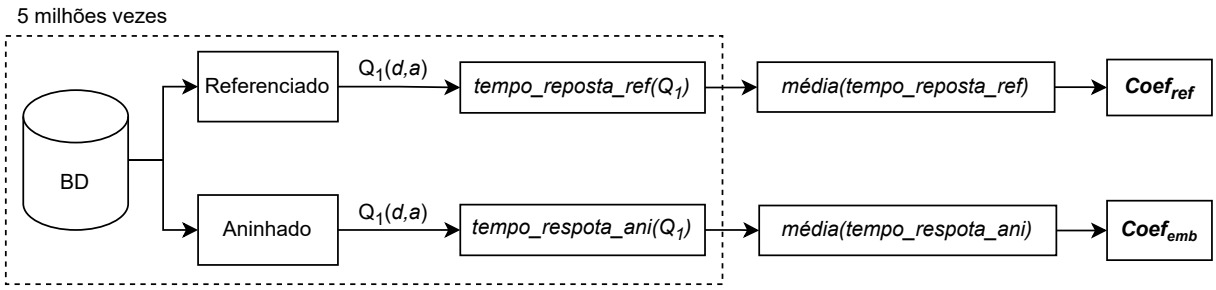


Figura 4.9: Processo para obter os coeficientes de ponderação no cenário de validação.

O banco de dados foi populado com 10 milhões de documentos para cada modelo. Esse volume de dados foi definido a fim de garantir que os experimentos fossem conduzidos em condições similares às utilizadas durante o treinamento dos modelos de regressão, assegurando consistência entre os cenários de treinamento e validação.

O objetivo principal dessa nova base de dados sintética foi verificar a precisão dos coeficientes de ponderação ao aplicar os modelos em um contexto prático. Além disso, o estudo buscou explorar o comportamento dos modelos em diferentes configurações de

consulta, variando o número de documentos recuperados e a quantidade de atributos processados por consulta.

A consulta Q_1, definida na Seção 4.4, foi executada mantendo os mesmos parâmetros estabelecidos, porém com uma variação controlada: cada execução processou um conjunto aleatório de d documentos e projetou um subconjunto aleatório de a atributos ($Q_1(d, a)$). Esta operação foi repetida sistematicamente 5 milhões de vezes tanto para o modelo referenciado quanto para o modelo aninhado, seguindo o fluxo experimental ilustrado na Figura 4.9.

Os tempos de resposta das consultas foram registrados ao longo das execuções e utilizados para calcular as médias dos tempos de resposta. Essas médias são representadas como:

- $média(tempo\_resposta\_ref)$ ou $\overline{ref}$ para o esquema referenciado;
- $média(tempo\_resposta\_ani)$ ou $\overline{emb}$ para o esquema aninhado

Por fim, os valores médios $\overline{ref}$ e $\overline{emb}$ foram utilizados para derivar os coeficientes de ponderação ($coef_{ref}$ e $coef_{emb}$) conforme as Equações 4.1 e 4.2. Os resultados obtidos para os coeficientes no caso de validação estão apresentados nas Tabelas 4.13, 4.14, 4.15 e 4.16.

Tabela 4.13: Coeficientes de ponderação para o caso de validação no modo centralizado no equipamento Eq_1.

	Com cache	Sem cache
$coef_{ref}$	0.67	0.56
$coef_{emb}$	0.33	0.44

Tabela 4.14: Coeficientes de ponderação para o caso de validação no modo distribuído no equipamento Eq_1.

	Com cache	Sem cache
$coef_{ref}$	0.65	0.66
$coef_{emb}$	0.35	0.34

Tabela 4.15: Coeficientes de ponderação para o caso de validação no modo distribuído no equipamento Eq_2.

	Com cache	Sem cache
$coef_{ref}$	0.60	0.57
$coef_{emb}$	0.40	0.43

Tabela 4.16: Coeficientes de ponderação para o caso de validação no modo distribuído no equipamento Eq_3.

	Com cache	Sem cache
$coef_{ref}$	0.64	0.60
$coef_{emb}$	0.36	0.40

Os valores de $coef_{ref}$ são consistentemente maiores em todos os experimentos (Tabelas 4.13, 4.14, 4.15, 4.16). Essa diferença é especialmente evidente no modo centralizado e nos equipamentos com maior capacidade de processamento, como o Eq_1.

O impacto do cache é um fator determinante para o desempenho das relações referenciadas. No modo centralizado, particularmente no equipamento Eq_1, o cache favorece significativamente as relações referenciadas, ampliando a diferença em relação às relações aninhadas. Por outro lado, nos equipamentos distribuídos (Eq_2 e Eq_3), a influência do cache é menos pronunciada, resultando em coeficientes mais equilibrados entre $coef_{ref}$ e $coef_{emb}$.

Tabela 4.17: Comparação de coeficientes do primeiro experimento com o caso de validação.

Equipamento	Com cache				Sem cache			
	$coef_{ref}$		$coef_{emb}$		$coef_{ref}$		$coef_{emb}$	
	Estimado	Validado	Estimado	Validado	Estimado	Validado	Estimado	Validado
Eq_1 centralizado	0.51	0.67	0.49	0.33	0.51	0.56	0.49	0.44
Eq_1 distribuído	0.52	0.65	0.48	0.35	0.52	0.66	0.48	0.34
Eq_2 distribuído	0.52	0.60	0.48	0.40	0.52	0.57	0.48	0.43
Eq_3 distribuído	0.52	0.64	0.48	0.36	0.51	0.60	0.49	0.40

A Tabela 4.17 mostra uma análise comparativa dos coeficientes de ponderação em diferentes configurações e equipamentos. Destacam-se a predominância das relações referenciadas em cenários de validação. Os valores de $coef_{ref}$ são consistentemente maiores em quase todos os experimentos. Isso sugere uma associação clara entre valores mais altos de coeficientes e as relações referenciadas, enquanto valores menores estão associados às relações aninhadas. No entanto, essa observação não permite concluir que as relações referenciadas sejam, de fato, mais eficientes do que as aninhadas em todos os cenários.

Os coeficientes de ponderação $coef_{ref}$ e $coef_{emb}$, obtidos a partir dos modelos de regressão múltipla, fornecem uma indicação da associação entre o desempenho e os esquemas de modelagem referenciado e aninhado. Eles representam apenas associações observadas entre os tempos de resposta e os tipos de relação avaliados nos experimentos. Assim, o usuário deve interpretar os coeficientes como uma ferramenta de apoio à decisão, considerando cuidadosamente as características do ambiente, o padrão de consultas e as necessidades específicas do sistema.

4.9 Considerações Finais

Experimentos realizados confirmaram que a recuperação de dados em relacionamentos aninhados é mais rápida do que em relacionamentos referenciados, resultado corroborado pelos modelos de regressão múltipla. Esses resultados estão alinhados com a literatura (Reis et al., 2018)(Gómez et al., 2016)(Shah et al., 2022), que destaca a eficiência de estruturas aninhadas devido à co-localização de dados relacionados em um único documento, reduzindo a necessidade de operações de união ou consultas múltiplas.

Para quantificar a diferença de desempenho e apoiar decisões na etapa de modelagem, foram calculados coeficientes de ponderação baseados nos tempos médios de recuperação de dados para relacionamentos referenciados ($\overline{ref}$) e aninhados ($\overline{emb}$). Esses tempos foram submetidos à descomposição porcentual normalizada, gerando os coeficientes $coef_{ref}$ e $coef_{emb}$.

Esses coeficientes representam pesos relativos que refletem a contribuição de cada tipo de relacionamento para o tempo total de recuperação de dados. A interpretação desses coeficientes é crucial. Um valor menor para $coef_{emb}$ indica maior eficiência das estruturas aninhadas em termos de velocidade de consulta, enquanto um $coef_{ref}$ mais elevado reflete o maior custo temporal associado às relações referenciadas.

A partir dos experimentos realizados, os valores médios de $coef_{ref} = 0{,}6$ e $coef_{emb} = 0{,}4$ foram estabelecidos como coeficientes de ponderação para relacionamentos referenciados e aninhados, respectivamente. Esses valores foram obtidos como a média de todos os $coef_{ref}$ e $coef_{emb}$ calculados em diferentes equipamentos e arquiteturas utilizadas, com e sem uso de cache (Tabela 4.2). No capítulo subsequente, esses coeficientes serão aplicados como ponderações de relacionamentos na avaliação de diferentes esquemas.

Foram determinados coeficientes de ponderação para atributos simples $coef_s$ (0.51) e complexos $coef_c$ (0.49) a partir de consultas diretas sobre esquemas nos modos centralizado e distribuído, medindo os tempos de resposta. Cada atributo foi classificado como simples ou complexo e os tempos médios de recuperação foram calculados e normalizados para obter os coeficientes. No entanto, é necessário aprofundar a análise para definir coeficientes específicos para cada tipo de dado simples (inteiro, *double*, *float*, *string*, data, etc.), pois cada um possui tempos distintos de recuperação. O mesmo ocorre com atributos complexos, cujo tamanho variável também influencia no tempo de resposta. Portanto, uma análise mais detalhada é essencial para melhorar a precisão dos coeficientes.

Capítulo 5

Métricas de Avaliação

Neste capítulo, as métricas de avaliação que são usados para encontrar o modelo de dados ótimos são descritos. São apresentadas quatro métricas específicas para a avaliação de esquemas (Gómez et al., 2021) (Kuszera et al., 2020): *completude*, que verifica se os esquemas satisfazem as consultas estabelecidas; *padrão de acesso*, que avalia a organização dos relacionamentos, diferenciando entre referenciados e aninhados; *custo de recuperação*, que mede o impacto da estrutura dos relacionamentos e atributos no desempenho; e *redundância*, que identifica a duplicação de dados entre esquemas. Essas métricas utilizam coeficientes de ponderação para capturar a complexidade real, considerando tanto os aspectos estruturais quanto as interações entre consultas e esquemas.

A Seção 5.1 apresenta as definições formais necessárias para o entendimento das métricas propostas. Na Seção 5.2, é apresentada a definição da métrica de *completude*. A Seção 5.3 descreve a métrica de *padrão de acesso*. Na Seção 5.4, é definida a métrica de *custo de recuperação* dos esquemas. A Seção 5.5 apresenta a métrica de *redundância*, explicando seu funcionamento e a interpretação dos resultados. Finalmente, a Seção 5.6 apresenta dois cenários de validação das métricas propostas.

5.1 Definições Formais

No contexto dos sistemas de bancos de dados `NoSQL` orientados a documentos a ausência de um esquema ideal pode levar a problemas como redundância de dados, alto custo de recuperação de informações e dificuldade de acessibilidade. Para enfrentar esses desafios, as métricas desempenham um papel central ao possibilitar a avaliação, comparação e otimização de esquemas com base em características fundamentais, incluindo a estrutura dos relacionamentos e a natureza dos atributos, bem como a capacidade de atender às consultas definidas.

A avaliação de esquemas requer uma definição de elementos fundamentais que compõem o cenário de análise. Para este propósito, considera-se um modelo baseado em três conjuntos principais: coleções, esquemas e consultas. Esses conjuntos são formalmente descritos a seguir.

O conjunto das coleções $\mathtt{C}$ é composto por todas as coleções envolvidas no caso de uso analisado, abrangendo tanto aquelas utilizadas nas consultas quanto as presentes nos esquemas. Formalmente, $\mathtt{C} = \{c_1, c_2, ..., c_k\}$, onde $k \geq 1$ representa o número de coleções existentes.

A Tabela 5.1 e a Figura 5.1 descrevem um cenário composto por quatro consultas distintas atuando sobre o esquema e_1. O conjunto completo de coleções $\mathtt{C}$ é definido como $\mathtt{C} = \{A, B, C, D\}$. Ressalta-se que, embora o esquema e_1 este composto apenas pelas coleções A e B, o conjunto $\mathtt{C}$ mantém sua cardinalidade original ($|\mathtt{C}| = 4$), pois as consultas formuladas exigem o acesso às quatro coleções para seu processamento integral.

Tabela 5.1: Consultas definidas envolvendo quatro coleções.

N^o	**Descrição**
1	consulta envolvendo as coleções A e B
2	consulta envolvendo as coleções B e C
3	consulta envolvendo as coleções A e C
4	consulta envolvendo as coleções B, C e D

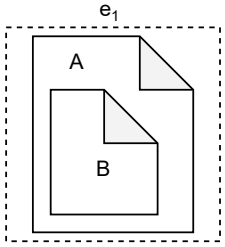


Figura 5.1: Esquema e_1 composto de duas coleções

No cenário apresentado na Tabela 5.2, que contém apenas uma consulta, e na Figura 5.2, observa-se que a consulta envolve apenas duas coleções. No entanto, os esquemas e_1 e e_2 consideram o uso de três coleções: A, B e C. Dessa forma, o conjunto $\mathtt{C}$ é definido como: $\mathtt{C} = \{A, B, C\}$.

Tabela 5.2: Consulta definida envolvendo duas coleções.

N^o	**Descrição**
1	consulta envolvendo as coleções A e B

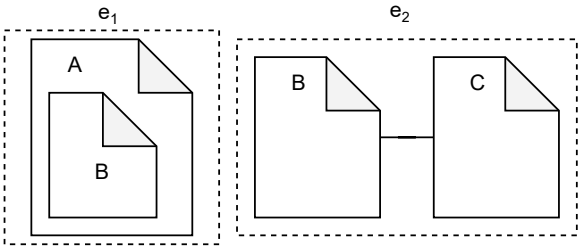


Figura 5.2: Esquema e_1 e e_2 composto de três coleções A, B e C.

O conjunto de esquemas E representa os esquemas utilizados para dar suporte às consultas, sendo definido como $\mathtt{E} = \{e_1, e_2, ..., e_m\}$. Cada esquema $e_j \in \mathtt{E}$ é um subconjunto não vazio de C, ou seja, $e_j \subseteq \mathtt{C}$ e $|e_j| \geq 1$ para $j = 1, 2, ..., m$ com $m \geq 1$, onde m indica o número total de esquemas.

A Tabela 5.3 e a Figura 5.3 apresentam um cenário composto por três esquemas, e_1, e_2 e e_3, e por duas consultas. Assim, o conjunto E é definido como: $\mathtt{E} = e_1, e_2, e_3$. Cada um desses esquemas é formado por coleções que pertencem ao conjunto $\mathtt{C} = A, B, C$.

Tabela 5.3: Consultas definidas envolvendo três coleções.

N^o	**Descrição**
1	consulta envolvendo as coleções A e B
2	consulta envolvendo as coleções B e C

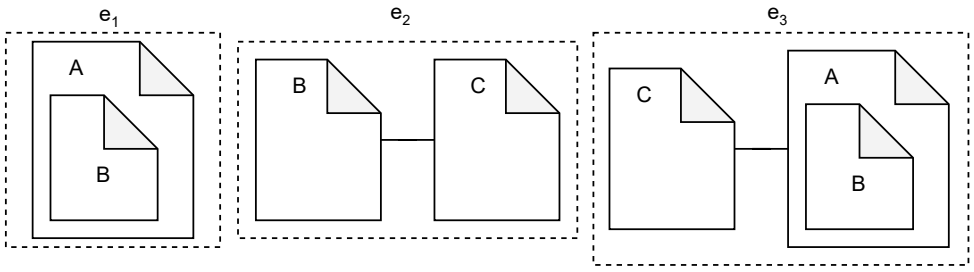


Figura 5.3: Conjunto de esquemas

O conjunto de consultas Q, definido como $\mathtt{Q} = \{q_1, q_2, ..., q_n\}$, contém as consultas previamente definidas que devem ser atendidas pelos esquemas. Cada consulta $q_i \in \mathtt{Q}$ é também um subconjunto não vazio de C, ou seja $q_i \subseteq \mathtt{C}$ e $|q_i| \geq 1$ para $i = 1, 2, ..., n$ com $n \geq 1$ representando o número total de consultas.

A interação entre os conjuntos C, E e Q define o escopo de análise das métricas propostas. Cada esquema e_j é formado por coleções pertencentes ao conjunto C e deve ser capaz de suportar um ou mais subconjuntos dessas coleções definidas pelas consultas q_i.

5.2 Métrica Completude

A métrica *completude* é responsável por avaliar se um conjunto de esquemas de $\mathtt{E}$ é capaz de atender a todas as consultas definidas no conjunto $\mathtt{Q}$. Essa métrica é fundamental para garantir que as consultas previstas possam ser processadas corretamente pelos esquemas disponíveis, considerando a existência das coleções e a definição de seus relacionamentos (esquemas).

A *completude* é uma métrica que assume valores no intervalo de 0 a 1, conforme o grau de atendimento às consultas definidas. Quando todas as consultas do conjunto $\mathtt{Q}$ são atendidas pelos esquemas $\mathtt{E}$, a métrica retorna 1, indicando cobertura integral. Quando nenhuma consulta é atendida, o valor é 0, refletindo ausência de cobertura. Nos demais casos, retorna um valor intermediário entre 0 e 1, representando a proporção de consultas atendidas. Essa abordagem permite avaliar tanto a aderência total quanto parcial dos esquemas às necessidades do sistema.

A *completude* é calculada considerando a correspondência entre as consultas $q_i \in \mathtt{Q}$ e os esquemas $e_j \in \mathtt{E}$. Formalmente, a métrica *completude* para um conjunto de esquemas $\mathtt{E}$ é definida como:

$$completude(\mathtt{E}) = \frac{Q_a}{m} \tag{5.1}$$

Onde:

- Q_a (consultas atendidas): é o número de consultas $q_i \in \mathtt{Q}$ que são atendidas por pelo menos um esquema $e_j \in \mathtt{E}$;
- m: é o número total de consultas no conjunto $\mathtt{Q}$;

A função que verifica se uma consulta q_i é atendida é dada por:

$$atende(\mathtt{E}, q_i) = existeColecao(\mathtt{E}, q_i) \times existeRelacionamento(\mathtt{E}, q_i) \tag{5.2}$$

Aqui:

- $existeColecao(\mathtt{E}, q_i)$: Verifica se todas as coleções que compõem q_i estão presentes em pelo menos um esquema $e_j \in \mathtt{E}$. Retorna 1 se as coleções existem e 0 caso contrário;
- $existeRelacionamento(\mathtt{E}, q_i)$: Verifica se os relacionamentos entre as coleções de q_i estão definidos em e_j. Retorna 1 se os relacionamentos existem e 0 caso contrário

A soma de todas as chamadas de $atende(\mathtt{E}, q_i)$ para $q_i \in Q$ resulta no valor Q_a. É formalmente representada pela equação:

$$Q_a = \sum atende(\mathtt{E}, q_i), \quad \forall q_i \in \mathtt{Q} \tag{5.3}$$

5.2.1 Funcionamento

O funcionamento do processo de avaliação de esquemas para atender às consultas é detalhado nos seguintes passos:

- Para cada consulta q_i, verifica-se se as coleções requeridas $\{c_1, c_2, ..., c_k\} \subseteq \mathtt{C}$ estão presentes nos esquemas $\mathtt{E}$. Essa avaliação é essencial para garantir que os dados exigidos pela consulta estejam disponíveis;
- Além da presença das coleções, é imprescindível confirmar que os relacionamentos entre elas, sejam referenciados ou aninhados, estejam devidamente definidos nos esquemas. A existência das coleções não é suficiente para garantir o suporte às consultas, pois, para que uma consulta seja efetivamente atendida, deve haver um caminho válido que conecte as coleções envolvidas. Esse caminho é essencial para permitir que os dados possam ser acessados e combinados conforme necessário para produzir a resposta esperada. Assim, após verificar a presença das coleções requeridas, é fundamental garantir que os relacionamentos entre elas estejam definidos;
- Para cada consulta q_i, calcula-se se ela foi atendida ($atende(\mathtt{E}, q_i) = 1$) e, ao final, soma-se o número de consultas definidas (Q_a). A métrica de *Completude* é, então, derivada como uma proporção do número de consultas atendidas pelo número total de consultas definidas

5.2.2 Exemplo

Considerando o cenário apresentado na Tabela 5.2 e na Figura 5.2, definem-se os conjuntos $\mathtt{C}$, $\mathtt{E}$ e $\mathtt{Q}$. O conjunto $\mathtt{C}$ é composto pelas coleções participantes do cenário, sendo definido como $\mathtt{C} = \{A, B, C\}$, uma vez que há três coleções envolvidas tanto nos esquemas e nas consultas definidas. O conjunto $\mathtt{E}$ é definido como $\mathtt{E} = \{e_1, e_2\}$, dado que o cenário apresenta apenas dois esquemas distintos. Por fim, o conjunto $\mathtt{Q}$ é estabelecido como $\mathtt{Q} = \{q_1\}$, onde q_1 representa a consulta 1 especificada na Tabela 5.2.

Para calcular a completude do cenário, é necessário inicialmente determinar os valores das funções *existeColecao*() e *existeRelacionamento*(). A Tabela 5.4 apresenta a análise de existência das coleções da consulta q_1 nos esquemas definidos no cenário. Observa-se que as coleções acessadas pela consulta q_1 estão presentes no esquema e_1, porém não estão totalmente disponíveis no esquema e_2. Contudo, considerando que a consulta pode ser analisada desde que pelo menos um dos esquemas contenha as coleções requeridas, é

possível prosseguir para a próxima etapa da avaliação. Finalmente, o valor retornado pela função *existeColeção*() será 1.

Tabela 5.4: Verificação da existência das coleções de q_1 nos esquemas e_1 e e_2.

		e_1	e_2
q_1	A	✓	
	B	✓	✓

Tabela 5.5: Verificação da existência de relacionamentos que atendam as coleções de q_1 nos esquemas e_1 e e_2.

		e_1	e_2
q_1	A	✓	
	B	✓	✓

A Tabela 5.6 apresenta a análise da existência de relacionamentos que suportam as coleções A e B da consulta q_1. Observa-se que apenas o esquema e_1 possui um relacionamento aninhado entre as coleções mencionadas. No entanto, considerando que a consulta q_1 é atendida por pelo menos um dos esquemas pertencentes ao conjunto analisado, a função *existeRelacionamento*() assume o valor 1, indicando a viabilidade da execução da consulta no cenário avaliado.

Tabela 5.6: Verificação da existência de relacionamentos que atendam as coleções de q_1 nos esquemas e_1 e e_2.

	e_1	e_2
q_1 (A,B)	✓	

Finalmente, como as funções *existeColecao*() e *existeRelacionamento*() retornam o valor 1, a métrica de completude também assume o valor 1. Esse resultado indica que o conjunto de esquemas avaliados é capaz de suportar plenamente as consultas definidas pelo usuário no cenário considerado.

5.2.3 Interpretação dos Resultados

A interpretação dos valores da métrica de *completude* obtidos na avaliação dos esquemas `E` em relação ao conjunto de consultas `Q` é resumida da seguinte forma:

- Um valor de *completude* igual a 1 indica que todos os esquemas em `E` são capazes de atender a 100% das consultas em `Q`. Esse é o cenário ideal, demonstrando que o conjunto de esquemas é completamente funcional para o caso de uso especificado;

- Quando o valor de completude está entre 0 e 1, significa que algumas consultas em `Q` não podem ser atendidas pelos esquemas disponíveis. Isso pode ser causado pela ausência de coleções, falta de relacionamentos ou definições incompletas nos esquemas;
- Um valor de completude igual a 0 reflete que nenhuma consulta em `Q` é suportada pelos esquemas `E`. Esse cenário é crítico e indica falhas no desenho ou incompatibilidades entre os esquemas e as consultas

5.3 Métrica Padrão de Acesso

A métrica *padrão de acesso* é projetada para avaliar como as coleções estão organizadas dentro dos esquemas em relação aos seus relacionamentos, sejam eles referenciados ou aninhados. Essa métrica é especialmente importante em bancos de dados NoSQL orientados a documentos, onde a estrutura dos relacionamentos impacta diretamente o desempenho das consultas, a escalabilidade e o custo de recuperação de dados.

O objetivo do *padrão de acesso* é medir a eficiência estrutural dos esquemas com base na forma como os relacionamentos entre as coleções são definidos. A métrica considera os dois tipos principais de relacionamentos referenciados e aninhados. Cada tipo de relacionamento possui implicações diferentes em termos de desempenho e complexidade, e a métrica utiliza *coeficientes de ponderação* para capturar essas diferenças.

A métrica *padrão de acesso* para um conjunto de esquemas `E` é definida como:

$$padraoAcesso(\texttt{E}) = contarRel(\texttt{E}) \tag{5.4}$$

Onde $padraoAcesso(\texttt{E})$ é o somatório dos relacionamentos ponderados em todos os esquemas $e_j \in \texttt{E}$:

$$contarRel(\texttt{E}) = \sum contar(e_j), \quad \forall e_j \in \texttt{E} \tag{5.5}$$

Para cada esquema e_j, a função $contar(e_j)$ contabiliza os relacionamentos referenciados (rel_{ref}) e aninhados (rel_{ani}), aplicando os respetivos coeficientes de ponderação $coef_{ref}$ e $coef_{emb}$:

$$contar(e_j) = \sum rel_{ref} \times coef_{ref} + \sum rel_{ani} \times coef_{emb} \tag{5.6}$$

5.3.1 Funcionamento

O processo para determinar o *padrão de acesso* do conjunto de esquemas `E` envolve os seguintes passos:

- Cada esquema e_j é analisado para identificar todos os relacionamentos definidos entre suas coleções, categorizando-os como referenciados ou aninhados;
- Cada tipo de relacionamento recebe um peso específico, determinado pelos coeficientes de ponderação $coef_{ref}$ para relacionamentos referenciados e $coef_{emb}$ para relacionamentos aninhados, previamente calculados e definidos na Seção 4.6. Esses coeficientes refletem o custo relativo de acesso a cada tipo de relacionamento, comumente baseado em análises de tempo de resposta e eficiência;
- O número total de relacionamentos ponderados é calculado para cada esquema e_j, e os resultados são somados para obter o padrão de acesso do conjunto `E`

5.3.2 Exemplo

Considerando o cenário apresentado na Tabela 5.3 e na Figura 5.3, são definidos os conjuntos `C`, `E` e `Q`. O conjunto `C` é composto pelas coleções envolvidas nas consultas e nos esquemas analisados, sendo definido como $\texttt{C} = \{A, B, C\}$. O conjunto `E` representa os esquemas do cenário e é definido como $\texttt{E} = \{e_1, e_2, e_3\}$. Por fim, o conjunto `Q` é composto pelas consultas consideradas e é definido como $\texttt{Q} = \{q_1, q_2\}$, onde q_1 corresponde à consulta 1 e q_2 à consulta 2, respectivamente.

A Tabela 5.7 apresenta a análise do padrão de acesso para os esquemas e_1, e_2 e e_3. Inicialmente, foram contabilizados os relacionamentos referenciados e aninhados presentes em cada esquema. Em seguida, foi realizado o cálculo ponderado utilizando os coeficientes de relacionamento $coef_{ref}$ (0,6) e $coef_{emb}$ (0,4) aplicados a cada esquema individualmente. O valor total da métrica corresponde à soma dos subtotais obtidos para cada esquema, resultando, conforme indicado na Tabela 5.7, em um total de 2 unidades.

Tabela 5.7: Tabela de análise padrão de acesso.

	Referenciado	Aninhado	Cálculo	Sub total
e_1	0	1	$0 \times 0.6 + 1 \times 0.4$	0.4
e_2	1	0	$1 \times 0.6 + 0 \times 0.4$	0.6
e_3	1	1	$1 \times 0.6 + 1 \times 0.4$	1

5.3.3 Interpretação dos Resultados

A seguir, detalha-se a interpretação dos diferentes valores que podem ser observados para o padrão de acesso dos esquemas:

- Valores mais altos indicam esquemas mais complexos, com um número elevado de relacionamentos, especialmente referenciados. Embora possam suportar estruturas de dados mais ricas, esses esquemas tendem a apresentar maior custo de navegação e recuperação de dados;
- Valores mais baixos representam esquemas mais simples, geralmente com menos relacionamentos ou predominância de relacionamentos aninhados. Esses esquemas podem ser mais rápidos para recuperar dados, mas podem ter limitações em termos de flexibilidade para consultas complexas

5.4 Métrica Custo de Recuperação

A métrica de *custo de recuperação* avalia o impacto da estrutura e do tamanho das coleções no esforço necessário para acessar os dados. Considera os relacionamentos (referenciados e aninhados) e os atributos (simples e complexos) definidos nos esquemas, utilizando coeficientes de ponderação que refletem a influência de cada componente na recuperação de informações.

O objetivo é quantificar o custo associado à navegação e ao acesso a dados em esquemas de bancos de dados `NoSQL` orientados a documentos, permitindo identificar estruturas que podem introduzir latência ou complexidade desnecessária em operações frequentes e fornecendo bases para otimizações.

O custo de recuperação para um conjunto de esquemas `E` é calculado como a soma do custo dos relacionamentos e dos atributos em todos os esquemas $e_j \in \texttt{E}$. Formalmente:

$$custoRecuperacao(\texttt{E}) = contarRel(\texttt{E}) + contarAtr(\texttt{E}) \tag{5.7}$$

Onde:

- $contarRel(\texttt{E})$, conta os relacionamentos do esquema `E` (Equação 5.5);
- $contarAtr(\texttt{E})$: Soma dos atributos ponderados em todos os esquemas e_j

A função $contarAtr(\texttt{E})$ é definida como:

$$contarAtr(\texttt{E}) = \sum contarAtributos(e_j), \quad \forall e_j \in \texttt{E} \tag{5.8}$$

Para cada esquema e_j, a função $contarAtributos(e_j)$ contabiliza os atributos simples e complexos utilizando coeficientes de ponderação $coef_s$ e $coef_c$ (definidos na Seção 4.7), respectivamente:

$$contarAtr(e_j) = \sum attr_{simple} \times coef_s + \sum attr_{complexo} \times coef_c \tag{5.9}$$

5.4.1 Funcionamento

O cálculo do *custo recuperação* para o conjunto de esquemas `E` é realizado seguindo os seguintes passos:

- Os relacionamentos entre as coleções são contabilizados, conforme definido na métrica padrão de acesso (Equação 5.5). Essa etapa reflete o custo estrutural associado à navegação entre documentos referenciados ou aninhados;
- Para cada esquema e_j, os atributos simples e complexos são identificados e ponderados de acordo com os coeficientes $coef_s$ e $coef_c$, que refletem a diferença no esforço de processamento entre os dois tipos de atributos;
- Os custos de relacionamentos e atributos são somados para cada esquema, e o valor resultante é utilizado para calcular o custo total de recuperação do conjunto `E`

5.4.2 Exemplo

Com base no cenário apresentado na Tabela 5.1 e na Figura 5.1, posteriormente atualizada com atributos na Figura 5.4, definem-se os conjuntos `C`, `E` e `Q`. O conjunto `C` é composto pelas coleções envolvidas nas consultas e nos esquemas do cenário, sendo definido como $\texttt{C} = \{A, B, C, D\}$. O conjunto `E` corresponde aos esquemas considerados e é definido como $\texttt{E} = \{e_1\}$. Por fim, o conjunto `Q` representa as consultas analisadas, sendo estabelecido como $\texttt{Q} = \{q_1, q_2, q_3, q_4\}$.

Para determinar a métrica de custo de recuperação, é necessário calcular as funções $contarRel()$ e $contarAtr()$. A função $contarRel()$ é obtida de forma equivalente ao cálculo realizado na métrica de padrão de acesso. A Tabela 5.8 apresenta os resultados obtidos para a função $contarRel()$ no cenário analisado.

Tabela 5.8: Tabela de análise padrão de acesso.

	Referenciado	Aninhado	Cálculo	Sub total
e_1	0	1	$0 \times 0.6 + 1 \times 0.4$	0.4

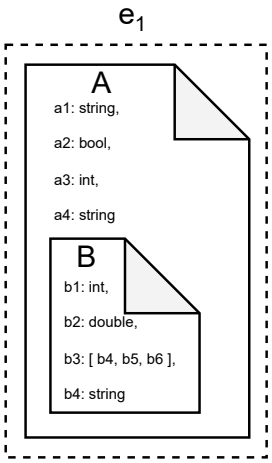


Figura 5.4: Esquema e_1 com atributos.

A Tabela 5.9 apresenta os resultados obtidos a partir do cálculo da função *contarAtr*(). Conforme indicado, foram contabilizados sete atributos simples distribuídos entre as coleções A e B, além de um atributo complexo, denominado "b3", presente na coleção B. O cálculo totaliza 4,06 unidades para essa função. Considerando os resultados anteriores, o valor final da métrica de custo de recuperação é obtido pela soma dos valores das funções *contarRel*() (0,4) e *contarAtr*() (4,06), resultando em um custo total de 4,46 unidades.

Tabela 5.9: Tabela de análise função *contarAtr*().

	Simples	Complexo	Cálculo	Sub total
e_1	7	1	$7 \times 0.51 + 1 \times 0.49$	4.06

5.4.3 Interpretação de Resultados

A seguir, apresenta-se a interpretação dos valores calculados para o custo de recuperação:

- Valores mais altos indicam esquemas mais complexos, com um grande número de relacionamentos e/ou atributos complexos. Esquemas com valores elevados podem apresentar maior latência e custo computacional para recuperar dados;
- Valores mais baixos representam esquemas mais simples, geralmente com menos relacionamentos ou predominância de atributos simples. Esses esquemas tendem a ser mais eficientes em termos de custo de recuperação, mas podem ter limitações na modelagem de dados complexos

5.5 Métrica Redundância

A métrica de *redundância* quantifica a duplicação de dados entre coleções nos esquemas de um banco de dados NoSQL orientado a documentos. Esta métrica identifica coleções repetidas em diferentes esquemas, fornecendo uma medida objetiva da replicação de dados. Essa métrica é essencial, pois a redundância pode aumentar o consumo de armazenamento, impactar negativamente o desempenho e comprometer a integridade dos dados.

A métrica de *redundância* para um conjunto de esquemas $\mathtt{E}$ é calculada como a soma das coleções repetidas entre cada esquema e_j e os demais esquemas $(\mathtt{E} - \{e_j\})$ do conjunto $\mathtt{E}$. Formalmente:

$$redundancia(\mathtt{E}) = \sum repetidas(\mathtt{E} - \{e_j\}, e_j), \quad \forall e_j \in \mathtt{E} \tag{5.10}$$

Onde:

- $repetidas(\mathtt{E} - \{e_j\}, e_j)$ conta as coleções repetidas do esquema e_j em $\mathtt{E}$.

5.5.1 Funcionamento

A metodologia para calcular a *redundância* de coleções no conjunto de esquemas $\mathtt{E}$ envolve os seguintes passos:

- Para cada esquema e_j, identifica-se o conjunto de coleções que o compõem;
- As coleções de e_j são comparadas com as coleções presentes nos demais esquemas $(\mathtt{E} - \{e_j\}, e_j)$ verificando duplicações exatas;
- O número de coleções duplicadas para cada esquema é somado, resultando no valor total da métrica para o conjunto $\mathtt{E}$

5.5.2 Exemplo

Considerando o cenário apresentado na Tabela 5.3 e na Figura 5.3, são definidos os conjuntos $\mathtt{C}$, $\mathtt{E}$ e $\mathtt{Q}$. O conjunto $\mathtt{C}$ representa as coleções envolvidas no cenário e é definido como $\mathtt{C} = \{A, B, C\}$. O conjunto $\mathtt{E}$ corresponde aos esquemas analisados, sendo definido como $\mathtt{E} = \{e_1, e_2, e_3\}$. Por fim, o conjunto $\mathtt{Q}$, que reúne as consultas consideradas, é definido como $\mathtt{Q} = \{q_1, q_2\}$.

A Tabela 5.10 apresenta os resultados do cálculo da redundância no conjunto de esquemas $\mathtt{E}$. Conforme indicado, as coleções A e C apresentam uma ocorrência redundante cada, enquanto a coleção B se repete duas vezes. O valor da métrica de redundância é obtido pela soma das repetições identificadas, resultando em um total de 4 unidades.

Tabela 5.10: Tabela de análise de redundância.

A	B	C	Sub total
1	2	1	4

5.5.3 Interpretação dos Resultados

A análise dos valores obtidos para a métrica de *redundância* de coleções no conjunto de esquemas `E` permite as seguintes interpretações:

- Valores mais altos indicam um alto nível de redundância, com muitas coleções duplicadas entre os esquemas. Esse cenário pode causar desperdício de recursos e dificuldade na manutenção de dados consistentes;
- Valores mais baixos representam esquemas mais eficientes em termos de armazenamento, com baixa ou nenhuma redundância

5.6 Cenários de Validação

Nesta seção, são apresentados dois estudos de caso para validar as métricas propostas. O objetivo é demonstrar a aplicabilidade das métricas em diferentes cenários de avaliação de esquemas de bancos de dados `NoSQL` orientados a documentos. Cada cenário aborda conjuntos específicos de coleções, consultas e esquemas, possibilitando uma análise prática das métricas de *completude*, *padrão de acesso*, *custo de recuperação* e *redundância*.

Para analisar cada cenário, segue-se o seguinte processo: primeiro, são analisadas as consultas para calcular a métrica da *completude*, utilizando a Equação 5.1. Em seguida, são analisados os esquemas para calcular as métricas de *padrão de acesso*, *custo de recuperação* e *redundância*. Uma vez que as métricas que avaliam os esquemas estão baseadas na análise dos relacionamentos, atributos e coleções, uma análise prévia é realizada para o cálculo de $contarRel(\texttt{E})$, $contarAtr(\texttt{E})$ e $repetidas(\texttt{E} - \{e_j\}, e_j)$. Os valores dos coeficientes foram obtidos através de uma análise de tempos de resposta com regressão múltipla, aplicando o método proposto na Seção 4.6. Assim, considera-se $coef_{ref}$ com o valor de 0.6 e o valor de 0.4 para o $coef_{emb}$. Enquanto isso, os valores para os coeficientes de ponderação de atributos foram 0.51 para $coef_s$ e 0.49 para $coef_c$.

5.6.1 Primeiro Cenário de Avaliação

O primeiro estudo de caso é baseado no cenário descrito por Gómez et al. (2018) e consiste em três coleções: *empresa*, *departamento* e *funcionário*. A escolha desse caso de uso não

foi aleatória, pois trata-se de um cenário já consolidado na literatura, no qual os autores também aplicam métricas para avaliar diferentes alternativas de modelagem em bancos de dados NoSQL. Por esse motivo, esse caso se mostrou particularmente relevante para o presente trabalho, ao permitir a replicação e a comparação direta dos resultados obtidos, validando a eficácia da abordagem proposta com base em um referencial comum.

São definidas sete consultas apresentadas na Tabela 5.11, cujos objetivos variam desde a busca por atributos específicos até a agregação de informações entre as coleções. Nove soluções foram geradas para suportar essas consultas, cada solução contém, no mínimo, um esquema, sendo que algumas podem incluir mais de um, conforme ilustrado na Figura 5.5. Por exemplo, a solução $\mathtt{E}_6$ é composta por três esquemas distintos (e_1, e_2, e_3).

Tabela 5.11: Consultas definidas para o primeiro cenário.

No	**Consulta**	**Coleções**
q_1	*Funcionários com um salário igual a $1000*	funcionário
q_2	*Funcionários com um salário superior a $1000*	funcionário
q_3	Funcionários com o maior salário	funcionário
q_4	*Funcionários com o maior salário por empresa e o ID da empresa*	funcionário, empresa
q_5	*Funcionários com o maior salário por empresa e o nome da empresa*	funcionário, empresa
q_6	*O salário mais alto*	funcionário
q_7	*Informações das empresas, incluindo o nome de seus departamentos*	empresa, departamento

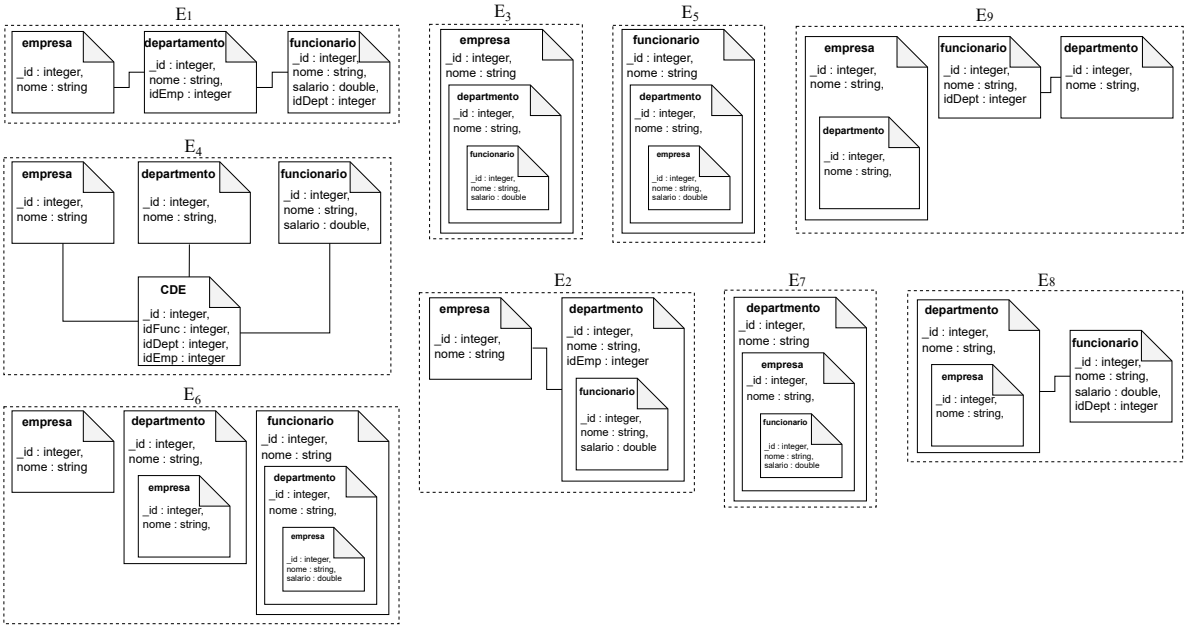


Figura 5.5: Esquemas para o primeiro cenário (link para os esquemas).

Avaliação da métrica completude

A Tabela 5.12 apresenta os resultados da análise da métrica *completude* para cada consulta q_i nas nove soluções definidas no primeiro cenário. A coluna "eC" avalia a existência das coleções que participam de uma consulta q_i em pelo menos um esquema e_j de $\mathtt{E}$ ($existeColecao(\mathtt{E}, q_i)$), enquanto a coluna "eR" avalia a existência de um relacionamento entre as coleções de uma consulta q_i em pelo menos um esquema e_j de $\mathtt{E}$. Na coluna "R", é avaliado $atende(\mathtt{E}, q_i) = existeColecao(\mathtt{E}, q_i) \times existeRelacionamento(\mathtt{E}, q_i)$.

A Tabela 5.13 apresenta os resultados da métrica *completude* para cada esquema. Os resultados mostram que os esquemas e_1 a e_8 atendem integralmente a todas as consultas definidas. Por outro lado, o esquema e_9 apresenta limitações ao não atender às consultas q_4 e q_5. Essa falha decorre da ausência de um relacionamento que conecte adequadamente as coleções envolvidas nessas consultas.

Tabela 5.12: Análise de $existeColecao(\mathtt{E}, q_i)$ e $existeRelacionamento(\mathtt{E}, q_i)$ para cada consulta nos nove esquemas.

	$\mathtt{E}_1$			$\mathtt{E}_2$			$\mathtt{E}_3$			$\mathtt{E}_4$			$\mathtt{E}_5$			$\mathtt{E}_6$			$\mathtt{E}_7$			$\mathtt{E}_8$			$\mathtt{E}_9$		
	eC	eR	R	eC	eR	R	eC	eR	R	eC	eR	R	eC	eR	R	eC	eR	R	eC	eR	R	eC	eR	R	eC	eR	R
q_1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
q_2	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
q_3	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
q_4	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0
q_5	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0
q_6	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
q_7	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Tabela 5.13: Resultados da métrica completude para as nove soluções.

$\mathtt{E}_1$	$\mathtt{E}_2$	$\mathtt{E}_3$	$\mathtt{E}_4$	$\mathtt{E}_5$	$\mathtt{E}_6$	$\mathtt{E}_7$	$\mathtt{E}_8$	$\mathtt{E}_9$
1	1	1	1	1	1	1	1	0.7

Avaliação da métrica padrão de acesso

Para calcular o da métrica *padrão de acesso*, analisamos inicialmente os relacionamentos de cada esquema e_j da solução $\mathtt{E}$. A Tabela 5.14 apresenta a análise dos relacionamentos referenciados e aninhados. Por exemplo, a solução $\mathtt{E}_1$ possui um único esquema, com dois relacionamentos referenciados e nenhum aninhado. Para o modelo $\mathtt{E}_1$, ao aplicar os coeficientes de ponderação para cada tipo de relacionamento, obtém-se um valor final para o *padrão de acesso* de 1.2. Por outro lado, a solução $\mathtt{E}_6$ é composta por três esquemas(e_1, e_2 e e_3). O esquema e_1 não apresenta relacionamentos, enquanto e_2 e e_3 possuem, respectivamente, um e dois relacionamentos aninhados. Após a aplicação dos coeficientes de ponderação, o valor total do *padrão de acesso* para a solução $\mathtt{E}_6$ também é de 1.2. Esse

procedimento de análise e cálculo foi realizado para todas as soluções, permitindo uma comparação quantitativa da métrica *padrão de acesso* entre elas.

Tabela 5.14: Análise de *padrão de acesso* primeiro cenário.

Soluções	Esquemas	Relacionamentos referenciados	Relacionamentos Aninhados	Cálculo com coeficientes	Total
$\mathtt{E}_1$	e_1	2	0	$2 \times 0.6 + 0 \times 0.4 = 1.2$	1.2
$\mathtt{E}_2$	e_1	1	1	$1 \times 0.6 + 1 \times 0.4 = 1$	1
$\mathtt{E}_3$	e_1	0	2	$0 \times 0.6 + 2 \times 0.4 = 0.8$	0.8
$\mathtt{E}_4$	e_1	3	0	$3 \times 0.6 + 0 \times 0.4 = 1.8$	1.8
$\mathtt{E}_5$	e_1	0	2	$0 \times 0.6 + 2 \times 0.4 = 0.8$	0.8
$\mathtt{E}_6$	e_1	0	0	$0 \times 0.6 + 0 \times 0.4 = 0$	1.2
	e_2	0	1	$0 \times 0.6 + 1 \times 0.4 = 0.4$	
	e_3	0	2	$0 \times 0.6 + 2 \times 0.4 = 0.8$	
$\mathtt{E}_7$	e_1	0	2	$0 \times 0.6 + 2 \times 0.4 = 0.8$	0.8
$\mathtt{E}_8$	e_1	1	1	$1 \times 0.6 + 1 \times 0.4 = 1$	1
$\mathtt{E}_9$	e_1	0	1	$0 \times 0.6 + 1 \times 0.4 = 0.4$	1
	e_2	1	0	$1 \times 0.6 + 0 \times 0.4 = 0.6$	

A solução $\mathtt{E}_4$ apresenta o maior valor total, com 1.8. Isso ocorre porque o esquema e_1 de $\mathtt{E}_4$ contém três relacionamentos referenciados, cada um ponderado pelo coeficiente 0.6. Esse tipo de relacionamento geralmente implica maior complexidade, pois requer múltiplas operações para acessar os dados, aumentando o custo estrutural e a dificuldade de navegação. As soluções $\mathtt{E}_3$, $\mathtt{E}_5$ e $\mathtt{E}_7$ têm a menor pontuação total, com 0.8 cada. Essas soluções possuem esquemas com dois relacionamentos aninhados, cada um com peso de 0.4. Relacionamentos aninhados tendem a ser mais simples, pois os dados estão embutidos em um único documento, reduzindo o custo de navegação e tornando os esquemas menos complexos.

Avaliação da métrica custo de recuperação

Para o cálculo da métrica *custo de recuperação* no primeiro cenário, é necessário obter os valores de $contarRel(\mathtt{E})$ e $contarAtr(\mathtt{E})$ (Equação 5.7). Os valores de $contarRel(\mathtt{E})$, que avaliam os relacionamentos referenciados e aninhados, já foram calculados na etapa anterior durante a análise do *padrão de acesso*. Assim, nesta etapa, calcula-se exclusivamente $contarAtr(\mathtt{E})$, que considera os atributos simples e complexos definidos em cada esquema.

A Tabela 5.15 apresenta os resultados detalhados do cálculo de $contarAtr(\mathtt{E})$ para cada solução $\mathtt{E}$, utilizando os coeficientes de ponderação previamente definidos: $coef_s = 0.51$ para atributos simples e $coef_c = 0.49$ para atributos complexos. O *custo de recuperação* total é então obtido somando $contarRel(\mathtt{E})$ e $contarAtr(\mathtt{E})$.

Os resultados mostram que $\mathtt{E}_4$ apresenta o maior custo de recuperação total (7.41), devido ao maior número de relacionamentos referenciados e à predominância de atributos

Tabela 5.15: Análise de *custo de recuperação* primeiro cenário.

Soluções	Esquemas	Atributos simples	Atributos complexos	Cálculo com coeficientes	$contarAtr(E)$	$contarRel(E)$	Total
$\mathtt{E}_1$	e_1	9	0	$9 \times 0.51 + 0 \times 0.49 = 4.59$	4.59	1.2	5.79
$\mathtt{E}_2$	e_1	8	1	$8 \times 0.51 + 1 \times 0.49 = 4.57$	4.57	1.0	5.57
$\mathtt{E}_3$	e_1	7	2	$7 \times 0.51 + 2 \times 0.49 = 4.55$	4.55	0.8	5.35
$\mathtt{E}_4$	e_1	11	0	$11 \times 0.51 + 0 \times 0.49 = 5.61$	5.61	1.8	7.41
$\mathtt{E}_5$	e_1	7	2	$7 \times 0.51 + 2 \times 0.49 = 4.55$	4.55	0.8	5.35
$\mathtt{E}_6$	e_1	2	0	$2 \times 0.51 + 0 \times 0.49 = 1.02$	8.1	1.2	9.3
	e_2	4	1	$4 \times 0.51 + 1 \times 0.49 = 2.53$			
	e_3	7	2	$7 \times 0.51 + 2 \times 0.49 = 4.55$			
$\mathtt{E}_7$	e_1	7	2	$7 \times 0.51 + 2 \times 0.49 = 4.55$	4.55	0.8	5.35
$\mathtt{E}_8$	e_1	8	1	$8 \times 0.51 + 1 \times 0.49 = 4.57$	4.57	1.0	5.57
$\mathtt{E}_9$	e_1	4	1	$4 \times 0.51 + 1 \times 0.49 = 2.53$	5.08	1.0	6.08
	e_2	5	0	$5 \times 0.51 + 0 \times 0.49 = 2.55$			

simples. Por outro lado, $\mathtt{E}_3$, $\mathtt{E}_5$ e $\mathtt{E}_7$ têm valores mais baixos (5.35), devido à combinação equilibrada de relacionamentos aninhados e atributos complexos.

Avaliação da métrica redundância

A Tabela 5.16 apresenta a análise da métrica *redundância* no primeiro cenário, calculando o número de coleções repetidas em cada solução $\mathtt{E}$. A *redundância* ocorre quando uma mesma coleção aparece em mais de um esquema e_j dentro de uma solução $\mathtt{E}$, o que pode aumentar o consumo de armazenamento e comprometer a eficiência da gestão dos dados.

Tabela 5.16: Análise de coleções repetidas em cada esquema.

$\mathtt{E}_1$	$\mathtt{E}_2$	$\mathtt{E}_3$	$\mathtt{E}_4$	$\mathtt{E}_5$	$\mathtt{E}_6$	$\mathtt{E}_7$	$\mathtt{E}_8$	$\mathtt{E}_9$
0	0	0	0	0	3	0	0	1

As soluções $\mathtt{E}_1$, $\mathtt{E}_2$, $\mathtt{E}_3$, $\mathtt{E}_4$, $\mathtt{E}_5$, $\mathtt{E}_7$ e $\mathtt{E}_8$ apresentam valor 0 na contagem de coleções repetidas, indicando que seus esquemas não compartilham coleções duplicadas. Essa característica é desejável, pois minimiza o consumo de recursos e reduz o risco de inconsistências entre dados duplicados. A solução $\mathtt{E}_6$ possui 3 coleções repetidas, sendo a mais redundante entre todas. Essa repetição pode indicar um desenho que privilegia a duplicação para facilitar consultas específicas, mas com o custo de maior armazenamento e complexidade de manutenção. A solução $\mathtt{E}_9$ apresenta 1 coleção repetida. Embora menos redundante que $\mathtt{E}_6$, essa repetição ainda pode representar um uso ineficiente de recursos, especialmente se as consultas relacionadas puderem ser atendidas sem a duplicação.

Finalmente, a Tabela 5.17 apresenta os resultados consolidados das métricas para cada solução no primeiro cenário. As métricas avaliadas incluem padrão de acesso, custo de recuperação, redundância e completude, além de um valor total que combina os resultados das três primeiras métricas.

Tabela 5.17: Resultados final das métricas no primeiro cenário.

Soluções	Padrão acesso	Custo recuperação	Redundância	Total	Completude
E_1	1.2	5.79	0	6.99	1
E_2	1	5.57	0	6.57	1
E_3	0.8	5.35	0	6.15	1
E_4	1.8	7.41	0	9.21	1
E_5	0.8	5.35	0	6.15	1
E_6	1.2	9.3	3	13.5	1
E_7	0.8	5.35	0	6.15	1
E_8	1	5.57	0	6.57	1
E_9	1	6.08	1	8.08	0.7

A métrica *padrão de acesso* varia entre 0.8 e 1.8, com E_4 apresentando o maior valor (1.8) devido ao maior número de relacionamentos referenciados, o que aumenta a complexidade estrutural. Soluções como E_3, E_5 e E_7 possuem os menores valores (0.8), refletindo maior simplicidade com predominância de relacionamentos aninhados.

A métrica *custo de recuperação* é mais elevado em E_6 (9.3), consequência de múltiplos esquemas com alta presença de atributos simples e complexos. Em contrapartida, soluções como E_3, E_5 e E_7 apresentaram o menor custo (5.35), indicando um equilíbrio entre os tipos de atributos.

A métrica *redundância* é nula na maioria das soluções, com exceção de E_6 e E_9. E_6 é a solução mais redundante (3), enquanto E_9 apresenta uma redundância moderada (1), sugerindo a necessidade de otimizações no desenho estrutural para evitar coleções repetidas.

A métrica *completude* foi plenamente atingida (1) por quase todas as soluções, exceto E_9, que obteve um valor de 0.7. Essa limitação ocorre devido à falta de relacionamentos necessários para atender a todas as consultas E_4 e E_5.

O valor total é uma soma da métrica *padrão de acesso*, *custo de recuperação* e *redundância*, destacando-se E_6 como a solução com maior valor (13.5), devido à alta redundância e custo de recuperação. Em contrapartida, E_3, E_5 e E_7 apresentam os menores valores (6.15).

5.6.2 Segundo Cenário de Avaliação

Nesta seção, são apresentados os resultados da aplicação das métricas de análise no segundo cenário, descrito em Kuszera et al. (2020), que inclui quatro coleções: *cliente*, *pedido*, *linha_pedido* e *produto*. A escolha desse estudo de caso se justifica pelo fato de que o autor também utiliza métricas para comparar diferentes esquemas de modelagem em

bancos de dados NoSQL, com foco específico em cenários de consultas. Isso torna o cenário especialmente apropriado para este trabalho, pois possibilita uma análise comparativa dos resultados obtidos a partir de um caso já discutido na literatura.

A Figura 5.6 mostra quatro esquemas gerados para a análise. Foram definidas sete consultas específicas para avaliação, conforme a Tabela 5.18. É importante destacar que os cenários de validação foram mantidos em sua forma original, conforme apresentados nos artigos fontes, garantindo a fidelidade aos contextos de estudo propostos pelos autores. Por esse motivo, os termos em inglês foram preservados nas figuras.

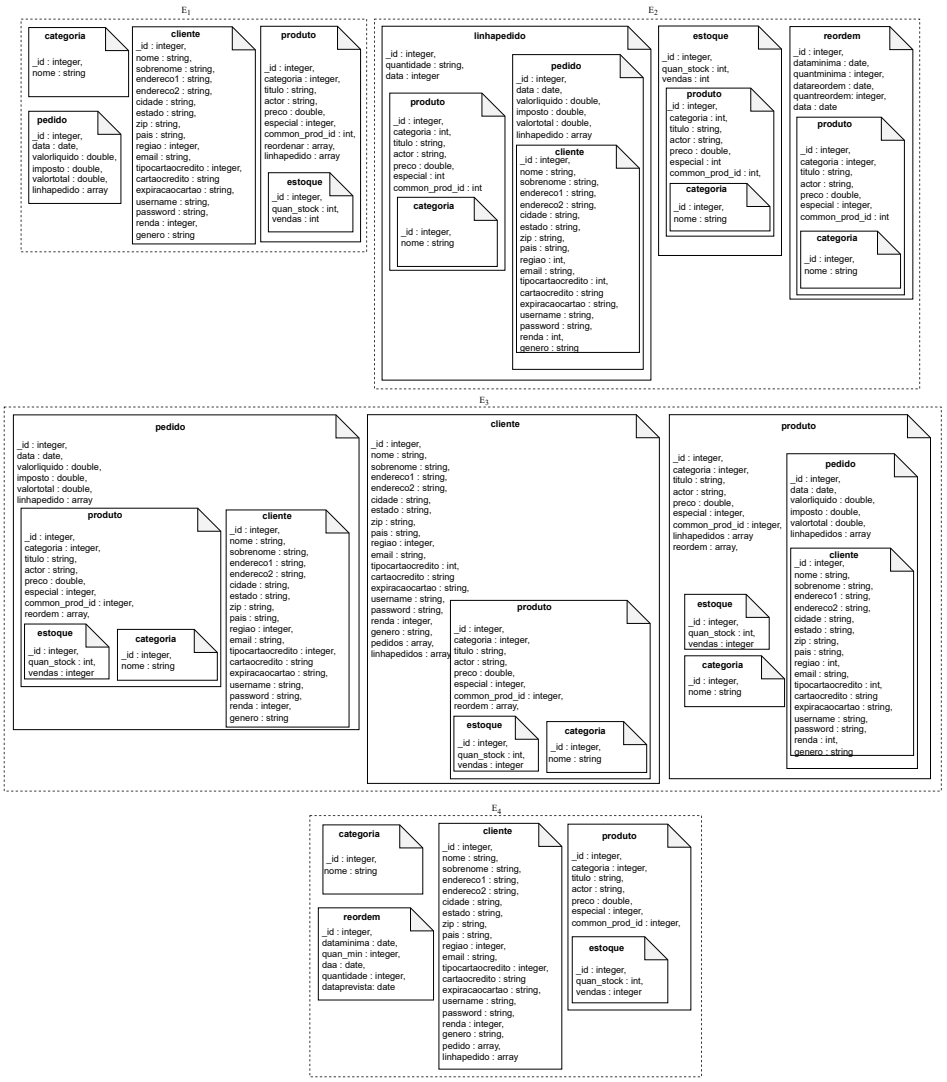


Figura 5.6: Esquemas para o segundo cenário (link para os esquemas).

Tabela 5.18: Consultas definidas para o segundo cenário.

No	**Consulta**	**Coleções**
q_1	*Selecionar todos os dados dos clientes onde o id do cliente é igual a 1.*	cliente
q_2	*Selecionar todos os dados dos produtos juntamente com o inventário, e onde o identificador do produto é igual a 1*	produto, estoque
q_3	*Selecionar todos os dados dos pedidos juntamente com as linhas de pedido, onde o id do pedido é igual a 1*	pedido, linha_pedido
q_4	*Selecionar todos os dados dos clientes juntamente com os pedidos, linhas de pedidos e produtos, e a data do pedido está entre '2009-01-01' e '2009-01-02'*	cliente, pedido, linha_pedido, produto
q_5	*Selecionar todos os dados dos produtos juntamente com as linhas de pedidos, os pedidos e os clientes, e onde o preço do produto está entre 29 e 30*	produto, linha_pedido, pedido, cliente
q_6	*Selecionar todos os dados dos pedidos juntamente com os clientes, e as linhas de pedidos, onde a data do pedido está entre '2009-01-01' e '2009-01-02'*	pedido, cliente, linha_pedido
q_7	*Selecionar todos os dados do inventário juntamente com as linhas de pedido, onde o identificador do pedido é igual a 1*	linha_pedido, estoque

Avaliação da métrica completude

A análise da métrica *completude* foi realizada para verificar o suporte das consultas por cada esquema. A Tabela 5.19 apresenta os resultados da análise de existência de coleções e relacionamentos.

Tabela 5.19: Análise de $existeColecao(\mathtt{E}, q_i)$ e $existeRelacionamento(\mathtt{E}, q_i)$ para cada consulta nos quatro esquemas.

	$\mathtt{E}_1$			$\mathtt{E}_2$			$\mathtt{E}_3$			$\mathtt{E}_4$		
	eC	eR	R	eC	eR	R	eC	eR	R	eC	eR	R
q_1	1	1	1	1	1	1	1	1	1	1	1	1
q_2	1	1	1	1	1	1	1	1	1	1	1	1
q_3	0	0	0	1	1	1	0	0	0	0	0	0
q_4	0	0	0	1	1	1	0	0	0	0	0	0
q_5	0	0	0	1	1	1	0	0	0	0	0	0
q_6	0	0	0	1	1	1	0	0	0	0	0	0
q_7	0	0	0	1	0	0	0	0	0	0	0	0

Tabela 5.20: Resultados da métrica completude para o segundo cenário.

$\mathtt{E}_1$	$\mathtt{E}_2$	$\mathtt{E}_3$	$\mathtt{E}_4$
0.3	0.9	0.3	0.3

Dos resultados observa-se, que $\mathtt{E}_2$ é o único esquema que atende a 90% das consultas, enquanto os demais esquemas oferecem suporte a apenas 30%. Essa limitação decorre da ausência de coleções ou relacionamentos necessários para responder às consultas mais complexas.

Avaliação da métrica padrão de acesso

De maneira análoga ao primeiro cenário, o cálculo da métrica *padrão de acesso* foi realizado com base na análise dos relacionamentos presentes nos esquemas. A Tabela 5.21 apresenta os resultados obtidos. Observa-se que a solução $\mathtt{E}_3$ possui o maior valor (4.4), refletindo a maior complexidade estrutural devido ao elevado número de relacionamentos aninhados. Em contrapartida, as soluções $\mathtt{E}_1$ e $\mathtt{E}_4$ exibem os menores valores (0.4), caracterizando esquemas com estruturas mais simples.

Tabela 5.21: Análise do *padrão de acesso* no segundo cenário.

Soluções	Esquemas	Relacionamentos referenciados	Relacionamentos aninhados	Cálculo com coeficientes	Total
$\mathtt{E}_1$	e_1	0	0	$0 \times 0.6 + 0 \times 0.4 = 0$	0.4
	e_2	0	0	$0 \times 0.6 + 0 \times 0.4 = 0$	
	e_3	0	1	$0 \times 0.6 + 1 \times 0.4 = 0.4$	
	e_4	0	0	$0 \times 0.6 + 0 \times 0.4 = 0$	
$\mathtt{E}_2$	e_1	0	4	$0 \times 0.6 + 4 \times 0.4 = 1.6$	3.2
	e_2	0	2	$0 \times 0.6 + 2 \times 0.4 = 0.8$	
	e_3	0	2	$0 \times 0.6 + 2 \times 0.4 = 0.8$	
$\mathtt{E}_3$	e_1	0	4	$0 \times 0.6 + 4 \times 0.4 = 1.6$	4.4
	e_2	0	3	$0 \times 0.6 + 3 \times 0.4 = 1.2$	
	e_3	0	4	$0 \times 0.6 + 4 \times 0.4 = 1.6$	
$\mathtt{E}_4$	e_1	0	0	$0 \times 0.6 + 0 \times 0.4 = 0$	0.4
	e_2	0	0	$0 \times 0.6 + 0 \times 0.4 = 0$	
	e_3	0	1	$0 \times 0.6 + 1 \times 0.4 = 0.4$	
	e_4	0	0	$0 \times 0.6 + 0 \times 0.4 = 0$	

Avaliação da métrica custo de recuperação

De forma análoga ao primeiro cenário, o cálculo da métrica *custo de recuperação* requer a determinação de $contarAtr(\mathtt{E})$ e $contarRel(\mathtt{E})$. No entanto, os valores de $contarRel(\mathtt{E})$ para todas as soluções já foram previamente calculados. Assim, procede-se apenas ao cálculo de $contarAtr(\mathtt{E})$.

A Tabela 5.22 apresenta os resultados obtidos. A solução $\mathtt{E}_3$ apresenta o maior custo total (64.09), devido à elevada quantidade de atributos simples e complexos em seus esquemas, além do alto valor de *contarRel*(E) (4.4). Por outro lado, $\mathtt{E}_4$ possui o menor custo total (19.74), destacando-se pela simplicidade de seus esquemas e pela baixa contribuição de *contarRel*(E) (0.4).

Quanto à análise dos atributos (*contarAtr*(E)) a solução $\mathtt{E}_3$ apresenta o maior valor (59.69), evidenciando o impacto significativo de seus atributos simples e complexos. Por outro lado, $\mathtt{E}_4$ possui o menor valor (19.34), devido ao menor número de atributos definidos em seus esquemas. Os valores consolidados de *contarAtr*(E) para cada solução mostra como a distribuição de atributos simples e complexos impacta o custo.

Tabela 5.22: Análise de *custo de recuperação* segundo cenário.

Soluções	Esquemas	Atributos simples	Atributos complexos	Cálculo com coeficientes	*contarAtr*(E)	*contarRel*(E)	Total
$\mathtt{E}_1$	e_1	6	1	$6 \times 0.51 + 1 \times 0.49 = 3.55$	20.32	0.4	20.72
	e_2	2	0	$2 \times 0.51 + 0 \times 0.49 = 1.02$			
	e_3	10	3	$10 \times 0.51 + 3 \times 0.49 = 6.57$			
	e_4	18	0	$18 \times 0.51 + 0 \times 0.49 = 9.18$			
$\mathtt{E}_2$	e_1	38	0	$38 \times 0.51 + 0 \times 0.49 = 19.38$	33.15	3.2	36.35
	e_2	12	0	$12 \times 0.51 + 0 \times 0.49 = 6.12$			
	e_3	15	0	$15 \times 0.51 + 0 \times 0.49 = 7.65$			
$\mathtt{E}_3$	e_1	36	5	$36 \times 0.51 + 5 \times 0.49 = 20.81$	59.69	4.4	64.09
	e_2	30	5	$30 \times 0.51 + 5 \times 0.49 = 17.75$			
	e_3	36	6	$36 \times 0.51 + 6 \times 0.49 = 21.13$			
$\mathtt{E}_4$	e_1	2	0	$2 \times 0.51 + 0 \times 0.49 = 1.02$	19.34	0.4	19.74
	e_2	18	2	$18 \times 0.51 + 2 \times 0.49 = 10.16$			
	e_3	10	0	$10 \times 0.51 + 0 \times 0.49 = 5.1$			
	e_4	6	0	$6 \times 0.51 + 0 \times 0.49 = 3.06$			

A análise evidencia que $\mathtt{E}_4$ é a solução mais eficiente em termos de custo de recuperação, enquanto $\mathtt{E}_3$ é a mais complexa, com custos elevados devido à alta presença de atributos e relacionamentos. $\mathtt{E}_1$ oferece um equilíbrio entre simplicidade e custo, e $\mathtt{E}_2$ pode ser considerado um intermediário, apresentando maior custo do que $\mathtt{E}_1$ e $\mathtt{E}_4$, mas ainda inferior ao de $\mathtt{E}_3$.

Avaliação da métrica redundância

Os resultados da análise da métrica *redundância* no segundo cenário estão apresentados na Tabela 5.23.

Tabela 5.23: Análise da *redundância*.

$\mathtt{E}_1$	$\mathtt{E}_2$	$\mathtt{E}_3$	$\mathtt{E}_4$
0	4	9	0

$\mathtt{E}_3$ apresentou o maior número de coleções repetidas (9), evidenciando um desenho que privilegia a duplicação de dados. Apesar de facilitar certas operações, essa abordagem

resulta em maior consumo de armazenamento e maior complexidade de manutenção. E_2 registrou 4 coleções repetidas, representando um nível intermediário de redundância. Embora menos eficiente que E_1 e E_4 ainda é menos impactante que E_3. E_1 e E_4 não possuem coleções repetidas (0), indicando desenhos mais otimizados e eficientes.

A redundância em E_3 reflete um desenho que pode ser adequado para cenários de consulta intensiva, mas apresenta desafios em termos de armazenamento e consistência de dados. Esse nível de duplicação requer maior atenção para evitar inconsistências em operações de atualização. E_2, apesar de apresentar redundância moderada, ainda pode ser otimizado para reduzir a duplicação de dados sem comprometer o desempenho nas consultas. E_1 e E_4 destacam-se como soluções mais equilibradas, com estruturas otimizadas que evitam duplicações.

Tabela 5.24: Resultados final das métricas no segundo cenário.

Soluções	Padrão acesso	Custo recuperação	Redundância	Total	Completude
E_1	0.4	20.72	0	21.12	0.3
E_2	3.2	36.35	4	43.55	0.9
E_3	4.4	64.09	9	77.49	0.3
E_4	0.4	19.74	0	20.14	0.3

A Tabela 5.24 apresenta os resultados finais. Apesar da simplicidade de E_1 e do seu baixo *custo de recuperação* (20.72), a sua baixa *completude* (0.3) limita seu uso em cenários onde é necessário suporte abrangente às consultas. E_2 Combina alta *completude* (0.9) com um *custo de recuperação* intermediário (36.35), sendo uma solução balanceada, embora apresente *redundância* moderada (4). E_3 é a solução mais complexa e redundante, com altos custos (64.09) e *completude* limitada (0.3). E_4 destaca-se como a solução mais eficiente em termos de custo (19.74) e ausência de *redundância* (0), mas sua baixa *completude* (0.3) restringe sua aplicabilidade.

5.7 Considerações Finais

Este capítulo descreve quatro métricas de avaliação: *completude*, *padrão de acesso*, *custo de recuperação* e *redundância*. Essas métricas foram selecionadas por capturarem aspectos complementares e essenciais da qualidade dos esquemas. A *completude* assegura que todas as informações necessárias estejam disponíveis nas consultas, o *padrão de acesso* reflete a complexidade de navegação entre coleções, o *custo de recuperação* estima o volume de dados transferidos durante uma consulta, e a *redundância* avalia a duplicação de dados no armazenamento. Juntas, essas métricas fornecem uma visão abrangente sobre a eficiência e a adequação dos esquemas modelados para diferentes cenários de uso.

As métricas propostas possibilitam uma avaliação comparativa e quantitativa, permitindo a distinção entre diferentes esquemas com base em sua complexidade estrutural. Conforme observado, menores valores das métricas estão associados a esquemas mais simples, enquanto valores mais elevados correspondem a esquemas de maior complexidade. Tais métricas são fundamentais no capítulo subsequente, onde serão incorporadas como parte da função objetivo de um algoritmo meta-heurístico, contribuindo para a otimização e seleção de esquemas mais adequados.

Capítulo 6

Otimização Heurística: Determinação da Melhor Solução Encontrada

Neste capítulo, é apresentada a forma pela qual o algoritmo meta heurístico *Variable Neighborhood Search* (`VNS`) é empregado para encontrar a solução mais eficiente, levando em consideração as consultas e coleções envolvidas em determinado modelo. Para tanto, o `VNS` utiliza as métricas e coeficientes de ponderação estabelecidos ao longo dos capítulos anteriores. Este capítulo está dividido em duas seções. Na Seção 6.1, é apresentado o pseudocódigo do algoritmo `VNS`, juntamente com as regras de validação, as métricas de avaliação integradas ao algoritmo e os operadores de perturbação utilizados. Na Seção 6.2, são descritos dois cenários de validação.

6.1 Algoritmo `VNS`

O algoritmo `VNS` é um algoritmo meta-heurístico utilizado para resolver problemas de otimização combinatória e contínua (Mladenović & Hansen, 1997). Ele opera explorando sistematicamente diferentes estruturas de vizinhança de uma solução inicial, alternando entre busca local e perturbações para escapar de ótimos locais e buscar soluções globais de melhor qualidade.

No tratamento de problemas de otimização, pode-se optar por meta-heurísticas baseadas em uma única solução ou em uma população de soluções, conforme Abualigah et al. (2021). A preferência por uma abordagem de solução única, como o `VNS`, simplifica o processo de análise ao focar na evolução de uma única configuração ao longo da execução, em vez de gerenciar uma população de soluções potenciais. Dentre as meta-heurísticas de solução única, como `Tabu Search`, `Simulated Annealing`, `Iterated Local Search`,

`Variable Space Search` e `Guided Local Search`, o `VNS` destaca-se por sua estrutura conceitualmente simples e pela menor demanda de ajuste de parâmetros (Hansen & Mladenović, 2001). Esses atributos facilitam sua aplicação em experimentos iniciais para identificar a melhor solução, tornando-o uma escolha prática e eficiente para o problema em questão.

A escolha do `VNS` como meta-heurística justifica-se por sua capacidade de explorar eficientemente espaços de busca combinatórios complexos, conforme demonstrado por Hansen et al. (2019). No contexto deste estudo, onde o espaço de soluções corresponde ao conjunto de todas as configurações possíveis de relacionamentos (referência, aninhamento ou ausência) entre coleções em bancos de dados NoSQL (formalizado na Equação 6.1), o `VNS` destaca-se por superar dois desafios críticos: a natureza exponencial do espaço de busca, cujo tamanho cresce com o número de coleções, inviabilizando métodos exaustivos; a necessidade de equilibrar adequadamente a diversificação, explorando novas regiões, com a intensificação por meio de refinamento local, estratégia intrínseca à alternância sistemática de vizinhanças que caracteriza o `VNS`. Essa dupla capacidade torna-o particularmente adequado para otimizar esquemas documentais, onde operações de `CRUD` convencionais mostram-se computacionalmente proibitivas frente à complexidade combinatória do problema.

$$|\mathcal{S}| = 3^{n(n-1)} \tag{6.1}$$

A Figura 6.1 descreve, em termos gerais, o funcionamento do algoritmo `VNS`, cujo objetivo é determinar a solução mais ótima a partir de qualquer solução inicial, ou seja, de qualquer conjunto de esquemas previamente definido. No diagrama de fluxo, destacam-se três módulos principais: *avaliar solução*, *perturbar solução* e *validar solução*.

A atividade *avaliar solução* tem como objetivo realizar uma avaliação quantitativa dos esquemas da solução, empregando as métricas *completude*, *padrão de acesso* e *redundância*, conforme definidas no Capítulo 5. A atividade *perturbar solução* é responsável por realizar modificações nos esquemas para explorar o espaço de busca com todas as possíveis soluções, sendo que a Tabela 6.1 apresenta detalhadamente as sete operações de perturbação estabelecidas para essa finalidade. Por fim, a atividade *validar solução* aplica doze regras de validação para assegurar a consistência e coerência dos esquemas.

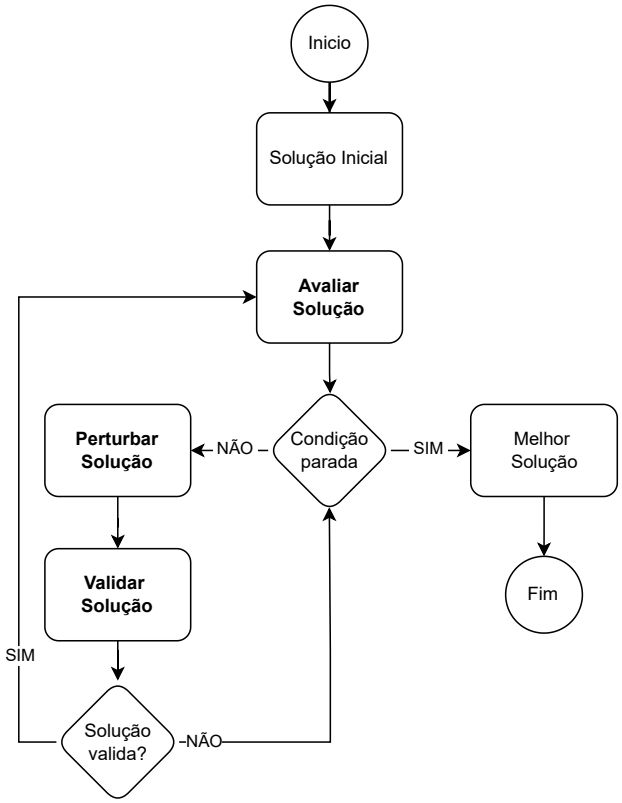


Figura 6.1: Diagrama de fluxo do algoritmo `VNS` para obtenção da solução mais ótima.

Tabela 6.1: Operações de perturbação usadas na atividade *perturbar solução*.

Operações Perturbação	Objetivo	Algoritmo
Modificar relacionamentos	Altera os relacionamentos em um esquema, alternando entre os tipos referenciado e aninhado.	3
Adicionar Relacionamentos	Insere um relacionamento (referenciado ou aninhado) em um esquema.	4
Adicionar coleções	Adiciona uma coleção ao esquema e estabelece uma relação aleatória com uma coleção existente.	5
Eliminar coleções	Remove uma coleção extrema do esquema e desfaz suas relações.	6
Gerar esquemas	Constrói um esquema válido com n coleções.	7
Eliminar esquemas	Remove aleatoriamente um esquema de um conjunto de esquemas.	8
Reinicializar esquemas	Cria um novo esquema do mesmo tamanho e substitui o anterior.	9

Adicionalmente, a Figura 6.2 apresenta a arquitetura funcional do algoritmo `VNS`, destacando suas principais entradas, os módulos centrais — avaliar solução, perturbar solução e validar solução — bem como a saída gerada pelo processo. A Figura 6.2 evidencia ainda a distribuição dos algoritmos entre os respectivos módulos, permitindo uma visualização clara da organização interna e do fluxo de execução do método proposto.

6.1.1 Pseudocódigo

O Algoritmo 1 descreve o procedimento para determinar a melhor solução possível para um determinado problema. Esse processo utiliza como entrada uma solução inicial (formada por um conjunto de esquemas), as consultas que devem ser atendidas e o número total

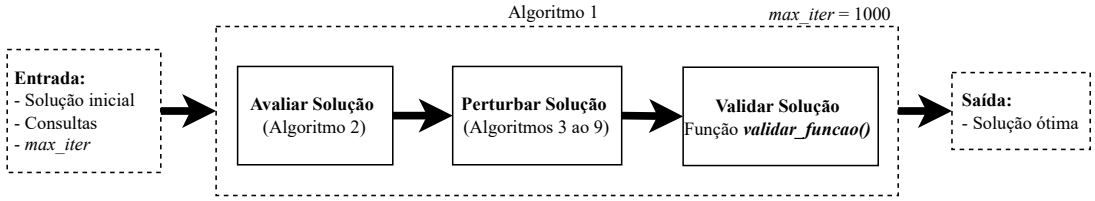


Figura 6.2: Arquitetura do algoritmos `VNS`.

de iterações a ser executado. Como saída, o algoritmo gera uma solução otimizada. Para gerar a solução final, o algoritmo fundamenta-se em três módulos principais: o módulo $funcao\_objetivo()$, responsável por avaliar o *fitness* (ou custo) de cada solução; o módulo $perturbar\_solucao()$, que introduz pequenas modificações à solução atual para explorar novas possibilidades; e o módulo $validar\_solucao()$, encarregado de verificar se a solução resultante é coerente e consistente. Na sequência, cada um desses elementos é explicado em detalhe.

Algoritmo 1: `VNS` para determinar o esquema mais ótimo

Entrada: Consultas $consultas$, Solução inicial $solucao\_inicial$ e número iterações max_iter=1 000
Saída: Solução mais otimizada
$melhor\_solucao \leftarrow solucao\_inicial$;
$melhor\_fitness \leftarrow$ funcao_objetivo($consultas, melhor\_solucao$);
$iteracoes \leftarrow 0$;
Enquanto $iteracoes < max\_iter$ **Fazer**
  $op \leftarrow$ operação aleatória de $\{0, 1, 2, 3, 4, 5, 6\}$;
  $nova\_solucao \leftarrow$ perturbar_solucao($melhor\_solucao, op$);
  Se $\neg validar\_solucao(nova\_solucao)$ **Então**
    $iteracoes \leftarrow iteracoes + 1$;
    Continuar
  $novo\_fitness \leftarrow$ funcao_objetivo($consultas, nova\_solucao$);
  Se $novo\_fitness < melhor\_fitness$ **Então**
    $melhor\_solucao \leftarrow nova\_solucao$;
    $melhor\_fitness \leftarrow novo\_fitness$;
  $iteracoes \leftarrow iteracoes + 1$;
Retornar *melhor_solucao;*

Entradas e saída do algoritmo

As entradas do algoritmo são compostas pelas consultas, pela solução inicial e pelo número de iterações. Em particular, tanto as consultas quanto a solução foram definidas

formalmente no Capítulo 5, sendo representadas pelos conjuntos `Q` e `E` respectivamente, juntamente com o conjunto de coleções `C`.

Cada consulta $q_i \in$ `Q` pode ser associada a um subconjunto de coleções do conjunto `C`. Em termos de implementação, uma consulta q_i pode ser representada por um par chave-valor em um dicionário, onde a chave corresponde à consulta e o valor é uma lista das coleções necessárias para responder a consulta. A Figura 6.3 ilustra um exemplo de representação das consultas que a solução deve atender. No exemplo, a variável `Q` contém três consultas, cada qual associada a um subconjunto do conjunto de coleções `C` = {1, 2, 3, 4}. Em particular, a chave q_1 e os valores associados [1, 2, 3] indica que a consulta q_1 requer as coleções 1, 2 e 3 para ser atendida; a consulta q_2 requer as coleções 1 e 3; e a consulta q_3 requer as coleções 1, 3 e 4.

```
Q = {
  'q1': [ 1, 2, 3 ],
  'q2': [ 1, 3 ],
  'q3': [ 1, 3, 4 ]
}
```

Figura 6.3: Representação das consultas.

Por outro lado, uma solução `E` é um conjunto de esquemas que devem atender as consultas definidas em `Q`. Em termos formais, se `E` é uma solução, então `E` $= \{e_1, e_2, ..., e_m\}$ com $m \geq 1$. Cada esquema e_j $(1 \geq j \geq m)$ satisfaz $e_j \subseteq$ `C` e $e_j \neq \emptyset$. Em termos de implementação, um esquema $e_j \in$ `E`, é definido mediante uma matriz $M^{(j)}$ de dimensão $k \times k$, onde k é o numero de coleções de `C`. Cada linha e cada coluna dessa matriz correspondem, respectivamente, a uma das coleções em `C`. Formalmente tem-se:

$$M^{(j)} = \begin{bmatrix} m_{1,1}^{(j)} & m_{1,2}^{(j)} & \cdots & m_{1,k}^{(j)} \\ m_{2,1}^{(j)} & m_{2,2}^{(j)} & \cdots & m_{2,k}^{(j)} \\ \vdots & \vdots & \ddots & \vdots \\ m_{k,1}^{(j)} & m_{k,2}^{(j)} & \cdots & m_{k,k}^{(j)} \end{bmatrix}, \quad \text{onde } 0 \leq l_1, l_2 < k.$$

Nessa matriz, cada elemento $\left(m_{l_1,l_2}^{(j)}\right)$ descreve o valor associado à coleção c_{l_1} (linha) e a coleção c_{l_2} (coluna). Os valores possíveis para cada posição na matriz são 0, 1, 2, 3 e 4. A seguir, apresenta-se a codificação de cada um desses valores.

A diagonal principal da matriz $M^{(j)}\left(m_{l,l}^{(j)}\right)$ indica se a coleção c_l participa ou não do esquema e_j. Então adota-se:

$$m_{l,l}^{(j)} = \begin{cases} 1, & \text{se a coleção } c_l \text{ participa de } e_j, \\ 2, & \text{se a coleção } c_l \text{ não participa de } e_j. \end{cases}$$

Os elementos fora da diagonal principal da matriz e_j podem assumir apenas os valores 0, 3 ou 4. Os valores 3 e 4 representam relacionamentos direcionados entre coleções que efetivamente participam de e_j. O valor 3 indica referência (relação referenciada) e o valor 4 indica aninhamento (relação aninhada). Esses relacionamentos são unidirecionais, de modo que $m_{l_1,l_2}^{(j)} \in \{3,4\}$ não implica necessariamente o mesmo valor em $m_{l_2,l_1}^{(j)}$. Caso contrário, se não há relacionamento ou se ao menos uma das coleções não participa do esquema, o valor atribuído é 0. Então adota-se:

$$m_{l_1,l_2}^{(j)} = \begin{cases} 3, & \text{se a coleção } c_{l_1} \text{ referencia } c_{l_2}, \\ 4, & \text{se a coleção } c_{l_1} \text{ aninha a } c_{l_2}, \\ 0, & \text{não existe relacionamento entre as coleções.} \end{cases}$$

Para ilustrar como a matriz representa diferentes cenários de relacionamento entre coleções, considere-se a solução `E`, formada pelos três esquemas e_1, e_2 e e_3, ou seja $\texttt{E} = \{e_1, e_2, e_3\}$. As Figuras 6.4, 6.5 e 6.6 mostram cada esquema individualmente. Na Figura 6.4, à esquerda, visualiza-se o esquema e_1 com três coleções (A, B e C), utilizando a notação gráfica descrita no Capítulo 3. Nesse diagrama, a coleção A referencia a coleção B, enquanto B aninha C. À direita, exibe-se a matriz correspondente a e_1. Observa-se que todas as posições na diagonal principal são iguais a 1, sinalizando a participação de A, B e C no esquema. A posição [0,1] tem o valor 3, pois A referencia B, e a posição [1,2] possui o valor 4, representando o aninhamento de C em B. As demais posições recebem o valor 0, indicando ausência de outros relacionamentos.

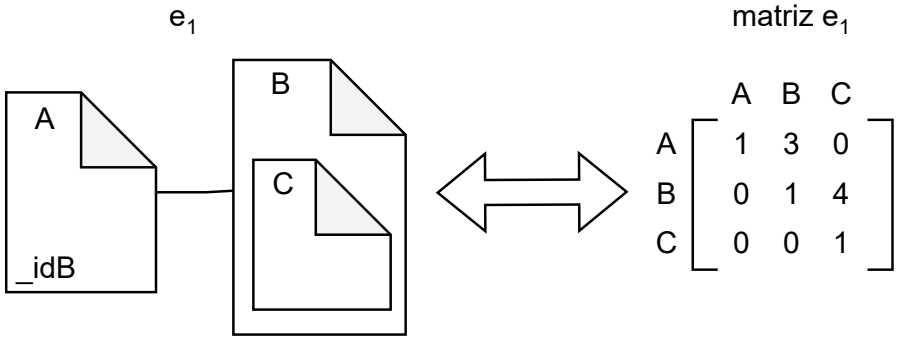


Figura 6.4: Representação gráfica e matricial do esquema e_1.

Na Figura 6.5, observa-se o esquema e_2, no qual apenas as coleções A e C participam efetivamente, enquanto B está presente no conjunto completo de coleções mas não integra

este esquema. Ainda assim, a matriz permanece com dimensão 3×3, mantendo a posição correspondente a B na diagonal com o valor 2, para indicar sua não participação. As coleções A e C aparecem com o valor 1 na diagonal principal, assinalando que participam de e_2. Fora da diagonal, a entrada [0,2] igual a 3 representa a referência de A para C, enquanto as demais posições são 0, indicando ausência de outros relacionamentos. Assim, a matriz codifica tanto a presença de A e C no esquema quanto a ausência de B e o relacionamento de referência unidirecional de A para C.

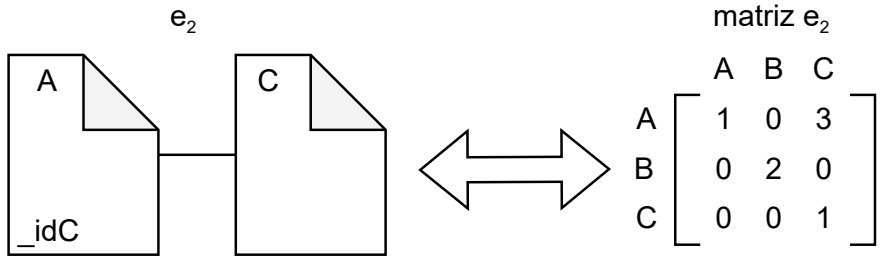


Figura 6.5: Representação gráfica e matricial do esquema e_2

Na Figura 6.6, observa-se o esquema e_3 envolvendo as coleções A, B e C. À esquerda, o diagrama mostra que A referencia C (indicando um atributo *_idC* em A) e que C referencia B (indicando um atributo *_idB* em C), enquanto B não referencia nenhuma das outras coleções. À direita, a matriz de e_3 confirma essa configuração: todas as coleções participam (valor 1 na diagonal principal) e as posições [0,2] e [2,1] são iguais a 3, sinalizando relações de referência unidirecionais. O restante das posições fora da diagonal são 0, indicando ausência de outras conexões. Desse modo, a matriz retrata de forma clara os relacionamentos definidos no esquema, assim como a participação de A, B e C.

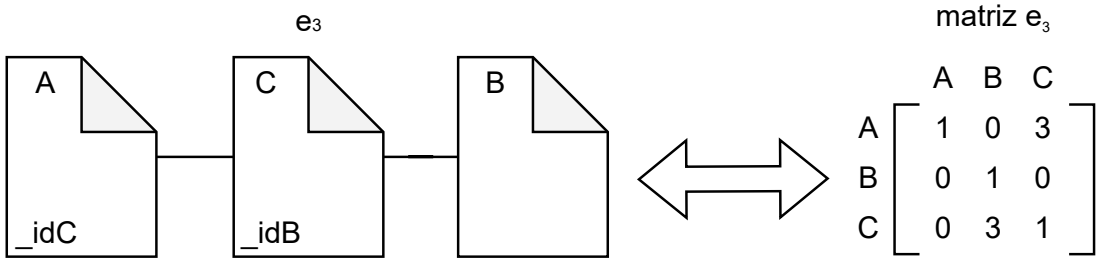


Figura 6.6: Representação gráfica e matricial do esquema e_3

A Figura 6.7 apresenta a variável *solucao_inicial*, que contém três esquemas: e_1, e_2 e e_3. Assim, tanto a Figura 6.7 quanto a Figura 6.3 ilustram as entradas formatadas em estrutura de dados para o Algoritmo 1.

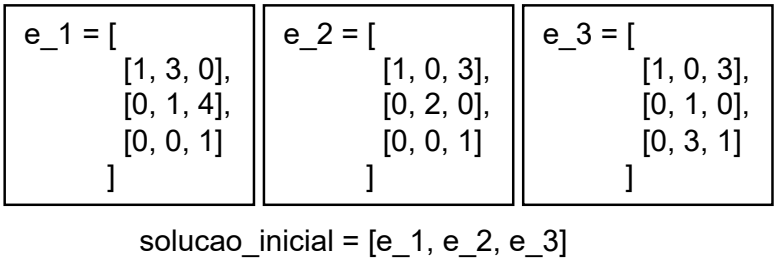


Figura 6.7: Representação da solução inicial.

6.1.2 Regras de Validação

A representação matricial de uma solução viabiliza uma manipulação computacional ágil e eficiente. No entanto, tais operações podem comprometer a consistência e a coerência lógica dos esquemas. Com o intuito de preservar a integridade estrutural da representação, foram definidas regras formais empregadas pela função *validar_esquema*()[1], no Algoritmo 1, para assegurar a validade dos esquemas gerados:

- Os únicos valores válidos que a matriz pode assumir são 0, 1, 2, 3 e 4. Os valores 1 e 2 são usados para representar a participação ou não das coleções no esquema. Os valores 3 e 4 são utilizados para representar relações referenciadas e aninhadas, respectivamente. Os demais elementos devem conter o valor 0;
- Uma coleção aninhada não pode ser referenciada por uma coleção externa, e vice-versa, conforme ilustrado na Figura 6.8, onde a coleção A tenta referenciar diretamente o documento C, o que não é possível porque C está aninhado dentro do documento B;
- Um esquema deve conter, no mínimo, uma coleção. Assim, é obrigatório que a diagonal da matriz possua pelo menos um elemento com valor igual a 1;
- Considera-se, no máximo, um aninhamento de 2 níveis em um esquema;
- Os relacionamentos, sejam referenciados ou aninhados, entre duas coleções são direcionais. Isso quer dizer que a relação tem um sentido: de uma coleção para outra, mas não necessariamente o contrário;
- Se duas coleções estão relacionadas por referência ou aninhamento, os valores correspondentes na diagonal dessas coleções devem ser iguais a 1;

[1]https://tinyurl.com/2k2b8bfh

- Não é permitido que uma coleção estabeleça uma autorreferência, ou seja, uma relação consigo mesma;
- Se uma coleção faz parte de um esquema, ela deve estar relacionada a pelo menos outra coleção, salvo no caso de ser a única coleção presente no esquema. Essa condição garante que o esquema represente um conjunto de coleções inter-relacionadas, alinhado ao objetivo de otimizar o agrupamento de dados com interações significativas. Caso uma coleção permaneça isolada, sem qualquer relação com as demais, ela é considerada, no contexto deste trabalho, como um esquema independente. Isso ocorre porque, do ponto de vista de desempenho e padrão de acesso, coleções isoladas não contribuem para os benefícios estruturais esperados dentro de um esquema compartilhado. Operacionalmente, essa regra é verificada na matriz de relacionamento, onde a linha ou a coluna correspondente à coleção deve conter ao menos um valor igual a 3 ou 4, indicando a existência de uma relação com outra coleção;
- Se uma coleção não participa do esquema (representada pelo valor 2 na diagonal), a linha e a coluna associadas a essa coleção devem conter exclusivamente valores iguais a 0. Isso indica que a coleção não faz parte do esquema e não possui relações com nenhuma outra coleção do mesmo;
- Cada coleção pode participar de um esquema apenas uma única vez;
- Não podem existir ciclos nos esquemas, ou seja, a estrutura resultante deve ser acíclica;
- Todas as coleções que participam de um esquema (diagonal = 1) devem estar diretamente conectadas por meio de relações, garantindo a coesão da estrutura. Isso implica que não pode haver subconjuntos isolados de coleções participantes

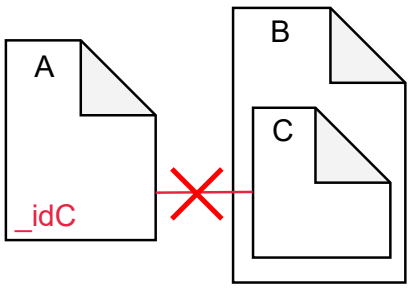


Figura 6.8: Error referenciando uma coleção aninhada.

Todas as regras de validação descritas foram implementadas na função *validar_esquema*(). Além disso, essa função é empregada na geração de *validar_solucao*(), a qual é responsável por validar um conjunto de esquemas utilizado no Algoritmo 1. O objetivo dessa

validação é assegurar que a estrutura matricial do esquema esteja em conformidade com os critérios de consistência e coerência previamente estabelecidos.

6.1.3 Métricas de Avaliação

O cálculo da função $fitness$, representada no Algoritmo 1 como $funcao\_objetivo()$, utiliza três métricas fundamentais para avaliar soluções: *completitude*, *padrão de acesso* e *redundância*. Essas métricas, definidas no Capítulo 5, são combinadas de forma estruturada no Algoritmo 2 para quantificar a qualidade da solução proposta. O algoritmo incorpora uma penalização proporcional à incompletude. A seguir, detalham-se as métricas utilizadas:

- Pontuação completitude (p_c): calcula a proporção de consultas atendidas pela solução, por meio da função completitude();
- Pontuação de padrão de Acesso (p_p_a): quantifica as relações referenciadas e aninhadas nos esquemas de uma solução, utilizando a função $padrao\_acesso()$;
- Pontuação de redundância (p_r): quantifica a repetição de coleções, através da função $redundancia()$;
- Penalização de completitude ($pen\_completitude$): calcula uma penalização proporcional à incompletude da solução, ou seja, à proporção de consultas não atendidas. Essa penalização é obtida multiplicando o valor da incompletude (1 - p_c) por um fator de escala (1000), garantindo que soluções com baixa cobertura de consultas sejam fortemente penalizadas

Algoritmo 2: Avaliar Solução ($função\_objetivo()$)

Entrada: Conjunto de consultas *consultas*, Solução *solucao*
Saída: Pontuação total da solução
$p\_c \leftarrow$ completitude($consultas, solucao$);
$p\_p\_a \leftarrow$ padrao_acesso($solucao$);
$p\_r \leftarrow$ redundancia($solucao$);
$pen\_completitude \leftarrow (1 - p\_c) \times 1000$;
$p\_total \leftarrow p\_c + p\_p\_a + p\_r + pen\_completitude$;
Retornar *p_total;*

O resultado quantitativo da $função\_objetivo()$ (p_total) é definida como a soma dos componentes p_c, p_p_a e p_r, acrescida de um termo de penalização por incompletude, $pen\_completitude$. Dessa forma, o algoritmo `VNS` direciona sua busca minimizando a

função de $fitness$, utilizando estratégias de intensificação e diversificação para explorar o espaço de soluções e aprimorar iterativamente a solução candidata.

A métrica de *custo de recuperação* não foi considerada nesta etapa, pois seu cálculo exige um processamento computacional elevado, já que seria necessário adicionar os atributos de cada coleção para realizar cálculos detalhados sobre os tempos de recuperação. Além disso, é necessário contar com coeficientes de recuperação ajustados para cada tipo de dado simples (como inteiros, floats, strings, datas, etc.) e para diferentes tamanhos e estruturas de dados complexos, o que demandaria uma análise ainda mais granular e específica que está fora do escopo desta fase. Por essas razões, optou-se por focar nas métricas que garantem uma avaliação da qualidade estrutural das soluções, mantendo a viabilidade computacional do processo de otimização.

6.1.4 Estrategias de Perturbação

A perturbação desempenha um papel crucial no algoritmo `VNS`, atuando como o mecanismo que possibilita escapar de mínimos locais e explorar novas regiões do espaço de soluções. Neste contexto, foram definidas sete estratégias de perturbação, a saber: modificar relacionamentos, adicionar relacionamentos, adicionar coleção, eliminar coleção, gerar esquema, eliminar esquema e reiniciar esquema. Essas estratégias foram implementadas na função *perturbar_solucao*(), utilizada no Algoritmo 1 para aplicar a perturbação sobre as soluções.

Modificar relacionamentos

O Algoritmo 3 tem como objetivo alterar os relacionamentos presentes em um esquema, com a finalidade de explorar novas configurações que possam melhorar a qualidade da solução. Para tanto, a entrada do algoritmo consiste em uma matriz denominada *esquema*, na qual cada elemento representa um relacionamento entre coleções, com especial atenção aos valores 3 e 4, que indicam os tipos de relacionamento suscetíveis à modificação. Para isso, o algoritmo identifica, de forma sistemática, os pares de índices (i, j) nos quais o valor presente corresponde a 3 ou 4. Em seguida, seleciona aleatoriamente um desses pares e procede com a troca do valor alternando entre 3 e 4 de modo a modificar a relação existente. Após a modificação, o algoritmo valida o esquema; se o novo esquema for considerado válido, a alteração é mantida e o esquema modificado é retornado. Caso contrário, o algoritmo reverte a modificação, restabelecendo o valor original, e retorna o esquema inalterado. Dessa forma, a saída do algoritmo é o esquema resultante do processo de modificação e validação, contribuindo para a diversificação da busca e evitando a convergência prematura para mínimos locais.

Algoritmo 3: Modificar relacionamentos

Entrada: Matriz do esquema *esquema*

Saída: *esquema* modificado se válido; caso contrário o esquema original

relacionamentos $\leftarrow \{(i, j) \mid i \neq j \text{ e } esquema[i][j] \in \{3, 4\}\}$;

Se *relacionamentos* $\neq \emptyset$ **Então**

- $(i, j) \leftarrow$ escolha aleatória de *relacionamentos*;
- *anterior* $\leftarrow$ *schema*$[i][j]$;
- *esquema*$[i][j] \leftarrow$ **se** *esquema*$[i][j] = 4$ **então** 3 **senão** 4;
- **Se** *validar_esquema*(*esquema*) **Então**
  - **Retornar** *esquema*;
- **Senão**
  - *esquema*$[i][j] \leftarrow$ *anterior*;

Retornar *esquema*;

Adicionar relacionamentos

O Algoritmo 4 tem como objetivo inserir um novo relacionamento em um esquema, de modo a explorar configurações alternativas que possam aprimorar a solução. Para tanto, o algoritmo inicia identificando todas as posições candidatas, isto é, todos os pares de índices (i, j) tais que $i \neq j$ e o valor presente em *esquema*$[i][j]$ seja 0, indicando a ausência de um relacionamento.

Em seguida, a partir do conjunto de candidatos, é realizada uma seleção aleatória de um par (i, j). Após a seleção, o algoritmo determina, também de forma aleatória, um novo valor a ser atribuído à posição escolhida, optando entre os valores 3 e 4, que representam os tipos de relacionamento admissíveis. Essa alteração é aplicada ao esquema, e o novo estado é submetido a uma validação por meio da função *validar_esquema*(), que verifica se a modificação resulta em uma configuração viável do esquema.

Caso o esquema modificado seja validado com sucesso, o algoritmo retorna o esquema atualizado, consolidando a inserção do relacionamento. Por outro lado, se a validação não for satisfatória, o algoritmo reverte a modificação, restaurando o valor 0 na posição afetada, e retorna o esquema original. Esse procedimento assegura que apenas as alterações que conduzam a soluções válidas sejam mantidas, contribuindo para a robustez e eficácia do processo de busca no espaço de soluções.

Algoritmo 4: Adicionar Relacionamento

Entrada: Matriz do esquema *esquema*

Saída: *esquema* modificado se válido; caso contrário, o original

$candidatos \leftarrow \{(i, j) \mid i \neq j \text{ e } esquema[i][j] = 0\}$;

Se $candidatos \neq \emptyset$ **Então**

- $(i, j) \leftarrow$ escolha aleatória de *candidatos*;
- $novo\_valor \leftarrow$ escolha aleatória entre $\{3, 4\}$;
- $esquema[i][j] \leftarrow novo\_valor$;
- **Se** $validar\_esquema(esquema)$ **Então**
  - **Retornar** *esquema*;
- **Senão**
  - $esquema[i][j] \leftarrow 0$;

Retornar *esquema*;

Adicionar coleção

O Algoritmo 5 tem como objetivo incorporar uma nova coleção ao esquema, de modo a ampliar as possibilidades de relações entre os elementos. Inicialmente, o algoritmo identifica quais coleções já estão presentes—determinadas pelo valor 1 na diagonal da matriz— e quais estão faltando, indicadas pelos valores 2. Caso existam coleções faltantes, procede-se à seleção aleatória de uma coleção faltante e de uma coleção já existente. Em seguida, escolhe-se, de forma aleatória, um novo tipo de relação, representado pelos valores 3 ou 4, que será estabelecido entre a nova coleção e a coleção existente. Dependendo de outra escolha aleatória, a relação é criada tanto da nova coleção para a existente quanto na direção inversa. Após a inserção do relacionamento, o esquema modificado é validado pela função *validar_esquema*(). Se a validação for satisfatória, a alteração é mantida e o esquema atualizado é retornado; caso contrário, a modificação é revertida, restaurando os valores originais, e o esquema inalterado é devolvido.

Algoritmo 5: Adicionar Coleção

Entrada: Matriz do esquema *esquema*

Saída: *esquema* modificado se válido; caso contrário, o original

```
presentes ← { i | esquema[i][i] = 1 };
faltantes ← { i | esquema[i][i] = 2 };
Se faltantes ≠ ∅ Então
    nova_colecao ← escolha aleatória entre faltantes;
    colecao_existente ← escolha aleatória entre presentes;
    nova_relacao ← escolha aleatória entre {3, 4};
    Se escolha aleatória entre {True, False} Então
        esquema[nova_colecao][nova_colecao] ← 1;
        esquema[nova_colecao][colecao_existente] ← nova_relacao;
    Senão
        esquema[nova_colecao][nova_colecao] ← 1;
        esquema[colecao_existente][nova_colecao] ← nova_relacao;
    Se validar_esquema(esquema) Então
        Retornar esquema;
    esquema[nova_colecao][nova_colecao] ← 2;
    esquema[nova_colecao][colecao_existente] ← 0;
    esquema[colecao_existente][nova_colecao] ← 0;
Retornar esquema;
```

Eliminar coleção

O Algoritmo 6 tem como objetivo remover uma coleção extrema — definida como uma coleção ativa que não possui relacionamentos de saída, mas recebe relacionamentos de entrada no esquema representado por uma matriz — com o intuito de otimizar a estrutura relacional. Para isso, o algoritmo percorre todas as coleções (representadas pelos índices da matriz) aplicando os critérios de seleção predefinidos para identificar as coleções classificadas como "extremos". Essas coleções que satisfazem os critérios estabelecidos são então agrupadas no conjunto denominado "extremos".

Caso o conjunto de extremos não esteja vazio, o algoritmo seleciona aleatoriamente uma coleção a ser eliminada. Para essa coleção selecionada, todas as relações horizontais e verticais (ou seja, os elementos da linha e da coluna correspondentes) que possuem os valores 3 ou 4 são removidas, sendo substituídos por 0, o que indica a ausência de relacionamento. Em seguida, a diagonal da coleção eliminada é alterada para o valor 2, sinalizando sua exclusão do conjunto de coleções ativas.

Por fim, o esquema modificado é validado por meio da função *validar_esquema()*. Se o esquema atender aos critérios estabelecidos pela validação, a modificação é mantida e o esquema atualizado é retornado; caso contrário, o algoritmo retorna o esquema original. Esse procedimento assegura que apenas alterações que resultem em uma configuração válida do esquema sejam efetivadas.

Algoritmo 6: Eliminar coleção

Entrada: Matriz do esquema *esquema*

Saída: *esquema* modificado se válido; caso contrário, o original

```
extremos ← {};
para i ← 0 até tamanho(esquema) − 1 faça
    referencia_a_outras ← existe algum j, com j ≠ i, tal que
     esquema[i][j] ∈ {3, 4};
    referenciado_por_outras ← existe algum j, com j ≠ i, tal que
     esquema[j][i] ∈ {3, 4};
    Se (não referencia_a_outras) e (referenciado_por_outras) e
     (esquema[i][i] = 1) Então
        Adicione i a extremos;
Se extremos ≠ ∅ Então
    colecao_a_eliminar ← escolha aleatória entre extremos;
    para j ← 0 até tamanho(esquema) − 1 faça
        Se esquema[colecao_a_eliminar][j] ∈ {3, 4} Então
            esquema[colecao_a_eliminar][j] ← 0;
        Se esquema[j][colecao_a_eliminar] ∈ {3, 4} Então
            esquema[j][colecao_a_eliminar] ← 0;
    esquema[colecao_a_eliminar][colecao_a_eliminar] ← 2;
    Se validar_esquema(esquema) Então
        Retornar esquema;
Retornar esquema
```

Gerar esquema

O Algoritmo 7 tem como objetivo construir um esquema válido a partir de um número n de coleções, em que todas as coleções participam ativamente. Inicialmente, o algoritmo gera uma matriz $n \times n$ preenchida com zeros e, em seguida, atribui o valor 1 à diagonal da matriz, indicando a presença de cada coleção.

Posteriormente, são definidas as relações habilitadas entre as coleções, considerando todos os pares ordenados (i, j) com $i \neq j$. A partir desse conjunto, o algoritmo seleciona

de forma aleatória relações unidirecionais. Para cada par selecionado, é verificado se não há relações já existentes nos sentidos opostos, isto é, se as posições $esquema[i][j]$ e $esquema[j][i]$ estão ambas com valor zero. Quando essa condição é satisfeita, um tipo de relação (valor 3 ou 4) é escolhido aleatoriamente e atribuído à posição (i, j) da matriz, registrando-se a relação na lista de relações selecionadas.

Além disso, se o tipo de relação escolhido for 4, o algoritmo impõe uma restrição adicional: garante-se que a coleção destino, representada pelo índice j, não possua outras relações de saída. Para isso, todas as demais posições da linha j, exceto a coluna j (diagonal), são zeradas.

O processo de seleção continua até que o número de relacionamentos definidos seja igual a $n - 1$. Ao final, o esquema gerado é submetido a uma validação, realizada pela função $validar\_esquema()$. Se o esquema atender aos critérios de validade, ele é retornado; caso contrário, o algoritmo retorna a mensagem "Esquema inválido".

Algoritmo 7: Gerar Esquema Válido

Entrada: Inteiro n representando o número de coleções
Saída: Matriz *esquema* válido

```
esquema ← matriz n × n preenchida com 0;
para i ← 0 é n − 1 faça
    esquema[i][i] ← 1;
relacoes_habilitadas ← {(i, j) | i, j ∈ {0, ..., n − 1} e i ≠ j};
relacoes_selecionadas ← [];
Enquanto |relacoes_selecionadas| < n − 1 Fazer
    (i, j) ← escolha aleatória de relacoes_habilitadas;
    Se esquema[i][j] = 0 e esquema[j][i] = 0 Então
        tipo_relacao ← escolha aleatória entre {3, 4};
        esquema[i][j] ← tipo_relacao;
        Adicione (i, j) a relacoes_selecionadas;
        Se tipo_relacao = 4 Então
            para k ← 0 é n − 1 faça
                Se k ≠ j Então
                    esquema[j][k] ← 0;
valido ← validar_esquema(esquema);
Se valido Então
    Retornar esquema;
```

Eliminar esquema

O Algoritmo 8 tem como finalidade remover um dos esquemas presentes em uma solução composta por múltiplos esquemas, contribuindo para a diversificação ou o refinamento da solução. A entrada do algoritmo é uma lista denominada *solucao*, que contém um ou mais esquemas, e a saída é a própria lista modificada após a remoção de um dos esquemas.

Inicialmente, o algoritmo verifica se a lista *solucao* possui mais de um esquema. Caso a lista contenha apenas um esquema, o algoritmo retorna imediatamente a lista sem realizar qualquer modificação, assegurando a integridade da solução. Se houver mais de um esquema, o algoritmo seleciona aleatoriamente um índice, correspondendo ao esquema a ser eliminado. Em seguida, esse esquema é removido da lista *solucao* e a lista atualizada é retornada.

Algoritmo 8: Eliminar esquema

Entrada: Solução *solucao*

Saída: *solucao*

Se $tamanho(solucao) \geq 1$ **Então**

    $eliminar\_esquema \leftarrow$ escolha aleatória entre 0 e tamanho($solucao$) $- 1$;

    Remova o esquema na posição *eliminar_esquema* de *solucao*;

Retornar *solucao*;

Reinicializar esquema

A reinicialização de esquemas tem como objetivo evitar que o Algoritmo 1 fique preso em mínimos locais, um fenômeno comum em problemas de otimização (Hansen et al., 2019). Isso ocorre porque o espaço de busca pode conter várias soluções locais ótimas, onde perturbações não geram melhorias significativas. Quando o algoritmo depende exclusivamente de modificações incrementais, ele pode ficar restrito a uma região subótima sem explorar outras possibilidades com potencial de melhor desempenho. Para superar essa limitação, a reinicialização busca gerar um novo esquema válido, permitindo escapar de mínimos locais e ampliando a exploração do espaço de busca.

Para isso, o Algoritmo 9 recebe um esquema pertencente a uma solução e gera um novo esquema de mesmo tamanho. Em seguida, verifica se o novo esquema é válido mediante a função *validar_esquema()*. Caso seja válido, ele é retornado como resultado; caso contrário, o algoritmo mantém e retorna o esquema original de entrada.

Algoritmo 9: Reinicializar esquema

Entrada: *esquema*

Saída: *esquema*

$n \leftarrow$ tamanho(*esquema*);

novo_esquema $\leftarrow$ gerar_esquema_valido(n);

valido $\leftarrow$ validar_esquema(*novo_esquema*);

Se *valido* **Então**

  Retornar *novo_esquema*;

Senão

  Retornar *esquema*;

6.2 Cenários de Validação

Nesta seção, são apresentados dois cenários de validação, ambos correspondentes aos descritos no Capítulo 5. O processo de validação consiste em definir as consultas que deverão ser suportadas e preparar uma solução inicial, a qual servirá como entrada para o algoritmo. Como resultado, o algoritmo deverá retornar uma solução otimizada para atender às consultas definidas.

6.2.1 Primeiro cenário

No primeiro cenário, o conjunto de coleções $\mathtt{C}$ é composto pelas coleções *funcionario*, *departamento* e *empresa*. Para facilitar o processamento, as coleções são codificadas da seguinte forma: *funcionario*=1, *departamento*=2 e *empresa*=3. Dessa forma, o conjunto $\mathtt{C}$ é definido como:

$$\mathtt{C} = \{1, 2, 3\}$$

As consultas que devem ser atendidas estão especificadas na Tabela 6.2. Observa-se que as consultas 1, 2, 3 e 6 utilizam apenas a coleção *funcionario*. Já as consultas 4 e 5 requerem as coleções *funcionario* e *empresa*. Por fim, a consulta 7 depende das coleções *empresa* e *departamento*. Assim, definimos o conjunto Q, segundo as dependências das consultas, como:

$$\mathtt{Q} = \{q_1, q_2, q_3\}$$

Onde, q_1 corresponde às consultas que dependem exclusivamente da coleção *funcionario* para serem atendidas (consultas 1, 2, 3 e 6); q_2 engloba as consultas que exigem as coleções

funcionario e *empresa* (consultas 4 e 5); e q_3 representa as consultas que necessitam das coleções *empresa* e *departamento* (consulta 7).

Tabela 6.2: Consultas definidas para o primeiro cenário.

No	Consulta	Coleções
1	*Funcionários com um salário igual a $1000*	funcionário
2	*Funcionários com um salário superior a $1000*	funcionário
3	Funcionários com o maior salário	funcionário
4	*Funcionários com o maior salário por empresa e o ID da empresa*	funcionário, empresa
5	*Funcionários com o maior salário por empresa e o nome da empresa*	funcionário, empresa
6	*O salário mais alto*	funcionário
7	*Informações das empresas, incluindo o nome de seus departamentos*	empresa, departamento

Por outro lado, as soluções que deveriam fornecer suporte às consultas estão detalhadas na Figura 5.5. Das nove propostas, apenas oito serão consideradas, uma vez que a solução $\mathtt{E}_4$ introduz a coleção (CDE) que não corresponde ao conjunto das coleções `C` analisado. A seguir, é detalhada a solução $\mathtt{E}_6$ como exemplo ilustrativo.

A Figura 6.9 apresenta a solução $\mathtt{E}_6$ composta por três esquemas. Assim, definimos o conjunto $\mathtt{E}_6$ como $\mathtt{E}_6 = \{e_1, e_2, e_3\}$.

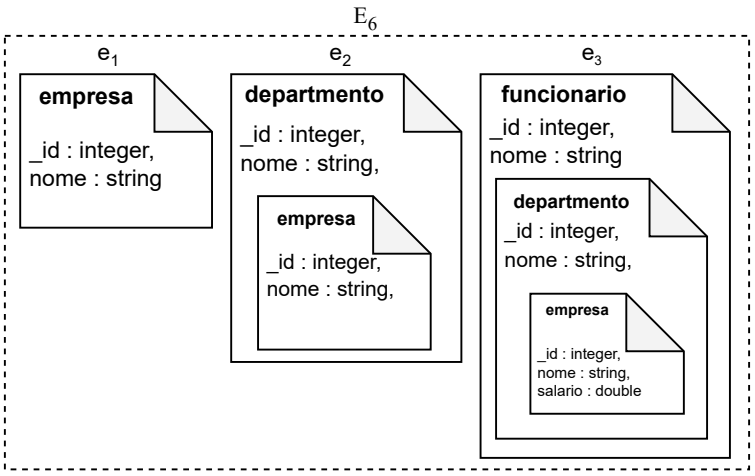


Figura 6.9: Solução $\mathtt{E}_6$ composta de três esquemas

A Figura 6.10 apresenta a representação matricial dos esquemas e_1, e_2 e e_3. A matriz do esquema e_1 contém apenas uma coleção, representada pelo número 1 no terceiro valor da diagonal principal, correspondente à coleção `empresa`. As coleções `funcionario` e `departamento` não fazem parte do esquema, razão pela qual o primeiro e segundo valor na diagonal principal foram atribuídos como 2. Por ser `empresa` a única coleção participante, não há relacionamentos no esquema, resultando no preenchimento do restante da matriz com zeros.

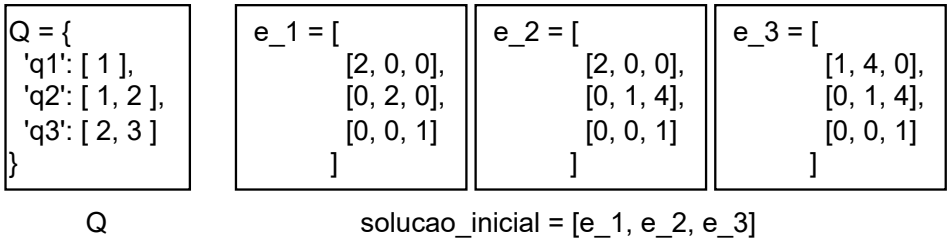


Figura 6.10: Consultas e esquemas referentes à solução inicial do cenário 1, apresentados em formato codificado.

Na matriz do esquema e_2, participam duas coleções: `departamento` e `empresa`. Por essa razão, os valores na segunda e terceira diagonais são 1, enquanto o valor 2 na primeira diagonal indica que a coleção `funcionario` não faz parte do esquema. Em relação aos relacionamentos, o esquema apresenta um aninhamento entre as coleções `departamento` e `empresa`, representado pelo valor 4 na posição [2,3], indicando que `empresa` está aninhado dentro de `departamento`. O restante da matriz é preenchido com zeros, evidenciando a ausência de outras relações diretas entre as coleções.

De forma similar, na matriz do esquema e_3, participam três coleções: `funcionario`, `departamento` e `empresa`, cada uma representada pelo valor 1 em sua respectiva posição na diagonal principal, indicando sua presença no esquema. Além disso, a matriz apresenta relações por aninhamento entre as coleções `departamento` e `empresa`, bem como entre `funcionario` e `departamento`, ambas indicadas pelo valor 4 nas posições correspondentes. Isso significa que os documentos da coleção `departamento` estão incorporados dentro da coleção `funcionario`, enquanto os documentos de `empresa` estão aninhados dentro de `departamento`. O restante da matriz é preenchido com zeros, evidenciando a ausência de outras relações diretas entre as coleções.

Finalmente, a entrada para o algoritmo `VNS` é composta pelas consultas codificadas (`Q`), pela solução inicial codificada (`solucao_inicial`) e pelo número máximo de iterações (`max_iter` = 1000), conforme apresentado na Figura 6.10. A variável `Q` representa o conjunto de consultas utilizadas, sendo que q_1 depende da coleção 1 (`funcionario`); q_2 depende das coleções 1 e 2 (`funcionario` e `departamento`); e q_3 depende das coleções 2 e 3 (`departamento` e `empresa`). A variável `solucao_inicial` corresponde à solução de partida para o processo de busca, enquanto `max_iter` determina o limite superior de iterações permitidas para a execução do algoritmo.

A Figura 6.11 apresenta a solução inicial antes e após a otimização pelo algoritmo `VNS`. A solução otimizada contém apenas dois esquemas, resultantes da eliminação de um dos esquemas presentes na estrutura original. Além disso, os dois esquemas remanescentes apresentam uma estrutura distinta em relação à original.

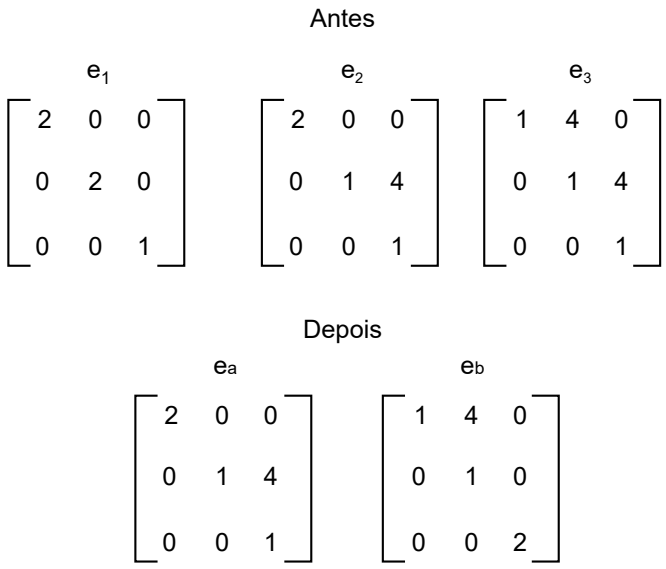


Figura 6.11: Antes e depois da solução otimizada.

A Tabela 6.3 apresenta a comparação das métricas entre a solução original ($\mathtt{E}_6$) e a solução otimizada (`E novo`). Os resultados evidenciam que a otimização impactou principalmente as métricas de *padrão de acesso* e *redundância*, reduzindo seus valores na solução final. Essa redução indica uma melhoria na estrutura dos esquemas, tornando a solução mais eficiente em termos de acessibilidade e minimização de redundâncias, sem comprometer a completude das consultas. Adicionalmente, a Figura 6.12 ilustra a diferença entre as soluções $\mathtt{E}_6$ e `E novo`.

Tabela 6.3: Comparação das métricas entre a solução original e a otimizada.

Solução	Completude	Padrão de acesso	Redundância	Total
E6	1	1.2	3	5.2
E novo	1	**0.8**	**1**	**2.8**

Apesar das modificações, a nova solução atende a todas as consultas definidas no conjunto `Q` e apresenta as seguintes pontuações nas métricas avaliadas: *completude* = 1, *padrão de acesso* = 0.8, *redundância* = 1 e *penalização* = 0. A métrica *completude* possui valor 1 porque todas as consultas são satisfeitas. O *padrão de acesso* recebe a pontuação 0.8, calculada com base na soma dos relacionamentos referenciados e aninhados, ponderados por seus respectivos coeficientes. Por fim, a métrica *redundância* atinge o valor 1, pois a coleção *empresa* se repete uma vez na nova solução. O valor total da métrica da nova solução é 2.8 que é o valor obtido da soma da *completude*, *padrão de acesso*, *redundância* e penalização. A análise para o restante das soluções do cenário 1 é apresentado na Tabela 6.4.

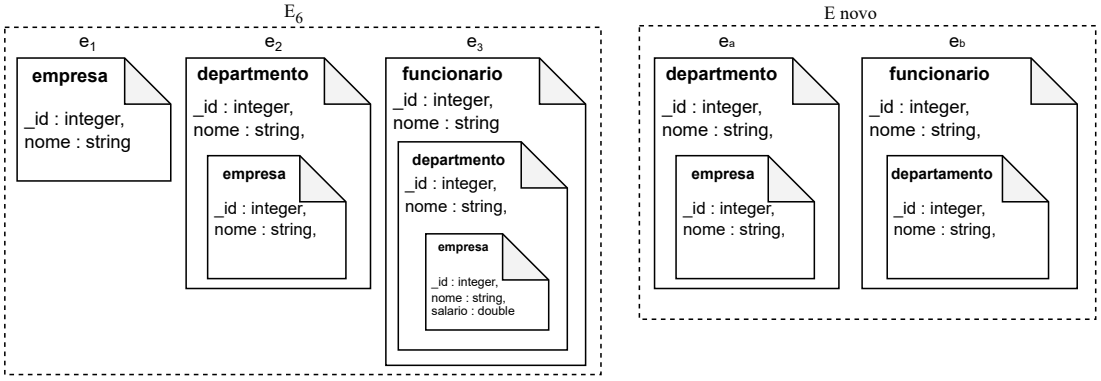


Figura 6.12: Comparação da solução antes e depois da otimização.

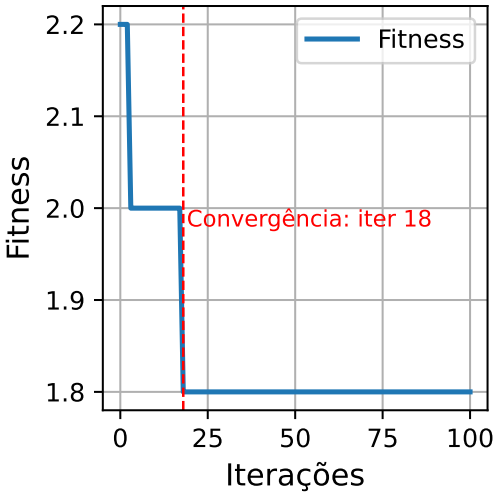
De forma análoga, analisaram-se os demais esquemas das soluções E1, E_2, E_3, E_5, E_7, E_8 e E_9. A Tabela 6.4 apresenta as métricas antes e depois da aplicação do algoritmo VNS para cada solução. O algoritmo conseguiu otimizar em 0,2 unidades na métrica total das soluções E_1, E_2 e E_9, com redução observada apenas na métrica *padrão de acesso*. Nas soluções E_3, E_5, E_7 e E_8, o algoritmo não obteve melhorias adicionais, possivelmente devido à simplicidade dessas soluções, compostas por um único esquema, tornando a redução das métricas uma tarefa mais complexa.

Tabela 6.4: Comparação das métricas entre as soluções avaliadas.

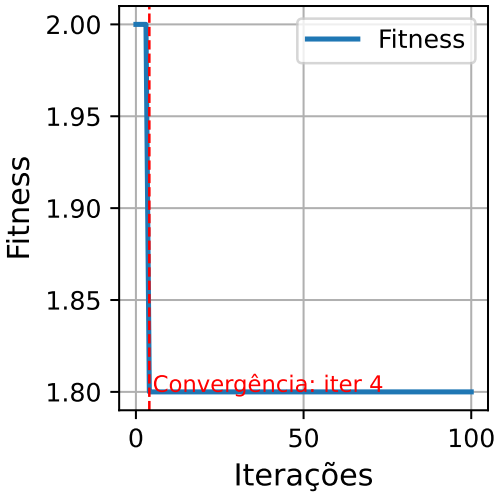
Solução original					**Solução otimizada**			
Solução	Compl.	Padrão	Redun.	Total	Compl.	Padrão	Redun.	Total
E_1	1	1.2	0	**2.2**	1	1	0	**2**
E_2	1	1	0	**2**	1	0.8	0	**1.8**
E_3	1	0.8	0	1.8	1	0.8	0	1.8
E_5	1	0.8	0	1.8	1	0.8	0	1.8
E_7	1	0.8	0	1.8	1	0.8	0	1.8
E_8	1	1	0	2	1	1	0	2
E_9	1	1	1	**3**	1	0.8	1	**2.8**

A Figura **??** apresenta a convergência do algoritmo VNS para oito soluções distintas, numeradas de E_1 a E_9, com exceção da solução E_4, que não foi considerada. Em geral, observa-se que o algoritmo demonstrou rápida convergência para a maioria das soluções, com variações discretas nos valores finais de `fitness`. A solução E_1, por exemplo, atingiu a convergência na 24ª iteração. A solução E_2 apresentou uma convergência ainda mais precoce, ocorrendo na 8ª iteração, com resultado semelhante. As soluções E_3, E_5 e E_7 exibem um comportamento distinto, caracterizado pela ausência de variação no valor de `fitness` ao longo das iterações, o que pode indicar uma convergência imediata ou uma estagnação em um ótimo local, sugerindo que o algoritmo não obteve melhoria ao longo do processo de busca.

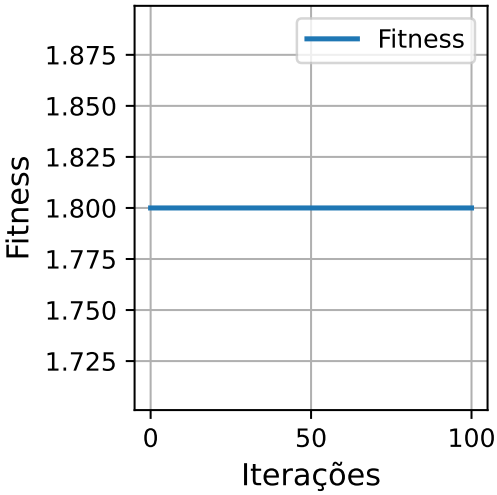
A solução E_6 se destaca por apresentar o maior valor inicial de `fitness` (acima de 5.0), demonstrando um processo de otimização mais expressivo. Esta solução convergiu na 23ª iteração. A solução E_8 apresentou a convergência mais rápida entre todas as analisadas, ocorrendo já na 3ª iteração, o que demonstra alta eficiência, embora com resultado final próximo ao das demais soluções. Por fim, a solução E_9 atingiu a convergência na 10ª iteração, alcançando um valor de `fitness` de aproximadamente 2.8, sendo uma das melhores soluções em termos de qualidade final da função objetivo.



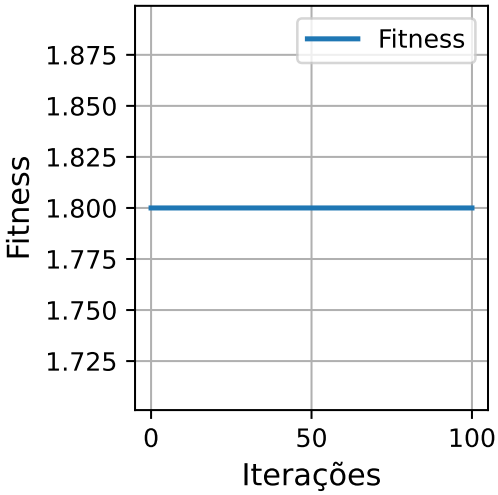
a) Solução E_1



b) Solução E_2

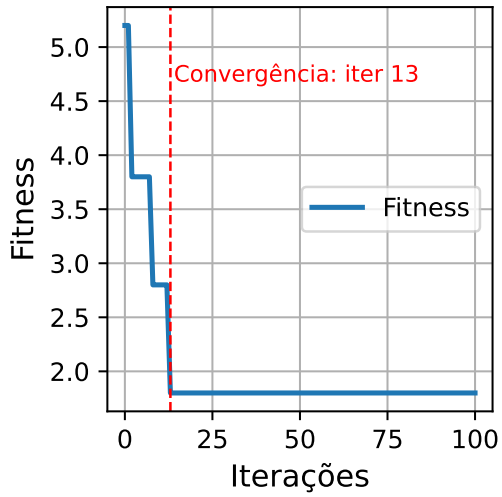


c) Solução E_3

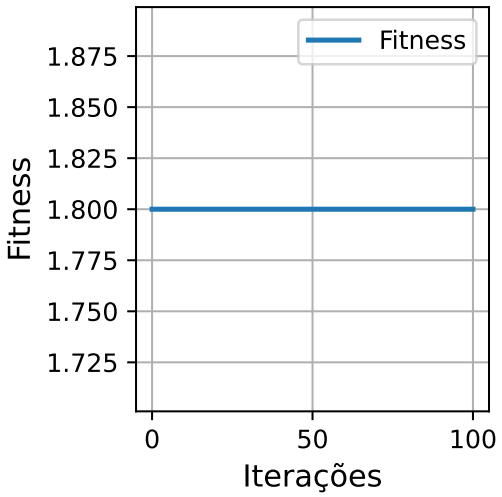


d) Solução E_5

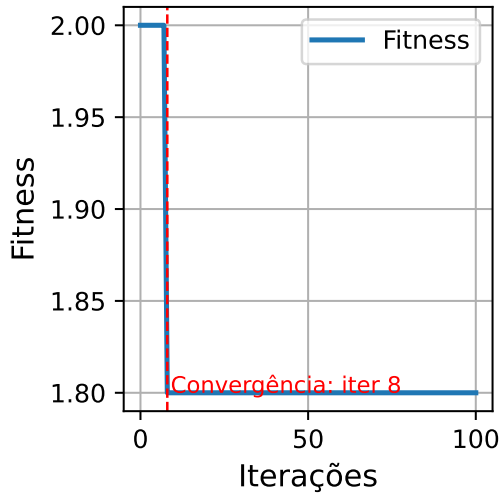
Figura 6.13: Diagramas de convergência das soluções do cenário 1.



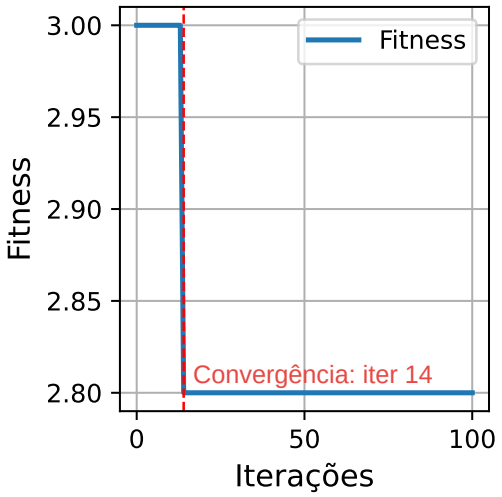
a) Solução E_6



b) Solução E_7



c) Solução E_8



d) Solução E_9

Figura 6.14: Diagramas de convergência das soluções do cenário 1.

6.2.2 Segundo Cenário

No segundo cenário, analisa-se um sistema de gestão de vendas. Para isso, são apresentadas quatro soluções, conforme ilustrado na Figura 5.6. As coleções envolvidas neste cenário são: *cliente*, *produto*, *pedido*, *estoque* e *categoria*. De modo análogo ao cenário 1, as coleções foram codificadas da seguinte forma: *cliente* = 1, *produto* = 2, *pedido* = 3,

estoque = 4 e *categoria* = 5. Assim, o conjunto $\mathtt{C}$ das coleções é definido como:

$$\mathtt{C} = \{1, 2, 3, 4, 5\}$$

As consultas que devem ser atendidas estão detalhadas na Tabela 6.5. A partir da análise da tabela, observa-se que as consultas 4 e 5 requerem as mesmas coleções. Dessa forma, é possível codificar essas duas consultas em uma única, simplificando a implementação. Assim, o conjunto $\mathtt{Q}$ é definido com base nas dependências entre as consultas e as coleções, conforme a seguinte representação:

$$\mathtt{Q} = \{q_1, q_2, q_3, q_4, q_5, q_6\}$$

Onde, q_1 corresponde à consulta que depende exclusivamente da coleção *cliente* (consulta 1); q_2 refere-se à consulta que depende das coleções *produto* e *estoque* (consulta 2); q_3 está associada à coleção *pedido* (consulta 3); q_4 engloba as coleções *cliente*, *produto* e *pedido* (consultas 4 e 5); q_5 abrange as coleções *cliente* e *pedido* (consulta 6); e q_6 corresponde à coleção *estoque* (consulta 7).

Tabela 6.5: Consultas definidas para o segundo cenário.

No	Consulta	Coleções
1	*Selecionar todos os dados dos clientes onde o id_cliente é igual a 1.*	cliente
2	*Selecionar todos os dados dos produtos juntamente com o inventário, e onde o identificador do produto é igual a 1*	produto, estoque
3	*Selecionar todos os dados dos pedidos juntamente com as linhas de pedido, onde o id do pedido é igual a 1*	pedido
4	*Selecionar todos os dados dos clientes juntamente com os pedidos, linhas de pedidos e produtos, e a data do pedido está entre '2009-01-01' e '2009-01-02'*	cliente, pedido, produto
5	*Selecionar todos os dados dos produtos juntamente com as linhas de pedidos, os pedidos e os clientes, e onde o preço do produto está entre R$29 e R$30*	cliente, pedido, produto
6	*Selecionar todos os dados dos pedidos juntamente com os clientes , e as linhas de pedidos, onde a data do pedido está entre '2009-01-01' e '2009-01-02'*	pedido, cliente
7	*Selecionar todos os dados do inventário juntamente com as linhas de pedido, onde o identificador do pedido é igual a 1*	estoque

As soluções que atendem às consultas estão detalhadas na Figura 5.6. Dentre essas, analisaremos a solução $\mathtt{E}_1$ como exemplo ilustrativo, de forma análoga ao que foi realizado no Cenário 1.

A Figura 6.15 apresenta a solução $\mathtt{E}_1$ de forma simplificada, com a omissão dos atributos para facilitar a visualização e destacar apenas a estrutura das coleções. Essa simplificação visa proporcionar uma melhor compreensão da organização dos esquemas, sem detalhar os atributos específicos. A solução é composta por quatro esquemas, e, com base nisso, definimos o conjunto $\mathtt{E}_1$ da solução como $\mathtt{E}_1 = \{e_1, e_2, e_3, e_4\}$.

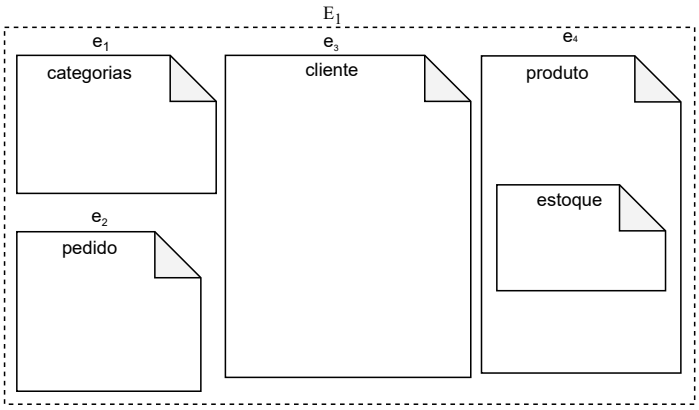


Figura 6.15: Solução $\mathtt{E}_1$ do cenário de validação 2.

A representação matricial dos esquemas e_1, e_2, e_3, e_4 é mostrada na Figura 6.16. O esquema e_1 contempla exclusivamente a coleção *categoria* como componente do esquema, cuja participação é indicada pelo valor 1 no quinto elemento da diagonal principal da matriz.

De forma análoga, os esquemas e_2 e e_3 são constituídos exclusivamente por uma única coleção: *pedido* e *cliente*, respectivamente. A participação dessas coleções é representada pelos valores 1 no terceiro e no primeiro elemento da diagonal principal, respectivamente. O esquema e_4 é composto por duas coleções: *produto* e *estoque*, cuja participação é indicada pelo valor 1 no segundo e no quarto elemento da diagonal principal, respectivamente. Além disso, esse esquema apresenta um relacionamento de aninhamento, na qual a coleção *produto* aninha a coleção *estoque*. Esse relacionamento é representada na matriz pelo valor 4 na posição [1,3]. Em todos os casos, o restante da matriz é preenchido com zeros, evidenciando a ausência de outras relações entre as coleções.

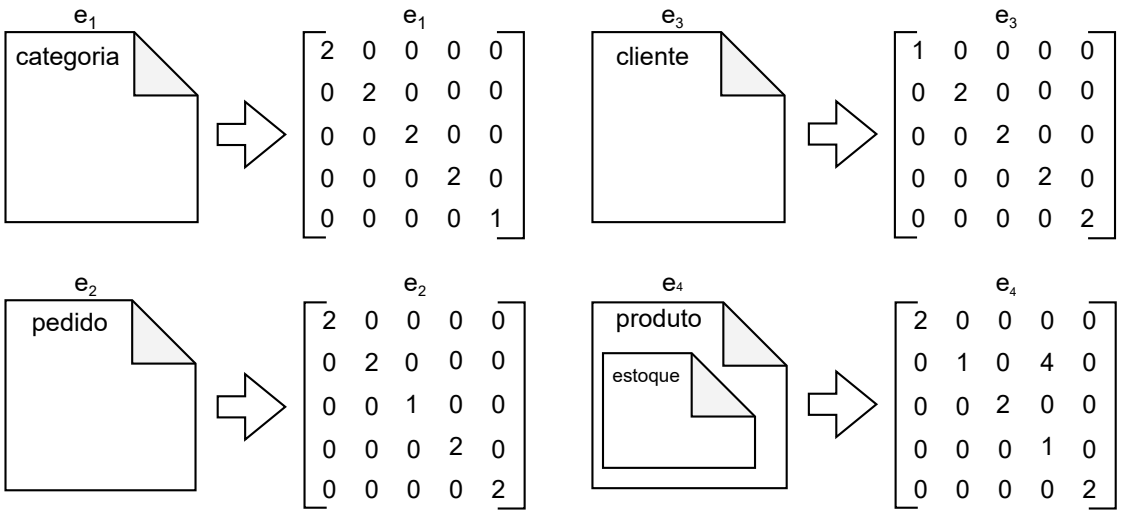


Figura 6.16: Representação matricial dos quatro esquemas da solução $\mathtt{E}_1$.

A entrada do algoritmo `VNS` para o Cenário 2 é composta pelo conjunto de consultas codificadas (`Q`), pela solução inicial codificada (`solucao_inicial`) e pelo número máximo de iterações definido como `max_iter` = 1000, conforme ilustrado na Figura 6.17. As consultas apresentam diferentes dependências em relação às coleções: q_1 está vinculada à coleção 1; q_2, às coleções 2 e 4 (`produto` e `estoque`); q_3, à coleção 3 (`pedido`); q_4, às coleções 1, 2 e 3 (`cliente`, `produto` e `pedido`); q_5, às coleções 1 e 3 (`cliente` e `pedido`); e q_6, à coleção 5 (`categoria`).

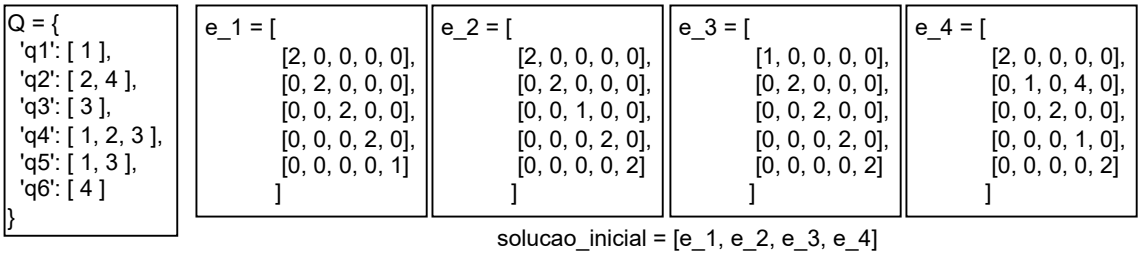


Figura 6.17: Consultas e esquemas referentes à solução inicial do cenário 2, apresentados em formato codificado.

Após a execução do algoritmo, a solução inicial, composta por quatro esquemas, foi reduzida a dois esquemas. A Figura 6.18 ilustra a diferença entre a configuração antes e depois da execução do algoritmo na representação matricial. Na solução resultante, apenas o esquema e_4 foi mantido da versão anterior, sendo renomeado como e_y no novo arranjo. Por outro lado, os esquemas e_1, e_2 e e_3 foram eliminados, e um novo esquema, denominado e_x, foi gerado. Na configuração final, as consultas q_1, q_3, q_4 e q_5 são atendidas pelo esquema e_x, enquanto as consultas q_2 e q_6 são atendidas pelo esquema e_y. A Figura 6.19 apresentá a diferença visual da solução antes e depois da execução do algoritmo.

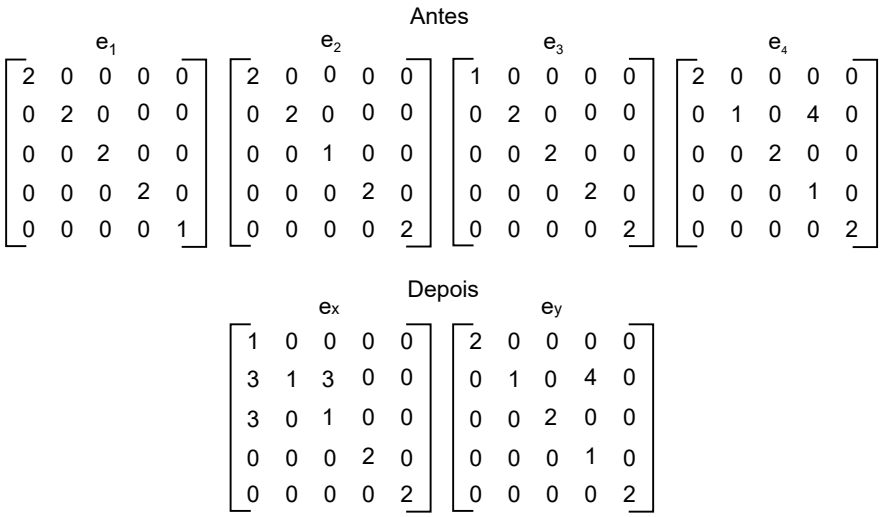


Figura 6.18: Antes e depois da solução otimizada cenário 2.

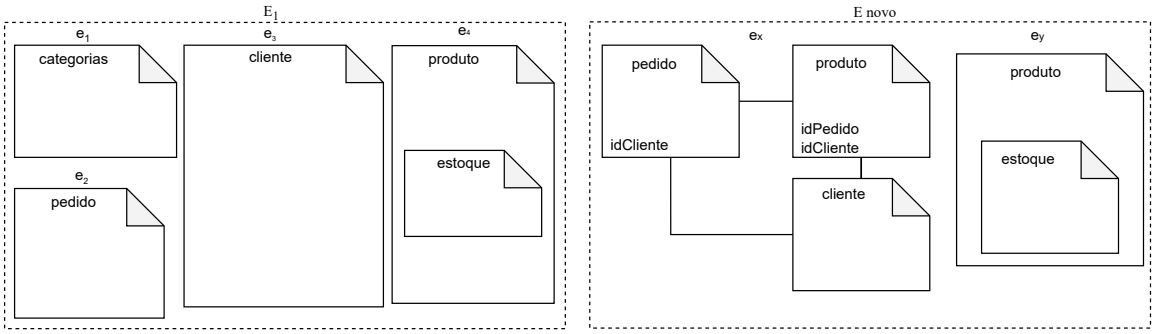


Figura 6.19: Comparação visual da solução antes e depois da otimização cenário 2.

A Tabela 6.6 apresenta a análise das métricas da solução antes e depois da execução do algoritmo. Inicialmente, a solução $\mathtt{E}_1$ é composta por quatro esquemas, sendo que três deles contêm apenas uma única coleção. Esses três esquemas atendem exclusivamente às consultas cuja dependência está limitada a uma única coleção (q_1, q_3), o que impede o atendimento de consultas que requerem acesso a múltiplas coleções. Por outro lado, o esquema e_4, formado pelas coleções *produto* e *estoque*, atende às consultas q_2 e q_6. Como consequência, a métrica de *completude* da solução inicial assume um valor de 0.6, indicando que nem todas as consultas são atendidas. Além disso, a métrica de *padrão de acesso* é de 0.4, pois em toda a solução apenas o esquema e_4 utiliza relacionamento por aninhamento, sem apresentar redundância de coleções. A métrica total para essa solução é de 1 unidade, evidenciando que, embora eficiente em termos de estrutura, a solução não é capaz de atender a todas as consultas. Após a execução do algoritmo, uma nova solução é gerada, composta por dois esquemas: e_x e e_y. O esquema e_y permanece idêntico ao esquema e_4 da solução original, enquanto os esquemas e_1, e_2 e e_3 são eliminados, dando

origem ao novo esquema e_x. A nova configuração permite o atendimento de todas as consultas, resultando em um valor de *completude* igual a um. No entanto, o *padrão de acesso* e a *redundância* aumentam, elevando a métrica total para 4.8.

Tabela 6.6: Comparação das métricas entre a solução original e a otimizada no cenário 2.

Solução	Completude	Padrão de Acesso	Redundância	Total
$\texttt{E}_1$	0.6	0.4	0	1
`E novo`	1	2.8	**1**	4.8

De maneira análoga, foram analisados os demais esquemas das soluções $\texttt{E}_2$, $\texttt{E}_3$ e $\texttt{E}_4$. A Tabela 6.7 apresenta um resumo das métricas antes e depois da execução do algoritmo para cada uma dessas soluções. Dessa forma, as métricas de *completude*, *padrão de acesso* e *redundância* são calculadas tanto na solução inicial quanto após a aplicação do algoritmo.

No caso da solução $\texttt{E}_2$, tanto a métrica de *completude* quanto a de *redundância* foram otimizadas. A *completude* foi aprimorada ao garantir o atendimento de todas as consultas, enquanto a *redundância* foi reduzida em relação à solução inicial. Embora a métrica de *padrão de acesso* tenha aumentado ligeiramente, esse acréscimo é justificado pelo ganho significativo em termos de cobertura das consultas. Para a solução $\texttt{E}_3$, a métrica total foi reduzida de 14.4 para 7.8, representando uma melhoria substancial nas métricas de *padrão de acesso*, e *redundância*, ao mesmo tempo em que a *completude* foi mantida. Por fim, na solução $\texttt{E}_4$, a métrica total aumentou de 0.9 para 2.2. No entanto, a *completude* foi aprimorada, garantindo o atendimento de 100% das consultas, o que justifica a alteração nos demais parâmetros.

Tabela 6.7: Comparação das métricas entre as soluções avaliadas no cenário 2.

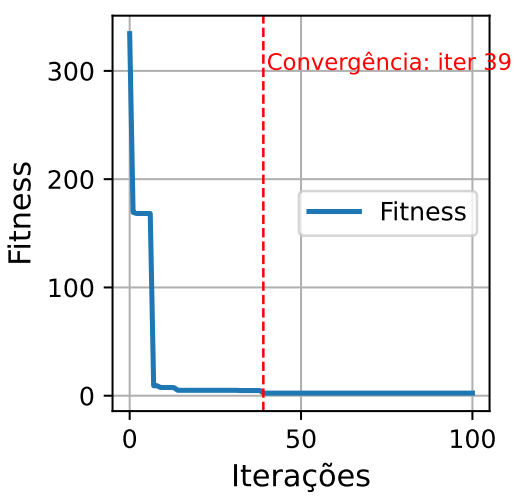
Solução Inicial					**Solução Otimizada**			
Solução	Compl.	Padrão de Acesso	Redun.	Total	Compl.	Padrão de Acesso	Redun.	Total
$\texttt{E}_2$	**0.83**	1.2	1	3.03	**1**	1.4	0	2.4
$\texttt{E}_3$	1	4.4	9	14.4	1	2.8	4	7.8
$\texttt{E}_4$	**0.5**	0.4	0	0.9	**1**	1.2	0	2.2

A Figura 6.20 apresenta os diagramas de convergência das soluções $\texttt{E}_1$, $\texttt{E}_2$, $\texttt{E}_3$ e $\texttt{E}_4$ obtidas na execução do algoritmo `VNS` aplicadas ao cenário 2. Cada subgráfico ilustra a evolução do valor de `fitness` ao longo de 1000 iterações, permitindo analisar o comportamento e a eficiência do algoritmo em cada instância do problema.

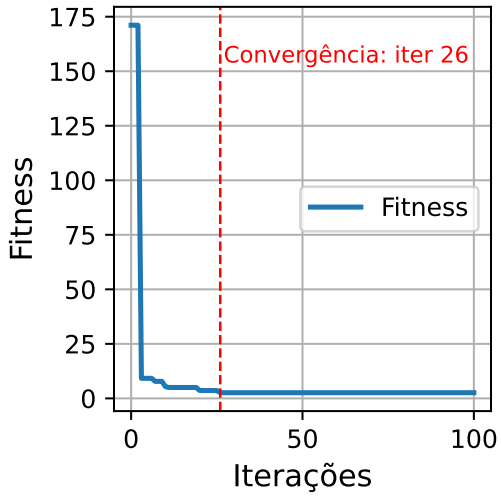
Observa-se que todas as soluções convergem rapidamente nas primeiras iterações, com a estabilização dos valores de `fitness` ocorrendo antes da centésima iteração. Especificamente, as soluções $\texttt{E}_1$, $\texttt{E}_2$, $\texttt{E}_3$ e $\texttt{E}_4$ convergiram nas iterações 46, 35, 34 e 51, respectivamente, conforme indicado pelas linhas tracejadas em azul. Este comportamento evidencia

a capacidade do algoritmo `VNS` de encontrar soluções ótimas ou próximas do ótimo em um número reduzido de iterações, o que demonstra sua eficiência computacional.

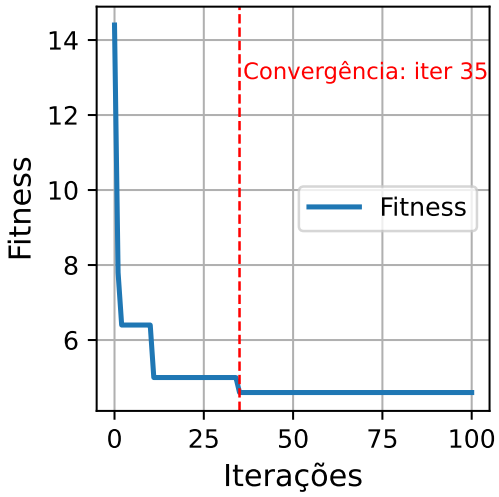
Entretanto, nota-se uma variação considerável nos valores iniciais de `fitness` entre as soluções. Por exemplo, a solução $\mathtt{E}_4$ apresenta um valor inicial significativamente mais alto (acima de 400), enquanto a solução $\mathtt{E}_3$ inicia com um valor inferior a 15. Essa diferença sugere que a complexidade ou a natureza das instâncias pode influenciar o desempenho inicial do algoritmo, embora todas alcancem uma estabilização próxima ao zero, indicando convergência para soluções de boa qualidade.



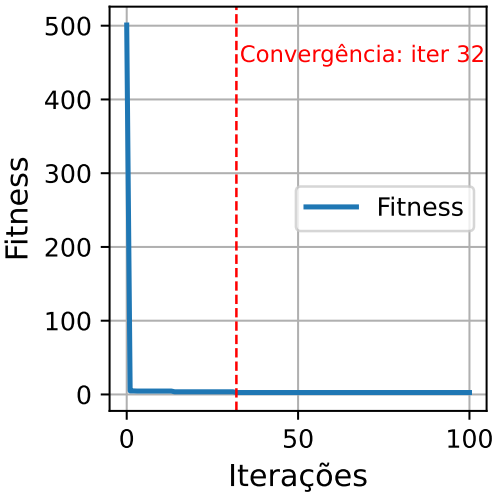
a) Solução $\mathtt{E}_1$



b) Solução $\mathtt{E}_2$



c) Solução $\mathtt{E}_3$



d) Solução $\mathtt{E}_4$

Figura 6.20: Diagramas de convergência das soluções do cenário 2.

6.3 Limitações

Apesar dos avanços alcançados, este estudo apresenta limitações técnicas que devem ser consideradas para futuras investigações no domínio de bancos de dados `NoSQL` orientados a documentos. Essas restrições, descritas a seguir, refletem escolhas metodológicas e delimitações de escopo, mas não comprometem a validade dos resultados obtidos, indicando direções para refinamentos e extensões:

- Os coeficientes de ponderação foram calibrados exclusivamente com base em operações de leitura, conforme Vera-Olivera et al. (2023). Consequentemente, as métricas de avaliação (completude, padrão de acesso, custo de acesso e redundância) e sua integração na função objetivo do algoritmo `VNS` também se restringem a esse tipo de operação. Essa limitação exclui a consideração de operações de escrita, atualização e exclusão, potencialmente subestimando o impacto de cargas de trabalho heterogêneas no desempenho dos esquemas otimizados;

- A análise de coleções aninhadas foi limitada a uma profundidade máxima de dois níveis de aninhamento. Essa restrição simplifica a modelagem e a manipulação algorítmica, mas pode não refletir cenários reais em bancos de dados `NoSQL`. Essa delimitação pode reduzir a aplicabilidade dos esquemas otimizados em casos que demandem maior flexibilidade estrutural;

- O operador de perturbação "Eliminar Coleção" foi projetado para remover apenas coleções "extremas" (i.e., aquelas nas extremidades de um relacionamento), devido à complexidade computacional de eliminar coleções intermediárias. Essa operação exige decisões adicionais sobre a reconfiguração de relacionamentos e a fusão de coleções adjacentes, o que aumenta o custo algorítmico e foi evitado para manter a eficiência do `VNS`;

- Cada esquema individual admite apenas uma instância de cada coleção, embora a mesma coleção possa reaparecer em diferentes esquemas de uma solução

6.4 Ameaças

O presente trabalho, embora cuidadosamente conduzido, está sujeito a algumas ameaças que devem ser consideradas ao interpretar os resultados obtidos.

Uma possível ameaça reside na definição e operacionalização das métricas utilizadas. Embora as quatro métricas escolhidas (completude, padrão de acesso, custo de recuperação e redundância) sejam fundamentadas na literatura, a forma como foram aplicadas

pode variar em relação a outras abordagens. Além disso, a construção do banco de dados sintético, embora cuidadosamente projetada para generalizar comportamentos, ainda é uma abstração da realidade e pode não capturar todas as nuances dos dados reais.

A escolha dos cenários de estudo pode influenciar os resultados. Ainda que os casos de uso de Gómez et al. (2018) e Kuszera et al. (2020) tenham sido selecionados por já aplicarem métricas em seus experimentos, a forma como os dados foram gerados ou como os esquemas foram implementados neste trabalho pode introduzir vieses. Também é possível que ajustes manuais ou decisões de modelagem específicas tenham impactado a execução das consultas ou os resultados das métricas.

Os resultados obtidos são baseados em dados sintéticos e em cenários específicos. Portanto, a generalização para outros domínios de aplicação ou cargas de trabalho reais deve ser feita com cautela. Ainda que o banco de dados sintético tenha sido projetado para simular padrões comuns em NoSQL, ele pode não refletir com exatidão os comportamentos de sistemas em produção com dados altamente heterogêneos e dinâmicos.

6.5 Considerações Finais

Para implementar o `VNS`, foi necessário representar os esquemas como matrizes e as consultas como dicionários, tornando-os adequados às operações do algoritmo. Assim, o `VNS` recebe como entrada uma solução inicial (conjunto de esquemas) e um conjunto de consultas. A avaliação quantitativa das soluções é feita aplicando coeficientes de ponderação às métricas de avaliação, integradas à $funcao\_objetivo()$ do algoritmo, o que possibilita uma análise robusta do desempenho das soluções geradas. A exploração e análise das possibilidades de soluções foram realizadas exclusivamente no nível lógico, sem a necessidade de considerar aspectos do nível físico, como frequência de acesso às consultas, índices ou espaço de armazenamento. Essa abordagem proporcionou maior flexibilidade e independência na geração e avaliação dos esquemas, focando na otimização lógica das soluções. Além disso, o algoritmo busca otimizar a solução tendo como objetivo principal a maximização da eficiência dos esquemas enquanto garante o atendimento total a todas as consultas.

Capítulo 7

Conclusões

O projeto de esquemas em bancos de dados NoSQL orientados a documentos ainda carece de um padrão consolidado. Na prática, os projetistas recorrem a diagramas ER, UML, XML, JSON e outras notações nos níveis conceitual e lógico para desenhar esquemas. Mesmo assim, os esquemas resultantes nem sempre são os mais ótimos, pois a qualidade final depende da experiência do projetista e da própria complexidade combinatória do problema, que cresce exponencialmente com o número de coleções e com os tipos de relacionamentos. A abordagem proposta prioriza o atendimento integral da métrica de *completitude*, assegurando que todas as consultas predefinidas sejam plenamente atendidas. Além disso, busca, sempre que possível, minimizar as métricas de *padrão de acesso* e *redundância*, concentrando-se exclusivamente no nível lógico. Para isso, adotou-se uma estratégia baseada em meta-heurística, utilizando o algoritmo VNS.

O objetivo geral desta tese foi apresentado mediante a implementação do VNS. Essa estratégia permitiu gerar esquemas otimizados que atendem integralmente ao conjunto de consultas, com redução significativa da métrica *redundância* e *padrão de acesso* da solução. A abordagem no nível lógico eliminou a dependência de avaliações custosas no nível físico, tornando o método acessível a projetistas com diferentes níveis de experiência técnica.

Os objetivos específicos foram atendidos, contribuindo para a robustez da solução proposta:

- As relações aninhadas permitem uma recuperação de dados mais rápida, pois todos os dados necessários estão contidos em um único documento, eliminando a necessidade de consultas adicionais para obter dados referenciados. No entanto, embora a recuperação de dados seja mais rápida em relacionamentos aninhados, a literatura não menciona precisamente o quão rápida é essa recuperação em comparação a uma relação por referência. É por isso foram estabelecidos coeficientes de ponderação para os relacionamentos referenciados e aninhados, permitindo uma avaliação do

impacto dessas relações na eficiência dos esquemas, conforme publicado em (Vera-Olivera et al., 2023). Esses coeficientes enriqueceram a função objetivo do VNS, garantindo avaliações alinhadas aos requisitos de desempenho;

- As métricas de *completude*, *padrão de acesso*, *custo de acesso* e *redundância* foram desenvolvidas e integradas ao algoritmo, possibilitando a quantificação objetiva da qualidade dos esquemas sem recorrer a operações CRUD. Essas métricas, validadas em cenários experimentais, demonstraram ser robustas e adequadas para o contexto NoSQL (Vera-Olivera & Holanda, 2024);

- Operadores de perturbação foram projetados para alterar sistematicamente as soluções, promovendo uma exploração diversificada do espaço de busca e evitando a convergência para mínimos locais. Esses operadores foram cruciais para a eficácia do VNS, garantindo a descoberta de soluções próximas do ótimo global;

- O VNS foi implementado integrando as métricas de avaliação, coeficientes de ponderação e operadores de perturbação

Em síntese, os resultados obtidos confirmam que a combinação de métricas, coeficientes de ponderação e busca guiada por meta-heurística constitui uma alternativa viável e eficaz para o projeto de esquemas em bancos de dados NoSQL orientados a documentos. A estratégia proposta não apenas reduz o esforço manual e a dependência da experiência do projetista, como também oferece um processo de modelagem sistemático e reproduzível, capaz de se adaptar a diferentes conjuntos de consultas e requisitos. Além de validar a pertinência do VNS para problemas de otimização de esquemas, esta tese estabelece um referencial metodológico que pode ser estendido a outras métricas, a cenários de avaliação em nível físico ou a diferentes paradigmas NoSQL. Dessa forma, contribui-se para o avanço do estado da arte e cria-se uma base sólida para investigações futuras voltadas à automação do desenho de bancos de dados em ambientes dinâmicos e orientados a grandes volumes de dados.

7.1 Resultados Acadêmicos

Durante o desenvolvimento desta tese, foram obtidas as seguintes publicações relacionadas diretamente à pesquisa:

- "Métricas para Análise de Esquemas em Banco de Dados NoSQL Orientado a Documentos." Simpósio Brasileiro de Banco de Dados (SBBD). SBC, 2024;

- "Análise de desempenho em banco de dados nosql orientado a documentos: Um Índice para comparação de modelos de dados." Simpósio Brasileiro de Banco de Dados (SBBD). SBC, 2023;

- "Data modeling and nosql databases-a systematic mapping review." ACM Computing Surveys (CSUR) 54.6 (2021): 1-26

7.2 Trabalhos Futuros

Para trabalhos futuros que podem ampliar a robustez, generalização e aplicabilidade prática da abordagem proposta para otimização de esquemas em bancos de dados `NoSQL` orientados a documentos:

- Este trabalho calibrou coeficientes de ponderação e métricas de avaliação com base exclusivamente em operações de leitura. Pesquisas futuras podem incorporar operações de escrita, atualização e exclusão na definição de métricas e coeficientes, permitindo uma avaliação mais abrangente do desempenho dos esquemas otimizados. Essa extensão exigirá a reformulação da função objetivo do algoritmo `VNS` para refletir o impacto de diferentes tipos de operações, potencialmente utilizando técnicas de aprendizado de máquina para modelar padrões de acesso dinâmicos;

- O trabalho focou exclusivamente no paradigma orientado a documentos. Estudos futuros podem adaptar a abordagem para outros paradigmas `NoSQL`, como chave-valor, colunar ou de grafos, que apresentam desafios distintos na modelagem e otimização de esquemas. Essa extensão exigirá a reformulação das métricas e operadores de perturbação para refletir as características específicas de cada paradigma, bem como a avaliação da transferibilidade do `VNS` em diferentes contextos de dados;

- A autorreferência entre coleções, o aninhamento com profundidade superior a dois níveis e a presença de atributos internos nas coleções não foram considerados no escopo deste trabalho, uma vez que esses elementos aumentam a complexidade computacional do problema. No entanto, reconhece-se que esses três fatores podem impactar significativamente o desempenho da modelagem e a eficiência dos esquemas gerados. Assim, tais aspectos são apontados como relevantes e promissores para investigações futuras, visando uma abordagem mais abrangente e aderente à complexidade dos cenários encontrados em bancos de dados NoSQL do mundo real;

- Propõe-se como trabalho futuro a exploração de diferentes configurações do sistema gerenciador de banco de dados `MongoDB`, considerando aspectos como particiona-

mento, replicação, índices compostos e compactação de dados, com o objetivo de avaliar como essas opções influenciam a eficiência dos esquemas no nível físico;

- Embora as métricas propostas neste trabalho tenham sido aplicadas no contexto de bancos de dados orientados a documentos, seu potencial de generalização permite a extrapolação para outros paradigmas `NoSQL`, como bancos de dados em grafos, chave-valor e orientados a colunas. Dessa forma, futuras pesquisas podem estender e especializar essas métricas para diferentes modelos, respeitando as particularidades de cada paradigma e ampliando a aplicabilidade da abordagem

Referencias

Abadi, D., Ailamaki, A., Andersen, D., Bailis, P., Balazinska, M., Bernstein, P. A., Boncz, P., Chaudhuri, S., Cheung, A., Doan, A., et al. (2022). The seattle report on database research. *Communications of the ACM*, *65*(8), 72–79.

Abdelhedi, F., Ait Brahim, A., Atigui, F., & Zurfluh, G. (2017). MDA-based approach for NoSQL databases modelling. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *10440 LNCS*, 88–102. https://doi.org/10.1007/978-3-319-64283-3_7

Abdelhedi, F., Ait Brahim, A., & Zurfluh, G. (2018). Formalizing the mapping of UML conceptual schemas to column-oriented databases. *International Journal of Data Warehousing and Mining*, *14*(3), 44–68. https://doi.org/10.4018/IJDWM.2018070103

Abdelhedi, F., Brahim, A., Atigui, F., & Zurfluh, G. Big data and knowledge management: How to implement conceptual models in NoSQL systems'. In: *3*. 2016, 235–240.

Abdelhedi, F., Brahim, A., Atigui, F., & Zurfluh, G. Logical unified modeling for NoSQL databases. In: *1*. 2017, 249–256.

Abdelhedi, F., Brahim, A., Atigui, F., & Zurfluh, G. UMLtoNoSQL: Automatic transformation of conceptual schema to NoSQL databases. In: *2017-October*. 2018, 272–279. https://doi.org/10.1109/AICCSA.2017.76

Abdelhedi, F., Brahim, A. A., Rajhi, H., Ferhat, R. T., & Zurfluh, G. (2021). Automatic Extraction of a Document-oriented NoSQL Schema. *ICEIS (1)*, 192–199.

Abualigah, L., Elaziz, M. A., Hussien, A. G., Alsalibi, B., Jalali, S. M. J., & Gandomi, A. H. (2021). Lightning search algorithm: a comprehensive survey. *Applied Intelligence*, *51*(4), 2353–2376.

Akintoye, S., Bagula, A., Isafiade, O., Djemaiel, Y., & Boudriga, N. (2019). Data model for cloud computing environment. *Lecture Notes of the Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering, LNICST*, *275*, 199–215. https://doi.org/{10.1007/978-3-030-16042-5_19}

Andor, C., Pârv, B., & Suciu, D. (2019). Using Latency Metrics in NoSQL Database Performance Benchmarking. *Studia Universitatis Babeş-Bolyai Informatica*, *64*(1), 39–50.

Angles, R. The property graph database model. In: *2100*. 2018.

Banerjee, S., & Sarkar, A. (2016). Ontology Driven Meta-Modeling for NoSQL Databases: A Conceptual Perspective. *International Journal of Software Engineering and its Applications*, *10*(12), 41–64. https://doi.org/10.14257/ijseia.2016.10.12.05

Banerjee, S., & Sarkar, A. Logical level design of NoSQL databases. In: 2017, 2360–2365. https://doi.org/10.1109/TENCON.2016.7848452

Bansal, N., Sachdeva, S., & Awasthi, L. K. (2023). A workload-driven approach for automatic schema generation for document stores. *Proceedings of the 6th Joint International Conference on Data Science & Management of Data (10th ACM IKDD CODS and 28th COMAD)*, 133–133.

Bermbach, D., Müller, S., Eberhardt, J., & Tai, S. Informed Schema Design for Column Store-Based Database Services. In: 2016, 163–172. https://doi.org/10.1109/SOCA.2015.29

Bugiotti, F., Cabibbo, L., Atzeni, P., & Torlone, R. (2014). Database design for NoSQL systems. *International Conference on Conceptual Modeling*, 223–231.

Carvalho, I., Sá, F., & Bernardino, J. (2023). Performance evaluation of NoSQL document databases: couchbase, CouchDB, and MongoDB. *Algorithms*, *16*(2), 78.

Chen, L., Davoudian, A., & Liu, M. (2022). A workload-driven method for designing aggregate-oriented NoSQL databases. *Data & Knowledge Engineering*, *142*, 102089.

Chillón, A., Morales, S., Ruiz, D., & Molina, J. Exploring the visualization of schemas for aggregate-oriented nosql databases? In: *1979*. 2017, 72–85.

Chiș-Rațiu, A., & Buchmann, R. Design and implementation of a diagrammatic tool for creating RDF graphs. In: *2238*. 2018, 37–48.

Daniel, G., Sunyé, G., & Cabot, J. (2016). UMLtographDB: Mapping conceptual schemas to graph databases. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *9974 LNCS*, 430–444. https://doi.org/10.1007/978-3-319-46397-1_33

De Lima, C., & Dos Santos Mello, R. A workload-driven logical design approach for NoSQL document databases. In: 2015. https://doi.org/10.1145/2837185.2837218

De Lima, C., & dos Santos Mello, R. (2015). A workload-driven logical design approach for NoSQL document databases. *Proceedings of the 17th international conference on information integration and web-based applications & services*, 1–10.

De Virgilio, R., Maccioni, A., & Torlone, R. (2014). Model-driven design of graph databases. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *8824*, 172–185.

Elmasri, R. (2008). *Fundamentals of database systems*. Pearson Education India.

Gómez, P., Casallas, R., & Roncancio, C. (2016). Data schema does matter, even in NoSQL systems! *2016 IEEE Tenth International Conference on Research Challenges in Information Science (RCIS)*, 1–6.

Gómez, P., Roncancio, C., & Casallas, R. (2018). Towards quality analysis for document oriented bases. *International Conference on Conceptual Modeling*, 200–216.

Gómez, P., Roncancio, C., & Casallas, R. (2021). Analysis and evaluation of document-oriented structures. *Data & Knowledge Engineering*, *134*, 101893.

Győrödi, C. A., Dumşe-Burescu, D. V., Zmaranda, D. R., & Győrödi, R. Ş. (2022). A comparative study of MongoDB and document-based MySQL for big data application data management. *Big Data and Cognitive Computing*, *6*(2), 49.

Hamouda, S., & Zainol, Z. Document-Oriented Data Schema for Relational Database Migration to NoSQL. In: *2018-January*. 2018, 43–50. https://doi.org/10.1109/Innovate-Data.2017.13

Hansen, P., & Mladenović, N. (2001). Variable neighborhood search: Principles and applications. *European Journal of Operational Research*, *130*(3), 449–467. https://doi.org/https://doi.org/10.1016/S0377-2217(00)00100-4

Hansen, P., Mladenović, N., Brimberg, J., & Pérez, J. A. M. (2019). *Variable neighborhood search*. Springer.

Hewasinghage, M., Seghouani, N., & Bugiotti, F. (2018). Modeling strategies for storing data in distributed heterogeneous NoSQL databases. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *11157 LNCS*, 488–496. https://doi.org/10.1007/978-3-030-00847-5_35

Hewasinghage, M., Abelló, A., Varga, J., & Zimányi, E. (2020). DocDesign: cost-based database design for document stores. *Proceedings of the 32nd International Conference on Scientific and Statistical Database Management*, 1–4.

Hewasinghage, M., Abelló, A., Varga, J., & Zimányi, E. (2021). A cost model for random access queries in document stores. *The VLDB Journal*, *30*(4), 559–578.

Hewasinghage, M., Nadal, S., Abelló, A., & Zimányi, E. (2023). Automated database design for document stores with multicriteria optimization. *Knowledge and Information Systems*, *65*(7), 3045–3078.

Imam, A., Basri, S., Ahmad, R., Aziz, N., & Gonzalez-Aparicio, M. New cardinality notations and styles for modeling NoSQL document-store databases. In: *2017-December*. 2017, 2765–2770. https://doi.org/10.1109/TENCON.2017.8228332

Imam, A., Basri, S., Ahmad, R., & González Aparicio, M. T. (2019). Schema proposition model for NoSQL applications. *Recent Trends in Data Science and Soft Computing*.

Imam, A., Basri, S., Ahmad, R., Watada, J., & González-Aparicio, M. (2018). Automatic schema suggestion model for NoSQL document-stores databases. *Journal of Big Data*, *5*(1). https://doi.org/10.1186/s40537-018-0156-1

Imam, A. A., & Basri, e. a. (2018). Data modeling guidelines for NoSQL document-store databases. *International Journal of Advanced Computer Science and Applications, 9*.

Imam, A. A., Basri, S., Ahmad, R., Wahab, A. A., González-Aparicio, M. T., Capretz, L. F., Alazzawi, A. K., & Balogun, A. O. (2020). Dsp: Schema design for non-relational applications. *Symmetry*, *12*(11), 1799.

Imam, A. A., Basri, S., Ahmad, R., Watada, J., & González-Aparicio, M. T. (2018). Automatic schema suggestion model for NoSQL document-stores databases. *Journal of Big Data*, *5*, 1–17.

James, G., Witten, D., Hastie, T., Tibshirani, R., et al. (2013). *An introduction to statistical learning* (Vol. 112). Springer.

Kaur, K., & Rani, R. Modeling and querying data in NoSQL databases. In: 2013, 1–7. https://doi.org/10.1109/BigData.2013.6691765

Kitchenham, B., Pretorius, R., Budgen, D., Brereton, O. P., Turner, M., Niazi, M., & Linkman, S. (2010). Systematic literature reviews in software engineering–a tertiary study. *Information and software technology*, *52*(8), 792–805.

Kuszera, E. M., Peres, L. M., & Didonet Del Fabro, M. (2020). Query-based metrics for evaluating and comparing document schemas. *International Conference on Advanced Information Systems Engineering*, 530–545.

Kuszera, E. M., Peres, L. M., & Del Fabro, M. D. (2022). Exploring data structure alternatives in the RDB to NoSQL document store conversion process. *Information Systems*, *105*, 101941.

la Vega, A., García-Saiz, D., Blanco, C., Zorrilla, M., & Sánchez, P. (2018). Mortadelo: A model-driven framework for NoSQL database design. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *11163 LNCS*, 41–57. https://doi.org/10.1007/978-3-030-00856-7_3

la Vega, A., García-Saiz, D., Blanco, C., Zorrilla, M., & Sánchez, P. (2020). Mortadelo: Automatic generation of NoSQL stores from platform-independent data models. *Future Generation Computer Systems*, *105*, 455–474.

Li, X., Ma, Z., & Chen, H. QODM: A query-oriented data modeling approach for NoSQL databases. In: 2014, 338–345. https://doi.org/10.1109/WARTIA.2014.6976265

Li, Y., Gu, P., & Zhang, C. (2014). Transforming UML class diagrams into HBase based on meta-model. *2014 International Conference on Information Science, Electronics and Electrical Engineering*, *2*, 720–724.

Lima, C., & Mello, R. (2016). On proposing and evaluating a NoSQL document database logical approach. *International Journal of Web Information Systems*, *12*(4), 398–417. https://doi.org/10.1108/IJWIS-04-2016-0018

Llano-Ríos, T. F., Khalefa, M., & Badia, A. (2020). Evaluating NoSQL systems for decision support: An experimental approach. *2020 IEEE International Conference on Big Data (Big Data)*, 2802–2811.

Martins de Sousa, V., & del Val Cura, L. Logical design of graph databases from an entity-relationship conceptual model. In: 2018, 183–189. https://doi.org/{10.1145/3282373.3282375}

Mior, M. J., Salem, K., Aboulnaga, A., & Liu, R. (2017). NoSE: Schema design for NoSQL applications. *IEEE Transactions on Knowledge and Data Engineering*, *29*(10), 2275–2289.

Mior, M. Automated schema design for NoSQL databases. In: 2014, 41–45. https://doi.org/10.1145/2602622.2602624

Mladenović, N., & Hansen, P. (1997). Variable neighborhood search. *Computers & operations research*, *24*(11), 1097–1100.

Mozaffari, M., & Nazemi, E. (2023). Automatic NoSQL Schema Design: A Workload-Driven Schema Design Approach for NoSQL Wide Column Stores. *Journal of Soft Computing and Information Technology*, *11*(4), 19–35.

Muhammad, A. Y., & Azizah, F. N. (2022). Conversion of entity-relationship model to NoSQL document-oriented database logical model using workload information and entity update frequency. *2022 9th International Conference on Advanced Informatics: Concepts, Theory and Applications (ICAICTA)*, 1–6.

Naka, E., & Guliashki, V. (2021). Optimization techniques in data management: a survey. *Proceedings of the 2021 7th International Conference on Computing and Data Engineering*, 8–13.

Nogueira, I., Romdhane, M., & Darmont, J. Modeling data lake metadata with a data vault. In: 2018, 253–261. https://doi.org/10.1145/3216122.3216130

Orel, O., Zakošek, S., & Baranović, M. (2017). Property oriented relational-to-graph database conversion [Konverzija relacijskih u grafovske baze podataka orijentirana na svojstva]. *Automatika*, *57*(3), 836–845. https://doi.org/10.7305/automatika.2017.02.1581

Petersen, K., Vakkalanka, S., & Kuzniarz, L. (2015). Guidelines for conducting systematic mapping studies in software engineering: An update. *Information and Software Technology*, *64*, 1–18.

Pokorný, J. Conceptual and database modelling of graph databases. In: *11-13-July-2016*. 2016, 370–377. https://doi.org/10.1145/2938503.2938547

Reis, D. G., Gasparoni, F. S., Holanda, M., Victorino, M., Ladeira, M., & Ribeiro, E. O. (2018). An evaluation of data model for NoSQL document-based databases. *Trends and Advances in Information Systems and Technologies: Volume 1 6*, 616–625.

Reniers, V., Van Landuyt, D., Rafique, A., & Joosen, W. Schema design support for semi-structured data: Finding the sweet spot between NF and De-NF. In: *2018-January*. 2018, 2921–2930. https://doi.org/10.1109/BigData.2017.8258261

Reniers, V., Van Landuyt, D., Rafique, A., & Joosen, W. (2020). A workload-driven document database schema recommender (DBSR). *International Conference on Conceptual Modeling*, 471–484.

Roy-Hubara, N., Rokach, L., Shapira, B., & Shoval, P. (2017). Modeling Graph Database Schema. *IT Professional*, *19*(6), 34–43. https://doi.org/10.1109/MITP.2017.4241458

Roy-Hubara, N., Rokach, L., Shapira, B., & Shoval, P. (2018). Evaluation of a design method for graph database. *Lecture Notes in Business Information Processing*, *318*, 291–303. https://doi.org/10.1007/978-3-319-91704-7_19

Roy-Hubara, N., Sturm, A., & Shoval, P. (2023). Designing NoSQL databases based on multiple requirement views. *Data & knowledge engineering*, *145*, 102149.

Saha, S., & Sachdeva, S. (2024). Designing Document stores using Feature Model and Application Workload. *2024 Third International Conference on Electrical, Electronics, Information and Communication Technologies (ICEEICT)*, 1–5.

Santisteban, J., & Ticona-Herrera, R. Modeling a persistent graph. In: 2018, 15–22. https://doi.org/10.1109/MICAI-2017.2017.00011

Santos, M., & Costa, C. (2016). Data models in NoSQL databases for big data contexts. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *9714 LNCS*, 475–485. https://doi.org/10.1007/978-3-319-40973-3_48

Schram, A., & Anderson, K. MySQL to NoSQL data modeling challenges in supporting scalability. In: 2012, 191–202. https://doi.org/10.1145/2384716.2384773

scikit-learn. (2024). $sklearn.linear_model.LinearRegression - scikit - learndocumentation$ [Accessed: 2025-04-30]. scikit-learn. https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html

Sedlmeier, M., & Gogolla, M. (2014). Design and prototypical implementation of an integrated graph-based conceptual data model. *Frontiers in Artificial Intelligence and Applications*, *272*, 376–395. https://doi.org/10.3233/978-1-61499-472-5-376

Shah, M., Kothari, A., & Patel, S. (2022). Influence of schema design in nosql document stores. *Mobile Computing and Sustainable Informatics: Proceedings of ICMCSI 2021*, 435–452.

Shin, K., Hwang, C., & Jung, H. (2017). NoSQL database design using UML conceptual data model based on peter chen's framework. *International Journal of Applied Engineering Research*, *12*(5), 632–636.

Shoval, P. A method for modeling a schema for graph databases. In: *8*. 2018, 99–104.

Simsion, G., & Witt, G. (2004). *Data modeling essentials*. Elsevier.

Suárez-Otero, P., Suárez-Cabal, M., & Tuya, J. Leveraging conceptual data models for keeping cassandra database integrity. In: 2018, 398–403.

Vágner, A. Store and visualize EeR in Neo4j. In: 2018. https://doi.org/10.1145/3284557.3284694

Vajk, T., Deák, L., Fekete, K., & Mezei, G. (2013). Automatic NoSQL schema development: A case study. *Artificial Intelligence and Applications*, 656–663.

Van Erven, G., Silva, W., Carvalho, R., & Holanda, M. (2018). GRAPHED: A graph description diagram for graph databases. *Advances in Intelligent Systems and Computing*, *745*, 1141–1151. https://doi.org/10.1007/978-3-319-77703-0_111

Varga, V., Jánosi-Rancz, K., & Kálmán, B. (2016). Conceptual design of document NoSQL database with formal concept analysis. *Acta Polytechnica Hungarica*, *13*(2), 229–248.

Varga, V., Săcărea, C., & Molnar, A. (2018). Conceptual Graphs Based Modeling of Semistructured Data. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *10872 LNAI*, 167–175. https://doi.org/10.1007/978-3-319-91379-7_13

Vera, H., & et. al. Data modeling for NoSQL document-oriented databases. In: *1478*. 2015, 129–135.

Vera, H., Boaventura, W., Holanda, M., Guimaraes, V., & Hondo, F. (2015). Data modeling for NoSQL document-oriented databases. *CEUR Workshop Proceedings*, *1478*, 129–135.

Vera-Olivera, H., Alvarez-Mamani, E., & Holanda, M. (2023). Análise de Desempenho em Banco de Dados NoSQL Orientado a Documentos: Um Índice para Comparação de

Modelos de Dados. *Anais do XXXVIII Simpósio Brasileiro de Bancos de Dados*, 26–38. https://doi.org/10.5753/sbbd.2023.231721

Vera-Olivera, H., & et. al. (2021). Data Modeling and NoSQL Databases - A Systematic Mapping Review. *ACM Comput. Surv.*, *54*(6). https://doi.org/10.1145/3457608

Vera-Olivera, H., & Holanda, M. (2024). Métricas para Análise de Esquemas em Banco de Dados NoSQL Orientado a Documentos. *Anais do XXXIX Simpósio Brasileiro de Bancos de Dados*, 381–393. https://doi.org/10.5753/sbbd.2024.240646

Villa, F., Moreno, F., & Guzmán, J. (2018). An Analysis of a Methodology that Transforms the Entity-Relationship Model into a Conceptual Model for a Graph Database. *International Conference for Emerging Technologies in Computing*, 70–83.

Wakuta, Y., Mior, M., Zenmyo, T., Sasaki, Y., & Onizuka, M. (2023). NoSQL Schema Design for Time-Dependent Workloads. *arXiv preprint arXiv:2303.16577*.

Yoo, K., Park, S., & Lee, S.-G. RDB2Graph: A generic framework for modeling relational databases as graphs. In: *1312*. 2014, 148–151.

Zhang, Z. (2017). Graph Databases for Knowledge Management. *IT Professional*, *19*(6), 26–32. https://doi.org/10.1109/MITP.2017.4241463

Zhao, G., Huang, W., Liang, S., & Tang, Y. (2013). Modeling MongoDB with relational model. *2013 Fourth International Conference on Emerging Intelligent Data and Web Technologies*, 115–121.

Zhao, M., Liu, Y., & Zhou, P. Towards a systematic approach to graph data modeling: Scenario-based design and experiences. In: *2016-January*. 2016, 634–637. https://doi.org/10.18293/SEKE2016-119