Comparação Sistemática da Melhoria de Desempenho na Migração de Bancos de Dados Relacionais para NoSQL

Kaique Schio[1], Gabriel Martinez[1], Pedro Ayres[1], Lucas Segabinazzi[1], Vinicius Pereira[1], Davi Pereira[1], Vitor Balsanello[1], Maicon Bernardino[1]

[1]Universidade Federal do Pampa (UNIPAMPA)
97.546-550 – Alegrete – RS – Brasil

{kaiqueschio, gabrieldutra, pedrofantinel, lucassegabinazzi, viniciusdp, davipereira, vitorbalsanello}.aluno@unipampa.edu.br, bernardino@acm.org

Abstract. *The rapid growth of Big Data has exposed scalability and performance limitations in traditional relational databases such as MySQL, motivating the adoption of NoSQL solutions like MongoDB. Although NoSQL offers flexibility and horizontal scalability, migrating from relational systems involves significant challenges, including schema redesign and changes in transactional properties. This paper proposes an experimental study that systematically compares the performance improvement in the migration from MySQL to MongoDB. The methodology includes a controlled test environment, the generation of a large-scale synthetic e-commerce dataset, an ETL-based migration process, and benchmarking workloads focused on read, write, and aggregation operations. The study aims to provide quantitative evidence and preliminary results to support informed decision-making in database migration projects.*

Resumo. *O rápido crescimento do Big Data evidenciou limitações de escalabilidade e desempenho em bancos de dados relacionais tradicionais, impulsionando a adoção de soluções NoSQL, como o MongoDB. Embora o NoSQL ofereça maior flexibilidade e escalabilidade horizontal, a migração a partir de sistemas relacionais envolve desafios significativos, incluindo o redesenho de esquemas e mudanças nas propriedades transacionais. Este artigo propõe um estudo experimental que compara sistematicamente o desempenho na migração de MySQL para MongoDB. A metodologia contempla um ambiente controlado de testes, a geração de um conjunto de dados sintético em larga escala para comércio eletrônico, um processo de migração baseado em ETL e a execução de benchmarks focados em operações de leitura, escrita e agregação, apresentando resultados preliminares para apoiar a tomada de decisão em projetos de migração de bancos de dados.*

1. Introdução

A crescente digitalização dos serviços gerou um volume massivo de dados, um cenário conhecido como *Big Data*. Nesse contexto, bancos de dados relacionais tradicionais, apesar de sua robustez, enfrentam algumas limitações [Basso and Gomes 2015]. Como alternativa, o paradigma NoSQL surgiu para oferecer maior flexibilidade e escalabilidade horizontal, sendo ideal para dados não estruturados. Soluções como o MongoDB, por exemplo, têm sido adotadas por empresas que buscam mais agilidade no processamento de informações em larga escala [Friess 2013].

Diante desses benefícios, muitas organizações consideram migrar seus sistemas de modelos relacionais para plataformas NoSQL. Contudo, essa transição é um processo complexo que vai além da transferência de registros, exigindo a readequação de esquemas e a garantia de integridade dos dados. A decisão se torna mais crítica ao considerar a troca de paradigmas transacionais, substituindo as garantias de consistência imediata do modelo ACID pelo modelo BASE, que prioriza a disponibilidade [Murari and Cunha 2014].

Enquanto estudos recentes, como o de [Garbin et al. 2024], têm focado em automatizar e medir a complexidade do processo de migração (ETL), identificando gargalos como o número de chaves estrangeiras, ainda persiste uma lacuna na análise quantitativa do resultado dessa migração. Faltam estudos experimentais que comparem o desempenho de cargas de trabalho (leitura, escrita e agregação) nos sistemas pré e pós-migração. É esta lacuna que o presente artigo visa preencher.

Nosso objetivo é fornecer uma análise quantitativa de cada abordagem em cargas de trabalho específicas (leitura, escrita e agregação). Com isso, buscamos oferecer subsídios para apoiar a tomada de decisão em projetos de migração de bancos de dados.

O artigo atual está organizado da seguinte forma: a Seção 2 aborda a fundamentação teórica, discutindo os paradigmas relacional e NoSQL. A Seção 3 apresenta uma revisão dos trabalhos relacionados sobre o tema. A Seção 4 detalha o desenvolvimento metodológico do estudo, a configuração do ambiente e modelagem de dados. A Seção 5 trata do processo de migração de dados (ETL). A Seção 6 apresenta o protocolo de benchmark de desempenho. A Seção 6.2 discute os resultados preliminares obtidos. Por fim, a Seção 7 traz as considerações finais do trabalho.

2. Fundamentação Teórica

2.1. Bancos de Dados Relacionais e Não Relacionais

Os bancos de dados relacionais (RDBMS), definidos formalmente em [Codd 1970], estruturam as informações em tabelas com esquemas fixos e relacionamentos bem definidos, utilizando a linguagem SQL (*Structured Query Language*) para consulta e manipulação dos dados. Esses sistemas baseiam-se nas propriedades ACID [Gray 1981] (Atomicidade, Consistência, Isolamento e Durabilidade), garantindo integridade referencial e transações confiáveis, o que os torna amplamente utilizados em aplicações tradicionais, como sistemas bancários, ERPs e soluções corporativas que exigem alta consistência dos dados.

Apesar de sua robustez em cenários com dados estruturados e relacionamentos complexos, os RDBMS apresentam limitações em contextos modernos. A rigidez do esquema dificulta adaptações rápidas a mudanças nos requisitos de negócio, e o tratamento de grandes volumes de dados heterogêneos ou semiestruturados pode comprometer a escalabilidade e o desempenho.

Nesse contexto, surgem os bancos de dados não relacionais, definidos por [Garbin et al. 2024] como sistemas de gerenciamento de dados desenvolvidos para contornar as limitações do modelo relacional. Esses sistemas suportam diferentes modelos de dados, orientados a documentos (MongoDB, CouchDB), chave-valor (Redis, DynamoDB), grafos (Neo4j, ArangoDB) e orientados a colunas (Cassandra, HBase).

Bancos não relacionais caracterizam-se pela maior flexibilidade de esquema, permitindo que a estrutura dos dados evolua conforme as necessidades da aplicação, sem

a necessidade de migrações complexas. Além disso, oferecem escalabilidade horizontal nativa, possibilitando a distribuição de dados e cargas de processamento entre múltiplos servidores de forma eficiente. Essas características tornam os sistemas NoSQL especialmente adequados para aplicações modernas, sistemas distribuídos, redes sociais, Internet das Coisas (IoT) e cenários de Big Data.

2.2. Processo de Migração de RDBMS para NoSQL

A decisão sobre a metodologia de migração depende de uma série de fatores e resulta em grande complexidade técnica [Garbin et al. 2024], que decorre das diferenças paradigmáticas: bancos relacionais são baseados em normalização, relacionamentos explícitos por meio de chaves estrangeiras e esquemas rígidos, enquanto bancos NoSQL privilegiam desnormalização, agregação de dados relacionados em estruturas únicas e esquemas flexíveis. Essas diferenças impactam diretamente o design conceitual, a organização física do armazenamento e os padrões de consulta implementados na aplicação.

O processo de migração segue três etapas principais: extração dos dados do sistema relacional, preservando a semântica e os relacionamentos; transformação, que inclui desnormalização estratégica, conversão para estruturas NoSQL (como documentos aninhados, grafos ou famílias de colunas) e adaptação de restrições de integridade; e carga dos dados no novo sistema, garantindo consistência e desempenho.

3. Trabalhos Relacionados

Atualmente, existem várias abordagens que exploram desde a avaliação de características de bancos NoSQL até os estudos de caso experimentais de desempenho e desenvolvimento de ferramentas para ajudar a migração.

Estudos como o de [Friess 2013], focaram na avaliação de múltiplas ferramentas NoSQL, como Redis, Cassandra e MongoDB, analisando alguns critérios como armazenamento e arquitetura. Foi concluído que o MongoDB é uma opção viável para quem migra de banco de dados relacionais, por conta da simplicidade da instalação e uso. De forma análoga, [Tomio 2015] explorou o funcionamento dos softwares NoSQL para entender as suas diferenças em relação aos sistemas tradicionais, para fornecer uma visão mais clara sobre os benefícios e as limitações de cada ferramenta.

Também houveram diversos trabalhos que realizaram comparações de desempenho. [Basso and Gomes 2015] compararam MySQL e MongoDB em um cenário de Big Data, e concluíram que o MongoDB apresentou vantagens substanciais em tempo de resposta de consultas, em parte devido à sua estrutura que não exige a junção de múltiplas tabelas. De forma semelhante, [Rabelo and Cândido 2017] observaram que o MongoDB demonstrou desempenho superior ao PostgreSQL em tempo de resposta e escalabilidade.

Por outro lado, através do estudo de [Souza and Santana 2017] foi revelado um cenário onde o MySQL superou o RavenDB, e isso mostra que a superioridade de desempenho do NoSQL não é uma garantia e pode depender de caso de uso e de uma tecnologia específica.

Outros estudos compararam diferentes bancos de dados NoSQL entre si. Por exemplo, utilizando a ferramenta YCSB, [Melo 2016] avaliou o desempenho de MongoDB, Couchbase e OrientDB. Em sua conclusão, o Couchbase apresentou um melhor

tempo de resposta. [Barros et al. 2017] focou no contexto de dados e comparou MongoDB e Cassandra com o MySQL, e verificou-se que o MongoDB obteve o menor tempo de inserção.

O estudo de [Garbin et al. 2024] propõe uma metodologia automatizada para migração de PostgreSQL para MongoDB baseada em regras de desnormalização orientadas por cardinalidade. Os resultados indicam que o principal fator de impacto no tempo de migração não é o volume de dados, mas a quantidade de chaves estrangeiras, devido à complexidade das verificações necessárias para definição da estratégia de agregação.

4. Metodologia

Esta seção apresenta a base metodológica do estudo, descrevendo a estrutura experimental construída para garantir a transparência e a reprodutibilidade da pesquisa.

4.1. Ambiente Experimental

4.1.1. Configuração de Hardware e Rede

Para evitar contenção de recursos, adotou-se uma arquitetura com duas máquinas distintas: um nó cliente, responsável pela execução do benchmark e coleta de métricas, e um nó de banco de dados, dedicado à hospedagem do SGBD avaliado (MySQL ou MongoDB). A Tabela 1 apresenta um resumo das especificações deste ambiente.

Tabela 1. Especificações do Ambiente Experimental

Componente	Nó Cliente	Nó de Banco de Dados
Sistema Operacional	Ubuntu Server 22.04 LTS	Ubuntu Server 22.04 LTS
CPU	Intel Xeon 12 Cores, 2.30 GHz	Intel Xeon 12 Cores, 2.30 GHz
Memória RAM	64 GB DDR4 ECC	64 GB DDR4 ECC
Armazenamento	1 TB NVMe SSD (ext4)	1 TB NVMe SSD (ext4)
Rede	10 Gbps Ethernet	10 Gbps Ethernet
SGBD	N/A	MySQL 8.0.33 / MongoDB 6.0.6

4.1.2. Tecnologias e Configuração do Ambiente

O Nó de Banco de Dados utilizou o sistema operacional Ubuntu Server 22.04 LTS em ambas as instalações, eliminando variações de desempenho associadas ao sistema operacional. Para garantir a integridade experimental, entre a execução completa dos testes de cada SGBD o servidor era totalmente formatado e reinstalado a partir de uma imagem-mestra. A execução das suítes de benchmark a partir de um estado limpo e idêntico assegura que o sistema de banco de dados seja a principal variável independente, fortalecendo a validade dos resultados e mitigando vieses ambientais [Shull et al. 2008].

Foram avaliadas as versões MySQL Community Server 8.0.33 e MongoDB Community Edition 6.0.6, ambas com configurações padrão otimizadas para o ambiente. No MySQL, adotou-se a prática recomendada de configurar o parâmetro `innodb_buffer_pool_size` para 50 GB (aproximadamente 80% da memória RAM disponível). Esta configuração garante que praticamente todo o volume de dados (estimado em 50 GB) seja mantido em memória principal, eliminando os tradicionais gargalos

de I/O em disco e permitindo avaliar o desempenho puro das estruturas de processamento (*in-memory workload*).

No nó cliente, a aplicação de benchmark foi implementada em Python 3.10. Para eliminar os gargalos estruturais do *Global Interpreter Lock* (GIL) e suportar altíssima concorrência na própria aplicação testadora, utilizou-se a biblioteca `multiprocessing` nativa do Python, interagindo através dos drivers oficiais `mysql-connector-python` e `pymongo` [MongoDB Inc. 2024, Oracle Corporation 2024, Python Software Foundation 2021].

4.2. Conjunto de Dados e Modelagem

Foi gerado um conjunto de dados sintético que simula as operações de uma plataforma de e-commerce. Este domínio foi escolhido por sua complexidade relacional inerente [Han et al. 2011].

4.2.1. Geração e Estrutura do Conjunto de Dados Relacional

O esquema relacional original, implementado no MySQL, foi projetado de forma normalizada e é composto por cinco tabelas principais: `clientes`, `produtos`, `pedidos`, `itens_pedido` (relacionamento $N : N$) e `avaliacoes`.

Para simular um ambiente de produção e refletir um contexto de Big Data, o banco de dados foi populado com os seguintes volumes, totalizando aproximadamente 50 GB de armazenamento bruto:

- 1 milhão de `clientes` e 100.000 `produtos`
- 5 milhões de `pedidos` e 20 milhões de `itens_pedido`
- 10 milhões de `avaliacoes`

Esta escala excede a capacidade de cache imediato do servidor, forçando operações em disco.

4.2.2. Estratégia de Desnormalização e Modelagem Documental

A migração para o MongoDB baseou-se na desnormalização. O objetivo foi minimizar o uso de operações de junção (`$lookup`). As seguintes regras foram aplicadas:

1. **Incorporação de Relacionamentos Delimitados:** Os registros de `itens_pedido` foram incorporados aos documentos de `pedidos`.
2. **Referenciamento de Relacionamentos Não Delimitados:** As `avaliacoes` foram mantidas em coleção separada devido ao crescimento não delimitado [Sadalage and Fowler 2012].
3. **Snapshotting para Preservação Histórica**: Dados imutáveis na transação (ex: preço na compra) foram copiados para os documentos incorporados [Kleppmann 2017].

A Tabela 2 ilustra este mapeamento estrutural.

Tabela 2. Mapeamento do Esquema Relacional (MySQL) para o Modelo Documental (MongoDB)

MySQL	MongoDB	Justificativa
`clientes`	Coleção `clientes`: {_id, nome, email, endereco, data_cadastro}	Entidade de nível superior. Mapeamento direto.
`produtos`	Coleção `produtos`: {_id, nome, descricao, preco, categoria}	Entidade independente. Mapeamento direto.
`pedidos`, `itens_pedido`	Coleção `pedidos`: {_id, cliente_id, data, status, endereco, itens:[{produto_id, nome, qtd, preco}]}	Incorporação para otimizar leituras de pedidos completos.
`avaliacoes`	Coleção `avaliacoes`: {_id, produto_id, cliente_id, nota, comentario, data}	Referenciamento para evitar crescimento excessivo dos documentos.

5. Processo de Migração de Dados (ETL)

A transferência do conjunto de dados do MySQL para o MongoDB foi realizada através de um processo de Extração, Transformação e Carga (ETL). Utilizou-se um *script* personalizado em Python para proporcionar total controle sobre a lógica de desnormalização [Python Software Foundation 2021].

O processo foi dividido nas etapas:

1. **Extração (*Extract*):** Realizada utilizando `SQLAlchemy` em lotes para evitar esgotamento da memória.
2. **Transformação (*Transform*):** Executada em memória. Os documentos da coleção `pedidos` foram construídos agregando os registros da tabela `itens_pedido`, realizando a junção na camada de aplicação.
3. **Carga (*Load*):** Os documentos foram carregados no MongoDB usando a biblioteca `pymongo` [MongoDB Inc. 2024] por meio do método otimizado `insert_many()`.

Essa troca arquitetônica ilustra que, embora bancos NoSQL simplifiquem as consultas de leitura na aplicação, frequentemente eles transferem a complexidade das junções para a etapa de preparação e ingestão dos dados.

6. Metodologia de Teste de Desempenho

Para avaliar o desempenho do MySQL e do MongoDB, foi definido um protocolo de benchmarking rigoroso inspirado no *Yahoo! Cloud Serving Benchmark (YCSB)* [Cooper et al. 2010].

6.1. Definição das Cargas de Trabalho

Em todas as cargas, a seleção de registros seguiu uma distribuição Zipfiana, na qual a probabilidade de acesso ao elemento de rank k é dada por:

$$P(k) = \frac{1/k^s}{\sum_{i=1}^{N} 1/i^s} \tag{1}$$

Onde N representa o número total de elementos (os registros armazenados no banco de dados, como os pedidos cadastrados) e $s > 0$ é o parâmetro de concentração.

Isso simula acessos não uniformes (”hotspots”). A Tabela 3 resume as três cargas (Workloads A, B e C) projetadas para o teste.

Tabela 3. Cargas de Trabalho do Benchmark

Carga	Cenário	Leitura/Escrita	Operações Principais
A	Histórico de pedidos	95% L / 5% A	MySQL: `SELECT + JOIN`; MongoDB: `find_one()`
B	Criação de pedidos	50% L / 50% E	MySQL: `INSERT` transacional; MongoDB: `insert_one()`
C	Agregação por categoria	100% L	MySQL: `GROUP BY / AVG()`; MongoDB: `$lookup + $group`

6.2. Execução dos Testes e Coleta de Métricas

O cliente de benchmark simulou 100 usuários concorrentes durante 30 minutos por teste. Cada teste teve cinco repetições (descartando-se a primeira por ser de *warm-up*). O cache do sistema foi limpo entre repetições (*cold start*).

Os KPIs coletados foram: ***Throughput* (Vazão)** mensurado em operações por segundo (ops/s), e a **Latência Média** em milissegundos (ms).

sectionResultados Preliminares e Discussão

Esta seção apresenta os resultados sintéticos preliminares das execuções do *benchmark*, consolidando as medições de vazão e tempo de resposta, descritas na Tabela 4.

Tabela 4. Resultados Preliminares de Desempenho (Throughput e Latência Média)

	MySQL 8.0		MongoDB 6.0	
Carga (Workload)	**Vazão (ops/s)**	**Latência (ms)**	**Vazão (ops/s)**	**Latência (ms)**
A (Leitura Intensiva)	24.390	4,1	90.909	1,1
B (Escrita Intensiva)	8.928	11,2	26.315	3,8
C (Analítica / Agregação)	2.816	35,5	909	110,0

Os resultados demonstram que, mesmo operando em um cenário predominantemente *in-memory* (onde a latência de disco é minimizada), para a **Carga A**, o modelo de *embedding* (desnormalização) do MongoDB forneceu um salto exponencial no *throughput* de leitura (de 24.390 para assustadores 90.909 ops/s). A recuperação do documento na memória como um objeto único revelou-se consideravelmente mais eficiente e leve para o processador do que as operações de varredura e relacionamento dinâmico necessárias pelo `JOIN` em memória do MySQL.

Na **Carga B**, referente a operações de inserção e processamento de carrinhos, o MongoDB também obteve ampla vantagem. A inserção de um único documento JSON evitou os bloqueios estruturais e os registros transacionais complexos exigidos pelo MySQL (modelo ACID), reduzindo a latência de $11,2$ ms para apenas $3,8$ ms, permitindo uma vazão superior a 26.000 ops/s.

Entretanto, a **Carga C** evidenciou que a flexibilidade NoSQL tem o seu preço no quesito analítico. O MySQL superou amplamente o MongoDB no processamento

de agrupamentos em grande escala, atingindo o triplo de vazão (2.816 ops/s contra apenas 909 ops/s). A dependência do operador `$lookup` e dos estágios de *pipeline* no MongoDB para relacionar coleções independentes acarreta um alto custo de CPU e manipulação de memória local, revalidando a eficiência e otimização madura do motor relacional para cálculos analíticos.

7. Considerações Finais

A migração entre bancos de dados relacionais e NoSQL vem ganhando destaque, impulsionada pelo crescimento de aplicações que processam grandes volumes de informação. Embora bancos relacionais, como o MySQL, ainda sejam a escolha soberana para contextos que exigem agrupamentos analíticos profundos, eles perdem agilidade estrutural para cargas que demandam leitura e escrita massiva de entidades autocontidas.

Os resultados preliminares, baseados em simulações em memória (64 GB de RAM), demonstram quantitativamente que as soluções NoSQL orientadas a documento reduzem consideravelmente a latência de I/O em CPU para resgate de dados desnormalizados. Contudo, essa transição não elimina a complexidade, apenas a movimenta: o esforço computacional abstrato do motor relacional migra para a etapa arquitetural do processo de ETL e para o controle rigoroso da modelagem na camada de aplicação.

Conclui-se que os paradigmas são estritamente complementares. Arquiteturas híbridas despontam como uma solução equilibrada e pragmática. Para os trabalhos futuros, propõe-se a aplicação da avaliação em ecossistemas reais distribuídos de microsserviços e o estudo do impacto do crescimento do banco de dados quando este ultrapassa os limites físicos da memória RAM do servidor (gerando paginação em disco).

Referências

Barros, J. R. A. et al. (2017). Análise de desempenho de banco de dados relacionais e não relacionais em dados genômicos. *Revista de Informática Teórica e Aplicada*, 24(2):11–27.

Basso, C. d. A. M. and Gomes, B. C. K. (2015). Desempenho de banco de dados não relacionais com big data. In *12th CONTECSI - International Conference on Information Systems and Technology Management*.

Codd, E. F. (1970). A relational model of data for large shared data banks. *Communications of the ACM*, 13(6):377–387.

Cooper, B. F., Silberstein, A., Tam, E., Ramakrishnan, R., and Sears, R. (2010). Benchmarking cloud serving systems with ycsb. In *Proceedings of the 1st ACM Symposium on Cloud Computing (SoCC)*, pages 143–154.

Friess, I. I. (2013). Análise de bancos de dados nosql e desenvolvimento de uma aplicação. Technical report, Universidade Federal de Santa Maria.

Garbin, T. S., Duarte, D., Schreiner, G. A., and da Silva Feitosa, S. (2024). Uma abordagem para migração de banco de dados relacional para nosql orientado a documentos. In *Escola Regional de Banco de Dados (ERBD)*, pages 21–30. SBC.

Gray, J. (1981). The transaction concept: Virtues and limitations. In *Proceedings of the 7th International Conference on Very Large Data Bases*, pages 144–154.

Han, J., Haihong, E., Le, G., and Du, J. (2011). Survey on nosql database.

Kleppmann, M. (2017). *Designing Data-Intensive Applications*. O'Reilly Media.

Melo, T. C. d. (2016). Estudo de caso de bancos nosql no contexto de big data. Technical report, Universidade de Caxias do Sul.

MongoDB Inc. (2024). *PyMongo Documentation*. Available at: https://pymongo.readthedocs.io/.

Murari, M. A. and Cunha, G. B. (2014). Desenvolvimento de um software para migração de um banco de dados relacional firebird para o não relacional mongodb. Technical report, Universidade Federal de Santa Maria.

Oracle Corporation (2024). *MySQL Connector/Python Developer Guide*. Available at: https://dev.mysql.com/doc/connector-python/en/.

Python Software Foundation (2021). *Python Language Reference, Version 3.10*. Available at: https://docs.python.org/3.10/.

Rabelo, D. F. and Cândido, M. V. I. (2017). Análise de desempenho de banco de dados nosql em um sistema que utiliza um banco de dados relacional e não relacional para armazenamento de dados. Repositório AEE.

Sadalage, P. J. and Fowler, M. (2012). *NoSQL Distilled: A Brief Guide to the Emerging World of Polyglot Persistence*. Addison-Wesley Professional, Boston, MA.

Shull, F., Singer, J., and Sjøberg, D. I. K. (2008). *Guide to Advanced Empirical Software Engineering*. Springer.

Souza, F. J. d. and Santana, P. H. A. d. (2017). Estudo de caso: AnÁlise entre banco de dados relacional e nÃo relacional. Repositório AEE.

Tomio, G. V. (2015). Utilizando a tecnologia de banco de dados nosql: um caso prático. Master's thesis, Universidade Tecnológica Federal do Paraná.