

Análise comparativa entre bancos de dados relacionais e não relacionais

Francisco Teixeira Rocha Aragão¹, Gabriel Pains de Oliveira Cardoso¹,
Iasmin Correa Araujo¹

¹Universidade Federal de Minas Gerais (UFMG)
Belo Horizonte – MG – Brasil

Abstract. *This paper aims to perform a comparative analysis of different Database Management Systems (DBMS), including both relational (SQL) and non-relational (NoSQL) databases. The objective is to illustrate the performance of each system across different tasks, highlighting their respective strengths and weaknesses. The systems analyzed in this study are PostgreSQL, MongoDB, Redis, and Cassandra.*

Resumo. *Este artigo busca realizar uma análise comparativa entre diferentes sistemas de gerência de banco de dados (SGBDs), sendo eles relacionais (SQL) e não relacionais (NoSQL). Desse modo, o objetivo é exemplificar o desempenho entre cada aplicação para diferentes tarefas, exemplificando os pontos fortes e fracos em cada abordagem. Foram analisados neste projeto o PostgreSQL, MongoDB, Redis e Cassandra.*

1. Introdução

Com o avanço da tecnologia e o aumento na geração de dados ao longo dos anos [Sharma and Navdeti 2015], a escolha do sistema de gerenciamento de banco de dados (SGBD) é um fator determinante para o desempenho e a escalabilidade das aplicações. Em um cenário com diferentes abordagens para cada situação, sistemas distribuídos, aplicações web e desenvolvimento em baixo nível, apresentam requisitos únicos que demandam soluções personalizadas e específicas. Tal fato é um dos motivadores do trabalho, que tem como objetivo entender melhor os sistemas a nossa volta e como podemos modelar cada situação da maneira mais eficiente.

Os **bancos de dados relacionais** (SQL) são baseados em um modelo tabular estruturado, no qual os dados são organizados em tabelas com linhas e colunas, e as relações entre as entidades são definidas por chaves primárias e estrangeiras. Dessa forma, conseguimos facilmente modelar relacionamentos como "ligações" entre diferentes tabelas, e escalar essas operações para contextos maiores. Exemplos clássicos incluem o *MySQL*, *PostgreSQL* e *SQLite*.

Por outro lado, os **bancos de dados não relacionais** (NoSQL) surgiram como uma resposta à necessidade de maior flexibilidade e agilidade [MongoDB Inc. 2024] em aplicações que manipulam grandes volumes de dados sem estrutura definida. Esses sistemas adotam diferentes modelos de armazenamento, como *Key-Value Stores* (ex.: *Redis*), *Document Stores* (ex.: *MongoDB*) e *Column-Family Stores* (ex.: *Cassandra*), cada um otimizado para cenários específicos de uso.

Assim, esse trabalho tem como foco realizar uma análise comparativa entre bancos de dados SQL e NoSQL, considerando diferentes situações e problemas de modo a evidenciar os compromissos associados a cada uma dessas estratégias. Serão avaliadas métricas básicas em cada um desses cenários, como tempo, memória gasta e recursos computacionais utilizados durante operações de leitura, criação, escrita e remoção de dados nos registros. O objetivo final é evidenciar como cada estilo de banco de dados atende a um propósito específico, entendendo que eles podem coexistir em um domínio complexo, de modo a obter o melhor desempenho geral no sistema.

A seguir, o artigo está organizado da seguinte forma: na Seção 2 são apresentadas as ferramentas utilizadas; na Seção 3, os trabalhos relacionados que embasam a análise; na Seção 4, a metodologia empregada e os testes realizados; na Seção 5, os resultados obtidos; e, por fim, na Seção 6, são discutidas as conclusões e as possibilidades de trabalhos futuros.

2. Ferramentas utilizadas

Os bancos de dados selecionados para este estudo foram o PostgreSQL, o Redis, o MongoDB e o Apache Cassandra. Esses sistemas foram escolhidos principalmente por sua popularidade, ampla adoção no mercado e por representarem diferentes modelos de dados e arquiteturas (relacional, chave-valor, documento e colunas largas distribuídas), priorizando também as arquiteturas NoSQL mais utilizadas no mercado. Assim, permitem uma comparação significativa entre paradigmas variados de banco de dados e seus respectivos comportamentos em cenários diversos.

PostgreSQL [PostgreSQL Global Development Group 2025] é um sistema de banco de dados objeto-relacional de código aberto, com mais de 30 anos de desenvolvimento ativo, validado por sua robustez, conformidade com SQL e por suportar cargas complexas de dados. Esse banco de dados atende principalmente aplicações que exigem integridade transacional, joins complexos, constraints, stored procedures e um modelo relacional bem definido. Ele é adequado para cenários onde a estrutura dos dados é bem conhecida de antemão e onde operações analíticas, integridade referencial e consistência forte são requisitos centrais, garantindo uma forte ACID.

Redis [Redis Ltd. 2025] é um servidor de estrutura de dados em memória, classificado como um sistema NoSQL de chave-valor, que suporta diferentes tipos de valor, como strings, hashes, listas, conjuntos ordenados, streams, entre outros. Ele se adequa a contextos de aplicações que demandam latência extremamente baixa e alta taxa de leitura/escrita: caches, sessões, filas de mensagens em tempo real ou ranking em memória. Deve ser usado onde a persistência no disco ou a modelagem relacional complexa não são os principais objetivos ou podem ser sacrificados em favor de desempenho.

MongoDB [MongoDB Inc. 2025] é um banco de dados de documentos, projetado para facilitar o desenvolvimento de aplicações e permitir escalabilidade, armazenando os registros como documentos (semelhantes a JSON/BSON) que permitem estrutura flexível e aninhamentos. Ele é utilizado em aplicações onde o esquema dos dados pode evoluir, existem muitos campos opcionais ou heterogeneidade entre registros, ou onde a modelagem tradicional de tabelas rígidas seria muito custosa. De maneira geral, esse banco de dados tem sido bastante utilizado integrado a sistemas com arquiteturas de microserviços.

Apache Cassandra [Apache Software Foundation 2025] é um banco de dados

NoSQL distribuído, baseado em um modelo de colunas largas particionado, projetado para lidar com grandes volumes de dados, replicação entre data centers e alta disponibilidade sem ponto único de falha. Ele é apropriado para cenários de ingestão massiva de dados, time series distribuídas, aplicações que exigem escalabilidade horizontal linear e tolerância a falhas, e onde consultas complexas ou joins não são o foco principal. A prioridade do banco é throughput elevado e disponibilidade ao invés de consistência imediata estrita.

3. Trabalhos relacionados

Diversos estudos ao longo da última década têm explorado comparações entre bancos de dados relacionais e não relacionais, com foco no desempenho e na adequação a diferentes cenários de uso. Entre os trabalhos, temos o de **Li e Manoharan (2013)** [Li and Manoharan 2013] que é um dos motivadores para o presente projeto. Os autores realizaram experimentos envolvendo operações fundamentais de leitura, escrita, deleção e iteração, em diferentes sistemas NoSQL (MongoDB, CouchDB, Cassandra, Hypertable, RavenDB e Couchbase) e um banco SQL (Microsoft SQL Server Express). Os resultados mostraram que, embora os bancos NoSQL apresentem vantagens em certas operações, nem todos superam o desempenho dos sistemas SQL.

Em um estudo mais recente, **Khan et al. (2023)** [Khan et al. 2023] conduziram uma análise comparativa entre *MySQL* e *MongoDB*, utilizando o tempo de carregamento, resposta e recuperação como métricas principais. Os autores concluíram que o *MySQL* apresentou desempenho superior em todas as etapas testadas, sendo mais eficiente e estável que o *MongoDB* nos cenários de operações básicas, como leitura, pesquisa e escrita. Mesmo que a comparação feita pelo trabalho seja reduzida, um dos pensamentos que podemos ter, e que motiva o trabalho atual, é que bancos não relacionais são mais dependentes do tipo de operação realizada e do contexto de sua utilização.

Em conjunto, esses trabalhos indicam que não existe uma abordagem superior para todos os cenários. O artigo de [Ahmad et al. 2022] comenta esse fato mostrando que bancos de dados relacionais mantêm desempenho consistente em operações transacionais e estruturadas, enquanto bancos NoSQL são melhores em cenários de alta escalabilidade e em que a agilidade é um fator primordial. Desse modo, o presente trabalho propõe reforçar essa discussão ao incluir uma maior base comparativa para análise, fazendo uma análise experimental envolvendo *SQLite3*, *PostgreSQL*, *Redis*, *Cassandra* e *MongoDB*.

4. Metodologia

A metodologia adotada neste trabalho baseia-se na comparação entre os diferentes bancos de dados escolhidos, a partir de problemas que exploram as características específicas de cada tecnologia. Inicialmente, foi considerada a possibilidade de utilizar um único problema para todos os bancos, a fim de padronizar a comparação. No entanto, essa abordagem poderia gerar vieses nos resultados, uma vez que um problema modelado de forma mais natural em uma tecnologia específica tenderia a favorecer seu desempenho em detrimento das demais.

Dessa forma, optou-se por definir quatro problemas distintos, cada um pensado para explorar melhor as particularidades de um tipo de banco de dados — relacional, orientado a documentos, chave-valor e de colunas largas. Cada problema, entretanto,

foi implementado em todos os bancos de dados analisados, permitindo observar tanto a adequação conceitual de cada tecnologia quanto seu desempenho nas operações CRUD (criação, leitura, atualização e exclusão). Essa abordagem possibilita uma avaliação prática e comparativa, considerando diferentes métricas, de modo a identificar os pontos fortes e limitações de cada sistema em diferentes contextos de aplicação.

Os problemas escolhidos foram:

1. Sistema para gerência de clientes, itens e pedidos (cliente realiza pedido que possui vários itens). Deve ser possível realizar consultas relacionais. Possui adequação conceitual ao **PostgreSQL**.
2. Sistema para acesso rápido de informações em um sistema de e-commerce. A partir de uma chave deve ser possível recuperar as informações associadas rapidamente. Possui adequação conceitual ao **Redis**.
3. Sistema voltado ao gerenciamento de perfis de usuários. Deve ser possível fazer a atualização parcial dos perfis de forma massiva. Possui adequação conceitual ao **MongoDB**.
4. Sistema de armazenagem de uma *grande quantidade* de métricas de sensores IoT, atualizadas frequentemente. Deve ser possível realizar inserções concorrentes e consultas rápidas. Possui adequação conceitual ao **Cassandra**.

As modelagens específicas de cada problema para cada SGBD estão definidas no repositório do trabalho¹.

4.1. Testes

A fim de testar e comparar os diferentes SGBDs, cada problema será aplicado nos quatro SGBDs simultaneamente e uma suíte de testes será executada, coletando métricas para as análises. Ainda, cada aplicação de testes será realizada com magnitude de dados diferentes, variando de milhares a milhões de registros, a fim de testar escalabilidade.

O ambiente de testes consiste em quatro imagens Docker pré-configuradas para cada SGDB de forma que estes executem suas operações como servidores locais. Dessa forma, cada problema tem sua suíte de testes unificada, automatizando a execução dos testes e padronizando a coleta de métricas.

A suíte de testes por sua vez, será desenvolvida em Python, por três motivos principais:

1. **Padronização do ambiente de execução.** Cada SGBD executa como servidor e recebe comandos pela API correspondente em Python.
2. **Padronização da coleta de métricas.** Cada SGBD tem seus próprios comandos e métodos de mensurar performance (eg. EXPLAIN ANALYZE no PostgreSQL). Logo, a fim de evitar diferenças baseadas em implementação e outros fatores, optou-se por realizar a medição de métricas via Python.
3. **Automatização e reprodutibilidade.** A automatização diminui erros em execuções manuais e garante um estado de sistema que pode ser reproduzido, ou seja, cada execução dos testes deve exibir resultados iguais dentro do esperado.

¹https://github.com/Francisco-aragao/SQL_x_NoSQL

As métricas coletadas serão: **i) tempo de execução, ii) uso de memória (RAM), iii) uso de processamento (CPU) e iv) heurísticas adicionais.** A categoria iv) será coletada quando possível, pois pode variar em disponibilidade e execução de cada SGDB, um exemplo de métrica nessa categoria é: quais SGBD paralelizam e qual o grau de paralelização.

Por fim, após a execução da suíte de testes conforme descrito acima, as métricas coletadas serão analisadas qualitativamente, comparando as métricas para um mesmo problema. Ainda, será analisada a adequação de cada SGBD a cada problema, a fim de compreender se existem SGDBs mais adequados para uma classe de modelagem de problemas e qual a vantagem oferecida nessa adequação.

5. Resultados obtidos

...

6. Conclusões e trabalhos futuros

...

7. Referências

- Ahmad, F., Malik, M. H., Faheem, M., Adnan, M., and Khan, K. (2022). Sql and nosql database software architecture performance analysis and assessments: A systematic literature review. *arXiv preprint arXiv:2209.06977*. Accessed: 26 Oct. 2025.
- Apache Software Foundation (2025). Cassandra architecture overview. Accessed: 26 Oct. 2025.
- Khan, M. Z., Zaman, F. U., Adnan, M., Imroz, A., Rauf, M. A., and Phul, Z. (2023). Comparative case study: An evaluation of performance computation between sql and nosql database. *Sindh Journal of Headways in Software Engineering*, 1(2).
- Li, Y. and Manoharan, S. (2013). A performance comparison of sql and nosql databases. In *2013 IEEE Pacific Rim Conference on Communications, Computers and Signal Processing (PACRIM)*.
- MongoDB Inc. (2024). When to use nosql: Nosql databases explained. Accessed: 26 Oct. 2025.
- MongoDB Inc. (2025). Introduction to mongodb. Accessed: 26 Oct. 2025.
- PostgreSQL Global Development Group (2025). About postgresql. Accessed: 26 Oct. 2025.
- Redis Ltd. (2025). What is redis? Accessed: 26 Oct. 2025.
- Sharma, A. and Navdeti, M. (2015). Analysis on big data over the years. *International Journal of Scientific and Research Publications*, 5(1):1–4. ISSN 2250-3153.