Relatório Final: Análise Comparativa de Bancos de Dados Relacionais e NoSQL em um Sistema de Pedidos

Autor: Cláudio Pontes
Curso: Engenharia de Dados
Data: 13 de Julho de 2025
link do github: https://github.com/cllaud99/puc_nosql_trabalho_final

1. Introdução

No cenário tecnológico atual, caracterizado pelo Big Data e por aplicações distribuídas de alta performance, a escolha do sistema de gerenciamento de banco de dados (SGBD) é uma decisão arquitetural crítica. De um lado, os bancos de dados relacionais (RDBMS), como o MySQL, consolidaram-se como a espinha dorsal de sistemas que exigem consistência e integridade transacional (ACID). Do outro, bancos de dados NoSQL, como o MongoDB, emergiram como soluções flexíveis e escaláveis, ideais para lidar com grandes volumes de dados não estruturados ou semi-estruturados.

Este trabalho tem como objetivo principal compreender na prática os princípios que diferenciam esses dois modelos. Para isso, foi implementado um cenário real de um sistema de pedidos multicanal, utilizando uma abordagem híbrida que integra MySQL e MongoDB. O estudo avalia as vantagens e desvantagens de cada modelo, realiza uma análise de desempenho em operações de carga e consulta, e conclui com uma análise sobre a adequação de cada tecnologia para diferentes tipos de dados no contexto proposto.

2. Fundamentação Teórica

2.1. MySQL

O MySQL é um sistema de gerenciamento de banco de dados relacional (RDBMS) open-source amplamente adotado, conhecido por sua robustez, confiabilidade e forte adesão ao modelo transacional ACID (Atomicidade, Consistência, Isolamento e Durabilidade). Ele organiza os dados em tabelas com esquemas estritos, garantindo a integridade e a consistência dos dados por meio de chaves primárias, estrangeiras e constraints. É a escolha padrão para aplicações que dependem de transações complexas e dados estruturados, como sistemas financeiros e de e-commerce.

2.2. MongoDB

O MongoDB é um SGBD NoSQL líder, orientado a documentos. Ele armazena dados em documentos flexíveis no formato BSON (uma representação binária de JSON), o que permite que os esquemas evoluam facilmente com as necessidades da aplicação. Sua arquitetura foi projetada para escalabilidade horizontal (sharding), oferecendo alta performance na leitura e escrita de grandes volumes de dados, sendo ideal para catálogos de produtos, sistemas de gerenciamento de conteúdo, e armazenamento de logs ou dados de sensores.

2.3. Comparativo Teórico

Característica	MySQL (Relacional)	MongoDB (NoSQL)
Modelo de Dados	Tabelas, linhas e colunas	Coleções e Documentos (JSON/BSON)
Esquema	Estrito e pré-definido (Schema-on-write)	Flexível e dinâmico (Schema-on-read)
Transações	Suporte completo ao padrão ACID	Parcial (a nível de documento, com transações multi-documento disponíveis)
Escalabilidade	Predominantemente Vertical (Scale-up)	Predominantemente Horizontal (Scale-out)
Linguagem de Consulta	SQL (Structured Query Language)	MQL (Mongo Query Language) / Aggregation Framework

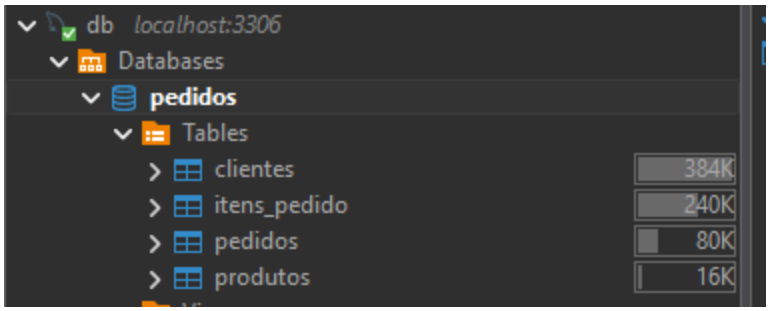
3. Metodologia

3.1. Cenário Proposto: Sistema de Pedidos Multicanal

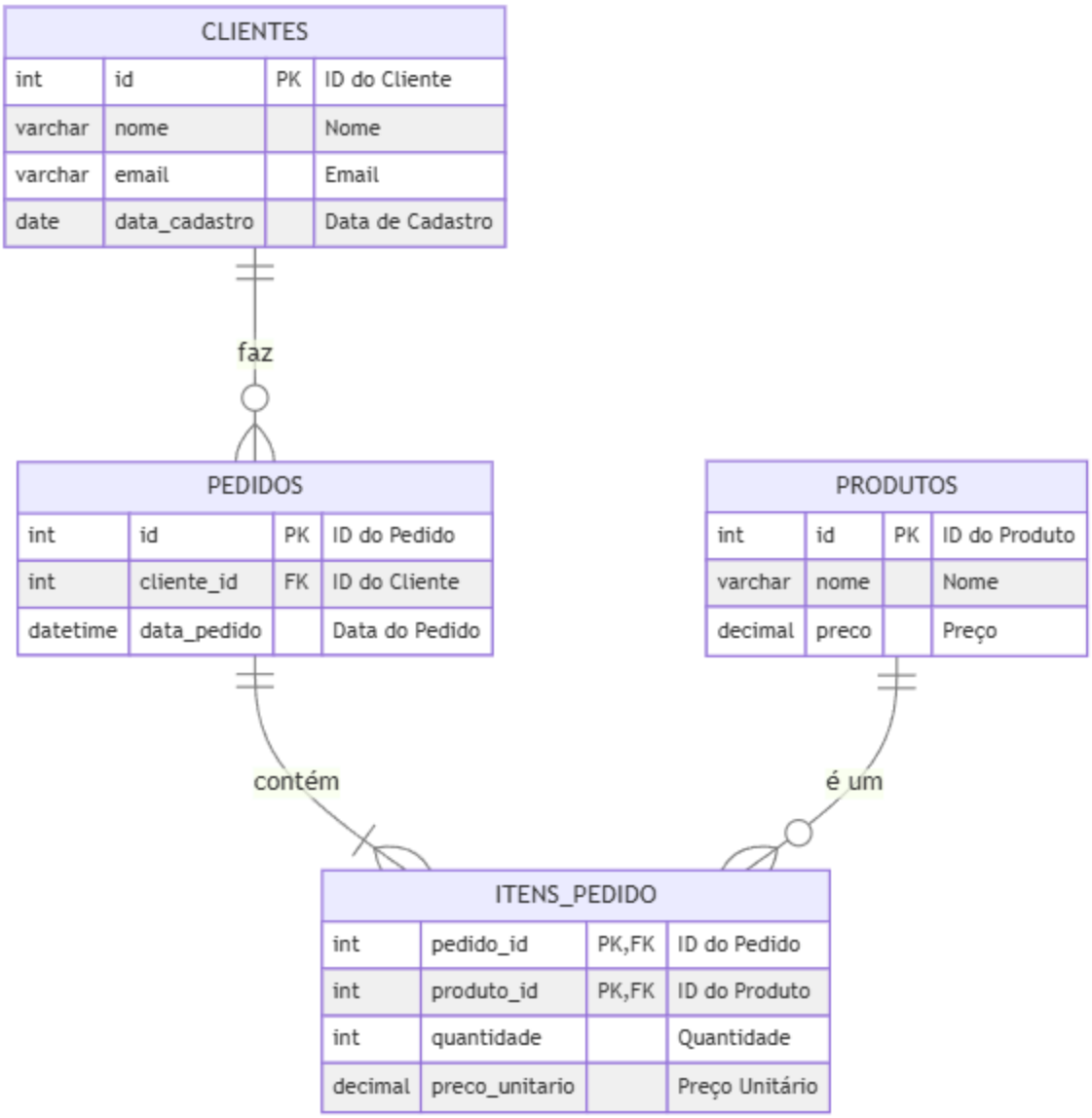
Para a análise, foi modelado um sistema de e-commerce onde os dados foram estrategicamente divididos:

- **MySQL**: Responsável por armazenar dados transacionais e estruturados, que são o núcleo do negócio: clientes, produtos, pedidos e itens_pedido. A integridade

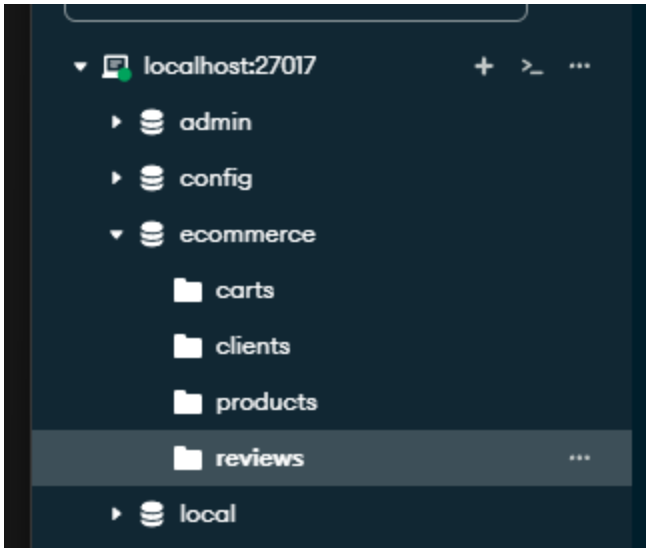
referencial e a consistência são cruciais para estas entidades.



seu schema ficou da seguinte forma:



- **MongoDB**: Utilizado para dados semi-estruturados e de alta volumetria, que são importantes mas não críticos para a transação em si: avaliacoes de produtos e carrinhos de compra em tempo real, também foi necessária a implementação de mais colections para fazer a migração dos dados conforme print abaixo



4. Consultas Comparativas

Foram realizadas três consultas de agregação para comparar o desempenho dos dois bancos de dados em cenários de análise de dados.

1. **Total de Pedidos por Cliente**: Uma consulta para contar quantos pedidos cada cliente realizou. No MySQL, isso envolve um JOIN entre clientes e pedidos com um GROUP BY. No MongoDB, exigiria uma agregação na coleção de pedidos.
2. **Total Vendido por Produto**: Calcula o valor total vendido para cada produto. No MySQL, requer um JOIN entre produtos e itens_pedido. No MongoDB, uma agregação sobre os itens dentro dos pedidos.
3. **Gasto Médio por Cliente**: Calcula o valor médio gasto por cada cliente em todos os seus pedidos. Esta é a consulta mais complexa, envolvendo múltiplos JOINs e agregações no MySQL.

5. Análise de Desempenho

Os benchmarks foram executados medindo o tempo de operações de escrita (carga

inicial) e leitura (consultas agregadas).

5.1. Desempenho de Escrita (Carga de Dados)

A tabela abaixo mostra o tempo, em segundos, para inserir 2.000 registros em cada sistema.

Operação	Banco de Dados	Tempo (segundos)
Escrita de reviews / carts	MongoDB	**~0.274** (soma)
Escrita de clientes, produtos, pedidos, itens_pedido	MySQL	**~1.200** (soma)

Análise:
O MongoDB demonstrou uma performance de escrita significativamente superior. Isso ocorre porque, ao inserir um documento (como uma avaliação ou um carrinho), a operação é atômica e autocontida. O MongoDB não precisa verificar chaves estrangeiras, atualizar múltiplos índices complexos ou garantir a mesma rigidez transacional do MySQL. O MySQL, por outro lado, incorre em uma sobrecarga (overhead) a cada INSERT para garantir a conformidade com o ACID, validar constraints e atualizar os índices que permitem a performance em leituras complexas.

5.2. Desempenho de Leitura (Consultas Agregadas)

A tabela abaixo pivota os resultados para uma comparação direta, destacando a diferença de performance.

Consulta	Tempo MySQL (s)	Tempo MongoDB (s)	Performance (MySQL mais rápido)
total_pedidos_por_cliente	0.057	2.386	**~41.9x**

total_vendido_por_produto	0.006	0.020	**~3.3x**
avg_gasto_por_cliente	0.036	2.257	**~62.7x**

Análise:
Os resultados aqui são drasticamente diferentes e revelam a principal força do modelo relacional. O MySQL foi ordens de magnitude mais rápido em todas as consultas agregadas. As razões para essa disparidade são:

- **Otimizador de Consultas do SQL**: Os SGBDs relacionais possuem otimizadores de consulta extremamente maduros que analisam a query, avaliam os índices disponíveis e criam um plano de execução altamente eficiente para JOINs e agregações.
- **Indexação para Relações**: Os índices no MySQL são projetados para acelerar as junções entre tabelas.
- **Aggregation Framework do MongoDB**: Embora poderoso, o framework de agregação do MongoDB, especialmente com o uso do operador $lookup (que simula um JOIN), não é tão otimizado para relações complexas quanto um SGBD relacional nativo. As operações de desnormalização e junção em tempo de execução são computacionalmente caras, como evidenciado pelos tempos superiores a 2 segundos nas consultas mais complexas.

6. Análise de Consistência: Eventual vs. Transacional

- **MySQL (Consistência Transacional)**: No nosso cenário, o MySQL garante que uma operação de pedido (que envolve inserir em pedidos e itens_pedido, e talvez atualizar um estoque em produtos) seja **atômica**. Ou todas as etapas são concluídas com sucesso, ou nenhuma é. Após a confirmação (COMMIT), os dados estão imediatamente consistentes e visíveis para todas as outras transações. Isso é fundamental para a lógica de negócio principal.

- **MongoDB (Consistência Eventual)**: Embora em uma instância única o MongoDB se comporte de forma fortemente consistente, seu modelo é projetado para sistemas distribuídos onde a consistência eventual é uma realidade. No nosso caso, isso se aplica à natureza dos dados: se uma avaliacao de produto demorar alguns milissegundos para ser replicada e se tornar visível para todos os usuários, o impacto no negócio é mínimo. A prioridade para esses dados é a alta disponibilidade e a performance de escrita, e não a consistência imediata.

7. Ferramentas e Tecnologias Utilizadas

- **Bancos de Dados**: MySQL 8.0, MongoDB Community Server
- **Linguagem**: Python 3.11
- **Bibliotecas Python**:
  - pymongo: Driver oficial para interação com o MongoDB.
  - sqlalchemy e pymysql: Para conexão e ORM com o MySQL.
  - pandas: Para manipulação e transformação de dados em memória.
  - faker: Para geração de dados de teste.
  - loguru: Para configuração de logs.
- **Containerização**: Docker, Docker Compose
- **IDE/Ferramentas de BD**: Visual Studio Code, DBeaver

8. Conclusão

Este trabalho demonstrou empiricamente a validade de uma arquitetura de dados híbrida. A integração entre MySQL e MongoDB foi realizada com sucesso, permitindo uma análise clara das forças e fraquezas de cada sistema.

Conclui-se que:

1. O **MySQL** é a ferramenta indiscutivelmente superior para o núcleo transacional de um sistema. Sua performance em consultas agregadas que envolvem a junção de dados estruturados é excepcional, e sua garantia de consistência ACID é indispensável para a integridade dos dados de negócio como pedidos e clientes.
2. O **MongoDB** brilha em cenários que exigem flexibilidade de esquema e alta performance de escrita para dados semi-estruturados ou independentes. Para casos de uso como carrinhos de compra, logs de navegação ou avaliações de produtos, sua arquitetura orientada a documentos e escalável é mais adequada e eficiente.