15° Congresso de Inovação, Ciência e Tecnologia do IFSP - 2024

Estudo comparativo entre bancos de dados

GUILHERME Q. P. DE A. CRUZ[1], CLAUDIO H. YAMAMOTO[2]

[1] Graduando em Ciência da Computação, IFSP, Câmpus Salto, guilherme.quirino@aluno.ifsp.edu.br
[2] Doutorado em Ciências no programa Ciências de Computação e Matemática Computacional pela USP, professor eletivo da área de informática – Programação e Banco de Dados – do IFSP Câmpus Salto, haruo@ifsp.edu.br

Área de conhecimento (Tabela CNPq): 1.03.03.03-0 Banco de Dados

RESUMO: Considerando a importância do tratamento correto das informações em sistemas digitais, a escolha de um sistema de gerenciamento de banco de dados apropriado se torna essencial para a qualidade do software a ser desenvolvido. A grande quantidade de opções disponíveis atualmente, de diferentes tipos e arquiteturas, dificulta essa escolha. Com o objetivo de auxiliar na escolha adequada de um SGBD, este trabalho se propõe a conduzir um estudo comparativo entre SGBDs selecionados, com a realização de testes de desempenho seguindo protocolos padronizados a fim de observar as características e o desempenho de acordo com os critérios estabelecidos.

PALAVRAS-CHAVE: *benchmarking*; banco de dados.

Comparative study between databases

ABSTRACT: Considering the importance of the correct treatment of information in digital systems, the apropriate database management system choice becomes essential to the quality of the software being developed. The large amount of options currently available, of different types and architectures, makes this choice harder. Aiming to help choosing a proper DBMS, this paper proposes a comparative study between the selected DBMS, executing benchmarks according to standardized protocols to observe the characteristics and performance according to the established criteria.

KEYWORDS: benchmarking; databases.

INTRODUÇÃO

Uma grande parte da responsabilidade pelo desempenho de um sistema recai sobre o SGBD (Sistema de Gerenciamento de Banco de Dados), responsável pelo tratamento e armazenamento das informações (Kausar *et al.*, 2022). As múltiplas opções disponíveis atendem às mais diversas aplicações, com SGBDs de vários tipos e arquiteturas.

É possível dividir os SGBDs em dois grandes grupos: os que utilizam *SQL* (*Structured Query Language*) e os chamados *NoSQL* (*Not only SQL*). De acordo com Kausar *et al.* (2022), SGBDs que seguem o paradigma *NoSQL* surgem da necessidade de manipular grandes quantidades de dados e servir sistemas dinâmicos com uma grande quantidade de usuários, e não necessariamente para substituir os SGBDs relacionais, normalmente associados ao uso de *SQL* como linguagem.

Como evidenciam os estudos de Khan *et al.* (2023), há motivos que levam |à escolha de um SGBD, como o formato e tamanho dos dados a serem armazenados, a estrutura física disponível e a

velocidade das operações, no caso de sistemas críticos. Em vários trabalhos, o foco foi realizar algum tipo de comparação entre diferentes SGBDs por meio de *benchmarks*, já existentes ou elaborados pelos autores, evidenciando que há um interesse em determinar as vantagens e desvantagens de cada SGBD em diversos cenários. Alguns exemplos são os estudos de Kausar *et al.* (2022), focados na análise de algumas opções de SGBDs *NoSQL*, e o trabalho de Makris *et al.* (2021), mais específico da área de dados espaçotemporais em grande escala. Raasveldt *et al.* (2018) e Taipalus (2023) apontaram diversos problemas fundamentais nesse tipo de artigo, e entre eles, a falta de reprodutibilidade e detalhes importantes de configuração.

Um *benchmark* bastante conhecido e utilizado é o *TPC-C*, usado para estressar sistemas com atividades complexas de leitura e atualização, visando simular um ambiente real e testar a capacidade de resposta do sistema a diversos usuários simultâneos (*Transaction Processing Performance Council*, 2010). Apesar de apresentar um sistema que foi projetado com o modelo relacional em mente, é possível adaptá-lo para SGBDs de outros modelos. Em resumo, o sistema proposto simula uma série de armazéns com um estoque de itens, que são vendidos a clientes em diferentes setores através de pedidos.

Considerando o contexto apresentado, este trabalho se propõe a realizar um estudo de desempenho de três SGBDs de diferentes modelos de dados.

MATERIAL E MÉTODOS

O desenvolvimento dos códigos e os testes serão realizados em um computador pessoal.

Os softwares a serem utilizados nos testes são:

• SGBDs: PostgreSQL (*The PostgreSQL Global Development Group*, 2024), MongoDB (MongoDB INC., 2024) e Redis (Redis Ltd., 2024), escolhidos a fim de abranger os principais modelos de dados existentes de acordo com o ranking do site DB-Engines (*Red Gate Software*, 2024).

• Node.js: *APIs (Application Programming Interface)* para conexão com os bancos de dados.

• Python: chamadas às *APIs*, medições e geração de gráficos.

• Docker: conteinerização das *APIs* e dos SGBDs.

O desenvolvimento das *APIs* seguirá o padrão *REST* e terá como foco a realização de simples validações, quando necessário, conexão com os bancos de dados e realização das operações que irão compor os testes.

Com o *backend* funcional, serão gerados dados fictícios para compor as massas de testes. Apesar dos diferentes modelos de dados, o intuito é que os SGBDs armazenem as mesmas informações.

Os produtos dos testes serão gráficos que medirão a performance dos bancos de dados, de modo a permitir uma avaliação objetiva, com os critérios de comparação sendo o tempo de resposta e o espaço em disco utilizado, utilizando bases de dados de escalas diferentes.

Há dois momentos principais em cada teste: o carregamento do banco com as informações iniciais e a realização de uma série de transações, sendo ambos uma versão simplificada do proposto pelo *TPC-C*. Dados com relação aos armazéns, setores, clientes, itens e seus respectivos estoques fazem parte dos dados iniciais, necessários à realização das transações, que estão descritas na Tabela 1.

TABELA 1. Detalhes das transações

Transação	Descrição	Tipos de operações envolvidas
Pedido	Um cliente faz um pedido, composto por uma série de itens e suas respectivas quantidades. São lidas as taxas de armazém e setor, que são aplicadas a um total que será subtraído do saldo do cliente e adicionado ao saldo do setor	Leitura, escrita e atualização

Transação	Descrição	Tipos de operações envolvidas
Pagamento	Um cliente é encontrado pelo CNPJ, que atua como uma chave não primária, e o valor informado é acrescentado ao seu saldo	Leitura, escrita e atualização
Status do pedido	Dado o CNPJ de um cliente, o estado do seu pedido mais recente é retornado, que pode ser entregue ou não entregue	Leitura
Entrega	A partir do id de um pedido, são verificadas as quantidades em estoque de cada item necessário; se há estoque o suficiente, o estado do pedido é alterado para entregue	Leitura e atualização
Nível do estoque	Para um determinado setor, são verificadas e retornadas as quantidades necessárias para a realização da entrega de todos os pedidos pendentes	Leitura

RESULTADOS E DISCUSSÃO

O presente trabalho considerou as recomendações de Raasveldt *et al.* (2018) e Taipalus (2024) e seguiu um desenvolvimento metódico desde o princípio, para criar um ambiente de testes altamente reprodutível, além de tornar acessíveis todos os produtos, como códigos, arquivos de configuração e dados de teste.

A aplicação proposta é um subconjunto do que é utilizado no *TPC-C*, não sendo nem tão complexa e nem tão simplificada, com detalhes o suficiente para estressar os sistemas e permitir comparações mais semelhantes ao que se veria em um cenário real.

Com a facilidade de reprodução em mente, o Docker foi escolhido para a execução de contêineres, que podem ser considerados processos isolados do restante do sistema, tanto para cada SGBD como para cada *API* responsável pela conexão com o banco e retorno dos dados, seguindo um padrão e executando através do Node.js, “um ambiente de execução de JavaScript disponível para várias plataformas […]” (*OpenJS Foundation*, 2024, tradução própria).

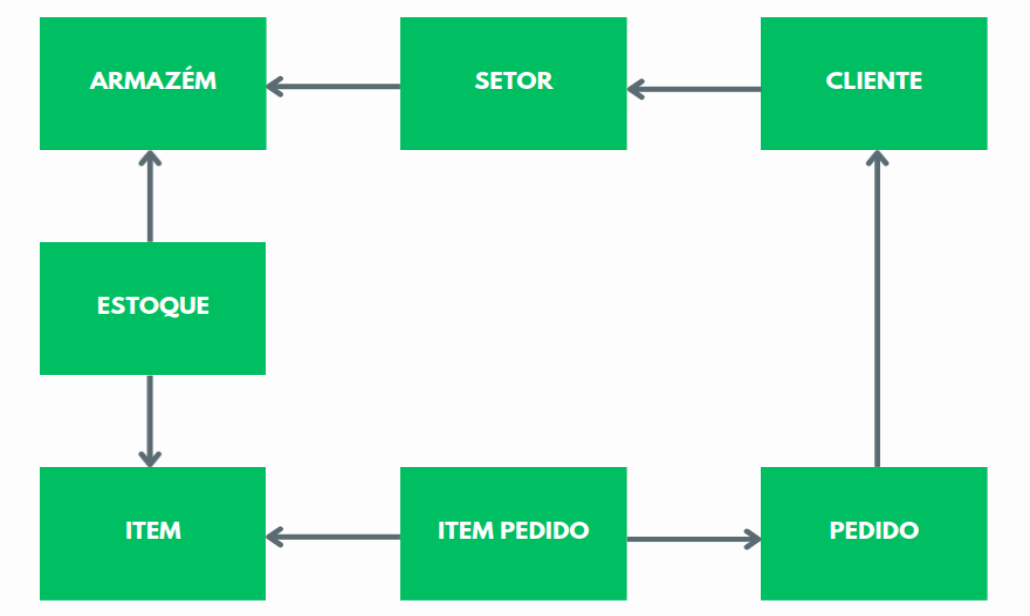

FIGURA 1. Esquema de tabelas

Na Figura 1, que representa o esquema desenvolvido com base no proposto pelo *TPC-C*, a direção das setas indica que há um relacionamento entre as tabelas por meio de uma chave estrangeira ou equivalente. Além disso, o sistema foi projetado com base no modelo relacional, mas ajustes podem ser feitos para tirar proveito das vantagens que cada modelo oferece, possibilitando uma comparação mais justa.

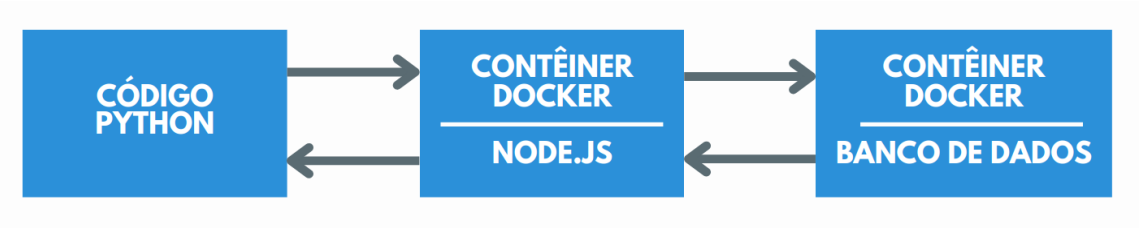

FIGURA 2. Modelo da solução

Na Figura 2, as setas indicam o fluxo dos dados, que são gerados do lado esquerdo. Foram desenvolvidos códigos em Python para a geração de uma massa de dados, envio às *APIs* e medição dos tempos de resposta, com a possibilidade de alteração do volume da carga de testes, que se baseia na quantidade de armazéns e de itens. Seguiu-se a execução dos testes e coleta dos dados, finalizando com a análise e comparação do desempenho dos bancos de dados, individualmente e entre si.

Pensando nas particularidades dos SGBDs, uma espécie de *framework* comum foi criado e serviu como base para o desenvolvimento das APIs. Ele define as rotas básicas que serão utilizadas, para orientar o desenvolvimento e padronizar o código de execução dos testes.

CONCLUSÕES

Uma vez que uma das principais utilidades da comparação de diferentes ferramentas seja o levantamento de informações que possam melhor embasar a escolha de uma delas, a aplicação de testes seguindo um formato rigoroso, transparente e possível de ser verificado é uma maneira eficaz de realizar uma coleta de dados relevantes. Tendo em vista a preocupação com a possibilidade de reprodução dos testes realizados, aspecto considerado essencial em pesquisas do tipo, pode-se dizer que os métodos adotados contribuem de forma positiva para esse objetivo, mesmo considerando que o trabalho ainda esteja em andamento.

Os resultados obtidos e a análise e comparação do desempenho dos SGBDs selecionados podem servir de base para a escolha entre as opções disponíveis, a nível de paradigma ou mais específico.

CONTRIBUIÇÕES DOS AUTORES

Todos os autores contribuíram com o desenvolvimento e revisão do trabalho e aprovaram a versão submetida.

AGRADECIMENTOS

Aos que auxiliaram no desenvolvimento do trabalho, pela ajuda e suporte, e aos familiares e amigos, pelo apoio constante.

REFERÊNCIAS

KAUSAR, Mohammad Abu; NASAR, Mohammad; SOOSAIMANICKAM, Arockiasamy. **A Study of Performance and Comparison of NoSQL Databases: MongoDB, Cassandra, and Redis Using YCSB**. Indian Journal of Science and Technology 15(31): 1532-1540. 2022. DOI: https://doi.org/10.17485/IJST/v15i31.1352. Disponível em: https://indjst.org/articles/a-study-of-performance-and-comparison-of-nosql-databases-mongodb-cassandra-and-redis-using-ycsb. Acesso em: 27 out. 2024.

KHAN, W.; KUMAR, T.; ZHANG, C.; RAJ, K.; ROY, A.M.; LUO, B. **SQL and NoSQL Database Software Architecture Performance Analysis and Assessments—A Systematic Literature Review**. Big Data Cogn. Comput. 2023, 7, 97. DOI: https://doi.org/10.3390/bdcc7020097. Disponível em: https://www.mdpi.com/2504-2289/7/2/97. Acesso em: 27 out. 2024.

MAKRIS, Antonios; TSERPES, Konstantinos; SPILIOPOULOS, Giannis; ZISSIS, Dimitrios; ANAGNOSTOPOULOS, Dimosthenis. **MongoDB Vs PostgreSQL: A comparative study on performance aspects**. Geoinformatica 25, 243–268. 2021. DOI: https://doi.org/10.1007/s10707-020-00407-w. Disponível em: https://link.springer.com/article/10.1007/s10707-020-00407-w. Acesso em: 28 out. 2024.

MONGODB INC. **O que é o MongoDB? - Manual do MongoDB v7.0**. 2024. Disponível em: https://www.mongodb.com/pt-br/docs/manual/. Acesso em: 28 out. 2024.

OPENJS FOUNDATION. **Node.js - Run JavaScript Everywhere**. 2024. Disponível em: https://nodejs.org/en. Acesso em: 28 out. 2024.

RAASVELDT, Mark; HOLANDA, Pedro; GUBNER, Tim; MÜHLEISEN, Hannes. **Fair Benchmarking Considered Difficult: Common Pitfalls In Database Performance Testing**. In: Workshop on Testing Database Systems (DBTest'18), 2018, Texas, EUA. Anais eletrônicos [...] Nova York, EUA, Association for Computing Machinery, 2018. DOI: https://doi.org/10.1145/3209950.3209955. Disponível em: https://hannes.muehleisen.org/publications/DBTEST2018-performance-testing.pdf. Acesso em: 27 out. 2024.

RED GATE SOFTWARE. **DB-Engines Ranking per database model category**. 2024. Disponível em: https://db-engines.com/en/ranking_categories. Acesso em: 27 out. 2024.

REDIS LTD. **Docs**. Disponível em: https://redis.io/docs/latest/. Acesso em: 27 out. 2024.

TAIPALUS, Toni. **Database management system performance comparisons: A systematic literature review**. The Journal of Systems & Software, Finlândia, v. 208, 2023. DOI: https://doi.org/10.1016/j.jss.2023.111872. Disponível em: https://www.sciencedirect.com/science/article/pii/S0164121223002674. Acesso em: 27 out. 2024.

THE POSTGRESQL GLOBAL DEVELOPMENT GROUP. **PostgreSQL: Documentation: 17: PostgreSQL 17.0 Documentation**. 2024. Disponível em: https://www.postgresql.org/docs/17/index.html. Acesso em: 27 out. 2024.

TRANSACTION PROCESSING PERFORMANCE COUNCIL. **TPC BENCHMARK C. Revision 5.11**. 2010. Disponível em: https://www.tpc.org/TPC_Documents_Current_Versions/pdf/tpc-c_v5.11.0.pdf. Acesso em 27 out. 2024.