EXPLORANDO AS DIFERENÇAS E SIMILARIDADES ENTRE BANCOS DE DADOS RELACIONAIS, NOSQL E NEWSQL

Victor Hugo Ramos[1], Ana Cláudia Garcia Barbosa[2]

Resumo: O crescente volume de dados e a necessidade de desempenho e escalabilidade têm impulsionado o desenvolvimento de novos paradigmas de sistemas de gerenciamento de banco de dados. Este trabalho tem como objetivo comparar os modelos Relacional, NoSQL e NewSQL, analisando suas diferenças e similaridades quanto à consistência, desempenho e escalabilidade. A metodologia envolveu experimentos práticos com PostgreSQL, MongoDB e SingleStore, em cenários de inserção massiva, a fim de observar o comportamento de cada sistema sob carga crescente. Os resultados apontaram padrões distintos de desempenho e eficiência, refletindo as particularidades estruturais e arquiteturais de cada modelo. Constatou-se que cada abordagem apresenta vantagens específicas conforme o contexto de uso, sendo possível identificar diferentes graus de equilíbrio entre consistência, flexibilidade e velocidade. Assim, o estudo contribui para a compreensão crítica sobre a aplicação adequada de cada paradigma frente às demandas contemporâneas de gestão de dados.

Palavras-chave: sistemas de gerenciamento de banco de dados; bancos de dados relacionais; bancos NoSQL; bancos NewSQL; escalabilidade; desempenho.

[1]Curso de Ciência da Computação, Universidade do Extremo Sul Catarinense (Unesc), vr3438@unesc.net

[2]Orientadora. Curso de Ciência da Computação, Universidade do Extremo Sul Catarinense (Unesc), agb@unesc.net

ABSTRACT: The increasing volume of data and the demand for performance and scalability have driven the development of new database management paradigms. This study aims to compare the Relational, NoSQL, and NewSQL models, analyzing their differences and similarities regarding consistency, performance, and scalability. The methodology involved practical experiments using PostgreSQL, MongoDB, and SingleStore, in large-scale insertion scenarios to observe each system's behavior under growing loads. The results indicated distinct performance and efficiency patterns, reflecting the structural and architectural specificities of each model. It was found that each approach presents advantages depending on its context of use, with different balances between consistency, flexibility, and speed. Thus, this study contributes to a critical understanding of the proper application of each paradigm in light of contemporary data management demands.

Keywords: database management systems; relational databases; NoSQL databases; NewSQL databases; scalability; performance.

1 INTRODUÇÃO

Os sistemas de gerenciamento de banco de dados (SGBDs) desempenham papel central no desenvolvimento de aplicações modernas, sendo responsáveis pelo armazenamento, organização e recuperação eficiente das informações. Desde a proposta do modelo relacional por Edgar F. Codd na década de 1970, os SGBDs evoluem continuamente para atender às demandas crescentes de desempenho, escalabilidade e integridade dos dados. Conforme Codd (1970), o modelo relacional estabeleceu uma base teórica sólida ao utilizar a álgebra relacional e assegurar o controle transacional por meio das propriedades ACID — atomicidade, consistência, isolamento e durabilidade. Esses mecanismos estruturais garantem que operações de escrita sejam executadas de forma confiável, evitando estados parciais e mantendo a integridade lógica do banco, o que explica sua adoção predominante em aplicações corporativas e financeiras. A padronização da linguagem SQL, destacada por Chamberlin (2012), reforçou ainda mais essa consolidação.

Os bancos relacionais também se caracterizam pelo uso de esquemas rigidamente definidos e técnicas de normalização que minimizam redundâncias e preservam relações entre entidades, conforme apontado por Berg, Seymour e Goel (2012). Entretanto, pesquisas recentes mostram que a arquitetura relacional tradicional, concebida para ambientes centrali-

zados, apresenta limitações quando aplicada a cenários modernos que demandam escalabilidade horizontal e flexibilidade estrutural. Estudos como os de Rasheed, Qutqut e Almasalha (2019) evidenciam que bancos relacionais enfrentam dificuldades ao lidar com grandes volumes de dados semi-estruturados e com a execução eficiente em ambientes distribuídos. Essa limitação se tornou mais evidente com a popularização da computação em nuvem, Big Data e IoT, conforme discutido em Wei et al. (2021) e reforçado por Dasan e Joshi (2025), que destaca a crescente pressão por soluções capazes de combinar desempenho e elasticidade.

Diante dessa necessidade, surgiram os bancos de dados NoSQL, projetados para oferecer maior flexibilidade na modelagem de dados e melhor desempenho em cenários distribuídos. Conforme Sadalage e Fowler (2013), esses sistemas adotam modelos variados — chave-valor, documentos, colunar ou grafos — permitindo armazenar informações sem rigidez de esquema. Sua arquitetura distribuída possibilita replicação, particionamento de dados (sharding) e balanceamento de carga, segundo Rasheed, Qutqut e Almasalha (2019). No entanto, essa flexibilidade implica compromissos importantes: modelos NoSQL geralmente não implementam garantias ACID completas, operando com consistência eventual. O Teorema CAP, proposto por Brewer (2000) e formalizado por Gilbert e Lynch (2002), explica essas escolhas de projeto ao demonstrar que sistemas distribuídos não podem oferecer simultaneamente consistência forte, disponibilidade plena e tolerância à partição.

Como resposta às limitações dos modelos tradicionais, surgiu a categoria dos bancos de dados NewSQL, concebida para unir a consistência dos bancos relacionais com a escalabilidade dos sistemas NoSQL. Segundo Pavlo e Aslett (2016), os sistemas NewSQL foram projetados para suportar cargas transacionais intensas (OLTP) em arquiteturas distribuídas sem abrir mão da linguagem SQL e das propriedades ACID. Entre suas características destacam-se a replicação automática, isolamento forte, alta disponibilidade e controle de concorrência multiversão (MVCC), além do uso de algoritmos distribuídos como Paxos e Raft (Bernstein; Goodman, 1983; Wei et al., 2021). Exemplos representativos incluem o Google Cloud Spanner, que utiliza sincronização via TrueTime para garantir consistência global (Cloud, 2023), o CockroachDB, que replica dados entre múltiplos nós utilizando o protocolo Raft (Labs, 2022), e o VoltDB, projetado para execução de transações em memória com baixa latência (Data, 2022). O MariaDB Xpand, por sua vez, oferece escalabilidade horizontal mantendo

compatibilidade com o ecossistema MySQL (MariaDB Corporation, 2023). Estudos recentes, como Zhang, Megargel e Jiang (2025), reforçam que sistemas NewSQL têm avançado significativamente em desempenho e escalabilidade em arquiteturas distribuídas, especialmente quando submetidos a cenários de alta taxa de transações. O panorama também é discutido por Knob et al. (2019), que examinam diferenças conceituais e práticas entre diversas soluções NewSQL e apontam seu potencial para workloads OLTP em larga escala.

Diante desse cenário de evolução tecnológica, o objetivo geral deste trabalho foi realizar uma análise comparativa entre os modelos SQL, NoSQL e NewSQL, considerando suas características fundamentais, bem como seu comportamento prático em operações de inserção em larga escala. Para atingir esse objetivo, buscou-se compreender as principais estruturas de cada modelo, comparar seu desempenho em cenários experimentais, identificar vantagens e limitações e refletir sobre a adequação de cada abordagem a diferentes tipos de aplicação.

Este trabalho está organizado da seguinte forma: na Introdução, apresentam-se o contexto, a fundamentação teórica e os objetivos do estudo. A seção seguinte apresenta os Materiais e Métodos, descrevendo os sistemas selecionados e os procedimentos experimentais adotados. Na seção de Discussão e Resultados, são apresentados os dados obtidos e a análise crítica comparativa. Por fim, a Conclusão sintetiza os principais achados e oferece sugestões para pesquisas futuras.

2 MATERIAIS E MÉTODOS

A etapa metodológica deste trabalho teve como objetivo validar os conceitos apresentados na fundamentação teórica por meio de experimentos comparativos envolvendo três modelos distintos de sistemas gerenciadores de banco de dados (SGBDs): SQL, NoSQL e NewSQL. Os testes buscaram observar o comportamento dessas tecnologias em operações de inserção em larga escala, analisando desempenho, estabilidade e comportamento sob aumento de carga.

Os experimentos foram conduzidos utilizando três SGBDs amplamente representativos de suas respectivas classes: PostgreSQL (SQL), MongoDB (NoSQL) e SingleStore (NewSQL). Os testes com PostgreSQL e MongoDB foram executados em um ambiente local, utilizando um computador com processador AMD Ryzen 5 5600 de 6 núcleos, 16 GB de memória RAM e sistema operacional Windows 11 de 64 bits. Já o SingleStore

foi executado na plataforma SingleStore Cloud, que fornece infraestrutura distribuída gerenciada. Essa diferença de ambiente foi considerada na interpretação dos resultados, evitando comparações diretas de desempenho entre ambientes diferentes.

Em todos os sistemas, os dados inseridos seguiram a mesma estrutura lógica, simulando entidades como clientes, produtos e transações. O volume total processado foi de 2 milhões de registros, independentemente do modelo utilizado. Após a conclusão das inserções, foi verificado se o total de inserções planejadas havia sido armazenado corretamente, garantindo que nenhum sistema descartasse registros ou apresentasse inconsistências.

2.1 MODELO RELACIONAL

O modelo relacional de bancos de dados, proposto por Edgar F. Codd na década de 1970, estabeleceu uma forma estruturada para organização e manipulação de dados por meio de relações representadas em tabelas, compostas por linhas e colunas. Essa abordagem, fundamentada na álgebra relacional, permitiu a criação de modelos lógicos independentes do armazenamento físico, facilitando o acesso e garantindo consistência nas operações sobre os dados (Codd, 1970; Chamberlin, 2012).

Neste estudo, o PostgreSQL foi utilizado como representante do paradigma relacional. A instalação utilizada corresponde à versão 17, administrada por meio da ferramenta pgAdmin 4, que possibilitou a criação das tabelas, execução dos comandos SQL e acompanhamento dos processos de inserção. O ambiente foi mantido com as configurações padrão, sem ajustes de otimização, assegurando condições homogêneas de comparação com os demais sistemas avaliados.

Os testes contaram com scripts SQL responsáveis por inserir diferentes volumes de registros — variando até 2 milhões de linhas — com o objetivo de medir o tempo total das operações e a estabilidade sob alta carga. O tempo de execução das inserções foi medido utilizando funções internas do PostgreSQL, como `clock_timestamp()` combinada com `EXTRACT(EPOCH FROM ...)`, que registraram o intervalo total entre o início e o fim de cada operação.

Essa configuração permitiu avaliar o comportamento do PostgreSQL em operações de escrita de alta intensidade, observando principalmente o tempo necessário para concluir grandes volumes de inserções. Embora o sistema ofereça recursos avançados de controle de concorrência

e integridade, tais aspectos não foram avaliadas, que se concentrou exclusivamente na execução e medição das inserções em ambiente local.

2.2 MODELO NOSQL

Com o crescimento exponencial de dados e a necessidade de sistemas mais escaláveis e flexíveis, surgiram os bancos de dados NoSQL (Not Only SQL) como alternativa ao modelo relacional tradicional, priorizando alta disponibilidade, escalabilidade horizontal e desempenho em grandes volumes de dados (Sadalage; Fowler, 2013). Diferentemente dos SGBDs baseados em tabelas e esquemas rígidos, o NoSQL opera com estruturas mais dinâmicas e modelos variados de armazenamento — como chave-valor, orientado a documentos, colunar e de grafos.

Neste estudo, o MongoDB foi adotado como representante do modelo NoSQL, utilizando o formato orientado a documentos. O sistema foi instalado em ambiente local, sob as mesmas condições de hardware aplicadas aos demais bancos avaliados. O gerenciamento das coleções, a execução das operações e o monitoramento dos tempos de processamento foram realizados por meio da ferramenta MongoDB Compass, que fornece o tempo de importação diretamente na interface.

Foram avaliados dois cenários de inserção, ambos totalizando 2 milhões de registros, variando apenas o tamanho individual dos arquivos JSON: 20 arquivos com 100.000 documentos cada e 10 arquivos com 200.000 documentos cada.

As inserções foram realizadas exclusivamente por meio da importação desses arquivos, previamente gerados com a mesma estrutura de dados utilizada nos demais modelos.

O objetivo da comparação entre os dois cenários foi analisar se o aumento do volume por operação influenciaria o desempenho do MongoDB, considerando que operações em lotes maiores tendem a reduzir a sobrecarga por inserção e possibilitar maior aproveitamento do paralelismo interno do sistema.

A métrica utilizada foi o tempo total de importação registrado automaticamente pelo MongoDB Compass para cada arquivo, permitindo a mensuração precisa do tempo gasto em cada cenário sem necessidade de scripts adicionais. É importante destacar que os testes não incluíram a análise de aspectos como replicação, particionamento ou consistência eventual, que embora relevantes na literatura NoSQL, não faziam parte do escopo deste estudo.

2.3 NEWSQL

O SingleStore foi utilizado como representante do modelo NewSQL neste estudo. O sistema foi executado por meio da plataforma SingleStore Cloud, acessada através de seu console SQL online, utilizando configurações padrão de desempenho. A escolha dessa tecnologia baseou-se em sua arquitetura moderna, que combina a linguagem SQL tradicional com mecanismos distribuídos projetados para cargas OLTP em larga escala.

De acordo com a documentação oficial da plataforma, o SingleStore adota um modelo de execução distribuída com processamento em memória (in-memory rowstore) e pipelines paralelos de ingestão, que permitem acelerar operações de escrita em lote e reduzir latências em cargas intensivas (SingleStore, 2023). Além disso, o sistema emprega dois modos complementares de armazenamento — rowstore, otimizado para transações, e columnstore, voltado para consultas analíticas — possibilitando uso em cenários HTAP (Hybrid Transactional/Analytical Processing) sem a necessidade de replicação adicional de dados.

Para a execução dos testes, foram utilizados scripts SQL automatizados responsáveis por gerar e enviar comandos INSERT de forma sequencial até totalizar 2 milhões de registros. O tempo de execução foi medido por meio das funções internas NOW() e TIMESTAMPDIFF(), permitindo registrar o intervalo entre o início e o término das operações. Ao final de cada execução, verificou-se também se o número total de inserções correspondia ao esperado, garantindo a integridade do procedimento.

Diferentemente do PostgreSQL e do MongoDB, que foram instalados e executados em ambiente local, o SingleStore operou em um ambiente remoto. A validação consistiu em confirmar se o número total de registros inseridos correspondia ao planejado.

3 RESULTADOS E DISCUSSÃO

Nesta seção, são apresentados os resultados obtidos durante os testes de inserção e a análise correspondente.

Os testes com o PostgreSQL tiveram como foco mensurar o tempo de inserção, sem avaliação específica de propriedades transacionais ou mecanismos internos de controle de concorrência. O sistema concluiu a inserção de 2 milhões de registros em 9,60 segundos, utilizando scripts SQL automatizados compostos por comandos INSERT sequenciais.

Durante a execução dos testes, o PostgreSQL processou toda a carga sem erros, interrupções ou duplicações, demonstrando estabilidade

operacional sob o volume aplicado. Essa observação não constitui uma validação formal das propriedades ACID, porém nas condições testadas, o sistema manteve um comportamento previsível e reproduzível entre execuções. Embora aspectos como isolamento, gerenciamento de locks e controle de concorrência — amplamente documentados por Elmasri e Navathe (2010) — não tenham sido avaliados diretamente, o desempenho percebido confirma a robustez do modelo relacional em operações de escrita sequencial.

O resultado obtido reforça que, mesmo não sendo otimizado para inserções massivas, o PostgreSQL lida adequadamente com cargas elevadas quando operado em ambiente configurado de forma estável, preservando consistência e previsibilidade — características que explicam sua ampla adoção em aplicações corporativas, financeiras e governamentais.

Esse equilíbrio entre integridade e desempenho reflete o propósito estrutural dos bancos relacionais: priorizar a confiabilidade e a durabilidade dos dados mesmo com maior custo computacional durante operações intensivas.

Os testes realizados com o MongoDB tiveram como objetivo mensurar o impacto do tamanho dos lotes de inserção no tempo total de processamento, sem avaliação de propriedades transacionais ou características internas do mecanismo de escrita. O foco da análise se manteve ao verificar comportamento observável: o tempo necessário para completar a importação dos arquivos JSON.

Durante os experimentos, foram executadas duas configurações distintas. Na primeira, foram importados 20 arquivos contendo 100 mil documentos cada, totalizando 2 milhões de registros, com tempo total de 16,19 segundos. Na segunda configuração, o mesmo volume foi distribuído em 10 arquivos de 200 mil documentos, resultando em 13,40 segundos.

Ambas as execuções foram realizadas no MongoDB Compass, que fornece o tempo de execução após a conclusão de cada importação.

A Tabela 1 apresenta os tempos registrados, evidenciando que a redução do número de arquivos resultou em menor tempo total de inserção. A importação utilizando arquivos maiores foi aproximadamente 17% mais rápida, pois cada operação de carregamento envolve etapas internas de preparação, validação e alocação de recursos. Assim, ao diminuir a quantidade de importações necessárias, reduz-se também a sobrecarga operacional, contribuindo diretamente para um menor tempo total de importação.

Tabela 1 – Comparação de desempenho do MongoDB com diferentes tamanhos de lote

Cenário	**Arquivos**	**Registros por arquivo**	**Tempo total (s)**
Lote menor	20	100.000	16,19
Lote maior	10	200.000	13,40
Diferença percentual	–	Mesmo total de 2 milhões	**-17%**

Fonte: Do autor.

A diferença entre os cenários indica que o dimensionamento adequado dos lotes influencia diretamente o tempo total de inserção. De forma geral, os resultados demonstram que o MongoDB executou as operações de escrita de maneira estável e com desempenho condizente ao volume processado. A diferença entre os dois cenários pode ser explicada pelo fato de que, em bancos NoSQL orientados a documentos, o custo de cada operação envolve etapas como validação de schema, serialização BSON e commit no armazenamento. Conforme discutido por Sadalage e Fowler (2013) e reforçado por Rasheed, Qutqut e Almasalha (2019), operações em lote tendem a reduzir essa sobrecarga, pois amortizam o custo de inicialização por operação e permitem maior paralelismo interno durante a escrita.

O SingleStore, representante do modelo NewSQL, apresentou o menor tempo de inserção entre os bancos avaliados. Nos testes realizados, o sistema concluiu a escrita de 2 milhões de registros em 7,48 segundos. Esse desempenho evidencia que o SingleStore lida de forma eficiente com operações de escrita intensiva, mesmo utilizando apenas uma instância em ambiente cloud.

Durante a execução, não foram observadas falhas, interrupções ou perdas de registros, indicando estabilidade operacional sob o volume aplicado. Embora este estudo não tenha avaliado formalmente propriedades transacionais ou o comportamento distribuído do sistema, o desempenho consistente demonstra eficiência na execução das operações e previsibilidade no tempo total de processamento.

A boa performance observada pode ser interpretada devida as características apontadas na literatura do NewSQL, que descrevem sistemas dessa categoria como arquiteturas projetadas para otimizar cargas OLTP sem abrir mão do uso da linguagem SQL (Pavlo; Aslett, 2016). Estudos como Wei et al. (2021) e Zhang, Megargel e Jiang (2025) destacam que técnicas internas de paralelismo, processamento em memória e me-

canismos modernos de controle de concorrência contribuem para reduzir a latência nas operações de escrita — o que é coerente com o comportamento registrado nos testes.

Por outro lado, é importante destacar que este estudo não avaliou recursos típicos de ambientes distribuídos, como replicação entre nós, escalabilidade horizontal ou isolamento serializável. Sendo assim, a análise concentra-se apenas no comportamento observado: o SingleStore executou a carga de inserção de forma mais rápida, apresentando o melhor resultado entre os bancos testados.

De modo geral, os testes indicam que o SingleStore oferece um bom equilíbrio entre desempenho e uso de SQL, mostrando-se eficiente para operações de escrita em grande volume. Embora suas capacidades completas — como HTAP ou execução distribuída — não tenham sido objeto deste experimento, o desempenho apresentado reforça o potencial do modelo NewSQL para aplicações que exigem alto throughput com suporte à linguagem SQL tradicional.

A Tabela 2 exibe o tempo total para inserir 2 milhões de registros em cada modelo.

Tabela 2 – Demonstrativo dos tempos de inserção de cada banco de dados.

Modelo	**Banco de dados**	**Tempo total (s)**
Relacional	PostgreSQL	9,60
NoSQL (melhor cenário)	MongoDB	13,40
NewSQL	SingleStore	7,48

Fonte: Do autor.

A Tabela 3 apresenta uma síntese das principais características dos três modelos de sistemas de gerenciamento de banco de dados — SQL, NoSQL e NewSQL — com base na literatura especializada. A partir dela, é possível identificar os pontos fortes e as limitações de cada abordagem, evidenciando como as necessidades de escalabilidade, consistência e desempenho evoluíram ao longo das gerações de SGBDs. A análise dos critérios demonstra essa trajetória: de sistemas inicialmente centrados em consistência e estrutura rígida para soluções mais flexíveis e orientadas à escalabilidade. Desta forma, fica evidenciado que cada modelo apresenta vantagens específicas conforme o tipo de aplicação, reforçando a importância de compreender as características e o contexto de uso antes da escolha da tecnologia- de banco de dados.

Tabela 3 – Comparação entre bancos de dados SQL, NoSQL e NewSQL.

Critério	**SQL**	**NoSQL**	**NewSQL**
Modelo de dados	Tabelas (schema rígido)	Flexível: doc/coluna/grafo	SQL distribuído
Consistência	ACID forte	CAP (consistencia eventual)	ACID forte
Escalabilidade	Vertical	Horizontal nativa	Horizontal nativa
Desempenho	Bom em OLTP	Ótimo em grandes volumes	Alto em OLTP distrib.
Flexibilidade	Baixa	Alta	Moderada
Casos de uso	ERP, financeiro	Redes sociais, IoT	Bancos digitais, telecom
Limitações	Escala horiz. limitada	Pouco suporte a transações críticas	Ecossistema jovem
Sob carga	Escala mal	Escala bem	Escala bem c/ consistência

Fonte: Do autor.

Os resultados obtidos durante os testes demonstram que o modelo SQL continua a se manter sólido em cenários que exigem integridade e confiabilidade transacional, sendo amplamente utilizado em aplicações críticas, como sistemas bancários e corporativos. Seu principal desafio reside na escalabilidade horizontal, uma vez que as arquiteturas originais foram pensadas na utilização em ambientes centralizados.

Já os bancos NoSQL destacam-se pela alta performance em inserções massivas e pela flexibilidade estrutural, características que os tornam ideais para aplicações de Big Data, IoT e redes sociais. No entanto, a ausência de consistência imediata e o suporte limitado a transações complexas ainda restringem seu uso em contextos que exigem confiabilidade total dos dados.

O modelo NewSQL demonstrou ser uma evolução equilibrada entre os dois paradigmas anteriores, mantendo as propriedades ACID e o uso do SQL, ao mesmo tempo em que oferece escalabilidade horizontal e alta disponibilidade. O desempenho observado no SingleStore confirma a capacidade dos bancos NewSQL de combinar velocidade e consistência, apresentando resultados expressivos mesmo sob altas cargas de inserção.

4 CONCLUSÃO

Com base nos testes realizados e na pesquisa feita, verificou-se que o modelo relacional (PostgreSQL) mantém-se como uma solução sólida e confiável para aplicações que exigem alta integridade e conformidade com o padrão SQL, embora apresente limitações de escalabilidade horizontal em cenários de grande volume de escrita. Além disso, destaca-se por sua maturidade tecnológica e ampla adoção no mercado, oferecendo recursos como controle de concorrência multiversão (MVCC) e otimização de consultas, que asseguram desempenho estável em operações complexas. Apesar das restrições de escalabilidade, evoluções recentes em replicação e particionamento indicam que o modelo relacional ainda tem espaço para evolução em ambientes distribuídos e híbridos.

O modelo NoSQL (MongoDB) apresentou desempenho consistente nas inserções em larga escala, confirmando sua capacidade de lidar com grandes volumes de dados estruturados em documentos, ainda que com desempenho inferior aos modelos SQL e NewSQL na métrica específica avaliada. Embora os tempos obtidos tenham sido superiores, o comportamento registrado demonstra boa capacidade de processamento em operações de escrita massiva, especialmente quando utilizados lotes maiores. Esses resultados estão alinhados às características tradicionalmente descritas na literatura sobre bancos NoSQL, que destacam a flexibilidade estrutural e o suporte a arquiteturas distribuídas como fatores que favorecem aplicações que lidam com dados desestruturados e em grande escala. No entanto, tais aspectos não foram avaliados experimentalmente neste estudo — a análise restringiu-se ao desempenho de inserções locais, sem testar replicação, tolerância a falhas ou mecanismos de distribuição. Dessa forma, os achados confirmam o potencial do MongoDB para cenários que priorizam flexibilidade e velocidade de ingestão, enquanto sua limitação em oferecer consistência imediata reforça a necessidade de avaliar cuidadosamente seu uso em aplicações que demandam garantias transacionais mais rígidas.

O modelo NewSQL (SingleStore) destacou-se como um ponto de convergência entre os paradigmas SQL e NoSQL, apresentando o menor tempo de inserção entre os sistemas avaliados durante toda a carga de 2 milhões de registros. Os testes foram executados por meio da SingleStore Cloud, utilizando o console SQL online para envio dos comandos SQL. Dessa forma, o desempenho observado reflete o comportamento do

SingleStore executando operações de escrita a partir de um cliente remoto, sem envolver configuração de cluster distribuído, replicação ou outros recursos avançados da plataforma. Apesar disso, o SingleStore apresentou o menor tempo total entre os sistemas testados, indicando eficiência em operações de inserção massiva mesmo em um ambiente não otimizado e sem ajustes específicos de desempenho. Esse resultado está em concordância com a literatura, que descreve os bancos NewSQL como soluções projetadas para combinar propriedades ACID e linguagem SQL com arquiteturas modernas de alta escalabilidade. Entretanto, tais características — como processamento distribuído, tolerância a falhas e elasticidade horizontal — não foram avaliadas neste estudo, restringindo a análise ao comportamento observado no cenário testado. Ainda assim, o desempenho obtido demonstra que o SingleStore é uma alternativa promissora para aplicações que demandam alta taxa de escrita aliada à manutenção de consistência transacional.

Do ponto de vista analítico, os resultados evidenciaram que não existe um modelo universalmente superior, mas sim soluções específicas para diferentes contextos, sendo que cada tecnologia atende a propósitos distintos.

O estudo concentrou-se apenas em operações de inserção, não incluindo testes de leitura, atualização, exclusão ou cenários com múltiplas conexões concorrentes. Também não foram analisados consumo de recursos (CPU, memória e disco) ou mecanismos internos de transação, o que limita a avaliação mais completa dos SGBDs. Além disso, somente três bancos foram testados, restringindo a abrangência da comparação.

Como trabalhos futuros, recomenda-se: incluir testes de leitura, atualização, exclusão e concorrência; medir o uso de recursos de hardware durante os experimentos; ampliar a comparação para outros SGBDs, como Cassandra, Couchbase, CockroachDB e MariaDB Xpand; avaliar os sistemas em cenários distribuídos.

Essas extensões podem fornecer uma análise mais completa e fortalecer a comparação entre SQL, NoSQL e NewSQL.

O estudo, portanto, não encerra a discussão, mas serve como ponto de partida para novas pesquisas sobre desempenho e adaptação dos SGBDs em ambientes distribuídos e escaláveis.

REFERÊNCIAS

BERG, K.; SEYMOUR, T.; GOEL, R. **History of databases**. International

Journal of Management and Information Systems, 2012, v. 17, n. 1, p. 29–36.

BERNSTEIN, P. A.; GOODMAN, N. **Multiversion concurrency control—theory and algorithms**. ACM Transactions on Database Systems, 1983, v. 8, n. 4, p. 465–483.

BREWER, E. A. Towards robust distributed systems. In: PORTLAND, OR. **PODC**. [S.l.], 2000. v. 7, n. 10.1145, p. 343–477.

CHAMBERLIN, D. D. **Early history of sql**. IEEE Annals of the History of Computing, 2012, v. 34, n. 4, p. 78–82.

CLOUD, G. **Cloud Spanner Documentation**. 2023. <https://cloud.google.com/spanner/docs>. Acesso em: 28 ago. 2023.

CODD, E. F. **A relational model of data for large shared data banks**. Communications of the ACM, 1970, ACM, v. 13, n. 6, p. 377–387.

DASAN, J. R. P. maria; JOSHI, A. **Exploring latest nosql, newsql and sql database technologies**. Publishing House “Baltija Publishing”, 2025.

DATA, V. A. **VoltDB: In-Memory Database for Real-Time Applications**. 2022. <https://www.voltactivedata.com>. Acesso em: 04 dez. 2022.

ELMASRI, R.; NAVATHE, S. **Fundamentals of database systems, ch. 3**. [S.l.]: USA: Addison-Wesley Publishing Company„ 2010.

GILBERT, S.; LYNCH, N. **Brewer’s conjecture and the feasibility of consistent, available, partition-tolerant web services**. ACM SIGACT News, 2002, ACM, v. 33, n. 2, p. 51–59.

KNOB, R. R. et al. **Uma análise de soluções newsql**. Anais do Simpósio Brasileiro de Banco de Dados (SBBD), 2019.

LABS, C. **CockroachDB: Distributed SQL Database**. 2022. <https://www.cockroachlabs.com>. Acesso em: 04 dez. 2022.

MariaDB Corporation. **MariaDB Xpand Documentation**. 2023. Acesso em: 15 fev. 2025. Disponível em: <https://mariadb.com/docs/xpand/>.

PAVLO, A.; ASLETT, M. **What’s really new with newsql?** ACM Sigmod Record, 2016, ACM New York, NY, USA, v. 45, n. 2, p. 45–55.

RASHEED, Y.; QUTQUT, M. H.; ALMASALHA, F. **Overview of the current status of nosql database**. International Journal of Computer Science and Network Security (IJCSNS), 2019, IJCSNS, v. 19, n. 4, p. 47–53.

SADALAGE, P. J.; FOWLER, M. **NoSQL distilled: a brief guide to the emerging world of polyglot persistence**. Crawfordsville, Indiana: Pearson Education, 2013.

SINGLESTORE. **SingleStoreDB Architecture Overview**. 2023. Disponível em: https://docs.singlestore.com/cloud/architecture/.

WEI, X. et al. Unifying timestamp with transaction ordering for {MVCC} with decentralized scalar timestamp. In: **18th USENIX Symposium on Networked Systems Design and Implementation (NSDI 21)**. [S.l.: s.n.], 2021. p. 357–372.

ZHANG, Z.; MEGARGEL, A.; JIANG, L. **Performance evaluation of newsql databases in a distributed architecture**. IEEE Access, 2025, IEEE.