Banco de Dados Relacional e Não Relacional: Uma Breve Comparação Analítica

Lucas Henrique de Oliveira Lacerda e Leonardo Barreto Campos

Instituto Federal de Educação, Ciência e Tecnologia da Bahia (IFBA)
Av. Sérgio Vieira de Mello, 3150, Zabelê – Vitória da Conquista/BA – Brasil

lucas___lacerda@hotmail.com, leonardobcampos@ifba.edu.br;

Abstract. *This paper addresses the growing demand for effective solutions in the business environment, leading to system integration through relational databases. However, the pursuit of efficiency has driven the migration to NoSQL databases, aiming to enhance performance and gain competitive advantages. This paper aims to justify this technological shift based on the need for companies to improve data management and tackle scalability and flexibility challenges. The core problem revolves around the feasibility of migrating to NoSQL databases in data integration activities, with a focus on the difficulties and benefits of this transition.*

Resumo. *Este artigo aborda a demanda crescente por soluções eficazes no ambiente empresarial, levando à integração de sistemas por meio de bancos de dados relacionais. No entanto, a busca por eficiência tem impulsionado a migração para bancos de dados NoSQL, visando aprimorar o desempenho e conquistar vantagens competitivas. Este trabalho busca justificar essa mudança tecnológica com base na necessidade das empresas de melhorar a gestão de dados e enfrentar desafios de escalabilidade e flexibilidade. O cerne do problema é a viabilidade da migração para bancos de dados NoSQL em atividades de integração de dados, com foco nas dificuldades e nos benefícios dessa transição.*

1. Introdução

A gestão eficiente de dados desempenha um papel fundamental no sucesso das empresas, pois fornece *insights* estratégicos, melhora a eficiência operacional e garante a segurança das informações [Allam e Dhunny 2020]. Segundo a SERPRO (2020), o volume estimado de informações digitais gerado no mundo em 2020 foi de 5,4 zettabytes e que até 2025 a humanidade terá produzido cerca de oito vezes mais dados do que em 2020. Estes dados ressaltam a importância de implementar sistemas eficientes para armazenar, gerenciar e processar essas informações de forma eficaz.

Com a crescente disponibilidade de dados, torna-se ainda mais decisivo para as empresas implementarem estratégias e tecnologias que lhes permitam extrair valor desses dados e obter vantagem competitiva no mercado. A análise de dados oferece as empresas a capacidade de identificar tendências, padrões e perceções que podem impulsionar a tomada de decisões mais informadas e estratégicas.

Ao analisar dados de forma eficiente, as empresas podem identificar oportunidades de crescimento, entender melhor o comportamento do cliente, otimizar processos internos, personalizar produtos ou serviços e antecipar as necessidades do mercado. Essas vantagens competitivas podem levar a um melhor posicionamento no mercado, aumento da eficiência operacional e, em última análise, ao sucesso empresarial.

Nesse sentido, os bancos de dados (BDs) se destacam como uma das soluções mais amplamente utilizadas pelas empresas. No entanto, é importante destacar que existem dois tipos distintos de bancos de dados: os relacionais, baseados em Linguagem de Consulta Estruturada do inglês *Structured Query Language* (SQL) e os não-relacionais (NoSQL, *Not Only* SQL). Enquanto os bancos de dados SQL seguem um modelo de tabela relacional e possuem um esquema fixo, os bancos de dados NoSQL oferecem maior flexibilidade e escalabilidade, permitindo o armazenamento de grandes volumes de dados não estruturados.

Diante da diversidade de sistemas adotados pelas empresas, surge a necessidade de uma comunicação eficiente entre estes dois tipos de BDs, a fim de garantir um funcionamento mais ágil e eficiente dos softwares. A implementação de uma comunicação efetiva entre sistemas representa uma inovação que traz retornos rápidos e eficientes, tornando as empresas mais competitivas no mercado. Contudo, diante da complexidade dessa integração entre sistemas, surgem questionamentos cruciais que permeiam a tomada de decisões estratégicas das empresas.

A viabilidade da migração da tecnologia de bancos de dados relacionais para bancos de dados NoSQL em atividades de integrações de dados emerge como uma questão essencial. Essa problemática central pode ser desdobrada em três perguntas fundamentais, a saber: **(i)** É possível e viável implementar essa mudança tecnológica em todas ou quase todas as soluções de integrações de dados existentes? **(ii)** Quais são as principais dificuldades enfrentadas no desenvolvimento de sistemas que integram informações de dados ao realizar essa migração? e **(iii)** A adoção de bancos de dados NoSQL pode trazer benefícios significativos em termos de desempenho e escalabilidade do sistema?

2. CONCEITOS TEÓRICOS

2.1 Bancos de Dados Relacionais

O modelo relacional é amplamente utilizado e possui importantes características que garantem a consistência dos dados. Como afirma Martins Filho (2015), "o modelo relacional molda os dados de forma que eles fiquem organizados no formato de tabelas, de maneira mais convencional, em relações". Além disso, o modelo relacional tem uma linguagem padrão para manipulação de dados, o SQL, como observado por Kokay (2012).

Além da restrição de integridade, outra característica importante do modelo relacional é a normalização, processo que visa manter a consistência das informações ao seguir uma série de passos propostos por Edgar Frank Codd, criador do modelo.

Segundo Machado e De Abreu (1996), "qualquer operação pode ocasionar problemas como perda de informação caso o BD não esteja normalizado". No entanto,

esse processo pode aumentar o custo de desempenho, já que toda consulta do tipo junção necessita realizar a busca em várias tabelas para recuperar o registro.

Por outro lado, os bancos NoSQL geralmente não seguem o processo de normalização, o que pode ser justificado pela desnormalização ser uma das características desses bancos. Como afirma Cattell (2011), "desnormalização é o uso de técnicas de modelagem de dados para aumentar o desempenho em consultas, sacrificando, em alguns casos, a consistência dos dados".

2.2 Bancos de Dados NoSQL

Atualmente, a abordagem do banco de dados NoSQL é voltada principalmente para tecnologias e aplicações da web, já que muitas dessas aplicações exigem uma grande capacidade de armazenamento, que ultrapassa a capacidade dos bancos de dados relacionais (Hecht & Jablonski, 2011).

Existem vários tipos de bancos NoSQL, cada um com tecnologias específicas, incluindo modelo chave-valor, orientados a grafos, orientados a colunas e orientados a documentos. Para este trabalho, foi escolhido o banco de dados orientado a documentos, uma das tecnologias NoSQL mais famosas e utilizadas, código-aberto e que salva os dados de maneira simples e de fácil compreensão e utilização.

Os sistemas que utilizam esse tipo de banco de dados possuem, em sua grande maioria, características específicas, como maior facilidade para implementar API (Interface de Programação de Aplicativos) para acesso aos dados, escalabilidade horizontal e esquema dinâmico ou sem esquema (LÓSCIO ET AL., 2011).

Por essas características, os bancos NoSQL se tornam mais adequados para armazenar uma grande quantidade de dados estruturados, semiestruturados ou reestruturados, onde não há necessidade de normalização das informações que serão salvas (FOWLER, 2012).

2.3 Modelo Orientados a Documentos

O modelo orientado a documentos é baseado no armazenamento de dados em coleções de documentos no formato JSON. Cada documento contém uma chave única de identificação especial "_id", que é exclusiva também dentro da coleção de documentos, o que permite a identificação global do documento na coleção (HECHT e JABLONSKI, 2011).

Ao contrário do modelo relacional, o modelo orientado a documentos não possui um esquema fixo e rígido, permitindo a atualização e inclusão de novos campos sem a necessidade de alterar a estrutura do documento inteiro. Isso proporciona mais flexibilidade na modelagem de dados e no processo de desenvolvimento de aplicativos (LÓSCIO et al., 2011).

2.4 Integração de Banco de Dados

A integração de dados é o processo de combinar dados de diversas fontes diferentes para criar uma visão unificada e abrangente. Segundo Kimball e Ross (2013), a integração de dados envolve a coleta, transformação e combinação de dados de diferentes sistemas e fontes de dados para fornecer uma visão consistente das informações.

De acordo com Rahm e Bernstein (2001), a heterogeneidade dos dados é um desafio comum enfrentado na integração de dados, pois os dados podem apresentar diferenças de formatos, estruturas e semântica. Para superar essa heterogeneidade, existem várias abordagens e técnicas. Por exemplo, a extração, transformação e carga (ETL) é uma abordagem comum, onde os dados são extraídos das fontes, passam por transformações para se adequar a um esquema comum e são carregados em um repositório central (Lenzerini, 2002). Outra abordagem é a virtualização de dados, que permite acessar os dados em tempo real, sem a necessidade de consolidá-los em um local centralizado (Halevy et al., 2005).

De forma prática, existem diversas opções disponíveis para realizar integrações, tais como o uso de API (Interface de Programação de Aplicações), compartilhamento de dados eletrônicos, como blocos de notas e planilhas, e integração de banco a banco.

A primeira seria API (*Aplication Programming Interface*), onde as APIs (Interface de Programação de Aplicações) são um conjunto de rotinas, protocolos e ferramentas que permitem a comunicação entre diferentes sistemas e plataformas de forma segura e eficiente. Essa opção de integração é amplamente utilizada em empresas de todos os portes e setores, pois permite o compartilhamento de informações em tempo real, reduzindo a necessidade de retrabalho e aumentando a produtividade das equipes. Além disso, a utilização de APIs pode ajudar a otimizar processos internos e externos das empresas, gerando uma vantagem competitiva no mercado.

A segunda, compartilhamento de dados eletrônicos: O compartilhamento de arquivos eletrônicos entre diferentes sistemas é uma opção de integração menos segura e eficiente do que as APIs, mas ainda é utilizada por algumas empresas. Essa opção pode envolver o uso de blocos de notas, planilhas, documentos em formato PDF, entre outros. Apesar de ser uma opção mais simples, o compartilhamento de arquivos eletrônicos pode gerar problemas de compatibilidade, segurança e atualização de informações, o que pode prejudicar o desempenho e a eficiência dos sistemas integrados.

A terceira e última integração de banco a banco: A integração de banco a banco envolve a conexão dos bancos de dados dos diferentes sistemas, permitindo o acesso e o compartilhamento de informações entre eles. Essa opção de integração é mais complexa do que as outras duas, mas pode ser uma solução mais eficiente para empresas que precisam integrar grandes volumes de dados entre diferentes sistemas. Para implementar a integração de banco a banco, é necessário investir em infraestrutura e em equipes especializadas em banco de dados, além de garantir a segurança e a compatibilidade dos sistemas integrados. No entanto, uma vez implementada, essa opção pode gerar benefícios significativos para as empresas, como a redução de erros, o aumento da eficiência e a melhoria da tomada de decisões.

3. Trabalhos correlatos

A seleção dos trabalhos apresentados nesta seção foi realizada por meio da plataforma Google Scholar, utilizando as palavras-chave "comparação de bancos de dados SQL e NoSQL". A pesquisa abrangeu publicações entre 2022 e 2023, e a escolha foi delimitada com o objetivo de garantir a relevância e atualidade das informações na revisão da literatura. A busca resultou em 151 trabalhos acadêmicos. Após a leitura dos resumos, foram selecionados dois trabalhos que atendiam ao critério de inclusão, sendo diretamente relacionados ao tema deste artigo.

No estudo conduzido por Galvão Filho e da Silva (2022), foi realizado um processo de migração de uma base de dados PostgreSQL para uma solução NoSQL, utilizando a ferramenta ETL Pentaho Data Integration (PDI). Os resultados obtidos destacaram as principais vantagens dessa migração, incluindo alta disponibilidade, escalabilidade e capacidade de lidar com grande volume de dados. No entanto, é importante ressaltar que durante o processo de atualização da configuração do banco de dados, existe a possibilidade da aplicação ficar *offline* por alguns minutos.

No segundo trabalho selecionado, escrito por Knijnik (2022), foi realizado um estudo comparativo entre os sistemas de gerenciamento de banco de dados MongoDB e PostgreSQL, com foco no contexto de jogos digitais. O estudo revelou diversas diferenças entre os dois sistemas. De acordo com os resultados, o estudo destacou a complexidade da modelagem de dados em bancos de dados relacionais, em comparação com o MongoDB, que se mostrou mais eficiente na etapa de inserção de dados, não exigindo alterações prévias. Além disso, o MongoDB demonstrou ser mais econômico em termos de uso de memória.

No que diz respeito às operações de filtragem, o estudo constatou que o MongoDB foi mais eficiente em operações simples, enquanto o PostgreSQL apresentou vantagens em comparações mais complexas, como testes de “maior”. O PostgreSQL também obteve melhor desempenho ao retornar tabelas de um determinado tipo, diferentemente do MongoDB, que retornou todas as entradas em uma coleção. O estudo concluiu que esses resultados podem auxiliar na escolha do banco de dados mais adequado para o desenvolvimento de jogos digitais.

Ambos os estudos contribuem significativamente para a compreensão das complexidades envolvidas nas escolhas de banco de dados. O trabalho de Galvão Filho e da Silva (2022), destaca os desafios práticos enfrentados durante a migração, oferecendo uma análise detalhada dos obstáculos e benefícios encontrados nesse processo específico. Por sua vez, o estudo de Knijnik (2022) fornece insights valiosos sobre a seleção de bancos de dados em ambientes específicos, como no desenvolvimento de jogos digitais.

O que diferencia esta pesquisa é sua dedicação específica à eficiência do processo de integração de dados entre bancos de dados relacionais e não-relacionais. Enquanto os trabalhos anteriores exploram aspectos específicos, como migração e seleção de bancos de dados em ambientes particulares, a presente abordagem concentra-se em compreender e otimizar a integração de dados em cenários que envolvem ambos os tipos de bancos.

4. Material e Métodos

A compreensão da dinâmica entre os sistemas de frente de loja e retaguarda é essencial para contextualizar o ambiente de teste utilizado neste estudo. Nos sistemas empresariais, o sistema de frente de loja, comumente conhecido como PDV (Ponto de Venda), desempenha o papel fundamental de registrar as transações comerciais realizadas no estabelecimento, incluindo vendas, pagamentos e outras atividades de varejo. Por outro lado, o sistema de retaguarda, geralmente constituído por um banco de dados centralizado, é responsável por armazenar e gerenciar os dados de todas as lojas ou filiais da empresa. Esses dados são utilizados para fins administrativos, como por exemplo controle de estoque, gerenciamento de clientes e emissão de relatórios gerenciais, entre outros. A compreensão dessa lógica operacional nos permite adentrar na descrição do ambiente de teste empregado para simular o processo de integração de dados.

Para realizar esta pesquisa, foi desenvolvido um ambiente de teste que simula o processo de integração de dados entre sistemas. Utilizamos duas amostras de PDV (Ponto de Venda/Caixa), uma virtualizada e outra executada diretamente no host, além de um banco de dados de retaguarda (simulando o banco da matriz).

No ambiente de teste, foi implementado o SQL Server Express, que hospeda dois BDs: um representa os dados de um sistema *front-end* de uma loja, e o outro agindo como um banco de retaguarda, onde esse seria o banco da matriz da empresa com centrando todos os dados dos bancos das lojas. Segue a Figura 1, que ilustra esse processo, destacando em vermelho a parte que foi simulada.

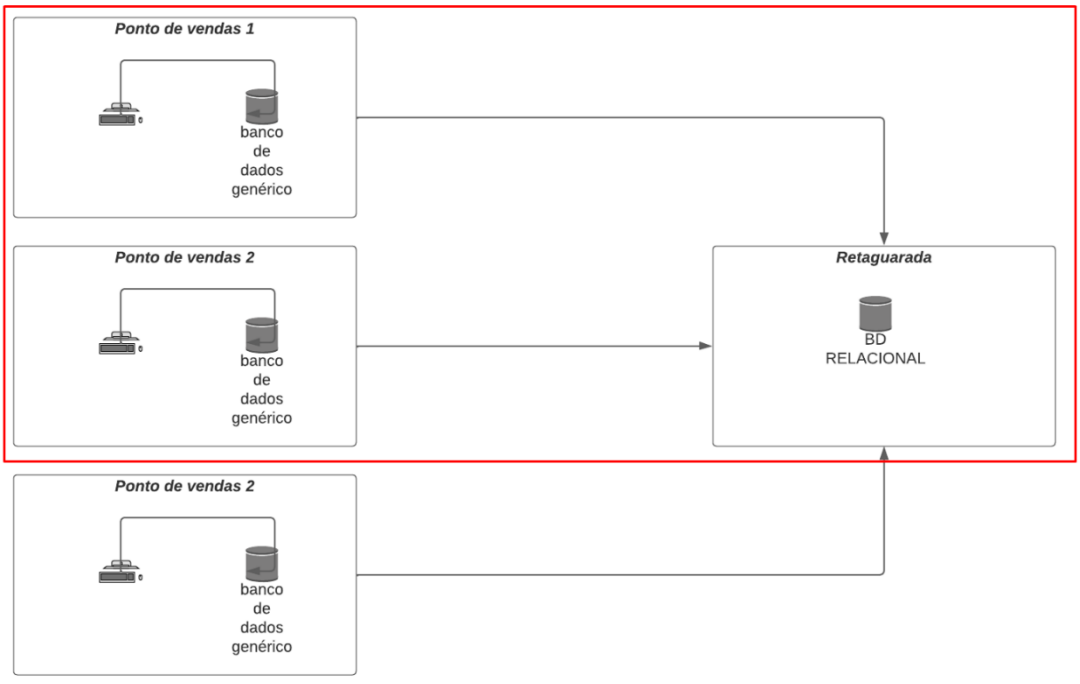

Figura 1. Processo de Integração PDV e Retaguarda

Nessa simulação, optamos por criar uma tabela idêntica nas duas bases, mantendo os mesmos campos e estrutura. Isso foi feito para direcionar o foco do estudo exclusivamente para o processo de integração de dados. Deixamos em aberto, para trabalhos futuros, a realização de testes relacionados à extração e encaminhamento dos dados para tabelas adicionais. O esquema relacional adotado é apresentado na Figura 2.

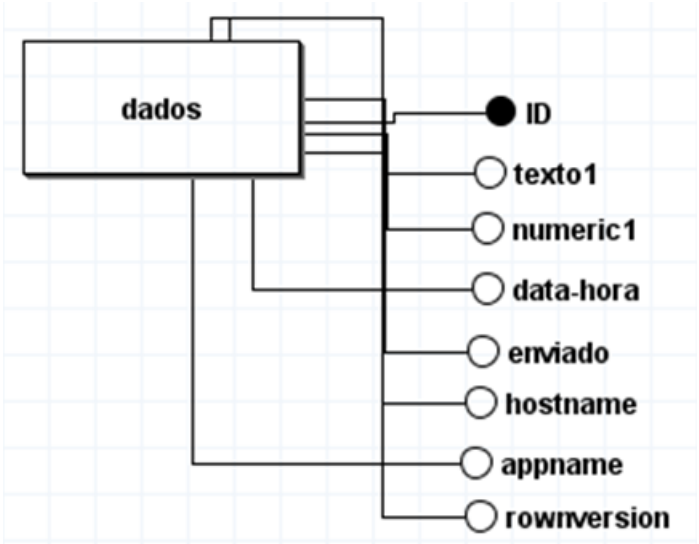

Figura 2. Esquema relacional da do BD da simulação

Na Tabela 1, são apresentadas as informações detalhadas sobre as colunas, acompanhadas por uma breve descrição de cada uma, referentes ao esquema do banco PDV.

Tabela 1. **Colunas da tabela com descrição do BD de simulação do banco PDV**

Colunas	Descrição
ID	Como chave *primary* do registro
Texto1	Uma coluna Texto1 do tipo *varchar* (max), com o conteúdo em *varchar* da informação integrada ou um JSON com os dados para ser extraído posteriormente
Numeric1	Numero podendo ser a chave do registro que está subindo
Data_hora	Para ter a informação de buscar de quando o registro subiu
Enviado	Sendo utilizado para saber se aquela informação já foi integrada ‘S’ (como pendente) e ‘P’ (processado)
Hostaname	Seria coluna default para identificar a maquina que subiu a informação;
APPname	Seria coluna default para identificar a aplicação que subiu a informação;
Rownversion	Tipo de dados numéricos exclusivos gerado automaticamente para saber a versão do registro da linha

O processo de integração foi conduzido da seguinte maneira: um script, agendado no sistema operacional do PDV, estabeleceu conexão com o banco de dados front-end (Database 1) e transferiu os registros da tabela dados para o banco de dados de retaguarda (Database 2), mantendo uma estrutura semelhante à tabela do banco de origem (PDV).

Além disso, o script registrou o tempo médio de execução, fornecendo informações cruciais sobre o desempenho da integração. No Database 2, foi implementada uma trigger para registrar o momento da inserção de dados, gerando um log adicional. Esses registros da trigger foram comparados com os registros gerados pelo script, contribuindo para a análise de desempenho. O esquema relacional da tabela da retaguarda pode ser observado na Figura 3.

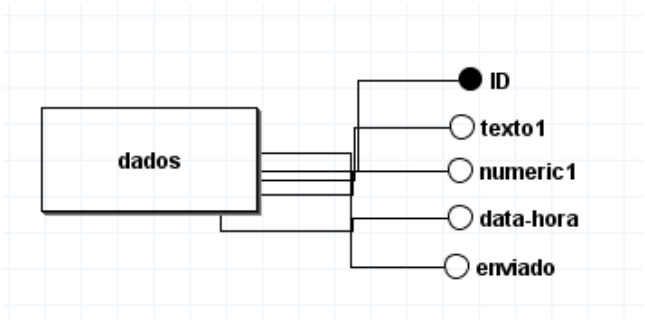

Figura 3. Esquema relacional da do BD da simulação no esquema retaguarda

A seguir, na Tabela 2 é apresentado os dados contendo as informações detalhadas sobre as colunas e uma breve descrição de cada uma sobre o esquema do banco retaguarda:

Tabela 2. **Colunas da tabela com descrição do BD de simulação do banco Retaguarda**

Colunas	Descrição
ID	Como chave *primary* do registro
Texto1	Uma coluna Texto1 do tipo *varchar* (max), com o conteúdo em *varchar* da informação integrada ou um JSON com os dados para ser extraído posteriormente
numeric1	Número podendo ser a chave do registro que está subindo
Data_hora	Para ter a informação de buscar de quando o registro subiu
Enviado	Sendo utilizado para saber se aquela informação já foi integrada (‘S’ como pendente) e ‘P’ (processado)

Em seguida, o mesmo teste foi realizado, mas desta vez substituindo o banco de dados relacional SQL Server pelo MongoDB como banco de dados de retaguarda. Os resultados deste estudo incluem a média de tempo necessário para realizar o processo de integração usando o banco de dados relacional (SQL Server). Posteriormente, o Database 2 foi substituído por um banco de dados NoSQL, especificamente o MongoDB, e o

mesmo processo de integração foi executado. Por fim, foram comparados os tempos de execução e o desempenho entre os dois cenários, seguido de uma avaliação da eficácia e da eficiência de cada abordagem. Para garantir uma comparação justa entre os dois cenários, utilizou-se as seguintes especificações de equipamentos e tecnologias apresentados na Tabela 3.

Tabela 3. Hardware, Software e Linguagem de Programação

Hardware	
Disco Rígido	SSD ThinkLife de 480GB
Memória RAM	12 GB Kingston
Processador	Intel Core i5 6400
Software	
Banco de dados Não Relacional	MongoDB
Manipulação do BD relacional	Visual Studio
Manipulação do BD Não Relacional	MongoDB Compass
Banco de dados Relacional	SQL Server Express
Sistema Operacional	Windows 10 Home e Ubuntu 22.04.3 (VM)
Linguagem de Programação Python	
Biblioteca	**Funcionalidade**
Pyodbc	Conector Python para SQL Server
Configparser	Realiza leitura de arquivos de configuração .INI
Datetime	Registrar data e hora nos logs
Tqdm	Exibir uma barra de carregamento no terminal
Signal	Permitir o uso de uma tecla de cancelamento
Pymongo	Conector Python com o MongoDB
Psutil	Coletar dados da CPU e RAM do sistema
Pytz	Obter o fuso horário atualizado do Brasil

Os códigos e scripts usados neste estudo estão disponíveis no seguinte repositório do GitHub : https://github.com/lucas-lacerda-darkon/TCC2-Integration-databases. Esses recursos e configurações garantiram que os testes fossem realizados em um ambiente controlado e proporcionaram uma base sólida para a comparação entre as abordagens de banco de dados relacional e não-relacional.

5. Resultados

5.1 BDs Relacionais

No cenário inicial, procedeu-se com a sincronização entre dois bancos de dados relacionais no SQL Server Express. Para tanto, foi estabelecida uma instância, com um banco já populado com os dados destinados à integração, enquanto o outro permaneceu vazio.

Esses dados foram inseridos automaticamente por meio de um script que preenchia as informações de forma sequencial. Cada entrada foi identificada por um ID único como chave primária, seguido por uma coluna de texto e outra de tipo numérico. A data e hora foram registradas utilizando a função "*getdate*()", enquanto uma coluna de status foi preenchida como "N" que nesse processo é utilizado como pendente onde é do tipo caractere.

As colunas de "*hostname*" e "*appname*" foram deixadas como nulas, pois seriam preenchidas pela aplicação conectada ao banco de dados. Vale ressaltar que, como a carga inicial foi realizada via script, essas informações não foram preenchidas inicialmente, foi incluída uma coluna de "*rowversion*", gerando um ID binário para cada linha inserida, possibilitando a verificação do último registro inserido, conforme mostra a Figura 4.

100 %

Resultados | Mensagens

	id	texto1	numeric1	data_hora	enviado	hostname	appname	rowversion
1	1	registro: 1	1	2024-01-30 13:34:29.693	S	NULL	NULL	0x00000000000007D1
2	2	registro: 2	2	2024-01-30 13:34:29.700	S	NULL	NULL	0x00000000000007D2
3	3	registro: 3	3	2024-01-30 13:34:29.703	S	NULL	NULL	0x00000000000007D3
4	4	registro: 4	4	2024-01-30 13:34:29.703	S	NULL	NULL	0x00000000000007D4
5	5	registro: 5	5	2024-01-30 13:34:29.707	S	NULL	NULL	0x00000000000007D5
6	6	registro: 6	6	2024-01-30 13:34:29.710	S	NULL	NULL	0x00000000000007D6
7	7	registro: 7	7	2024-01-30 13:34:29.710	S	NULL	NULL	0x00000000000007D7
8	8	registro: 8	8	2024-01-30 13:34:29.713	S	NULL	NULL	0x00000000000007D8
9	9	registro: 9	9	2024-01-30 13:34:29.713	S	NULL	NULL	0x00000000000007D9
10	10	registro: 10	10	2024-01-30 13:34:29.717	S	NULL	NULL	0x00000000000007DA
11	11	registro: 11	11	2024-01-30 13:34:29.717	S	NULL	NULL	0x00000000000007DB
12	12	registro: 12	12	2024-01-30 13:34:29.720	S	NULL	NULL	0x00000000000007DC

Figura 4. Dados inserido no banco PDV

Os registros apresentados na Tabela 4 indicam que, no cenário relacional, a quantidade significativa de 1.000.002 registros foi copiada sem ocorrência de erros, demonstrando a eficácia da integração.

A média de uso da CPU, registrada em 7.00%, e a média de uso de RAM, mantida em 50.45%, sugerem uma utilização eficiente dos recursos do sistema durante o processo. Além disso, os dados de data e hora de início e conclusão (05/11/2023 11:55 e 05/11/2023 13:54, respectivamente) fornecem informações temporais determinantes para avaliar a duração do procedimento de integração.

Tabela 4. Resultados utilizando Banco de Dados Relacional

Característica	Resultado
Quantidade copiada	1.000.002 registros
Quantidade de erros	0
Média de uso da CPU	7.00%
Média de uso de RAM	50.45%
Data e hora de início	2023-11-05_11-55-41
Data e hora de conclusão	2023-11-05 13:54:50
Hostname do Banco 1	.\SQLEXPRESS
Hostname do Banco 2	.\SQLEXPRESS
Nome do Banco 1	Database1
Nome do Banco 2	Database2

Esses dados foram gerados pelo script em Python ao final da execução. Ao final do processo é salvo em um arquivo de texto os dados gerados.

5.2 BD Não-Relacional

No segundo cenário, repetiu-se o processo do Cenário 01, com a distinção de que o banco de dados no qual os dados seriam inseridos foi substituído por um banco não-relacional, o MongoDB, totalizando 1.000.002 registros.

Os resultados de desempenho obtidos estão detalhados na Tabela 5, fornecendo uma visão abrangente das métricas relacionadas a essa configuração específica. Essa abordagem permitiu explorar as diferenças e semelhanças no processo de integração ao utilizar um banco de dados não-relacional em comparação com a abordagem relacional do Cenário 01.

Tabela 5. Resultados utilizando Banco Não-relacional

Resultados	
Quantidade copiada	1.000.002 registros
Quantidade de erros	0
Média de uso da CPU:	7.66%
Média de uso de RAM:	51.16%
Data e hora de início	2023-11-05_15-42-06
Data e hora de conclusão	2023-11-05 17:49:31
Hostname do Banco 1	.\SQLEXPRESS
Hostname do Banco 2	.\SQLEXPRESS
Nome do Banco 1	Database1
Nome do Banco 2	Database1

5.3 BD Relacional versus BD Não-Relacional utilizando VM

Após conduzir os testes por meio de uma Máquina Virtual (VM), notamos variações nos resultados em comparação com os testes anteriores realizados em um ambiente distinto. Essa disparidade pode ser atribuída ao compartilhamento de recursos da máquina hospedeira, o que resulta em uma distribuição desigual de capacidade de processamento e memória, impactando diretamente o desempenho dos processos executados na VM.

Além disso, é importante ressaltar que o sistema operacional utilizado na VM difere do sistema presente na máquina que abriga o banco de retaguarda. Esses fatores combinados contribuem para as divergências identificadas nos resultados dos testes.

Os resultados dos testes revelam uma disparidade significativa nos tempos de execução entre o SQL Server e o MongoDB quando o processo foi executado em uma Máquina Virtual (VM). Embora ambos os testes tenham obtido sucesso na transferência de 1.000.000 de registros, o tempo necessário para concluir o processo variou consideravelmente. O SQL Server registrou um tempo de início às 14:05:53 e uma conclusão às 18:34:06, enquanto o MongoDB iniciou às 19:19:25 e finalizou às 23:17:41, conforme mostra a Tabela 6.

Tabela 6. Resultados BD Relacional e BD Não-Relacional pela VM

MongoDB			*SQL Server*		
CPU	RAM	Tempo	CPU	RAM	TEMPO
14.72%	81.78%	3:58:16	16.06%	88.72%	4:28:13

Essa diferença no tempo de execução pode ser atribuída a várias razões, incluindo a arquitetura de cada banco de dados, a eficiência das operações de leitura e gravação de dados pela VM e a forma como cada sistema lida com a gestão de recursos em um ambiente virtualizado. Além disso, as métricas de uso da CPU e RAM também divergem entre os dois bancos de dados, sugerindo diferentes demandas de recursos durante o processo de integração.

6. Comparação de Desempenho entre Bancos de Dados Relacional e Não-Relacional pelo o host

Ao comparar os resultados dos dois cenários de integração de dados um usando o banco de dados relacional SQL Server e o outro usando o banco de dados não-relacional MongoDB, conforme mostra a Tabela 7.

Tabela 7. Resultado Não-Relacional e Relacional pelo host

BD Não-Relacional		**BD Relacional**	
Quantidade copiada:	1.000.002 registros	Quantidade copiada:	1.000.002 registros
Quantidade de erros:	0	Quantidade de erros:	0
Média de uso da CPU:	7.66%	Média de uso da CPU:	7.00%
Média de uso de RAM:	51.16%	Média de uso de RAM :	50.45%
Data e hora de início:	05/11/2023 15:42	Data e hora de início:	05/11/2023 11:55
Data e hora de conclusão:	05/11/2023 17:49	Data e hora de conclusão:	05/11/2023 13:54

Algumas observações relevantes podem ser feitas:

- Quantidade Copiada: Ambos os cenários conseguiram copiar com sucesso a mesma quantidade de registros, ou seja, 1.000.002 registros, sem erros no processo. Portanto, em termos de quantidade, ambos foram igualmente eficientes.
- Uso de Recursos: A média de uso da CPU foi ligeiramente maior no cenário MongoDB, atingindo 7.66%, enquanto o cenário SQL Server registrou 7.00%. Da mesma forma, a média de uso de RAM também foi um pouco mais alta no cenário MongoDB, com 51.16%, em comparação com 50.45% no cenário SQL Server. Embora as diferenças sejam pequenas, indicam que o MongoDB pode requerer um pouco mais de recursos em comparação com o SQL Server para executar a mesma tarefa de integração.

- Tempo de Execução: Os horários de início e término da operação revelam que o cenário MongoDB levou mais tempo para concluir a integração, com uma duração total de 2 horas e 7 minutos. Enquanto isso, o cenário SQL Server concluiu a operação em 1 hora e 59 minutos. Isso indica que o SQL Server foi mais rápido na execução da tarefa de integração de dados.
- Hosts e Bancos de Dados: No cenário MongoDB, o banco de dados de destino (Banco 2) estava hospedado localmente no host “localhost”. Por outro lado, no cenário SQL Server, o Banco 2 estava localizado no mesmo host do Banco 1 (Banco 2 e Banco 1 estavam em ".\SQLEXPRESS"). Isso sugere que o cenário MongoDB envolveu uma comunicação entre hosts, o que pode ter contribuído para o aumento do tempo de execução.

A transição da tecnologia de bancos de dados relacionais para bancos de dados NoSQL, no contexto de atividades de migração de dados, apresenta desafios e oportunidades significativos. Entre as dificuldades enfrentadas, destaca-se a necessidade de adaptar ou reescrever os sistemas existentes para se alinharem com a estrutura flexível dos bancos de dados NoSQL.

Bancos de dados NoSQL oferecem flexibilidade e escalabilidade, permitindo o armazenamento eficiente de grande volume de dados não estruturados. Esta mudança pode resultar em ganhos significativos de desempenho e escalabilidade do sistema. Vale ressaltar que os resultados deste trabalho demonstraram que, em determinadas circunstâncias (como o processo de integração que foi apresentado nesse artigo), o consumo de recursos do banco de dados NoSQL foi superior ao do banco de dados relacional, destacando a importância de uma análise detalhada na escolha da tecnologia de banco de dados dependendo dos requisitos específicos do sistema e do cenário de uso.

7. Considerações Finais

A análise comparativa entre os cenários de integração de dados, envolvendo bancos de dados relacionais e não-relacionais, ressalta a capacidade de ambos para a execução bem-sucedida da tarefa. No entanto, o cenário que incorpora um banco de dados relacional, como o *SQL Server*, apresentou um desempenho levemente superior, evidenciado por uma utilização mais eficiente de recursos e um tempo de execução mais curto. Essa constatação destaca a importância crítica de uma seleção cuidadosa da tecnologia de banco de dados, levando em consideração os requisitos e metas específicos de projetos de integração de dados.

Este estudo corrobora a viabilidade da transição tecnológica de bancos de dados relacionais para *NoSQL* em projetos de integração de dados. Ambos os modelos revelaram-se eficazes, mas a escolha entre *SQL* e *NoSQL* deve ser deliberada, considerando as particularidades de cada solução e os objetivos do projeto.

As adversidades identificadas durante o processo de integração ressaltam a necessidade de um planejamento minucioso nas escolhas de tecnologias. Focar na preservação da integridade dos dados durante a transição entre sistemas, levando em consideração tanto os aspectos computacionais quanto as especificações dos desenvolvedores na criação de rotinas e scripts, é crucial. Esse enfoque é fundamental para criar um processo eficaz e garantir uma integração bem-sucedida das informações.

A simulação realizada neste artigo evidencia que não haveria ganhos significativos em termos de desempenho e escalabilidade, quando focalizando exclusivamente no processo de integração. No entanto, é vital salientar que o estudo concentrou-se nessa fase específica, e há um leque de possíveis otimizações e melhorias que podem ser exploradas em etapas subsequentes do ciclo de vida do sistema.

No contexto de futuras investigações e melhorias, destacamos algumas oportunidades: (i) implementar uma rotina no banco de dados 1 (PDV) que extrai dados de outras possíveis tabelas para a tabela de integração, estabelecendo uma rotina de encapsulamento dos registros. Esses registros encapsulados podem, então, ser copiados para o banco de retaguarda, utilizando o mesmo script de integração deste artigo, e passar por um subsequente processo de desencapsulamento; (ii) realizar simulações utilizando índices para avaliar se há uma possível melhora no desempenho do processo e das operações de DML (insert, update, delete) e consultas de logs; (iii) implementar um mecanismo de expurgo nas tabelas de integração, seja por meio de gatilhos ou agendamentos internos do próprio banco, visando manter a eficiência do ambiente ao longo do tempo; (iv) explorar a implementação de técnicas de paralelismo ou distribuição para otimizar ainda mais o processo de integração, especialmente quando lidando com grandes volumes de dados; (v) investigar estratégias adicionais de segurança e auditoria, como a implementação de criptografia nos dados durante o processo de integração, garantindo uma camada extra de proteção.

Referências

ALLAM, Zaheer; DHUNNY, Zaynah A. (2019) On big data, artificial intelligence and smart cities. Cities, v. 89, p. 80-91.

FOWLER,M. NoSQL. Disponível em: https://martinfowler.com/articles/nosql-intro.html. Acesso em: 08 maio 2023.

Galvão Filho, C. R., & da Silva, E. D. O. (2022). Migração de Bases de Dados Relacionais para Banco de Dados NoSQL Utilizando Técnicas de ETL. *Caderno de Estudos em Sistemas de Informação*, *7*(2).

HECHT, R.; JABLONSKI, S. NoSQL evaluation: A use case oriented survey. In: International Conference on Cloud Computing and Services Science (CLOSER). 2011. p. 553-562.

HECHT, R.; JABLONSKI, J. (2011) Non-relational database management systems for service-centric computing. In: 2011 International Conference on Collaboration Technologies and Systems (CTS), 2011, Philadelphia. Proceedings of the 2011 International Conference on Collaboration Technologies and Systems (CTS), p. 36-43.

Kimball, R., & Ross, M. (2013). The Data Warehouse Toolkit: The Definitive Guide to Dimensional Modeling. John Wiley & Sons.

KNIJNIK, David Mees. Comparação de SGBDs mongoDB e postgreSQL para jogos digitais. 2022.

Kokay, A. (2012). Introdução ao modelo de dados relacional e à linguagem SQL. Novatec Editora.

LÓSCIO, B. F. et al. Banco de dados NoSQL(2011): conceitos e exemplos. Revista Brasileira de Computação Aplicada, v. 3, n. 1, p. 12-24 .

LÓSCIO, B. F. et al. (2011) Um estudo sobre bancos de dados NoSQL: conceitos, características e modelagem. In: VIII Congresso Nacional de Informática (CoNI), 2011, Fortaleza. Anais do VIII Congresso Nacional de Informática (CoNI), p. 129-142.

Martins Filho, J. F. (2015). Sistemas de gerenciamento de banco de dados. In: M. A. Casanova, M. A. G. de Menezes, & R. A. de Oliveira, Informática e Sociedade: Novas Perspectivas para o Século XXI (pp. 133-150). Editora da UERJ.

Machado, F. N. L., & DE ABREU, M. M. (1996). Bancos de dados: projeto, implementação e administração. Editora LTC.

SERPRO. Intra Serpro. Disponível em: http://intra.serpro.gov.br/tema/noticias-tema/o-futuro-esta-nos-dados. Acesso em: 8 jun. 2023.

NoSQL Databases. Disponível em <http://nosql-database.org/>. Em: 1 dez 2022.