Banco de dados - estudo comparativo de desempenho entre o banco de dados relacional PostgreSQL e não relacional Redis

Filipe Ferreira Oliveira[1], **Daiane Mastrangelo Tomazeti**[1]

[1]Tecnologia em Análise e Desenvolvimento de Sistemas - Instituto Federal de São Paulo - Campus Hortolândia (IFSP-HTO)

filipe.o@aluno.ifsp.edu.br, daianetomazeti@ifsp.edu.br

Abstract. *An increase in the volume of data with the advancement of technology in people's daily lives required organizations to adopt more efficient data management. Thus, the NoSQL approach – Not Only SQL – emerged to meet this demand for data storage. The NoSQL, or non-relational, approach has divergences from the relational model and its approach to manipulating data in tables. This work presents a comparative research between a relational model database, PostgreSQL, and a non-relational key-value database, Redis, in order to analyze their differences and performances.*

Resumo. *O aumento do volume de dados com o avanço da tecnologia no cotidiano das pessoas exigia que as organizações adotassem uma gestão mais eficiente dos dados. Dessa forma, a abordagem NoSQL – Not Only SQL – surgiu para suprir essa demanda de armazenamento de dados. A abordagem NoSQL, ou não relacional, possui divergências com o modelo relacional e sua abordagem voltada à manipulação de dados em tabelas. Este trabalho apresenta uma análise comparativa entre um banco de dados de modelo relacional, PostgreSQL, e um banco de dados não relacional baseado em chave-valor, Redis, a fim de analisar suas diferenças e performances.*

1. Introdução

A cada ano, a tecnologia se torna cada vez mais presente no cotidiano das pessoas, seja por meio das redes sociais ou de sua presença nas organizações. Dessa forma, o volume de informações em circulação cresce mais e mais a cada instante, exigindo das organizações uma gestão mais estrategicamente eficiente destes dados, de modo que, sejam capazes de acompanhar a grande demanda requerida pela população. Nesse cenário, os bancos de dados se viram necessários, pois sua capacidade de armazenar

dados e resgatá-los de forma clara, era a forma ideal para lidar com essa adversidade (Pintar 2016).

Com o passar do tempo foi desenvolvido uma grande gama de modelos de bancos de dados, tendo cada um suas próprias especificações e aplicabilidades. Entre esses modelos, o que mais se destacou foi o relacional, sua simplicidade e capacidade de realizar consultas mais complexas fez com que este modelo dominasse o mercado, e sendo até hoje o mais utilizado no meio (DB-ENGINE, 2024).

Contudo o modelo relacional possui algumas limitações quando voltado para projetos de grande escalabilidade e agilidade com grande volume de dados. Os bancos de dados NoSQL (*Not Only SQL*) surgiram para atender a esta demanda de armazenamento de dados, provendo ferramentas mais eficientes para lidar com grande fluxo de informações. As bases de dados não relacionais podem ser divididas em quatro categorias, sendo elas, orientado a documentos, orientado a coluna, orientado a grafos e, por último, orientado a armazenamento em chave-valor, onde cada uma possui seu próprio conceito e suas peculiaridades. Apesar disso, é importante notar que independente do Sistema Gerenciador de Banco de Dados (SGBD) escolhido, sendo ele relacional ou não, e que por mais que possuam suas especialidades e pontos fortes, ainda sim, carregam algumas limitações consigo, entre elas a *performance* e escalabilidade (Pintar 2016).

Este trabalho tem como objetivo fazer uma análise comparativa de desempenho de um banco de dados relacional e um banco de dados não relacional, baseado em chave-valor, fornecendo dados através de operações de CRUD. Para este estudo, optou-se por utilizar o PostgreSQL como banco de dados relacional, por estar entre as três bases de dados relacionais gratuitas de maior destaque no *ranking* mundial, segundo a DB-ENGINES (2024). Para o banco de dados não relacional, optou-se pelo Redis, por possuir uma versão gratuita e ser bastante conhecido pela comunidade por sua velocidade, visto que sua estrutura chave-valor o permite realizar operações de forma mais dinâmica, além de ser o banco de dados de chave-valor de maior destaque segundo o *ranking* mundial, também segundo a DB-ENGINES (2024). Uma das contribuições deste artigo é apresentar como diferentes abordagens de armazenamento impactam o desempenho, a escalabilidade e a eficiência de aplicações com diferentes requisitos de processamento de dados.

2. Referencial Teórico

Nesta seção são apresentados os conceitos básicos necessários para a compreensão dos temas abordados, além de entender como eles se relacionam com o estudo apresentado.

2.1. Banco de Dados Relacional

De acordo com Date (2004), “um sistema de banco de dados pode ser definido como um sistema computadorizado cuja finalidade é armazenar informações e permitir que os usuários busquem e atualizem essas informações quando as solicitar.”

O modelo relacional, proposto por Edgar Frank “Ted” Codd, matemático e pesquisador da IBM (*International Business Machines*), em 1970, se baseia em conceitos matemáticos conhecidos como teoria dos conjuntos e lógica de predicado (Takai et al., 2005). Devido à necessidade de aumentar a independência de dados nos sistemas de gestão de Banco de Dados, os primeiros sistemas comerciais do modelo relacional foram disponibilizados em 1980 e, até nos dias de hoje, seguem sendo implementados em diversos sistemas atuais (Elmasri et al., 2005).

A estrutura do modelo relacional se caracteriza pelo registro dos dados em tabelas, utilizando de linhas e colunas. Com isso, de acordo com Tavares (2012), o modelo relacional se basearia nos conceitos de Entidade e Relação. Dessa forma, uma entidade, conhecida como tabela, é formada por atributos que traduzem o tipo de dado que será armazenado e que será organizado em registros (Takai, et al., 2005). Outro conceito fundamental do modelo relacional é a restrição de integridade, associada ao uso de chaves primárias e estrangeiras.

De acordo com Brito (2010, p. 1)

> [...] As chaves representam uma forma simples e eficaz de associação entre as tabelas do banco de dados. A chave primária foi criada com o objetivo de identificar de forma única as tuplas da tabela e ainda de determinar a ordem física dessas tuplas. A chave estrangeira permite uma relação de dependência entre atributos de tabelas distintas, de forma que os valores permitidos em um atributo dependem dos valores existentes em outro atributo.

Com as funcionalidades descritas e recursos como a garantia de integridade dos dados, controle e coerência, recuperação de falhas e seguranças de dados, foram essenciais para manter os sistemas gerenciadores de banco de dados relacionais em destaque em diversos ambientes organizacionais (Brito, 2010, p. 2).

2.1.1. Limitações do Modelo Relacional

Nos últimos tempos, com o crescimento do volume de dados, juntamente de sua complexidade, fez com que o modelo relacional demonstrasse problemas em suprir a demanda de escalabilidade mais frequentemente (Bhatia, 2019, p. 443).

Os principais problemas encontrados no modelo relacional ao ser utilizado era conciliar a base de dados com a demanda por escalabilidade (Brito, 2010, p. 2). Com isso, uma das soluções encontradas foi a de distribuir o banco de dados em várias

máquinas, porém, dessa forma, um outro problema começou a surgir, pois há um aumento na complexidade geral da base de dados e de sua gestão quando está distribuída em várias instâncias (Souza; Santana, 2017, p. 17).

Outros fatores limitantes também surgiram quando os bancos de dados relacionais começaram a lidar com grandes volumes de informação.

Segundo Souza, Santana (2017, p. 17):

> As propriedades ACID [Atomicidade, Consistência, Isolamento e Durabilidade] de uma base de dados relacional obrigam que tenham uma distribuição e não isolamento de dados.
> Escalabilidade: as bases de dados escalam, porém não escalam facilmente. É possível escalar vertical e/ou horizontalmente, mas a complexidade exigida na gestão dos múltiplos nós da base de dados, ou o custo do hardware para suportar a distribuição, são entraves.

2.1.2. SQL

A "Linguagem de Consulta Estruturada", SQL (ou, *Structured Query Language*), foi desenvolvida pela IBM ao comissionar um grupo para construir um protótipo das ideias de Ted Codd, que propôs uma linguagem para manipular os dados em tabelas relacionais (Beaulieu, 2010).

Brito (2010, p.1) afirma que "Sua simplicidade e alto poder de expressão fizeram do SQL a linguagem de consulta de dados mais utilizada no mundo e ajudou a consolidar a posição dominante do modelo relacional."

As instruções de definição, consulta e atualização dos dados na SQL podem ser classificados em duas categorias: Linguagem de Definição de Dados (DDL) e Linguagem de Manipulação de Dados (DML). Dessa forma, as consultas DDL são usadas para criar e definir esquemas de bancos de dados. Enquanto as consultas DML, são usadas para manipular os dados dentro dos bancos de dados (Prescott, 2015).

2.1.3. PostgreSQL

O PostgreSQL é um sistema gerenciador de banco de dados relacional de código aberto, o seu desenvolvimento teve início no ano de 1986, na Universidade Berkeley, na Califórnia (EUA), como resultado do projeto conhecido como POSTGRES, cujo objetivo era resolver os problemas que existiam no modelo relacional de banco de dados. (PostgreSQL, 2024)

Conhecido por possuir uma arquitetura comprovada, confiabilidade, integridade de dados e conjunto robusto de recursos, o PostgreSQL também oferece suporte à ACID desde 2001 e possui ótimos complementos como o PostGIS, utilizado para dados geoespaciais. Com isso, o PostgreSQL é considerado por muitos como a tecnologia de banco de dados, gratuita, favorita dos desenvolvedores, ficando atrás apenas do MySQL.

(Microsoft Azure, 2024).

2.2. Banco de Dados NoSQL

Em 1998, o termo NoSQL (*Not Only SQL*) surgiu em meio a uma solução de banco de dados que não oferecia uma interface SQL, mas que ainda se baseava na arquitetura relacional (Brito, 2010). Em seguida, o termo começou a representar soluções que buscavam atingir um processamento de dados de forma rápida e eficiente sem ter que seguir os padrões rígidos propostos pelo modelo relacional. (Aniceto; Xavier, 2014, p. 17).

Para atender à demanda de grandes volumes de dados distribuídos, às estruturas NoSQL possuem algumas características em comum que, de acordo com Aniceto, Xavier (2014, p.17), são:

> - **Prover uma grande escalabilidade horizontal.** Essa característica consiste em cada aplicação ser capaz de aumentar seu número de nós com a camada IaaS (Infraestrutura como serviço), ou seja, é a grande capacidade de requisição de mais recursos de armazenamento e processamento.
> - **Prover uma baixa latência.** Essa característica é obtida por meio da escalabilidade horizontal que se caracteriza pela otimização do tempo de um grande processamento dividindo ele em vários pequenos processos que são distribuídos em diferentes máquinas, assim o tempo de resposta diminui para uma ordem de alguns poucos milissegundos.
> - **Prover grande disponibilidade de acesso.** Essa característica é obtida por meio do grande processamento que permite um acesso de um número de usuário maior que os gerenciadores de bancos de dados relacionais.
> - **Prover um processamento de grandes dados**, seja uma grande quantidade de ciclos de leitura e escrita ou uma quantidade massiva de acessos de usuários.

Por fim, apesar de certas características em comum, os bancos de dados NoSQL se diferenciam de acordo com o modelo de dados utilizado, onde cada um possui suas próprias singularidades e, diferente do modelo relacional, os NoSQL podem ser organizados em quatro categorias: Baseado em Chave-Valor; Baseado em Documentos; Baseado em Coluna; Baseado em Grafos.

2.2.1. Modelo Chave-Valor

Segundo Aniceto, Xavier (2014, p.18) o modelo baseado em chave-valor foi o precursor do movimento NoSQL, pois foi a partir deste modelo que surgiram os conhecidos *Amazon DynamoDB* e o *Google BigTable*, que em 2004 foi analisado e responsável por fazer os bancos de dados não relacionais serem vistos como uma alternativa aos modelos SQL.

Pintar (2016, p.5) afirma que “O sistema chave-valor (*key-value*) é considerado

simples e permite que nós acessemos os dados através de uma tabela de *hash*, na qual existe uma chave e um valor para determinado dado". Além disso, este modelo se caracteriza por sua facilidade de ser implementado, permitindo que os dados sejam acessados rapidamente, além de permitir que novos valores possam ser inseridos a qualquer momento, sem que haja conflitos com os dados já armazenados (Aniceto; Xavier, 2014, p. 18).

2.2.2. Limitações do NoSQL

Por mais flexíveis e poderosos que os bancos de dados NoSQL sejam, ainda assim, estas estruturas possuem algumas limitações. De acordo com Aniceto, Xavier (2014, p.18), em 2000 o professor Eric Brewer propôs o teorema CAP, que demonstra as limitações dos sistemas escaláveis, consequentemente englobando os NoSQL. Para Aniceto, Xavier (2014, p.18):

> As três principais características segundo o teorema CAP são: consistência dos dados (*Consistency*), disponibilidade (*Availability*) e tolerância ao particionamento (*Partition Tolerance*).
>
> - Consistência: propriedade que dita como e quando o sistema está em uma versão consistente após a execução de uma operação. Um sistema distribuído é considerado consistente se todos os usuários leitores têm a mesma visão de uma atualização feita por um usuário que escreve no sistema, logo após essa atualização ser efetuada.
> - Disponibilidade: propriedade na qual um sistema é projetado e desenvolvido que permita ao usuário ler e escrever a qualquer momento no banco de dados.
> - Tolerância ao particionamento: característica em que um sistema pode executar corretamente, apesar da divisão física da rede. Também pode ser compreendido como a habilidade de um sistema de lidar a adição e remoção dinâmica de nós de recursos. Entende-se que um sistema com alta tolerância ao particionamento é um sistema altamente escalável.

Dessa forma, o teorema propõe que em um sistema escalável só é possível garantir apenas duas das três características, pois elas tendem a se anular. Assim, é possível concluir que ao ter disponibilidade e tolerância ao particionamento é impossível manter consistência, pois um usuário pode escrever e ler a qualquer momento, mas o banco de dados não conseguiria estar sempre conectado à versão do usuário (Aniceto; Xavier, 2014, p. 18).

2.2.3. Redis

Redis é um banco de dados NoSQL que armazena a estrutura de dados em formato

chave-valor na memória. E segundo DB-Engines (2024) é o armazenamento de chave-valor de código aberto mais conhecido atualmente.

Desenvolvido em 2009 por Salvatore Sanfilippo, na Itália, com o objetivo de atender a uma demanda interna de problemas de *performance* na análise de *logs* em tempo real de sua própria empresa. Com o sucesso do uso de sua ferramenta, Salvatore decidiu tornar Redis *open-source*, e a partir desse momento, a estrutura ganhou popularidade e sendo usada por grandes empresas, como *IFood*, *Niantic* e entre outras (Redis, 2024).

Por armazenar os dados em memória, o Redis, proporciona um armazenamento e um processamento mais dinâmico e ágil visto que, o banco de dados, não precisa acessar o disco para consultar ou gravar dados. Ele disponibiliza desde estruturas de dados simples, como *Lists*, *Sets* ou *Strings*, até estruturas mais complexas como *Hyperloglog* (estima a cardinalidade de um conjunto), *Bitmaps* (armazena estados lógicos e binários) e até índices geoespaciais (Redis, 2024).

Por fim, para evitar a perda dos dados caso o servidor seja desligado, o Redis possibilita a criação de *Snapshots*, cópias feitas de todo o conteúdo presente em memória em intervalos de tempo pré-determinados, além de permitir gravar estes dados em disco, em um formato compacto e otimizado (Redis, 2024).

3. Trabalhos Correlatos

Para o levantamento de informações de trabalhos correlatos foram utilizadas as plataformas BDTD (Biblioteca Digital Brasileira de Teses e Dissertações) e Google Acadêmico, com a pesquisa sendo feita a partir das palavras chaves: banco de dados relacionais, banco de dados não relacionais e análise de desempenho.

No trabalho “Análise de desempenho de banco de dados NoSQL em um sistema que utiliza um banco de dados relacional e não relacional para o armazenamento de dados”, é abordada a implementação de um banco de dados não relacional orientado a documentos, MongoDB, e de um banco de dados relacional, PostgreSQL, a fim de realizar testes de *performance* e comparar seus resultados em um ambiente de tecnologia de uma empresa contábil. Para avaliar o tempo de resposta de cada banco, os autores [Rabelo e Cândido 2017] executaram testes de cargas nas funcionalidades de importar e de visualizar planos de contas, além da importação de balancetes (documentos que mostram o saldo de todas as contas de uma empresa em determinado período de tempo) e da geração de relatórios. Ao fim, foram realizadas três rodadas de testes em ambas bases de dados, onde o primeiro teste consistia em ler um arquivo de 1674 registros, o segundo 10.000 registros e o último 100.000 registros. Como resultado, o tempo de

resposta do banco de dados não relacional, MongoDB, obteve um desempenho superior em todos os cenários propostos.

No documento "Comparação de performance entre um banco de dados relacional e dois bancos não-convencionais do tipo NoSQL", o objetivo proposto pelos autores [Costa e Castro 2021] é realizar uma pesquisa comparativa entre as tecnologias de bancos de dados relacional e não relacional com problemas tipicamente inseridos dentro do contexto relacional. Para a análise foram utilizados o PostgreSQL como banco de dados relacional, o MongoDB como banco de dados não relacional baseado em documentos e, por fim, o Apache Cassandra como banco de dados não relacional baseado em colunas. Como parte da metodologia, os autores utilizaram *Eclipse Oxygen 3*, uma *IDE* (Ambiente de Desenvolvimento Integrado) para desenvolvimento de aplicações Java, onde criaram um projeto de conexão para cada banco escolhido e utilizaram instrumentos da linguagem Java para medir o tempo de execução das requisições propostas. Para os testes, em cada banco, foram utilizadas as quatro operações de CRUD (*Create, Read, Update e Delete*) em duas situações, onde na primeira foram empregados 10 entradas nos bancos e na segunda utilizaram 50 entradas. Como resultado, os autores destacam que MongoDB teve uma *performance* inferior às outras bases de dados, enquanto o banco de dados relacional, PostgreSQL, se manteve na média em todas as operações propostas, demonstrando ser a melhor opção para aplicações que utilizam a linguagem Java. E, por fim, o banco de dados Cassandra obteve resultados excelentes nos testes de leitura e atualização, se mostrando uma melhor solução para aplicações que façam uso de muitas consultas e modificações de dados.

Conforme mostrado nos estudos citados, os autores demonstram terem realizado a pesquisa de *performance* em bancos de dados não relacionais, majoritariamente, nos modelos orientados a documentos, apresentando como esses modelos lidaram com as situações propostas e quão proveitosos os resultados obtidos foram em comparação com os bancos de dados relacionais. É importante ressaltar que o presente trabalho tem como objetivo realizar uma pesquisa comparativa entre as bases de dados PostgreSQL e Redis e compreender como estas estruturas podem lidar com as situações propostas, tendo em vista que o PostgreSQL é amplamente utilizado para o armazenamento estruturado e com consultas complexas, enquanto o Redis é conhecido por sua velocidade e o uso do banco de dados em memória.

4. Metodologia

Primeiramente, foi realizada uma revisão bibliográfica sobre banco de dados e seus

modelos relacionais e não relacionais. Em seguida, foram escolhidos os bancos de dados que serão utilizados neste estudo, sendo o PostgreSQL, para representar o banco de dados relacional, e o Redis, para representar o banco de dados não relacional baseado em chave-valor. Foram escolhidos estas bases de dados, pois o PostgreSQL está entre as três bases de dados relacionais gratuitas de maior destaque no *ranking* mundial, já o Redis é o banco de dados de chave-valor de maior destaque segundo o *ranking* mundial, ambos os dados segundo a DB-ENGINES (2024). Além disso, o Redis possui uma versão gratuita e, por ser um modelo de estrutura chave-valor, funciona baseado na memória, assim oferecendo uma alta velocidade de resposta e propriedades que favorecem operações rápidas de leitura e escrita (Redis, 2024).

Para a realização do experimento foram criadas duas bases de dados similares, uma para cada SGBD, utilizando as versões do PostgreSQL 17 para o banco relacional, e Redis Insight 7.4.0 com scripts em JSON para o banco não relacional.

Com isso, a Figura 1 ilustra a metodologia utilizada neste trabalho.

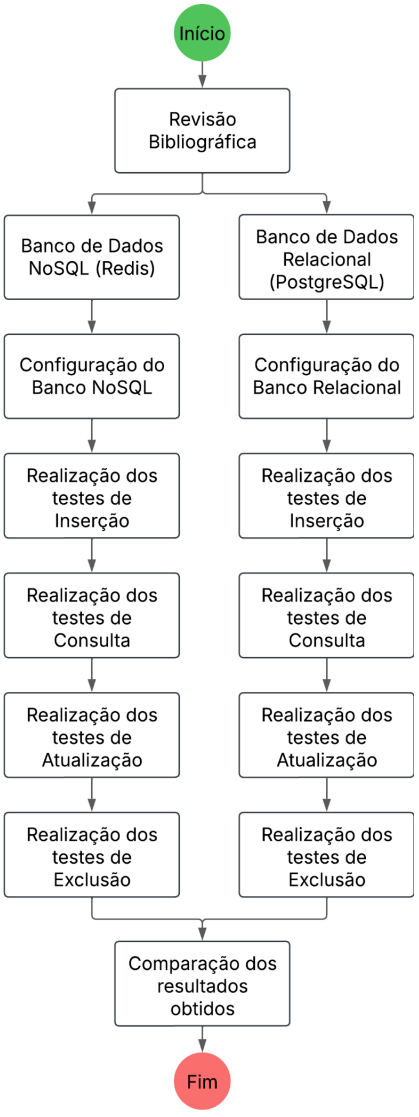


Figura 1. Protocolo de desenvolvimento do trabalho

Vale ressaltar que, Redis Insight é uma interface gráfica para o uso do banco de dados Redis, que proporciona funcionalidades como uma interface de linha de comandos integrada (CLI) e recursos para testes de desempenho como o Redis Workbench. Para o uso do Redis, é necessário acessar seu website, onde é necessário criar um registro e selecionar um plano apropriado que supra as necessidades demandadas. Para este trabalho foi utilizado o plano gratuito do Redis que possui 30MB de memória disponível, limitando seu potencial, porém, ainda assim, adequado para estudos de caso referentes a esse banco.

5. Desenvolvimento

A fim de proceder com a pesquisa comparativa, os testes estão divididos em quatro categorias: Inserção, Consulta, Atualização e Exclusão. Essas categorias correspondem às quatro operações de CRUD *(Create (e Insert), Read, Update e Delete)*. Cada categoria apresenta três experimentos envolvendo quantidades crescentes de registros, com o cálculo do tempo de respostas das pesquisas realizadas, utilizando ferramentas nativas de cada aplicação que demonstram o intervalo de execução das consultas propostas.

Para realizar os testes no banco de dados relacional, foi utilizada a estrutura de tabela exibida na Figura 2.

```
                    Table "public.cliente"
      Column       |         Type          | Collation | Nullable | Default
-------------------+-----------------------+-----------+----------+---------
 id_cliente        | integer               |           | not null |
 nome_cliente      | character varying(50) |           | not null |
 sobrenome_cliente | character varying(50) |           | not null |
 email_cliente     | character varying(50) |           | not null |
 genero_cliente    | character varying(50) |           | not null |
Indexes:
    "cliente_pkey" PRIMARY KEY, btree (id_cliente)
```

Figura 2. Tabela cliente criada para realizar os testes

Como a base de dados não relacional Redis não possui suporte ao SQL, foi utilizada a estrutura para a inserção de dados mostrada na Figura 3.

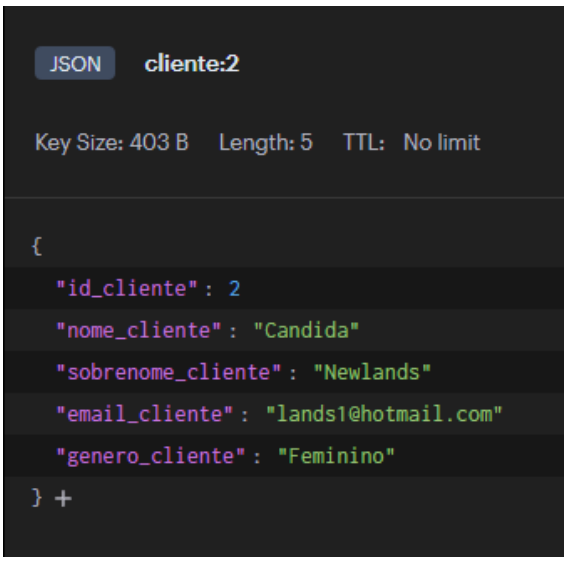


Figura 3. Estrutura de chave-valor no Redis

Na primeira categoria os bancos devem inserir, para o primeiro teste, 100 registros no banco, para o segundo teste, 1.000 registros no banco, e por fim, para o terceiro teste, 10.000 registros. Para executar os testes de inserção no PostgreSQL e no Redis foram utilizados os comandos presentes na Tabela 1.

Tabela 1. Exemplo de inserção no PostgreSQL e no Redis

Inserção (Insert)	
PostgreSQL	INSERT INTO cliente (id_cliente, nome_cliente, sobrenome_cliente, email_cliente, genero_cliente) values (2, 'Candida', 'Newlands', 'lands1@hotmail.com', 'Feminino');
Redis	JSON.SET cliente:2 $ '{"id_cliente":2,"nome_cliente":"Candida","sobrenome_cliente":"Newlands", "email_cliente":"lands1@hotmail.com","genero_cliente":"Feminino"}'

Na segunda categoria, os bancos de dados devem realizar uma consulta ordenada nos testes registrados anteriormente, seguindo suas próprias regras de consulta.

Para realizar as consultas nas bases de dados foram utilizadas operações de consultas com foco em apresentar os dados cadastrais dos clientes de uma empresa X. A Tabela 2 mostra os comandos utilizados para realizar as operações de busca no PostgreSQL.

Tabela 2. Comandos de consulta de dados utilizados no PostgreSQL

POSTGRESQL	
100 registros	SELECT * FROM cliente ORDER BY nome_cliente ASC LIMIT 100;
1.000 registros	SELECT * FROM cliente ORDER BY nome_cliente ASC LIMIT 1000;
10.000 registros	SELECT * FROM cliente ORDER BY nome_cliente ASC LIMIT 10000;

Porém, diferente dos bancos de dados relacionais que utilizam a linguagem SQL,

o Redis não possui suporte a consultas SQL, dessa forma é necessário usar de outros métodos para efetuar suas operações de consulta. Para este trabalho foi aplicada a linguagem JSON para desenvolver os procedimentos de pesquisa, pois este módulo provê uma manipulação eficiente na memória, além de disponibilizar diversos comandos para efetuar as consultas. Através do comando *FT.CREATE idx_cliente ON JSON PREFIX 1 "cliente:" SCHEMA $.nome_cliente AS nome_cliente TEXT SORTABLE* foi criado um índice (*index*) para organizar as junções dos dados dentro do banco e servir de base para realizar as pesquisas no banco. Na Tabela 3 é possível visualizar os comandos utilizados para realizar as operações de busca no Redis.

Tabela 3. Comandos de consulta de dados utilizados no Redis

REDIS	
100 registros	FT.SEARCH idx_cliente "*" SORTBY nome_cliente ASC LIMIT 0 100
1.000 registros	FT.SEARCH idx_cliente "*" SORTBY nome_cliente ASC LIMIT 0 1000
10.000 registros	FT.SEARCH idx_cliente "*" SORTBY nome_cliente ASC LIMIT 0 10000

Na terceira categoria de testes, as bases de dados devem atualizar os dados que estão inseridos nelas. Nestes experimentos, ambos os bancos de dados devem atualizar o domínio do e-mail dos clientes para '@gmail.com'. Para executar esses testes foram usados os comandos no PostgreSQL e no Redis mostrados na Tabela 4.

Tabela 4. Comandos de atualização de dados utilizados no PostgreSQL e Redis

Atualização (Update)	
PostgreSQL	UPDATE usuarios SET email = REGEXP_REPLACE(email, '@.*$', '@gmail.com');
Redis	JSON.MERGE cliente:2 $.email_cliente '"lands1@gmail.com"'

Por fim, para a quarta categoria as bases de dados devem excluir os dados inseridos anteriormente. Para isso, a Tabela 5 demonstra os comandos utilizados para excluir os dados dos bancos de dados.

Tabela 5. Comandos de exclusão de dados no PostgreSQL e Redis

Exclusão (Delete)	
PostgreSQL	TRUNCATE cliente;
Redis	FLUSHDB

6. Resultados Obtidos

Nesta seção, são apresentados os resultados obtidos nas pesquisas realizadas em cada

modelo de banco de dados.

6.1. Configuração de hardware

Os testes foram realizados em um computador com um processador Intel Core i5-7400CPU; 16GB de memória RAM; SSD 465GB; Windows 10 como Sistema Operacional. Para ambas as instâncias de banco de dados foram utilizados ambientes similares de execução.

6.2. Resultados e análise comparativa

Como apresentado, os testes foram divididos em quatro categorias e as figuras e gráficos abaixo mostram os resultados obtidos nas experiências realizadas nos bancos de dados.

6.2.1. Resultados dos testes de inserção (*insert*)

A Figura 4 mostra os resultados, em milissegundos (ms), adquiridos nos testes de inserção com 100 registros nos bancos de dados.

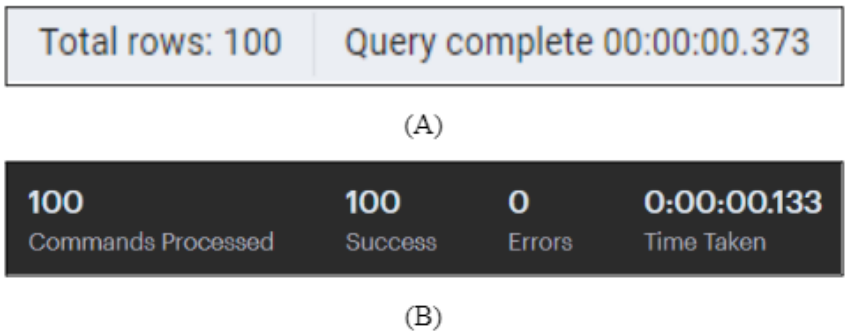


Figura 4. Resultado do teste de inserção com 100 registros (A) no PostgreSQL e (B) no Redis

A Figura 5 exibe o tempo de operação, em milissegundos (ms), ao inserir 1.000 dados nos bancos de dados.

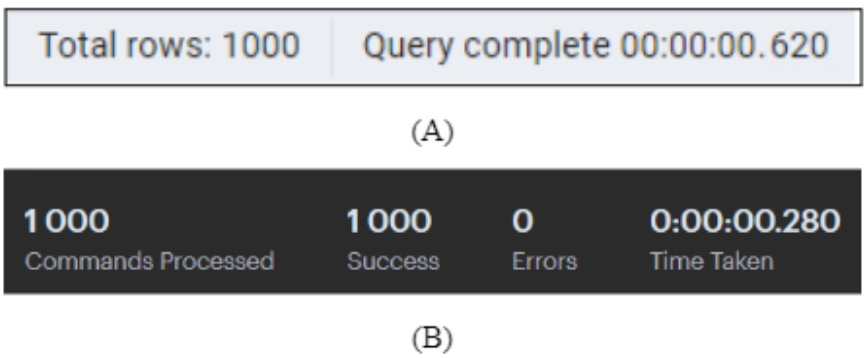


Figura 5. Resultado do teste de inserção com 1.000 registros (A) no PostgreSQL e (B) no Redis

A Figura 6 apresenta os resultados dos testes, em milissegundos (ms) com 10.000 dados inseridos.

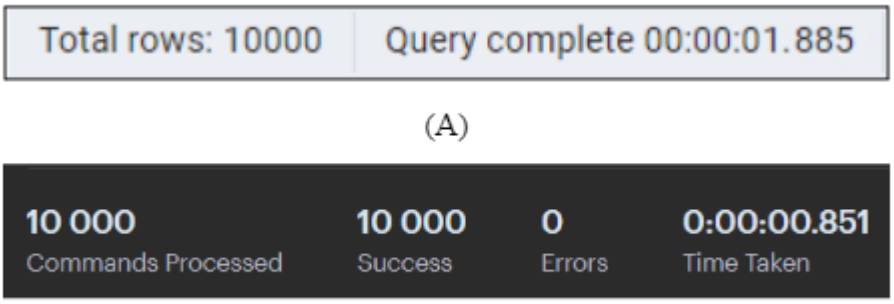


Figura 6. Resultado do teste de inserção com 10.000 registros (A) no PostgreSQL e (B) no Redis

Por fim, a Figura 7 exibe os resultados de todas as operações de inserções realizadas, e compara os tempos médios de funcionamento entre os bancos de dados utilizados.

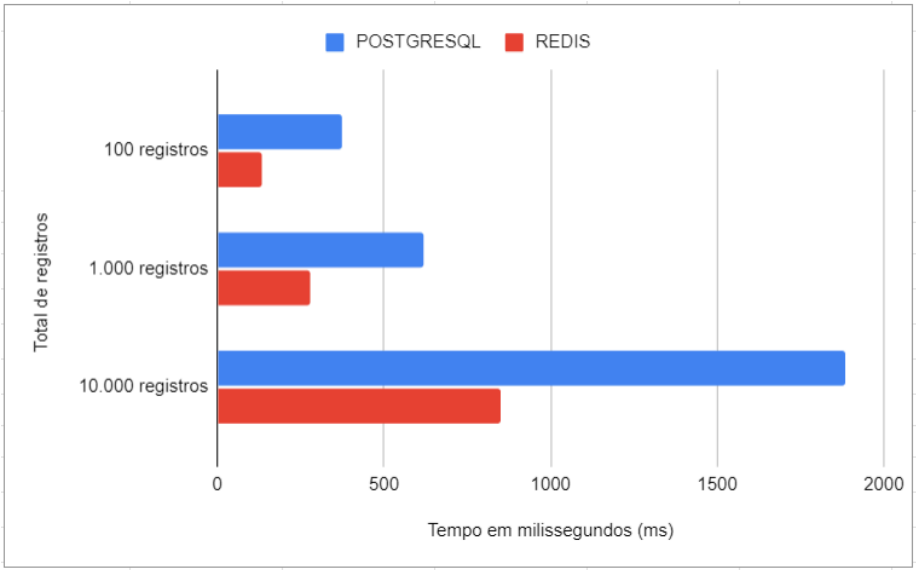


Figura 7. Resultados das operações de inserções em milissegundos

Após a execução dos testes de inserção constatou-se que o Redis obteve um desempenho superior ao PostgreSQL, visto que o segundo consumiu aproximadamente o dobro de tempo de resposta que o banco de dados relacional.

6.2.2. Resultados dos testes de consulta (*read*)

Para a segunda categoria de avaliação, foram feitos testes de consultas para retornar os dados dos clientes mantidos nos bancos, a Figura 8 mostra os resultados obtidos dos experimentos realizados com a consulta de 100 dados nos bancos de dados, além de dispor os tempos de operação calculados em milissegundos (ms).

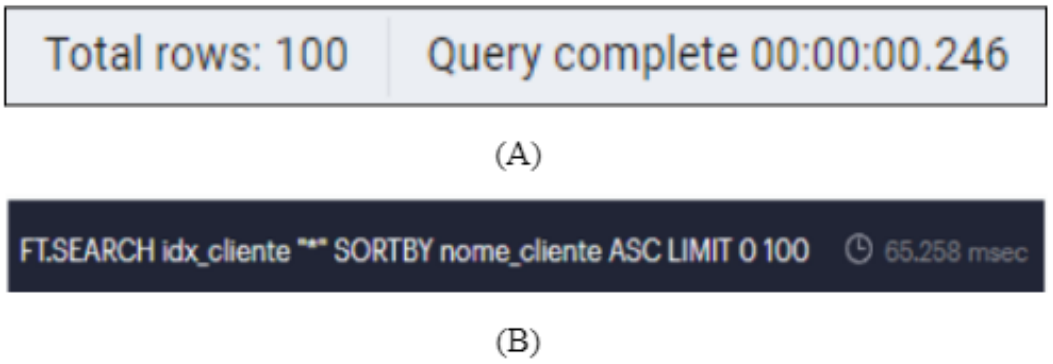


Figura 8. Resultado do teste de consulta com 100 registros (A) no PostgreSQL e (B) no Redis

O tempo médio das operações de consulta com 1.000 entradas obtidas estão presentes na Figura 9.

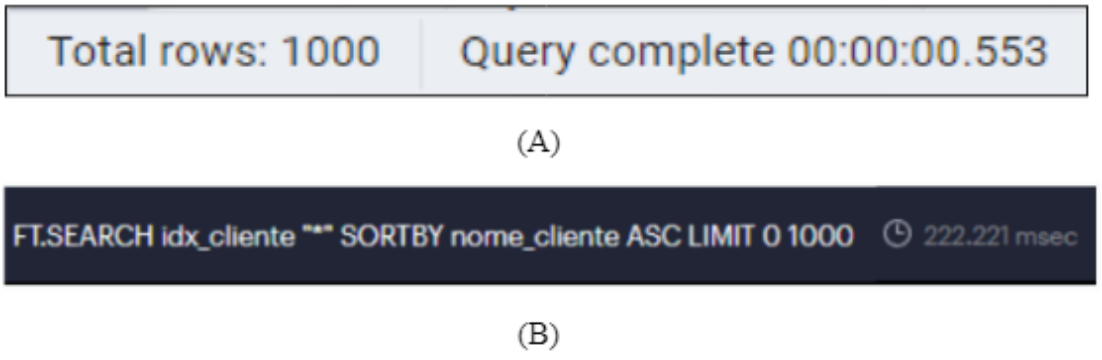


Figura 9. Resultado do teste de consulta com 1.000 registros (A) no PostgreSQL e (B) no Redis

Na Figura 10 é possível verificar o tempo de operação das consultas realizadas com 10.000 entradas nos bancos de dados.

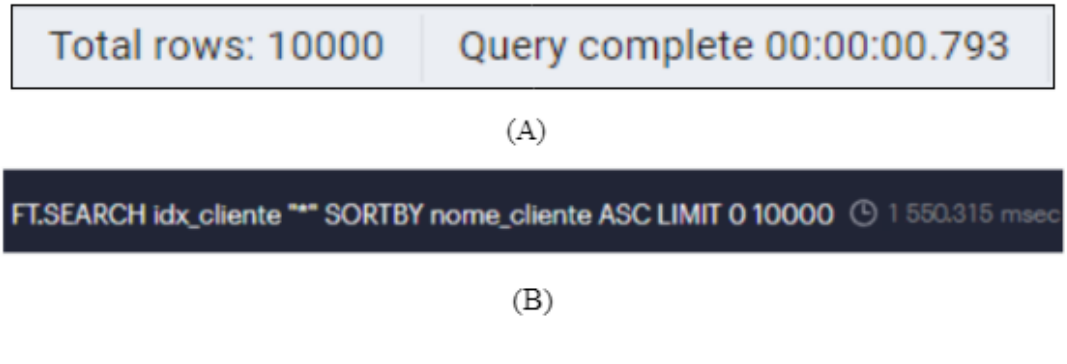


Figura 10. Resultado do teste de consulta com 10.000 registros (A) no PostgreSQL e (B) no Redis

Por fim, na Figura 11 é possível ver um gráfico de comparação dos resultados obtidos nos testes de consulta realizados no PostgreSQL e no Redis.

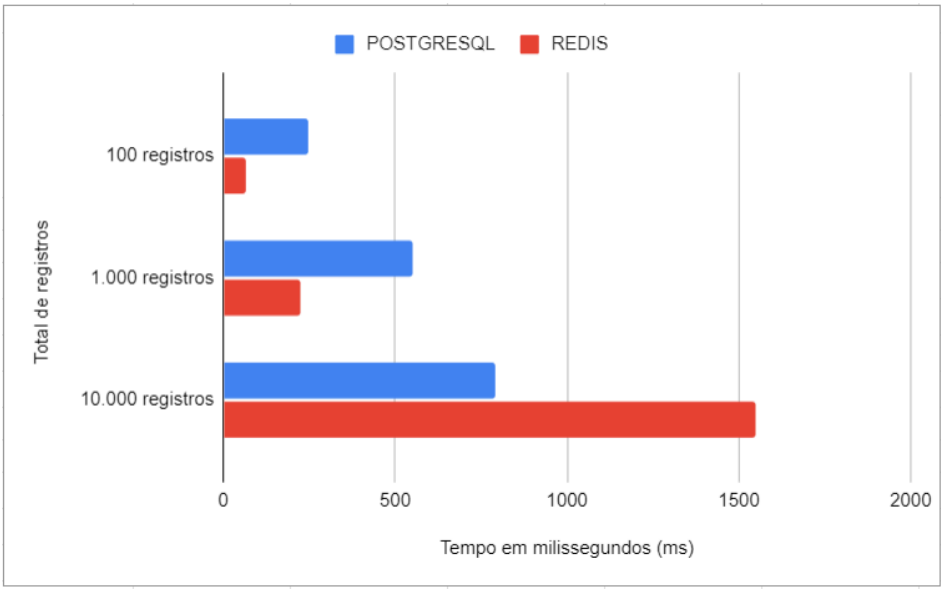
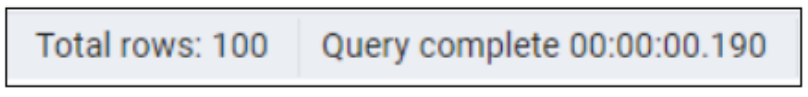


Figura 11. Resultados das operações de consulta em milissegundos

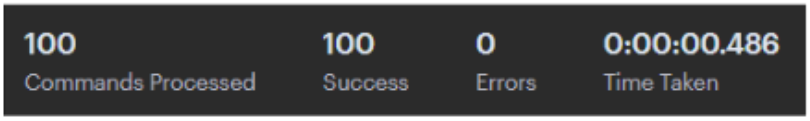
Após a execução destes testes é possível perceber que, a princípio, o banco de dados Redis obteve um desempenho melhor que o PostgreSQL, porém para o experimento contendo 10.000 registros, sua performance acabou sendo inferior à do seu concorrente. Esta queda de desempenho na última consulta se dá à limitação da memória fornecida pela versão gratuita do Redis.

6.2.3. Resultados dos testes de atualização (*update*)

Para os testes de atualização, os bancos de dados atualizaram os domínios dos e-mails dos clientes para '@gmail.com'. Com isso, a Figura 12 demonstra o tempo de execução das operações de atualização propostas.



(A)



(B)

Figura 12. Resultado do teste de atualização com 100 registros (A) no PostgreSQL e (B) no Redis

A Figura 13 demonstra o tempo de execução dos testes de atualização com 1.000 entradas de dados.

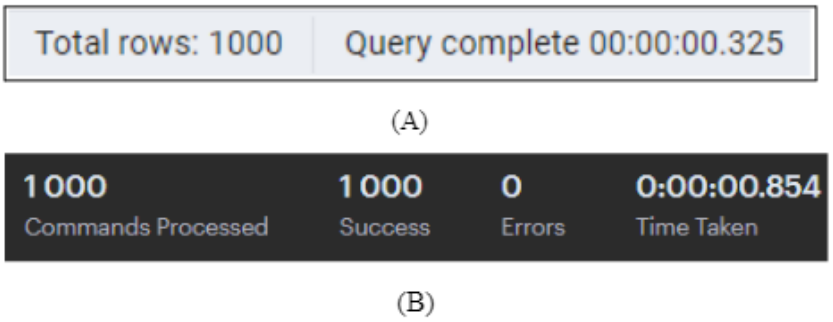


Figura 13. Resultado do teste de atualização com 1.000 registros (A) no PostgreSQL e (B) no Redis

Com isso, a Figura 14 exibe os resultados de execução dos testes de atualização com 10.000 registros.

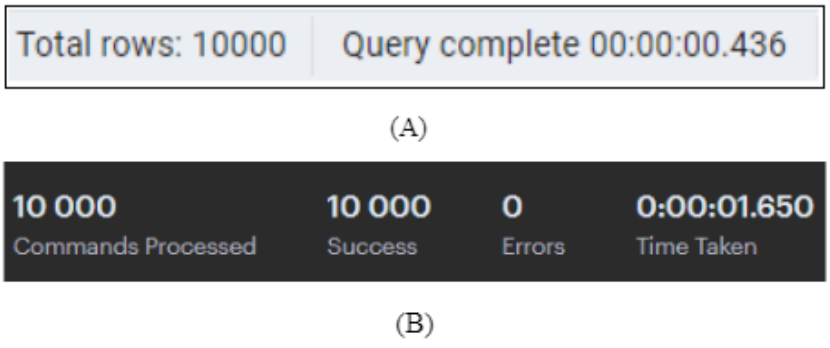


Figura 14. Resultado do teste de atualização com 10.000 registros (A) no PostgreSQL e (B) no Redis

Por fim, a Figura 15 exibe os resultados dos dois bancos de dados em relação aos testes de atualização, comparando os resultados.

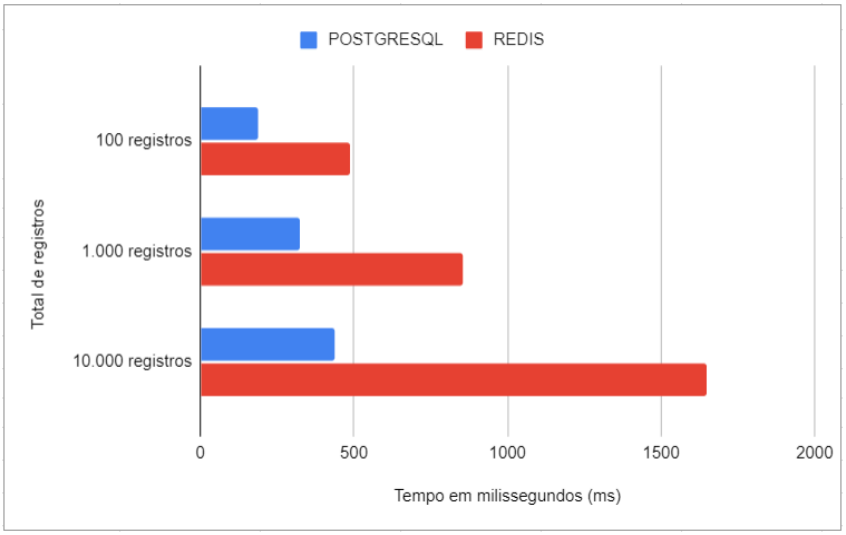


Figura 15. Resultados das operações de atualização em milissegundos

É possível perceber que para os testes de atualização o Redis obteve uma *performance* bem inferior comparado ao PostgreSQL, levando praticamente o dobro de tempo de execução das operações. O motivo dessa baixa resposta se dá pelo fato do Redis ser limitado ao realizar consultas com grande grau de complexidade, principalmente levando em conta sua estrutura chave-valor onde o valor mantém todos os dados inseridos.

6.2.4. Resultados dos testes de exclusão (*delete*)

Por último, foram realizados testes de exclusão dos dados inseridos nos bancos. A Figura 16 mostra o tempo de operação que os bancos de dados levaram para excluir seus dados.

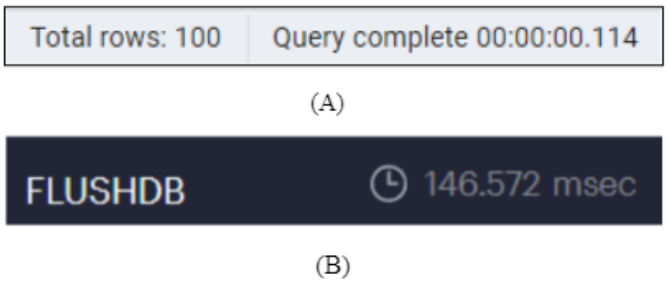


Figura 16. Resultado do teste de exclusão com 100 registros (A) no PostgreSQL e (B) no Redis

Na Figura 17 são apresentados os resultados dos testes de exclusão com 1.000 entradas nas bases de dados.

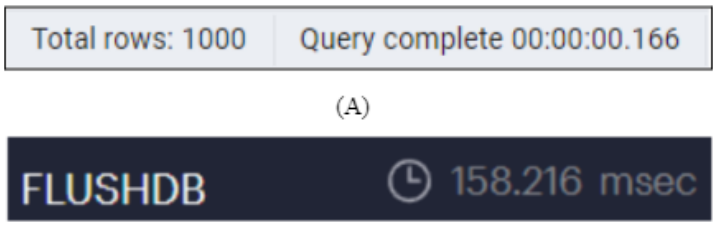


Figura 17. Resultado do teste de exclusão com 1.000 registros (A) no PostgreSQL e (B) no Redis

A Figura 18 exibe os resultados das operações de exclusão com 10.000 registros nos bancos de dados.

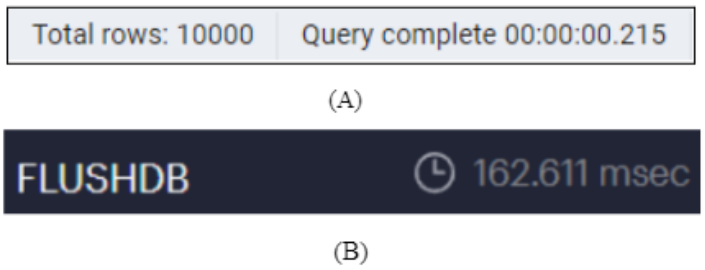


Figura 18. Resultado do teste de exclusão com 10.000 registros (A) no PostgreSQL e (B) no Redis

Por fim, a Figura 19 apresenta um gráfico comparando os resultados dos testes de exclusão realizados no PostgreSQL e no Redis.

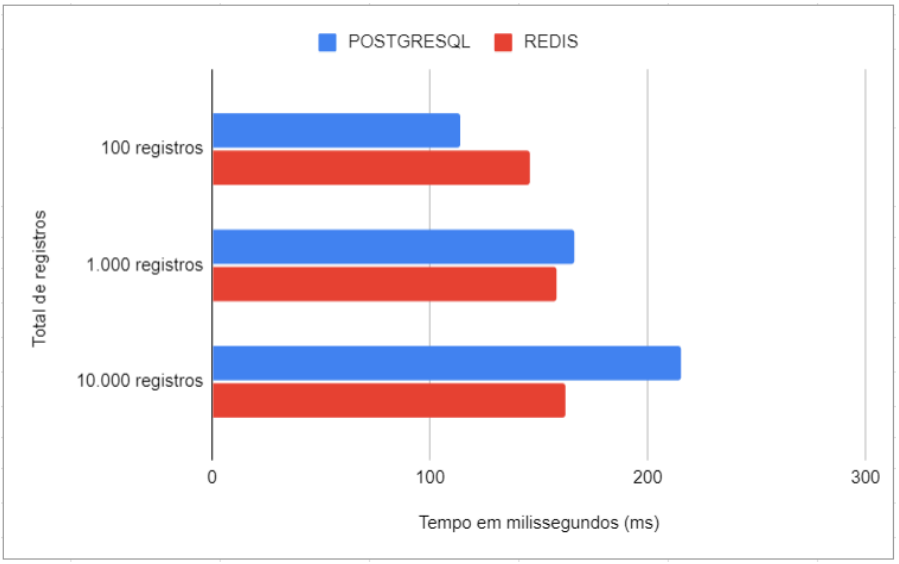


Figura 19. Resultados das operações de exclusão em milissegundos

Com os resultados apresentados, é possível perceber que o PostgreSQL obtém um

resultado superior ao Redis ao excluir 100 registros, porém para os testes de exclusão com 1.000 e 10.000 registros, o Redis obteve um desempenho melhor que o PostgreSQL.

7. Conclusão

O objetivo deste trabalho foi realizar uma pesquisa comparativa entre os bancos de dados PostgreSQL e Redis, analisando o tempo de resposta que estas bases levaram para efetuar operações do CRUD. Foram aplicados e integrados conhecimentos adquiridos nas disciplinas de Banco de Dados, Engenharia de *Software*, Estruturas de Dados, Metodologia Científica e Projetos de Sistemas I e II .

Os resultados obtidos nos experimentos realizados demonstraram que em três das categorias de testes propostas, o banco de dados não relacional, Redis, obteve uma *performance* superior ao desempenho apresentado pelo banco de dados relacional, PostgreSQL. Por outro lado, o Redis obteve uma *performance* inferior ao PostgreSQL ao realizar os testes de atualização de dados, esse resultado se dá ao fato que os bancos de dados de chave-valor possuem uma estrutura mais simples, onde toda a informação armazenada da chave inserida estará ligada a um único valor, dessa forma, fazendo com que para realizar consultas mais complexas, o banco tem que passar por um processo de obter os dados completos da chave escolhida, executar o processamento necessário e, por fim, armazenar os dados completos novamente. Assim, criando um processo mais demorado de execução e causando um problema de desempenho.

Entretanto, o PostgreSQL e o Redis possuem uma estrutura e uma modelagem de dados distintas, visto que o primeiro segue o modelo relacional que permite consultas complexas e relacionamentos entre tabelas, enquanto a segunda base de dados segue no modelo chave-valor que oferece estruturas de dados específicas para determinadas operações em memória. Dessa maneira, houve uma limitação na definição de parâmetros de testes e na adaptação do mesmo conjunto de dados para ambas as tecnologias.

Em suma, o Redis se manteve consistente em grande partes dos experimentos realizados, o tornando uma escolha valiosa para usuários que desejam otimizar o desempenho em suas estruturas de sistemas, com ressalvas a operações de dados mais complexas. Enquanto o PostgreSQL se torna viável ao precisar realizar consultas mais robustas

8. Referências Bibliográficas

ANICETO, R. C.; XAVIER, R. F. **Um estudo sobre a utilização do banco de dados**

NoSQL cassandra em dados biológicos. 2014. viii, 50 f., il. Monografia (Bacharelado em Ciências da Computação) — Universidade de Brasília, Brasília, 2014.

BOCHI, Juarez da Silva. **Programação Paralela no Banco de Dados Chave-Valor Redis**. Diss. PUC–Rio, 2015

BRITO, Ricardo W. **Bancos de Dados NoSQL x SGBDs Relacionais: Análise Comparativa**. Faculdade Farias Brito e Universidade de Fortaleza, 2010

COSTA, L, de F,; Castro, A, F. (2021). **Comparação de performance entre um banco de dados relacional e dois bancos não-convencionais do tipo NoSQL** — Universidade Federal Rural do Semi-Árido - UFERSA

DATE, C. J.; **Introdução de Sistemas de Banco de Dados**. 8. ed. Rio de Janeiro : Elsevier, 2004. 865 p. Disponível em: https://books.google.com.br/books?id=xBeO9LSlK7UC&printsec=frontcover&hl=pt-BR&source=gbs_ge_summary_r&cad=0#v=onepage&q&f=false

DB-Engines - Knowledge Base of Relational and NoSQL Database Management Systems. Disponível em: <https://db-engines.com/en/>. Acesso em 18/05/2024

ELMASRI, R.; Navathe, S. B. **Sistema de Banco de Dados,** 2005. 89p. [São Paulo :Pearson Addison Wesley]. Disponível em: <https://tonysoftwares.com.br/attachments/article/5297/Sistema_de_banco_de_dados_Navathe.pdf>.

Microsoft Azure. **O que é PostgreSQL.** Disponível em: <https://azure.microsoft.com/pt-br/resources/cloud-computing-dictionary/what-is-postgresql>. Acesso em: 18/05/2024

PINTAR[1], P. H. R.; de Souza POLETTO, A. S. R. (2016). **Um estudo exploratório acerca de banco de dados nosql comparado aos bancos de dados relacionais**. Fórum científico da fema-anais, 15.

PostgreSQL. **Site oficial do PostgreSQL**. 2024. Disponível em: <https://www.postgresql.org/>. Acesso em: 18/05/2024.

RABELO, D.F; CÂNDIDO, M.V.I. (2017). **Análise de desempenho de banco de dados NoSQL em um sistema que utiliza um banco de dados relacional e não relacional para armazenamento de dados**.

Redis. **Site oficial do Redis**. 2024. Disponível em: <https://redis.io/>. Acesso em: 18/05/2024

SOARES, Antônio Michael Farias. **Implementação e avaliação de banco de dados chave-valor com aprendizagem de índice**. 2022. Trabalho de Conclusão de Curso (Bacharelado em Ciência da Computação) - Campus de Crateús, Universidade Federal do Ceará, Crateús, 2022.

TAKAI, O, K.; Italiano, I. C.; Ferreira, J. E. **Introdução a Banco de Dados**, 2005.

Tavares, Rosângela Ribeiro. **Base de Dados e Desenvolvimento de Aplicações Web**, 2012. [Cabo Verde : s.n.].

Documento Digitalizado Restrito

Artigo de TCC do aluno Filipe Ferreira Oliveira

Assunto: Artigo de TCC do aluno Filipe Ferreira Oliveira
Assinado por: Daiane Tomazeti
Tipo do Documento: Comprovante
Situação: Finalizado
Nível de Acesso: Restrito
Hipótese Legal: Informação Pessoal - dados pessoais e dados pessoais sensíveis (Art. 31 da Lei nº 12.527/2011)
Tipo do Conferência: Cópia Simples

Documento assinado eletronicamente por:

- **Daiane Mastrangelo Tomazeti**, **PROFESSOR ENS BASICO TECN TECNOLOGICO**, em 10/03/2025 16:58:02.

Este documento foi armazenado no SUAP em 10/03/2025. Para comprovar sua integridade, faça a leitura do QRCode ao lado ou acesse https://suap.ifsp.edu.br/verificar-documento-externo/ e forneça os dados abaixo:

Código Verificador: 1960940
Código de Autenticação: 89646caf8e