Aplicabilidade da Persistência Poliglota em Sistemas Potencialmente Escaláveis

Applicability of Poliglote Persistence in Potentially Scalable Systems

Keven Leone dos Santos [1] orcid.org/0000-0003-1275-8719
Andrêza Leite de Alencar [2] orcid.org/0000-0002-7083-0646

[1] Escola Politécnica de Pernambuco, Universidade de Pernambuco, Recife, Brasil,
[2] Departamento de Computação, Universidade Federal Rural de Pernambuco, Recife, PE, Brasil.

E-mail do autor principal: Keven Leone dos Santos kls@ecomp.poli.br

Resumo

Para lidar esta crescente geração de informação, é preciso que os sistemas suportem essa capacidade de dados e melhorem distribuição de seus serviços. Este artigo tem como principal objetivo o desenvolvimento de uma aplicação utilizando a persistência poliglota, utilizando MongoDB, Redis, MYSQL como bases de dados. Os passos necessários para desenvolvimento do estudo de caso, explicando o fluxo de atividades e entradas das etapas seguintes. Foram criados experimentos para avaliar o uso de duas abordagens: modelo relacional e poliglota, observando o consumo de hardware e tempo gasto para execução dos experimentos em diferentes aspectos. Como vantagem de se utilizar a persistência poliglota pode-se destacar a velocidade na execução de operações de manipulação de dados e alta disponibilidade, e do relacional o baixo consumo de hardware e bom desempenho em algumas operações. Ambos os modelos citados tem grande valor no mercado de trabalho, são ferramentas indispensáveis para o armazenamento de dados, como toda migração de dados apresenta um risco real de perda de informações, é importante avaliar os riscos para viabilidade de se utilizar a persistência poliglota.

Palavras-Chave: NoSQL. Open-source. Banco de Dados, Persistência Poliglota.

Abstract

To handle this growing information generation, systems need to support this data capacity and improve the distribution of their services. This article has as main objective the development of an application using polyglot persistence, using MongoDB, Redis, MYSQL as databases. The steps necessary to develop the case study, explaining the flow of activities and inputs of the following steps. Experiments and Results: Experiments were developed to evaluate the use of two approaches: relational and polyglot model, observing the hardware consumption and the time spent to execute the experiments in different aspects. The advantage of using polyglot persistence is speed in executing data manipulation and high availability operations, and relational low hardware consumption and good performance in some operations.Both models cited have great value in the labor market, are indispensable tools for data storage, as any migration of data presents a real risk of loss of information, it is important to evaluate the risks for feasibility of using polyglot persistence.

Key-words: *NoSQL. Open-source. Databases, Polyglot Persistence.*

DOI: 1025286/repa.v5i1.1245

1 INTRODUÇÃO

Nos últimos anos estamos presenciando um crescimento exponencial de dados. A origem destes dados é diversa, mas se caracterizam, principalmente de dados oriundos da Web e de Redes Sociais que, em geral, armazenam resultados das inúmeras interações com equipamentos eletrônicos como computadores, aparelhos celulares, GPS, tráfego urbano, entre outros. A expectativa é que nos próximos cinco anos sejam gerados mais dados que nos últimos 5000 anos **[1]**. Uma das maiores preocupações atuais é o que fazer com tanta informação, desde a forma de armazená-las quanto processá-las. Nesse contexto, novas formas de armazenamento foram criadas nos últimos anos afim de superar problemas de performance do modelo relacional e melhorar o armazenamento e recuperação dos dados. Estas novas ferramentas abordam novos paradigmas de bancos de dados que têm ganhado cada vez mais espaço entre as empresas que necessitam de modelos de dados mais flexíveis para armazenar suas informações.

Da perspectiva de desenvolvimento de aplicações, os bancos de dados relacionais são amplamente adotados **[2]**. Embora que, por muito tempo esta perspectiva tenha sido indiscutível, atualmente existem modelos cada vez mais atraentes, como os bancos não relacionais, conhecidos como NoSQL. Assim, a integração de soluções utilizando bancos relacionais e não relacionais numa mesma aplicação têm crescido muito, dando espaço à chamada Persistência Poliglota.

Neste artigo, um sistema de ativos de tecnologia será utilizado como estudo de caso. Este sistema armazena dados sobre os funcionários e os equipamentos que estes possuem na empresa. Embora muitos equipamentos tenham dados comuns, juntar todos os dados dos equipamentos em uma única tabela no modelo relacional pode ser um problema, tanto para a manutenção quanto para a inserção de dados, já que a tabela por ser genérica, pode deixar os dados esparsos, ou seja com muitas colunas sem informação.

Assim, iremos migrar um banco de dados relacional para uma abordagem de persistência poliglota, utilizando os seguintes modelos:

- MongoDB (NOSQL);
- REDIS (NoSQL);
- MySQL (SQL);
- NodeJS (Linguagem de programação).

A linguagem NodeJS foi utilizada para a comunicação entre as bases de dados, servidor e o cliente solicitante das requisições.

Após a migração para a solução poliglota, foram realizados experimentos onde foi possível analisar os índices de desempenho utilizando o modelo poliglota. Como resultado foi possível identificar que, com a persistência poliglota, se obtêm mais velocidade nas operações de CRUD, além de ter um esquema flexível a mudanças, facilitando manutenções e escalando a aplicação e os bancos de dados de acordo com a necessidade e utilização.

Nas seções seguintes, serão detalhados os objetivos deste trabalho bem como a aplicação do estudo de caso e os resultados obtidos nos experimentos.

2 OBJETIVOS

Este trabalho tem como principal objetivo a realização de experimentos para comparação de uma aplicação de persistência relacional com a persistência poliglota, utilizando os conceitos da persistência poliglota em um ambiente real. Objetivos específicos:

- Migrar uma solução de banco de dados de modelo relacional para uma solução poliglota com uso de bases: REDIS, MONGODB e MySQL.
- Avaliar a performance das duas abordagens (relacional e poliglota) para identificar os ganhos/benefícios alcançados com a utilização do modelo não relacional.
- Identificar os índices de desempenho e consolidar a aplicabilidade da persistência poliglota neste estudo de caso.

3 TRABALHOS RELACIONADOS

Diversos trabalhos têm sido desenvolvidos nos últimos anos com novas propostas de uso para a persistência poliglota seu conceito define o uso de diferentes tecnologias de bancos de dados para lidar com diferentes necessidades de armazenamento **[3]**.

Podendo destacar o trabalho de NANCE; *et al*, **[4]**. Que levanta a proposta da utilização da persistência poliglota em um sistema de E-commerce apresentando as melhores tecnologias NoSQL a serem adotadas e reforçando a importância de não esquecer ou ficar preso apenas uma tecnologia ou segmento, exemplificando a importância dos modelos relacionais e não relacionais no desenvolvimento de aplicações.

No estudo de GESSERT; *et al*, **[1]**. Os autores utilizam uma API REST para se comunicar com as diversas bases de dados mostrando a força de um novo paradigma que vem ganhando força dos

softwares como serviço, neste caso utilizando o conceito de BaaS (Back-and as a Service) e DBaaS (Database as a Service).

O trabalho de ARAÚJO; TIMES; SILVA **[3]**. Descreve a utilização da persistência poliglota para armazenamento de dados de saúde e como se utilizar, mostrando as possibilidades de uso de um sistema de armazenamento de dados heterogêneo.

Muitas pesquisas ainda estão em desenvolvimento, a persistência poliglota passou a ser tendência devido a importância de armazenar grande volume de dados, que são transitados constantemente pela rede atraindo novas pesquisas que envolvem as melhores práticas e adotando novos conceitos destas tecnologias.

Os trabalhos apresentados acima utilizam a persistência poliglota em diferentes situações de mercado, como saúde, engenharia, comércio eletrônico e outros, exemplificando como pode ser utilizado os modelos poliglotas, e relacionais.

Os trabalhos abordam a utilização conjunta entre modelo relacional e poliglota sem à necessidade de escolher um único modelo no desenvolvimento de soluções, de acordo com o problema que a organização busca resolver.

Neste artigo será trabalhado a transição de um modelo relacional para a persistência poliglota, verificando e comparando o desempenho entre os modelos citados.

4 FUNDAMENTAÇÃO TEÓRICA

Para a realização do estudo de caso neste trabalho será utilizado o conceito da Persistência Poliglota, onde diferentes bancos de dados são desenhados para resolver problemas diversos **[5]**, como performance e alta disponibilidade. A partir das qualidades de cada modelo de banco de dados utilizados por essa pesquisa, foi criado um barramento para unir as bases de dados, um serviço REST, que é um estilo arquitetural onde a transferência de estados representacionais ocorre sobre o protocolo HTTP **[6]** e assim realizar os experimentos vistos no projeto.

4.1 Conhececndo o NoSQL

Not **O**nly **SQL** mais conhecido como NoSQL, surgiu com a proposta de gerenciamento de grandes volumes de dados com pouca ou sem nenhuma estrutura que precisam de alta disponibilidade e escalabilidade, se originou como consequência a ineficiência de modelos relacionais para as tarefas citadas.

Uma das grandes vantagens do noSQL é a escalabilidade do serviço de forma horizontal e a tolerância a falhas, além disso existe uma enorme variedade de soluções, voltada para atender uma necessidade que encaixa melhor a determinadas situações. A seguir estão listados os modelos mais populares de armazenamento com seus principais serviços.

- Key/Value – Chave Valor - É um modelo simples de dados, armazena apenas chave e valor dentro de um HASH, suporta grandes volumes de dados **[7]**, Ex: Redis.
- Wide Columns Store – Família de Colunas - Modelo responsável por armazenar colunas ao invés de linhas e altamente otimizados para inserção massiva de dados **[7]**, Ex: Cassandra, HBase.
- Document Store – Documentos - No modelo de documentos são inseridos dados semiestruturados como Json e XML, cada registro possui uma identificação única **[7]**, Ex: MongoDB, CouchDB.
- Graph Store – Grafos - Os grafos possuem um modelo bem diferenciado dos demais, utiliza um conjunto de vértices e arestas, utilizando um conjunto de nós e relacionamentos para se conectarem **[8]**, Ex: Neo4J, InfoGrid.

4.2 Persistência Poliglota

Persistência Poliglota define o uso de diferentes bases de dados para lidar com diferentes necessidades de armazenamento **[9]**, fugindo do padrão de utilizar apenas um banco central para toda a aplicação. Utiliza em cada modelo de banco de dados um que atenda melhor a necessidade de determinados serviços com melhor eficácia. A principal ideia de usar persistência poliglota é armazenar dados estruturados em uma abordagem relacional enquanto o semiestruturado ou não estruturado utilizar o modelo não relacional **[3]**.

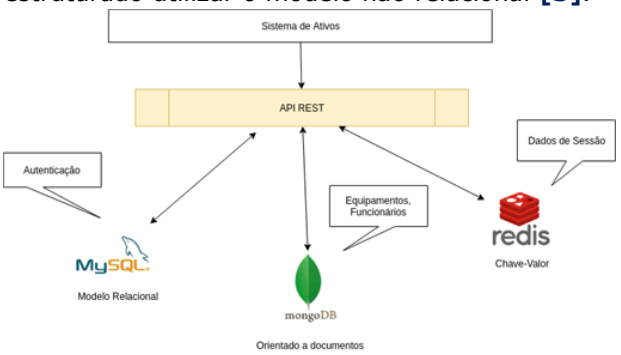

Figura 1. Exemplo adotado para elaboração da pesquisa.

DOI: 1025286/repa.v5i1.1245

Neste sistema de ativos de informática, utiliza-se diferentes bases de dados: MySQL, MongoDB, Redis, a melhor forma de manter a consistência de dados é criando um barramento, neste caso uma API Rest que será responsável pela conversação entre as diferentes bases de dados e através dessa API as informações poderão ser recuperadas de forma fácil pelo Front-end da aplicação ou até uma chamada por dispositivos móveis.

4.3 Diferenças entre SQL e NoSQL

As diferenças entre os dois modelos são diversas. Enquanto o modelo relacional está preocupado nas transações ACIDs e seguem um padrão para o armazenamento de dados, utilizando Linhas e Colunas, os bancos não relacionais possuem estrutura flexível criadas para atender necessidades diferentes. A Tabela 1 apresenta um resumo das principais diferenças.

Tabela 1: Diferenças entre SQL e NoSQL.

Características	Relacional	Não Relacional
Armazenamento	Informação armazenada em tabelas, tendo a necessidade de criar uma ou mais tabelas para estabelecer relacionamentos	A informação é salva em apenas um registro e estrutura em forma de documento, como Json, XML e outros
Escalabilidade	Em grande parte, são escaláveis verticalmente, significa dizer que uma máquina servidora cresce em sua grande maioria com seu Hardware, seja uma adição de CPU, Memória RAM ou Disco.	Utiliza escalonamento horizontal, com clusters distribuídos de Hardware
Estrutura	Pré-Definida, baseadas em tabelas com linhas e colunas	Dinâmica, podendo ser de Documentos, Chave Valor, Grafos, Orientada a Colunas
Linguagem	Universal no contexto SQL com poucas variâncias	Completamente variante dependendo do modelo do banco
Foco	Integridade	Desempenho, Disponibilidade, Escalabilidade

O modelo relacional é fundamentado no princípio de que dados são guardados em tabelas e colunas **[10]**.

A ideia por trás do modelo não relacional é entregar ao usuário um modelo de dados específico com esquemas flexíveis para criação de aplicações modernas, estes modelos são otimizados para armazenar um grande volume de dados.

4.4 Bancos de dados com esquema flexível

A partir do momento que o usuário quer armazenar informações em um banco relacional, primeiro se define um esquema, é definido uma estrutura para o banco de dados que diz quais as tabelas e colunas existentes e o tipo de dados que são armazenados.

Neste esquema é definido as colunas, os tipos de dados suportados por elas e outras propriedades que são opcionais como tamanho de caracteres, valores padrões dentre outros.

Um dos pontos negativos do modelo relacional é solicitação de mudanças, Modelos relacionais apesar de sua maturidade e consistência não seguem as novas dinâmicas de desenvolvimento de software, que sempre estão em constantes mudanças, até mesmo durante o ciclo de desenvolvimento ou a manutenção evolutiva **[1]**.

Atualmente este acoplamento pode ser reduzido ou até extinto com o NoSQL o armazenamento de dados é flexível e não se tem uma estrutura definida, ela pode ser ajustada a qualquer momento sem necessidade de alteração de esquema do componente. Cada solução tem suas características para o armazenamento das informações.

No caso do MongoDB o dado tem estrutura de documento em formato JSON ou BSON. O MongoDB não possui esquema definido, ou tipagem como: String, Boolean, Integer, date, etc, entretanto, esquemas explícitos tem a vantagem de permitir a checagem de tipo e validações para prevenir o que o dado seja corrompido **[11]**.

5 METODOLOGIA

O desenvolvimento do presente trabalho será um estudo de caso, que irá avaliar o uso da persistência poliglota. A execução deste trabalho seguiu as etapas do processo abaixo:

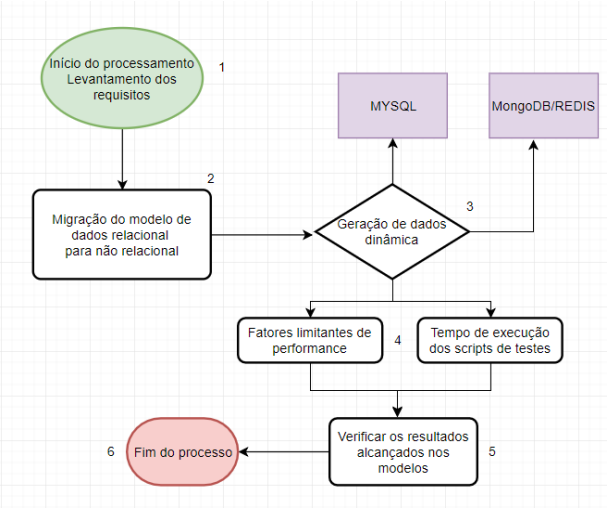

Figura 2 – Fluxo da metodologia

1. Levantamento dos requisitos necessários para desenvolvimento dos modelos a serem utilizados (SQL e Poliglota), visualizando o modelo relacional proposto na Seção 6 e verificando quais bases de dados podem ser utilizadas para a migração poliglota.

2. Realizar a migração do modelo de dados relacional para uma abordagem não relacional, de acordo com a Figura 4, buscando separar os dados que se encaixem melhor em diferentes bases de dados não relacionais.

3. A partir do modelo criado na solução poliglota, um algoritmo é utilizado para gerar os dados necessários nos testes de desempenho (etapa 4 do fluxograma), na base de dados relacional e poliglota.

4. Execução de scripts para popular a base de dados, utilizados nos testes de desempenho. Verificar se possui fator limitante no teste, como estrutura dos dados utilizados em diferentes bases de dados. As métricas serão utilizadas na etapa 5.

5. Após realização dos testes de desempenho entre os modelos apresentados os resultados são utilizados como evidência para análise de desempenho entre os modelos utilizados.

6. Nesta etapa todos os testes e dados foram criados, e os resultados serão apresentados na Seção 7 deste artigo.

6 IMPLEMENTAÇÃO

A migração da base de dados relacional para a poliglota que será utilizada neste estudo de caso está representada de forma resumida na Figura 3.

É possível verificar que na tabela de ativos tecnológicos existem diversas chaves estrangeiras relacionadas a outras tabelas, neste exemplo foi exibida apenas o relacionamento entre ativos tecnológicos, computador e funcionário.

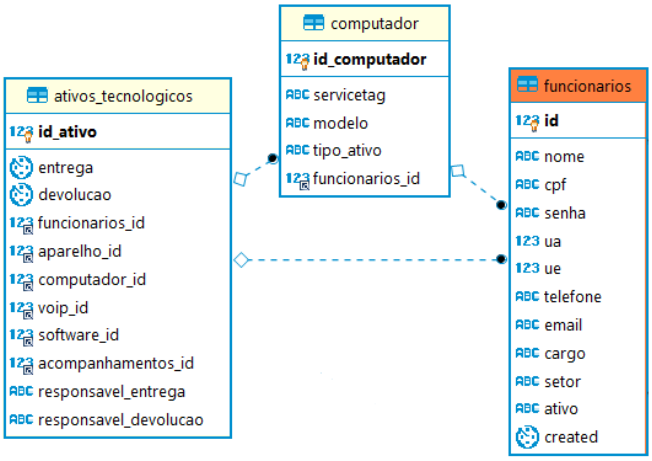

Figura 3. Modelagem Entidade Relacionamento

Em seguida os dados das tabelas na Figura 3 serão migradas para o MongoDB, Redis e MongoDB.

DOI: 1025286/repa.v5i1.1245

```
new Schema({
    id: ObjectId,
    nome: String,
    cpf: String,
    ua: String,
    ue: String,
    telefone: String,
    email: String,
    cargo: String,
    setor: String,
    ativo: Boolean,
    endereco: [{
        cidade: String,
        pais: String,
        estado: String,
        bairro: String,
        rua: String,
        numero: Number,
        cep: String,
        atual: Boolean
    }],
    ativos_tecnologicos: [{
        imei: Number,
        servicetag: String,
        mac: String,
        modelo: String,
        chips: Number,
        tipo_ativo: String,
        chip_ativado: Boolean,
        numero: String,
        entrega: String,
        devolucao: String,
        em_uso: Boolean,
        serial_number: Number,
        responsavel_entrega: String,
        responsavel_devolucao: String,
        serial_key: String,
        software: String,
        expira: Boolean,
        anexos: [],
        acompanhamentos: []
    }],
    created: {type: Date, default: Date.now},
})
```

Figura 4. Modelo Não Relacional (Mongo).

Os dados contidos nas tabelas presentes na Figura 3 foram inseridos dentro de uma única coleção (tabela) como mostra a Figura 4, sem a necessidade de criar novas coleções para estabelecer relacionamentos entre os dados.

Neste caso específico a tabela de ativos tecnológicos e endereço se tornaram um Array dentro do objeto principal de funcionário, e, embora na Figura 4 exiba outros campos que não são utilizados na tabela de computador estes dados podem ser omitidos na hora do cadastro de um novo ativo de computador para um determinado funcionário.

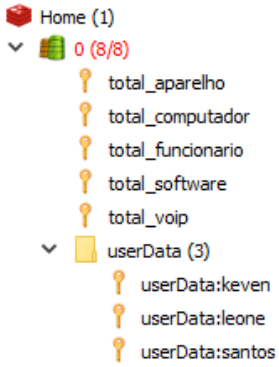

Figura 5. Modelagem Chave-valor (Redis).

O Redis apresenta uma modelagem simples e seu propósito é servir como consulta direta a alguns dados do sistema como contadores e logs dos usuários.

De forma dinâmica ao efetuar login ao sistema uma nova chave é criada caso não exista dentro do Hash userData. Esses hashes são responsáveis por armazenar alguns dados do usuário, como último acesso e nível acesso.

Após a migração, o MySQL foi utilizado apenas para controle de acesso. Para isto, foi criado uma única tabela chamada funcionários, com as seguintes colunas CPF, NOME, SENHA, ATIVO (VERDADEIRO OU FALSO).

Para a comunicação com as bases de dados e o servidor foi utilizada a API REST. Os seguintes passos precisam ser respeitados para se obter um resultado da API Rest.

- **Autenticar o usuário com a rota /api/login** - Ao enviar um usuário e senha válido é retornado uma chave de acesso (Token).
- **Utilizar o token gerado nas demais rotas através do Header da requisição.**

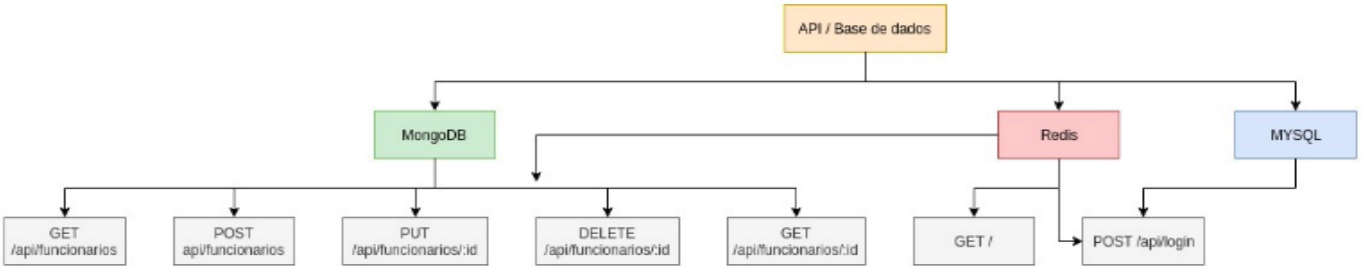

Figura 6. Rotas e bases de dados.

A organização das rotas e as bases de dados estão descritas na Figura 6, com exceção da rota de login, todas as demais rotas utilizam o MongoDB para armazenamento de dados.

Cada rota utiliza o conceito da persistência poliglota como apresentado na Figura 6.

Os dados utilizados nos testes foram gerados a partir de uma biblioteca que gera dados aleatórios, assim foi possível automatizar a geração de informações trazendo um ambiente mais dinâmico para os testes a serem realizados, que será melhor apresentada na Seção 6.1.

O uso das diversas bases de dados será melhor detalhado a seguir.

O Redis é utilizado neste projeto para armazenar contadores, logs de ações de usuários. O Redis está presente em todas as rotas e são executados em conjunto com o MongoDB, seu papel é inserir e recuperar valores seguindo a regra de negócio específica para cada rota, por ser um banco que armazena as informações na memória do servidor todas as transações são feitas rapidamente.

Através deste trabalho conjunto do MongoDB e Redis foi possível eliminar a necessidade de fazer consultas que retornem um total de itens cadastrados, é o caso do total de funcionários que foi centralizado no Redis. Assim sempre que um funcionário ou usuário é cadastrado este valor é incrementado.

Devido à estrutura relacional do modelo proposto na Figura 3, optou-se pela adoção do MongoDB. Que é forte com tipos variantes de dados, listas e objetos com facilidade.

Por esta razão o MongoDB está presente na maioria das rotas, como fonte principal de armazenamento.

O MYSQL está tratando apenas a autenticação e cadastro de usuários, como a maioria das bases de dados em empresas armazena registros de seus funcionários em um modelo relacional fica mais fácil tratar esta relação.

Tendo em vista que um login não é feito constantemente e não precisa ser escalonado como no caso dos documentos armazenados no MongoDB, o fato do MySQL realizar transações ACID é um fator importante para assegurar a confiança das operações realizadas.

6.1 Geração de Dados

Os dados podem ser gerados ou manipulados a partir de uma chamada a API REST enviando uma requisição com as informações referentes ao exemplo apresentado na Figura 4 para as rotas apresentadas na Figura 6.

Para a geração de dados, foi utilizado um algoritmo que é responsável por criar os dados fictícios e inserir no banco de dados. Assim, é possível garantir uma maior variabilidade dos dados e arrays que são criados com diferentes tamanhos.

Para a realização dos testes, foi gerada uma massa de dados idêntica, que foram utilizadas no MySQL e no modelo Poliglota.

7 EXPERIMENTO E RESULTADOS

Alguns testes de desempenho foram realizados com a solução implementada. Para o processamento dos testes foi utilizada uma máquina com 16GBs de Memória RAM, processador I7 da 7º Geração, HD SSD e Windows 10 como sistema operacional.

Todos os testes foram efetuados 5 vezes para verificação das seguintes características: Consumo de Memória, CPU e tempo de execução das operações. Dessa forma é possível a realização dos testes em um processo de CRUD (Cadastro, leitura, atualização e remoção), de forma unitária ou em larga escala.

Os experimentos foram realizados em uma única máquina, de forma não distribuídas e cada teste pôde ser realizado utilizando uma massa de dados de quantidade variante, de acordo com o experimento. Cada experimento realizou um ou mais operações de CRUD, que estão descritas no título de cada figura e os experimentos foram realizados comparando o modelo poliglota com o relacional.

A massa de dados pode variar de acordo com o teste e estará melhor descrita em seus respectivos testes.

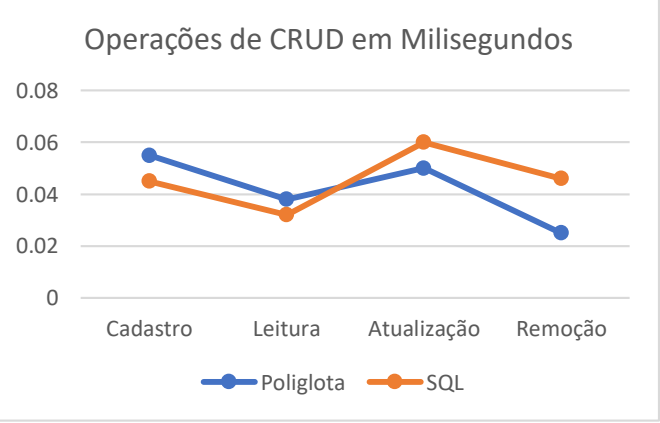

Figura 7. Médias do tempo de execução para cada operação de CRUD - Teste unitário (1 Registro).

DOI: 1025286/repa.v5i1.1245

Ambos os modelos apresentaram tempo de execução semelhantes e com resultados empatados em 2 testes cada.

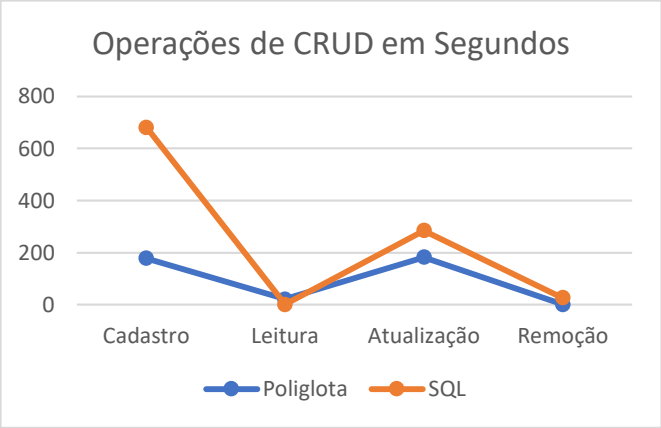

Figura 8. Médias do tempo de execução para cada operação de CRUD – Massa com 100.000 Registros.

No teste realizado da figura 8 a discrepância no tempo gasto para execução do teste foi maior. Com exceção da leitura, o SQL levou tempo maior em todas as operações realizadas.

No cadastro, o tempo médio para inserção dos dados do SQL chegou até 3 vezes mais tempo que o modelo Poliglota. Devido a esta grande diferença, foram realizados os testes presentes nas Figuras 9 e 10. Nestes, foi medido o tempo gasto para o cadastro dos dados em diferentes quantitativos de registros, de 100 até 1 milhão, onde cada "K" representa mil registros e "KK" milhão de registros. O eixo X representa o número de registros processados e no Y o tempo gasto.

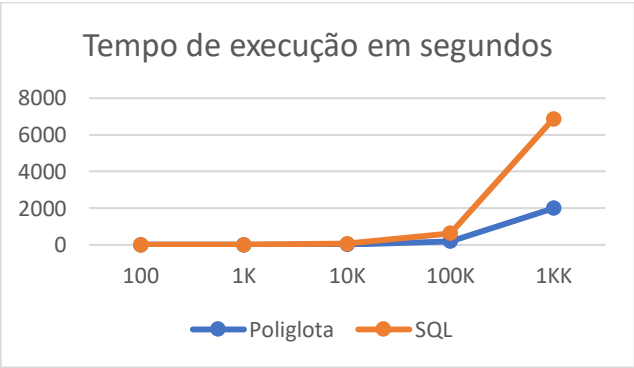

Figura 9. Médias do tempo de execução para cada operação de cadastro.

Durante a realização do teste da figura 9 foi utilizado o modelo relacional proposto na Figura 3, o modelo poliglota obteve melhor tempo na execução do cadastro, comparado ao SQL, durante a realização do teste o modelo relacional chegou a durar o triplo do tempo para execução da mesma tarefa no modelo poliglota.

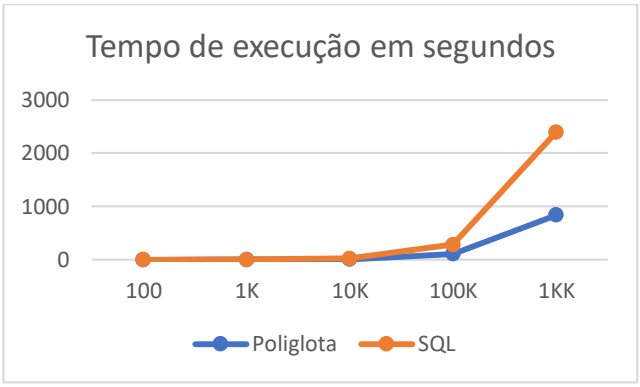

Figura 10. Médias do tempo de execução para cada operação de cadastro simples.

O teste da Figura 10 foi realizado com um cadastro simples, que utiliza apenas a tabela de funcionários como mostra a Figura 3 e, no poliglota desconsidera as listas e Arrays de ativos tecnológicos, endereço, acompanhamentos e anexos de ativos tecnológicos que foram utilizados no teste passado (Figura 9).

O modelo poliglota apresentou melhor resultado levando menos tempo para execução do teste em diferentes níveis de registros.

Além do tempo, foi verificado o consumo de Memória RAM e de CPU dos bancos de dados durante a execução das operações de CRUD utilizando 100.000 registros, como mostra as Figuras 11 e 12.

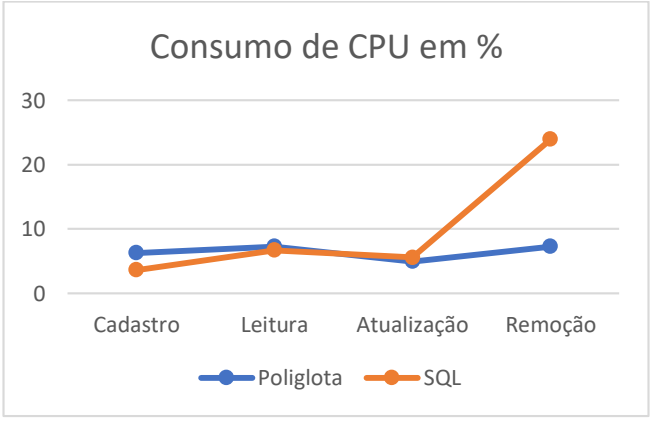

Figura 11 – Recursos de Hardware (CPU)

O consumo de CPU foi moderado e baixo durante os testes, destacando o maior consumo de CPU na operação de leitura e remoção de dados, o resultado

do teste ficou empatado, cada modelo se destacou no menor consumo de CPU 2 vezes cada.

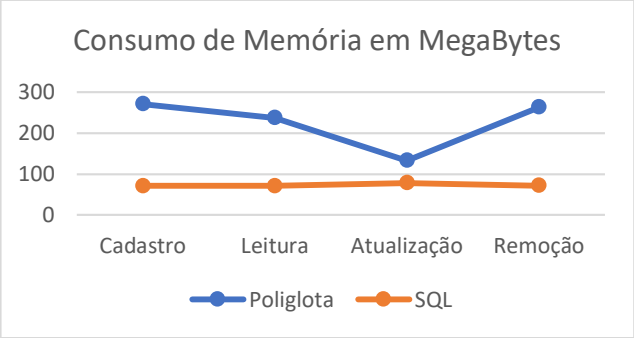

Figura 12 – Recursos de Hardware (RAM)

No teste de memória (Figura 12) o modelo Poliglota consumiu mais memória RAM que o SQL em todos as operações, o consumo de memória do SQL foi estável com pouca oscilação, diferente do modelo poliglota que o consumo de memória variou em mais de 100 MBs em uma das operações.

Os indicadores de desempenho podem variar por alguns fatores, dentre eles Hardware, infraestrutura, base de dados, estrutura dos dados e outros detalhes, um fator que pode explicar essa diferença na execução das operações em diferentes modelos pode ser a forma que cada banco estrutura suas informações, como também as engines utilizadas no SQL e Poliglota que podem não ser a mais otimizada para execução das operações mencionadas em grande escala.

8 CONSIDERAÇÕES FINAIS E TRABALHOS FUTUROS

Este artigo teve como objetivo conhecer a persistência poliglota e como esta pode ser empregada em projetos de pequeno a grande porte. Compatível com a maioria das linguagens de programação do mercado, esse modelo pode ser utilizado com diferentes bancos de dados no desenvolvimento de software.

A análise apresenta uma possibilidade diferente de lidar com o desenvolvimento de sistemas voltadas aos dias atuais, com a grande variância das informações os bancos de dados não relacionais podem ser agregados em diversas etapas de um sistema de informação.

Durante os testes foi possível verificar os índices de desempenho das bases de dados utilizadas, seus pontos fortes serão descritos logo abaixo.

MongoDB e Redis tiveram um alto nível de desempenho na execução dos testes, porém consumindo mais recursos de Hardware comparado ao SQL e entregando resultados em tempo inferior ao modelo relacional.

O SQL consumiu menos recursos de Hardware, porém na maioria das operações apresentou um tempo maior na execução dos experimentos para realizar as operações de CRUD.

No desenvolvimento de aplicações, há espaço para a utilização de ambos os modelos. Estes podem trabalhar em conjunto, explorando as características e vantagens de cada um.

Cada modelo tem suas vantagens e desvantagens, podendo destacar de vantagem no modelo poliglota a facilidade de manipulação de dados, banco de dados flexível e o desempenho no tempo de execução e consumo de CPU. No SQL, as vantagens foram o baixo consumo de memória e velocidade no processamento de algumas operações.

Uma desvantagem observada na persistência poliglota foi o consumo de memória maior e a necessidade de um maior investimento na infraestrutura, tendo em vista que o poliglota utiliza como as bases de dados o MongoDB e o Redis neste estudo de caso.

Como desvantagem do relacional pode-se mencionar que este, além de possuir um modelo rígido (não flexível), apresentou alto consumo de CPU e maior tempo na execução das operações de CRUD.

Todos os experimentos foram realizados sem modificações estruturais nos bancos de dados utilizados. O que pode explicar os valores obtidos nos experimentos, como o consumo de memória, CPU, e tempo de execução, possivelmente os experimentos seriam diferentes com ajustes adicionais nestas configurações, uma investigação mais ampla seria necessária para descobrir.

Para utilizar essa abordagem em um problema real, é importante analisar custos e riscos que envolvem uma migração de dados. É recomendada a avaliação de um especialista para verificar se há necessidade real para realizar uma migração para a persistência poliglota, e avaliar se mudanças na configuração da base de dados por meio de índices, engines trariam resultados melhores ou semelhantes ao artigo.

Para trabalhos futuros, esta pesquisa pretende aplicar a solução em novas análises de desempenho dentro da persistência poliglota abrangendo novos indicadores, bases de dados e realizar o testes utilizando diferentes engines na camada física para o

DOI: 1025286/repa.v5i1.1245

MongoDB e MySQL, como InnoDB, TocuDB, e verificar se com ajustes na configuração das bases de dados pode alcançar melhores resultados.

REFERÊNCIAS

[1] GESSERT, F. *et al.* **Towards a Scalable and Unified REST API for Cloud Data Stores**, 2013.

[2] ISSA, A.; SCHILTZ. **Document Orient Databases**. Université Libre Bruxelles. 2015.

[3] ARAÚJO, A.M.C.; TIMES, V.C.; SILVA, M.U. **A Framework for Polyglot Persistence of the Electronic Health Record.** Universidade Federal de Pernambuco, 2016.

[4] NANCE, C.; LOSSER, T.; LYPE, R.; HARMON, G. **NOSQL VS RDBMS – WHY THERE IS ROOM FOR BOTH**. Rev. Association for Information Systems AIS Eletronic Library (AISeL), v. 5, n. 18, p. 110-116, 2013.

[5] SALADAGE, P.J; FOWLER, M. NoSQL Distiled: **A Brief Guide to the Emerging World of Polyglot Persistence**. Rev. Pearson Education. México. 1º ed. cap 1. 2 e 3, p. 23-30, 2012.

[6] ANDERSON, C. **Banco de dados, persistência poliglota e NoSQL**. Rev. iMasters, v.14, n.4, p. 49-51, 2015.

[7] GRADY, S. **The New King Makers: how developers conquered teh world**, New Relic Inc All Rights Reserved. Estados Unidos. 2º ed. cap. 1,2,3,4,5 e 6, p. 36 – 44, 2013.

[8] OLIVEIRA, S.S. **Bancos de Dados não relacionais; Um novo paradigma para armazenamento de dados em Sistemas de Ensino Colaborativo**. Revista da Escola de Administração Pública do Amapá. Macapá, v.2, n.1, p. 184, 2014.

[9] KAUR, K., RANI, R. **Managing Data in Healthcare Information Systems: Many Models, One Solution**; IEEE Computer Society, 2015.

[10] HEUSER, C.A. **Projeto de Banco de Dados. Série livros didáticos**. UFRGS. 2º ed. cap.1, p. 56-61, 1998.

[11] SCHEIBEL, G. **Uma arquitetura de software para armazenamento de trajetórias por meio da técnica de persistência poliglota**. Dissertação de Mestrado. Universidade do Estado de Santa Catarina. Programa de Pós-Graduação em Computação Aplicada. Joinville, 2016.