CENTRO FEDERAL DE EDUCAÇÃO TECNOLÓGICA CELSO SUCKOW DA FONSECA - CEFET/RJ

Uma análise de desempenho de funções criptográficas nativas de Sistemas Gerenciadores de Banco de Dados Open Source

André de Souza Rosa

Monografia apresentada ao Curso de Graduação em Sistemas de Informação, do Centro Federal de Educação Tecnológica Celso Suckow da Fonseca, CEFET/RJ Campus Nova Friburgo, como parte dos requisitos necessários à obtenção do certificado de Bacharel em Sistemas de Informação

Orientador(a): Nilson Mori Lazarin

Nova Friburgo

Junho 2023

AGRADECIMENTOS

Agradeço primeiramente a Deus, por ter me dado a oportunidade e força para começar e chegar até o fim do curso, apesar de todas as dificuldades que se impuseram nesse período. Agradeço a minha família, que esteve presente me incentivando e me auxiliando no que fosse possível para que eu pudesse me dedicar aos estudos e em honra da memória de meu pai, Mário da Silva Rosa, que se alegrou comigo quando consegui minha vaga no curso e sempre foi exemplo de persistência na busca por objetivos. Agradeço também aos colegas e professores, que fizeram parte de toda minha história nesse período e sempre contribuíram nos momentos de dificuldade. A conclusão desse trabalho não só é um grande passo para minha vida profissional, mas também para meu crescimento e amadurecimento pessoal, obrigado a todos que fizeram parte dessa trajetória.

RESUMO

Uma análise de desempenho de funções criptográficas nativas de Sistemas Gerenciadores de Banco de Dados Open Source

Com o crescimento dos ataques cibernéticos ao redor do mundo, surgiram legislações que regulam, entre outras tarefas, como os dados pessoais devem ser armazenados para garantir segurança e a prevenção de acesso indevido aos mesmos. No meio computacional, a implementação de medidas de segurança geram impactos no desempenho de sistemas, tornando a relação Desempenho X Segurança um desafio a ser tratado pelos projetistas de banco de dados no momento da seleção do SGBD a ser utilizado. Esse trabalho tem como objetivo avaliar o desempenho de SGBDs open source na utilização de funções criptográficas nativas, que empreguem criptografia simétrica. Os SGBDs avaliados nesse estudo foram o MariaDB, o PostgreSQL e o Firebird. Para avaliação, foi utilizada uma aplicação desenvolvida, responsável por fazer a conexão e operações que realizavam inserção, busca, atualização e remoção de registros nos SGBDs. Foram realizados 10 testes, para cada faixa de carga de trabalho, sendo de 1000, 2000, 4000 e 8000 registros, para operações com dados em claro e operações utilizando as funções criptográficas, que empregaram o algoritmo AES com chave de 128 bits. Foram adotadas como métrica de avaliação o tempo médio consumido na execução das operações, a média de utilização de CPU e ocupação média de memória RAM durante a execução dos testes. Para capturar os resultados, foi utilizada ferramenta Atop. Avaliando o impacto que a criptografia causou a cada SGBD, quando comparado o desempenho com a utilização de dados em claro, o mais afetado foi o PostgreSQL, sendo o menos afetado o MariaDB. O MariaDB foi o mais rápido em tempo de execução quando comparado aos demais, sendo o Firebird mais lento. Porém, o modo de operação utilizado como padrão pelo MariaDB é o mais frágil dentre os utilizados nos SGBDs. Dessa forma, avaliando a relação Segurança X Desempenho, com base nas configurações e forma de testes adotadas nesse trabalho, o PostgreSQL é o mais recomendável.

Palavras-chave: SGBD relacional; criptografia simétrica; benchmark; SGBD open source ; funções criptográficas nativas

ABSTRACT

A performance analysis of cryptographic functions native to Open Source Database Management Systems

With the growth of cyber-attacks around the world, legislation has emerged that regulates, among other tasks, how personal data should be stored to ensure security and prevent unauthorized access to them. In the computational environment, the implementation of security measures generate impacts on the performance of systems, making the Performance X Security relationship a challenge to be dealt with by database designers when selecting the DBMS to be used. This work aims to evaluate the performance of open source DBMS in the use of native cryptographic functions, which employ symmetric cryptography. The DBMSs evaluated in this study were MariaDB, PostgreSQL and Firebird. For evaluation, an application was developed, responsible for making the connection and operations that performed insertion, search, update and removal of records in the DBMS. Ten tests were carried out, for each workload range, with 1000, 2000, 4000 and 8000 records, for operations with plaintext and operations using cryptographic functions, which employed the AES algorithm with a 128-bit key. The average time consumed in the execution of operations, the average CPU utilization and average RAM memory occupation during the execution of the tests were adopted as evaluation metrics. To capture the results, the Atop tool was used. Evaluating the impact that cryptography caused to each DBMS, when compared to the performance with the use of plaintext, the most affected was PostgreSQL, the least affected was MariaDB. MariaDB was the fastest in terms of runtime when compared to the others, with Firebird being the slowest. However, the operation mode used as default by MariaDB is the most fragile among those used in DBMSs. Thus, evaluating the Security X Performance relationship, based on the configurations and form of tests adopted in this work, PostgreSQL is the most recommended.

Keywords: relational DBMS; symmetric encryption ; benchmark; Open source DBMS; native cryptographic functions

SUMÁRIO

1- Introdução

Um grave problema do meio digital que afeta negativamente a vida de pessoas ao redor do mundo é o vazamento de dados sigilosos. Segundo o *Massachusetts Institute of Tecnology* (MIT), no ano de 2019 ocorreram mais de 205 milhões de dados vazados de brasileiros. Com relação a organizações, foram 40 milhões afetadas por vazamento de CNPJ, nome fantasia, razão social e data de constituição (DIGITAL, 2021). Durante os seis primeiros meses de 2021 foram mais de 4,6 bilhões de credenciais vazadas ao redor do mundo, sendo esse valor 387 % maior que os vazamentos registrados durante todo ano de 2019 (CNN, 2021).

Diante do cenário de grandes vazamentos de dados e do contexto de crescente digitalização das diversas interações da sociedade, já existem mais de cem países com marcos regulatórios para proteção de dados pessoais (ATHENIENSE, 2018). Grande parte dessas legislações, incluindo a brasileira, são inspiradas na versão da União Europeia, a *General Data Protection Regulation* (GDPR), que vigora desde maio de 2018. A legislação brasileira para esse fim, a Lei Geral de Proteção de Dados (LGPD) foi aprovada em agosto de 2018 e entrou em vigor desde 18 de setembro de 2020 (TRUZZI, 2020).

O conceito geral de segurança de computadores é definido como proteção direcionada para um sistema de informação automatizado, visando preservar a confidencialidade, a integridade e a disponibilidade dos recursos de sistemas de informação, que compreendem hardwares, softwares, firmwares, informações/dados e telecomunicações.(STALLINGS, 2015)

O conceito da confidencialidade, um dos requisitos centrais da GDPR e LGPD (INFOSEC, 2022), visa manter controle autorizado de acesso e divulgação de informações, garantindo meios para tal. É composto por dois conceitos, o de confidencialidade de dados e privacidade. A confidencialidade de dados define o controle de acesso de informações privadas e confidenciais. O conceito de privacidade define que indivíduos tem o direito de controlar quais dados e informações sobre eles podem ser obtidas e armazenadas, assim como por quem elas podem ser obtidas, a forma de acesso e para quem elas podem ser exibidas (STALLINGS, 2015).

O conceito da integridade tem como requisito a prevenção da alteração ou destruição indevida da informação. É composto pelos conceitos de integridade de dados e integridade do sistema. A integridade de dados define que informações e softwares sejam modificados apenas por formas pré-definidas e autorizadas. A integridade do sistema define que o mesmo execute suas funções de forma ilesa, livre de manipulações indevidas do sistema (STALLINGS, 2015). O conceito da disponibilidade tem como requisito o acesso e confiável da informação. É baseado na garantia do acesso aos dados e serviços do sistema em tempo integral para usuários autorizados (STALLINGS, 2015).

A LGPD tem como princípios fundamentais no tratamento de dados a Finalidade, Adequação, a Necessidade, o Livre Acesso, a Qualidade dos Dados, a Transparência, a Segurança, a Prevenção, a Não Discriminação, e Responsabilização (CIDADANIA, 2021). Os princípios da LGPD que norteiam esse trabalho são o de segurança e prevenção. De forma geral, o princípio da Segurança define a aplicação de medidas técnicas e administrativas para resguardar os dados pessoais de acesso não autorizado e de situações, acidentais ou ilícitas, que ocasionem perda, alteração, comunicação ou divulgação não desejada. O princípio da Prevenção prevê a implementação de medidas para evitar danos oriundos do tratamento de dados (CIDADANIA, 2021). No contexto dos sistemas de informação, um instrumento possível para aplicar medidas que resguardam os princípios de segurança e prevenção dos dados sensíveis é o Sistema Gerenciador de Banco de Dado (SGBD).

Os SGBDs são softwares responsáveis por, entre outras funcionalidades, gerenciar o acesso e o armazenamento dos dados (HEUSER, 2009). No contexto de segurança de dados, parte das políticas de segurança a serem implementadas pelas organizações, se encontram na adequação da camada de persistência dos dados, utilizando criptografia. A criptografia consiste nos processos de tornar o texto em claro em um texto cifrado, através da encriptação, e retornar do texto cifrado para o texto em claro, através da decriptação (STALLINGS, 2015).

Os SGBDs possuem funções nativas de cifragem de determinados tipos de dados, que auxiliam na manutenção segura de dados sensíveis no banco de dados. As funções criptográficas dos SGBDs utilizam na sua implementação algoritmos tradicionais de criptografia. Dentre eles, temos como exemplo o *Advanced Encryption Standard* (AES), o *Triple Data Encrypition Standard* (3DES) e o Blowfish.

Para auxiliar na tomada de decisões de projetistas de bancos de dados e organizações,

trabalhos têm sido propostos com o intuito de comparar o desempenho de diferentes SGBDs, proprietários e open source. As comparações, são realizadas utilizando softwares de benchmark e outros, construídos para avaliação de desempenho, aplicando diferentes parâmetros de comparação, para avaliar o comportamento dos SGBDs em operações com volumes de dados variados.

1.1- Definição do Problema

A implementação de medidas de segurança no armazenamento de dados, utilizando a criptografia, é fundamental para o atendimento das legislações. Entretanto, traz a tona o problema da relação entre Desempenho x Segurança, já que o consumo de recursos computacionais se elevam ao realizar o processo de codificação/decodificação de grandes volumes de dados.

Uma limitação existente nos SGBDs, de forma geral, está na variedade de algoritmos de criptografia que oferecem. Alguns SGBDs tem suporte a poucos algoritmos de criptografia simétrica nas suas funções, como o SQLite (TELLE, 2021). Outros possuem extensões implementadas para possuir acesso a mais recursos (POSTGRESQL, 2023b). O algoritmo utilizado pelas funções criptográficas e o modo de operação em que são aplicadas influenciam no nível de segurança alcançado e também no seu desempenho.

Dessa forma, ao se criar um sistema, para escolher o SGBD a ser utilizado na implementação de medidas de segurança, também deve ser feita uma avaliação do impacto que o algoritmo e modo de operação utilizado causarão no desempenho em relação à segurança que oferecem. Essa tarefa comparativa não é simples, tendo em vista que o desempenho dos SGBDs varia conforme os tipos de operação (inclusão, consulta, exclusão e alteração) em razão do volume de dados a serem manipulados no processo, assim como os algoritmos de criptografia suportados pelo SGBD em questão exercem influência no desempenho final das operações, quando aplicadas medidas de segurança no armazenamento de dados.

1.2- Objetivo

Esse trabalho tem como objetivo realizar uma avaliação de desempenho de funções criptográficas nativas dos SGBDs open source MariaDB, FireBird e PostgreSQL. Serão realizados testes dos SGBDs por instruções da linguagem Standard Query Language (SQL), utilizando diferentes faixas de carga de trabalho, com dados oriundos de uma base pública, executando comandos de inserção, consulta, atualização e remoção de registros. Serão feitos testes com dados em claro e aplicando as funções criptográficas nativas de seu respectivo SGBD.

As métricas utilizadas para avaliação dos SGBDs serão o tempo médio de execução gasto na execução das instruções, média de CPU utilizada e média de memória RAM ocupada durante as instruções.

Ao realizar essa avaliação, esse trabalho visa contribuir com uma indicação de SGBD relacional open source que possa ser adotado na implementação de ambiente de armazenamento de dados, considerando o impacto causado ao empregar funções criptográficas simétricas nativas dos SGBDs avaliados, para satisfazer a relação Segurança x Desempenho.

1.3- Estrutura do Trabalho

No capítulo 2, de Fundamentação teórica, serão apresentados conceitos utilizados ao longo do trabalho. No capítulo 3, serão apresentados Trabalhos Relacionados ao tema abordado, no capítulo 4 será apresentada a metodologia utilizada no experimento de comparação dos SGBDs, no capítulo 5 serão apresentados os resultados obtidos através dos testes realizados e no capítulo 6 serão apresentadas as conclusões do trabalho.

2- Fundamentação Teórica

Neste capítulo, serão abordados conceitos dos algoritmos e de tecnologias utilizadas na abordagem do trabalho. Serão apresentados os conceitos do modo de operação, do algoritmo de criptografia simétrica AES e uma apresentação superficial dos SGBDs MariaDB, PostgreSQL e Firebird.

2.1- Modo de operação

Uma cifra de bloco utiliza bloco de texto de tamanho fixo, de comprimento de X bits, e uma chave como entrada, produzindo uma saída de bloco de texto cifrado de tamanho X. Se a entrada submetida for maior do que X bits, ela pode ser dividida em blocos de X bits.

Já um Modo de operação é uma técnica utilizada para aperfeiçoar o efeito de um algoritmo de criptografia ou adaptá-lo para uma aplicação, como a cifra de bloco para uma sequência de blocos de dados. Existem cinco modos de operação são utilizados com cifras de bloco simétricas, que utilizam a mesma chave para cifrar e decifrar dados (STALLINGS, 2015). Nesse trabalho, serão utilizadas 3 modos de operação, o ECB, o CBC e o OFB.

2.1.1 Eletronic Codebook (ECB)

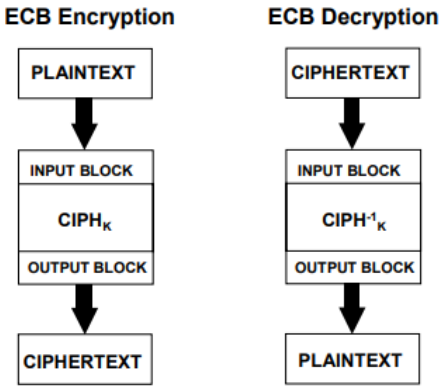
Nesse modo de operação, cada bloco de texto em claro é cifrado e decifrado de forma independente, utilizando a mesma chave. Para um bloco de texto em claro de b bits, sendo b um tamanho fixo, existe um texto cifrado exclusivo, para uma determinada chave. Uma mensagem de entrada maior que b é quebrada em blocos de b bits, sendo o último bloco completado caso a quantidade de dados não alcance o tamanho b. Se o

mesmo bloco de texto em claro de tamanho b ocorrer mais de uma vez na mensagem, o texto cifrado será o mesmo (STALLINGS, 2015).

A repetição de textos cifrados idênticos para entradas iguais de blocos de tamanho b de textos em claro, faz com que a aplicação do ECB não seja recomendável para mensagens longas. A repetição pode ocasionar padrões de pares de textos claros/cifrados a serem explorados por criptoanalistas (STALLINGS, 2015).

A Figura 1 exemplifica o funcionamento do modo de operação ECB. Um texto em claro é dividido em blocos de tamanhos fixos. Os blocos são processados de forma independente, onde cada um dos blocos de texto em claro é submetido ao algoritmo criptográfico juntamente com a chave criptográfica, que é comum a todos os blocos. Ao final do processamento do algoritmo criptográfico, é gerado então um bloco de texto cifrado. O processo de deciframento é semelhante, sendo submetido ao algoritmo criptográfico o bloco de texto cifrado e a chave utilizada na cifragem.

Figura 1 – Modo de operação ECB



(DWORKIN, 2001)

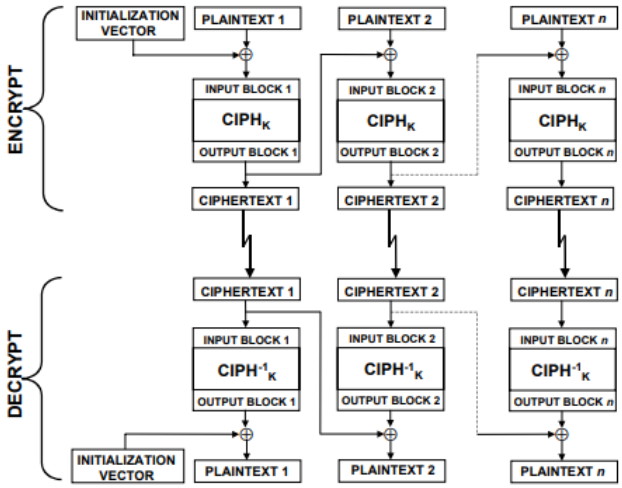
2.1.2 Cipher Block Chaining (CBC)

No CBC, o texto em claro é divido em blocos de tamanho fixo. O primeiro bloco de texto em claro passa por uma operação de XOR com um vetor de inicialização, conhecido pelo emissor e receptor dos dados, e depois submetido ao algoritmo de encriptação. A partir disso, a entrada da função de encriptação passa a ser a saída da operação XOR do bloco de texto em claro a ser cifrado com o bloco cifrado anteriormente. A mesma

chave é utilizada para cada processo de cifragem. A entrada da função de encriptação não possui nenhum relacionamento fixo com o texto em claro, dessa forma, repetições de bloco de tamanho b de texto em claro não produzem a mesma saída cifrada. Caso falte dados para preencher o último bloco a ser cifrado, o mesmo também é preenchido (STALLINGS, 2015). O processo citado é exemplificado pela Figura 2, onde cada bloco de texto em claro, na Figura representado por *plaintext*, faz parte da mesma mensagem a ser cifrada.

No deciframento dos dados, o vetor de inicialização passa por um XOR com a saída da função de decriptação para dar origem ao primeiro bloco de dados decifrados. Os blocos decifrados posteriormente são provenientes de uma operação XOR, do resultado da função de decriptação do bloco cifrado atual com o bloco de dados que passou pelo processo de decriptação anteriormente. O vetor de inicialização é um bloco do mesmo tamanho do bloco de cifra (STALLINGS, 2015).

Figura 2 – Modo de operação CBC



(DWORKIN, 2001)

2.1.3 Modo Output Feedback (OFB)

No modo de operação OFB, para cifrar o primeiro bloco, o vetor de inicialização é submetido a função de encriptação. A saída dessa função passa por uma operação XOR com o primeiro bloco de dados em claro, dando origem ao primeiro bloco cifrado

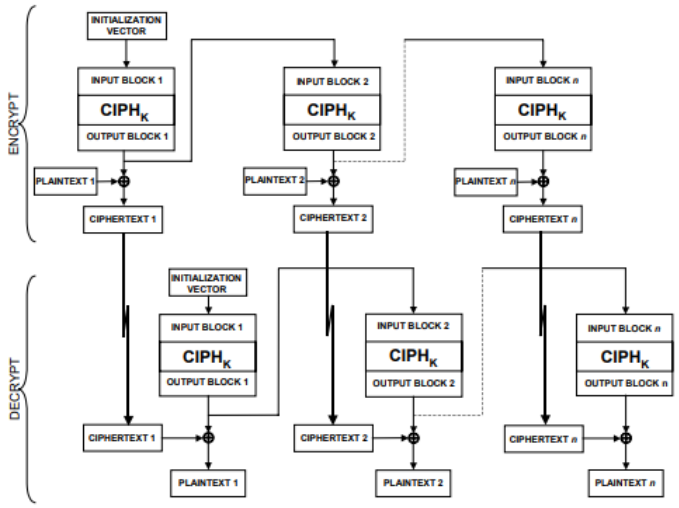
(STALLINGS, 2015).

A saída da função criptográfica de um bloco serve de alimentação para a função criptográfica do bloco seguinte, onde se repete o processo de operação XOR do bloco em claro com a saída da função criptográfica atual para originar o próximo bloco cifrado. Considerando os blocos de tamanho b, se o último bloco for de tamanho menor que b, os bits mais significativos são usados para a operação XOR, os bits correspondentes entre a diferença de b com os mais significativos utilizados são descartados.Todas as funções de encriptação utilizam a mesma chave (STALLINGS, 2015).

O vetor de inicialização para o OFB deve ser único para cada execução de encriptação, sendo denominado como *nonce*. Isso ocorre pelo fato de que a função de encriptação do primeiro bloco recebe apenas a chave e o vetor de inicialização, sendo a saída um fluxo de bits fixo. Considerando duas mensagens diferentes com bloco de texto em claro idêntico, na mesma posição, um atacante poderia determinar essa parte do fluxo de bits (STALLINGS, 2015).

Para decriptar os dados o processo é semelhante ao de cifragem, se diferenciando apenas a aplicação dos blocos cifrados na operação XOR onde antes eram aplicados os blocos de texto em claro para cifragem (STALLINGS, 2015).

Figura 3 – Modo de operação OFB



(DWORKIN, 2001)

2.2- Advanced Encryption Standard (AES)

O AES é uma técnica de cifra simétrica de bloco que foi padronizada pelo NIST no ano de 2001. Ele trabalha com entradas de blocos de texto de 128 bits /16 bytes e utiliza chave criptográfica que pode ter tamanho especificado entre 128, 192 ou 256 bits, o que faz com que haja uma designação indicativa de qual chave o algoritmo esteja utilizando na ocasião (AES-128, AES-192, AES-256) (STALLINGS, 2015).

Os dados do bloco de entrada são organizados e manipulados através de uma matriz de bytes de quatro linhas, denominada estado, cujo valor da quantidade de colunas resulta da divisão do tamanho do bloco de entrada por 32 bits (DWORKIN et al., 2001). O número N de rodadas utilizada pelo algoritmo para encriptação varia, de acordo com o tamanho do bloco de entrada e do tamanho da chave de encriptação (DWORKIN et al., 2001), sendo N = 10 rodadas para a versão padrão do AES-128, N = 12 rodadas para o AES-192 e N = 14 rodadas para o AES-256 (STALLINGS, 2015).

Figura 4 – Exemplo de Matriz Estado

State array

$s_{0,0}$	$s_{0,1}$	$s_{0,2}$	$s_{0,3}$
$s_{1,0}$	$s_{1,1}$	$s_{1,2}$	$s_{1,3}$
$s_{2,0}$	$s_{2,1}$	$s_{2,2}$	$s_{2,3}$
$s_{3,0}$	$s_{3,1}$	$s_{3,2}$	$s_{3,3}$

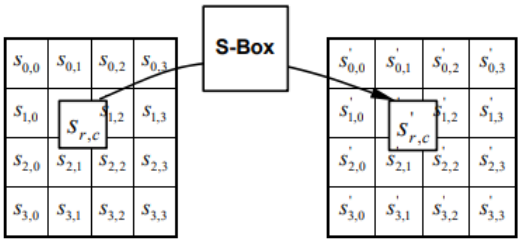
(DWORKIN et al., 2001)

O processo de transformação realizado durante as rodadas se dá da seguinte maneira: antes da primeira rodada é utilizada uma função de transformação chamada AddRoundKey. Nas N-1 rodadas iniciais temos SubBytes, ShiftRows, MixColumns e novamente a AddRoundKey. Por fim, na última rodada são utilizadas apenas três das funções (STALLINGS, 2015).

A função SubBytes manipula uma matriz de 16 x 16 de bytes, na qual seu conteúdo é uma permutação de 256 valores possíveis formados por 8 bits, chamada de S-Box (STALLINGS, 2015). É feita a substituição dos bytes individuais que formam estado por um existente na S-Box (DWORKIN et al., 2001).

A função ShiftRows consiste no deslocamento de bytes pertencentes às linhas de estado nela mesmo, onde a primeira linha não é alterada, na segunda linha há uma

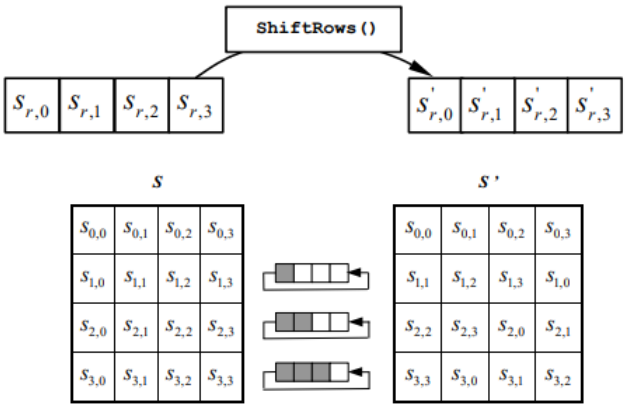
Figura 5 – SubBytes



(DWORKIN et al., 2001)

movimentação circular de 1 byte para a esquerda. Na terceira linha esse movimento também é realizado por 2 bytes e na quarta por 3 bytes (STALLINGS, 2015).

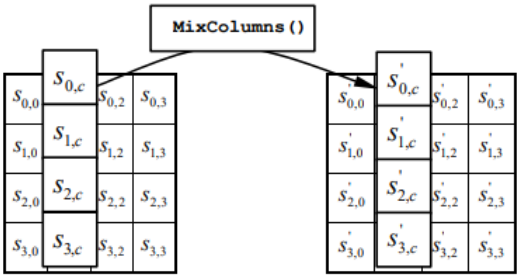
Figura 6 – ShiftRows



(DWORKIN et al., 2001)

A função de transformação MixColumns trabalha misturando os dados de cada coluna da matriz estado. Em função dos quatro bytes que compõem a coluna, o valor de cada byte é mapeado para um novo (STALLINGS, 2015). A função MixColumns usa multiplicação de matrizes e adição XOR bit a bit para gerar os resultados.

Figura 7 – MixColumns



(DWORKIN et al., 2001)

A função AddRoundKey, de forma geral, trabalha realizando uma operação XOR

entre cada um dos bits da tabela estado com os bits da subchave criptográfica utilizada na rodada (DWORKIN et al., 2001). A subchave em questão consiste em uma matriz de mesmo tamanho de estado, gerada a cada rodada a partir da chave criptográfica original. O escalonador de chaves do algoritmo fica a cargo de criar essas subchaves, criando uma chave também para a primeira rodada onde apenas a função AddRoundKey é utilizada (DWORKIN et al., 2001).

2.3- MariaDB

O MariaDB é um SGBD relacional open source desenvolvido pelos desenvolvedores originais do MySQL, como um *fork* do próprio, depois que o MySQL foi adquirido pela Oracle Corporation em 2009 (PEARCE, 2013). Suas últimas versões lançadas incluem recursos para atender formatos JSON, sistema de informação geográfica (GIS), análise colunar, compatibilidade com Oracle, SQL distribuído para escalabilidade massiva e muitas outras características (MARIADB, 2023a).

Existem recursos compatíveis e não compatíveis entre versões do MariaDB e do MySQL (MARIADB, 2023d). A respeito da criptografia, a assinatura da função nativa do MariaDB (MARIADB, 2023b) é semelhante à implementada no MySQL (ORACLE, 2023), porém a implementação do MariaDB tem algumas limitações.

As funções AES_ENCRYPT e AES_DECRYPT no MariaDB só recebem como argumento o texto a ser cifrado/decifrado, e a chave a ser utilizada. Nativamente, o tamanho de chave utilizado é de 128 bits. Para utilizar chaves de 256 bits, é necessário fazer mudança no código-fonte do SGDB. Também não é possível alterar o modo de operação utilizado, que por padrão é o ECB. Existe requisição em aberto para sejam implementadas as possibilidades de mais opções no modo de operação e de utilização de vetor de inicialização, mas até a conclusão desse trabalho, a requisição continua em aberto (GOLUBCHIK, 2023). No MySQL, essas funções podem receber mais alguns argumentos, e o modo de operação e o tamanho da chave utilizada podem ser configurados por uma variável de sistema chamada 'block_encryption_mode', porém no MariaDB essa variável não existe (MARIADB, 2023e).

2.4- PostgreSQL

O PostgreSQL é um SGBD objeto-relacional open source que usa e estende a linguagem SQL. O PostgreSQL teve sua origem no ano de 1986 como parte do projeto POSTGRES da Universidade da Califórnia em Berkeley (POSTGRESQL, 2023a).

O PostgreSQL funciona nos principais sistemas operacionais, implementa a estrutura ACID (Atomicidade, Consistência, Isolamento e Durabilidade) nas instruções desde 2001 e possui uma série de extensores que ampliam a quantidade de funcionalidades suportadas, como o PostGIS, que adiciona suporte para armazenar, indexar e consultar dados geográficos (POSTGRESQL, 2023a).

O PostgreSQL possui a extensão *pgcrypto*, que disponibiliza diversas funções criptográficas. São elas as funções gerais de *hash*, funções de *hash* de senha, funções de criptografia PGP, funções de criptografia bruta e funções de dados aleatórios (POSTGRESQL, 2023b). Para que a extensão seja ativada, é necessário conectar na base desejada e executar o comando: CREATE EXTENSION IF NOT EXISTS pgcrypto;

As funções de criptografia PGP implementam a parte de criptografia do padrão OpenPGP (RFC 4880). Suporta chaves simétricas e chaves públicas. Para ambos os casos, podem ser feitas manipulações opcionais de dados, como compactação, conversão para UTF-8 e/ou conversão de finais de linha. Os dados são prefixados com um bloco de bytes aleatórios, que funciona de forma equivalente a utilizar um vetor de inicialização aleatório. São anexados um *hash* SHA1 no bloco de bits aleatórios e ao fim tudo é criptografado com a chave de sessão e colocado no pacote de dados (POSTGRESQL, 2023b).

As funções de criptografia bruta não possuem recursos avançados do PGP. Elas usam a chave do usuário diretamente no processo de cifragem, não fornecem nenhum meio de verificação de eventuais alterações em dados criptografados, todo gerenciamento de parâmetros da função ficam por conta do usuário, inclusive o vetor de inicialização e não tem manipulação opcional de dados (POSTGRESQL, 2023b).

2.5- Firebird

O Firebird teve origem como um derivado do código do Borland InterBase 6.0, e tem uma licença baseada na Mozilla Public License (CANTU, 2010). Firebird é um SGBD relacional open source, que suporta modelo integrado de usuário único até implantações de vários bancos de dados de volume de dados de 2 Terabytes e volumes de dados na casa de Gigabytes em execução com centenas de clientes simultâneos (FIREBIRD, 2023b). O SGBD pode ser instalado em diversos Sistemas Operacionais (Windows, Linux, MacOS, HP-UX, AIX, Solaris e outros). Possui arquitetura multigeracional que permite o desenvolvimento e suporte de aplicativos híbridos OLTP e OLAP. OLTP refere-se a processamento de transações on-line, enquanto o OLAP se refere a processamento analítico on-line (IBM, 2023). O Firebird tem suporte à *stored procedures* e *triggers*, e tem suporte a grande parte do padrão SQL92 (FIREBIRD, 2023b).

As funções criptográficas ao nível de campo foram disponibilizadas a partir do Firebird 4.0. As funções de cifragem e decifragem (ENCRYPT()/DECRYPT()) possuem diversos parâmetros, como o texto a ser cifrado/decifrado, o algoritmo de criptografia a ser aplicado, o modo de operação utilizado e a chave criptográfica. Alguns outros parâmetros são aplicados dependendo do modo de operação escolhido, como o vetor de inicialização, que não é utilizado para o ECB e para modos de operação de cifra de fluxo, ctr_type, que só é aplicado para o moto de operação *The Counter* (CTR), ctr_length, que só é aplicado para o modo CTR e o initial_counter, que só é aplicado ao modo de operação de cifra de fluxo CHACHA20 (FIREBIRD, 2023a).

A versão das chaves utilizadas não se define na atribuição do algoritmo, mas pelo tamanho da chave utilizada. Uma chave de 16 bytes indica a versão AES-128, uma chave de 24 bytes indica AES-192 e uma chave de 32 bytes indica AES-256 (FIREBIRD, 2023a).

3- Trabalhos Relacionados

Nesse capítulo, serão apresentados alguns trabalhos publicados que abordam, de forma direta ou indireta, os conceitos utilizados nesse trabalho. Ao realizar um levantamento nesse segmento de pesquisa, espera-se encontrar possíveis contribuições que esse trabalho pode proporcionar com o escopo adotado.

Para chegar até os trabalhos acadêmicos aqui abordados, foi realizado foi realizada uma pesquisa seguindo passos do mapeamento sistemático, que teve como *string* base para busca, utilizada no mecanismo virtual de pesquisa Google Scholar: “ (”benchmark”or ”comparison”or ”Analysis”) AND (”DBMS”or ”database”) AND (”encryption”or ”cryptography”) ”

Foi feita uma filtragem para que fossem retornados apenas trabalhos a partir do ano de 2017. Na ocasião que foi realizada essa pesquisa, foram retornados 171 resultados. Num primeiro momento, foi feita a análise dos abstracts de cada um dos trabalhos retornados. Desses trabalhos, não foi possível ter acesso ao conteúdo de 6 trabalhos, por serem de plataformas pagas. Um trabalho foi identificado como resultado duplicado da busca e 19 foram aceitos para uma segunda etapa de avaliação.

A segunda etapa de avaliação consistiu em uma avaliação qualitativa, onde foi analisado o conteúdo de cada trabalho. Para avaliar o conteúdo dos trabalhos, foram elaboradas 6 perguntas:

1. “O Trabalho descreve o experimento utilizado para comparação dos SGBDs? ”;
2. “Autores descrevem limitações do estudo? ”;
3. “O objetivo do trabalho está claramente definido? ”;
4. “Os resultados dos experimentos foram expostos de forma clara? ”;
5. “O trabalho apresenta variedade (3 ou mais) SGBDs avaliados? “ ;
6. “O trabalho apresenta variedade (3 ou mais) de algoritmos de criptografia avaliados? “ ;

Para cada uma dessas perguntas, foram elaboradas 3 respostas possíveis, cada uma com uma pontuação: “não” valendo 0 pontos, “parcialmente” valendo 0,5 ponto

e “sim” valendo 1 ponto. Com a soma dos pontos de cada resposta, foi aplicada uma nota para cada trabalho avaliado, sendo selecionados as 3 melhores pontuações. Esses trabalhos serão abordados nesse capítulo.

3.1- Performance benchmarking of data-at-rest encryption in relational databases (ISTIFAN; MAKOVAC, 2022)

O trabalho aponta como contexto de problema, a falta de estudos para mensurar como a taxa de Transação Por Minuto (TPM) afeta o desempenho de diferentes bancos de dados relacionais, quando aplicado o mesmo algoritmo para criptografia de dados em repouso (data-at-rest), com número crescente de usuários simultâneos. O algoritmo abordado em questão é o AES.

O objetivo do trabalho é avaliar se existe diferença significante de desempenho, baseado em taxa de transferência, quando utilizadas diferentes versões de chaves com o algoritmo AES, para SGBDs que utilizam a mesma tecnologia para criptografia de dados em repouso. Os SGBDs abordados são o MariaDB e o MySQL e é utilizado o AES-192 para comparação de desempenho dos mesmos.

Correlacionando o objetivo do trabalho dos autores com o trabalho aqui apresentado, há em comum a avaliação de desempenho da aplicação da criptografia simétrica com o algoritmo AES em SGBDs relacionais, sendo o MariaDB uma seleção comum entre os trabalhos. Porém, a forma de emprego da criptografia se difere, visto que os autores aplicam a funcionalidade de criptografia de dados em repouso, comum entre o MariaDB e MySQL, nos dados armazenados na sua totalidade. Os autores não deixam explícito no texto como a criptografia é aplicada, ficando a cargo da implementação utilizada pela ferramenta que utilizaram para realizar os testes.

No trabalho aqui apresentado, a criptografia foi aplicada por meio da utilização das funções criptográficas nativas dos SGBDs abordados, que se aplica a campos específicos das tabelas, empregadas no nível da aplicação através das instruções SQL. Os SGBDs abordados nesse trabalho não possuem tecnologia comum entre si de criptografia de dados em repouso, como a abordada pelos autores nativamente, se aproximando mais nas funções nativas de criptografia simétrica.

A abordagem utilizada pelos autores foi descrita, inicialmente, por um estudo para conhecer como os SGBDs utilizados funcionavam e o que seria necessário para implementar a criptografia de dados em repouso nos mesmos. Após isso, foi selecionado o software para benchmark a ser utilizado nos experimentos. Foi selecionado para esse fim o HammerDB, com a justificativa de ser aderente aos SGBDs mais utilizados no mercado e ser utilizado também em outros trabalhos desse segmento. Os SGBDs foram instalados e configurados para escutar a mesma porta TCP, de forma que um não pudesse estar em execução ao mesmo tempo que o outro. No trabalho aqui proposto, para execução dos testes foi desenvolvida uma aplicação própria, que será abordada posteriormente no capítulo 4, instalada em máquinas virtuais independentes para cada um dos SGBDs, utilizando a mesma configuração, também apresentada no capítulo 4.

Os autores realizaram algumas configurações em ambos os SGBDs para otimização. Eles justificam que, como os dois sistemas são similares, dado que o MariaDB é uma versão bifurcada desenvolvida pela comunidade do MySQL, utilizam arquivos de configuração também similares. Dessa forma, foram feitos ajustes para garantir condições iguais. Foram configurados o tamanho da buffer pool, threads concorrentes, tamanho do arquivo de log do InnoDB e quantidade de threads de criptografia. No trabalho aqui apresentado, não ocorreu nenhuma otimização nos SGBDs avaliados.

O HammerDB foi utilizado pelos autores para criar o conjunto de dados nos SGBDs. Ao criar os dados, o benchmark também criou stored procedures utilizadas para mensurar a taxa de transferência durante a execução dos testes. O HammerDB também criou, de forma explicita, índices, em quantidades iguais para os dois SGBDs, já que conjunto de dados utilizado em ambos foi o mesmo. O benchmark foi configurado da mesma forma para os dois SGBDs, os testes foram executados separadamente na mesma máquina física, que continha apenas os softwares necessários para os experimentos.

O volume de dados utilizado pela ferramenta de benchmark para os testes, foi originado do no esquema TPROC-C, para cargas de trabalho OLTP. O protocolo TPROC-C implementa um sistema de que simula pedidos de clientes para uma empresa. As stored procedures criadas pela ferramenta são: New Order, Payment, Delivery, Order-Status e Stock-Level. Todas elas utilizam alguma forma de leitura, escrita ou atualização de dados, e são selecionadas aleatoriamente para a execução das cargas de trabalho. No trabalho apresentado, os dados utilizados para carga de trabalho são oriundos de um conjunto de dados públicos, que será abordado no capítulo 4. Os testes que usam os mesmos são

realizados em ordem bem definida, também apresentada no capítulo 4.

Os Autores realizaram testes de carga para cada um dos SGBDs, com cada versão de chaves do AES e também sem emprego de criptografia. Cada teste é formado por 5 subtestes com quantidades diferentes de usuários virtuais, sendo de 5, 10, 15, 20 e 25 usuários. Os SGBDs utilizados pelo trabalho aqui apresentado possuem algumas limitações com relação às funções criptográficas nativas que não permitiram o emprego de diferentes tamanhos de chaves nos testes, como será abordado no capítulo 4. Dessa forma, nesse quesito, não foi possível estabelecer uma correlação de desempenho.

Cada SGBD foi submetido a 4 testes de carga, de 120 linhas. Para cada uma dessas linhas, é calculado o valor médio de TPM por linha em cinco subtestes com diferentes números de usuários virtuais. Ao detalhar a quantidade dos testes, os autores descrevem que:

> "Cada um desses subtestes consiste no valor médio de TPM por linha em cinco testes de amostra com a mesma quantidade de usuários virtuais ativados. Um teste de carga completo consiste em cinco subtestes com cinco testes de amostra para cada subteste. Resumindo, 3.000 linhas de dados para cada teste de carga. "

Os testes foram configurados para passar por uma fase inicial de aceleração de 1 minuto, onde os usuários virtuais se conectam ao banco de dados e armazenam dados em cache no buffer do banco de dados. Após isso, se inicia um período de testes de 3 minutos, em que dados são coletados nos 2 primeiros minutos, pois o HammerDB tende a desconectar os usuários antes que se atinja a marca de 3 minutos. Os dados coletados durante o minuto de aceleração e do último minuto dos 3 foram descartados, pois o interesse é apenas nos dados onde o teste está em desempenho máximo, para não comprometer a aferição a TPM.

Os resultados foram coletados e representados em diagramas de caixa para visualizar o desempenho de cada versão do AES. Foi aplicada uma análise ANOVA sobre os resultados para validação dos mesmos. Verificou-se que, levar apenas o TPM como critério de comparação entre os SGBDs poderia levar a uma conclusão errada, pois a ferramenta de benchmark utilizada possuía limitações que influenciavam nesse sentido. Os autores adotaram então uma segunda métrica, a de Novas Ordens Por Minuto (New Orders Per Minute – NOPM).

A métrica NOPM foi avaliada por meio de testes de estresse utilizando o AES-192 em ambos SGBDs. Essa decisão foi tomada, pois, verificou-se que nos primeiros testes

de carga ocorreu um aumento na taxa de transferência do MySQL, enquanto essa versão era utilizada. Análises ANOVA também foram utilizadas para validar os resultados. As métricas adotadas divergem das revisadas nos trabalhos relacionados, apresentados no decorrer do artigo, que em sua maioria foram a taxa de uso de CPU e memória RAM. As métricas TPM e NOPM são geradas pela própria ferramenta de benchmark utilizada, independente de implementações particulares dos SGBDs.

O trabalho aqui apresentado difere do trabalho dos autores em relação à quantidade de testes feitas, nas métricas utilizadas e na forma de aferir as mesmas. Nesse trabalho, foram adotadas faixas diferentes de caga de trabalho, tando para dados em claro quanto para testes empregando as funções criptográficas. Foram aferidos o tempo de execução médio das operações realizadas, o percentual médio de uso de CPU e média de memória RAM ocupada durante a execução dos testes. Os autores tomaram suas métricas, próprias da ferramenta HammerDB, no tempo pré-estabelecido citado, já o trabalho aqui apresentado aferiu as métricas considerando a finalização do script de teste no tempo em que cada SGBD conseguiu concluir as operações. Porém, um ponto em comum é a desconsideração do tempo do primeiro segundo de medição do percentual de CPU utilizado nas operações, assim como os autores desconsideraram o primeiro minuto por serem resultados discrepantes. A estrutura dos testes e a forma de aferição das métricas do trabalho aqui apresentado serão mais detalhadas no capítulo 4.

Com base nos resultados dos testes e validações feitas, os autores chegaram a algumas conclusões, comparando do desempenho das versões de chaves do AES no próprio SGBD e comparando o desempenho de ambos. Concluíram que, no MySQL, existe diferença de desempenho significativa em termos de TPM do AES-192 em comparação ao AES-256. Nesse sentindo de comparação, no MariaDB não ocorre uma diferença significativa entre as versões de chaves do AES, salientando não haver grande diferença de desempenho quando aplicadas chaves criptográficas de tamanhos diferentes.

Os autores concluíram que, ao comparar o desempenho do AES-192 dos dois SGBDs, MariaDB superou o Mysql nos testes de carga e de estresse. O MariaDB superou o MySQL com diferença significativa em TPM nos testes de carga, com essa diferença confirmada nos testes de estresse que adotou a métrica NOPM. Esse resultado auxilia na confirmação da seleção do MariaDB como um dos SGBDs a serem avaliados nesse trabalho, em detrimento do MySQL, embora que em última instância seja difícil comparar

o resultado alcançado pelo MariaDB nos dois trabalhos, pela diferença de métricas adotadas. O trabalho apresentado busca enriquecer essa linha de pesquisa comparando o MariaDB com os demais SGBDs adotados.

Na parte de discussões gerais, os autores relatam que a princípio pretendiam utilizar também bancos proprietários da Microsoft e da Oracle. Porém, o contrato de licenciamento das duas empresas não permitia a publicação de benchmark dos seus produtos sem o consentimento da empresa. Assim, por não conseguirem contato com as empresas a tempo, optaram por utilizar o MariaDB e o MySQL nos estudos, por estarem sob a General Public License (GLP). Essa é uma das motivações pela opção de avaliar apenas SGBDs open source no trabalho aqui apresentado.

3.2- Impact of Symmetric-Key Encryption on Performance of Relational and Nosql Database to Preserve Data Confidentiality (KILONZO, 2020)

O autor aborda como contexto, o constante aumento de ataques cibernéticos a dados de organizações e falhas de segurança das bases de dados que as mesmas possuem. Nesse contexto, é apontada a utilização da criptografia como uma resposta possível aos problemas. Porém, o trabalho relata que a perda de desempenho que o emprego da criptografia acarreta, considerando o tempo de execução, a utilização de CPU e utilização de memória, faz com que as organizações e indivíduos tenham dificuldades na implementação dessas medidas.

O trabalho ressalta que existem pesquisas sendo feitas na utilização de determinados algoritmos de criptografia em bancos de dados específicos, que concluem que, de fato, a perda de desempenho para operações de grande alcance é severa. O autor afirma então que, são necessárias mais pesquisas que abordem mais algoritmos de criptografia aplicados em diferentes bancos de dados.

O objetivo do trabalho é avaliar o impacto dos algoritmos de criptografia no desempenho de diferentes SGBDs e mensurar o consumo de recursos que ocorre. O trabalho se propôs a utilizar um algoritmo de criptografia simétrica, AES, para aplicar criptografia em nível de banco de dados, empregando Transparent Data Encryption (TDE) para proteger todo o banco de dados enquanto protege os dados em repouso. O objetivo

do autor se assemelha ao objetivo do trabalho aqui apresentado no sentido de avaliar o impacto causado pelo emprego de criptografia simétrica no desempenho de SGBDs, porém se difere na forma de utilização. O recurso para emprego de TDE não está presente em todos SGBDs, como no Firebird, que para habilitar o mesmo precisa de um plugin proprietário externo (IBSURGEON, 2023). O trabalho aqui apresentado visa aplicar criptografia simétrica por meio de funções nativas dos SGBDs avaliados, para campos específicos das tabelas, empregadas ao nível da aplicação.

Para expandir os tipos de SGBDs abordados, o autor adotou sistemas de duas categorias diferentes, SGBDs Relacionais e NoSQL. Como representantes dos SGBDs relacionais, foram utilizados o MySQL 8.0 e o Oracle 12c. Para representar a categoria NoSQL, foram utilizados o MongoDB e o Cassandra. Foi desenvolvida uma aplicação web para se conectar aos SGBDs. O trabalho aqui proposto limitou seu escopo a SGBDs relacionais open source, empregando também uma aplicação desenvolvida para realizar a conexão e as operações com os SGBDs.

A metodologia de pesquisa adotada pelo trabalho foi uma mistura de pesquisa descritiva e experimental. A respeito da parte de pesquisa descritiva, o autor se baseia em revisão de registros existentes e materiais da literatura.

A respeito da pesquisa experimental, foram realizados experimentos com o AES utilizando chaves de 128 e 256 bits, para medir seu desempenho. Foram avaliados o tempo de resposta e uso de CPU no emprego do algoritmo de criptografia. Foi utilizada uma ferramenta de análise de desempenho, capaz de se conectar aos dois tipos de SGBDs, que comparava os resultados enquanto as funcionalidades dos SGBDs eram executadas. As funcionalidades em questão são de inserção, consulta, atualização e remoção (Create, Read, Update, Delete - CRUD). As métricas adotadas pelo autor são próximas das adotadas pelo trabalho aqui apresentado, que avalia o tempo médio de execução das operações, a média de uso de CPU durante a execução das operações, com o acréscimo da média de memória RAM ocupada durante a execução dos testes.

Em sua abordagem, o trabalho aplicou TDE sobre toda a base de dados em ambos os tipos de SGBDs utilizados. Essa forma de criptografia abrange todos os arquivos físicos da base de dados e seus backups. O trabalho aponta, como vantagem, não haver necessidade de modificações na camada de aplicação, ser um processo invisível para o usuário e suportar espelhamento e envio de log. A abordagem adotada pelo autor tem vantagem, nesse sentido, em relação à abordagem aplicada no trabalho aqui proposto,

visto que para adesão de uma nova modelagem na base de dados ou acréscimo de um novo SGBD avaliado, haveria necessidade de modificações significativas na aplicação para realizar a adaptação da mesma.

O trabalho do autor empregou a Intel advanced CPU (Intel AES-NI) para buscar uma performance melhor. É citado um trabalho publicado pela Intel que aponta a ocorrência de espera da CPU nas operações de cifragem/decifragem, causando sobrecarga computacional e comprometendo o desempenho. O hardware traz novas instruções para auxiliar no problema. No trabalho citado, essa CPU não foi testada em outros algoritmos além do AES e também não foi testada em outros SGBDs além do Oracle. O trabalho aqui apresentado não utilizou nenhum hardware com propósito específico para auxiliar de emprego de criptografia durante os testes.

A ferramenta web foi construída utilizando java web com Apache Tomcat. Para mensurar a performance e recursos utilizados pelos SGBDs, foram utilizadas as ferramentas de desenvolvimento web do Internet Explorer. Para realizar os testes de carga, foi utilizada a ferramenta ApacheBench. A base de dados utilizada para teste foi construída com a estrutura de apenas uma tabela nos bancos de dados relacionais, com os mesmos tipos de atributos. Nos SGBDs NoSQL, foi utilizada também apenas uma estrutura de dados, porém com diferentes atributos de um SGBD para outro. No trabalho aqui apresentado, foi utilizada a ferramenta Atop para monitorar e coletar os resultados dos testes executados pela aplicação desenvolvida. A base de dados utilizada pelo trabalho aqui apresentado foi modelada utilizando 5 tabelas diferentes. Essa modelagem pode ser vista no capítulo 4 e foi a mesma adotada em todos os SGBDs testados.

Os experimentos foram realizados executando as operações de CRUD nos SGBDs com as instruções AES-NI ativadas. Também foram executados testes executando operações CRUD nos SGBDs criptografados com TDE, utilizando diferentes tamanhos de chaves do AES (128 e 256 bits), com as instruções AES-NI ativas e não ativadas. As ferramentas para mensurar os dados foram mantidas para que os resultados fossem comparados posteriormente.

Os testes de estresse através do ApacheBench foram executados apenas nas operações de consulta em todos os SGBDs. O autor fez essa opção com justificativa de tempo disponível no seu cronograma e pelo fato de que essas operações não demandariam nenhuma mudança específica na aplicação web e não daria nenhuma vantagem a um SGBD sobre os demais. O trabalho não especifica claramente a quantidade de

repetições ou cargas utilizadas nos testes de operações CRUD e testes de carga com o ApacheBench.

Ao serem comparados os resultados dos testes de tempo de resposta e utilização de CPU para o AES-128 com o AES-NI ativo, os SGBDs relacionais consumiram mais tempo na inserção de registros que os SGBDs NoSQL. No entanto, nas operações de remoção os SGBDs relacionais tiveram melhor desempenho. Nas operações de atualização, o Oracle foi melhor que todos os outros SGBDs, sendo o MySQL o pior.

Os resultados para a mesma comparação anteriormente citada, utilizando AES-256, mostram um acréscimo de tempo e utilização de CPU de todos os SGBDs, sendo o Cassandra o menos afetado. Os SGBDs relacionais tiveram uma piora significativa de desempenho nas operações de remoção, sendo o Oracle mais efetivo que os demais na operação de Update. O Cassandra teve desempenho superior em todos as operações que o MongoDB, tanto em relação a consumir menos tempo de resposta, quanto a menor percentual de CPU ocupada.

Os testes de estresse realizados com o ApacheBench para as operações de leitura, comparando os resultados entre o AES-128 e AES-256, mostram que os SGBDs relacionais tiveram um aumento médio de consumo no tempo de resposta de 20%, enquanto os NoSQL tiveram um acréscimo de 6%. No geral, todos os SGBDs tiveram melhor desempenho com o AES-128. Concluiu-se também que os SGBDs NoSQL têm o desempenho menos afetado pelo emprego da criptografia que os SGBDs relacionais.

Os resultados dos testes sem a utilização das instruções AES-NI nas operações CRUD, mostram que todos os SGBDs tiveram desempenho superior com AES-128, sendo o Oracle mais afetado quando utilizado AES-256. Os SGBDs NoSQL tiveram melhor desempenho em todas as operações em comparação aos relacionais, sendo o Cassandra mais efetivo que o MongoDB com AES-128 e um pouco inferior que o mesmo ao utilizar AES-256.

Ao avaliar os resultados dos testes de estresse para operações de consulta, usando o ApacheBench com as instruções AES-NI desativadas, foi observado um aumento médio de 16% no tempo consumido do AES-128 para o AES-256. O Oracle e o Cassandra foram os mais afetados com a variação de chave. O MySQL teve um desempenho pior que o Oracle com o AES-128. Nesse contexto, a categoria NoSQL também foi superior à relacional, sendo o Cassandra mais efetivo que o MongoDB.

Nas suas conclusões, o autor não faz nenhuma afirmativa com relação aos

comparativos entre os SGBDs, mas ressalta a melhoria de desempenho quando ativadas as instruções AES-NI. O trabalho aqui apresentado visa contribuir com a indicação de uma opção de SGBD relacional dentre os avaliados, tendo em vista a comparação das métricas adotadas considerando a relação 'Segurança X Desempenho'. O tipo de criptografia adota pelo autor, a TDE, permitiu que fossem feitos testes observando a variação do tamanho de chaves, a qual é uma limitação do trabalho aqui apresentado. Porém, não considerou o modo de operação utilizado pelos SGBDs como característica relevante, visto que não foi explicitamente abordado em seu texto, mesmo que pode isso possa ser um fator de decisão quando se busca maior segurança dos dados.

3.3- Understanding and Benchmarking the Impact of GDPR on Database Systems (SHASTRI et al., 2019)

O trabalho está inserido no contexto da necessidade que as organizações têm de atender as definições legais estabelecidas pela GDPR, no tratamento de dados que têm acesso. Os autores elencam alguns aspectos da lei que tornam a tarefa de adequação à mesma difícil, sendo um desses aspectos, o fato de que vários requisitos da GDPR fazem contraste aos princípios de design e práticas operacionais dos sistemas de computação modernos.

O objetivo do trabalho é fazer uma análise da GDPR pela perspectiva do SGBD, para avaliar o quão aderente à GDPR são os SGBDs, em suas características e funcionalidades e o impacto que a adequação a essa legislação causa no desempenho dos mesmos. Os autores afirmam que a lei, em alguns casos, é ambígua. Nesses casos, eles adotaram como interpretação o pior caso possível, em relação ao impacto a se causar no desempenho para se atingir a conformidade com a GDPR. O emprego da criptografia visa proporcionar anonimização dos dados pessoais, a qual é uma das demandas da LGPD, legislação do Brasil inspirada na GDPR. O objetivo do trabalho aqui proposto, de avaliar o desempenho de funções criptográficas nativas dos SGBDs open source selecionados, busca contribuir com a indicação de uma dessas tecnologias que sofra menos impacto em termos de custo computacional ao realizar a tarefa de anonimizar os dados através da criptografia, embora não seja parte do escopo avaliar a capacidade de aderência dos

mesmos a legislação na completude de recursos que oferece.

Ao adotar a interpretação do pior caso possível para se atingir conformidade com a GDPR, os autores identificaram 3 pontos-chave. No primeiro, identificaram o que chamaram de explosão de metadados, onde:

> "cada item de dados pessoais é associado a até sete propriedades de metadados (como finalidade, tempo de vida, objeções, etc.) que regem seu comportamento".

Como resultado dessa associação, os dados pessoais ganham "status" de entidade ativa, com próprias regras. Dessa forma, a GDPR não permite mais que os dados sejam tratados como mercadoria substituível, passível de alteração por outra de igual valor, espécie, qualidade e quantidade. Do ponto de vista dos autores, isso implica significativamente na forma que são realizadas as operações de controle e de caminho dos dados.

O segundo ponto-chave observado foi o contraste entre o objetivo da GDPR pela proteção de dados como design e padrão e o design tradicional de sistemas que buscam otimizar desempenho, custo e confiabilidade. Exemplificaram o caso da manutenção de auditoria dos acessos aos dados, onde toda operação de leitura desencadeia uma operação de escrita. O terceiro ponto-chave identificado foi que a GDPR permite novas formas de interação com os dados armazenados, possibilidade de foi explorada ao longo do trabalho.

Os autores desenvolveram uma ferramenta de benchmark, o GDPRbench, que representa as funcionalidades do armazenamento de dados realizado por empresas que manipulam dados pessoais. Ele tem como propósito avaliar a conformidade de SGBDs com a GDPR, fornecendo métricas quantificáveis sobre informações básicas da aderência do SGDB à GDPR e do desempenho alcançado das operações realizadas sob essa legislação. O benchmark utiliza 4 cargas de trabalho que, segundo os autores, não são capturadas por outros softwares do mesmo segmento: Controlador, Cliente, Processador e Regulador. Ele captura também 3 métricas para cada carga e trabalho: correção, tempo de conclusão e sobrecarga de armazenamento.

Para testar a conformidade dos SGBDs à GDPR, foram selecionados 3 SGBDs, o Redis, que é um SGBD NoSQL de armazenamento em memória, o PostgreSQL, que é um SGBD relacional open source e um SGBD relacional comercial empresarial. O nome real do SGBD empresarial não foi divulgado no trabalho, pois os autores não obtiveram

licença para compartilhar publicamente os resultados de benchmark deste sistema, sendo chamado então de System-C.

Os SGBDs adotados para teste foram modificados para que tivessem conformidade com a GDPR, porém sem que as modificações implicassem em melhoria de desempenho. Ao aplicar essas modificações, os SGBDs se tornaram de 2 a 5 vezes mais lentos para cargas de trabalho tradicionalmente utilizadas em outros benchmarks, devido à necessidade de monitorar e manter log das operações, por definição da GDPR. Ao avaliar esses SGBDs utilizando o GDPRbench, foi observado que as consultas desse formato retornam um grande volume de dados e metadados, o que diminui o desempenho significativamente. O software de benchmark utilizado para testes com a carga de trabalho tradicional foi o Yahoo Cloud Serving Benchmark, e os experimentos foram executados na plataforma Chameleon Cloud. Correlacionando com o trabalho aqui proposto, como carga de trabalho, esse trabalho usou um conjunto de dados públicos em formato .csv, sem nenhum tratamento específico, para simular operações CRUD com as funções criptográficas, utilizando ferramenta tradicional para aferir consumo de recurso computacional, sem que nenhuma modificação tivesse sido feita nas configurações dos SGBDs que interferisse no desempenho, permanecendo assim como são comumente utilizados. Os resultados obtidos com os testes podem ser verificados no capítulo 5.

A Figura 8 mostra uma tabela criada pelos autores para representar quais requisitos da GDPR os SGBDs analisados atendem com suas características nativas. A classificação “Partial” sinaliza que o recurso nativo do SGBD teve que ser incrementado por bibliotecas de terceiros e/ou alterações de código para atender aos requisitos do GDPR.

Figura 8 – Tabela de recursos disponíveis aderentes à GDPR

	Redis	PostgreSQL	System-C
TTL	Partial	No	No
Encryption	No	Partial	Full
Auditing	Full	Full	Full
Metadata indexing	No	Full	Full
Access control	No	Full	Full

Tabela de recursos disponíveis aderentes à GDPR (SHASTRI et al., 2019)

Em suas discussões sobre os resultados obtidos, os autores apontam que ocorre de fato uma degradação de desempenho, quando realizada as mudanças necessárias para alcançar conformidade com a GDPR. Nesse sentido, o trabalho recomenda uma

análise minuciosa dos engenheiros e administradores dos SGBD aos impactos que a implementação de conformidade à GDPR pode causar. Também concluíram que os SGBDs atuais não escalam bem para cargas de trabalho aderentes à GDPR. O PostgreSQL, nesse sentido, teve desempenho melhor que o Redis, mas também apresenta perda de desempenho conforme a carga de trabalho aumenta. O PostgreSQL também foi um dos SGBDs avaliados pela perspectiva do trabalho aqui proposto, sendo possível observar a reação do seu desempenho ao aumento de carga, mesmo que com cargas de dados tradicionais, no capítulo 5, principalmente em relação ao tempo médio de execução para as maiores faixas de carga de trabalho, que corrobora com os resultados encontrados pelos autores.

Os autores também concluem que é mais fácil atingir a conformidade com a GDPR nos SGBDs relacionais que nos NoSQL. O Redis precisou de duas alterações no design interno, enquanto os outros dois precisaram apenas de mudanças de configuração e scrpits externos. O Redis também apresentou uma queda de desempenho maior que os relacionais.

4- Metodologia

Neste capítulo, será apresentada a metodologia utilizada para conduzir a análise de desempenho dos SGBDs utilizando função criptográfica nativa. Serão apresentadas a estrutura dos testes realizados, a motivação das escolhas feitas ao selecionar o algoritmo de criptografia simétrica, dos modos de operação e o impacto que as diferenças e limitações que os SGBDs possuem entre si que influenciaram essas decisões.

O objetivo desse trabalho é avaliar o desempenho de funções criptográficas nativas dos SGBDs MariaDB, FireBird e PostgreSQL. Para isso, serão aplicadas diferentes faixas de cargas de trabalho aos SGBDs através de uma aplicação desenvolvida, utilizando instruções SQL de inserção, consulta, atualização e remoção de registros.

As cargas de trabalho utilizadas serão obtidas de dados lidos em um conjunto de dados contido em um arquivo do formato .CSV. As faixas de carga de trabalho serão de 1000 registros, 2000 registros, 4000 registros e 8000 registros.

Serão realizados testes em claro e utilizando as funções criptográficas. Dessa forma, será possível avaliar o impacto que as funções criptográficas causam no desempenho do SGBD em questão assim como também realizar o comparativo dos resultados alcançados entre os 3 SGBDs.

Os critérios utilizados para avaliar o desempenho das funções criptográficas serão o tempo médio de execução gasto na execução das instruções, média de CPU utilizada e média de memória RAM ocupada durante as instruções. A aferição dessas métricas será realizada por uma ferramenta instalada no Sistema Operacional (S.O.), que irá monitorar o consumo de recursos.

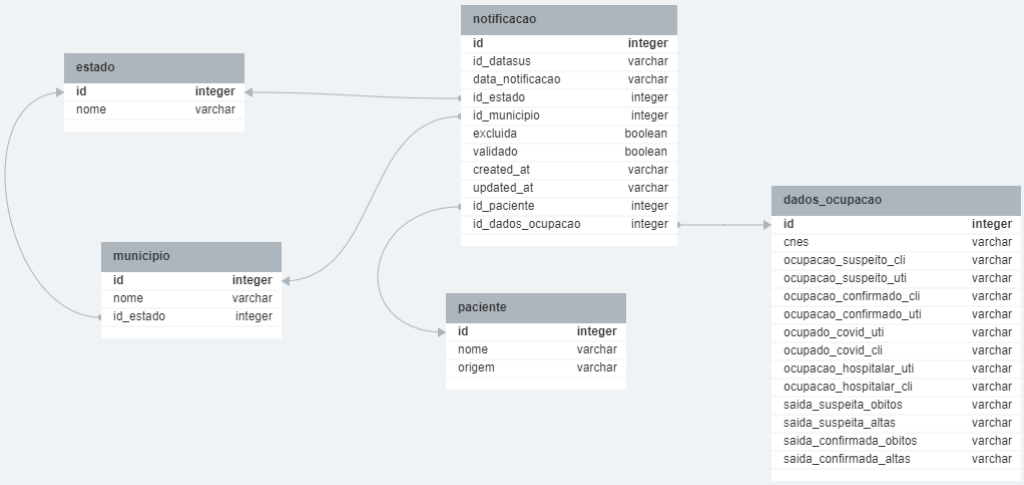
O ambiente que será utilizado para o experimento consiste em 3 máquinas virtuais de mesma configuração, cada uma com um dos SGBDs adotados. A aplicação desenvolvida será instalada em cada uma das máquinas.

4.1- Conjunto de dados utilizado

O conjunto de dados utilizado foi adquirido na plataforma openDataSus, oficial do Ministério da Saúde brasileiro (OPENDATASUS, 2022). Esses dados são referentes a registros de ocupação hospitalar durante a pandemia de COVID-19, contendo dados de diversas regiões do Brasil. Foram reunidos dados de ocupação de leitos clínicos e de Unidade de Terapia Intensiva (UTI) do SUS, utilizados para atender pacientes com casos suspeitos ou confirmados de COVID-19 (ocupação SRAG / COVID-19). Até a conclusão desse trabalho, foi verificado que esses dados continuam a ser atualizados. A versão do conjunto de dados utilizados foi adquirida em novembro de 2022, e conta com 745015 linhas de registro do arquivo .csv.

Para retratar o conjunto de dados na camada de persistência utilizada nesse trabalho, as características presentes no conjunto de dados foram representadas em 5 entidades. O modelo relacional definido pode ser visto na Figura 9.

Figura 9 – Modelo conceitual de entidades adotado



A decisão de dividir o conteúdo do conjunto de dados em 5 tabelas foi tomada, para que se tornasse possível realizar operações condicionais nas operações de UPDATE e DELETE envolvendo 2 tabelas distintas. Dessa forma, instruções mais complexas podem ser elaboradas, exigindo mais do desempenho dos SGBDs. A base de dados foi modelada com a mesma estrutura nos 3 SGBDs[1].

[1]Bases de dados disponíveis em https://github.com/ergor86/baseDeDados

4.2- Definição do algoritmo de criptografia simétrica adotado

Esse trabalho tem como foco empregar apenas o segmento de criptografia simétrica disponibilizada pelos SGBDs adotados para estudo. Diante disso, foi feito um levantamento sobre os algoritmos suportados pelos SGBDs, visto na Tabela 1.

Algoritmo	MariaDB	PostgreSQL	FireBird
AES	Sim	Sim	Sim
Blowfish	Não	Sim	Sim
3DES	Sim [1]	Sim [2]	Não

[1] Apenas quando configurado com TLS support.
[2] Apenas com extensão OpenSSL.

Tabela 1 – Algoritmos de criptografia simétrica suportados pelos SGBDs

Ao realizar o levantamento, verificou-se que apenas o algoritmo AES é suportado por todos os SGBDs, sendo então o selecionado para estudo. Uma ressalva a ser feita é que, como apontado no capítulo 2, o MariaDB só suporta nativamente a versão AES-128, essa será a única versão de chaves utilizada.

Após a definição do AES como algoritmo adotado pelo trabalho, foi verificado também qual modo de operação poderia ser adotado. O único modo suportado pelo MariaDB é o ECB (MARIADB, 2023b), como abordado no capítulo 2 ao abordar o SGBD.

No caso do Firebird, ele suporta o CBC, ECB e o OFB. Porém, para utilizar o ECB e o CBC, o texto de entrada precisa ser de tamanho múltiplo da cifra de bloco utilizada. Caso o texto não corresponda a esse pré-requisito, o ajuste fica a cargo do usuário, que precisa preencher o espaço restante com o número 0 ou caractere de espaço vazio. Houve esforço para tentar produzir uma rotina que ajustasse a entrada o tamanho correto, porém não houve sucesso. Essa limitação foi constatada com o desenvolvimento de código da aplicação utilizada para os testes já em andamento avançado e com a modelagem da estrutura da base já estabelecida para os dados utilizados como carga de trabalho citados na seção 4.1. Uma mudança brusca nos dados utilizados como carga de trabalho implicaria numa grande mudança do código como um todo e na modelagem da base. Com intuito de utilizar os dados de base pública sem alteração e pela limitação de tempo para conclusão do trabalho, foi adotado o OFB como modo de operação para o Firebird. Nesse modo, o uso do vetor de inicialização é obrigatório, que também deve

corresponder ao tamanho do bloco do algoritmo utilizado (FIREBIRD, 2023a).

Em relação ao PostgreSQL, como abordado no capítulo 2, existem as possibilidades de aplicar o algoritmo AES através das funções de criptografia PGP e das funções de criptografia bruta. As funções de criptografia PGP não permitem a definição de um modo de operação diferente do CBC e também não permitem a definição de um vetor de inicialização fixo, como abordado no capítulo 2. As funções de criptografia bruta permitem a utilização do modo de operação ECB e CBC, também permitindo a definição do vetor de inicialização manualmente (POSTGRESQL, 2023b).

Do ponto de vista de segurança, utilizar o vetor de inicialização fixo não é recomendável (STALLINGS, 2015). Porém, o propósito do trabalho é avaliar o desempenho das funções criptográficas sob as condições mais próximas possíveis. Como no Firebird irá ser utilizado o vetor de inicialização fixo, no PostgreSQL será adotada a versão das funções de criptografia bruta, com modo de operação CBC. A mesma chave de encriptação também será usada nas funções de todos os algoritmos. Dessa forma, a configuração adotada em relação ao algoritmo utilizado, tamanho da chave criptográfica empregada e do modo de operação pode ser vista na Tabela 2.

	MariaDB	**PostgreSQL**	**Firebird**
Algoritmo	AES	AES	AES
Tamanho da chave	128 bits	128 bits	128 bits
Modo de Operação	ECB	CBC	OFB

Tabela 2 – Configuração adotada para a utilização do AES

A sintaxe das funções adotadas seguem exemplificadas na Tabela 3. A função de cifragem do MariaDB recebe como entrada dados no formato *string*, e tem como saída dados no formato *binary string*. A função do PostgreSQL recebe e retorna dados no formato *bytea*. A função do Firebird pode receber dados no formato *blob* ou *binary string*, retornando no formato *BLOB SUB_TYPE BINARY* para entradas *blob* e *VARBINARY* para entradas *binary string*. Essa relação de tipos de entradas e saídas paras as funções de encriptação pode ser vista na Tabela 4.

A função de decriptação do PostgreSQL tem a necessidade de ter o resultado decifrado submetido a uma transformação de tipo, um *cast*, para o tipo Bytea, sinalizada no próprio parâmetro de entrada da função, e a saída convertida para ASCII, como exemplificado na Figura 11. Do contrário, o SGBD lança erro.

As funções criptográficas do Firebird têm uma demanda por entradas já no formato hexadecimal. Caso esse requisito não seja atendido, o SGBD não lança erro, porém a mensagem decifrada vem com uma série de caracteres não inteligíveis associados. Passando a entrada no formato hexadecimal, essa falha na decriptação não ocorre. Porém, para se ter acesso ao texto em claro de forma inteligível, é necessário que a saída da função seja convertida de hexadecimal para binário de forma externa, já que a função não trara disso. Essas informações acerca do Firebird não constam na documentação (FIREBIRD, 2023a) de forma clara. Para descobrir esses aspectos foi demandado muito tempo de pesquisa, pois realizar os comandos com o texto decriptado, carregado com os caracteres indesejados, causaria discrepância entre as instruções SQL utilizadas no Firebird com as instruções dos outros SGBDs.

Algoritmo	MariaDB	PostgreSQL	FireBird
AES - encriptação	AES_ENCRYPT (str,key_str)	ENCRYPT_IV(data bytea, key bytea, iv bytea, type text) returns bytea *type text = ‘aes-cbc’	ENCRYPT (input USING <algorithm> [MODE<mode>] KEY key [IV iv] [<ctr_type>] [CTR_LENGTH ctr_length] [COUNTER initial_counter])
AES - decriptação	AES_DECRYPT (crypt_str, key_str)	DECRYPT_IV(data bytea, key bytea, iv bytea, type text) returns bytea *type text = ’aes-cbc’	DECRYPT (encrypted_input USING<algorithm> [MODE <mode>] KEY key [IV iv] [<ctr_type] [CTR_LENGTH ctr_length] [COUNTER initial_counter])

Tabela 3 – Sintaxe das funções criptográficas utilizadas.

SGBD	Tipo de entrada	Tipo de retorno
MariaDB	string	binary string
PostgreSQL	bytea (binary string)	bytea
Firebird	blob ou binary string	varbinary ou blob

Tabela 4 – Tipos de entradas e retornos das funções de cifragem de cada um dos SGBDs

4.3- Aplicaçâo utilizada para os experimentos

A aplicaçâo desenvolvida[2] para os experimentos tem como finalidade executar as instruções SQL direcionadas ao SGDB, com os dados em claro e utilizando as funções criptográficas. A aplicação foi construída utilizando PHP (PHP: Hypertext Preprocessor), uma linguagem de *script* open source de uso geral, porém muito aplicada para desenvolvimento web (GROUP, 2023).

O paradigma de programação adotado foi o de Programação Orientada a Objetos (POO), com as classes separadas por responsabilidades, com a organização inspirada no padrão de projeto MVC (Model, View, Controller). O padrão MVC não foi utilizado integralmente na implementação, pois não serão utilizadas páginas HTML ou classes para a camada de visão e, consequentemente, de controle.

Foi definido um grupo de classes chamado *model*, inspiradas na Model do MVC, que contem classes para armazenar os dados referentes as 5 tabelas modeladas na camada de persistência, descritas na seção 4.1.

O grupo de classes do pacote *dao*, que fazem alusão ao padrão Data Access Object (DAO), reúne as classes para fazer o acesso e as operações entre a aplicação e os SGBDs. As classes do grupo chamado *service* fazem uso das classes *dao* e *model* para realizar as funcionalidades de inserção, busca, atualização e remoção nos SGBDs. A estrutura principal dos diretórios da aplicação pode ser vista na Figura 10a.

Dentro do diretório das classes *dao*, existe um diretório voltado para cada SGBD, contendo as classes para a interação com SGBD em questão, empregando as funções criptográficas. As classes que fazem a conexão com os SGBDs e fazem interação com o banco, sem empregar criptografia, estão fora dos diretórios individuais dos SGBDs, pois são aplicadas a todos. A organização do diretório *dao* pode ser visto na Figura 10b. Um exemplo de instrução SQL aplicada por um método da classe 'EstadoDaoPostgreSql.php' pode ser visto na Figura 11.

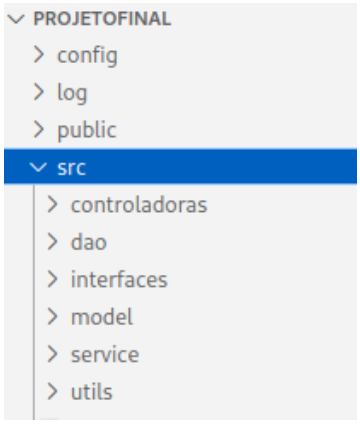
Existe uma classe *dao* para cada classe de modelo, com métodos de inserção, busca e remoção. O único grupo de classe *dao* que possui método de alteração é o que interage com a tabela 'estado', por decisão de projeto que será abordada mais adiante.

O diretório *service* contêm um diretório para cada SGDB. Dentro do diretório de

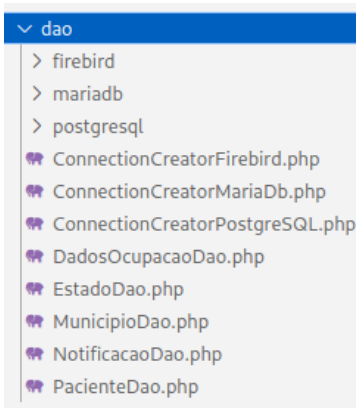
[2]A aplicação está disponível em: https://github.com/ergor86/aplicacaoTCC/tree/main

cada SGBD, existe uma classe de serviço para cada operação com a base de dados, sendo que existe uma service para as operações utilizando a função criptográfica do SGBD e uma versão da classe de serviço realizando as operações com os dados em claro. Fora dos diretórios dos SGBDs, existe uma classe de serviço chamada 'ServiceArquivo.php'. Essa classe possui o método que faz a leitura do arquivo .csv, que contêm o conjunto de dados utilizados para carga de trabalho nos testes. Essa classe também possui o método que lê o conteúdo do arquivo *rounds.conf*, que se encontra dentro do diretório *config*, na raiz do projeto. O arquivo *rounds.conf* contêm a quantidade de linhas de registros que devem ser lidas do arquivo .csv para execução dos testes. A estrutura do diretório *service* pode ser vista na Figura 10c.

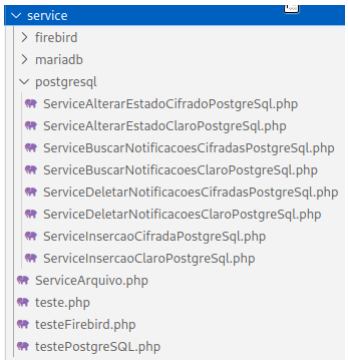
Figura 10 – Estrutura dos diretórios principal, DAO e Service.



(a) Estrutura principal dos diretórios da aplicação



(b) Estrutura do diretório *dao*



(c) Estrutura do diretório *service*

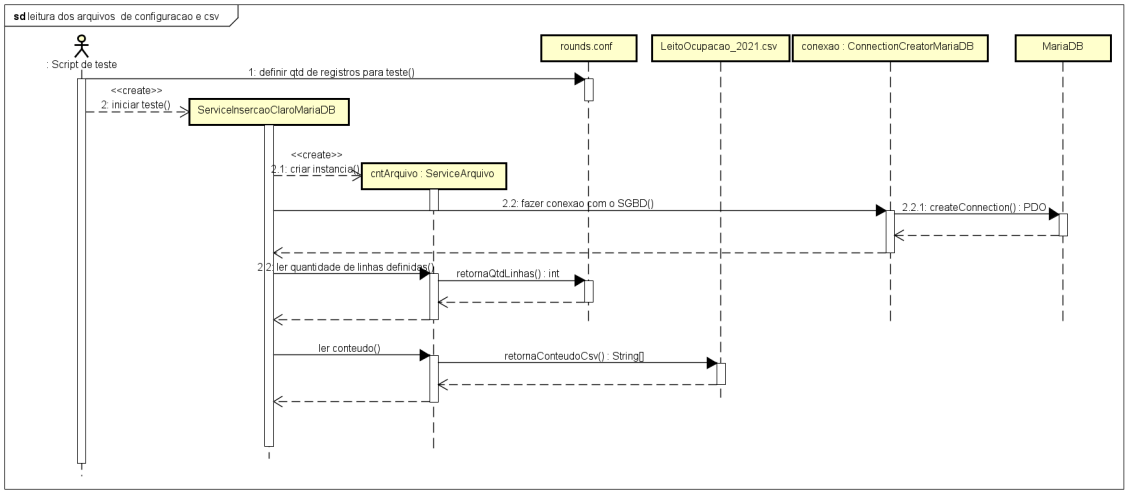
Figura 11 – Exemplo de instrução SQL utilizando função de decifrar dados no PostgreSQL

```
$sqlSelect = "SELECT CONVERT_FROM(decrypt_iv(nome::bytea, :key, '0123456789123456', :algoritmo),
        'SQL_ASCII') as nome
            from estado where id =:id";
```

Para melhor compreensão do funcionamento dessas classes de serviço, foram definidos diagramas de sequência que exemplificam as etapas que elas percorrem ao longo da sua execução. Foram feitos diagramas de sequência para as classes de serviço de inserção, alteração e exclusão, por possuírem interações mais complexas. Nesses diagramas, foram suprimidos os argumentos dos métodos e mensagens de texto que retornam da classe de serviço para o prompt de execução do script, para não tumultuar a imagem. O fluxo da mesma classe de serviço foi dividido em diagramas separados para melhor dimensionamento e compreensão das imagens.

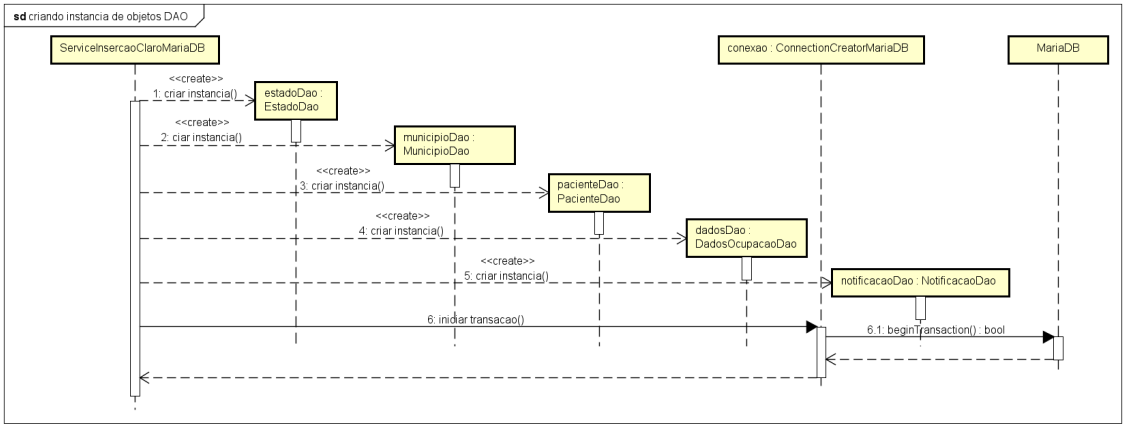
De forma geral, as classes de serviço de inserção iniciam sua execução fazendo a leitura da quantidade de linhas desejadas do arquivo .csv, através da utilização de objeto da classe 'ServiceArquivo.php'. No caso de inserção de registros, como exemplo, temos as etapas de inserção com dados em claro no MariaDB, visto que o processo é o mesmo na execução com os demais SGBDs, mesmo aplicando as funções criptográficas nas instruções SQL. A Figura 12 demonstra o início do processo, com a inserção de conteúdo e leitura do arquivo *rounds.conf*, que tem em seu conteúdo a quantidade de registros a serem utilizados no teste eo estabelecimento de conexão com o SGDB. Esse conteúdo é definido como parâmetro do script que dispara o teste de inserção. Após a leitura desse arquivo, a respectiva quantidade de linhas de registros é lida do arquivo .csv e seu conteúdo armazenado num *array*.

Figura 12 – Leitura do arquivo rounds.conf e do arquivo de carga de dados.



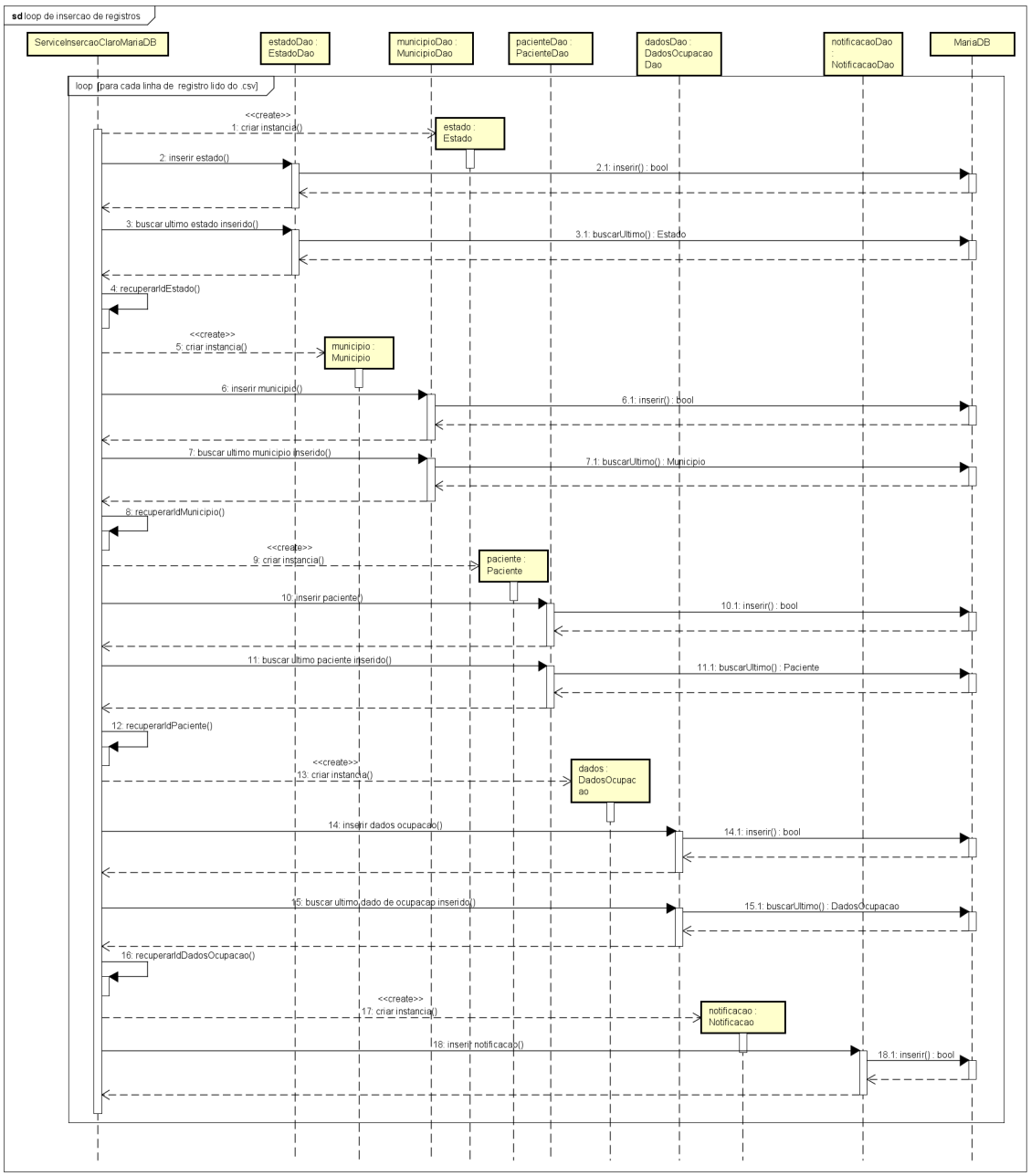
Tendo sido feita a leitura dos dados para carga de trabalho a partir do arquivo .csv, são instanciados os objetos do grupo de classes *dao* e é iniciada a transação para dar início as operações com o SGBD. Essa etapa pode ser vista na Figura 13

Figura 13 – Definição dos objetos para interação com o SGBD e início da transação



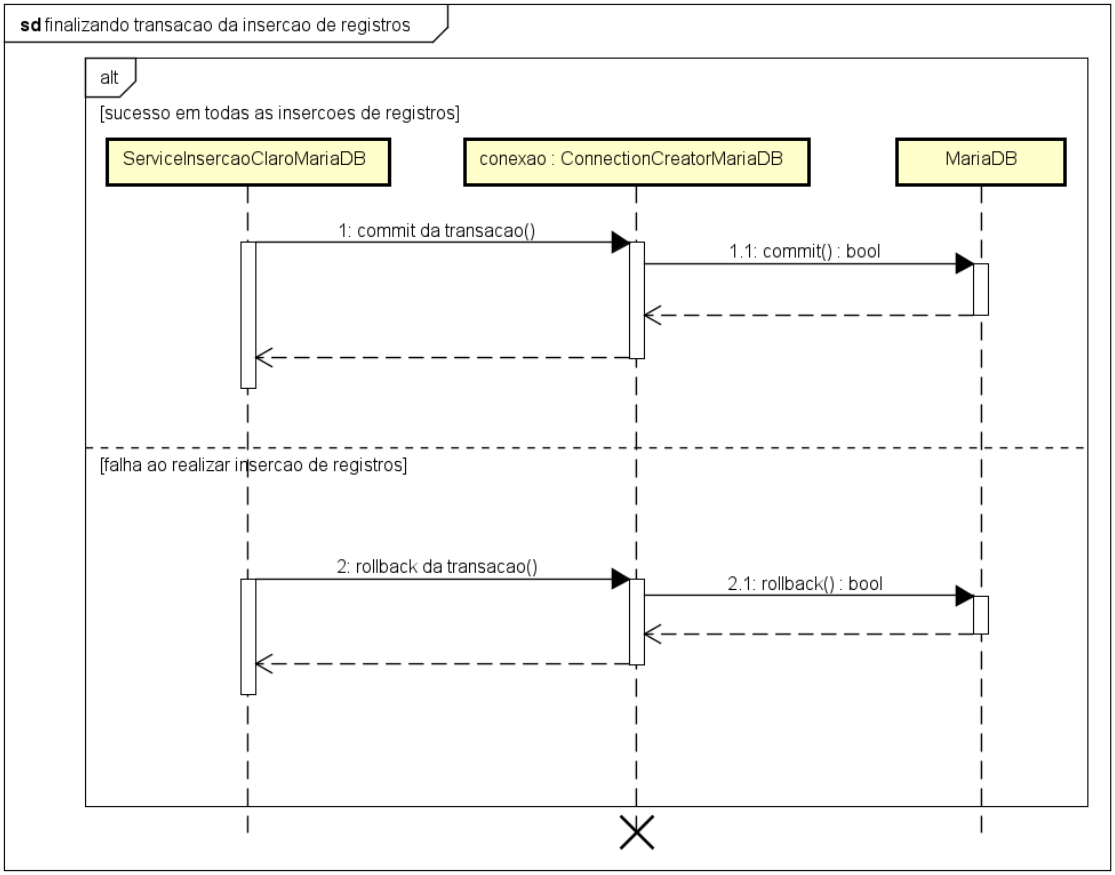
Após isso, para cada linha lida, em um laço de repetição, são instanciados objetos das classes de modelo, com base no conteúdo retornado do conjunto de dados do arquivo .csv. Esse processo inicia pela definição do objeto da classe Estado, e inserção de seu conteúdo na base de dados por meio de método de um objeto pertencente ao grupo das classes *dao* da tabela correspondente. É feita então uma busca do conteúdo desse último conteúdo inserido, agora com a informação adicional do id atribuído pelo SGBD. Esse processo se repete para todas as outras classes de domínio, com exceção da classe 'Notificacao'. O objeto da classe 'Notificacao' recebe como parâmetro, o conteúdo do conjunto de dados referente a notificação, e o id de todos os outros objetos, que inseridos na base de dados são chaves extrangeiras para os registros de suas respectivas tabelas. Após instanciado o objeto, seu conteúdo também é inserido na base por sua respectiva classe de *dao*. A função utilizada para cada inserção realizada retorna um valor booleano para indicar o sucesso ou falha da operação. Esse processo pode ser visto na Figura 14.

Figura 14 – Inserção dos registros de carga de trabalho no SGBD



Após a inserção de todos os registros, é verificado se houve sucesso em todas as operações realizadas. Caso essa condição seja atendia, é feito o *commit* da transação, do contrário, é feito o *rollback* do processo. Essa verificação pode ser vista na Figura 15.

Figura 15 – Finalizando a transação de inserção.



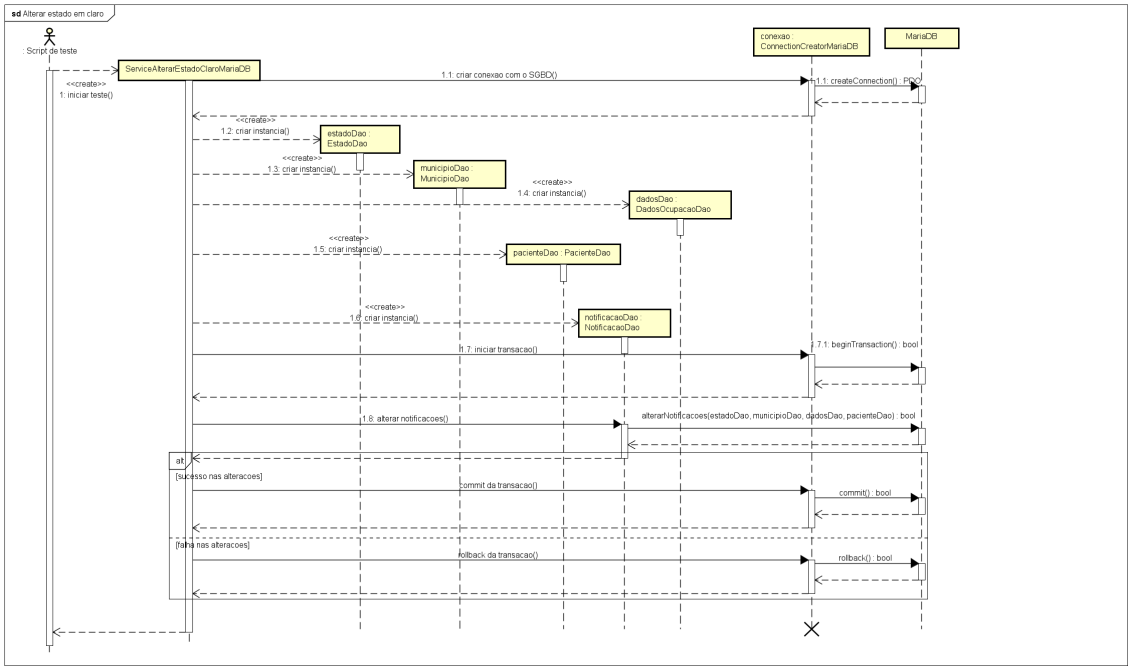
As classes de serviço de busca no SGBD utilizam um objeto *dao* de Notificacao para buscar todas as notificações inseridas na base de dados. Ele recebe como parâmetro, além da chave criptográfica na sua versão para dados cifrados, um objeto *dao* de cada uma das outras classes. É feita uma busca de todas as notificações armazenadas. A partir do registro de cada uma, são feitas buscas dos demais registros utilizando a chave estrangeira como referência e montado cada objeto que compõe a notificação. Ao fim, o método retorna um *array* de notificações.

As classes de serviço de alteração são voltadas para alterar o estado atrelado a cada notificação. Essa decisão foi adotada com o intuito de possibilitar fazer comparações condicionais utilizando duas tabelas diferentes, para adicionar complexidade à instrução SQL. Como exemplo, temos diagramas de sequência com etapas envolvidas no teste de alteração de registros no MariaDB com dados em claro, utilizando a classe de serviço 'ServiceAlterarEstadoClaroMariaDB'. O fluxo de execução é igual nos demais SGBDs,

mesmo quando aplicadas as funções de criptografia nas instruções SQL.

De forma geral, o fluxo de execução começa com o script de teste de alteração chamando a classe de serviço, que realiza a conexão com o SGBD e cria uma instância de cada classe *dao* necessária para realizar a alteração dos registros. Após criadas as instâncias, é iniciada a transação com o SGBD para realizar as operações de alteração. Para realizar as alterações, a classe de serviço aciona a instância da classe 'NotificacaoDao', que possui o método para executar a operação de alteração, chamado 'alterarNotificacoes()'. Esse método retorna um valor booleano que varia de acordo com sucesso ou falha ao realizar as alterações. É feita a verificação do valor retornado e, em caso de sucesso, é feito o *commit* da transação, no caso de falha, é feito o *rollback* da mesma. A Figura 16 apresenta a visão geral do fluxo de execução da classe de serviço para alteração de registros de estados vinculados a cada notificação que esteja mantida na base.

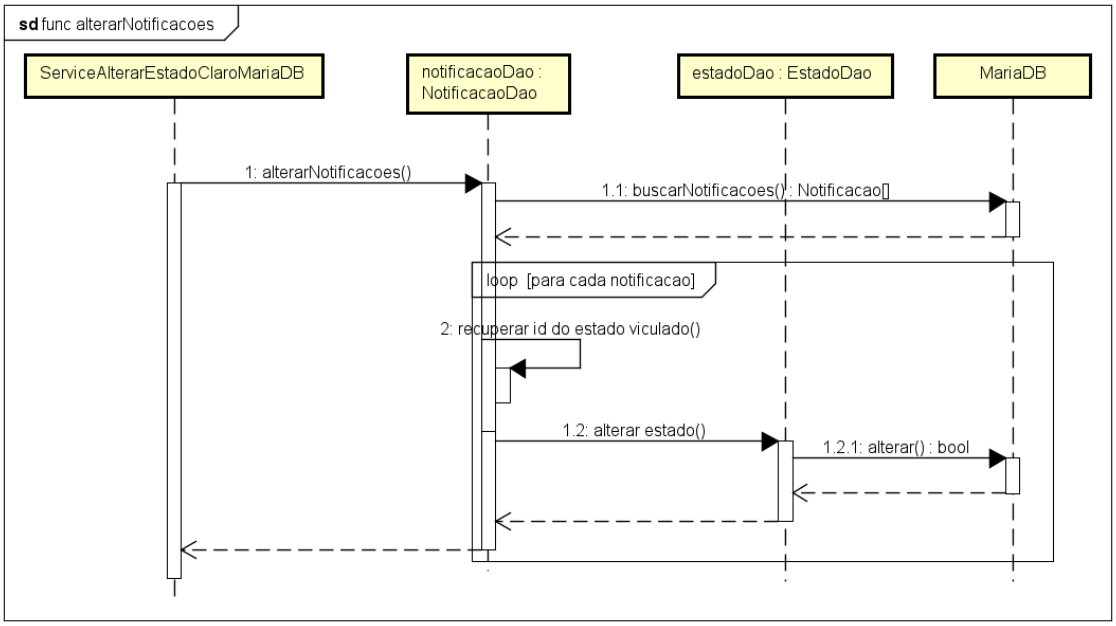
Figura 16 – Fluxo geral da alteração de registros no SGBD.



A Figura 17 apresenta o diagrama de sequência que detalha o fluxo de execução da função 'alterarNotificacoes()'. O objeto da classe 'NotificacaoDao' executa o método de alteração, que recebe como parâmetro um objeto *dao* das demais classes do grupo *dao*. O método de alteração executa o método de busca de todas as notificações. De

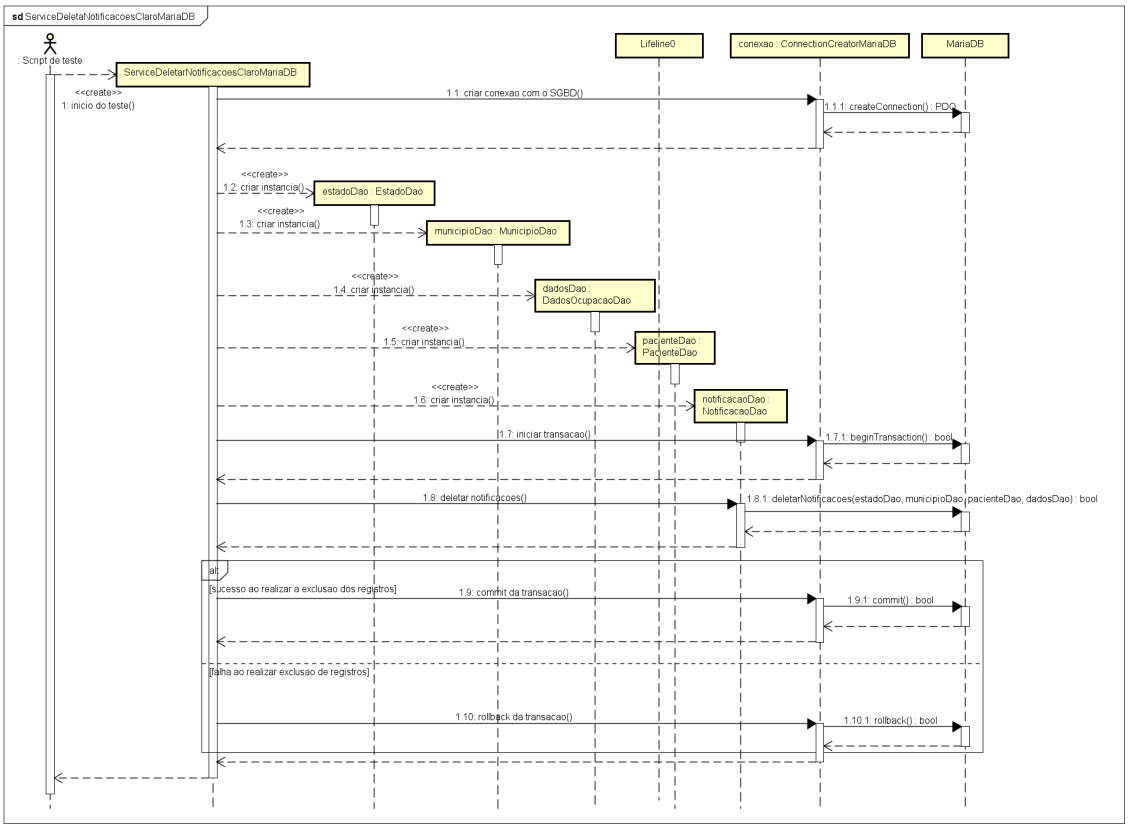
posse do *array* de notificações, ele percorre cada uma delas e através do id do estado atrelado, utiliza um objeto da classe ’EstadoDao’ para modificá-lo. No método alterar de estado, é verificado se o estado atrelado à notificação é o estado de Alagoas, caso não seja, é mudado o nome para tal.

Figura 17 – Fluxo da função de alteração de registros no SGBD.



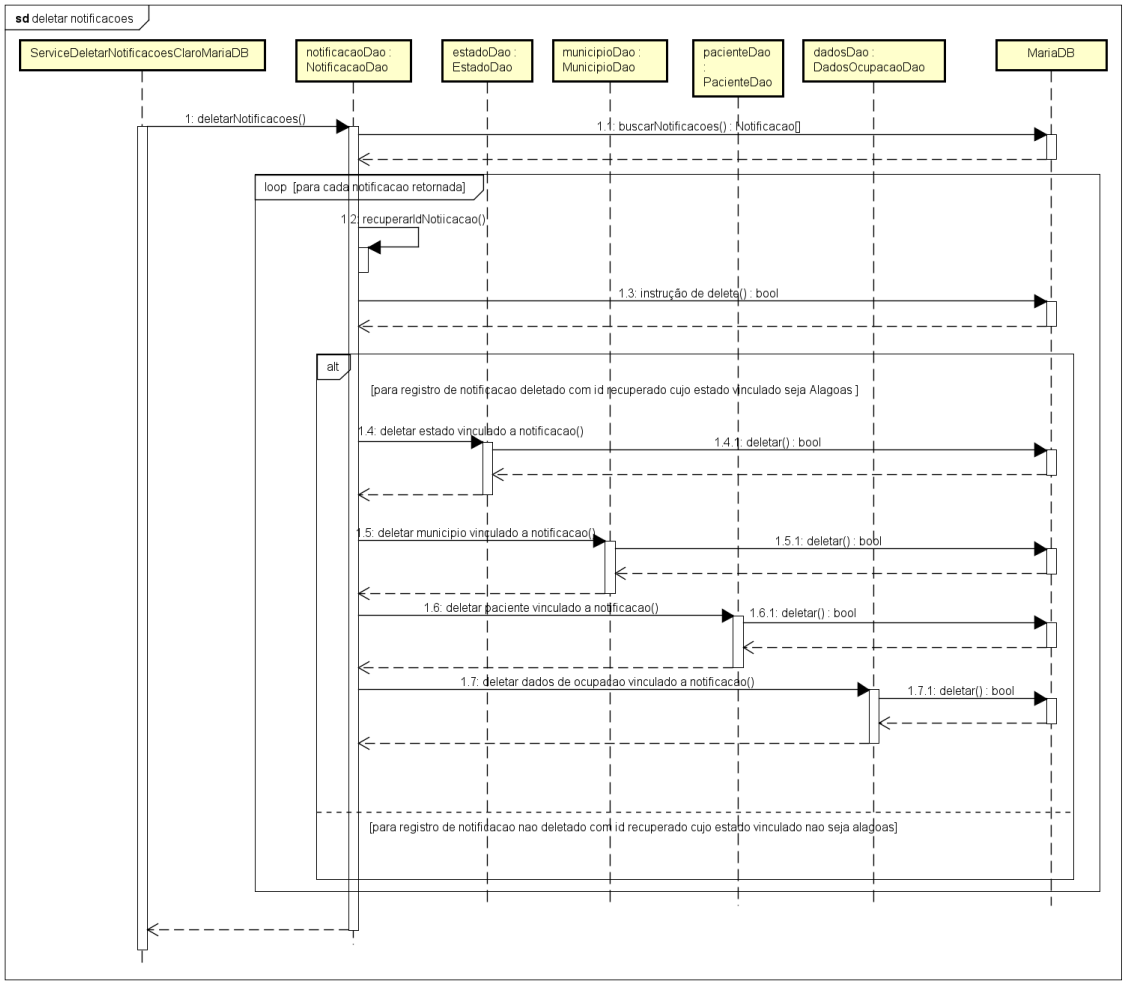
As classes de serviço de deletar registros tem um princípio de funcionamento parecido com as de alteração. A diferença se dá quando a classe de serviço de deletar registros executa o método ’deletarNotificacoes()’ de um objeto da classe ’NotificacaoDao’, que recebe como parâmetro um objeto das demais classes do grupo *dao*. A apresentação, de um ponto de vista geral, do fluxo de execução envolvendo a classe de serviço ’ServiceDeletarNotificacoesClaroMariaDB’ pode ser vista na Figura 18.

Figura 18 – Fluxo geral de remoção de registros no SGBD.



A Figura 19 apresenta o fluxo que envolve a função 'deletarNotificacoes()'. O método de deletar executa o método de busca de todas as notificações. Para cada notificação do *array* de notificações retornado, é executada uma instrução SQL de delete, que verifica se o estado atrelado à notificação tem o nome de Alagoas. Sendo a condicional verdadeira, é deletada a notificação e a partir do id dos demais objetos que compõem o objeto notificacao, são removidos os registros associados das demais tabelas, utilizado o método de deletar de cada outro objeto das classes *dao*.

Figura 19 – Fluxo da função de remoção ’deletarNotificacoes()’.

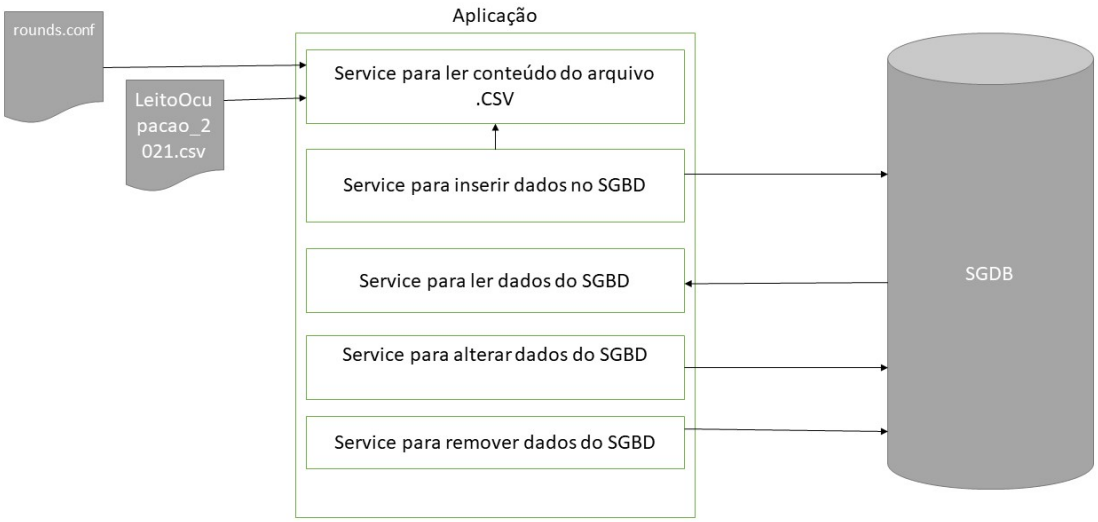


As instruções SQL utilizadas nos métodos das classes *dao* possuem a mesma estrutura para todos os SGBDs, se diferenciando apenas na função criptográfica nativa de cada um. No caso do Firebird, em razão da necessidade dos dados a serem cifrados/decifrados estarem em hexadecimal, como apontado na seção 4.2, foram utilizadas funções do PHP para tal. Porém, essa conversão só foi feita nos métodos de inserir e alterar o nome do estado, onde a condição de verificação necessita de comparar as strings para executar a operação. A conversão não foi feita para os demais campos de todas as tabelas para não aumentar o processamento de dados realizados no Firebird de forma ainda mais discrepante dos demais SGBDs. Todos os campos das tabelas de conteúdo textual (*varchar*) foram submetidos às funções criptográficas na execução de testes desse segmento.

A aplicação desenvolvida ficará hospedada dentro da mesma máquina virtual em

que se encontra o SGBD, sendo um SGBD por máquina. Uma visão geral da disposição da aplicação com o SGBD pode ser vista na Figura 20.

Figura 20 – Disposição da aplicação em relação ao SGBD.



4.4- Estrutura dos testes

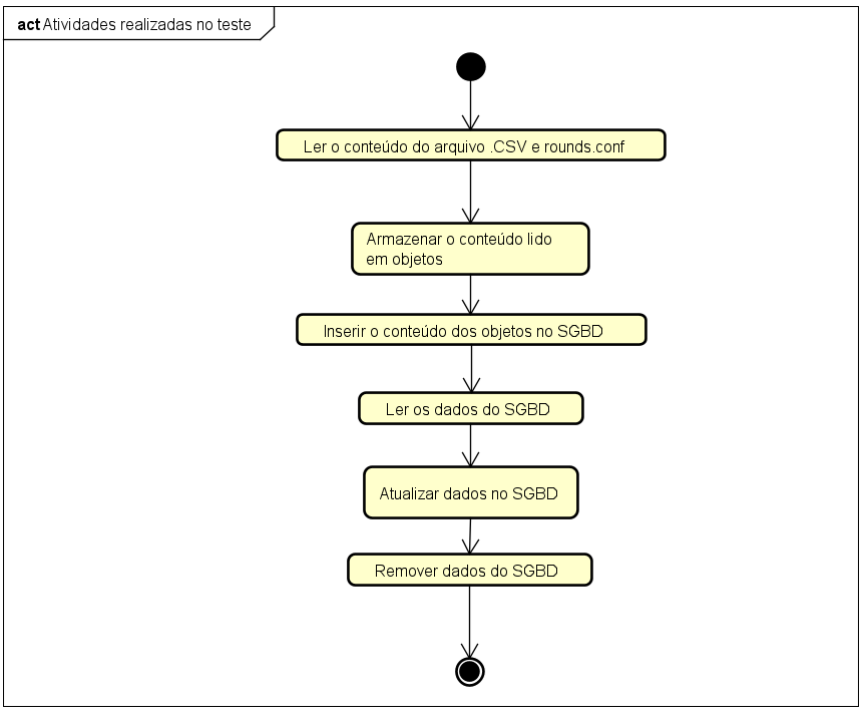
Os testes para medir o desempenho dos SGBDs foram realizados com faixas de cargas de trabalho. Foram realizados 10 testes para cada faixa de carga de trabalho. As faixas de carga são 1000 registros, 2000 registros, 4000 registros e 8000 registros. Esses testes foram realizados, na mesma quantidade, utilizando as funções criptográficas e executadas as operações com os dados em claro, para cada SGBD.

A execução de cada teste foi feita por meio de um *script* definido em *Shell Script*, rodado via terminal. O *script* recebe como parâmetro a quantidade de linhas a serem lidas do arquivo .csv para utilização nos testes. O parâmetro informado é atribuído ao conteúdo do arquivo *rounds.conf*, que será acessado pela classe de serviço ”ServiceArquivo.php”, que lê a quantidade determinada de linhas do arquivo .csv. Foi desenvolvido um *script* para execução dos testes com funções criptográficas e um *script* para operações com

textos em claro para cada SGBD.

O roteiro executado pelos testes consistiu, no primeiro momento, da execução da *service* de inserção dos dados no SGBD, depois da *service* de leitura dos dados, seguida da *service* de atualização e da *service* de remoção de dados. De um ponto de vista geral, o mapeamento das atividades realizadas na execução do teste pode ser visto na Figura 21.

Figura 21 – Atividades realizadas durante os testes



Para medir o desempenho dos SGBDs durante a execução dos testes, foram mensurados o tempo médio de execução de cada operação realizada, o percentual médio de CPU utilizado durante cada operação e a quantidade média de memória RAM ocupada durante cada operação.

Essas métricas foram capturadas utilizando a ferramenta Atop, um monitor de desempenho para Linux, que registra as atividades de todos os processos, mesmo que interrompidos durante o período de observação. Através do ponto de vista do S.O., ele pode capturar as atividades relativas a (ATOPTOOL.NL, 2023):

> uso de CPU, memória, swap, discos (incluindo LVM) e camadas de rede, e para cada processo (e thread) mostra, por exemplo, a utilização da CPU, crescimento da memória, utilização do disco, prioridade, nome de usuário, estado e código de saída

Nesse trabalho, foi definido o intervalo de 1 segundo para registro do monitoramento das métricas observadas em arquivos de log, até o fim do processo. As instruções do Atop foram utilizadas nas construções dos *scripts*. Os arquivos gerados com os resultados dos testes são armazenados no diretório *log*, que fica na raiz do projeto.

4.5- Ambiente utilizado

O ambiente que os testes foram executados utilizaram a mesma configuração na máquina virtual. A máquina virtual utilizou como S.O. a distribuição Debian GNU/Linux 11 (bullseye), contou com 4 GB de memória RAM e o processador foi limitado a utilização de apenas 1 núcleo, com o intuito de evitar vantagem de execução entre os modos de operação. Isso pelo fato de que o ECB pode ser executado em paralelo, enquanto o CBC pode ser parcialmente paralelizado e o OFB não ser paralelizável (DETOMINI, 2010).

A versão dos SGBDs utilizados foram o PostgreSQL 14.7, o MariaDB 10.8.5 e o Firebird 4.0.2. Para interagir com o PostgreSQL, foi utilizada a ferramenta pgAdmin 4, para interagir com o Firebird fo utilizada a ferramenta FlameRobin e para interagir com o MariaDB foi utilizado seu cliente nativo via terminal. Os SGBDs foram utilizados com a sua configuração padrão, sem nenhuma alteração que privilegiasse o desempenho em relação a alguma limitação própria ou em relação aos demais.

5- Resultados

Nesse capítulo, serão apresentados os resultados aproximados obtidos após a execução dos testes e serão feitas algumas considerações sobre. Cada teste realizado deu origem a um arquivo com o monitoramento da CPU utilizada durante a execução, da memória RAM ocupada e do SWAP utilizado para cada operação (Insert, Select, Update, Delete). Porém, foi observado que em todos os testes realizados, a área de SWAP não foi utilizada, por nenhum SGBD, sendo assim desconsiderada como métrica. Também, para cada teste, foi gerado um arquivo com o registro do tempo gasto na execução de cada operação. Assim, para cada teste, foram gerados 13 arquivos.

Foram realizados 10 testes para cada faixa de carga de trabalho (1000, 2000, 4000 e 8000 registros). Desconsiderando os arquivos com registros de SWAP, cada teste deu origem a 9 arquivos a serem analisados. Levando em conta que a mesma carga de teste foi aplicada para execução com dados em claro e com a aplicação de criptografia, para cada um dos SGBDs, o total de arquivos com registros capturados analisados foram de 2160 arquivos[1].

A análise dos resultados consistiu, no primeiro momento, na média dos valores registrados, que fazem parte do escopo de métricas adotadas, em cada arquivo. Ao analisar o percentual de CPU utilizado, com exceção da primeira linha do arquivo, para cada linha de dados registrados, foram somados os valores referentes ao uso de CPU por parte do sistema com o valor de CPU usado pelo usuário. Ao final do arquivo, foi calculada a média dos valores somados e o desvio padrão relacionado. A primeira linha dos testes de CPU foi ignorada devido ao valor extremamente discrepante com os demais registros, pois se trata de um período de inicialização da operação. Lembrando que cada linha reflete ao período de 1 segundo de captura.

Ao fazer o primeiro passo da análise como os arquivos referentes a ocupação de memória RAM durante a execução dos testes, foram levados em conta o total de memória disponível (MT) e a memória livre (ML). Foi realizada a subtração dos valores de MT por ML para cada linha, sendo calculada ao fim do arquivo a média dos valores e o desvio

[1] Os arquivos dos resultados obtidos com os testes podem ser vistos em: https://github.com/ergor86/resultados-tcc

padrão.

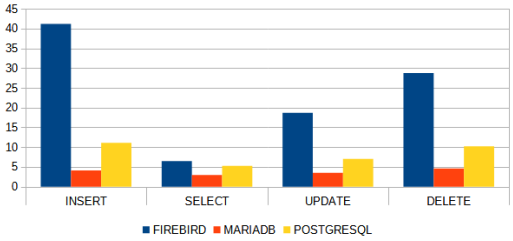
Após calculada a média e desvio padrão das métricas em cada arquivo, foi calculada a média dos resultados obtidos nesses arquivos por faixa de carga de trabalho para cada operação. Com base nesses resultados, foi feita a comparação do desempenho alcançado entre os SGBDs, tendo como parâmetro as métricas utilizadas.

5.1- Comparativo dos SGBDs com relação à média do tempo de execução em claro

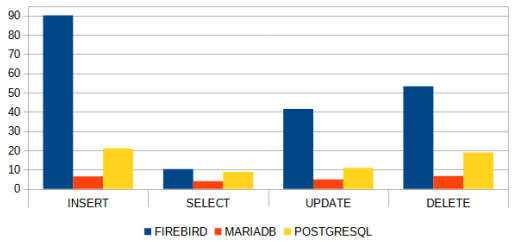
A Tabela 5 expõe os resultados aproximados do processo de testes, com relação ao tempo médio de execução em cada faixa de carga de dados, para cada SGBD. Nesse critério, vemos que o Firebird teve o pior desempenho em relação aos demais, principalmente para operações envolvendo escrita de dados, consumindo muito mais tempo do que o PostgreSQL, que obteve o segundo melhor resultado. O MariaDB obteve um desempenho muito superior aos demais. Porém, se faz necessária a ressalva da influência do modo de operação, já que o ECB utilizado por ele, não faz uso de vetor de inicialização e, portanto, tendo as instruções mais simples que os demais. A partir do desvio padrão de cada média apresentada por faixa, é possível verificar o grau de variação dos dados utilizados para cálculo em torno da média (WOLFFENBÜTTEL, 2006), sendo o Firebird o SGBD que, de modo geral, apresenta a maior variação. A Figura 22 retrata os dados expostos na Tabela 5 em forma de gráfico.

		Firebird		MariaDB		PostgreSQL	
		Média	**Desv. Pad.**	**Média**	**Desv. Pad**	**Média**	**Desv. Pad.**
INSERT	**1K**	41,207	3,3	4,095	0,4	11,103	0,8
	2K	90,097	11,0	6,490	0,1	20,944	1,9
	4K	213,517	18,5	11,160	0,4	40,671	2,2
	8K	503,994	23,8	19,611	0,4	88,328	1,9
SELECT	**1K**	6,445	0,7	2,934	0,2	5,231	0,3
	2K	10,301	0,2	3,941	0,2	8,793	0,6
	4K	19,131	1,8	6,050	0,3	14,090	0,2
	8K	32,519	0,2	9,974	0,1	26,098	0,4
UPDATE	**1K**	18,680	3,1	3,472	0,3	6,985	0,6
	2K	41,549	3,2	4,847	0,1	10,962	0,4
	4K	122,870	1,7	7,604	0,1	19,742	0,2
	8K	412,445	4,0	13,447	0,5	39,654	0,3
DELETE	**1K**	28,774	2,2	4,582	0,9	10,177	0,7
	2K	53,249	6,5	6,638	0,2	18,840	0,3
	4K	105,224	12,6	11,219	0,5	39,383	0,3
	8K	192,680	11,0	19,805	0,6	97,329	1,5

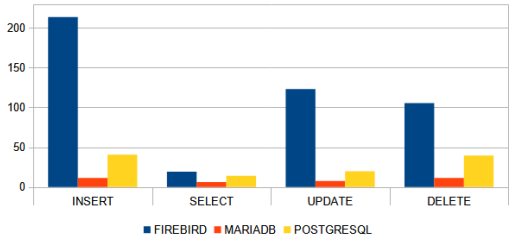
Tabela 5 – Comparação de tempo de execução (em segundos) das operações com os dados em claro



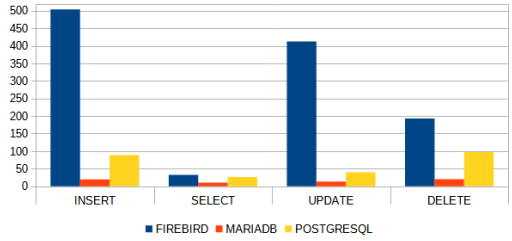
(a) Comparação de tempo médio de execução (em segundos)
para 1k registros em claro



(b) Comparação de tempo médio de execução (em segundos)
para 2k registros em claro



(c) Comparação de tempo médio de execução (em segundos)
para 4k registros em claro



(d) Comparação de tempo médio de execução (em segundos)
para 8k registros em claro

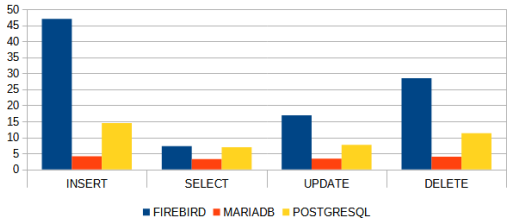
Figura 22 – Comparação de tempo médio de execução (em segundos) das operações com os dados em claro

5.2- Comparativo dos SGBDs com relação à média do tempo de execução utilizando as funções criptográficas

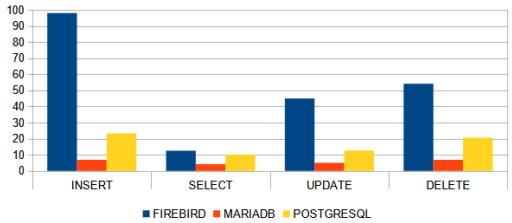
A Tabela 6 expõe os resultados aproximados obtidos a respeito do tempo médio de execução utilizado nas operações, a partir dos testes com funções criptográficas. É possível visualizar que, de forma geral, ocorreu um acréscimo na quantidade de tempo consumido para todos os SGBDs, mantendo-se no, entanto, o MariaDB com o melhor desempenho e o Firebird com o pior tempo alcançado. A Figura 23 expõe o comparativo desses dados para melhor compreensão.

		Firebird		MariaDB		PostgreSQL	
		Média	**Desv. Pad.**	**Média**	**Desv. Pad**	**Média**	**Desv. Pad.**
INSERT	**1K**	46,985	6,6	4,086	0,4	14,474	1,1
	2K	97,969	10,3	6,862	0,2	23,203	1,8
	4K	215,356	23,6	11,787	0,1	45,207	2,0
	8K	538,682	30,7	21,996	0,4	105,947	7,5
SELECT	**1K**	7,226	0,1	3,228	0,4	6,907	0,9
	2K	12,411	0,2	4,074	0,1	9,976	0,4
	4K	21,608	0,6	6,805	0,4	16,837	0,2
	8K	39,255	0,1	11,464	0,2	32,271	1,9
UPDATE	**1K**	16,854	0,3	3,351	0,4	7,648	0,2
	2K	44,961	2,9	4,874	0,1	12,544	0,1
	4K	123,281	2,0	8,029	0,3	22,958	0,3
	8K	422,281	2,9	14,661	0,4	46,053	0,7
DELETE	**1K**	28,451	2,9	3,968	0,1	11,272	0,5
	2K	54,085	4,7	6,807	0,4	20,599	0,4
	4K	103,929	8,6	11,590	0,4	43,500	0,9
	8K	210,872	15,8	21,084	0,3	108,676	1,3

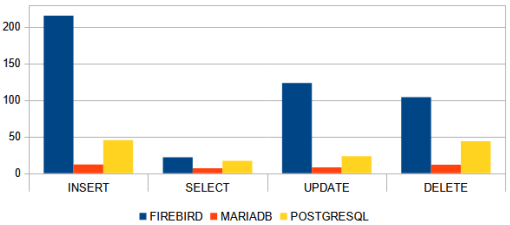
Tabela 6 – Comparação de tempo médio de execução (em segundos) das operações com os dados cifrados



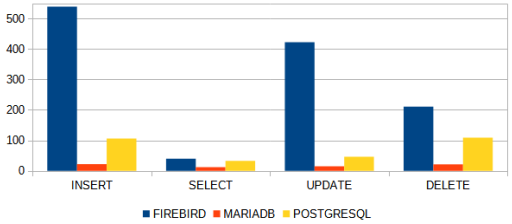
(a) Comparação de tempo de execução (em segundos)
para 1k registros cifrados



(b) Comparação de tempo de execução (em segundos)
para 2k registros cifrados



(c) Comparação de tempo de execução (em segundos)
para 4k registros cifrados



(d) Comparação de tempo de execução (em segundos)
para 8k registros cifrados

Figura 23 – Comparação de tempo médio de execução (em segundos) das operações utilizando funções criptográficas

5.3- Comparativo dos SGBDs com relação à média do uso de CPU utilizando dados em claro

A Tabela 7 expõe os resultados obtidos referentes ao uso de CPU durante a execução dos testes com dados em claro. É possível perceber que o MariaDB obteve a maior taxa de uso da CPU, seguido pelo PostgreSQL. O Firebird teve uma utilização da CPU menor que os demais. O uso de CPU reduzido do Firebird se dá pelo grande número de esperas que ocorrem durante a execução das operações.

Uma característica do Firebird que interfere nesse grande volume de esperas da CPU é que, na configuração padrão de cada base de dados armazenada no mesmo, existe uma funcionalidade chamada *forced writes*. Ela indica que o modo de escrita física no disco ao fazer o *commit* de cada transação pode ser síncrona, realizando a escrita imediata no disco no momento que cada operação de modificação e inserção de novos registros é concluída, ou assíncrona, onde a modificação ou inserção de registros são mantidas no cache da memória e o gerenciamento de efetivação em disco fica a cargo

do subsistema de Entrada/Saída do S.O. Por padrão, a *forced writes* é configurada para ocorrer de forma síncrona, pois a forma assíncrona é mais vulnerável a corrupção da base de dados caso ocorra uma interrupção ou desligamento inesperado do sistema antes que as alterações em memória sejam feitas em disco pelo S.O. (MEMBERS, 2011). Com essa configuração setada para ocorrer de forma síncrona, ocorrem então os tempos de espera de CPU, enquanto os dados são escritos em disco. Essa configuração continua presente na versão do Firebird utilizada nesse trabalho (FILIPPOV et al., 2023). A Figura 24, é referente a um dos arquivos de resultados a partir dos testes para aferir o uso de CPU na operação de inserção no Firebird com dados em claro, que evidencia a espera no uso da CPU devido às operações síncronas de escrita no disco.

Figura 24 – Exemplo de arquivo de resultado coletado de teste de uso de CPU na operação de Insert com dados em claro no Firebird

CPU	sys	11%	user	25%	irq	1%	idle	53%	wait	10%	
CPU	sys	31%	user	69%	irq	0%	idle	0%	wait	0%	
CPU	sys	29%	user	71%	irq	0%	idle	0%	wait	0%	
CPU	sys	32%	user	66%	irq	2%	idle	0%	wait	0%	
CPU	sys	33%	user	58%	irq	3%	idle	0%	wait	6%	
CPU	sys	34%	user	31%	irq	3%	idle	0%	wait	33%	
CPU	sys	27%	user	35%	irq	2%	idle	0%	wait	36%	
CPU	sys	20%	user	26%	irq	4%	idle	0%	wait	49%	
CPU	sys	22%	user	27%	irq	2%	idle	0%	wait	47%	
CPU	sys	28%	user	21%	irq	3%	idle	0%	wait	47%	
CPU	sys	27%	user	28%	irq	3%	idle	0%	wait	42%	
CPU	sys	28%	user	26%	irq	2%	idle	0%	wait	45%	
CPU	sys	24%	user	27%	irq	3%	idle	0%	wait	44%	
CPU	sys	23%	user	23%	irq	3%	idle	0%	wait	51%	
CPU	sys	19%	user	31%	irq	1%	idle	1%	wait	48%	
CPU	sys	23%	user	25%	irq	4%	idle	0%	wait	48%	
CPU	sys	22%	user	30%	irq	4%	idle	0%	wait	43%	
CPU	sys	26%	user	24%	irq	2%	idle	0%	wait	48%	
CPU	sys	20%	user	33%	irq	2%	idle	0%	wait	45%	
CPU	sys	27%	user	24%	irq	4%	idle	0%	wait	45%	

O MariaDB possui um recurso chamado *InnoDB Buffer Pool* que armazena dados e índices acessados com frequência em memória (MARIADB, 2023c). Em operações de escrita com tabelas InnoDB, que é um mecanismo de armazenamento padrão do MariaDB, os dados modificados são primeiramente armazenados no *InnoDB Buffer Pool*, podendo assim agrupar várias operações de escrita para serem gravadas em disco em lote, diminuindo a quantidade de operações de Entrada/Saída no disco (BLACK, 2021). Esse recurso faz parte da configuração padrão do MariaDB, que por consequência proporciona menos tempo de espera de CPU nas operações para escritas em disco, em contraste com o que ocorre no Firebird na sua configuração padrão.

O PostgreSQL possui como recurso da configuração padrão um log em memória das alterações em arquivos de dados a serem efetuadas na base, chamado *Write-Ahead Logging* (WAL). Sua função principal é garantir a integridade dos dados, pois com o registro das mudanças no log antes de serem efetivadas, caso ocorra alguma falha, é possível fazer a repetição das transações a partir desses registros. Em relação seu impacto na escrita em disco (POSTGRESQL, 2023c):

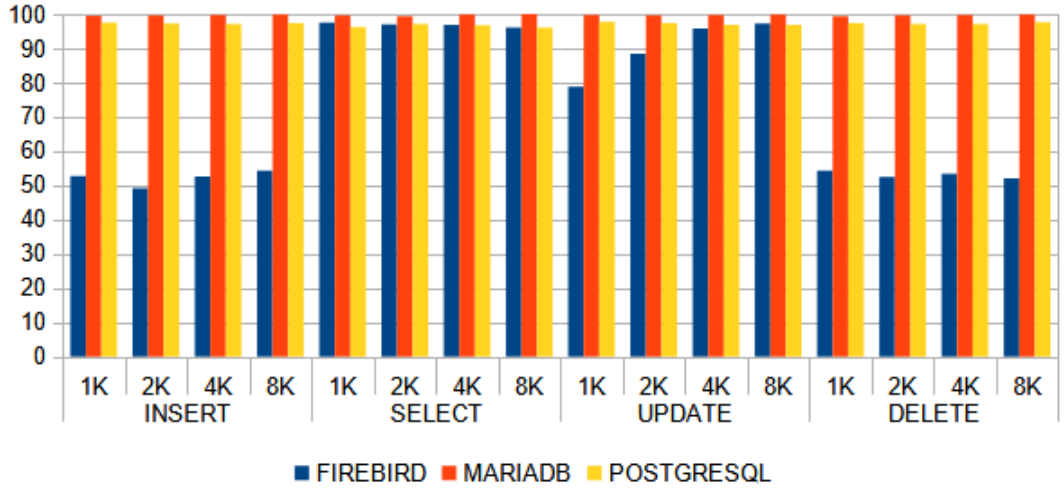
> O uso do WAL resulta em um número significativamente reduzido de gravações em disco, porque apenas o arquivo de log precisa ser liberado no disco para garantir que uma transação seja confirmada, em vez de cada arquivo de dados alterado pela transação. O arquivo de log é gravado sequencialmente e, portanto, o custo de sincronizar o log é muito menor do que o custo de liberar as páginas de dados. Isso é especialmente verdadeiro para servidores que lidam com muitas transações pequenas que afetam diferentes partes do armazenamento de dados. Além disso, quando o servidor está processando muitas pequenas transações simultâneas, um fsync do arquivo de log pode ser suficiente para confirmar muitas transações.

Dessa forma, devido a essa característica padrão que o PostgreSQL possui, ele sofre menos interrupções e tempos de espera que o Firebird em sua configuração padrão, o que se reflete numa média de ocupação de CPU maior. A Figura 25 exibe gráfico baseado nos resultados expostos na Tabela 7.

		Firebird		MariaDB		PostgreSQL	
		Média	**Desv. Pad.**	**Média**	**Desv. Pad**	**Média**	**Desv. Pad.**
INSERT	**1K**	52,6	5,0	99,5	0,6	97,5	0,6
	2K	49,1	5,4	99,6	0,6	97,3	0,2
	4K	52,6	4,1	99,7	0,2	97,1	0,2
	8K	54,2	2,0	99,9	0,05	97,4	0,1
SELECT	**1K**	97,5	0,6	99,6	0,7	96,2	5,3
	2K	97,0	0,4	99,4	0,7	97,1	0,5
	4K	96,8	0,3	99,8	0,3	96,7	0,3
	8K	96,0	0,2	100	0	96,0	0,2
UPDATE	**1K**	78,8	11,1	99,8	0,5	97,8	0,4
	2K	88,4	4,9	99,6	0,6	97,3	0,3
	4K	95,7	0,3	99,8	0,4	99,8	0,2
	8K	97,3	0,1	99,9	0,04	96,9	0,1
DELETE	**1K**	54,2	5,6	99,4	1,2	97,4	0,3
	2K	52,4	5,2	99,6	0,6	97,1	0,1
	4K	53,3	4,6	99,8	0,4	97,1	0,1
	8K	52,0	2,9	99,9	0,04	97,6	0,1

Tabela 7 – Comparação do percentual médio de CPU utilizado pelos SGBDs durante a execução das operações com dados em claro

Figura 25 – Comparação do percentual médio de uso de CPU nas operações com dados em claro



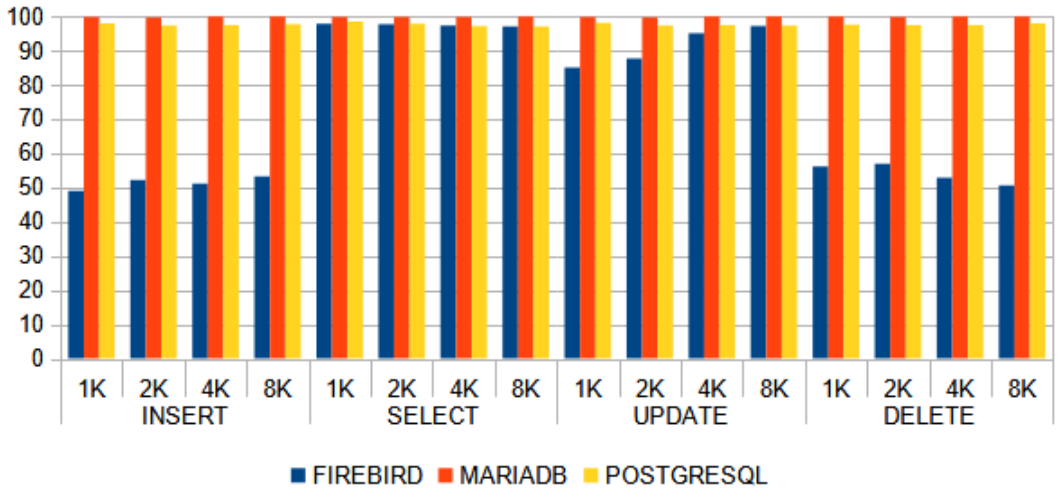
5.4- Comparativo dos SGBDs com relação à média do uso de CPU utilizando funções criptográficas

Os resultados referentes ao uso de CPU durante as operações utilizando funções criptográficas, expostos na Tabela 8, revelam que, de forma semelhante aos resultados das operações com dados em claro, o MariaDB possui o maior percentual de CPU utilizado, enquanto o FireBird continua com uma taxa de utilização menor. A diminuição do uso de CPU, vista em algumas faixas de carga de trabalho, ocorre com o aumento de ocorrência de estado de espera durante a execução das operações com as funções criptográficas. A Figura 26 expõe as diferenças nos resultados elencados na Tabela 8.

		Firebird		**MariaDB**		**PostgreSQL**	
		Média	**Desv. Pad.**	**Média**	**Desv. Pad**	**Média**	**Desv. Pad.**
INSERT	**1K**	48,9	4,9	99,7	0,3	97,9	0,2
	2K	52,1	5,8	99,6	0,5	97,3	0,3
	4K	51,1	4,2	99,9	0,1	97,3	0,1
	8K	53,1	2,2	99,9	0,1	97,6	0,1
SELECT	**1K**	97,8	0,5	99,7	0,5	98,4	0,3
	2K	97,7	0,3	99,8	0,5	97,7	0,4
	4K	97,2	0,2	99,8	0,4	97,1	0,3
	8K	97,0	0,1	99,9	0,03	97,0	0,1
UPDATE	**1K**	85,0	0,8	99,7	0,6	98,0	0,3
	2K	87,7	5,3	99,7	0,5	97,3	0,3
	4K	95,0	0,5	99,9	0,06	97,3	0,1
	8K	97,1	0,1	99,9	0,03	97,2	0,1
DELETE	**1K**	56,1	3,9	99,9	0,1	97,6	0,3
	2K	56,9	4,1	99,8	0,3	97,4	0,1
	4K	52,7	3,6	99,8	0,1	97,3	0,1
	8K	50,6	2,8	99,9	0,05	97,9	0,1

Tabela 8 – Comparação do percentual médio de CPU utilizado pelos SGBDs durante a execução das operações utilizando funções criptográficas

Figura 26 – Comparação do percentual médio de uso de CPU nas operações utilizando funções criptográficas



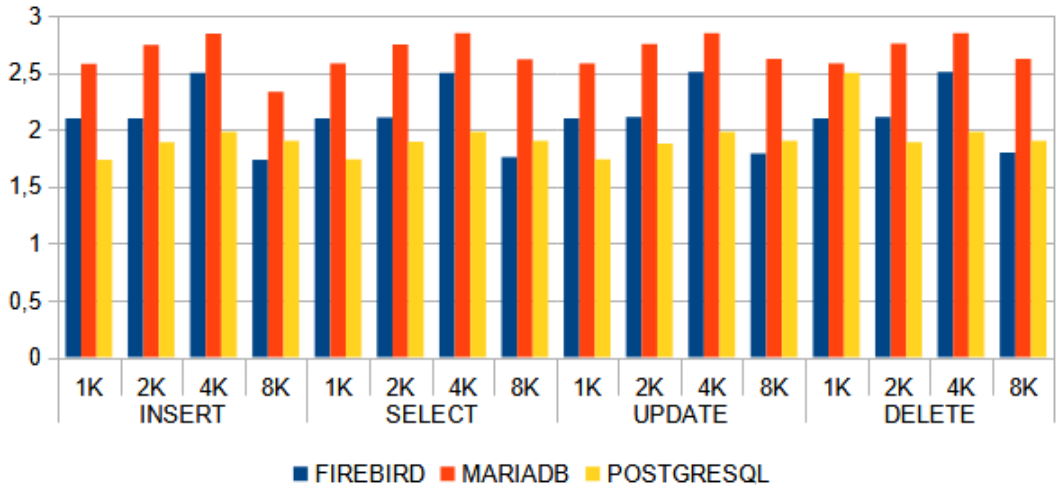
5.5- Comparativo dos SGBDs com relação à média de ocupação de memória RAM durante execução das operações com dados em claro

Os resultados referentes à ocupação de memória RAM, em operações com dados em claro, são apresentados na Tabela 9. De acordo com esses, de modo geral, o MariaDB foi o SGBD com maiores médias de ocupação de memória RAM. O PostgreSQL é o que possui menor taxa de ocupação de memória RAM na maior parte de cargas de trabalho aplicada, com exceção da faixa de 8000 registros. Nessa faixa, ele supera o Firebird em todas as operações. Uma peculiaridade percebida é que, na faixa de 4000 registros, os SGBDs atingem seu pico de ocupação em todas as operações. Os resultados da Tabela 9 são representados de forma visual na Figura 27.

		Firebird		MariaDB		PostgreSQL	
		Média	**Desv. Pad.**	**Média**	**Desv. Pad**	**Média**	**Desv. Pad.**
INSERT	**1K**	2,1	0	2,5	0,3	1,7	0,06
	2K	2,1	0,002	2,7	0,04	1,8	0,02
	4K	2,4	0,0003	2,8	0,02	1,9	0,04
	8K	1,7	0,3	2,3	0,9	1,8	0,0003
SELECT	**1K**	2,1	0	2,5	0,3	1,7	0,06
	2K	2,1	0,01	2,7	0,04	1,8	0,01
	4K	2,4	0,006	2,8	0,02	1,9	0,04
	8K	1,7	0,3	2,6	0,4	1,8	0,001
UPDATE	**1K**	2,1	0	2,5	0,3	1,7	0,07
	2K	2,1	0,02	2,7	0,04	1,8	0,04
	4K	2,5	0,02	2,8	0,02	1,9	0,04
	8K	1,7	0,3	2,6	0,4	1,9	0
DELETE	**1K**	2,1	0	2,5	0,3	2,5	0,07
	2K	2,1	0,02	2,7	0,04	1,8	0,03
	4K	2,5	0,02	2,8	0,02	1,9	0,04
	8K	1,8	0,3	2,6	0,4	1,8	0,0003

Tabela 9 – Comparação de ocupação de memória RAM (GB) durante a execução das operações com dados em claro.

Figura 27 – Comparação de ocupação de memória RAM (GB) nas operações utilizando dados em claro



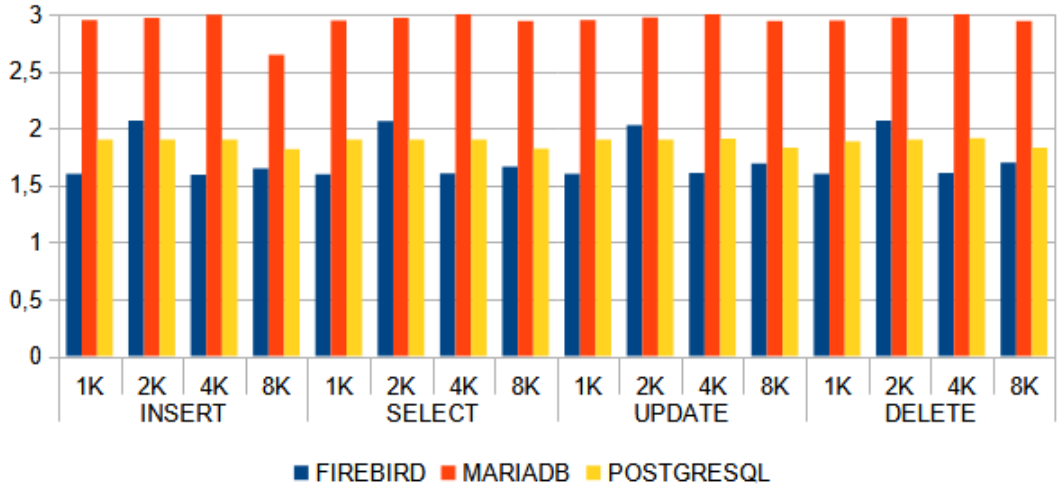
5.6- Comparativo dos SGBDs com relação à média de ocupação de memória RAM durante execução das operações utilizando as funções criptográficas

A Tabela 10 lista os resultados obtidos para os testes de ocupação de memória RAM durante a execução das operações, utilizando funções criptográficas. Com o emprego das funções criptográficas, é possível perceber que o PostgreSQL passa a ocupar mais memória RAM em relação ao Firebird em mais faixas que as verificadas, quando utilizados dados em claro. Com exceção da faixa de 2000 registros de carga de trabalho, o PostgreSQL ocupou mais memória RAM que o Firebird em todas as outras. Uma possível causa para essa mudança, é o aumento no tempo de execução das operações quando utilizadas as funções criptográficas, visto que o Firebird tem o processamento mais ocioso nesse cenário. O gráfico referente a Tabela 10 é apresentado na Figura 28.

		Firebird		MariaDB		PostgreSQL	
		Média	**Desv. Pad.**	**Média**	**Desv. Pad**	**Média**	**Desv. Pad.**
INSERT	**1K**	1,5	0,0006	2.9	0,1	1,9	0
	2K	2,0	0,2	2,9	0,01	1,9	0,003
	4K	1,5	0,2	2,9	0,008	1,9	0,003
	8K	1,6	0,1	2,6	0,8	1,8	0,07
SELECT	**1K**	1,5	0,004	2,9	0,1	1,9	0
	2K	2.0	0,2	2,9	0,01	1,9	0,003
	4K	1,6	0,2	2,9	0,009	1,9	0,005
	8K	1,6	0,09	2,9	0,03	1,8	0,07
UPDATE	**1K**	1,5	0,001	2,9	0,1	1,9	0
	2K	2,0	0,2	2,9	0,01	1,9	0
	4K	1,6	0,2	3,0	0,008	1,9	0,02
	8K	1,6	0,08	2,9	0,03	1,8	0,06
DELETE	**1K**	1,5	0,0009	2,9	0,1	1,8	0
	2K	2.0	0,2	2,9	0,01	1,9	0
	4K	1,6	0,2	3,0	0,008	1,9	0,02
	8K	1,6	0,07	2,9	0,03	1,8	0,06

Tabela 10 – Resultados dos testes avaliando a ocupação de memória RAM (GB) durante as operações utilizando funções criptográficas

Figura 28 – Comparação de ocupação de memória RAM (GB) nas operações utilizando funções criptográficas



5.7- Comparação do impacto causado no tempo médio de execução pelas funções criptográficas

Com base nos resultados obtidos dos testes de tempo de execução, com os dados em claro e utilizando as funções criptográficas, foi feito um comparativo de como o emprego das funções afeta o desempenho dos SGBDs. Para isso, foi primeiramente calculada a diferença percentual, entre os resultados obtidos dos testes utilizando as funções criptográficas em relação aos testes de tempo de execução com os dados em claro, em cada faixa de carga de trabalho, para cada operação realizada (INSERT, SELECT, UPDATE e DELETE). Após isso, para cada operação, foi calculada a média das diferenças percentuais encontradas em cada faixa de carga de trabalho. Tendo a média do cálculo diferencial de cada operação, foram calculadas as médias gerais do custo no desempenho, em relação a tempo de execução, para cada SGBD, apresentadas na Tabela 11.

O cálculo não foi realizado para as outras métricas devido à relação entre o tempo de execução consumido e a ocupação de memória e uso de CPU. Como exemplo, temos o Firebird, com um grande volume de interrupções que causam espera e subutilização da CPU, alongado o seu tempo de execução e apresentando uma menor ocupação de memória que os demais SGBDs na maior parte das operações realizadas, quando aplicadas as funções criptográficas. O Firebird apresenta um menor uso de CPU que os demais SGBDs na maior parte das operações e uma ocupação de memória intermediária entre os 3 avaliados, nas operações em claro, resultado num maior tempo de execução. Quando adicionada a complexidade das funções criptográficas, o Firebird passa a ter também uma ocupação de memória menor que os demais, na maior parte das cargas de trabalho, aumentando ainda mais o seu tempo de execução.

O PostgreSQL foi o mais afetado pela criptografia quando comparado seu desempenho com os dados em claro. Uma causa possível, baseado nos resultados obtidos, é a menor ocupação de memória RAM durante a execução das operações utilizando as funções criptográficas, ocasionando mais lentidão no processamento.

O MariaDB apresentou o menor impacto no tempo de execução com o emprego das funções criptográficas. O Uso de CPU e ocupação de memória RAM no MariaDB foram os maiores, tanto para operações com dados em claro, quanto para quando as

funções criptográficas, executando as operações em um tempo muito menor que os demais. O ponto a ser ponderado, é o modo de operação mais vulnerável a ataques, o que pelo critério de segurança, não é desejado.

			Custo no desempenho	
SGBD	**Algoritmo**	**Modo de Operação**	**Média aprox.**	**Desv. Pad. aproximado**
MariaDB	AES-128	ECB	4,6%	4,4
Firebird	AES-128	OFB	6,7%	7,3
PostgreSQL	AES-128	CBC	16,2%	5,01

Tabela 11 – Custo do emprego das funções criptográficas no tempo de execução das operações dos SGBDs

6- Conclusões

Esse trabalho apresentou uma análise de desempenho das funções criptográficas nativas de SGBDs open source, sendo os escolhidos para estudo o PostgreSQL, o Firebird e o MariaDB. O desempenho das funções criptográficas foi avaliado através das métricas de tempo médio de execução consumido, percentual médio de utilização de CPU, que são métricas similares às utilizadas no trabalho de (KILONZO, 2020), além da média de ocupação de memória RAM. Essas métricas foram aferidas durante as operações de inserção, busca, atualização e remoção de registros nos SGBDs. Para isso, foram executados 2 conjuntos de 10 testes que englobaram as 4 operações, para cada uma das faixas de carga de trabalho adotadas, sendo essas de 1000, 2000, 4000 e 8000 registros. Um conjunto de teste foi realizado com os dados em claro e o segundo conjunto aplicando as funções criptográficas. O conjunto de dados utilizados como carga de trabalho são oriundos de uma base pública do Ministério da Saúde do Brasil.

Os testes foram aplicados por meio de uma aplicação desenvolvida, sendo as métricas mensuradas via Atop, ferramenta instalada no sistema operacional. O mesmo ambiente foi utilizado para a execução dos testes em todos os SGBDs e todos os SGBDs foram utilizados com suas configurações padrão, sem nenhuma alteração que privilegiasse seu desempenho.

Diante dos resultados obtidos, foram feitas comparações do desempenho entre os SGBDs, conforme as métricas adotadas, através da exposição dos resultados por meio de tabelas e gráficos. Foi avaliado o custo do emprego das funções criptográficas em cada SGBD, comparando o tempo de execução consumido nos testes realizados com os dados em claro e aplicando as funções criptográficas. Diante dos resultados obtidos, foi concluído que o tempo de execução gasto durante as operações é influenciado pela relação do uso de CPU com a quantidade de memória RAM ocupada durante a execução das operações.

Foi verificado que, avaliando o impacto das funções criptográficas mediante tempo de execução, o MariaDB foi o menos afetado, tendo um custo médio calculado de aproximadamente 4,6%. Com um custo médio menor que os demais e levando em conta os comparativos de CPU e memória RAM utilizadas, conclui-se que o MariaDB faz um

melhor aproveitamento dos recursos computacionais disponíveis.

O MariaDB também foi objeto de um estudo de desempenho, abordado no capítulo 3 desse trabalho, utilizando criptografia para dados em repouso, aplicando o algoritmo AES (ISTIFAN; MAKOVAC, 2022). Nesse estudo, quando comparado com o MySQL, o MariaDB foi superior segundo as métricas adotadas, TPM e NOPM. Uma ressalva que não é abordada no estudo relacionado e que aqui também deve ser feita, é a fragilidade do modo de operação ECB, utilizado pelo MariaDB. No quesito de segurança, o fato de desencadear repetição dos textos cifrados para blocos de texto idênticos faz com que não seja de utilização recomendável (DWORKIN; STANDARDS; DIV, 2001). Também por se tratar de um modo mais simples, em comparação com o CBC e o OFB, pode ter exercido influência nos resultados obtidos.

A avaliação do impacto do emprego das funções criptográficas no tempo de execução do SGBDs mostra que o PostgreSQL é o mais afetado, com um custo médio de aproximadamente 16,2%. A premissa de que, o emprego de meios para garantir maior segurança dos dados também causa prejuízo no desempenho, também foi abordada no trabalho de (SHASTRI et al., 2019), relacionado no capítulo 3 desse trabalho. No trabalho citado, foi avaliado como modificações feitas em 3 SGBDs, sendo um deles o PostgreSQL, para que se tornassem aderentes a GDPR, iriam impactar no desempenho dos mesmos. O emprego de criptografia fez parte das modificações propostas, e embora os meios de avaliação tenham sido diferentes, concluiu-se que o PostgreSQL também sofreu forte impacto, principalmente com cargas maiores.

O Firebird obteve um custo médio intermediário entre os SGBDs avaliados, de aproximadamente 6,7%. Verificou-se, porém, que quando avaliado em relação aos demais em relação a tempo de execução, foi o mais lento, com CPU e memória RAM subutilizadas. Em relação ao uso de CPU abaixo dos demais, uma das causas possíveis é o uso por padrão do SGBD da função *forced writes*, enquanto os outros SGBDs, por padrão, utilizam mais operações de armazenamento de dados em memória para diminuir quantidade de operações de escrita em disco, como apontado na seção 5.3. O Firebird foi o SGBD com maior complexidade no uso das funções criptográficas, como abordado na seção 4.2.

Avaliando a necessidade de atender a demanda da relação entre Desempenho x Segurança, que surgiu com as legislações como a LGPD, mediante os testes resultados obtidos nesse trabalho, o PostgreSQL oferece mais segurança que o MariaDB, que foi o melhor avaliado em desempenho, e consegue ser mais rápido que o Firebird, que

também utiliza instruções com maior suporte a segurança que o MariaDB, mas com desempenho negativamente discrepante em relação aos demais. Cabe ao administrador do banco de dados avaliar se o custo de desempenho agregado com a utilização de criptografia no PostgreSQL é viável para seu modelo. Para uma situação onde não se tenha o armazenamento de dados sensíveis, por questão de desempenho, o SGBD indicado seria o MariaDB.

Vale ressaltar que os resultados e conclusões obtidas nesse trabalho não refletem uma posição universal, mas relativas apenas ao ambiente utilizado, ao tipo de criptografia empregado e aos tipos de testes realizados nesse trabalho. Após o encerramento das pesquisas desse trabalho, verificou-se o lançamento da mais nova versão do MariaDB, o MariaDB 11.2.0, ainda em versão alfa, que permite a utilização de outros tamanhos de chave criptográfica, outros modos de operação além do ECB e o emprego de vetor de inicialização (MARIADB, 2023d). Dessa forma, como trabalho futuro, poderia ser feita uma nova pesquisa para verificar se com o emprego de modos de operação mais complexos, assim como do vetor de inicialização, o desempenho do MariaDB ainda se manteria a frente dos demais SGBDs avaliados. Poderia ser feita, também, uma avaliação da viabilidade de uso das funções criptográficas nativas para um ambiente de base de dados distribuída, visto que não foi possível avaliar durante a execução desse trabalho pela complexidade e tempo que a construção do ambiente para os experimentos nesse sentido demandaria.

Referências

ATHENIENSE, Alexandre. **As mudanças na Lei de Proteção de Dados Pessoais**. [S.l.: s.n.], 2018. Disponível em: https://alexandre-atheniense.jusbrasil.com.br/noticias/662004296/as-mudancas-na-lei-de-protecao-de-dados-pessoais. Acessado em 8 de Agosto de 2022.

ATOPTOOL.NL. **atoptool.nl**. [S.l.: s.n.], 2023. Disponível em: https://www.atoptool.nl/. Acessado em 3 de junho de 2023.

BLACK, Daniel. **10.7 preview feature: InnoDB Bulk Insert**. [S.l.: s.n.], out. 2021. Disponível em : https://mariadb.org/10-7-preview-feature-innodb-bulk-insert/. Acessado em 08 de agosto de 2023.

CANTU, Carlos. **Conheça o Firebird em 2 minutos**. [S.l.: s.n.], 2010. Disponível em: https://www.firebirdnews.org/docs/fb2min_ptbr.html. Acessado em 31 de maio de 2023.

CIDADANIA, Ministério da. **Princípios da LGPD**. [S.l.: s.n.], abr. 2021. Disponível em: https://www.gov.br/cidadania/pt-br/acesso-a-informacao/lgpd/principios-da-lgpd. Acessado em 9 de Agosto de 2022.

CNN. **O mundo já registra 4,6 bilhões de dados vazados em 2021, diz PSafe**. [S.l.: s.n.], jul. 2021. Disponível em: `https://www.cnnbrasil.com.br/business/o-mundo-ja-registra-4-6-bilhoes-de-dados-vazados-em-2021-diz-psafe/`. Acessado em 16 de junho de 2022.

DETOMINI, Renan Corrêa. Exploração de Paralelismo em Criptografia Utilizando GPUs. **SJR Preto, Universidade Julio de Mesquita Filho**, 2010.

DIGITAL, Convergência. **Vazamento de dados registra crescimento de 500% no Brasil**. [S.l.: s.n.], set. 2021. Disponível em: `https://www.convergenciadigital.com.br/Seguranca/Vazamento-de-dados-registra-crescimento-de-500%25-no-Brasil-58191.html`. Acessado em 16 de junho de 2022.

DWORKIN, Morris; STANDARDS, NATIONAL INST OF; DIV, TECHNOLOGY GAITHERSBURG MD COMPUTER SECURITY. Recommendation for Block Cipher Modes of Operation. Methods and Techniques, 2001.

DWORKIN, Morris et al. **Advanced Encryption Standard (AES)**. en. [S.l.]: Federal Inf. Process. Stds. (NIST FIPS), National Institute of Standards e Technology, Gaithersburg, MD, 2001-11-26 2001. DOI: https://doi.org/10.6028/NIST.FIPS.197.

DWORKIN, Morris J. **SP 800-38A 2001 Edition. Recommendation for Block Cipher Modes of Operation: Methods and Techniques**. Gaithersburg, MD, USA, 2001.

FILIPPOV, Dmitry et al. **Firebird 4.0 Language Reference**. [S.l.: s.n.], ago. 2023. Disponível em :https://firebirdsql.org/file/documentation/html/en/refdocs/fblangref40/firebird-40-language-reference.html. Acessado em 08 de agosto de 2023.

FIREBIRD. **8.11 Cryptographic Functions**. [S.l.: s.n.], 2023. Disponível em: https://firebirdsql.org/file/documentation/chunk/en/refdocs/fblangref40/fblangref40-scalarfuncs-crypto.html. Acessado em 5 de maio de 2023.

_____. **Features**. [S.l.: s.n.], 2023. Disponível em: https://firebirdsql.org/en/features/. Acessado em 31 de maio de 2023.

GOLUBCHIK, Sergei. **MDEV-9069 extend AES_ENCRYPT() and AES_DECRYPT() to support IV and the algorithm**. [S.l.: s.n.], 2023. Disponível em: https://jira.mariadb.org/browse/MDEV-9069. Acessado em 30 de maio de 2023.

GROUP, The PHP. **O que é o PHP?** [S.l.: s.n.], 2023. Disponível em: https://www.php.net/manual/pt_BR/intro-whatis.php. Acessado em 1 de junho de 2023.

HEUSER, C.A. **Projeto de Banco de Dados**. 6. ed. [S.l.]: Bookman, 2009. (Série Livros Didáticos Informática UFRGS). ISBN 978-85-7780-382-8.

IBM. **O que é OLAP?** [S.l.: s.n.], 2023. Disponível em: https://www.ibm.com/br-pt/topics/olap. Acessado em 31 de maio de 2023.

IBSURGEON. **Firebird Encryption Plugin Framework**. [S.l.: s.n.], 2023. Disponível em : https://ib-aid.com/en/firebird-encryption-plugin-framework/. Acessado em 02 de agosto de 2023.

INFOSEC, GAT. **5 pilares de Segurança da Informação nas empresas**. [S.l.: s.n.], 2022. Disponível em: https://www.gat.digital/blog/5-pilares-da-seguranca-da-informacao/. Acessado em 9 de Agosto de 2022.

ISTIFAN, Stewart; MAKOVAC, Mattias. **Performance benchmarking of data-at-rest encryption in relational databases**. [S.l.: s.n.], 2022.

KILONZO, Alex M. **Impact of Symmetric-Key Encryption on Performance of Relational and Nosql Database to Preserve Data Confidentiality**. 2020. Tese (Doutorado) – United States International University-Africa.

MARIADB. **About MariaDB**. [S.l.: s.n.], 2023. Disponível em: https://mariadb.com/about-us/. Acessado em 30 de maio de 2023.

_____. **AES_ENCRYPT**. [S.l.: s.n.], 2023. Disponível em: https://mariadb.com/kb/en/aes_encrypt/. Acessado em 30 de maio de 2013.

_____. **InnoDB Buffer Pool**. [S.l.: s.n.], 2023. Disponível em : https://mariadb.com/kb/en/innodb-buffer-pool/. Acessado em 08 de agosto de 2023.

_____. **MariaDB versus MySQL: Compatibility**. [S.l.: s.n.], 2023. Disponível em: https://mariadb.com/kb/en/mariadb-vs-mysql-compatibility/. Acessado em 30 de maio de 2023.

_____. **System Variable Differences Between MariaDB 10.8 and MySQL 8.0**. [S.l.: s.n.], 2023. Disponível em: https://mariadb.com/kb/en/system-variable-differences-between-mariadb-10-8-and-mysql-8-0/. Acessado em 30 de maio de 2023.

MEMBERS, Firebird Project. **Protecting your data**. [S.l.: s.n.], jul. 2011. Disponível em : https://firebirdsql.org/manual/qsg2-safety.html. Acessado em 08 de agosto de 2023.

OPENDATASUS. **Registro de Ocupação Hospitalar COVID-19 - 2021**. [S.l.: s.n.], 2022. Disponível em: https://opendatasus.saude.gov.br/dataset/registro-de-ocupacao-hospitalar-covid-19/resource/1a1e9932-253f-441b-8d9c-aaf35f3022a0. Acessado em 4 de novembro de 2022.

ORACLE. **Encryption and Compression Functions**. [S.l.: s.n.], 2023. Disponível em: https://dev.mysql.com/doc/refman/8.0/en/encryption-functions.html#function_aes-encrypt. Acessado em 30 de maio de 2023.

PEARCE, Rohan. **Dead database walking: MySQL's creator on why the future belongs to MariaDB**. [S.l.: s.n.], mar. 2013. Disponível em: https://www2.computerworld.com.au/article/457551/dead. Acessado em 30 de maio de 2023.

POSTGRESQL. **PostgreSQL: About**. [S.l.: s.n.], 2023. Disponível em: https://www.postgresql.org/about/. Acessado em 31 de maio de 2023.

POSTGRESQL. **PostgreSQL: Documentation: 15: F.28. pgcrypto**. [S.l.: s.n.], mai. 2023. Disponível em: https://www.postgresql.org/docs/current/pgcrypto.html#PGCRYPTO-WITH-WITHOUT-OPENSSL/. Acessado em 14 de junho de 2023.

_____. **Write-Ahead Logging (WAL)**. [S.l.: s.n.], 2023. Disponível em : https://www.postgresql.org/docs/current/wal-intro.html. Acessado em 08 de agosto de 2023.

SHASTRI, Supreeth et al. Understanding and benchmarking the impact of GDPR on database systems. **arXiv preprint arXiv:1910.00728**, 2019.

STALLINGS, William. **Criptografia e segurança de redes: princípios e práticas**. 6. ed. São Paulo: Pearson Education do Brasil, 2015. ISBN 978-85-430-1450-0.

TELLE, Ulrich. **SQL Functions**. [S.l.: s.n.], 2021. Disponível em: `https://utelle.github.io/SQLite3MultipleCiphers/docs/configuration/config_sql_functions/`. Acessado em 9 de Agosto de 2022.

TRUZZI, Gisele. **A LGPD entrou em vigor: o que as empresas devem fazer agora?** [S.l.: s.n.], set. 2020. Disponível em: https://www.istoedinheiro.com.br/a-lgpd-entrou-em-vigor-o-que-as-empresas-devem-fazer-agora/. Acessado em 8 de Agosto de 2022.

WOLFFENBÜTTEL, Andréa. **O que é? Desvio padrão**. [S.l.: s.n.], jun. 2006. Disponívem em: http://desafios.ipea.gov.br/index.php?option=com_content&view=article&id=2104:catid=28. Acessado em 11 de junho de 2023.