UNIVERSIDADE FEDERAL DOS VALES DO JEQUITINHONHA E MUCURI
Curso de Graduação em Sistemas de Informação
Luciano Rodrigues Batista

DISCUSSÃO QUALITATIVA E QUANTITATIVA DE IMPLEMENTAÇÃO DE FUNCIONALIDADES DE UM SISTEMA DE E-COMMERCE SEGUNDO ALTERNATIVAS DE MODELAGENS DE DADOS BASEADAS NA ABORDAGEM DE PERSISTÊNCIA POLIGLOTA

Diamantina
2021

Luciano Rodrigues Batista

DISCUSSÃO QUALITATIVA E QUANTITATIVA DE IMPLEMENTAÇÃO DE FUNCIONALIDADES DE UM SISTEMA DE E-COMMERCE SEGUNDO ALTERNATIVAS DE MODELAGENS DE DADOS BASEADAS NA ABORDAGEM DE PERSISTÊNCIA POLIGLOTA

Trabalho de Conclusão de Curso apresentado ao curso de graduação em Sistemas de Informação, como parte dos requisitos exigidos para a obtenção título de Bacharel em Sistemas de Informação.

Orientador: Áthila Rocha Trindade

Diamantina
2021

MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DOS VALES DO JEQUITINHONHA E MUCURI

FOLHA DE APROVAÇÃO

Luciano Rodrigues Batista

DISCUSSÃO QUALITATIVA E QUANTITATIVA DE IMPLEMENTAÇÃO DE FUNCIONALIDADES DE UM SISTEMA DE E-COMMERCE SEGUNDO ALTERNATIVAS DE MODELAGENS DE DADOS BASEADAS NA ABORDAGEM DE PERSISTÊNCIA POLIGLOTA

Trabalho de Conclusão de Curso apresentado ao Curso de Sistemas de Informação da Universidade Federal dos Vales do Jequitinhonha e Mucuri, como requisitos parcial para conclusão do curso.

Orientador: Áthila Rocha Trindade

Data de aprovação: 15/09//2021

Prof. Dr. Áthila Rocha Trindade
Faculdade de Ciências Exatas - UFVJM

Prof. Dr. Alessandro Vivas Andrade
Faculdade de Ciências Exatas - UFVJM

Prof. Dr. Leonardo Lana de Carvalho
Faculdade de Ciências Exatas - UFVJM

Documento assinado eletronicamente por **Áthila Rocha Trindade**, **Servidor**, em 16/09/2021, às 08:46, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

Documento assinado eletronicamente por **Alessandro Vivas Andrade**, **Servidor**, em 16/09/2021, às 11:10, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

Documento assinado eletronicamente por **Leonardo Lana de Carvalho**, **Servidor**, em 23/09/2021, às

17:48, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

A autenticidade deste documento pode ser conferida no site https://sei.ufvjm.edu.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0, informando o código verificador **0461612** e o código CRC **1BEDE407**.

Referência: Processo nº 23086.008168/2021-07 SEI nº 0461612

Dedico este trabalho à minha família, que me deu o suporte necessário para alcançar e superar essa etapa, especialmente à minha mãe, Fabiana, e ao meu pai, Luciano.

AGRADECIMENTOS

Agradeço ao corpo docente e técnicos administrativos do curso de Sistemas de Informação da UFVJM-Campus Diamantina, que me guiaram durante a graduação e me tornaram o aluno que sou hoje.

Agradeço também aos integrantes da Atlética The Bug e da república IML, que trilharam comigo essa jornada, sendo fontes de alegria em meio aos desafios.

RESUMO

A diversidade de bancos de dados para aplicações atuais é um aspecto importante a se considerar, principalmente para aplicações comerciais. Diferentes mecanismos e estruturas de armazenamento providos pelos diferentes SGBDs (Sistemas Gerenciadores de Bancos de Dados) podem ocasionar em uma manipulação de dados com diferentes características: pior ou melhor desempenho nas buscas por informação, maior ou menor grau de garantia de consistência e durabilidade dos dados. Neste sentido, tem sido proposta a utilização de diferentes tipos de SGBDs para diferentes funcionalidades de um mesmo negócio, numa abordagem chamada de persistência poliglota de dados.Este trabalho objetivou, para o contexto de uma aplicação de comércio eletrônico, analisar o desempenho de diferentes operações de bancos de dados quando executadas sobre diferentes modelagens dos dados do sistema de *e-commerce* (utilizando modelagem relacional com SGBD MariaDB e não relacional com SGBD Mongo), a fim de fazer uma discussão quantitativa e qualitativa sobre a adequação de cada operação para cada modelagem de dados. Os resultados computacionais mostraram que é possível sugerir com maior clareza quais modelagens seriam mais adequadas na implementação de diferentes funcionalidades de um sistema de *e-commerce*, observando critérios de desempenho, consistência e durabilidade dos dados.

Palavras-chave: Banco de Dados Relacional. NoSql. Persistência Poliglota. Comércio Eletrônico.

ABSTRACT

The diversity of databases for current applications is an important aspect to consider, especially for commercial applications. Different mechanisms and storage structures provided by different DBMS (Database Management Systems) can lead to data manipulation with different characteristics: worse or better performance in information searches, greater or lesser degree of guarantee of data consistency and durability. In this sense, it has been proposed to use different types of DBMS for different functionalities of the same business, in an approach called polyglot data persistence. This work aimed, for the context of an e-commerce application, to analyze the performance of different operations of databases when run them on different data modeling of the e-commerce system (using relational modeling with DBMS MariaDB and non-relational with DBMS Mongo), in order to make a quantitative and qualitative discussion about the suitability of each operation for each model of data. The computational results showed tha it is possible to suggest with greater clarity which data modeling would be more suitable in the implementation of different functionalities of an e-commerce system, observing performance, consistency and data durability criteria.

Keywords: Relational Database. NoSql. Polyglot Persistence. E-commerce.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

BD	Banco de Dados
BSON	Binary Json
GB	Gigabyte
GHz	Gigahertz
JSON	JavaScript Object Nation
NoSQL	Not Only SQL
MB/s	Megabytes por segundo
MHz	Megahertz
NVME	Non-Volatile Memory Express
PHP	PHP: Hypertext Preprocessor
SGBD	Software Gerenciador de Banco de Dados
SQL	Structured Query Language
SSD	Solid State Drive
UFVJM	Universidade Federal dos Vales do Jequitinhonha e Mucuri

SUMÁRIO

1 INTRODUÇÃO

A utilização de Bancos de Dados (BDs) é uma realidade crescente que faz parte do cotidiano de todo mundo tecnológico. Eles estão presentes na utilização de telefones inteligentes, computadores, sites, softwares de todas escalas e em muitas outras situações. As aplicações atuais de BDs vão desde o simples armazenamento de dados até o seu processamento para geração e previsão de informações em diversos contextos: economia, políticas, esportes, clima, vendas, estudo de comportamento, dentre outras, logo é cada vez mais assertivo dizer que BDs são parte essencial da nova sociedade moderna.

A avaliação da importância dos BDs para a sociedade globalizada atual começa, sob um ponto de vista computacional, desde meados da década de 60, com o surgimento dos primeiros BDs computacionais com armazenamento simples. Avançando até os dias presentes, nos deparamos com uma realidade diferente, com exemplos de aplicações muito mais robustas, em cenários de análise em tempo real das informações, aplicadas a sistemas críticos, e utilizadas para geração de informações sobre quase todo o mundo digital.

Analisando sob a perspectiva comercial, a avaliação de qual BD utilizar pode ter grande impacto no funcionamento de empresas, afetando diretamente sua prestação de serviços, logo faz-se necessária a realização de pesquisas quantitativas e qualitativas sobre estes diferentes BDs, em diferentes aspectos, pensando-se na necessidade da informação sobre qual banco de dados é melhor para o negócio.

Embora os bancos de dados que se utilizam da modelagem relacional (surgidos na década de 70) sejam largamente utilizados até hoje, várias tecnologias de bancos de dados têm surgido desde aquele período. Podemos citar os bancos de dados de rede e hierárquicos, que também surgiram nessa mesma época, seguidos dos bancos de dados orientados a objetos, na década de 80.

Com a explosão da internet para uso comercial e o crescimento exponencial da quantidade e variedade dos dados em rede, uma nova tecnologia se destacou no mercado: os bancos de dados não relacionais, que focavam seu funcionamento em questões para as quais as tecnologia relacionais eram defeituosas.

Neste trabalho, considerando diferentes modelagens de uma mesma estrutura de dados de um comércio eletrônico, porém implementada para dois bancos de dados e modelagens diferentes (MariaDB e MongoDB), buscou-se indicadores de desempenho e consistência em operações básicas na tentativa de responder a questão: Para quais funcionalidades comuns de um *e-commerce* cada base de dados é melhor? Uma modelagem relacional (MariaDB) ou uma modelagem não-relacional (MongoDB) ?

1.1 Objetivos Gerais

O objetivo geral do seguinte trabalho é, a partir do estudo teórico e da formulação de uma aplicação piloto, realizar uma análise comparativa de modelagens alternativas de dados, no que diz respeito a:

1) Desempenho de operações básicas de manipulação de dados;
2) Garantia de consistência de dados.

1.2 Objetivos Específicos

Os objetivos específicos se resumem em:

1) Escolha de uma aplicação piloto;
2) Proposição de modelagens alternativas;
3) Elaboração de operações de inclusão, consulta, remoção e alteração de dados, visando suas análises.

Espera-se que os resultados deste protocolo experimental sirvam de base para sugerir quais contextos de uso da informação se adéquam mais às modelagens propostas.

Falando sobre a estrutura do trabalho, o capítulo 2 traz a revisão de literatura com uma abordagem histórica e também os conceitos fundamentais inerentes ao estudo.

O capítulo 3 - Materiais e Métodos - faz a descrição detalhada dos materiais e métodos utilizados na construção, com softwares, hardwares, e suas respectivas versões e especificações, além dos protocolos experimentais detalhados.

O capítulo 4 - Resultados e Discussão - mostra graficamente os resultados atingidos nos testes e discute-os sobre o ponto de vista da proposta principal.

O capítulo 5 - Considerações Finais - conclui o trabalho avaliando a obtenção dos resultados e conclusão dos objetivos, dando ao final, sugestões de estudos futuros.

2 REVISÃO DE LITERATURA

2.1 A História dos Bancos de Dados

A maneira mais tradicional de organizar dados, no período anterior a 1960, era por meio do armazenamento físico das informações em pastas, assim como ainda é feito nos dias de hoje. A partir do momento em que esse armazenamento físico começou a ficar inviável pelo seu tamanho, complexidade e dificuldade de efetuar consultas aos dados, pensou-se em um sistema computacional de armazenamento simples, que possibilitaria entender cada entidade (ou "coisa" do mundo real) e suas características como um arquivo de dados, o que acarretou em uma maior eficiência da manipulação dos dados. (SILBERSCHATZ; KORTH; SUDARSHAN, 2020).

Mesmo com a utilização destes softwares simples, as consultas ainda eram bastante dispendiosas, dado o fato de que o sistema não permitia facilmente que as entidades se relacionassem umas com as outras, tornando o trabalho muito menos intuitivo. (SILBERSCHATZ; KORTH; SUDARSHAN, 2020).

Assim como afirmado por Silberschatz, Korth e Sudarshan (2020), na década de 60 houve grande empenho da empresa *International Business Machines* (IBM) em criar BDs que relacionassem suas entidades, assim surgiram os conhecidos BDs hierárquicos e de rede. Os primeiros trabalhos com BDs em rede foram realizados por Charles Bachman, em 1964, caracterizando os dados por coleções de registros e relacionamentos por links, representados por meio de nós e linhas, respectivamente, permitindo o relacionamento de muitos para muitos entre os nós. Os BDs hierárquicos também utilizavam registros e links para representar os dados e relacionamentos, porém organizados em uma árvore com raiz. (HAIGH, 2011).

Ainda na década de 70, houve um grande avanço para a história dos bancos de dados relacionais, quando um pesquisador da IBM, Edgar Frank "Ted" Codd, publicou seu artigo intitulado *"A Relational Model for Data for Large Shared Data Banks"* (Um Modelo Relacional de Dados para Grandes Bancos de Dados Compartilhados), que inspirou os estudos sobre BDs relacionais, culminando na criação do Sistema Gerenciador de Banco de Dados (SGBD) denominado *R*, e da linguagem de consulta SQL - *Structured Query Language* (Linguagem de Consulta Estruturada) no final da década de 70. Interessante neste ponto conceituar SGBDs - Softwares Gerenciadores de Bancos de Dados, que são sistemas computadorizados cuja finalidade é armazenar informações e permitir que usuários acessem e atualizem informações nas bases de dados. (SILBERSCHATZ; KORTH; SUDARSHAN, 2020).

Em 1976, Dr. Peter Chen, um cientista da computação, propôs o Modelo de Entidade-Relacionamento (Modelo ER), uma modelagem que possibilitou abstrair e entender como os dados seriam utilizados, sem se preocupar com a estrutura lógica das tabelas. (THALHEIM, 2009).

Oliveira (2019) nos afirma que, dado que os conceitos fundamentais dos bancos de dados relacionais estavam estabelecidos, o passar das décadas possibilitou a disseminação e avanço de aplicações com o uso desta tecnologia, passando ela a ser considerada a tecnologia de bancos de dados mais comercialmente dominante. Nessa época surgiram diversos SGBDs

relacionais, tais como *DB2, Igress II, Informix Dynamic Server, Microsoft SQL Server, Oracle, Sybase Adaptive Server*,dentre outros. (DATE, 2004).

Apesar de sua larga utilização, ainda na década de 80 a comunidade de desenvolvimento de software começou a perceber que os BDs relacionais não eram adequados para a modelagem e armazenamento de aplicações em diversas áreas, tais como aplicações de áreas médicas, multimídias, física de energia elevada, etc; devido ao fato de aplicações desta natureza geralmente manipularem unidades de armazenamento de dados com estrutura muito variável (o que não se adequava ao modelo relacional, baseado na definição de tabelas com estrutura fixa). (DATE, 2004).

Assim, no final da década de 80 e início dos anos 90, em virtude também da crescente utilização dos conceitos oriundos das linguagens de programação orientada a objetos, surgiram os SGBDs orientados a objetos, que estruturam as informações como coleções de objetos de classes que se relacionam. No final da década de 90 surgem os SGBDs objeto-relacionais, que agregam conceitos da orientação objeto e do mundo relacional. (DATE, 2004).

Com o surgimento da *World Wild Web* e a intensificação do uso da internet durante a década de 90 e início do século 21, aumentou-se exponencialmente a quantidade de dados em rede, aumentando também a necessidade dos sistemas computacionais de adaptarem cada vez mais os seus bancos de dados às novas necessidades das companhias virtuais, como maior número de acessos simultâneos, e uma gerência de quantidades imensas de dados de tipos variados e altamente relacionados e o suporte ao processamento de altas taxas de transação (como em redes sociais). (SILBERSCHATZ; KORTH; SUDARSHAN, 2020; DATE, 2004).

Nesse contexto histórico, houve aumento na demanda por processamento mais rápido de dados, acesso simultâneo às bases de dados, bem como também a necessidade do chamado escalonamento horizontal dos sistemas computacionais. O escalonamento horizontal consiste em fazer com que várias máquinas menores trabalhem como um único sistema, ou seja, em um cluster, processando seus dados como uma unidade, para alcançar um desempenho maior. (FOWLER; SADALAGE, 2013).

Essa alternativa possibilita a utilização de hardware mais acessível, tornando a aplicação mais barata. Também, tal abordagem implica em alta confiabilidade do sistema, visto que mesmo que uma máquina falhe, o cluster como um todo pode ser criado para continuar funcionando. (FOWLER; SADALAGE, 2013).

O escalonamento horizontal surge como a opção mais barata e eficaz para implementação de um grande número de aplicações baseadas na Internet, entretanto os SGBDs relacionais não eram os mais indicados para tal arquitetura, pois não tinham sido projetados para este cenário computacional. Nesse momento surge a ideia dos BDs *Not Only SQL* (NoSQL), como solução para as inadequações dos BDs Relacionais. (DATE, 2004).

Date (2004) destaca que a primeira aparição do termo *NoSQL* foi para nomear um banco de dados relacional de código aberto criado por Carlo Strozzi, para definir seu sistema que não utilizava linguagem SQL para consulta, mas sim *shell scripts* para efetuar a manipulação das

informações. Importante entender também o conceito de *shell scripts*, que são como uma lista de comandos e parâmetros pré-definida que é executada em sequência, digitada pelo usuário na linha de comando. (JARGAS, 2008).

O termo *NoSQL*, como é conhecido hoje, surgiu de uma reunião de desenvolvedores interessados em abordagens alternativas para o gerenciamento de bancos de dados, em São Francisco, em junho de 2009. Apesar de não ser uma descrição exata das tecnologias daquela época, o termo que significa *Not Only SQL* (Não Somente SQL) foi utilizado para descrever bancos de dados que possuíssem um conjunto comum de características, tais como: utilizar modelagem de dados mais flexível que a relacional, ter bom desempenho em uma arquitetura em cluster, boa aplicabilidade para cenários de negócios da web, dentre outras. (DATE, 2004).

Muitas empresas adotaram a utilização do *NoSQL*, tais como o Facebook, com o projeto Cassandra, a Amazon com o Dynamo e a Google com a Bigtable. Podemos destacar também outros quatro BDs não relacionais mais conhecidos, que são o *MongoDB*, o *Hypertable*, o *CouchDB* e o *RavenDB*. (OLIVEIRA, 2019; MANOHARAN; LI, 2013).

Considera-se o *NoSQL* como um movimento que engloba diversas tecnologias que dispensam os BDs relacionais, e não uma tecnologia específica. É Importante também ressaltar que o *NoSQL* busca propor soluções para as situações onde o modelo relacional não se encaixa perfeitamente, e não servir como um substituto. (OLIVEIRA, 2019).

2.2 Banco de Dados Relacionas X NoSQL

Os bancos de dados, como afirma Silberschatz (2020) e Oliveira (2019), são um recipiente para uma coleção de arquivos de dados computadorizados que permite a automação de diversos tipos de negócios e diminuem a complexidade da aplicação. Já um Sistema Gerenciador de Banco de Dados (SGBD) tem a finalidade de armazenar informações e permitir que usuários acessem e atualizem elas. Sendo assim, podemos entender o Banco de Dados como os dados agrupados e o SGBD como um meio de acessa-los e modifica-los. (SILBERSCHATZ; KORTH; SUDARSHAN, 2020).

Segundo o que afirma Oliveira (2019), os BDs relacionais são tradicionalmente populares e difundidos. Eles funcionam baseado em relacionamentos entre tabelas que representam as entidades da aplicação, sendo que suas operações são geralmente feitas em linguagem *Structured Query Language* (SQL), através de transações. Transações são processos logicamente corretos de um ou mais acessos ao banco de dados, tais como leitura e atualização. (ELMASRI; NAVATHE, 2019).

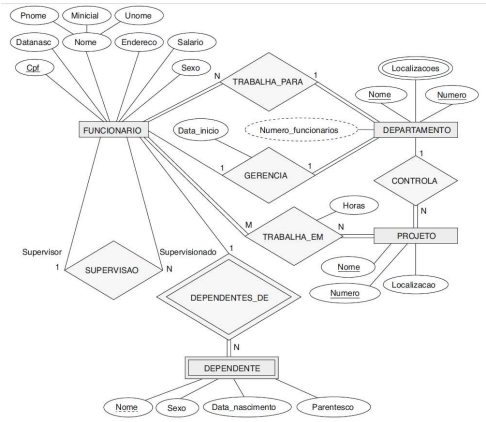
Geralmente para a representação gráfica simples dos BDs relacionais adota-se o modelo conceitual (Diagrama de Entidades e Relacionamentos) e o modelo lógico (Modelo Lógico Relacional). Estas representações gráficas auxiliam no processo inicial da construção do BD. (ELMASRI; NAVATHE, 2019)

A modelagem conceitual é a de mais alto nível, isto é, voltada para a visão geral do sistema, sendo de fácil entendimento para qualquer pessoa. A partir dessa modelagem, através do processo chamado de “mapeamento lógico”, é possível construir o modelo lógico do BD,

sendo esta uma abordagem mais exata do sistema, já esquematizando as informações em tabelas que serão futuramente utilizadas para a construção do modelo físico. (ELMASRI; NAVATHE, 2019).

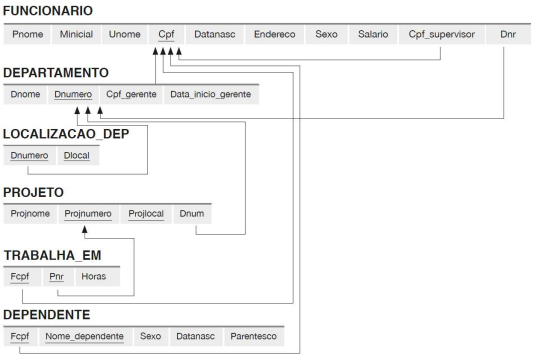
Para exemplificar a utilização das modelagens citadas acima, seguem dois diagramas (figura 1 e 2) sugeridos por Elmasri e Navathe (2019), ambos representando uma base de dados EMPRESA, com seus componentes, atributos e relações.

Figura 1 – Modelagem Conceitual - Diagrama ER para BD Empresa



Fonte: ELMASRI; NAVATHE, 2019.

Figura 2 – Modelagem Lógica - Base de dados Empresa



Fonte: ELMASRI; NAVATHE, 2019.

Toda transação dos BDs relacionais baseia o seu funcionamento nas propriedades ACID, que são definidas como:

> A – Atomicidade : Todas as operações da transação são executadas, ou seja, a transação será executada por completo ou não será executada.
>
> C – Consistência : Mesmo após uma operação ocorrer, o banco de dados deve permanecer consistente.
>
> I – Isolamento : Uma transação não será interferida por outras transações, é isolada das demais.
>
> D – Durabilidade : Uma vez que uma transação ocorra com sucesso, seu efeito não poderá ser desfeito, mesmo em caso de falha. (FERNANDES; SILVA, 2019, p.3).

Os aspectos positivos dessa abordagem relacional são a consistência e a redução da redundância. Em contrapartida a essas vantagens, com a intensificação do uso da internet no final da década de 90, foi exigido dos SGBDs o tratamento de um maior número de acessos simultâneos aos BDs, grandes volumes de dados variados, alta escalabilidade do sistema, flexibilidade dos esquemas e tratamento de dados de maneira distribuída, aspectos fracos das tecnologia relacionais. (DATE, 2004; FERNANDES; SILVA, 2019).

Fowler e Sadalage (2013) afirmam que, para atender essas demandas que não se adaptam aos BDs relacionais, temos os BDs *Not Only SQL* (*NoSQL*). Eles trabalham possibilitando que em sua estrutura *Binary JSON (BSON)*, sejam agregadas informações complexas e semiestruturadas, sendo acessadas com mais rapidez e disponibilidade. Galicz, Schmid e Reinhardt (2015) ainda destacam que nessa época (início dos anos 2000) a abordagem não relacional objetivava a eliminação de todas as limitações das relações rigorosas que eram impostas aos BDs relacionais.

Fowler e Sadalage (2013, p. 15) destacam um conceito importante para esse tipo de BD, que é a sua unidade de armazenamento, chamado de agregado: "armazenamentos explícitos de uma estrutura rica de dados intimamente relacionados que é acessada como uma unidade". Eles também afirmam que é importante também ressaltar que essa orientação a agregado auxilia bastante na execução no cluster, sendo esse o motivo crucial para ascensão do *NoSQL*.

Pode-se, como listado por Sadalage e Fowler (2013), citar três tipos de Bancos de Dados Orientados a Agregado:

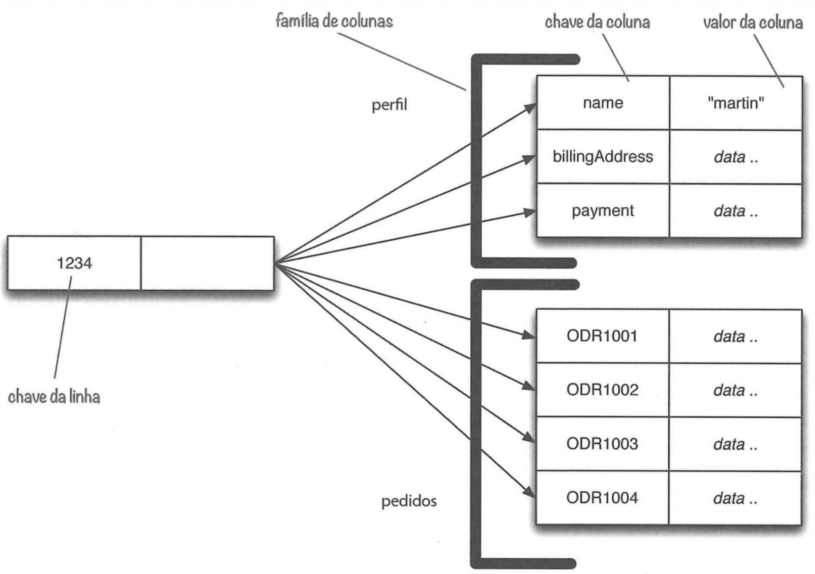
1) Família de Colunas;
2) Baseado em Chave-valor;
3) Documentos.

Ainda é ressaltada a existência de BDs que não se encaixam exatamente nas categorias, citando o *OrientDB*, que é considerado tanto um BD de documentos quanto de grafo. Tomando como exemplo o citado acima, podem haver tecnologias de BDs que misturam os conceitos de diferentes tipos de bancos. (FOWLER; SADALAGE, 2013).

Sahatqija *et al*. (2018) descrevem que os BDs de Família de Colunas trabalham com a ideia de agrupar diversas colunas para uma única linha, estruturando mais logicamente o banco e fazendo com que as leituras fiquem mais rápidas. Como mostrado na figura 3 abaixo, há uma única chave de linha que identifica todo um bloco de informações sobre um cliente, seu perfil e seus pedidos. (FOWLER; SADALAGE, 2013).

Fowler e Sadalage (2013) explicam que no BD de chave-valor tem-se uma chave identificadora para cada agregado, podendo ser adicionadas informações de qualquer estrutura no agregado, limitado somente pelo tamanho. A pesquisa nesse BD é feita exclusivamente pela chave que o identifica. Sahatqija *et al*. (2018) afirmam também que os dados desse tipo de BD podem ser armazenados tanto em forma de linhas, como numa base relacional, quanto em formatos *JavaScript Object Nation* (JSON), ou estruturas semiestruturadas similares. A seguir temos uma estrutura de chave-valor, na figura 4, onde cada chave identifica um tipo diferente de

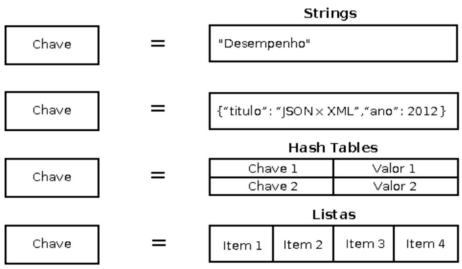
Figura 3 – Exemplo de BD de Família de Colunas



Fonte: FOWLER; SADALAGE, 2013.

dado. Em cada linha temos uma chave primária e para cada uma delas seus diferentes conjuntos de valores (logo, o nome chave-valor).

Figura 4 – Estrutura de Chave-Valor



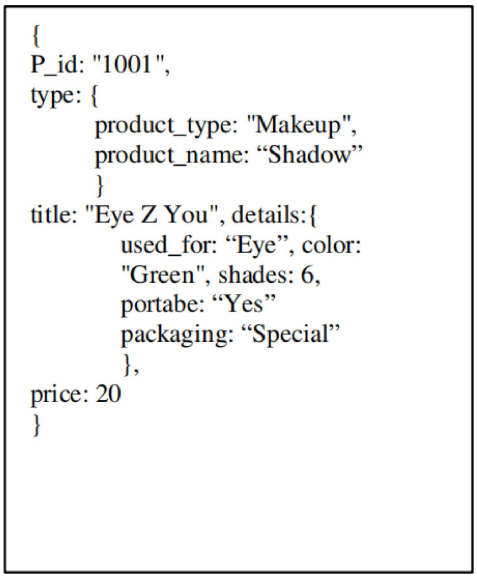
Fonte: OLIVEIRA, 2017.

Assim, tem-se liberdade para inserir nesse banco, porém a consulta é limitada somente pela chave, ou seja, não é possível realizar consulta de acordo com a estrutura do agregado. (FOWLER; SADALAGE, 2013).

No BD orientado a documentos, a estrutura do documento deve ser informada no ato da inserção dos dados, dessa forma a pesquisa pelos dados pode ser feita com base em qualquer componente da estrutura do documento, e não somente pela sua chave de identificação, além de se poder recuperar apenas parte do agregado numa consulta, o que é uma vantagem, pela sua praticidade. Sendo assim, para esse tipo de BD, a limitação fica na estrutura prévia que precisa existir no momento da inserção, mas a flexibilidade está na consulta, que pode ser feita pelos campos, e no armazenamento versátil, que pode se diferenciar entre os documentos. (FOWLER; SADALAGE, 2013).

Abaixo, na figura 5, podemos observar um exemplo da estrutura de um documento que guarda as informações do produtos de maquiagem. Nele há uma chave de identificação "P_id"e vários outros campos que guardam informações sobre o produto (tipo, título e preço). Note que há campos que são formados por outros campos em seu interior, o que gera grande flexibilidade para o armazenamento de dados.

Figura 5 – Modelagem de Dados Orientada a Documentos - BD Produtos



Fonte: MAOWA *et al.*, 2017.

Tendo-se em mente que uma coleção num banco de dados orientado a documentos seria o equivalente à tabela do BD Relacional, é importante destacar que a flexibilidade na estrutura desse tipo de BD se dá principalmente pelo fato de que a estrutura dos documentos pode diferir, mesmo que eles pertençam a uma mesma coleção. (NAYAK; PORIYA; POOJARY, 2013).

Os sistemas *NoSQL* seguem, ao invés das propriedades ACID, as propriedades *BASE- Basically Available, Soft state, Eventual consistency* (Basicamente Disponível, Estado Flexível e Consistência Eventual). (GALICZ; SCHMID; REINHARDT, 2015)

Brito (2013) citado por Claro (2017), explica sobre essas propriedades, definindo: basicamente disponível - o sistema funciona sem falhas; estado flexível - não há necessidade de consistência o tempo todo; consistência eventual - o sistema torna-se consistente no momento devido. Segundo o trabalho de Diana e Gerosa (2010), esse termo BASE define que o sistema deve tolerar inconsistências temporárias quando se prioriza a disponibilidade, e não define um conjunto de propriedades bem definidas, como as ACID.

Fowler e Sadalage (2013) apontam que as principais vantagens dos BDs não-relacionais são a compatibilidade com trabalho em cluster, o tratamento de grandes quantidades de dados em curto tempo e a flexibilidade de seus esquemas. Adversamente, eles ainda apontam

suas desvantagens: a perda de desempenho quando utilizado relacionamento entre seus agregados e a falta da consistência em lugar da alta disponibilidade.

É comumente citado o Teorema CAP em estudos, quando se fala da comparação das propriedades de bancos de dados distribuídos, destacando o seu significado, definido por Sahatqija (2018): CAP - *Consistency, Avaiability and Partition Tolerance* - Consistência, Disponibilidade e Tolerância a partição.

Sobre as suas propriedades, Sahatqija (2018) define:

a) Consistência - o dado é sempre o mesmo, em todas réplicas de todos servidores;
b) Disponibilidade: os dados devem estar permanentemente disponíveis sempre;
c) Tolerância à partição: a base de dados é funcional mesmo com falhas de rede ou nas máquinas.

Tratando-se da comparação entre a aplicação de diferentes bancos de dados para diversos contextos, o Teorema CAP traz uma análise interessante para este estudo. Assim como citado por Khine e Wang (2019), este teorema defende que, dada as três prioridades, consistência, disponibilidade e tolerância a partição, é impossível implementar todas as três em um único BD, sendo que o fortalecimento de duas dessas propriedades na base, sempre torna a terceira propriedade mais suscetível a falha.

Dada ainda a análise do trabalho, sob o ponto de vista do Teorema CAP definido acima, pode-se afirmar que, nos BDs relacionais, a consistência e disponibilidade são aspectos que se destacam, enquanto nos BDs não relacionais as características mais fortes são a disponibilidade e tolerência a partição. (FERNANDES; SILVA, 2019).

Assim, dada a necessidade da distribuição de dados das bases, as consistências tendem a ser comprometidas, tendo a comunidade de bancos de dados mudado, muitas vezes, para esse tipo de desenvolvimento. (KHINE; WANG, 2019).

Aricélio e Petrônio (2019) ainda elencam, no quadro da figura 6 as principais divergências existentes entre os BDs relacionais e não-relacionais, mostrando objetivamente as características de cada um, que foram citadas ao longo do capítulo.

2.3 Persistência Poliglota de Dados

Dadas as diferenças entre os SGBDs relacionais e aqueles classificados como sendo NoSQL apresentadas pela figura 6, pode-se pensar, num primeiro momento, que sempre haverá o uso destas diferentes tecnologias em contextos de negócio diferentes, entretanto, tem sido proposta a utilização destas diferentes abordagens de SGBDs para um mesmo contexto de aplicação, porém em funcionalidades diferentes. Tal abordagem é definida por Fowler e Sadalage (2013) como persistência poliglota de dados.

A justificativa para essa abordagem poliglota é que, para um contexto de negócio, podem existir diferentes grupos de funcionalidades de um sistema de informação que automatiza este negócio que exigem diferentes níveis de garantia das propriedades listas acima como: consistência, atomicidade, durabilidade, etc; e que utilizar uma mesma tecnologia de BD para au-

Figura 6 – Quadro de Comparação BDs Relacionais x BDs NoSQL

Característica	Relacional	NoSQL
Esquema	Rígido	Flexível
Tran. ACID	Sim	Não
A – Atomicidade	Sim	Não
C – Consistência	Sim	Não
I – Isolamento	Sim	Sim
D - Durabilidade	Sim	Não
CAP	CA	AP
Replicação	Sim	Sim
Part. Vertical	Sim	Não
Part. Horizontal	Não	Sim
Sharding	Não	Sim

Fonte: FERNANDES; SILVA, 2019.

tomatizar todas estas funcionalidades resultaria em soluções de baixo desempenho. (MAOWA *et al.*, 2017).

Considerando o afirmado acima, torna-se importante fazer estudos como o deste trabalho para apontar direcionamentos sobre possibilidades de uso de várias modelagens de dados para funcionalidades de diferentes contextos.

Fowler e Sadalage (2013) apontam que não somente a questão do desempenho deve ser avaliada, mas sim questões relativas à consistência, atomicidade e durabilidade, que possam garantir a boa execução de cada tipo de funcionalidade. Neste sentido, funcionalidades relativas à registro de operações financeiras, por exemplo (vendas, transferências de valores, etc), devem ocorrer sem erro, tendo o acesso exclusivo aos dados no momento de sua execução, e não podem ocorrer parcialmente. Já a geração de um relatório gerencial sobre o que determinados clientes compram em comum numa determinada região, pode ser gerado enquanto novos pedidos dos clientes estão sendo registrados, pois possivelmente a "defasagem"dos dados do relatório sobre os novos pedidos pode não ter um impacto relevante para quem for utilizar o relatório.

Oliveira (2017) afirma que pode não ser a melhor solução se utilizar de uma mesma abordagem de SGBD para implementar diferentes funcionalidades de um SI, como: registro de transações financeiras/comerciais diários do negócio, registro de log de usuário, geração de relatório gerenciais de *Business Intelligence* (BI), etc. No contexto de uma aplicação de *e-commerce*, por exemplo, Fowler e Sadalage (2013) argumentam que a abordagem NoSQL de armazenamento chave-valor poderia ser utilizada para armazenar os dados do carrinho de compras antes do pedido confirmado, mas que o registro dos pedidos confirmados pelo clientes seriam feitos numa base de dados relacional (maior garantia de consistência, durabilidade, isolamento, etc).

2.4 Estudos Relacionados

Há diversos estudos que comparam as características entre os BDs relacionais e não relacionais, assim como também há comparações de seus desempenhos em diferentes operações,

para diferentes cenários, utilizando tecnologias diversas. Destaca-se aqui alguns estudos que foram mais próximos, em termos de metodologia, ao presente trabalho.

Comparações de performance de operações básicas foram feitas por Abdullah e Zhuge (2015), utilizando código PHP, SGBDs MySQL e MongoDB. Nesse estudo, foram feitas inserções de mil a cem mil documentos/linhas nas respectivas bases de dados, com a demonstração gráfica dos tempos resultantes de cada BDs para as operações de inserção e consulta de dados. Nas conclusões é evidenciado que na maioria dos cenários de inserção de dados o MongoDB tem melhor desempenho, principalmente para inserção de imagens. Para as operações de consulta, o MongoDB também se mostrou mais eficiente, principalmente para recuperar informações em grandes bases.

Um estudo similar foi feito por Jose e Abraham (2020), utilizando o MySQL Workbench e o MongoDB Studio 3T. Utilizando das ferramentas descritas, os autores realizaram testes de performance de operações de consulta, alteração e inserção de dados em dois *datasets*, um sobre prédios da cidade de Chicago e outro sobre dados ambientais e de radiação de todo o planeta após o desastre no Japão em março de 2011. Os resultados, apresentados graficamente, demonstraram que o MongoDB tem melhor performance para executar todas as operações, principalmente para bases de dados grandes. Ainda é destacado ao final que, para empresas que lidam com *BigData*, a melhor opção é migrar para a utilização de BDs não relacionais.

Avaliando um terceiro estudo,Galicz, Schmid e Reinhardt (2015) avaliaram as performances dos bancos de dados PostgreSQL, *MongoDB* e *CouchDB* para uma aplicação geográfica que utilizou dados da *OpenStreetMap*. Avaliando-se as operações de consulta das informações geográficas em diferentes aspectos, conclui-se que, apesar de limitadas as operações geográficas disponíveis nas bases não relacionais, para operações de consulta de atributos, elas apresentam desempenho melhor, principalmente para operações com grandes bases de dados. Avaliadas as operações com funções geográficas, para bases pequenas com geometria complexa, a base relacional performou melhor.

3 MATERIAIS E MÉTODOS

O objeto de estudo do presente de trabalho é um banco de dados de comércio eletrônico, analisado sob diferentes modelagens e implementações em dois bancos de dados diferentes: MariaDB e MongoDB.

A ideia original de comparar aspectos desses dois bancos surgiu durantes a disciplina de Bancos de Dados II, num seminário onde foram explicados os fundamentos do MongoDB e dos BDs não-relacionais como um todo. Importante destacar que a disciplina também influenciou nas escolhas dos bancos de dados utilizados, quando se destacou o MongoDB nos estudos sobre bases não relacionais, e o MariaDB, que surgiu como uma alternativa vinda do MySQL estudado em sala de aula.

3.1 Materiais

Descrevendo as tecnologias utilizadas na experimentação do trabalho, inicia-se pelo hardware, composto por uma máquina Dell modelo Inspiron 7572 com:

a) Processador Intel Core i7 8550U 1.8GHz;

b) Armazenamento principal 8GB DDR4 2400MHz (1 slot);

c) Armazenamento secundário SSD NVME NM610 500GB 2100MB/s.

Os softwares utilizados na execução do estudo foram:

a) Sistema Operacional Windows 10 64 bits;

b) Linguagem de Programação PHP v.8.0.3;

c) Banco de dados MongoDB v.4.4.4;

d) Interface gráfica para usuário MongoCompass v.1.28.1;

e) Banco de dados MariaDB v.10.4.18;

f) Interface gráfica para usuário MySQl Workbench v.8.0.23;

g) Ambiente de desenvolvimento Microsoft Visual Studio Code v.1.59.1;

h) Servidor Apache v.2.4.46;

i) Gerenciador de drivers PHP Composer v.2.0.11;

j) Extensão Mongo para PHP v.8.0 .

3.2 Métodos

3.2.1 Configurações Prévias

Antes de iniciar a construção do testes, foram necessárias algumas configurações prévias de software, iniciando pelos downloads disponíveis em (links clicáveis no texto):

a) XAMPP;

b) MySQL Workbench;

c) MongoDB;

d) Composer;

e) Extensão Mongo para PHP;

f) MongoCompass;

g) Microsoft Visual Studio Code.

3.2.1.1 Configurações para o MongoDB

Feitos os downloads necessários, a extensão .dll nomeada “php_mongobd”deve ser colocada na pasta “xampp/php/ext”. Também é necessária, na mesma pasta, a edição do arquivo “php”(parâmetro de configuração), adicionando a linha “extension=php_mongodb.dll”na seção de “Dynamic Extensions”.

O Composer será usado para instalar a biblioteca do Mongo no PHP. Após instalado, abrir uma janela do prompt de comando e digitar, na pasta em que deseja ser feita a instalação, o comando:

```
composer require mongodb/mongodb
```

Após instalada a extensão e a biblioteca, sempre que for necessária a utilização de algum comando do MongoDB em um arquivo PHP, utilizar no início do código o seguinte comando

```
require 'vendor autoload.php';
```

3.2.1.2 Configurações para o MariaDB

Para que as inserções na base de dados relacional ocorram sem erros, são necessários ajustes em algumas configurações do servidor, no arquivo my.ini, que fica disponível na interface do XAMPP, em “Config”de MySQL.

Na seção “[mysqld]”, para configurar um tempo muito grande de espera de resposta de uma consulta, adicionar a linha:

```
wait_timeout=10000000;
```

Para permitir que o servidor mysql importe dados de arquivos locais à ele (na mesma máquina que a dele), editar a linha abaixo para:

```
local-infile = 1
```

Configurar o *buffer* de tabelas com 80% do valor total de memória principal da máquina (se atentando para o possível consumo já existente na memória) com a seguinte edição nas linhas:

```
innodb = ON;
innodb_buffer_pool_size = XXXM #80% da memória principal
```

3.2.2 Sobre a Aplicação Piloto

A aplicação piloto para desenvolvimento selecionada foi a de um banco de dados para um comércio eletrônico, que é um tipo de aplicação comum na Web e cujas informações podem ser armazenadas em bancos de dados relacionais ou *NoSQL*. O BD relativo a esta aplicação armazena informações sobre o cliente e sobre seus pedidos no comércio.

Essencialmente, as principais funcionalidades desses sistema são:

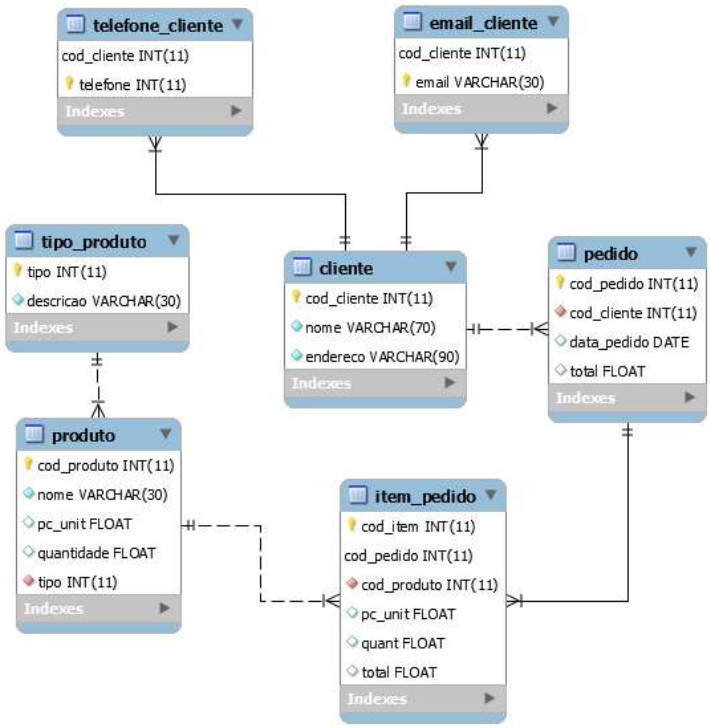
a) Armazenamento de dados sobre os clientes com suas informações pertinentes;

b) armazenamento de dados dos produtos do comércio;

c) Mapeamento dos pedidos por cliente, definindo data e preço total a partir da associação de informações.

Abaixo na figura 7 é possível observar o Diagrama Relacional de Tabelas da Base de Dados *E-commerce* gerado através do MySQL Workbench, que mostra, de maneira geral, suas entidades e os relacionamentos entre elas.

Figura 7 – Diagrama Relacional - *E-commerce*



Fonte: Próprio autor.

Para a estruturação relacional do banco, utilizou-se o MariaDB, através do MySQL Work Bench, que é uma ferramenta visual de design. Nele foi possível estruturar todas as tabelas e seus relacionamentos, detalhando todas as propriedades inerentes à aplicação.

No anexo A é possível observar o código de criação da base de dados relacional, que foi feita utilizando o MySql Workbench.

3.2.3 Modelagem NoSQL

A estruturação não-relacional do banco se deu através de paginas PHP que se conectaram ao MongoDB diretamente, realizando assim a criação da base de dados. Vale ressaltar a utilização do Mongo Compass, que é uma ferramenta que serve para a realização de operações básicas, mas que neste caso foi usada somente para melhor visualização dos dados através de sua interface gráfica. O algoritmo que cria as base de dados não relacionais é o mesmo utilizado nos testes de suas respectivas inserções.

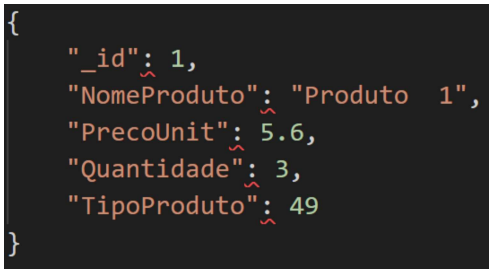
Na figura 8 é possível observar a estrutura básica do Banco de Dados E-commerce, na implementação não-relacional de uma coleção.

Figura 8 – Modelagem Não-Relacional - Uma Coleção *E-commerce*

Fonte: Próprio autor.

Para a modelagem não-relacional de duas coleções, as mesmas podem ser apresentadas pelas figuras 10 e 9, cada uma representando uma coleção da base, sendo a coleção shopping, que mapeia os clientes com seus pedidos, e a coleção produtos, que contém todos os produtos do comércio.

Figura 9 – Modelagem Não-Relacional - Duas Coleções - Produtos



Fonte: Próprio autor.

3.2.4 Protocolo Experimental

Os testes realizados nas bases de dados se resumem na execução dos algoritmos PHP que realizam as operações básicas nas diferentes modelagens dos bancos e criam arquivos '.csv' com o tempo de execução das operações.

Figura 10 – Modelagem Não-Relacional - Duas Coleções - Shopping

```
{"Codigocliente": 1,
    "Nome": "Cliente 1",
    "Endereco": "Endereco 1",
    "Pedidos": [{
        "Codigo": 1,
        "Data": "2019/05/15",
        "Itens": [{
            "Codigoitem": 1,
            "CodigoProduto": 744,
            "PrecoUnit": 3.74,
            "Quantidade": 3,
            "Total": 11.22
        }, {
            "Codigoitem": 2,
            "CodigoProduto": 187,
            "PrecoUnit": 5.6,
            "Quantidade": 6,
            "Total": 33.6
        }, {
            "Codigoitem": 3,
            "CodigoProduto": 171,
            "PrecoUnit": 5.89,
            "Quantidade": 4,
            "Total": 23.56
        }],
        "Total": 68.38
    },
    ...]
    "Fones": [72895600, 31583142, 61279915],
    "Emails": ["cliente1@provedor1.com.br", "cliente1@provedor2.com.br"]
}
```

Fonte: Próprio autor.

Importante ressaltar que, da maneira como os algoritmos foram implementados, os teste obedecem um mínimo estatístico de 30 repetições da operação, sendo que o tempo retornado nos arquivos “.csv”se restringem somente à contabilização do tempo das operações em si, e não da execução do algoritmo como um todo.

As modelagens propostas, e já citadas acima são:

a) Modelagem Relacional Única;

b) Modelagem Não-Relacional com uma coleção;

c) Modelagem Não-Relacional com duas coleções.

Quanto às operações básicas testadas nos algoritmos, temos:

a) Inserção de dados;

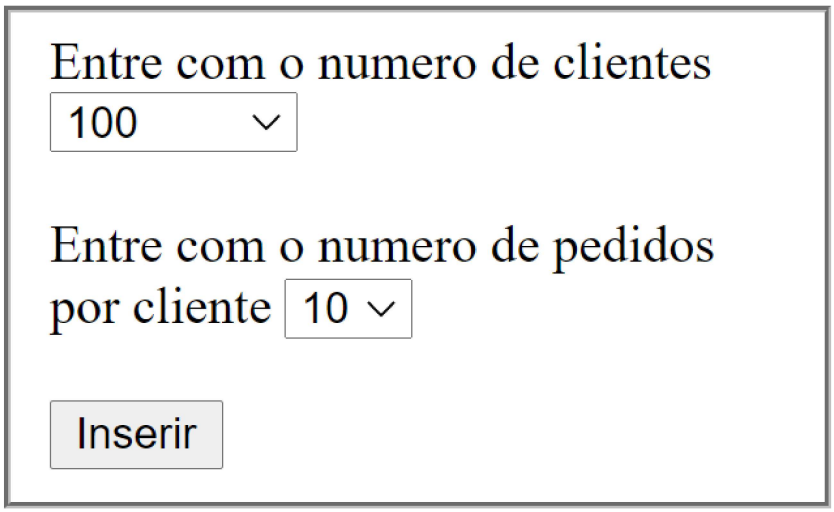
b) Consulta de dados;

c) Alteração de dados;

d) Remoção de dados;

Vale destacar que, afim de evitar a repetição desnecessária de código nos anexos, a interface das operações seguiu um padrão para todas execuções, logo, abaixo temos a figura 11 que o representa, e no anexo B, temos o seu respectivo código.

Figura 11 – Interface de Operações



Fonte: Próprio autor.

A execução dos testes se deu da seguinte forma, para as bases de dados relacionais:

a) Ativação do servidor Apache;
b) Execução do código PHP relativo a consulta;
c) Verificação, pelo MySql Workbench ou linha de comando, da efetividade da operação na base;
d) Salvamento dos arquivos CSVs gerados nas pastas do Google Drive para acesso coletivo do aluno e orientador;
e) Salvamento do algoritmo PHP resultante no Google Drive para acesso coletivo do aluno e orientador.

Para as bases de dados não-relacionais, o processo foi bastante parecido, adicionando apenas a ativação do MongoDB Database Server e mudando a interface gráfica, que no Mongo é a Compass.

Logo, antes que se apresentem os resultados obtidos, todos os algoritmos utilizados para efetuar as operações estão detalhados nos anexos B ao W.

4 RESULTADOS E DISCUSSÃO

Cada tamanho da base de dados foi definido por uma combinação entre o número de clientes e pedidos por cliente, evidenciando nesse aspecto a análise fatorial com cinco níveis para a variável clientes e 3 níveis para a variável número de pedidos, resultando nas bases de dados descritas abaixo:

a) Base 1: modelagens com 100 clientes e 10 pedidos por cliente;
b) Base 2: modelagens com 100 clientes e 30 pedidos por cliente;
c) Base 3: modelagens com 100 clientes e 50 pedidos por cliente;
d) Base 4: modelagens com 1000 clientes e 10 pedidos por cliente;
e) Base 5: modelagens com 1000 clientes e 30 pedidos por cliente;
f) Base 6: modelagens com 1000 clientes e 50 pedidos por cliente;
g) Base 7: modelagens com 10000 clientes e 10 pedidos por cliente;
h) Base 8: modelagens com 10000 clientes e 30 pedidos por cliente;
i) Base 9: modelagens com 10000 clientes e 50 pedidos por cliente;
j) Base 10: modelagens com 100000 clientes e 10 pedidos por cliente;
k) Base 11: modelagens com 100000 clientes e 30 pedidos por cliente;
l) Base 12: modelagens com 100000 clientes e 50 pedidos por cliente;
m) Base 13: modelagens com 1000000 clientes e 10 pedidos por cliente;
n) Base 14: modelagens com 1000000 clientes e 30 pedidos por cliente;
o) Base 15: modelagens com 1000000 clientes e 50 pedidos por cliente;

Os dados básicos dos clientes foram predefinidos, bem como os dados dos produtos. Para cada pedido, foram escolhidos aleatoriamente os produtos nele contidos, bem como a quantidade e o valor unitário, sendo o valor total de cada item de pedido calculado com base na quantidade e valor unitário. Os testes foram executados em uma máquina com 1.8GHz de poder de processamento, 8GB de memória RAM e 500GB de armazenamento SSD. As versões dos SGBDs utilizados foram: MongoDB versão 4.4.4 e MySql versão 8.0.3. As configurações padrão dos SGBDs foram mantidas.

4.1 Consultas escolhidas

As consultas elencadas foram as seguintes:

a) Inserção em massa de informações na base de dados: Neste caso todas informações da base de dados foram inseridas na base de dados inicialmente vazia. No caso da base de dados relacional, as informações foram transferidas de arquivos .csv para as tabelas através de execuções do comando LOAD DATA INFILE em arquivos bash chamados por um código PHP. No caso das bases de dados não relacionais, os dados foram estruturados na forma de uma documento do tipo

JSON em códigos PHP e então transferidas para a base de dados diretamente, através do método insert da classe *Bulk Write*. O tempo de processamento das inserções foi obtido através do uso da função PHP *microtime* exatamente antes e depois da chamada dos comandos de inserção.

b) Consulta de histórico de vendas de produtos: Para esta consulta, a cada execução, um produto diferente foi escolhido aleatoriamente e foram obtidas as informações sobre a venda deste nos pedidos dos clientes (o código do cliente, o código do produto, o nome do produto, o tipo do produto, o código do item no pedido, o preço unitário, a quantidade e o total do item referente ao produto). Os tempos de execução foram obtidos de maneira similar ao da consulta anterior, e os comandos relativos à consulta foram chamados à partir de um código PHP.

c) Consulta de sumário de produtos mais vendidos: Nesta consulta, a cada execução, são retornadas informações sobre os "n" produtos mais vendidos (onde "n" é escolhido aleatoriamente), a saber: o código do produto, o nome do produto, e o valor total das vendas do produto. Trata-se portanto de uma consulta que envolve um maior processamento computacional do que a anterior, pois é necessário que se calcule o somatório das vendas para cada produto, que se ordene os dados conforme o valor das vendas, para então obter aqueles requeridos. Os tempos de execução foram obtidos de maneira similar ao que foi feito nas consultas anteriores, e os comandos relativos à consulta foram chamados à partir de um código PHP.

d) Consulta de sumário de Clientes maiores compradores: Nesta consulta, a cada execução, são retornadas informações sobre os "n" clientes maiores compradores (onde "n" é escolhido aleatoriamente), a saber: o código do cliente, o nome do cliente e o somatório dos totais dos pedidos do cliente. De maneira similar à consulta anterior, envolve um maior processamento computacional em relação à consulta b).

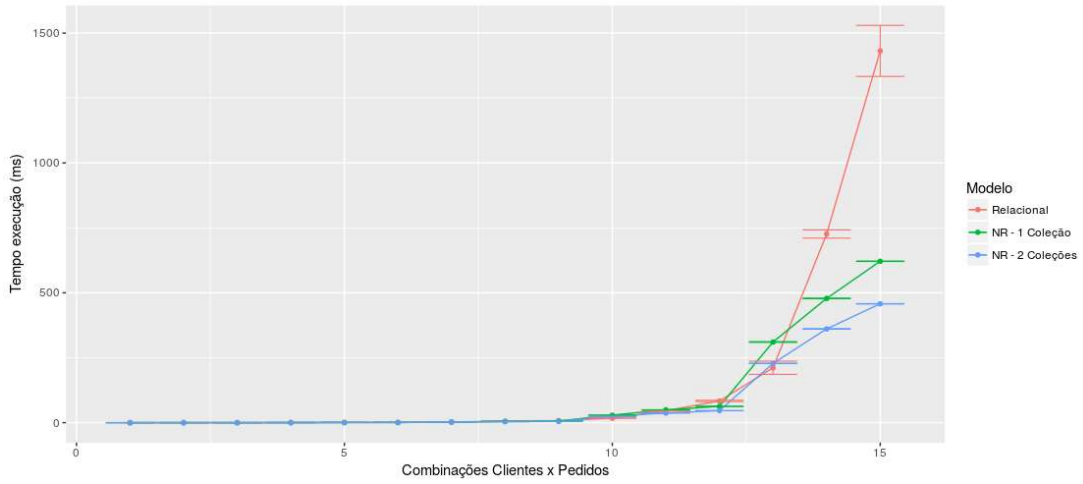
e) Alteração de informações de clientes: Essa consulta simula o processo típico de uma funcionalidade de alteração de dados, em que os dados de endereço de um cliente escolhido aleatoriamente são atualizados para um novo valor.

f) Exclusão de informações de clientes: Essa consulta simula o processo típico de uma funcionalidade de exclusão de dados, em que os dados de um cliente escolhido aleatoriamente são excluídos da base de dados.

4.2 Resultados

Uma vez determinadas as 6 consultas acima, a seguir serão apresentados os resultados referentes aos tempos de execução das mesmas em relação às 3 modelagens, considerando os diferentes tamanhos para as bases de dados.

Figura 12 – Inserção de Clientes



Fonte: Próprio autor.

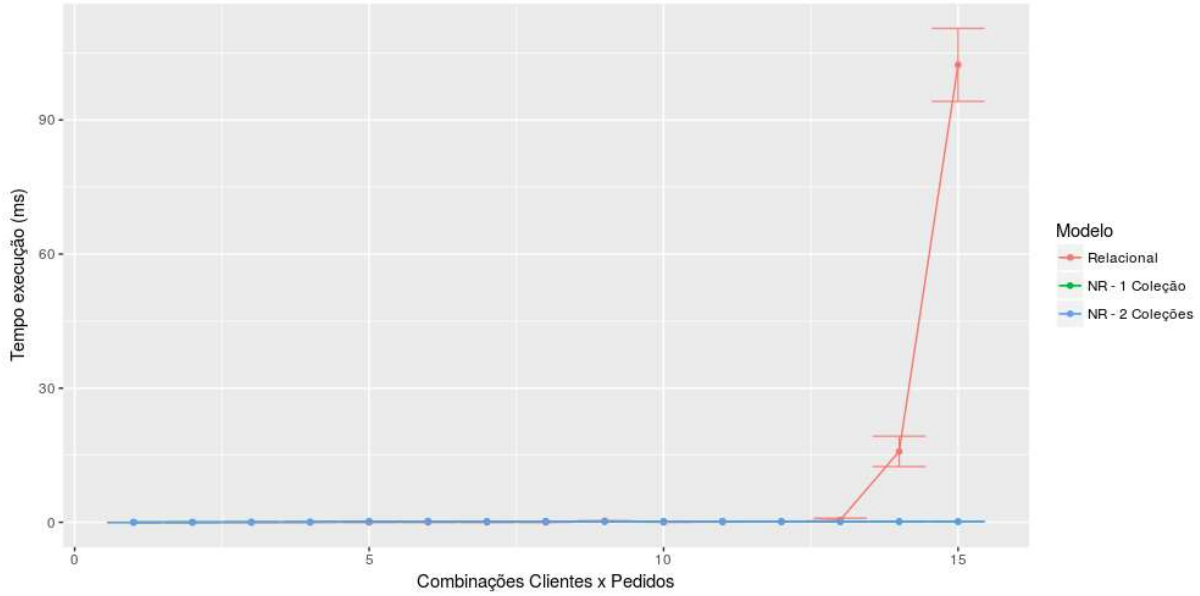
A figura 12 mostra a evolução dos tempos de execução médios (bem como os intervalos de confiança) de inserção de dados para diferentes tamanhos de bases de dados (Base 1 a Base 15), considerando os 3 tipos de modelagens. O gráfico sugere uma performance similar do procedimento de inserção para as 3 modelagens, para a maioria dos tamanhos de base de dados, sendo que para os maiores tamanhos de bases de dados começa-se a evidenciar uma diferença de performance, com o modelo não relacional com 2 coleções tendo melhor performance, seguido pelo modelo não relacional com 1 coleção, e o modelo relacional tendo pior desempenho, sugerindo uma pior escalabilidade.

Do ponto de vista estatístico, os dados das médias de tempo de inserção apresentaram distribuições não normais, para todas as modelagens (isso foi verificado com o teste de Shapiro-Wilk). Em função disto, a fim de verificar a existência de diferença estatística na performance do procedimento de inserção, considerando sua aplicação nas 3 modelagens para o conjunto das 15 diferentes bases de dados, foi executado o Teste não paramétrico de Friedman. O nível de significância estatística considerado em todos os testes foi de 0.05.

O resultado do teste (valor p = 0.4493) indica que, considerando os 15 tamanhos diferentes da base de dados, para nenhuma das modelagens o tempo médio de inserção foi predominantemente melhor do que os das demais modelagens. O que se pode inferir dessa informação e do gráfico da figura 1 é que, considerando o procedimento de inserção em massa de dados, para os tamanhos de bases de dados utilizadas, as 3 modelagens são igualmente competitivas, mas com uma sugestão de melhor escalabilidade das modelagens não relacionais.

A figura 13 ilustra a evolução dos tempos médios (e seus intervalos de confiança) de obtenção das informações das vendas de produtos escolhidos aleatoriamente, considerando as 3 modelagens e os 15 tamanhos de bases de dados. Novamente o gráfico sugere um comportamento competitivo das 3 modelagens até um certo tamanho de bases de dados, com o tempo de

Figura 13 – Consulta de Produtos Aleatórios

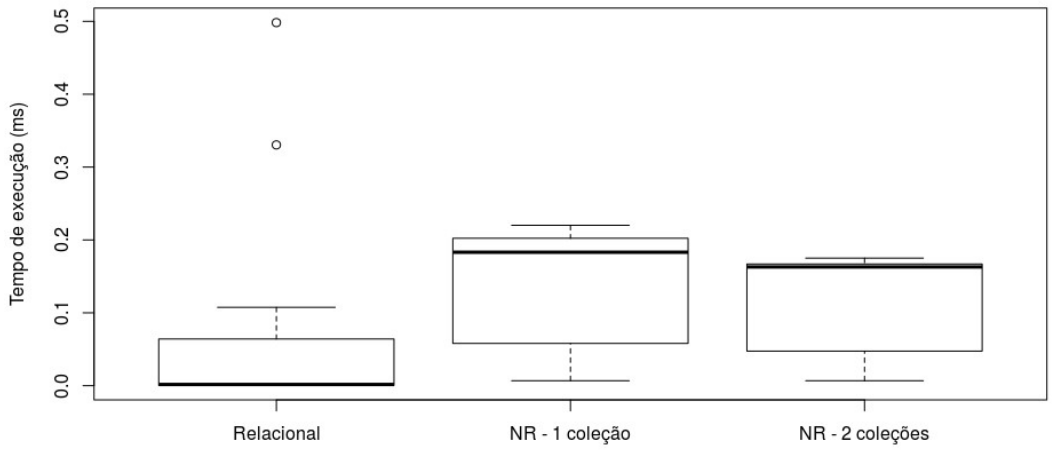


Fonte: Próprio autor.

execução da modelagem relacional tornando-se excessivamente alto em relação aos das demais modelagens.

A fim de explorar melhor as performances das 3 modelagens para a maioria dos tamanhos de BDs, a figura 14 apresenta boxplots com a distribuição dos tempos de consulta das 3 modelagens, considerando apenas os 13 primeiros tamanhos de BDs.

Figura 14 – Box Plot - Consulta Produtos Aleatórios - 13 Primeiros BDs



Fonte: Próprio autor.

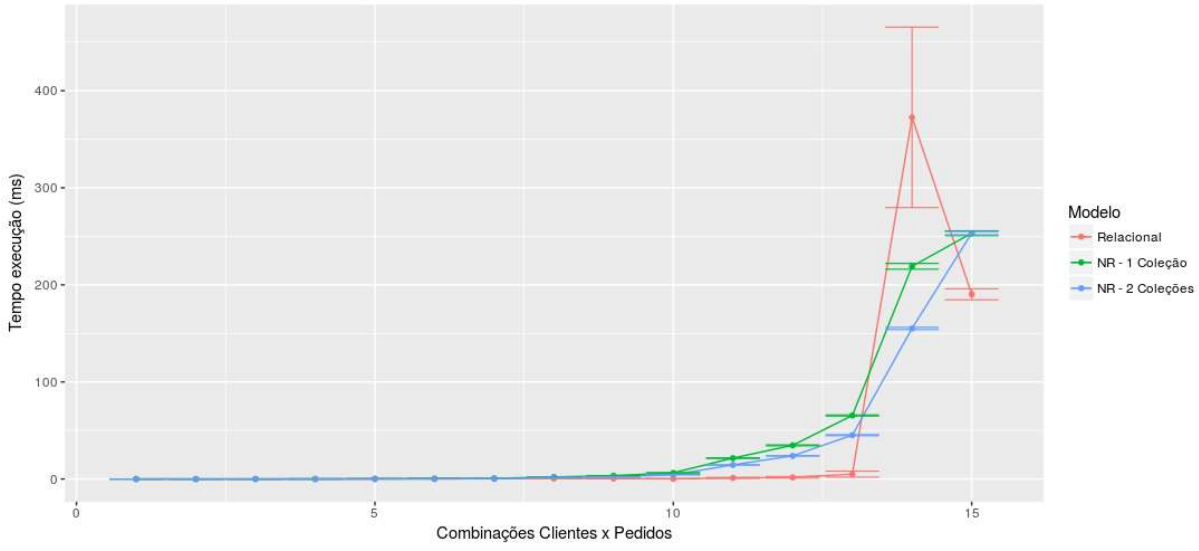
A figura nos permite verificar que a modelagem relacional tem uma distribuição de tempos de performance, em sua maioria, melhor que as demais, considerando-se os 13 primeiros tamanhos de bases de dados. Como na consulta anterior, o resultado parece evidenciar uma degradação da performance da consulta na modelagem relacional para os últimos tamanhos de base de dados, ou seja, uma pior escalabilidade.

Estatisticamente, em função da não normalidade da distribuição dos tempos médios para a consulta, foi feito o Teste não paramétrico de Friedman, que apontou diferenças nos resultados de tempo médio de consulta, considerando as 3 modelagens e os 15 tamanhos de bases de dados (valor p = 0.02237). Foram feitos então testes par a par não paramétricos de Wilcoxon para analisar as performances entre as 3 modelagens.

Os testes de Wilcoxon demonstraram uma superioridade da performance da modelagem não relacional de 2 coleções em relação à modelagem não relacional de 1 coleção; e 1 equivalência da performance da modelagem relacional em relação à performance das 2 modelagens não relacionais (em função da sua alta variação de tempos de execução, desde tempos de execução pequenos para menores tamanhos de BDs até altos tempos de execução para maiores tamanhos de BDs).

A figura 15 ilustra a evolução dos tempos médios de execução da consulta que traz informações sumarizadas sobre os produtos mais vendidos.

Figura 15 – Consulta de Produtos Mais Vendidos



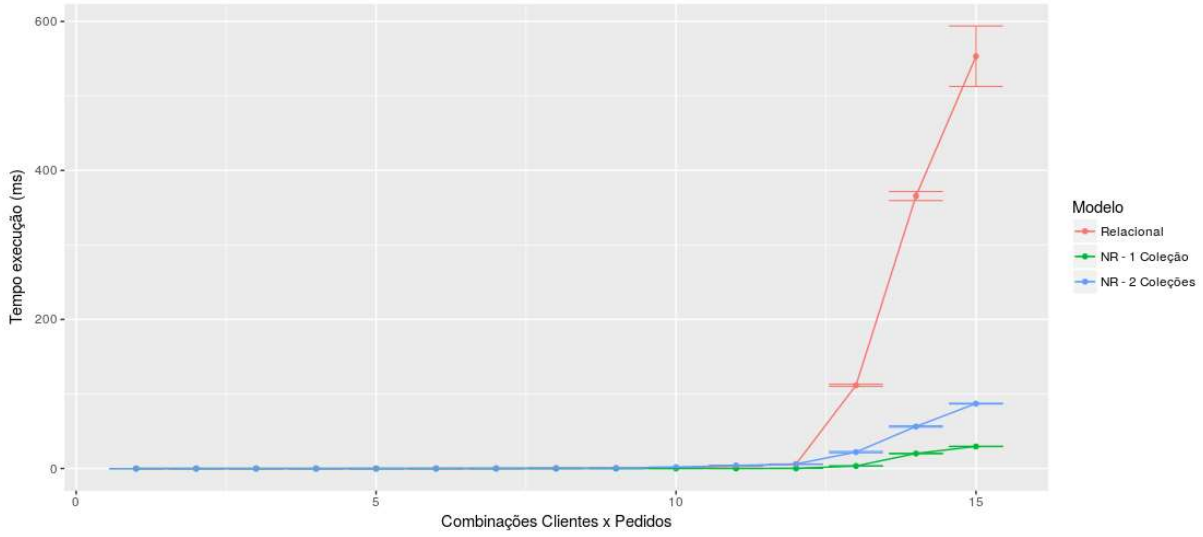
Fonte: Próprio autor.

Observando o gráfico, sugere-se um comportamento superior do modelo relacional até a primeira base de dados com 1 milhão de clientes. À partir daí, têm-se um comportamento de degeneração da performance (embora o último resultado apresente uma queda no tempo de execução). Novamente sugere-se que os últimos tamanhos de bancos de dados representam um limiar no comportamento robusto do modelo relacional.

Tendo as distribuições dos tempos médios de execução apresentado distribuições não normais, efetuou-se o teste não paramétrico de Friedman, a fim de verificar diferenças estatísticas entre as performances para as 3 modelagens ao longo dos 15 tamanhos de BDs. O resultado do teste evidenciou diferenças entre as performances das modelagens (p = 2.847e-05), e os testes de Wilcoxon par a par evidenciaram uma melhor performance do modelo relacional, seguido do modelo não relacional de 2 coleções e do modelo não relacional de 1 coleção (um pouco pior); apesar do comportamento mais instável do modelo relacional para os últimos 2 BDs.

Partindo para a próxima consulta, a figura 16 apresenta os tempos médios de execução obtidos nas 3 modelagens para a consulta que retorna os clientes mais compradores. Novamente observa-se a degeneração do desempenho da modelagem relacional para as últimas bases de dados. O gráfico sugere um melhor desempenho da modelagem não relacional de 1 coleção, seguido da não relacional de 2 coleções e da modelagem relacional.

Figura 16 – Consulta - Maiores Compradores

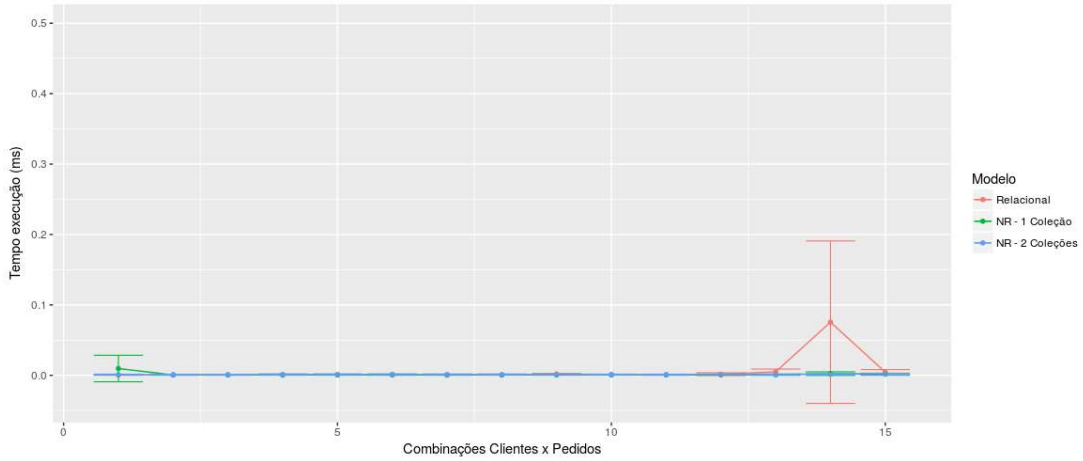


Fonte: Próprio autor.

Estatisticamente, feito o teste não paramétrico de Friedman (em função da não normalidade dos dados) para evidenciar possíveis diferenças estatísticas entre as performances da consulta para as 3 modelagens consideradas e os 15 tamanhos de BDs. O teste evidenciou diferenças (p = 3.372e-06) entre as performances das modelagens. Os testes par a par de Wilcoxon evidenciaram uma melhor performance para a modelagem não relacional de 1 coleção e uma equivalência estatística entre as abordagens não relacionais de 2 coleções e a abordagem relacional; apesar de a figura 16 apontar uma aparente pior escalabilidade do modelo relacional.

A consulta seguinte a ser analisada realiza uma típica alteração de dados, alterando dados de endereço de clientes escolhidos aleatoriamente. A figura 17 apresenta a evolução dos tempos médios de execução desta operação, para as 3 modelagens e os 15 tamanhos de bases de dados.

Figura 17 – Alteração de Dados dos Clientes

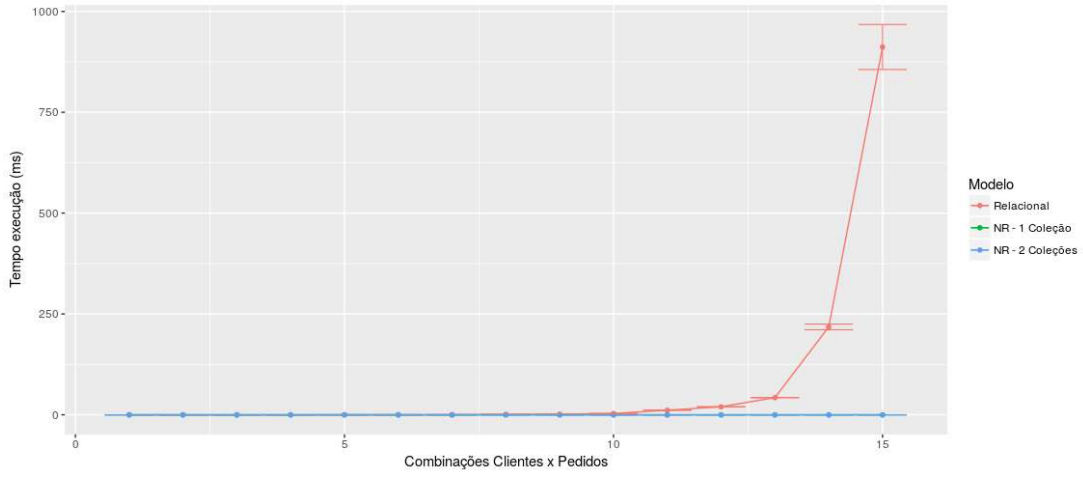


Fonte: Próprio autor.

O gráfico da figura 17 sugere uma performance bem similar das 3 modelagens, embora haja uma degeneração do desempenho do modelo relacional para o penúltimo tamanho do BD (o que, em primeira análise, não deveria ocorrer). O teste de Friedman confirmou a não diferença entre as performances das modelagens (p = 5,7e-02), a despeito do comportamento inesperado do modelo relacional para uma base de dados.

Por fim, a figura 18 apresenta a evolução dos tempos de execução médios de remoção de clientes escolhidos aleatoriamente para as 3 modelagens de dados.

Figura 18 – Remoção de Clientes



Fonte: Próprio autor.

O gráfico sugere uma performance pior da modelagem relacional em relação às modelagens não relacionais. Inclusive, devido à escala dos resultados da modelagem relacional ser muito maior do que os resultados das modelagens não relacionais, as linhas das modelagens não relacionais aparecem sobrepostas.

Estatisticamente, o teste de Friedman evidenciou diferenças entre as performances das modelagens (p = 5.748e-06), os testes pareados de Wilcoxon mostraram que a modelagem relacional apresentou pior desempenho, e as modelagens não relacionais foram equivalentes. Vale ressaltar aqui a grande diferença da variação dos tempos de execução do procedimento de remoção, considerando as modelagens relacional e não relacionais. Os tempos médios mínimos e máximos para o modelo relacional foram: 8.29e-03 e 9.11 e+02; enquanto que para os modelos não relacionais foram: 6 e-04 e 8e-03 (uma coleção) e 5 e-04 e 5e-03 (duas coleções).

Feitas as observações sobre as operações executadas, outro aspecto importante a se destacar para a avaliação das bases propostas são os seus tamanho. O espaço de armazenamento ocupado por uma base de dados é uma característica influente para avaliação de sua aplicabilidade, logo, os tamanhos das bases utilizadas é descrito na tabela 1 a seguir:

Tabela 1 – Tamanho das Diferentes Modelagens de Bases de Dados

Num. Clientes x Num.Pedidos	Mongo - Uma Coleção	Mongo - Duas Coleções	MariaDB
1.000.000 x 50	22.7GB	16.5GB + 93.6KB	9,56 GB
1.000.000 x 30	13.7GB	9.9GB + 93.6KB	5,78 GB
1.000.000 x 10	4.7GB	3.4GB + 93.6KB	2,15 GB
100.000 x 50	2.3GB	1.6GB + 93.6KB	946.39062500 MB
100.000 x 30	1.4GB	1015MB + 93.6KB	598.28125000 MB
100.000 x 10	480.6MB	353.2MB + 93.6KB	228.90625000 MB
10.000 x 50	232MB	167.7MB + 93.6KB	105.79687500 MB
10.000 x 30	140MB	101.5MB + 93.6KB	66.73437500 MB
10.000 x 10	48MB	35.2MB + 93.6KB	26.17187500 MB
1.000 x 50	23.2MB	16.8MB + 93.6KB	14.34375000 MB
1.000 x 30	14MB	10.1MB + 93.6KB	10.42187500 MB
1.000 x 10	4.8MB	3.5MB + 93.6KB	6.35937500 MB
100 x 50	2.3MB	1.7MB + 93.6KB	4.81250000 MB
100 x 30	1.4MB	1MB + 93.6KB	3.70312500 MB
100 x 10	490.9KB	359.4KB + 93.6KB	3.32812500 MB

Fonte: Próprio Autor

Pela observação da tabela, podemos concluir que, para as combinações com menos clientes e pedidos, as modelagens não-relacionais tem tamanho menor (até a base 1000 clientes x 10 produtos). Para as combinações com mais clientes e pedidos (1000 clientes x 30 pedidos, ou mais), a modelagem não-relacional de uma coleção é a maior, seguida pela não-relacional de duas coleções, e por último a base relacional. Segundo o que afirma Hows, Plugge e Membrey (2015), o armazenamento conjunto dos dados não relacionais no formato BSON, apesar de garantir a flexibilidade e o acesso rápido, ocupa mais espaço em relação ao armazenamento relacional, o que também é destacado por Raza *et al.* (2019).

4.3 Discussão de Resultados

Fowler e Sadalage (2013) afirmam que, segundo os defensores da persistência poliglota de dados, diferentes SGBDs foram projetados para resolução de diferentes problemas, e

a utilização de uma mesma abordagem ou modelagem de dados para todas as necessidades de um sistema pode resultar em soluções de baixo desempenho. Neste sentido, pode-se pensar na utilização de diferentes mecanismos de armazenamento de dados para diferentes necessidades, tais como: transações de negócio, relatórios gerenciais, BI (Business Intelligence), dados de gerenciamento de sessão de usuário, informações de logs de usuários, etc.

Assim, é importante salientar que, no contexto deste trabalho, foram analisadas as informações de performance das operações e a aplicação de cada uma no contexto do negócio considerado (*e-commerce*), para realizar uma discussão sobre a existência de uma modelagem dentre as apresentadas que seria a mais adequada para implementação de cada operação.

Primeiramente, de forma geral parece haver uma maior estabilidade, em termos de performance, das modelagens não relacionais em relação à modelagem relacional, que mantêm uma tendência de degeneração de sua performance para as bases de dados maiores. Isto posto, deve-se realizar uma discussão sobre o cenário de utilização de cada operação para a discussão sobre a adequação de utilização de uma ou outra modelagem.

Com relação à utilização de operações no contexto de um sistema de *e-commerce*, pode-se pensar na execução rotineira de várias operações de inserção, cada uma sendo executada para implementar uma transação do negócio, como: inserção de um novo produto para ser colocado à venda, inserção de um novo cliente, inserção de um pedido de cliente, etc.

Neste contexto de utilização, não faz sentido pensar em agregar os dados relativos à clientes, pedidos e produtos conjuntamente, como num agregado hierárquico de um esquema não relacional. Primeiramente, porque tais inserções vão ocorrer em momentos diferentes, em segundo lugar, devido ao fato de que há uma existência independente de entidades de dados (por exemplo, pode ocorrer a inserção de um novo produto inicialmente não associado à nenhum cliente ou pedido). Portanto, as modelagens não relacionais não são adequadas, pois como elas modelam os dados através de uma estrutura de informações "agregadas" umas às outras, no modelo não relacional de uma coleção deve-se sempre inserir documentos com informações sobre clientes, produtos, tipos de produtos e pedidos; e no de duas coleções inserir informações sobre clientes e pedidos numa coleção e produtos e seus tipos em outra. Então, porque considerar a operação de inserção massiva nesta discussão?

Pensando-se na utilização de operações de inserção em massa (a primeira consulta considerada neste trabalho), esta poderia ser utilizada para, periodicamente, incluir informações trazidas de uma base de dados das transações do *e-commerce* para uma outra base de dados, de uso gerencial, utilizada especificamente para visualização de informações, como num esquema de *Data Warehouse*, e que poderia ser implementada tanto num esquema relacional quanto num não relacional. Tal base seria menos sensível à problemas decorrentes de quebra de durabilidade e consistência dos dados, já que as transações não seriam perdidas (FOWLER; SADALAGE, 2013), pois estariam já gravadas na base de dados de origem. Para esta consulta, as três modelagens se mostram bem competitivas, sendo que para bases de dados maiores uma modelagem

não relacional parece mais adequada, com a modelagem com duas coleções apresentando menos redundância de informações.

Com relação às operações de visualização de dados históricos ou sumarizados que exigem busca por grande quantidade de informação (consultas 'b','c' e 'd'), estas poderiam implementar diferentes funcionalidades da base de dados gerencial citada anteriormente, e por se tratarem de operações de visualização, não incorreriam em problemas relacionados à durabilidade e consistência dos dados. Para as funcionalidades implementadas por operações deste tipo, as 3 modelagens poderiam ser utilizadas, com preferência para as modelagens não relacionais, em função do desempenho mais estável (com destaque para a consulta a dados dos clientes mais compradores).

Com relação às operações de alteração de dados, esta poderia ser utilizada para funcionalidades corriqueiras de um sistema de *e-commerce* para alteração de dados de produtos (preço, por exemplo), dados de clientes, etc. Nota-se que, no que se refere a desempenho, as três modelagens têm performance similar. Como funcionalidades que pressupõem alteração de dados num sistema de *e-commerce* possivelmente vão requerer um maior cuidado com relação à durabilidade e consistência dos dados, recomenda-se a utilização do modelo relacional, já que os modelos relacionais tem melhores mecanismos de garantia de integridade e consistência, o que nos bancos não relacionais só é garantido a nível de 1 documento(FOWLER; SADALAGE, 2013). Neste caso, uma alteração que envolvesse mais de um documento de uma coleção (alteração de preços de vários produtos, por exemplo), não garantiria ao seu final que a operação iria ser feita completamente sem erros para a base de dados.

Por fim, operações de remoção num sistema de *e-commerce* podem implementar funcionalidades de cancelamento de pedidos, de cadastro de clientes, etc. Tais operações são sensíveis a erros de durabilidade e consistência dos dados, e ainda que a modelagem relacional tenha um desempenho visivelmente pior, devem ser preferencialmente implementadas num modelo relacional. Similarmente ao caso da operação de inserção, como as modelagens não relacionais modelam as informações como estruturas agregadas, a remoção teria automaticamente um efeito cascata (remoção de um cliente removeria informações dos produtos que este comprou, por exemplo), o que pode não ser adequado para o contexto do negócio. Os resultados piores para a inserção e remoção no modelo relacional eram esperados, devido ao fato de que, como as informações na modelagem relacional estão estruturadas em várias unidades estruturadas de armazenamento (tabelas), a inserção e remoção de dados se dá por partes, ao passo que nos modelos não relacionais isso se dá de forma menos fragmentada.

5 CONCLUSÕES E CONSIDERAÇÕES FINAIS

O presente trabalho apresentou uma discussão sobre possibilidades de implementação de funcionalidades típicas de bancos de dados para um sistema de *e-commerce* sob a abordagem da persistência poliglota de dados. Observando critério quantitativo (tempo de execução das operações) e qualitativos (garantia de consistência e durabilidade dos dados), procurou-se indicar que tipos de operações seriam mais adequadas a serem implementadas por uma ou outra modelagem.

No geral, observa-se um comportamento mais estável no que se refere a desempenho das operações para as modelagens não relacionais, o que sugere que operações de consulta massiva de dados (tipicamente utilizadas como funcionalidades de relatórios gerenciais ou de BI), por também exigirem menor rigor no que se refere à garantia de consistência e durabilidade (característica dos SGBDs não relacionais), devam ser preferencialmente implementadas em esquemas não relacionais.

Por outro lado, operações que possam implicar em maiores problemas caso a consistência e durabilidade não seja garantida (como operações de registro, alteração ou remoção de informações do dia a dia do negócio), são mais adequadamente implementadas na modelagem relacional, também em função da estrutura dos dados ser mais adequada (a despeito da performance mais instável da modelagem relacional).

Um aspecto relativo à adoção de persistência poliglota que não foi abordado aqui e que é de extrema importância é o aspecto relacionado à gestão de projetos. É coerente que a decisão sobre a adoção da persistência poliglota passe por uma análise envolvendo questões de garantia de integridade dos dados e performance, mas também sobre os custos adicionais (financeiro, e de tempo) para que uma equipe de desenvolvimento possa desenvolver um Sistema de Informação que se comunique com diferentes tipos de SGBDs.

Um segundo aspecto não abordados, que também pertence ao escopo de treinamento (gestão de projetos), é a diferença nas sintaxes utilizadas para consulta nos diferentes BDs. Foi considerado que o MariaDB tem uma linguagem de consulta mais intuitiva do que o MongoDB. Para consultas mais simples, o MariaDB utiliza um texto maior e o Mongodb, um texto bastante simples. Como exemplo,abaixo temos dois trechos do código PHP para consultas utilizadas neste trabalho, ambas desempenhando a mesma operação de consulta a clientes.

Consulta de Clientes - MariaDB:

```
1 $query1 = "select c.*, e.email, t.telefone from cliente c join
      email_cliente e on c.cod_cliente=e.cod_cliente join telefone_cliente t
2           on c.cod_cliente=t.cod_cliente where c.cod_cliente='$cliente'";
3 $selecttable = mysqli_query($connection,$query1) or die(mysql_error());
```

Consulta de Clientes - MongoDB:

```
1 $query = array('Codigocliente' => $cliente);
2 $selectdocument = $collection->find($query);
```

Porém, para consultas mais complexas, ambas construções têm um trecho de código ”grande”, mas na base de dados relacional, o texto é mais fácil de ser compreendido. Como exemplo, temos os trechos de código abaixo que realizam a mesma operação de consulta dos maiores compradores da base.

Consulta de Maiores Compradores - MariaDB

```
$query = "select c.cod_cliente, nome, sum(p.total) as 'Soma' from cliente c
    join pedido p on c.cod_cliente=p.cod_cliente group by c.cod_cliente,
   nome order by 'Soma' desc limit ".$top;
$selecttable = mysqli_query($connection,$query) or die(mysql_error());
```

Consulta de Maiores Compradores - MongoDB

```
$query = array(array('$unwind' => '$Pedidos'),
               array('$group' => array('_id' => array('Codigo' => '
   $Codigocliente', 'Nome' => '$Nome'),
                                'sum' => array('$sum' => '$Pedidos.Total'))),
                array('$sort' => array('sum' => -1)),
                array('$limit' => $top),
                array('$project' => array('_id' => 1, 'sum' => 1)));
```

Uma última consideração a ser feita, antes das sugestões para os trabalho futuros é sobre a mudança das propriedades das bases de dados com o objetivo de melhorar suas características fracas. Podemos pensar, para o modelo relacional, em desativar as chaves estrangeiras que garantem a integridade referencial, e assim objetivar o ganho de desempenho no lugar de consistência. Para a base não relacional, podemos utilizar um campo que referencia um documento em outra coleção, objetivando o ganho de consistência entre diferentes coleções de dados no lugar de performance. Nenhuma destas técnicas foi aplicada no trabalho atual, visto que o trabalho se restringiu a avaliar quantitativa e qualitativamente os diferentes BDs com suas propriedades fundamentais.

Ao final do trabalho, foi possível observar que, através da experimentação e sua respectiva análise estatística, pôde-se encontrar melhores cenários de uso para as diferentes modelagens de bancos de dados propostas.

5.1 Sugestões Para o Futuro

A abordagem da avaliação de utilização de modelagens alternativas de dados apresentada aqui teve um escopo definido por alguns aspectos:

a) No que se refere à aplicação, foi escolhida uma aplicação de *e-commerce*;

b) No que se refere a SGBDs, foi escolhido MariaDB e Mongo;

c) No que se refere à arquitetura computacional, foi escolhido uma arquitetura não distribuída.

Pensando no que defende o Teorema CAP, sugere-se um trabalho futuro no qual possa ser feito um processo de particionamento em cluster do processamento dos dados de cada BD, para que assim possa ser avaliado também a propriedade de tolerância a particionamento.

Além disso, aplicações de outros contextos, com funcionalidades de outras naturezas poderiam ser analisadas, e também outras tecnologias de SGBDs relacionais e não relacionais poderiam ser experimentadas.

Como sugestão futura para os métodos de análise, também poderiam ser avaliados os consumos de memória principal e processamento durante os testes, visto que este são fatores cruciais na escolha de servidores de aplicações distribuídas. Também poderia ser analisada a relevância, para a performance das consultas, dos fatores número de clientes, número de pedidos por clientes, assim como também a interação entre eles, através do uso de métodos de análise de regressão.

REFERÊNCIAS

ABDULLAH, A.; ZHUGE, Q. ***From relational databases to nosql databases: performance evaluation.*** Research Journal of Applied Sciences, Engineering and Technology, 11(4): 434-439, May, 2015.

BRITO, R. W. **Bancos de Dados NoSQL x SGBDs Relacionais:Análise Comparativa.** Universidade de Fortaleza, p. 1-6. Disponível em https://tinyurl.com/66sz4ku8.

CLARO, C. A. M. **PERSISTÊNCIA POLIGLOTA EM BANCOS DE DADOS E FERRAMENTAS DE DESENVOLVIMENTO DE APLICATIVOS**. Monografia de Especialização. Universidade Tecnológica Federal do Paraná, 2017. Disponível em http://repositorio.roca.utfpr.edu.br/jspui/bitstream/1/12790/1/CT_TECSOL_II_2017_01.pdf .

DATE, C. J. **Introdução** a Sistemas de Banco **de Dado.** 1. ed. [S.1.]: GEN LTC, 2004. 896 p.

ELMASRI, R.; NAVATHE, S. B. Sistema de Banco de Dados. 1. ed. [S.1.]: Pearson Universidades, 2019. 1152 p.

FERNANDES, A. de S.; SILVA, P. C. L. **Comparativo Técnico de Tecnologias de Banco de Dados: Relacional, NoSql e NewSQL**. Instituto de Informática - Instituto Federal do Norte de Minas Gerais (IFNMG), p. 13, 2019. Disponível em: https://www.academia.edu/6559619/Comparativo_T%C3%A9cnico_entre_tecnologias_de_banco_de_dados_relacional_NoSQL_e_NewSQL.

FOWLER, M.; SADALAGE, P. J. **NoSqI Essencial - Um Guia Conciso Para o Mundo Emergente da Persistência Poliglota**. 1. ed. [S.1.]: Novatec, 2013. 216 p.

GALICZ, E.; SCHMID, S.; REINHARDT, W. **P*erformance investigation ofselected sql and nosql databases.*** 18th AGILE International Conference on Geographic Information Science. Lisbon, June, 9-12, 2015.

GEROSA, M. A.; DIANA, M. D. **NOSQL na Web 2.0: Um Estudo Comparativo de BancosNão-ReIacionais para Armazenamento de Dados na Web 2.0**. Universidade de São Paulo, 2010. Disponível em: https://www.ime.usp.br/~mdediana/nosql_wtdbd10.pdf.

HAIGH, T. C***harles w. bachman: database software pioneer.*** IEEE Annals of the History of Computing, 70-80, 2011.

HOWS, D.; PLUGGE, E.; MEMBREY, P. **Introdução ao MongoDB.** 1. ed. [S.l.]: Novatec, 2015. 168 p.

JARGAS, A. M. Shell Script Profissional. 1. ed. [S.1.]: Novatec, 2008. 480 p.

JOSE, B.; ABRAHAM, S. P***erformance analysis of nosql and relational databases with mongodb and mysql.*** International Multi-conference on computing, Communications, Electrical & Nanotechnology, 2036-2043, 2020.

KHINE, P. P.; WANG, Z. A ***review of polyglot persistence in the big data world.*** 1-24, 2019.

MANOHARAN, S.; LI, Y. A ***performance comparison ofsql and nosql databases.*** Conference: Communications, Computers and Signal Processing (PACRIM), 15-19, 2013.

MAOWA, J.; RAHMAN, M. 0.; MUSTAFA, R.; HOQUE, S. A ***comparative study on big data handling using relational and non-relational data model.*** International Journal of Data Mining & Knowledge Management Process (IJDKP), 7(3): 25-32, May, 2017.

NAYAK, A.; PORIYA, A.; POOJARY, D. **T*ype of nosql databases and its comparison with***

relational databases. International Journal of Applied Information Systems(IJAIS), 5(4):16-19, March, 2013.

OLIVEIRA, F. R. **Avaliação de desempenho de gerenciadores de bancos de dados multi-modelo em aplicações com persistência poliglota**. Dissertação de Mestrado. Faculdade Campo Limpo Paulista, 2017. Disponível em https://www.cc.faccamp.br/Dissertacoes/FabioRobertoOliveira.pdf.

OLIVEIRA, M. N. **Estudo Comparativo Entre Sistemas de Banco de Dados NoSQL e Relacional**. Trabalho de Conclusão de Curso. Universidade Federal de Pernambuco, 2019.

RAUFI, B.; ISMAILI, F.; AJDARI, J.; SAHATQIJA, K.; ZENUNI, X. C***omparison between relational and nosql databases.*** 41st International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO), 216-220, May, 2018.

RAZA, A.; MAJEED, M. A.; SHAFIQUE, M. U.; ALI, W. C***omparison between sql and nosql databases and their relationship with big data analytics.*** Asian Journal of Research in Computer Science, 4(2):1-10, 2019.

SILBERSCHATZ, A.; KORTH, H. F.; SUDARSHAN, S. Sistema de Banco de Dados. 7. ed. [S.1.]: GEN LTC, 2020. 754 p.

THALHEIM, B. **E*xtended entity-relationship model.*** Christian-Albrechts University Kie1,2009. Disponível em https://nanopdf.com/download/extended-entity-relationship-model_pdf

ANEXO A – ALGORITMO 1 - CRIAÇÃO DA BASE DE DADOS RELACIONAL

```
CREATE DATABASE  IF NOT EXISTS `ecommerce` /*!40100 DEFAULT CHARACTER SET
    latin1 */;
USE `ecommerce`;
-- MySQL dump 10.13  Distrib 5.5.60, for debian-linux-gnu (i686)
--
-- Host: 127.0.0.1    Database: ecommerce
-- ------------------------------------------------------
-- Server version 5.5.60-0ubuntu0.14.04.1

/*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
/*!40101 SET @OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS */;
/*!40101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION */;
/*!40101 SET NAMES utf8 */;
/*!40103 SET @OLD_TIME_ZONE=@@TIME_ZONE */;
/*!40103 SET TIME_ZONE='+00:00' */;
/*!40014 SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0 */;
/*!40014 SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS,
    FOREIGN_KEY_CHECKS=0 */;
/*!40101 SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='NO_AUTO_VALUE_ON_ZERO' */;
/*!40111 SET @OLD_SQL_NOTES=@@SQL_NOTES, SQL_NOTES=0 */;

--
-- Table structure for table `cliente`
--
#CRIAÇÃO DAS TABELAS

DROP TABLE IF EXISTS `cliente`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `cliente` (
  `cod_cliente` int(11) NOT NULL AUTO_INCREMENT,
  `nome` varchar(70) NOT NULL,
  `endereco` varchar(90) NOT NULL,
  PRIMARY KEY (`cod_cliente`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Dumping data for table `cliente`
--

LOCK TABLES `cliente` WRITE;
/*!40000 ALTER TABLE `cliente` DISABLE KEYS */;
/*!40000 ALTER TABLE `cliente` ENABLE KEYS */;
UNLOCK TABLES;

```

```
45 --
46 -- Table structure for table `email_cliente`
47 --
48
49 DROP TABLE IF EXISTS `email_cliente`;
50 /*!40101 SET @saved_cs_client     = @@character_set_client */;
51 /*!40101 SET character_set_client = utf8 */;
52 CREATE TABLE `email_cliente` (
53   `cod_cliente` int(11) NOT NULL,
54   `email` varchar(30) NOT NULL,
55   PRIMARY KEY (`cod_cliente`,`email`),
56   CONSTRAINT `email_cliente_ibfk_1` FOREIGN KEY (`cod_cliente`) REFERENCES
     `cliente` (`cod_cliente`)
57 ) ENGINE=InnoDB DEFAULT CHARSET=latin1;
58 /*!40101 SET character_set_client = @saved_cs_client */;
59
60 --
61 -- Dumping data for table `email_cliente`
62 --
63
64 LOCK TABLES `email_cliente` WRITE;
65 /*!40000 ALTER TABLE `email_cliente` DISABLE KEYS */;
66 /*!40000 ALTER TABLE `email_cliente` ENABLE KEYS */;
67 UNLOCK TABLES;
68
69 --
70 -- Table structure for table `item_pedido`
71 --
72
73 DROP TABLE IF EXISTS `item_pedido`;
74 /*!40101 SET @saved_cs_client     = @@character_set_client */;
75 /*!40101 SET character_set_client = utf8 */;
76 CREATE TABLE `item_pedido` (
77   `cod_item` int(11) NOT NULL,
78   `cod_pedido` int(11) NOT NULL,
79   `cod_produto` int(11) NOT NULL,
80   `preco_unit` float DEFAULT NULL,
81   `quantidade` int(11) DEFAULT NULL,
82   `total` float DEFAULT NULL,
83   PRIMARY KEY (`cod_item`,`cod_pedido`),
84   KEY `cod_pedido` (`cod_pedido`),
85   KEY `cod_produto` (`cod_produto`),
86   CONSTRAINT `item_pedido_ibfk_1` FOREIGN KEY (`cod_pedido`) REFERENCES `
     pedido` (`cod_pedido`),
87   CONSTRAINT `item_pedido_ibfk_2` FOREIGN KEY (`cod_produto`) REFERENCES `
     produto` (`cod_produto`)
88 ) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

```
89 /*!40101 SET character_set_client = @saved_cs_client */;
90
91 --
92 -- Dumping data for table `item_pedido`
93 --
94
95 LOCK TABLES `item_pedido` WRITE;
96 /*!40000 ALTER TABLE `item_pedido` DISABLE KEYS */;
97 /*!40000 ALTER TABLE `item_pedido` ENABLE KEYS */;
98 UNLOCK TABLES;
99
100 --
101 -- Table structure for table `pedido`
102 --
103
104 DROP TABLE IF EXISTS `pedido`;
105 /*!40101 SET @saved_cs_client     = @@character_set_client */;
106 /*!40101 SET character_set_client = utf8 */;
107 CREATE TABLE `pedido` (
108   `cod_pedido` int(11) NOT NULL AUTO_INCREMENT,
109   `cod_cliente` int(11) NOT NULL,
110   `data_pedido` date DEFAULT NULL,
111   `total` float DEFAULT NULL,
112   PRIMARY KEY (`cod_pedido`),
113   KEY `cod_cliente` (`cod_cliente`),
114   CONSTRAINT `pedido_ibfk_1` FOREIGN KEY (`cod_cliente`) REFERENCES `
      cliente` (`cod_cliente`)
115 ) ENGINE=InnoDB DEFAULT CHARSET=latin1;
116 /*!40101 SET character_set_client = @saved_cs_client */;
117
118 --
119 -- Dumping data for table `pedido`
120 --
121
122 LOCK TABLES `pedido` WRITE;
123 /*!40000 ALTER TABLE `pedido` DISABLE KEYS */;
124 /*!40000 ALTER TABLE `pedido` ENABLE KEYS */;
125 UNLOCK TABLES;
126
127 --
128 -- Table structure for table `produto`
129 --
130
131 DROP TABLE IF EXISTS `produto`;
132 /*!40101 SET @saved_cs_client     = @@character_set_client */;
133 /*!40101 SET character_set_client = utf8 */;
134 CREATE TABLE `produto` (
```

```
135   `cod_produto` int(11) NOT NULL AUTO_INCREMENT,
136   `nome` varchar(30) NOT NULL,
137   `pc_unit` float DEFAULT NULL,
138   `quantidade` float DEFAULT NULL,
139   `tipo` int(11) NOT NULL,
140   PRIMARY KEY (`cod_produto`),
141   KEY `tipo` (`tipo`),
142   CONSTRAINT `produto_ibfk_1` FOREIGN KEY (`tipo`) REFERENCES `tipo_produto
      ` (`tipo`)
143 ) ENGINE=InnoDB DEFAULT CHARSET=latin1;
144 /*!40101 SET character_set_client = @saved_cs_client */;
145
146 --
147 -- Dumping data for table `produto`
148 --
149
150 LOCK TABLES `produto` WRITE;
151 /*!40000 ALTER TABLE `produto` DISABLE KEYS */;
152 /*!40000 ALTER TABLE `produto` ENABLE KEYS */;
153 UNLOCK TABLES;
154
155 --
156 -- Table structure for table `telefone_cliente`
157 --
158
159 DROP TABLE IF EXISTS `telefone_cliente`;
160 /*!40101 SET @saved_cs_client     = @@character_set_client */;
161 /*!40101 SET character_set_client = utf8 */;
162 CREATE TABLE `telefone_cliente` (
163   `cod_cliente` int(11) NOT NULL,
164   `telefone` int(11) NOT NULL,
165   PRIMARY KEY (`cod_cliente`,`telefone`),
166   CONSTRAINT `telefone_cliente_ibfk_1` FOREIGN KEY (`cod_cliente`)
      REFERENCES `cliente` (`cod_cliente`)
167 ) ENGINE=InnoDB DEFAULT CHARSET=latin1;
168 /*!40101 SET character_set_client = @saved_cs_client */;
169
170 --
171 -- Dumping data for table `telefone_cliente`
172 --
173
174 LOCK TABLES `telefone_cliente` WRITE;
175 /*!40000 ALTER TABLE `telefone_cliente` DISABLE KEYS */;
176 /*!40000 ALTER TABLE `telefone_cliente` ENABLE KEYS */;
177 UNLOCK TABLES;
178
179 --
```

```
-- Table structure for table `tipo_produto`
--

DROP TABLE IF EXISTS `tipo_produto`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `tipo_produto` (
  `tipo` int(11) NOT NULL AUTO_INCREMENT,
  `descricao` varchar(30) NOT NULL,
  PRIMARY KEY (`tipo`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Dumping data for table `tipo_produto`
--

LOCK TABLES `tipo_produto` WRITE;
/*!40000 ALTER TABLE `tipo_produto` DISABLE KEYS */;
/*!40000 ALTER TABLE `tipo_produto` ENABLE KEYS */;
UNLOCK TABLES;

--
-- Dumping routines for database 'ecommerce'
--

#CRIAÇÃO DOS STORED PROCEDURES

/*!50003 DROP PROCEDURE IF EXISTS `sp_insereclientes` */;
/*!50003 SET @saved_cs_client      = @@character_set_client */ ;
/*!50003 SET @saved_cs_results     = @@character_set_results */ ;
/*!50003 SET @saved_col_connection = @@collation_connection */ ;
/*!50003 SET character_set_client  = utf8 */ ;
/*!50003 SET character_set_results = utf8 */ ;
/*!50003 SET collation_connection  = utf8_general_ci */ ;
/*!50003 SET @saved_sql_mode       = @@sql_mode */ ;
/*!50003 SET sql_mode              = '' */ ;
DELIMITER $$
CREATE DEFINER=`root`@`localhost` PROCEDURE `sp_insereclientes`(nclientes
    integer)
begin
    declare ncli, nfone, nemail, auxcli, auxfone, auxemail int default 0;

  start transaction;

    while (ncli<nclientes) do

```

```
226         set ncli = ncli + 1;
227      insert into cliente(nome, endereco) values(concat('cliente ', ncli),
    concat('endereco ', ncli));
228         select cod_cliente into auxcli from cliente order by cod_cliente
    desc limit 1;
229 
230         #select truncate((rand()*3),0)+1 into nfone;  --sorteia um nro
    entre 1 e 3
231         set nfone = 3; -- todo cliente terá 3 telefones
232 
233         set auxfone = 1;
234      while (auxfone<=nfone) do
235         insert into telefone_cliente(cod_cliente, telefone)
236                     values(auxcli,  (select truncate(rand()*99999999,0)));
237         set auxfone = auxfone+1;
238      end while;
239 
240         #select truncate((rand()*3),0)+1 into nemail;  -- sorte nro. entre
    1 e 3
241         set nemail = 2; -- todo cliente terá 2 emails
242 
243         set auxemail = 1;
244         while (auxemail<=nemail) do
245             insert into email_cliente(cod_cliente, email)
246           values(auxcli, concat('email_cli',auxcli,'_',auxemail,'@provedor.
    com.br'));
247        set auxemail = auxemail+1;
248      end while;
249 
250   end while;
251 
252      commit;
253 
254 end$$
255 DELIMITER ;
256 /*!50003 SET sql_mode              = @saved_sql_mode */ ;
257 /*!50003 SET character_set_client  = @saved_cs_client */ ;
258 /*!50003 SET character_set_results = @saved_cs_results */ ;
259 /*!50003 SET collation_connection  = @saved_col_connection */ ;
260 /*!50003 DROP PROCEDURE IF EXISTS `sp_inserepedidos` */;
261 /*!50003 SET @saved_cs_client      = @@character_set_client */ ;
262 /*!50003 SET @saved_cs_results     = @@character_set_results */ ;
263 /*!50003 SET @saved_col_connection = @@collation_connection */ ;
264 /*!50003 SET character_set_client  = utf8 */ ;
265 /*!50003 SET character_set_results = utf8 */ ;
266 /*!50003 SET collation_connection  = utf8_general_ci */ ;
267 /*!50003 SET @saved_sql_mode       = @@sql_mode */ ;
```

```
/*!50003 SET sql_mode              = '' */ ;
DELIMITER $$
CREATE DEFINER=`root`@`localhost` PROCEDURE `sp_inserepedidos`(
    quant_pedidos integer)
begin
      declare auxquant_pedidos, auxquant_produtos,
        auxquant_clientes, auxquant_itens integer default 0;
      declare pcunit, total_pedido float;
      declare quant, pedidos_item integer;

      select max(cod_cliente) into auxquant_clientes from cliente;
      select max(cod_produto) into auxquant_produtos from produto;


      while (auxquant_pedidos<quant_pedidos) do
            set auxquant_pedidos = auxquant_pedidos+1;

            insert into pedido(cod_cliente, data_pedido, total)
                        values(truncate(rand()*auxquant_clientes,0)+1, date
    (date(now())- truncate(rand()*10,0)),0);

            set auxquant_itens = 0;
            set total_pedido = 0;
            #set pedidos_item = truncate(rand()*quant_itens,0)+1; --sorteia
     nro. itens

      set pedidos_item = 5; -- todo pedido tera 5 itens

            while(auxquant_itens<pedidos_item) do
                  set auxquant_itens = auxquant_itens+1;
                  set pcunit = truncate((rand()*7),2);
                  set quant = truncate((rand()*7),0)+1;
                  insert into item_pedido(cod_item, cod_pedido, cod_produto,
     preco_unit,
                                          quantidade, total)
                  values(auxquant_itens, auxquant_pedidos, truncate(rand()*
    auxquant_produtos,0)+1,
                        pcunit, quant,pcunit*quant);
         set total_pedido = total_pedido + (pcunit*quant);

      end while;

      update pedido set total = total_pedido where cod_pedido =
    auxquant_pedidos;


      end while;
```

```
309
310 end$$
311 DELIMITER ;
312 /*!50003 SET sql_mode              = @saved_sql_mode */ ;
313 /*!50003 SET character_set_client  = @saved_cs_client */ ;
314 /*!50003 SET character_set_results = @saved_cs_results */ ;
315 /*!50003 SET collation_connection  = @saved_col_connection */ ;
316 /*!50003 DROP PROCEDURE IF EXISTS `sp_insereprodutos` */;
317 /*!50003 SET @saved_cs_client      = @@character_set_client */ ;
318 /*!50003 SET @saved_cs_results     = @@character_set_results */ ;
319 /*!50003 SET @saved_col_connection = @@collation_connection */ ;
320 /*!50003 SET character_set_client  = utf8 */ ;
321 /*!50003 SET character_set_results = utf8 */ ;
322 /*!50003 SET collation_connection  = utf8_general_ci */ ;
323 /*!50003 SET @saved_sql_mode       = @@sql_mode */ ;
324 /*!50003 SET sql_mode              = '' */ ;
325 DELIMITER ;;
326 CREATE DEFINER=`root`@`localhost` PROCEDURE `sp_insereprodutos`(
    quant_produtos integer)
327 begin
328
329   declare auxquant_tipos, auxquant_produtos integer default 0;
330
331   select max(tipo) into auxquant_tipos from tipo_produto;
332
333   while (auxquant_produtos<quant_produtos) do
334    set auxquant_produtos = auxquant_produtos + 1;
335
336      insert into produto(nome, pc_unit, quantidade, tipo)
337         values(concat('Produto ',auxquant_produtos), truncate((rand()*10)
    ,2),
338           truncate((rand()*7),0)+1, truncate((rand()*auxquant_tipos),0)+1);
339
340   end while;
341 end ;;
342 DELIMITER ;
343 /*!50003 SET sql_mode              = @saved_sql_mode */ ;
344 /*!50003 SET character_set_client  = @saved_cs_client */ ;
345 /*!50003 SET character_set_results = @saved_cs_results */ ;
346 /*!50003 SET collation_connection  = @saved_col_connection */ ;
347 /*!50003 DROP PROCEDURE IF EXISTS `sp_inseretipoprodutos` */;
348 /*!50003 SET @saved_cs_client      = @@character_set_client */ ;
349 /*!50003 SET @saved_cs_results     = @@character_set_results */ ;
350 /*!50003 SET @saved_col_connection = @@collation_connection */ ;
351 /*!50003 SET character_set_client  = utf8 */ ;
352 /*!50003 SET character_set_results = utf8 */ ;
353 /*!50003 SET collation_connection  = utf8_general_ci */ ;
```

```
354 /*!50003 SET @saved_sql_mode       = @@sql_mode */ ;
355 /*!50003 SET sql_mode              = '' */ ;
356 DELIMITER ;;
357 CREATE DEFINER=`root`@`localhost` PROCEDURE `sp_inseretipoprodutos`(
        quant_tipos integer)
358 begin
359   declare auxquant_tipos integer default 0;
360
361   while (auxquant_tipos<quant_tipos) do
362    set auxquant_tipos = auxquant_tipos + 1;
363
364      insert into tipo_produto(descricao) values(concat('Tipo_Produto ',
       auxquant_tipos));
365
366   end while;
367
368 end ;;
369 DELIMITER ;
370 /*!50003 SET sql_mode              = @saved_sql_mode */ ;
371 /*!50003 SET character_set_client  = @saved_cs_client */ ;
372 /*!50003 SET character_set_results = @saved_cs_results */ ;
373 /*!50003 SET collation_connection  = @saved_col_connection */ ;
374 /*!40103 SET TIME_ZONE=@OLD_TIME_ZONE */;
375
376 /*!40101 SET SQL_MODE=@OLD_SQL_MODE */;
377 /*!40014 SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS */;
378 /*!40014 SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS */;
379 /*!40101 SET CHARACTER_SET_CLIENT=@OLD_CHARACTER_SET_CLIENT */;
380 /*!40101 SET CHARACTER_SET_RESULTS=@OLD_CHARACTER_SET_RESULTS */;
381 /*!40101 SET COLLATION_CONNECTION=@OLD_COLLATION_CONNECTION */;
382 /*!40111 SET SQL_NOTES=@OLD_SQL_NOTES */;
383
384 -- Dump completed on 2019-04-18 19:35:27
385
386
387
388
389 DELIMITER $$
390 CREATE DEFINER=`root`@`localhost` PROCEDURE `sp_inseredadosecommerce`(
        nclientes integer, quant_produtos integer,
391 quant_tipos integer, quant_pedidos integer)
392 begin
393
394    declare dt1, dt2, dtdiff integer;
395
396    delete from email_cliente;
397    delete from telefone_cliente;
```

```
398     delete from item_pedido;
399     delete from pedido;
400     delete from cliente;
401     delete from produto;
402     delete from tipo_produto;
403 
404     alter table cliente auto_increment = 1;
405     alter table pedido auto_increment = 1;
406     alter table produto auto_increment = 1;
407     alter table tipo_produto auto_increment = 1;
408 
409 
410     set dt1 = unix_timestamp();
411     #set dt1 = UTC_TIMESTAMP();
412 
413     call sp_insereclientes(nclientes);
414     call sp_inseretipoprodutos(quant_tipos);
415     call sp_insereprodutos(quant_produtos);
416     call sp_inserepedidos(quant_pedidos);
417 
418     set dt2 = unix_timestamp();
419 
420     #set dt2 = UTC_TIMESTAMP();
421 
422     set dtdiff = dt2-dt1;
423 
424     select dt1 as 'Inicio da Inserção', dt2 as 'Fim da Inserção',
425     dtdiff as 'Tempo Gasto (em segundos)';
426 
427 
428 end$$
429 DELIMITER ;
```

Fonte: Próprio autor.

ANEXO B – ALGORITMO DE INTERFACE DE OPERAÇÕES

```
<form method="POST" action="<?php $_SERVER['PHP_SELF'];?>">
    <fieldset style="width: 30%;">
    <label> Entre com o numero de clientes</label>
        <select name="numclientes">
           <option>2</option>
           <option>100</option>
           <option>1000</option>
          <option>10000</option>
           <option>100000</option>
           <option>1000000</option>
        </select>
        <br>
        <br>

   <label> Entre com o numero de pedidos por cliente</label>
        <select name="numpedidosporcliente">
           <option>3</option>
           <option>10</option>
           <option>30</option>
           <option>50</option>
    </select>

    <br>
    <br>


    <button type="submit" value="Inserir" name = "Inserir">Inserir</button>

    </fieldset>
</form>
```

ANEXO C – ALGORITMO DE INSERÇÃO - BASE RELACIONAL

```php
<?php

set_time_limit(10000000); //Aumentar tempo limite para 5000min


if(isset($_POST["Inserir"])){

   //CABEÇALHO DO ARQUIVO
  $dados_csv = array();  //ARRAY DE DADOS QUE SERÃO INCLUÍDOS NO CSV DO TESTE
  $numclientes = $_POST['numclientes'];
  $numpedidosporcliente = $_POST['numpedidosporcliente'];
  $filenamecsv = $numclientes.'clientes'.$numpedidosporcliente.'pedidosporcliente'.'.csv';
  $csv = fopen($filenamecsv, "a");
  $head = array("Exec","NumCli","PedporCli","Time");
  fputcsv($csv, $head);
  fclose($csv);


  for ($contador_teste = 1; $contador_teste <= 30; $contador_teste++) {

    $dir = $numclientes.'clientes'.$numpedidosporcliente.'pedidosporcliente';
    ##importação dos dados para banco relacional
    $host = 'localhost';
    $user = 'root';
    $db = 'ecommerce';
    $pass = '';

    #CONEXÃO COM BASE DE DADOS
    try {
      $connection = mysqli_connect($host,$user,$pass,$db) or die(mysqli_error());
      //echo "CONEXÃO EFETUADA COM BASE DE DADOS!";
      //echo "<br />";
    } catch (\Throwable $th) {
      echo "Erro: ".$th->getMessage();
    }


    #SETAR E VERIFICAR A CONEXÃO COM PERMISSÃO PARA USO DE ARQUIVO LOCAL
    #DOCUMENTAÇÃO: https://www.php.net/manual/pt_BR/mysqli.options.php
    /*echo ("PERMISSÃO PARA LER ARQUIVO LOCAL: ");
    echo (*/mysqli_options ( $connection, MYSQLI_OPT_LOCAL_INFILE , 1 );/*)
    ;
```

```
42      echo "<br />";*/
43
44
45      //dropando as tabelas
46      try {
47        $query = "drop table item_pedido";
48        $drop = mysqli_query($connection,$query) or die(mysqli_error(
    $connection));
49
50        $query = "drop table email_cliente";
51        $drop = mysqli_query($connection,$query) or die(mysqli_error(
    $connection));
52
53        $query = "drop table telefone_cliente";
54        $drop = mysqli_query($connection,$query) or die(mysqli_error(
    $connection));
55
56        $query = "drop table pedido";
57        $drop = mysqli_query($connection,$query) or die(mysqli_error(
    $connection));
58
59        $query = "drop table cliente";
60        $drop = mysqli_query($connection,$query) or die(mysqli_error(
    $connection));
61
62        $query = "drop table produto";
63        $drop = mysqli_query($connection,$query) or die(mysqli_error(
    $connection));
64
65        $query = "drop table tipo_produto";
66        $drop = mysqli_query($connection,$query) or die(mysqli_error(
    $connection));
67
68        //echo "BASE ANTIGA APAGADA!";
69      } catch (\Throwable $th) {
70        echo "ERRO ENCONTRADO AO APAGAR BASE ANTIGA: ".$th->getMessage();
71      }
72
73
74
75      //$query = "set innodb_lock_wait_timeout =10000000";
76      //$drop = mysqli_query($connection,$query) or die(mysql_error());
77
78      try {
79        $query = "CREATE TABLE tipo_produto(tipo integer NOT NULL, descricao
    varchar(30) NOT NULL)";
```

```
80      $create = mysqli_query($connection,$query) or die(mysqli_error(
    $connection));
81
82      $query = "CREATE TABLE produto(cod_produto integer NOT NULL, nome
    varchar(30) NOT NULL, pc_unit float DEFAULT NULL, quantidade float
    DEFAULT NULL,tipo integer NOT NULL)";
83      $create = mysqli_query($connection,$query) or die(mysqli_error(
    $connection));
84
85      $query = "CREATE TABLE cliente(cod_cliente integer NOT NULL, nome
    varchar(70) NOT NULL, endereco varchar(90) NOT NULL)";
86      $create = mysqli_query($connection,$query) or die(mysqli_error(
    $connection));
87
88      $query = "CREATE TABLE telefone_cliente(cod_cliente integer NOT NULL,
     telefone integer NOT NULL)";
89      $create = mysqli_query($connection,$query) or die(mysqli_error(
    $connection));
90
91      $query = "CREATE TABLE email_cliente(cod_cliente integer NOT NULL,
    email varchar(30) NOT NULL)";
92      $create = mysqli_query($connection,$query) or die(mysqli_error(
    $connection));
93
94      $query = "CREATE TABLE pedido(cod_pedido integer NOT NULL,
    cod_cliente integer NOT NULL, data_pedido date DEFAULT NULL, total float
     DEFAULT NULL)";
95      $create = mysqli_query($connection,$query) or die(mysqli_error(
    $connection));
96
97      #$query = "CREATE TABLE item_pedido(cod_pedido integer not null,
    cod_item integer not null, pc_unit float, quant float, total float)";
98      $query = "CREATE TABLE item_pedido(cod_item integer not null,
    cod_pedido integer not null, cod_produto integer not null, pc_unit float
    , quant float, total float)";
99      $create = mysqli_query($connection,$query) or die(mysqli_error(
    $connection));
100
101     //echo "TABELAS RECRIADAS NA BASE!";
102   } catch (\Throwable $th) {
103     echo "ERRO ENCONTRADO AO RECRIAR TABELAS: ".$th->getMessage();
104   }
105   //criando o banco sem os índices
106
107
108
109
```

```
110     $totaltime = 0;   //variavel que armazena tempo de processamento das
    consultas (somente comandos insert)
111
112     //setando variaveis do ambiente do servidor mysql
113     $query = "SET GLOBAL local_infile='ON'";  //permite ao servidor copiar
    dados de arquivos locais
114     $setvar = mysqli_query($connection,$query) or die(mysqli_error(
    $connection));
115
116     $query = "SET GLOBAL wait_timeout = 10000000";     //seta tempo de
    time_out de consulta para tempo muito alto
117     $setvar = mysqli_query($connection,$query) or die(mysqli_error(
    $connection));
118
119
120     try {
121
122       #CLIENTE
123       $load = "LOAD DATA LOCAL INFILE 'C:/xampp/htdocs/MariaDB/Insert/
    Sem_chave_primaria/$dir/cliente.txt' INTO TABLE cliente
124       FIELDS TERMINATED BY '\t'
125       LINES TERMINATED BY '\n'
126       ";
127       $timebeforequery = microtime(true);
128       $load1 = mysqli_query($connection,$load);
129       $tempo_operacao = microtime(true)-$timebeforequery;
130       $totaltime += $tempo_operacao;
131
132       #EMAIL-CLIENTE
133       $load = "LOAD DATA LOCAL INFILE 'C:/xampp/htdocs/MariaDB/Insert/
    Sem_chave_primaria/$dir/email_cliente.txt' INTO TABLE email_cliente
134       FIELDS TERMINATED BY '\t'
135       LINES TERMINATED BY '\n'
136       ";
137       $timebeforequery = microtime(true);
138       $load1 = mysqli_query($connection,$load);
139       $tempo_operacao = microtime(true)-$timebeforequery;
140       $totaltime += $tempo_operacao;
141
142       #TELEFONE-CLIENTE
143       $load = "LOAD DATA LOCAL INFILE 'C:/xampp/htdocs/MariaDB/Insert/
    Sem_chave_primaria/$dir/telefone_cliente.txt' INTO TABLE
    telefone_cliente
144       FIELDS TERMINATED BY '\t'
145       LINES TERMINATED BY '\n'
146       ";
147       $timebeforequery = microtime(true);
```

```
148     $load1 = mysqli_query($connection,$load);
149     $tempo_operacao = microtime(true)-$timebeforequery;
150     $totaltime += $tempo_operacao;
151 
152     #PEDIDO
153     $load = "LOAD DATA LOCAL INFILE 'C:/xampp/htdocs/MariaDB/Insert/
    Sem_chave_primaria/$dir/pedido.txt' INTO TABLE pedido
154     FIELDS TERMINATED BY '\t'
155     LINES TERMINATED BY '\n'
156     ";
157     $timebeforequery = microtime(true);
158     $load1 = mysqli_query($connection,$load);
159     $tempo_operacao = microtime(true)-$timebeforequery;
160     $totaltime += $tempo_operacao;
161 
162     #ITEM_PEDIDO
163     $load = "LOAD DATA LOCAL INFILE 'C:/xampp/htdocs/MariaDB/Insert/
    Sem_chave_primaria/$dir/item_pedido.txt' INTO TABLE item_pedido
164     FIELDS TERMINATED BY '\t'
165     LINES TERMINATED BY '\n'
166     ";
167     $timebeforequery = microtime(true);
168     $load1 = mysqli_query($connection,$load);
169     $tempo_operacao = microtime(true)-$timebeforequery;
170     $totaltime += $tempo_operacao;
171 
172     #PRODUTO
173     $load = "LOAD DATA LOCAL INFILE 'C:/xampp/htdocs/MariaDB/Insert/
    Sem_chave_primaria/$dir/produto.txt' INTO TABLE produto
174     FIELDS TERMINATED BY '\t'
175     LINES TERMINATED BY '\n'
176     ";
177     $timebeforequery = microtime(true);
178     $load1 = mysqli_query($connection,$load);
179     $tempo_operacao = microtime(true)-$timebeforequery;
180     $totaltime += $tempo_operacao;
181 
182     #TIPO_PRODUTO
183     $load = "LOAD DATA LOCAL INFILE 'C:/xampp/htdocs/MariaDB/Insert/
    Sem_chave_primaria/$dir/tipo_produto.txt' INTO TABLE tipo_produto
184     FIELDS TERMINATED BY '\t'
185     LINES TERMINATED BY '\n'
186     ";
187     $timebeforequery = microtime(true);
188     $load1 = mysqli_query($connection,$load);
189     $tempo_operacao = microtime(true)-$timebeforequery;
190     $totaltime += $tempo_operacao;
```

```
191
192       //echo "TODAS INSERÇÕES EFETUADAS";
193       //echo "<br />";
194
195     } catch (\Throwable $th) {
196       echo "ERRO ENCONTRADO AO REALIZAR INSERÇÕES: ".$th->getMessage();
197     }
198
199     $dados_csv [$contador_teste] = array ($contador_teste,$numclientes,
    $numpedidosporcliente,$totaltime);
200
201
202   } #FIM LAÇO TESTE
203
204         #MOSTRA INFORMAÇÕES INSERIDAS NA TELA
205         echo "INFORMAÇÕES CSV: <br />";
206         foreach ($dados_csv as $mostra) {
207           print_r($mostra);
208           echo "<br />";
209         }
210         echo "<br />";
211         echo "<br />";
212
213          $csv = fopen($filenamecsv, "a"); //Parâmetro 'a' coloca cursor no
    final do arquivo
214
215          foreach ($dados_csv as $linha) {
216             fputcsv($csv, $linha);
217          }
218       echo "Arquivo .csv de resultados gerado!!<br />";
219       fclose($csv);
220     }
221 ?>
```

ANEXO D – ALGORITMO DE INSERÇÃO - BASE NÃO-RELACIONAL DE UMA COLEÇÃO

```php
<?php

include ("conexao.php");
   set_time_limit(30000000); //Aumentar tempo limite para 5000min

   $conexao -> dropDatabase('ecommerce');
   echo "Base de dados atual excluída para realização de próximo teste";
   echo "<br>";

if(isset($_POST["Inserir"])){


   $numclientes = $_POST['numclientes'];
   $numprodutos = 1000;
   $numtipoprodutos = 50;
   $numpedidosporcliente = $_POST['numpedidosporcliente'];

   $numpedidos = $numpedidosporcliente*$numclientes;

   $mongo = new MongoDB\Driver\Manager("mongodb://localhost:27017");

   // select a database
   //$db = $m->ecommerce;

   $db = $conexao->ecommerce;

   //$collection = $db->clients;

   $collection = $db->shopping;


   //deverá ser armazenado o tempo de execução das inserções.

   $totaltime = 0;  //variavel que armazena tempo de processamento das
    consultas

   $codcli = 1;

   while($codcli<=$numclientes){

   $document = '{"Codigocliente" : '.$codcli.', "Nome": "Cliente '.$codcli.
    '", "Endereco": "Endereco '.$codcli.'", "Pedidos": [';

   $npedidosteste = 1;

```

```
44    while($npedidosteste<=$numpedidosporcliente){
45       $totped = (mt_rand(1000,7000)/100);
46       $document = $document.'{"Codigo": '.$npedidosteste.', "Data":
      "2019/05/15", "Total": '.$totped.', "Itens": [';
47
48       $nitenspedidos = 1;
49
50       while($nitenspedidos<=3){
51
52           $prod = mt_rand(1,$numprodutos);
53           $tpprod = mt_rand(1,$numtipoprodutos);
54           $pcunit = (mt_rand(100,700)/100);
55           $quant = mt_rand(1,7);
56           $tot = $pcunit*$quant;
57           $document = $document.'{"Codigoitem": '.$nitenspedidos.', "
      CodigoProduto": '.$prod.', "NomeProduto": "Produto'.$prod.'",
58                                   "TipoProduto": '.$tpprod.', "PrecoUnit": '
      .$pcunit.', "Quantidade": '.$quant.', "Total": '.$tot.'}';
59
60           if ($nitenspedidos!=3) $document = $document.',';
61           $nitenspedidos = $nitenspedidos+1;
62       }
63       $document = $document.']';
64       $document = $document.'}';
65       if ($npedidosteste!=$numpedidosporcliente) $document = $document.',';
66       $npedidosteste = $npedidosteste + 1;
67    }
68
69    $fone1 = mt_rand(11111111,99999999);
70    $fone2 = mt_rand(11111111,99999999);
71    $fone3 = mt_rand(11111111,99999999);
72
73    $document = $document.'], "Fones": ['.$fone1.','.$fone2.','.$fone3.'], "
      Emails": ["cliente'.$codcli.'@provedor1.com.br","cliente'.$codcli.'
      @provedor2.com.br"]}';
74
75    //echo $document;
76
77    $json_obj = json_decode($document);
78
79    $bulk = new MongoDB\Driver\BulkWrite;//INSTANCIAÇÃO DE UM OBJETO
      BULKWRITE PARA CADA INSERÇÃO DE CADA DOCUMENTO CLIENTE
80    $bulk->insert($json_obj);
81    $timebeforequery = microtime(true);
82    $mongo->executeBulkWrite('ecommerce.shopping', $bulk);
83    $timequery = microtime(true)-$timebeforequery;
84    $totaltime = $totaltime + $timequery;
```

```
85
86
87      $codcli = $codcli+1;
88
89      }
90
91      echo "<br>";
92      echo "Time to insert clients (in seconds)";
93      echo "<br>";
94      echo $totaltime . ' s';
95
96      echo "<br>";
97      echo "tudo feito com sucesso";
98  }
99  ?>
```

ANEXO E – ALGORITMO DE INSERÇÃO - BASE NÃO-RELACIONAL DE DUAS COLEÇÕES

```
<?php
if(isset($_POST["Inserir"])){
   // connect to mongodb
      include ("conexao.php");
      set_time_limit(30000); //Aumentar tempo limite para 500min
      #$m = new Mongo( 'mongodb://localhost:27017');

      $numclientes = $_POST['numclientes'];
      $numprodutos = 1000; //valor fixo em 1000 produtos
      $numtipoprodutos = 50; //valor fixado em 50 tipos de produtos
      $numpedidosporcliente = $_POST['numpedidosporcliente'];
      $numpedidos = $numpedidosporcliente*$numclientes;
      $dados_csv = array(); //ARRAY DE DADOS QUE SERÃO INCLUÍDOS NO CSV DO TESTE
      $filenamecsv = $numclientes.'clientes'.$numpedidosporcliente.'pedidosporcliente'.'.csv';//ARQUIVO CSV
      $csv = fopen($filenamecsv, "a");
      $head = array("Exec","NumCli","PedporCli","Time");
      fputcsv($csv, $head);
      fclose($csv);

      $collectionprod = $ecommerce2->produto;

      //inserir na colecao de produtos os 1000 produtos

      $totaltime = 0; //variavel que armazena o tempo de processamento das consultas

      //INSTÂNCIA MONGO PARA O BULKWRITE
      $mongo = new MongoDB\Driver\Manager("mongodb://localhost:27017");



      $contador_teste = 1;
      $repeticoes = 30;
      while($contador_teste <= $repeticoes){//LAÇO REPETIÇÃO DO TESTE

         $codprod = 1;
         while($codprod <=$numprodutos){//INSERÇÃO COLEÇÃO PRODUTOS
            $tpprod = mt_rand(1,$numtipoprodutos);
            $pcunit = (mt_rand(100,700)/100);
            $quant = mt_rand(1,7);
            $document = '{"_id" : '.$codprod.', "NomeProduto": "Produto  '.$codprod.'", "PrecoUnit": '.$pcunit.', "Quantidade": '.$quant.',
            "TipoProduto": '.$tpprod.'}';
```

```
42
43          $json_obj = json_decode($document);
44          $bulk = new MongoDB\Driver\BulkWrite;//INSTANCIAÇÃO DE UM
    OBJETO BULKWRITE PARA CADA INSERÇÃO DE CADA DOCUMENTO CLIENTE
45
46
47          $timebeforequery = microtime(true);
48          $bulk->insert($json_obj);
49          $mongo->executeBulkWrite('ecommerce2.produto', $bulk);
50          #$collectionprod->insert($json_obj);   //MUDAR PARA BulkWrite
51          $timequery = microtime(true)-$timebeforequery;
52          $totaltime = $totaltime + $timequery;
53
54          $codprod = $codprod + 1;
55       }
56
57       $collectionshop = $ecommerce2->shopping;
58
59       $codcli = 1;
60
61       while($codcli<=$numclientes){//INSERÇÃO COLEÇÃO CLIENTES/PEDIDOS
62
63          $document = '{"Codigocliente" : '.$codcli.', "Nome": "Cliente '
    .$codcli.'", "Endereco": "Endereco '.$codcli.'", "Pedidos": [';
64
65          $npedidosteste = 1;
66
67          while($npedidosteste<=$numpedidosporcliente){
68             $totped = 0;
69             $document = $document.'{"Codigo": '.$npedidosteste.', "Data
    ": "2019/05/15", "Itens": [';
70
71             $nitenspedidos = 1;
72
73             while($nitenspedidos<=3){
74                $prod = mt_rand(1,$numprodutos);
75                $pcunit = (mt_rand(100,700)/100);
76                $quant = mt_rand(1,7);
77                $tot = $pcunit*$quant;
78                $totped = $totped + $tot; //VALOR TOTAL DO PEDIDO É A
    SOMA DOS ITENS
79                $document = $document.'{"Codigoitem": '.$nitenspedidos.',
     "CodigoProduto": '.$prod.', "PrecoUnit": '.$pcunit.', "Quantidade": '.
    $quant.', "Total": '.$tot.'}';
80
81                if ($nitenspedidos!=3) $document = $document.',';
82                $nitenspedidos = $nitenspedidos+1;
```

```
83              }
84              $document = $document.']';
85              $document = $document.', "Total": '.$totped.'';
86              $document = $document.'}';
87              if ($npedidosteste!=$numpedidosporcliente) $document =
    $document.',';
88              $npedidosteste = $npedidosteste + 1;
89          }
90
91          $fone1 = mt_rand(11111111,99999999);
92          $fone2 = mt_rand(11111111,99999999);
93          $fone3 = mt_rand(11111111,99999999);
94
95          $document = $document.'], "Fones": ['.$fone1.','.$fone2.','.
    $fone3.'], "Emails": ["cliente'.$codcli.'@provedor1.com.br","cliente'.
    $codcli.'@provedor2.com.br"]}';
96
97          //echo $document;
98
99          $json_obj = json_decode($document);
100         $bulk = new MongoDB\Driver\BulkWrite;//INSTANCIAÇÃO DE UM
    OBJETO BULKWRITE PARA CADA INSERÇÃO DE CADA DOCUMENTO CLIENTE
101
102
103         $timebeforequery = microtime(true);
104         $bulk->insert($json_obj);
105         $mongo->executeBulkWrite('ecommerce2.shopping', $bulk);
106         #$collectionshop->insert($json_obj);
107         $timequery = microtime(true)-$timebeforequery;
108         $totaltime = $totaltime + $timequery;
109
110         $codcli = $codcli+1;
111
112     }
113
114
115     $dados_csv[$contador_teste] = array (''.$contador_teste.','.
    $numclientes.','.$numpedidosporcliente.','.$totaltime);
116     $conexao -> dropDatabase('ecommerce2');
117
118     $totaltime = 0;
119     $contador_teste = $contador_teste + 1;
120   }//FIM LAÇO TESTE
121
122   echo "INFORMAÇÕES CSV: <br />";
123     foreach ($dados_csv as $mostra) {
124       print_r($mostra);
```

```
125             echo "<br />";
126         }
127         echo "<br />";
128         echo "<br />";
129 
130             $csv = fopen($filenamecsv, "a"); //Parâmetro 'a' coloca cursor
    no final do arquivo
131 
132             foreach ($dados_csv as $linha) {
133                fputcsv($csv, $linha);
134             }
135         echo "Arquivo .csv de resultados gerado!!<br />";
136         fclose($csv);
137 
138       echo "<br />";
139       echo "<br />";
140       echo "Tudo feito com sucesso";
141 }
142 ?>
```

ANEXO F – ALGORITMO DE CONSULTA - CLIENTES ALEATÓRIOS - BASE RELACIONAL

```php
<?php

set_time_limit(10000000); //Aumentar tempo limite para 5000min

if(isset($_POST["Consultar"])){
  $host = 'localhost';
  $user = 'root';
  $db = 'ecommerce';
  $pass = '';
  $connection = mysqli_connect($host,$user,$pass,$db) or die(mysqli_error());
  $numclientes = $_POST['numclientes'];
  $numpedidos = $_POST['numpedidosporcliente'];

  $dados_consulta = array(); //lista dos tempos de consulta do teste
  $nome_arquivo = $numclientes.'clientes'.$numpedidos.'pedidos'.'.csv'; //definição do nome do arquivo
  $csv = fopen($nome_arquivo, "a");
  $head = array("Exec","Tempo");
  fputcsv($csv, $head);
  fclose($csv);


  //clientes sorteados previamente, de acordo com os possíveis números de clientes
  $clientes100 = array
  (21,82,51,19,92,57,100,37,76,30,93,99,65,85,74,94,64,22,43,1,61,45,78,
  31,54,75,97,80,60,98);
  $clientes1000 = array
  (612,948,498,364,788,732,161,843,469,957,927,296,9,945,666,37,600,676,
  272,998,890,671,401,797,167,897,882,502,229,405);
  $clientes10000 = array
  (4968,9626,4054,5277,847,8524,8140,3572,2307,3692,1516,1356,5853,7264,
  2242,3207,3491,7551,8407,7265,9298,7553,2377,1657,
  2286,7105,7513,747,1717,6428);
  $clientes100000 = array
  (63869,8975,53955,8842,32484,15257,95199,52436,29417,85382,27058,42960,
  38275,91166,70567,6820,72572,95891,70099,60802,76271,82415,
  63576,57858,1644,22381,14805,10948,78940,89891);
  $clientes1000000 = array
  (695851,955745,222668,573136,235105,851612,601466,615710,447332,775822,
  326472,381625,330213,858213,708703,850373,661231,570709,
  174143,66100,211463,
  474710,657869,819024,326139,972229,620283,745046,644613,971838);

```

```
37
38  //Ativando os índices da base de dados
39  if ($stmt = mysqli_prepare($connection, "call sp_constroi_indices()")){
40
41    mysqli_stmt_execute($stmt);
42    mysqli_stmt_close($stmt);
43  }
44
45
46  //escolha de qual vetor de clientes utilizar
47  switch ($numclientes) {
48        case 100:
49          $vetorclientes = $clientes100;
50          break;
51        case 1000:
52          $vetorclientes = $clientes1000;
53          break;
54        case 10000:
55          $vetorclientes = $clientes10000;
56          break;
57        case 100000;
58          $vetorclientes = $clientes100000;
59          break;
60        case 1000000;
61          $vetorclientes = $clientes1000000;
62          break;
63        default:
64          $vetorclientes = $clientes100;
65  }
66
67
68  $ntestes = 1;
69
70  while($ntestes<=30){
71
72    $totaltime = 0;
73
74    $cliente = $vetorclientes[$ntestes-1];
75
76    $timebeforequery=microtime(true);
77    $query1 = "select c.*, e.email, t.telefone from cliente c join
     email_cliente e on c.cod_cliente=e.cod_cliente join telefone_cliente t
78          on c.cod_cliente=t.cod_cliente where c.cod_cliente='$cliente'";
79    $selecttable = mysqli_query($connection,$query1) or die(mysql_error());
80    $timequery = microtime(true)-$timebeforequery;
81    $totaltime = $totaltime+$timequery;
82
```

```
83
84      $timebeforequery=microtime(true);
85      $query2 = "select p.data_pedido, p.total, i.* from pedido p join
        item_pedido i on i.cod_pedido=p.cod_pedido
86              where p.cod_cliente='$cliente'";
87      $selecttable = mysqli_query($connection,$query2) or die(mysql_error());
88      $timequery = microtime(true)-$timebeforequery;
89      $totaltime = $totaltime+$timequery;
90
91      $dados_consulta[$ntestes-1]= array ($ntestes,$totaltime);//salvando
        tempo da iteração do teste no array
92      echo "<br>";
93      echo "Test of client  " . $cliente;
94
95      echo "<br>";
96      echo "Time to select data (in miliseconds)";
97      echo "<br>";
98
99
100     echo ($totaltime * 1000) . ' ms'; // in millseconds
101     $ntestes = $ntestes + 1;
102     echo "<br>";
103   }
104
105 echo "INFORMAÇÕES SALVAS: <br />";
106 foreach ($dados_consulta as $mostra) {
107   print_r($mostra);
108   echo "<br />";
109 }
110
111   $csv = fopen($nome_arquivo, "a"); //Parâmetro 'a' coloca cursor no final
      do arquivo
112                                     //$csv é um controlador de fluxo para o
      arquivo
113   //echo "DADOS SALVOS NO ARRAY";
114   foreach ($dados_consulta as $linha) {
115     fputcsv($csv, $linha);
116     }
117 echo "Arquivo .csv de resultados gerado!!<br />";
118 fclose($csv);
119
120
121 echo "<br>";
122 echo "tudo feito com sucesso";
123   }
124
125
```

```
126
127
128 ?>
```

ANEXO G – ALGORITMO DE CONSULTA - CLIENTES ALEATÓRIOS - BASE NÃO-RELACIONAL DE UMA COLEÇÃO

```
<?php

if(isset($_POST["Consultar"])){

   #CONEXÃO COM BASE DE DADOS
  include ("conexao.php");

  $numclientes = $_POST['numclientes'];
  $numpedidosporcliente = $_POST['numpedidosporcliente'];
  $dados_csv = array();#INFORMAÇÕES PARA SALVAR NO CSV
  $filenamecsv = $numclientes.'clientes'.$numpedidosporcliente.'
    pedidosporcliente'.'.csv';
  $csv = fopen("C:/Users/Luciano/Google Drive/Backup/UFVJM/TCC/testesTCC/
    Mongo/Consulta/".$filenamecsv, "a");
  $head = array("Exec","NumCli","PedporCli","Time");
  fputcsv($csv, $head);
  fclose($csv);

 //SELEÇÃO DE BASE DE DADOS E COLEÇÃO
 $db = $conexao->baseteste;
 $collection = $db->clientes;


  //clientes sorteados previamente, de acordo com os possíveis números de
    clientes
  $clientes100 = array
    (21,82,51,19,92,57,100,37,76,30,93,99,65,85,74,94,64,22,43,1,61,45,78,
  31,54,75,97,80,60,98);
  $clientes1000 = array
    (612,948,498,364,788,732,161,843,469,957,927,296,9,945,666,37,600,676,
  272,998,890,671,401,797,167,897,882,502,229,405);
  $clientes10000 = array
    (4968,9626,4054,5277,847,8524,8140,3572,2307,3692,1516,1356,5853,7264,
  2242,3207,3491,7551,8407,7265,9298,7553,2377,1657,
  2286,7105,7513,747,1717,6428);
  $clientes100000 = array
    (63869,8975,53955,8842,32484,15257,95199,52436,29417,85382,27058,42960,
  38275,91166,70567,6820,72572,95891,70099,60802,76271,82415,
  63576,57858,1644,22381,14805,10948,78940,89891);
  $clientes1000000 = array
    (695851,955745,222668,573136,235105,851612,601466,615710,447332,775822,
  326472,381625,330213,858213,708703,850373,661231,570709,
    174143,66100,211463,
  474710,657869,819024,326139,972229,620283,745046,644613,971838);

```

```
37
38  //criacao de indice na base de dados
39  //$collection->ensureIndex(array('Codigocliente' => 1));
40  $collection->createIndex(array('Codigocliente' => 1));
41  //db.shopping.ensureIndex({Codigocliente : 1});
42
43  //escolha de qual vetor de clientes utilizar
44  switch ($numclientes) {
45          case 100:
46            $vetorclientes = $clientes100;
47            break;
48          case 1000:
49            $vetorclientes = $clientes1000;
50            break;
51          case 10000:
52            $vetorclientes = $clientes10000;
53            break;
54          case 100000;
55            $vetorclientes = $clientes100000;
56            break;
57          case 1000000;
58            $vetorclientes = $clientes1000000;
59            break;
60          default:
61            $vetorclientes = $clientes100;
62  }
63
64
65  $ntestes = 1;
66
67  while($ntestes<=30){
68
69     $totaltime = 0;
70
71     $cliente = $vetorclientes[$ntestes-1];#-1 PQ ARRAY COMEÇA DA POSIÇÃO 0
72
73     $timebeforequery=microtime(true);
74     $query = array('Codigocliente' => $cliente);
75     $selectdocument = $collection->find($query);
76     $timequery = microtime(true)-$timebeforequery;
77     $totaltime = $totaltime+$timequery;
78
79     echo "<br>";
80     echo "Test of client  " . $cliente;
81     echo "<br>";
82
```

```
83      $dados_csv[$ntestes-1] = array($ntestes,$numclientes,
    $numpedidosporcliente,$totaltime);
84
85      echo "Time to select data (in miliseconds)";
86      echo "<br>";
87
88
89      echo ($totaltime * 1000) . ' ms'; // in millseconds
90      $ntestes = $ntestes + 1;
91      echo "<br>";
92   }#FIM LAÇO TESTE
93
94
95   #MOSTRA INFORMAÇÕES INSERIDAS NA TELA
96   echo "INFORMAÇÕES CSV: <br />";
97   foreach ($dados_csv as $mostra) {
98     print_r($mostra);
99     echo "<br />";
100  }
101  echo "<br />";
102  echo "<br />";
103
104   $csv = fopen("C:/Users/Luciano/Google Drive/Backup/UFVJM/TCC/testesTCC/
    Mongo/Consulta/".$filenamecsv, "a"); //Parâmetro 'a' coloca cursor no
    final do arquivo
105
106   foreach ($dados_csv as $linha) {
107      fputcsv($csv, $linha);
108   }
109
110    echo "Arquivo .csv de resultados gerado!!<br />";
111    fclose($csv);
112
113
114
115
116    echo "tudo feito com sucesso";
117    echo "<br>";
118
119    $conexao -> dropDatabase('baseteste');
120    echo "Base de dados excluída para realização de próximo teste";
121 }
122
123
124 ?>
```

ANEXO H – ALGORITMO DE CONSULTA - CLIENTES ALEATÓRIOS - BASE NÃO-RELACIONAL DE DUAS COLEÇÕES

```
<?php

if(isset($_POST["Consultar"])){

  #CONEXÃO
  include ("conexao.php");
  echo "Connection to database successfully";

   //SELEÇÃO DE BASE DE DADOS E COLEÇÃO
  $db = $conexao->baseteste;
  $collection1 = $db->shopping;
  $collection2 = $db->produto;

  $numclientes = $_POST['numclientes'];
  $numpedidosporcliente = $_POST['numpedidosporcliente'];
  $dados_csv = array();#INFORMAÇÕES PARA SALVAR NO CSV
  $filenamecsv = $numclientes.'clientes'.$numpedidosporcliente.'
    pedidosporcliente'.'.csv';
  $csv = fopen("C:/Users/Luciano/Google Drive/Backup/UFVJM/TCC/testesTCC/
    Mongo/2colections/consulta/modificado/".$filenamecsv, "a");
  $head = array("Exec","NumCli","PedporCli","Time");
  fputcsv($csv, $head);
  fclose($csv);

  //clientes sorteados previamente, de acordo com os possíveis números de
    clientes
  $clientes100 = array
    (21,82,51,19,92,57,100,37,76,30,93,99,65,85,74,94,64,22,43,1,61,45,78,
  31,54,75,97,80,60,98);
  $clientes1000 = array
    (612,948,498,364,788,732,161,843,469,957,927,296,9,945,666,37,600,676,
  272,998,890,671,401,797,167,897,882,502,229,405);
  $clientes10000 = array
    (4968,9626,4054,5277,847,8524,8140,3572,2307,3692,1516,1356,5853,7264,
  2242,3207,3491,7551,8407,7265,9298,7553,2377,1657,
  2286,7105,7513,747,1717,6428);
  $clientes100000 = array
    (63869,8975,53955,8842,32484,15257,95199,52436,29417,85382,27058,42960,
  38275,91166,70567,6820,72572,95891,70099,60802,76271,82415,
  63576,57858,1644,22381,14805,10948,78940,89891);
  $clientes1000000 = array
    (695851,955745,222668,573136,235105,851612,601466,615710,447332,775822,
  326472,381625,330213,858213,708703,850373,661231,570709,
    174143,66100,211463,
  474710,657869,819024,326139,972229,620283,745046,644613,971838);
```

```
37
38
39  //criacao de indice na base de dados
40  //$collection1->ensureIndex(array('Codigocliente' => 1));
41  //$collection2->ensureIndex(array('_id' => 1));
42
43  $collection1->createIndex(array('Codigocliente' => 1));
44  $collection2->createIndex(array('_id' => 1));
45  //db.shopping.ensureIndex({Codigocliente : 1});
46
47  //escolha de qual vetor de clientes utilizar
48  switch ($numclientes) {
49          case 100:
50            $vetorclientes = $clientes100;
51            break;
52          case 1000:
53            $vetorclientes = $clientes1000;
54            break;
55          case 10000:
56            $vetorclientes = $clientes10000;
57            break;
58          case 100000;
59            $vetorclientes = $clientes100000;
60            break;
61          case 1000000;
62            $vetorclientes = $clientes1000000;
63            break;
64          default:
65            $vetorclientes = $clientes100;
66  }
67
68
69  $ntestes = 1;
70
71  while($ntestes<=3){
72
73    $totaltime = 0;
74
75    $cliente = $vetorclientes[$ntestes-1];
76
77
78    $timebeforequery=microtime(true);
79    $query = array('Codigocliente' => $cliente);
80    $selectdocument = $collection1->find($query);
81    $timequery = microtime(true)-$timebeforequery;
82    $totaltime = $totaltime+$timequery;
83
```

```
84
85      $timebeforequery = microtime(true);
86      $query = array();
87      $selectdocument = $collection2->find($query);
88      $timequery = microtime(true)-$timebeforequery;
89      $totaltime = $totaltime+$timequery;
90
91      echo "<br>";
92      echo "Test of client  " . $cliente;
93      echo "<br>";
94
95      echo "Time to select data (in miliseconds)";
96      echo "<br>";
97
98
99      echo ($totaltime * 1000) . ' ms'; // in millseconds
100     $ntestes = $ntestes + 1;
101     echo "<br>";
102   }
103
104
105   echo "<br>";
106   echo "tudo feito com sucesso";
107
108   }
109 ?>
```

ANEXO I – ALGORITMO DE CONSULTA - MAIORES COMPRADORES - BASE RELACIONAL

```php
<?php

  set_time_limit(10000000);

  if(isset($_POST["Consultar"])){

      $host = 'localhost';
      $user = 'root';
      $db = 'ecommerce';
      $pass = '';
      $connection = mysqli_connect($host,$user,$pass,$db) or die(
    mysqli_error());
      $numclientes = $_POST['numclientes'];
      $numpedidos = $_POST['numpedidosporcliente'];

      $dados_consulta = array(); //lista dos tempos de consulta do teste
      $nome_arquivo = $numclientes.'clientes'.$numpedidos.'pedidos'.'.csv';
     //definição do nome do arquivo
      $csv = fopen($nome_arquivo, "a");
      $head = array("Exec","Tempo");
      fputcsv($csv, $head);
      fclose($csv);


      //vetor dos "n" maiores clientes compradores que devem ser obtidos a
    cada consulta
      $topclientes = array
    (52,98,45,1,15,92,21,46,80,33,13,69,11,86,93,17,2,24,54,51,85,65,40,96,
      73,99,95,29,60,83);

      //Ativando os índices da base de dados
      if ($stmt = mysqli_prepare($connection, "call sp_constroi_indices()")
    ){

        mysqli_stmt_execute($stmt);
        mysqli_stmt_close($stmt);
      }


      $ntestes = 1;

      while($ntestes<=30){

        $totaltime = 0;

```

```
41        $top = $topclientes[$ntestes-1];
42
43        $timebeforequery=microtime(true);
44        $query = "select c.cod_cliente, nome, sum(p.total) as 'Soma' from
    cliente c join
45                  pedido p on c.cod_cliente=p.cod_cliente group by c.
    cod_cliente, nome order by 'Soma' desc limit ".$top;
46        $selecttable = mysqli_query($connection,$query) or die(mysql_error
    ());
47        $timequery = microtime(true)-$timebeforequery;
48        $totaltime = $totaltime+$timequery;
49
50        $dados_consulta[$ntestes-1]= array ($ntestes,$totaltime);//salvando
     tempo da iteração do teste no array
51
52        echo "<br>";
53        echo "Test of client  " . $top;
54        echo "<br>";
55
56        echo "<br>";
57        echo "Time to select data (in miliseconds)";
58        echo "<br>";
59
60
61        echo ($totaltime * 1000) . ' ms'; // in millseconds
62        $ntestes = $ntestes + 1;
63        echo "<br>";
64     }#FIM LAÇO TESTE
65
66    echo "INFORMAÇÕES SALVAS: <br />";
67    foreach ($dados_consulta as $mostra) {
68       print_r($mostra);
69       echo "<br />";
70    }
71       $csv = fopen($nome_arquivo, "a"); //Parâmetro 'a' coloca cursor no
    final do arquivo
72                                  //$csv é um controlador de fluxo para o
    arquivo
73    //echo "DADOS SALVOS NO ARRAY";
74    foreach ($dados_consulta as $linha) {
75       fputcsv($csv, $linha);
76     }
77
78    echo "Arquivo .csv de resultados gerado!!<br />";
79    fclose($csv);
80
81    echo "<br>";
```

```
82     echo "tudo feito com sucesso";
83 
84      }
85 ?>
```

ANEXO J – ALGORITMO DE CONSULTA - MAIORES COMPRADORES - BASE NÃO-RELACIONAL DE UMA COLEÇÃO

```
<?php
if(isset($_POST["Consultar"])){

    set_time_limit(30000); //Aumentar tempo limite para 500min

    include ("conexao.php");
    echo "Connection to database successfully";

    $numclientes = $_POST['numclientes'];
    $numpedidosporcliente = $_POST['numpedidosporcliente'];
    $dados_csv = array();#INFORMAÇÕES PARA SALVAR NO CSV
    $filenamecsv = $numclientes.'clientes'.$numpedidosporcliente.'
    pedidosporcliente'.'.csv';
    $csv = fopen("C:/Users/Luciano/Google Drive/Backup/UFVJM/TCC/testesTCC/
    Mongo/Consulta/TopClientes/".$filenamecsv, "a");
    $head = array("Exec","NumCli","PedporCli","Time");
    fputcsv($csv, $head);
    fclose($csv);

     //SELEÇÃO DE BASE DE DADOS E COLEÇÃO
    $db = $conexao->baseteste;
    $collection = $db->clientes;

    //essa variavel armazena os "n" melhores clientes buscados a cada
    iteração do código
    $topclientes = array
    (52,98,45,1,15,92,21,46,80,33,13,69,11,86,93,17,2,24,54,51,85,65,40,96,
    73,99,95,29,60,83);


    //criacao de indice na base de dados utilizando codigo do cliente
    //$collection->ensureIndex(array('Codigocliente' => 1));
    $collection->createIndex(array('Codigocliente' => 1));


    $ntestes = 1;

    while($ntestes<=30){

       $totaltime = 0;

       $top = $topclientes[$ntestes-1];


       $timebeforequery=microtime(true);
```

```
42
43      $query = array(array('$unwind' => '$Pedidos'),
44                     array('$group' => array('_id' => array('Codigo' => '
  $Codigocliente', 'Nome' => '$Nome'),
45                             'sum' => array('$sum' => '$Pedidos.Total'))),
46                     array('$sort' => array('sum' => -1)),
47                     array('$limit' => $top),
48                     array('$project' => array('_id' => 1, 'sum' => 1)));
49
50
51      $selectdocument = $collection->aggregate($query);
52      $timequery = microtime(true)-$timebeforequery;
53      $totaltime = $totaltime+$timequery;
54
55      echo "<br>";
56      echo "Test of top client  " . $top;
57      echo "<br>";
58
59      $dados_csv[$ntestes-1] = array($ntestes,$numclientes,
  $numpedidosporcliente,$totaltime);
60
61      echo "Time to select data (in miliseconds)";
62      echo "<br>";
63
64
65      echo ($totaltime * 1000) . ' ms'; // in millseconds
66      $ntestes = $ntestes + 1;
67      echo "<br>";
68   }#FIM LAÇO TESTE
69
70     #MOSTRA INFORMAÇÕES INSERIDAS NA TELA
71  echo "INFORMAÇÕES CSV: <br />";
72  foreach ($dados_csv as $mostra) {
73    print_r($mostra);
74    echo "<br />";
75  }
76  echo "<br />";
77  echo "<br />";
78
79   $csv = fopen("C:/Users/Luciano/Google Drive/Backup/UFVJM/TCC/testesTCC/
   Mongo/Consulta/TopClientes/".$filenamecsv, "a"); //Parâmetro 'a' coloca
   cursor no final do arquivo
80
81   foreach ($dados_csv as $linha) {
82      fputcsv($csv, $linha);
83   }
84
```

```
85      echo "Arquivo .csv de resultados gerado!!<br />";
86      fclose($csv);
87
88
89      echo "tudo feito com sucesso";
90      echo "<br>";
91
92      $conexao -> dropDatabase('baseteste');
93      echo "Base de dados excluída para realização de próximo teste";
94  }
95  ?>
```

ANEXO K – ALGORITMO DE CONSULTA - MAIORES COMPRADORES - BASE NÃO-RELACIONAL DE DUAS COLEÇÕES

```
1
2  <?php
3
4  if(isset($_POST["Consultar"])){
5
6      set_time_limit(30000); //Aumentar tempo limite para 500min
7
8       #CONEXÃO COM BASE DE DADOS
9      include ("conexao.php");
10     echo "Connection to database successfully";
11     echo "<br/>";
12
13     $numclientes = $_POST['numclientes'];
14     $numpedidosporcliente = $_POST['numpedidosporcliente'];
15     $dados_csv = array();#INFORMAÇÕES PARA SALVAR NO CSV
16     $filenamecsv = $numclientes.'clientes'.$numpedidosporcliente.'
       pedidosporcliente'.'.csv';
17     $csv = fopen($filenamecsv, "a");
18     $head = array("Exec","NumCli","PedporCli","Time");
19     fputcsv($csv, $head);
20     fclose($csv);
21
22     //SELEÇÃO DE BASE DE DADOS E COLEÇÃO
23     $db = $conexao->baseteste;
24     $collection = $db->shopping;
25
26
27
28     //essa variavel armazena os "n" melhores clientes buscados a cada
       iteração do código
29     $topclientes = array
       (52,98,45,1,15,92,21,46,80,33,13,69,11,86,93,17,2,24,54,51,85,65,40,96,
30     73,99,95,29,60,83);
31
32
33     //criacao de indice na base de dados utilizando codigo do produto
34     //$collection->ensureIndex(array('Codigocliente' => 1));
35     $collection->createIndex(array('Codigocliente' => 1));
36
37
38     $ntestes = 1;
39
40     while($ntestes<=30){
41
42       $totaltime = 0;
```

```
43
44      $top = $topclientes[$ntestes-1];
45
46
47      $timebeforequery=microtime(true);
48
49      $query = array(array('$unwind' => '$Pedidos'),
50                      array('$group' => array('_id' => array('Codigo' => '
    $Codigocliente', 'Nome' => '$Nome'),
51                              'sum' => array('$sum' => '$Pedidos.Total'))),
52                      array('$sort' => array('sum' => -1)),
53                      array('$limit' => $top),
54                      array('$project' => array('_id' => 1, 'sum' => 1)));
55
56
57      $options = array("allowDiskUse" => true);//PARA MOSTRAR RESULTADO NA
    TELA, SEM ELE, A MEMÓRIA É INSUFICIENTE
58      $selectdocument = $collection->aggregate($query, $options);
59      //$selectdocument = $collection->aggregate($query);
60      $timequery = microtime(true)-$timebeforequery;
61      $totaltime = $totaltime+$timequery;
62
63      echo "<br>";
64      echo "Test of top clients  " . $top;
65      echo "<br>";
66
67
68      foreach ($selectdocument as $obj) {
69         echo "Sum: ".$obj->sum;
70         echo "<br/>";
71      }
72
73      #DADOS PARA CSV
74      $dados_csv[$ntestes-1] = array($ntestes,$numclientes,
    $numpedidosporcliente,$totaltime);
75
76
77       echo "Time to select data (in miliseconds)";
78       echo "<br>";
79
80
81       echo ($totaltime * 1000) . ' ms'; // in millseconds
82       $ntestes = $ntestes + 1;
83       echo "<br>";
84       echo "<br/>";
85       echo "<br/>";
86    }#FIM LAÇO TESTE
```

```
87
88
89
90      #MOSTRA INFORMAÇÕES INSERIDAS NA TELA
91      echo "INFORMAÇÕES CSV: <br />";
92      foreach ($dados_csv as $mostra) {
93          print_r($mostra);
94          echo "<br />";
95      }
96      echo "<br />";
97      echo "<br />";
98
99      $csv = fopen($filenamecsv, "a"); //Parâmetro 'a' coloca cursor no final
        do arquivo
100
101     foreach ($dados_csv as $linha) {
102         fputcsv($csv, $linha);
103     }
104
105      echo "Arquivo .csv de resultados gerado!!<br />";
106      fclose($csv);
107
108      echo "tudo feito com sucesso";
109      echo "<br>";
110
111 }
112 ?>
```

ANEXO L – ALGORITMO DE CONSULTA - PRODUTOS COM MAIOR VALOR DE VENDAS - BASE RELACIONAL

```php
<?php

 set_time_limit(10000000);

 if(isset($_POST["Consultar"])){

     $host = 'localhost';
     $user = 'root';
     $db = 'ecommerce';
     $pass = '';
     $connection = mysqli_connect($host,$user,$pass,$db) or die(
  mysqli_error());
     $numclientes = $_POST['numclientes'];
     $numpedidos = $_POST['numpedidosporcliente'];

     $dados_consulta = array(); //lista dos tempos de consulta do teste
     $nome_arquivo = $numclientes.'clientes'.$numpedidos.'pedidos'.'.csv';
   //definição do nome do arquivo
     $csv = fopen($nome_arquivo, "a");
     $head = array("Exec","Tempo");
     fputcsv($csv, $head);
     fclose($csv);


     //
     $topprodutos = array
  (52,98,45,1,15,92,21,46,80,33,13,69,11,86,93,17,2,24,54,51,85,65,40,96,
     73,99,95,29,60,83);

     //Ativando os índices da base de dados
     if ($stmt = mysqli_prepare($connection, "call sp_constroi_indices()")
  ){

       mysqli_stmt_execute($stmt);
       mysqli_stmt_close($stmt);
     }


     $ntestes = 1;

     while($ntestes<=30){

       $totaltime = 0;

       $top = $topprodutos[$ntestes-1];
```

```
42
43          $timebeforequery=microtime(true);
44          $query = "select p.cod_produto, sum(i.total) as 'Soma'
45                    from produto p join item_pedido i on p.cod_produto=i.
    cod_produto
46                    group by p.cod_produto order by 'Soma' desc limit ".$top;
47
48          $selecttable = mysqli_query($connection,$query) or die(mysql_error
    ());
49          $timequery = microtime(true)-$timebeforequery;
50          $totaltime = $totaltime+$timequery;
51
52          $dados_consulta[$ntestes-1]= array ($ntestes,$totaltime);//salvando
     tempo da iteração do teste no array
53
54          echo "<br>";
55          echo "Test of top produts  " . $top;
56          echo "<br>";
57
58          echo "<br>";
59          echo "Time to select data (in miliseconds)";
60          echo "<br>";
61
62
63          echo ($totaltime * 1000) . ' ms'; // in millseconds
64          $ntestes = $ntestes + 1;
65          echo "<br>";
66       }#FIM LAÇO TESTE
67
68    echo "INFORMAÇÕES SALVAS: <br />";
69    foreach ($dados_consulta as $mostra) {
70       print_r($mostra);
71       echo "<br />";
72    }
73
74    $csv = fopen($nome_arquivo, "a"); //Parâmetro 'a' coloca cursor no final
     do arquivo
75                                      //$csv é um controlador de fluxo para o
    arquivo
76    echo "DADOS SALVOS NO ARRAY";
77    foreach ($dados_consulta as $linha) {
78       fputcsv($csv, $linha);
79     }
80
81    echo "Arquivo .csv de resultados gerado!!<br />";
82    fclose($csv);
83
```

```
84      echo "<br>";
85      echo "tudo feito com sucesso";
86
87       }
88  ?>
```

ANEXO M – ALGORITMO DE CONSULTA - PRODUTOS COM MAIOR VALOR DE VENDAS - BASE NÃO-RELACIONAL DE UMA COLEÇÃO

```
<?php
if(isset($_POST["Consultar"])){

    set_time_limit(3000000); //Aumentar tempo limite para 500min

    include ("conexao.php");
    echo "Connection to database successfully";

    $numclientes = $_POST['numclientes'];
    $numpedidosporcliente = $_POST['numpedidosporcliente'];
    $dados_csv = array();#INFORMAÇÕES PARA SALVAR NO CSV
    $filenamecsv = $numclientes.'clientes'.$numpedidosporcliente.'
   pedidosporcliente'.'.csv';

     //SELEÇÃO DE BASE DE DADOS E COLEÇÃO
    $db = $conexao->ecommerce;
    $collection = $db->shopping;

    //essa variavel armazena os "n" melhores clientes buscados a cada
   iteração do código
    $topprodutos = array
   (52,98,45,1,15,92,21,46,80,33,13,69,11,86,93,17,2,24,54,51,85,65,40,96,
    73,99,95,29,60,83);


    //criacao de indice na base de dados utilizando codigo do cliente
    $collection->createIndex(array('Codigocliente' => 1));
    //$collection->createIndex(array('codigo_do_cliente' => 1));


    $ntestes = 1;

    while($ntestes<=30){

       $totaltime = 0;

       $top = $topprodutos[$ntestes-1];


       $timebeforequery=microtime(true);
      $query = array(array('$unwind' => '$Pedidos'),
                     array('$unwind' => '$Pedidos.Itens'),
                     array('$group' => array('_id' => array('Codigo' =>
   '$Pedidos.Itens.CodigoProduto'/*,
```

```
41                                                                  'Nome' => '$Pedidos.Itens.NomeProduto'*/),
42                                                  'sum' => array('$sum' => '$Pedidos.Itens.Total'))),
43                      array('$sort' => array('sum' => -1)),
44                      array('$limit' => $top),
45                      array('$project' => array('_id' => 1, 'sum' => 1)));
46
47
48      $options = array("allowDiskUse" => true);//PARA MOSTRAR RESULTADO NA TELA, SEM ELE, A MEMÓRIA É INSUFICIENTE
49      $selectdocument = $collection->aggregate($query,$options);//, $options);
50      //$selectdocument = $collection->aggregate($query);
51      $timequery = microtime(true)-$timebeforequery;
52      $totaltime = $totaltime+$timequery;
53
54       echo "<br>";
55       echo "Test of top products  " . $top;
56       echo "<br>";
57
58
59       //só pra testar se os dados estavam mesmo sendo obtidos
60       echo "---Dados Consultados---";
61       echo "<br>";
62       echo "<br>";
63
64      foreach ($selectdocument as $doc) {
65         //var_dump($doc);
66         echo "Codigo: ".$doc->_id->Codigo;
67         echo "<br/>";
68         echo "Sum: ".$doc->sum;
69         echo "<br/>";
70         echo "<br/>";
71      }
72       echo "<br>";
73
74
75       $dados_csv[$ntestes-1] = array($ntestes,$numclientes, $numpedidosporcliente,$totaltime);
76
77       echo "Time to select data (in miliseconds)";
78       echo "<br>";
79
80
81       echo ($totaltime * 1000) . ' ms'; // in millseconds
```

```
82          $ntestes = $ntestes + 1;
83          echo "<br>";
84         }#FIM LAÇO TESTE
85
86         #MOSTRA INFORMAÇÕES INSERIDAS NA TELA
87    echo "INFORMAÇÕES CSV: <br />";
88    foreach ($dados_csv as $mostra) {
89      print_r($mostra);
90      echo "<br />";
91     }
92
93    echo "<br />";
94    echo "<br />";
95
96     $csv = fopen($filenamecsv, "a"); //Parâmetro 'a' coloca cursor no final
       do arquivo
97
98     foreach ($dados_csv as $linha) {
99        fputcsv($csv, $linha);
100    }
101
102     echo "Arquivo .csv de resultados gerado!!<br />";
103     fclose($csv);
104
105     echo "tudo feito com sucesso";
106     echo "<br>";
107
108 }
109 ?>
```

ANEXO N – ALGORITMO DE CONSULTA - PRODUTOS COM MAIOR VALOR DE VENDAS - BASE NÃO-RELACIONAL DE DUAS COLEÇÕES

```
<?php

if(isset($_POST["Consultar"])){


   set_time_limit(30000); //Aumentar tempo limite para 500min

  #CONEXÃO COM BASE DE DADOS
   include ("conexao.php");
   echo "Connection to database successfully";
   echo "<br/>";

    $numclientes = $_POST['numclientes'];
    $numpedidosporcliente = $_POST['numpedidosporcliente'];
    $dados_csv = array();#INFORMAÇÕES PARA SALVAR NO CSV
    $filenamecsv = $numclientes.'clientes'.$numpedidosporcliente.'
   pedidosporcliente'.'.csv';
    $csv = fopen($filenamecsv, "a");
    $head = array("Exec","NumCli","PedporCli","Time");
    fputcsv($csv, $head);
    fclose($csv);

    // select a database and collection
    //$db = $m->ecommerce2;
    $db = $conexao->baseteste;
    $collection1 = $db->shopping;
    $collection2 = $db->produto;

    //essa variavel armazena os "n" melhores clientes buscados a cada
   iteração do código
    $topprodutos = array
   (52,98,45,1,15,92,21,46,80,33,13,69,11,86,93,17,2,24,54,51,85,65,40,96,
    73,99,95,29,60,83);


    //criacao de indice na base de dados utilizando codigo do produto
    //$collection1->ensureIndex(array('CodigoProduto' => 1));
    //$collection2->ensureIndex(array('_id' => 1));

    $collection1->createIndex(array('Codigocliente' => 1));
    $collection2->createIndex(array('_id' => 1));

    $ntestes = 1;

    while($ntestes<=30){
```

```
43
44      $totaltime = 0;
45
46      $top = $topprodutos[$ntestes-1];
47
48      $query = array(array('$unwind' => '$Pedidos'),
49                     array('$unwind' => '$Pedidos.Itens'),
50                     array('$group' => array('_id' => array('Codigo' => '
    $Pedidos.Itens.CodigoProduto'),
51                                             'sum' => array('$sum' => '
    $Pedidos.Itens.Total'))),
52                     array('$sort' => array('sum' => -1)),
53                     array('$limit' => $top),
54                     array('$project' => array('_id' => 1, 'sum' => 1)));
55
56      $timebeforequery=microtime(true);
57      $selectdocument1 = $collection1->aggregate($query);
58      $timequery = microtime(true)-$timebeforequery;
59      $totaltime = $totaltime+$timequery;
60
61
62      $query = array();
63      $timebeforequery=microtime(true);
64      $selectdocument2 = $collection2->find($query);
65      $timequery = microtime(true)-$timebeforequery;
66      $totaltime = $totaltime+$timequery;
67
68      echo "<br>";
69      echo "Test of top produtos  " . $top;
70      echo "<br>";
71
72      //só pra testar se os dados estavam mesmo sendo obtidos
73
74      try {
75
76        foreach ($selectdocument1 as $doc) {
77           echo "Codigo: ".$doc->_id->Codigo;
78           echo "<br/>";
79           echo "Sum: ".$doc->sum;
80           echo "<br/>";
81           echo "<br/>";
82        }
83      } catch (\Throwable $th) {
84        echo "Erro: ".$th->getMessage();
85      }
86
87
```

```
88          $dados_csv[$ntestes-1] = array($ntestes,$numclientes,
    $numpedidosporcliente,$totaltime);
89
90
91          echo "Time to select data (in miliseconds)";
92          echo "<br>";
93
94
95          echo ($totaltime * 1000) . ' ms'; // in millseconds
96          $ntestes = $ntestes + 1;
97          echo "<br>";
98          echo "<br/>";
99          echo "<br/>";
100      }#FIM LAÇO TESTE
101
102      echo "INFORMAÇÕES CSV: <br />";
103     foreach ($dados_csv as $mostra) {
104        print_r($mostra);
105        echo "<br />";
106     }
107     echo "<br />";
108     echo "<br />";
109
110     $csv = fopen($filenamecsv, "a"); //Parâmetro 'a' coloca cursor no final
    do arquivo
111
112     foreach ($dados_csv as $linha) {
113        fputcsv($csv, $linha);
114     }
115
116
117      echo "Arquivo .csv de resultados gerado!!<br />";
118      fclose($csv);
119
120      echo "<br>";
121      echo "tudo feito com sucesso";
122
123 }
124 ?>
```

ANEXO O – ALGORITMO DE CONSULTA - PRODUTOS ALEATÓRIOS - BASE RELACIONAL

```
<?php

 set_time_limit(10000000);

 if(isset($_POST["Consultar"])){

     $host = 'localhost';
     $user = 'root';
     $db = 'ecommerce';
     $pass = '';
     $connection = mysqli_connect($host,$user,$pass,$db) or die(
  mysqli_error());
     $numclientes = $_POST['numclientes'];
     $numpedidos = $_POST['numpedidosporcliente'];
     $dados_consulta = array(); //lista dos tempos de consulta do teste
     $nome_arquivo = $numclientes.'clientes'.$numpedidos.'pedidos'.'.csv';
   //definição do nome do arquivo

     $csv = fopen($nome_arquivo, "a");
     $head = array("Exec","Tempo");
     fputcsv($csv, $head);
     fclose($csv);


     //produtos sorteados previamente, de acordo com os possíveis números
  de clientes
     $vetorprodutos = array
  (172,493,858,2,41,849,747,752,417,76,269,408,439,463,633,80,336,623,400,
     948,935,301,316,613,630,252,970,913,214,810);

     //Ativando os índices da base de dados
     if ($stmt = mysqli_prepare($connection, "call sp_constroi_indices()")
  ){

       mysqli_stmt_execute($stmt);
       mysqli_stmt_close($stmt);
     }


     $ntestes = 1;

     while($ntestes<=30){

       $totaltime = 0;

```

```
41        $produto = $vetorprodutos[$ntestes-1];
42
43        $timebeforequery=microtime(true);
44        $query = "select c.cod_cliente, i.cod_item, i.cod_produto, p.nome,
    t.descricao, i.pc_unit, i.quant, i.total
45                  from cliente c join pedido pe on c.cod_cliente=pe.
    cod_cliente join item_pedido i on
46                  i.cod_pedido=pe.cod_pedido join produto p on i.
    cod_produto=p.cod_produto join tipo_produto
47                  t on p.tipo=t.tipo where p.cod_produto='$produto'";
48
49        $selecttable = mysqli_query($connection,$query) or die(mysql_error
    ());
50        $timequery = microtime(true)-$timebeforequery;
51        $totaltime = $totaltime+$timequery;
52
53        $dados_consulta[$ntestes-1]= array ($ntestes,$totaltime);//salvando
     tempo da iteração do teste no array
54
55        echo "<br>";
56        echo "Test of produto  " . $produto;
57        echo "<br>";
58
59
60        echo "<br>";
61        echo "Time to select data (in miliseconds)";
62        echo "<br>";
63
64
65        echo ($totaltime * 1000) . ' ms'; // in millseconds
66        $ntestes = $ntestes + 1;
67        echo "<br>";
68     }#FIM LAÇO TESTE
69
70    echo "INFORMAÇÕES SALVAS: <br />";
71    foreach ($dados_consulta as $mostra) {
72       print_r($mostra);
73       echo "<br />";
74    }
75
76
77    $csv = fopen($nome_arquivo, "a"); //Parâmetro 'a' coloca cursor no final
     do arquivo
78                                      //$csv é um controlador de fluxo para o
    arquivo
79    echo "DADOS SALVOS NO ARRAY";
80    foreach ($dados_consulta as $linha) {
```

```
81        fputcsv($csv, $linha);
82      }
83
84    echo "Arquivo .csv de resultados gerado!!<br />";
85    fclose($csv);
86
87
88    echo "<br>";
89    echo "tudo feito com sucesso";
90
91      }
92 ?>
```

ANEXO P – ALGORITMO DE CONSULTA - PRODUTOS ALEATÓRIOS - BASE NÃO-RELACIONAL DE UMA COLEÇÃO

```
<?php
if(isset($_POST["Consultar"])){

    set_time_limit(3000000); //Aumentar tempo limite para 500min

    include ("conexao.php");
    echo "Connection to database successfully";

    $numclientes = $_POST['numclientes'];
    $numpedidosporcliente = $_POST['numpedidosporcliente'];
    $dados_csv = array();#INFORMAÇÕES PARA SALVAR NO CSV
    $filenamecsv = $numclientes.'clientes'.$numpedidosporcliente.'
   pedidosporcliente'.'.csv';

    $csv = fopen($filenamecsv, "a");
    $head = array("Exec","NumCli","PedporCli","Time");
    fputcsv($csv, $head);
    fclose($csv);

     //SELEÇÃO DE BASE DE DADOS E COLEÇÃO
    $db = $conexao->ecommerce;
    $collection = $db->shopping;

    //produtos sorteados previamente, de acordo com os possíveis números de
     produtos
    $produtos = array
   (172,493,858,2,41,849,747,752,417,76,269,408,439,463,633,80,336,623,400,
    948,935,301,316,613,630,252,970,913,214,810);


    //criacao de indice na base de dados utilizando codigo do cliente
    $collection->createIndex(array('CodigoProduto' => 1));
    //$collection->createIndex(array('codigo_do_cliente' => 1));


    $ntestes = 1;

    while($ntestes<=30){

       $totaltime = 0;

       $produto = $produtos[$ntestes-1];

       $timebeforequery=microtime(true);

```

```
43        $query = array(array('$unwind' => '$Pedidos'),
44                       array('$unwind' => '$Pedidos.Itens'),
45                       array('$match' => array('Pedidos.Itens.CodigoProduto'
     => $produto)),
46                       array('$project' => array('Codigocliente' => 1, '
    Pedidos.Itens.Codigoitem' => 1,
47                                                 'Pedidos.Itens.
    CodigoProduto' => 1, 'Pedidos.Itens.NomeProduto' => 1,
48                                                 'Pedidos.Itens.TipoProduto'
     => 1, 'Pedidos.Itens.PrecoUnit' => 1,
49                                                 'Pedidos.Itens.Quantidade'
    => 1, 'Pedidos.Itens.Total' => 1)));
50
51
52
53        $options = array("allowDiskUse" => true);//PARA MOSTRAR RESULTADO NA
    TELA, SEM ELE, A MEMÓRIA É INSUFICIENTE
54        $selectdocument = $collection->aggregate($query,$options);//,
    $options);
55        //$selectdocument = $collection->aggregate($query);
56        $timequery = microtime(true)-$timebeforequery;
57        $totaltime = $totaltime+$timequery;
58
59         echo "<br>";
60         echo "Test of product  " . $produto;
61         echo "<br>";
62
63
64         //só pra testar se os dados estavam mesmo sendo obtidos
65         echo "---Dados Consultados---";
66         echo "<br>";
67         echo "<br>";
68
69
70         $dados_csv[$ntestes-1] = array($ntestes,$numclientes,
    $numpedidosporcliente,$totaltime);
71
72
73         echo ($totaltime * 1000) . ' ms'; // in millseconds
74         $ntestes = $ntestes + 1;
75         echo "<br>";
76        }#FIM LAÇO TESTE
77
78        #MOSTRA INFORMAÇÕES INSERIDAS NA TELA
79   echo "INFORMAÇÕES CSV: <br />";
80   foreach ($dados_csv as $mostra) {
81     print_r($mostra);
```

```
82      echo "<br />";
83     }
84
85    echo "<br />";
86    echo "<br />";
87
88     $csv = fopen($filenamecsv, "a"); //Parâmetro 'a' coloca cursor no final
       do arquivo
89
90     foreach ($dados_csv as $linha) {
91        fputcsv($csv, $linha);
92     }
93
94      echo "Arquivo .csv de resultados gerado!!<br />";
95      fclose($csv);
96
97
98      echo "tudo feito com sucesso";
99      echo "<br>";
100
101 }
102 ?>
```

ANEXO Q – ALGORITMO DE CONSULTA - PRODUTOS ALEATÓRIOS - BASE NÃO-RELACIONAL DE DUAS COLEÇÕES

```
<?php

if(isset($_POST["Consultar"])){



   set_time_limit(30000); //Aumentar tempo limite para 500min

  #CONEXÃO COM BASE DE DADOS
   include ("conexao.php");
   echo "Connection to database successfully";
   echo "<br/>";

    $numclientes = $_POST['numclientes'];
    $numpedidosporcliente = $_POST['numpedidosporcliente'];
    $dados_csv = array();#INFORMAÇÕES PARA SALVAR NO CSV
    $filenamecsv = $numclientes.'clientes'.$numpedidosporcliente.'
   pedidosporcliente'.'.csv';

    $csv = fopen($filenamecsv, "a");
    $head = array("Exec","NumCli","PedporCli","Time");
    fputcsv($csv, $head);
    fclose($csv);


    // select a database and collection
    //$db = $m->ecommerce2;
    $db = $conexao->baseteste;
    $collection1 = $db->shopping;
    $collection2 = $db->produto;

    //produtos sorteados previamente, de acordo com os possíveis números de
     produtos
    $produtos = array
   (172,493,858,2,41,849,747,752,417,76,269,408,439,463,633,80,336,623,400,
    948,935,301,316,613,630,252,970,913,214,810);


    //criacao de indice na base de dados utilizando codigo do produto
    //$collection1->ensureIndex(array('CodigoProduto' => 1));
    //$collection2->ensureIndex(array('_id' => 1));

    $collection1->createIndex(array('Codigocliente' => 1));
    $collection2->createIndex(array('_id' => 1));

```

```
43      $ntestes = 1;
44
45      while($ntestes<=30){
46
47          $totaltime = 0;
48
49          $produto = $produtos[$ntestes-1];
50
51          $query = array(array('$unwind' => '$Pedidos'),
52                         array('$unwind' => '$Pedidos.Itens'),
53                         array('$match' => array('Pedidos.Itens.CodigoProduto'
     => $produto)),
54                         array('$project' => array('Codigocliente' => 1, '
    Pedidos.Itens.Codigoitem' => 1,
55                                                   'Pedidos.Itens.
    CodigoProduto' => 1,
56                                                   'Pedidos.Itens.PrecoUnit'
    => 1,
57                                                   'Pedidos.Itens.Quantidade'
    => 1, 'Pedidos.Itens.Total' => 1)));
58
59          $timebeforequery=microtime(true);
60          $selectdocument1 = $collection1->aggregate($query);
61          $timequery = microtime(true)-$timebeforequery;
62          $totaltime = $totaltime+$timequery;
63
64
65          $query = array('_id' => $produto);
66          $timebeforequery = microtime(true);
67          $selectdocument2 = $collection2->find($query);
68          $timequery = microtime(true)-$timebeforequery;
69          $totaltime = $totaltime+$timequery;
70
71          echo "<br>";
72          echo "Test of produto  " . $produto;
73          echo "<br>";
74
75          $dados_csv[$ntestes-1] = array($ntestes,$numclientes,
    $numpedidosporcliente,$totaltime);
76
77
78          echo "Time to select data (in miliseconds)";
79          echo "<br>";
80
81
82          echo ($totaltime * 1000) . ' ms'; // in millseconds
83          $ntestes = $ntestes + 1;
```

```
84          echo "<br>";
85          echo "<br/>";
86          echo "<br/>";
87       }#FIM LAÇO TESTE
88
89       echo "INFORMAÇÕES CSV: <br />";
90      foreach ($dados_csv as $mostra) {
91         print_r($mostra);
92         echo "<br />";
93      }
94      echo "<br />";
95      echo "<br />";
96
97
98      $csv = fopen($filenamecsv, "a"); //Parâmetro 'a' coloca cursor no final
        do arquivo
99
100     foreach ($dados_csv as $linha) {
101        fputcsv($csv, $linha);
102     }
103
104
105      echo "Arquivo .csv de resultados gerado!!<br />";
106      fclose($csv);
107
108
109      echo "<br>";
110      echo "tudo feito com sucesso";
111
112 }
113 ?>
```

ANEXO R – ALGORITMO DE ALTERAÇÃO DE DADOS DO CLIENTE - BASE RELACIONAL

```php
<?php

set_time_limit(10000000); //Aumentar tempo limite para 5000min

if(isset($_POST["Alterar"])){
  $host = 'localhost';
  $user = 'root';
  $db = 'ecommerce';
  $pass = '';
  $connection = mysqli_connect($host,$user,$pass,$db) or die(mysqli_error()
   );
  $numclientes = $_POST['numclientes'];
  $numpedidos = $_POST['numpedidosporcliente'];
  $dados_consulta = array(); //lista dos tempos de consulta do teste


  //clientes sorteados previamente, de acordo com os possíveis números de
   clientes
  $clientes100 = array
   (21,82,51,19,92,57,100,37,76,30,93,99,65,85,74,94,64,22,43,1,61,45,78,
  31,54,75,97,80,60,98);
  $clientes1000 = array
   (612,948,498,364,788,732,161,843,469,957,927,296,9,945,666,37,600,676,
  272,998,890,671,401,797,167,897,882,502,229,405);
  $clientes10000 = array
   (4968,9626,4054,5277,847,8524,8140,3572,2307,3692,1516,1356,5853,7264,
  2242,3207,3491,7551,8407,7265,9298,7553,2377,1657,
  2286,7105,7513,747,1717,6428);
  $clientes100000 = array
   (63869,8975,53955,8842,32484,15257,95199,52436,29417,85382,27058,42960,
  38275,91166,70567,6820,72572,95891,70099,60802,76271,82415,
  63576,57858,1644,22381,14805,10948,78940,89891);
  $clientes1000000 = array
   (695851,955745,222668,573136,235105,851612,601466,615710,447332,775822,
  326472,381625,330213,858213,708703,850373,661231,570709,174143,66100,
  211463,474710,657869,819024,326139,972229,620283,745046,644613,971838);


  //Ativando os índices da base de dados
  if ($stmt = mysqli_prepare($connection, "call sp_constroi_indices()")){

    mysqli_stmt_execute($stmt);
    mysqli_stmt_close($stmt);
  }

```

```
39
40  //escolha de qual vetor de clientes utilizar
41  switch ($numclientes) {
42        case 100:
43          $vetorclientes = $clientes100;
44          break;
45        case 1000:
46          $vetorclientes = $clientes1000;
47          break;
48        case 10000:
49          $vetorclientes = $clientes10000;
50          break;
51        case 100000;
52          $vetorclientes = $clientes100000;
53          break;
54        case 1000000;
55          $vetorclientes = $clientes1000000;
56          break;
57        default:
58          $vetorclientes = $clientes100;
59  }
60
61
62  $ntestes = 1;
63
64  while($ntestes<=30){
65
66    $totaltime = 0;
67
68    $novoendereco = 'novo endereco do cliente';
69
70        $timebeforequery=microtime(true);
71        $query = "update cliente set endereco = '$novoendereco' where
     cod_cliente='$cliente'";
72        $selecttable = mysqli_query($connection,$query) or die(mysql_error
     ());
73        $timequery = microtime(true)-$timebeforequery;
74        $totaltime = $totaltime+$timequery;
75
76    $dados_consulta[$ntestes-1]= array ($ntestes,$totaltime);//salvando
     tempo da iteração do teste no array
77     echo "<br>";
78
79    echo "Test of client  " . $cliente;
80
81    //só pra testar se os dados estavam mesmo sendo obtidos
82    //$row = mysqli_fetch_array($selecttable, MYSQLI_NUM);
```

```
83      //printf ("%s (%s)\n", $row[0], $row[1]);
84
85
86      echo "<br>";
87      echo "Time to delete data (in miliseconds)";
88      echo "<br>";
89
90
91      echo ($totaltime * 1000) . ' ms'; // in millseconds
92      $ntestes = $ntestes + 1;
93      echo "<br>";
94    }
95
96  echo "INFORMAÇÕES SALVAS: <br />";
97  foreach ($dados_consulta as $mostra) {
98    print_r($mostra);
99    echo "<br />";
100 }
101
102 echo "<br>";
103 echo "tudo feito com sucesso";
104   }
105
106 ?>
```

ANEXO S – ALGORITMO DE ALTERAÇÃO DE DADOS DO CLIENTE - BASE NÃO-RELACIONAL DE UMA COLEÇÃO

```
<?php

set_time_limit(10000000); //Aumentar tempo limite para 5000min
include ("conexao.php");

echo "Connection to database successfully";
echo "<br>";
echo "<br>";

if(isset($_POST["Alterar"])){

  $numclientes = $_POST['numclientes'];
  $numpedidos = $_POST['numpedidosporcliente'];

  // select a database and collection
  $db = $conexao->ecommerce;
  $collection = $db->shopping;


  $dados_consulta = array(); //lista dos tempos de consulta do teste
  $nome_arquivo = $numclientes.'clientes'.$numpedidos.'pedidos'.'.csv'; //
    definição do nome do arquivo

  $csv = fopen($nome_arquivo, "a");
  $head = array("Exec","Tempo");
  fputcsv($csv, $head);
  fclose($csv);



  //clientes sorteados previamente, de acordo com os possíveis números de
    clientes
  $clientes100 = array
    (21,82,51,19,92,57,100,37,76,30,93,99,65,85,74,94,64,22,43,1,61,45,78,
  31,54,75,97,80,60,98);
  $clientes1000 = array
    (612,948,498,364,788,732,161,843,469,957,927,296,9,945,666,37,600,676,
  272,998,890,671,401,797,167,897,882,502,229,405);
  $clientes10000 = array
    (4968,9626,4054,5277,847,8524,8140,3572,2307,3692,1516,1356,5853,7264,
  2242,3207,3491,7551,8407,7265,9298,7553,2377,1657,
  2286,7105,7513,747,1717,6428);
  $clientes100000 = array
    (63869,8975,53955,8842,32484,15257,95199,52436,29417,85382,27058,42960,
  38275,91166,70567,6820,72572,95891,70099,60802,76271,82415,
```

```
63576,57858,1644,22381,14805,10948,78940,89891);
$clientes1000000 = array
 (695851,955745,222668,573136,235105,851612,601466,615710,447332,775822,
326472,381625,330213,858213,708703,850373,661231,570709,174143,66100,
211463,474710,657869,819024,326139,972229,620283,745046,644613,971838);


 //criacao de indice na base de dados
 $collection->createIndex(array('Codigocliente' => 1));


//escolha de qual vetor de clientes utilizar
switch ($numclientes) {
      case 100:
        $vetorclientes = $clientes100;
        break;
      case 1000:
        $vetorclientes = $clientes1000;
        break;
      case 10000:
        $vetorclientes = $clientes10000;
        break;
      case 100000;
        $vetorclientes = $clientes100000;
        break;
      case 1000000;
        $vetorclientes = $clientes1000000;
        break;
      default:
        $vetorclientes = $clientes100;
}


$ntestes = 1;

while($ntestes<=30){

  $totaltime = 0;

  $cliente = $vetorclientes[$ntestes-1];

  $mongo = new MongoDB\Driver\Manager("mongodb://localhost:27017");
  $bulk = new MongoDB\Driver\BulkWrite;
  $bulk->update(
    ['Codigocliente' => $cliente],
    ['$set' => ["Endereco" => "novo endereco do cliente"]]);

```

```
86
87      $timebeforequery=microtime(true);
88      $mongo->executeBulkWrite('ecommerce.shopping', $bulk);
89      $timequery = microtime(true)-$timebeforequery;
90      $totaltime = $totaltime+$timequery;
91      $dados_consulta[$ntestes-1]= array ($ntestes,$numclientes,$numpedidos,
        $totaltime);//salvando tempo da iteração do teste no array
92      echo "<br>";
93
94      echo "Test of client  " . $cliente;
95
96
97      echo "<br>";
98      echo "Time to select data (in miliseconds)";
99      echo "<br>";
100
101
102     echo ($totaltime * 1000) . ' ms'; // in millseconds
103     $ntestes = $ntestes + 1;
104     echo "<br>";
105   }#FIM DO LAÇO TESTE
106
107 echo "INFORMAÇÕES SALVAS: <br />";
108 foreach ($dados_consulta as $mostra) {
109   print_r($mostra);
110   echo "<br />";
111 }
112
113
114   $csv = fopen($nome_arquivo, "a"); //Parâmetro 'a' coloca cursor no final
      do arquivo
115                                     //$csv é um controlador de fluxo para o
      arquivo
116   //echo "DADOS SALVOS NO ARRAY";
117   foreach ($dados_consulta as $linha) {
118     fputcsv($csv, $linha);
119     }
120 echo "Arquivo .csv de resultados gerado!!<br />";
121 fclose($csv);
122
123
124
125 echo "<br>";
126 echo "tudo feito com sucesso";
127   }
128
129 ?>
```

ANEXO T – ALGORITMO DE ALTERAÇÃO DE DADOS DO CLIENTE - BASE NÃO-RELACIONAL DE DUAS COLEÇÕES

```
<?php

set_time_limit(10000000); //Aumentar tempo limite para 5000min
include ("conexao.php");

echo "Connection to database successfully";
echo "<br>";
echo "<br>";

if(isset($_POST["Alterar"])){

  $numclientes = $_POST['numclientes'];
  $numpedidos = $_POST['numpedidosporcliente'];

  // select a database and collection
  $db = $conexao->ecommerce;
  $collection = $db->shopping;


  $dados_consulta = array(); //lista dos tempos de consulta do teste
  $nome_arquivo = $numclientes.'clientes'.$numpedidos.'pedidos'.'.csv'; //
    definição do nome do arquivo

  $csv = fopen($nome_arquivo, "a");
  $head = array("Exec","Tempo");
  fputcsv($csv, $head);
  fclose($csv);



  //clientes sorteados previamente, de acordo com os possíveis números de
    clientes
  $clientes100 = array
    (21,82,51,19,92,57,100,37,76,30,93,99,65,85,74,94,64,22,43,1,61,45,78,
  31,54,75,97,80,60,98);
  $clientes1000 = array
    (612,948,498,364,788,732,161,843,469,957,927,296,9,945,666,37,600,676,
  272,998,890,671,401,797,167,897,882,502,229,405);
  $clientes10000 = array
    (4968,9626,4054,5277,847,8524,8140,3572,2307,3692,1516,1356,5853,7264,
  2242,3207,3491,7551,8407,7265,9298,7553,2377,1657,
  2286,7105,7513,747,1717,6428);
  $clientes100000 = array
    (63869,8975,53955,8842,32484,15257,95199,52436,29417,85382,27058,42960,
  38275,91166,70567,6820,72572,95891,70099,60802,76271,82415,
```

```
63576,57858,1644,22381,14805,10948,78940,89891);
$clientes1000000 = array
 (695851,955745,222668,573136,235105,851612,601466,615710,447332,775822,
326472,381625,330213,858213,708703,850373,661231,570709,174143,66100,
211463,474710,657869,819024,326139,972229,620283,745046,644613,971838);


 //criacao de indice na base de dados
 $collection->createIndex(array('Codigocliente' => 1));


//escolha de qual vetor de clientes utilizar
switch ($numclientes) {
      case 100:
        $vetorclientes = $clientes100;
        break;
      case 1000:
        $vetorclientes = $clientes1000;
        break;
      case 10000:
        $vetorclientes = $clientes10000;
        break;
      case 100000;
        $vetorclientes = $clientes100000;
        break;
      case 1000000;
        $vetorclientes = $clientes1000000;
        break;
      default:
        $vetorclientes = $clientes100;
}


$ntestes = 1;

while($ntestes<=30){

  $totaltime = 0;

  $cliente = $vetorclientes[$ntestes-1];

  $mongo = new MongoDB\Driver\Manager("mongodb://localhost:27017");
  $bulk = new MongoDB\Driver\BulkWrite;
  $bulk->update(
    ['Codigocliente' => $cliente],
    ['$set' => ["Endereco" => "novo endereco do cliente"]]);

```

```
86
87      $timebeforequery=microtime(true);
88      $mongo->executeBulkWrite('ecommerce.shopping', $bulk);
89      $timequery = microtime(true)-$timebeforequery;
90      $totaltime = $totaltime+$timequery;
91
92      $dados_consulta[$ntestes-1]= array ($ntestes,$numclientes,$numpedidos,
        $totaltime);//salvando tempo da iteração do teste no array
93      echo "<br>";
94
95      echo "Test of client  " . $cliente;
96
97
98      echo "<br>";
99      echo "Time to select data (in miliseconds)";
100     echo "<br>";
101
102
103     echo ($totaltime * 1000) . ' ms'; // in millseconds
104     $ntestes = $ntestes + 1;
105     echo "<br>";
106   }#FIM DO LAÇO TESTE
107
108 echo "INFORMAÇÕES SALVAS: <br />";
109 foreach ($dados_consulta as $mostra) {
110   print_r($mostra);
111   echo "<br />";
112 }
113
114
115   $csv = fopen($nome_arquivo, "a"); //Parâmetro 'a' coloca cursor no final
      do arquivo
116                                     //$csv é um controlador de fluxo para o
      arquivo
117   //echo "DADOS SALVOS NO ARRAY";
118   foreach ($dados_consulta as $linha) {
119     fputcsv($csv, $linha);
120     }
121 echo "Arquivo .csv de resultados gerado!!<br />";
122 fclose($csv);
123
124 echo "<br>";
125 echo "tudo feito com sucesso";
126   }
127
128 ?>
```

ANEXO U – ALGORITMO DE REMOÇÃO DO CLIENTE - BASE RELACIONAL

```php
<?php

set_time_limit(10000000); //Aumentar tempo limite para 5000min

if(isset($_POST["Remover"])){
  $host = 'localhost';
  $user = 'root';
  $db = 'ecommerce';
  $pass = '';
  $connection = mysqli_connect($host,$user,$pass,$db) or die(mysqli_error()
   );
  $numclientes = $_POST['numclientes'];
  $numpedidos = $_POST['numpedidosporcliente'];
  $dados_consulta = array(); //lista dos tempos de consulta do teste

  $nome_arquivo = $numclientes.'clientes'.$numpedidos.'pedidos'.'.csv'; //
   definição do nome do arquivo
  $csv = fopen($nome_arquivo, "a");
  $head = array("Exec","Tempo");
  fputcsv($csv, $head);
  fclose($csv);




  //clientes sorteados previamente, de acordo com os possíveis números de
   clientes
  $clientes100 = array
   (21,82,51,19,92,57,100,37,76,30,93,99,65,85,74,94,64,22,43,1,61,45,78,
  31,54,75,97,80,60,98);
  $clientes1000 = array
   (612,948,498,364,788,732,161,843,469,957,927,296,9,945,666,37,600,676,
  272,998,890,671,401,797,167,897,882,502,229,405);
  $clientes10000 = array
   (4968,9626,4054,5277,847,8524,8140,3572,2307,3692,1516,1356,5853,7264,
  2242,3207,3491,7551,8407,7265,9298,7553,2377,1657,
  2286,7105,7513,747,1717,6428);
  $clientes100000 = array
   (63869,8975,53955,8842,32484,15257,95199,52436,29417,85382,27058,42960,
  38275,91166,70567,6820,72572,95891,70099,60802,76271,82415,
  63576,57858,1644,22381,14805,10948,78940,89891);
  $clientes1000000 = array
   (695851,955745,222668,573136,235105,851612,601466,615710,447332,775822,
  326472,381625,330213,858213,708703,850373,661231,570709,174143,66100,
  211463,474710,657869,819024,326139,972229,620283,745046,644613,971838);

```

```
39
40  //Ativando os índices da base de dados
41  if ($stmt = mysqli_prepare($connection, "call sp_constroi_indices()")){
42
43    mysqli_stmt_execute($stmt);
44    mysqli_stmt_close($stmt);
45  }
46
47
48  //escolha de qual vetor de clientes utilizar
49  switch ($numclientes) {
50        case 100:
51          $vetorclientes = $clientes100;
52          break;
53        case 1000:
54          $vetorclientes = $clientes1000;
55          break;
56        case 10000:
57          $vetorclientes = $clientes10000;
58          break;
59        case 100000;
60          $vetorclientes = $clientes100000;
61          break;
62        case 1000000;
63          $vetorclientes = $clientes1000000;
64          break;
65        default:
66          $vetorclientes = $clientes100;
67  }
68
69  $ntestes = 1;
70
71  while($ntestes<=30){
72
73    $totaltime = 0;
74
75    $cliente = $vetorclientes[$ntestes-1];
76
77    $timebeforequery=microtime(true);
78    $query1 = "delete from item_pedido where cod_pedido in (select
     cod_pedido from pedido where cod_cliente = '$cliente')";
79    $selecttable = mysqli_query($connection,$query1) or die(mysqli_error(
     $connection));
80  //$selecttable = mysqli_query($connection,$query1) or die(mysql_error());
81    $timequery = microtime(true)-$timebeforequery;
82    $totaltime = $totaltime+$timequery;
83
```

```
84
85      $timebeforequery=microtime(true);
86      $query2 = "delete from pedido where cod_cliente = '$cliente'";
87      $selecttable = mysqli_query($connection,$query2) or die(mysqli_error(
        $connection));
88      $timequery = microtime(true)-$timebeforequery;
89      $totaltime = $totaltime+$timequery;
90
91
92      $timebeforequery=microtime(true);
93      $query3 = "delete from telefone_cliente where cod_cliente = '$cliente'"
        ;
94      $selecttable = mysqli_query($connection,$query3) or die(mysqli_error(
        $connection));
95      $timequery = microtime(true)-$timebeforequery;
96      $totaltime = $totaltime+$timequery;
97
98
99      $timebeforequery=microtime(true);
100     $query4 = "delete from email_cliente where cod_cliente = '$cliente'";
101     $selecttable = mysqli_query($connection,$query4) or die(mysqli_error(
        $connection));
102     $timequery = microtime(true)-$timebeforequery;
103     $totaltime = $totaltime+$timequery;
104
105     $timebeforequery=microtime(true);
106     $query5 = "delete from cliente where cod_cliente = '$cliente'";
107     $selecttable = mysqli_query($connection,$query5) or die(mysqli_error(
        $connection));
108     $timequery = microtime(true)-$timebeforequery;
109     $totaltime = $totaltime+$timequery;
110
111     $dados_consulta[$ntestes-1]= array ($ntestes,$totaltime);//salvando
        tempo da iteração do teste no array
112     echo "<br>";
113
114     echo "Test of client  " . $cliente;
115
116
117     echo "<br>";
118     echo "Time to delete data (in miliseconds)";
119     echo "<br>";
120
121
122     echo ($totaltime * 1000) . ' ms'; // in millseconds
123     $ntestes = $ntestes + 1;
124     echo "<br>";
```

```
125   }
126
127 echo "INFORMAÇÕES SALVAS: <br />";
128 foreach ($dados_consulta as $mostra) {
129   print_r($mostra);
130   echo "<br />";
131 }
132
133
134   $csv = fopen($nome_arquivo, "a"); //Parâmetro 'a' coloca cursor no final
      do arquivo
135                                     //$csv é um controlador de fluxo para o
      arquivo
136   //echo "DADOS SALVOS NO ARRAY";
137   foreach ($dados_consulta as $linha) {
138     fputcsv($csv, $linha);
139     }
140 echo "Arquivo .csv de resultados gerado!!<br />";
141 fclose($csv);
142
143 echo "<br>";
144 echo "tudo feito com sucesso";
145   }
146
147 ?>
```

ANEXO V – ALGORITMO DE REMOÇÃO DO CLIENTE - BASE NÃO-RELACIONAL DE UMA COLEÇÃO

```
<?php

set_time_limit(10000000); //Aumentar tempo limite para 5000min
include ("conexao.php");

echo "Connection to database successfully";
echo "<br>";
echo "<br>";

if(isset($_POST["Remover"])){

  $numclientes = $_POST['numclientes'];
  $numpedidos = $_POST['numpedidosporcliente'];

  // select a database and collection
  $db = $conexao->ecommerce;
  $collection = $db->shopping;


  $dados_consulta = array(); //lista dos tempos de consulta do teste
  $nome_arquivo = $numclientes.'clientes'.$numpedidos.'pedidos'.'.csv'; //
    definição do nome do arquivo

  $csv = fopen($nome_arquivo, "a");
  $head = array("Exec","Tempo");
  fputcsv($csv, $head);
  fclose($csv);



  //clientes sorteados previamente, de acordo com os possíveis números de
    clientes
  $clientes100 = array
    (21,82,51,19,92,57,100,37,76,30,93,99,65,85,74,94,64,22,43,1,61,45,78,
  31,54,75,97,80,60,98);
  $clientes1000 = array
    (612,948,498,364,788,732,161,843,469,957,927,296,9,945,666,37,600,676,
  272,998,890,671,401,797,167,897,882,502,229,405);
  $clientes10000 = array
    (4968,9626,4054,5277,847,8524,8140,3572,2307,3692,1516,1356,5853,7264,
  2242,3207,3491,7551,8407,7265,9298,7553,2377,1657,
  2286,7105,7513,747,1717,6428);
  $clientes100000 = array
    (63869,8975,53955,8842,32484,15257,95199,52436,29417,85382,27058,42960,
  38275,91166,70567,6820,72572,95891,70099,60802,76271,82415,
```

```
63576,57858,1644,22381,14805,10948,78940,89891);
$clientes1000000 = array
 (695851,955745,222668,573136,235105,851612,601466,615710,447332,775822,
326472,381625,330213,858213,708703,850373,661231,570709,174143,66100,
211463,474710,657869,819024,326139,972229,620283,745046,644613,971838);


 //criacao de indice na base de dados
 $collection->createIndex(array('Codigocliente' => 1));


//escolha de qual vetor de clientes utilizar
switch ($numclientes) {
      case 100:
        $vetorclientes = $clientes100;
        break;
      case 1000:
        $vetorclientes = $clientes1000;
        break;
      case 10000:
        $vetorclientes = $clientes10000;
        break;
      case 100000;
        $vetorclientes = $clientes100000;
        break;
      case 1000000;
        $vetorclientes = $clientes1000000;
        break;
      default:
        $vetorclientes = $clientes100;
}


$ntestes = 1;

while($ntestes<=30){

  $totaltime = 0;

  $cliente = $vetorclientes[$ntestes-1];

  $timebeforequery=microtime(true);
  $query = array('Codigocliente' => $cliente);
  $deletedocument = $collection->deleteOne($query);
  $timequery = microtime(true)-$timebeforequery;
  $totaltime = $totaltime+$timequery;

```

```
86
87      $dados_consulta[$ntestes-1]= array ($ntestes,$numclientes,$numpedidos,
        $totaltime);//salvando tempo da iteração do teste no array
88      echo "<br>";
89
90
91      echo "Test of client  " . $cliente;
92
93
94      echo "<br>";
95      echo "Time to select data (in miliseconds)";
96      echo "<br>";
97
98
99      echo ($totaltime * 1000) . ' ms'; // in millseconds
100     $ntestes = $ntestes + 1;
101     echo "<br>";
102   }#FIM DO LAÇO TESTE
103
104 echo "INFORMAÇÕES SALVAS: <br />";
105 foreach ($dados_consulta as $mostra) {
106   print_r($mostra);
107   echo "<br />";
108 }
109
110
111   $csv = fopen($nome_arquivo, "a"); //Parâmetro 'a' coloca cursor no final
      do arquivo
112                                     //$csv é um controlador de fluxo para o
      arquivo
113   //echo "DADOS SALVOS NO ARRAY";
114   foreach ($dados_consulta as $linha) {
115     fputcsv($csv, $linha);
116     }
117 echo "Arquivo .csv de resultados gerado!!<br />";
118 fclose($csv);
119
120
121
122 echo "<br>";
123 echo "tudo feito com sucesso";
124   }
125
126 ?>
```

ANEXO W – ALGORITMO DE REMOÇÃO DO CLIENTE - BASE NÃO-RELACIONAL DE DUAS COLEÇÕES

```
<?php

set_time_limit(10000000); //Aumentar tempo limite para 5000min
include ("conexao.php");

echo "Connection to database successfully";
echo "<br>";
echo "<br>";

if(isset($_POST["Remover"])){

  $numclientes = $_POST['numclientes'];
  $numpedidos = $_POST['numpedidosporcliente'];

  // select a database and collection
  $db = $conexao->ecommerce;
  $collection = $db->shopping;


  $dados_consulta = array(); //lista dos tempos de consulta do teste
  $nome_arquivo = $numclientes.'clientes'.$numpedidos.'pedidos'.'.csv'; //
    definição do nome do arquivo

  $csv = fopen($nome_arquivo, "a");
  $head = array("Exec","Tempo");
  fputcsv($csv, $head);
  fclose($csv);



  //clientes sorteados previamente, de acordo com os possíveis números de
    clientes
  $clientes100 = array
    (21,82,51,19,92,57,100,37,76,30,93,99,65,85,74,94,64,22,43,1,61,45,78,
  31,54,75,97,80,60,98);
  $clientes1000 = array
    (612,948,498,364,788,732,161,843,469,957,927,296,9,945,666,37,600,676,
  272,998,890,671,401,797,167,897,882,502,229,405);
  $clientes10000 = array
    (4968,9626,4054,5277,847,8524,8140,3572,2307,3692,1516,1356,5853,7264,
  2242,3207,3491,7551,8407,7265,9298,7553,2377,1657,
  2286,7105,7513,747,1717,6428);
  $clientes100000 = array
    (63869,8975,53955,8842,32484,15257,95199,52436,29417,85382,27058,42960,
  38275,91166,70567,6820,72572,95891,70099,60802,76271,82415,
```

```
63576,57858,1644,22381,14805,10948,78940,89891);
$clientes1000000 = array
 (695851,955745,222668,573136,235105,851612,601466,615710,447332,775822,
326472,381625,330213,858213,708703,850373,661231,570709,174143,66100,
211463,474710,657869,819024,326139,972229,620283,745046,644613,971838);


 //criacao de indice na base de dados
 $collection->createIndex(array('Codigocliente' => 1));


//escolha de qual vetor de clientes utilizar
switch ($numclientes) {
      case 100:
        $vetorclientes = $clientes100;
        break;
      case 1000:
        $vetorclientes = $clientes1000;
        break;
      case 10000:
        $vetorclientes = $clientes10000;
        break;
      case 100000;
        $vetorclientes = $clientes100000;
        break;
      case 1000000;
        $vetorclientes = $clientes1000000;
        break;
      default:
        $vetorclientes = $clientes100;
}


$ntestes = 1;

while($ntestes<=30){

  $totaltime = 0;

  $cliente = $vetorclientes[$ntestes-1];

  $timebeforequery=microtime(true);
  $query = array('Codigocliente' => $cliente);
  $deletedocument = $collection->deleteOne($query);
  $timequery = microtime(true)-$timebeforequery;
  $totaltime = $totaltime+$timequery;

```

```
86
87      $dados_consulta[$ntestes-1]= array ($ntestes,$numclientes,$numpedidos,
        $totaltime);//salvando tempo da iteração do teste no array
88      echo "<br>";
89
90
91      echo "Test of client  " . $cliente;
92
93
94      echo "<br>";
95      echo "Time to select data (in miliseconds)";
96      echo "<br>";
97
98
99      echo ($totaltime * 1000) . ' ms'; // in millseconds
100     $ntestes = $ntestes + 1;
101     echo "<br>";
102   }#FIM DO LAÇO TESTE
103
104 echo "INFORMAÇÕES SALVAS: <br />";
105 foreach ($dados_consulta as $mostra) {
106   print_r($mostra);
107   echo "<br />";
108 }
109
110
111   $csv = fopen($nome_arquivo, "a"); //Parâmetro 'a' coloca cursor no final
      do arquivo
112                                     //$csv é um controlador de fluxo para o
      arquivo
113   //echo "DADOS SALVOS NO ARRAY";
114   foreach ($dados_consulta as $linha) {
115     fputcsv($csv, $linha);
116     }
117 echo "Arquivo .csv de resultados gerado!!<br />";
118 fclose($csv);
119
120
121 echo "<br>";
122 echo "tudo feito com sucesso";
123   }
124
125 ?>
```

UFVJM