UNIVERSIDADE FEDERAL DO CEARÁ

CAMPUS DE QUIXADÁ

PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO

MESTRADO ACADÊMICO EM COMPUTAÇÃO

FRANCISCA LUZIA NOGUEIRA ARAÚJO

MODELO DE ARQUITETURA PARA USO DE BANCO DE DADOS HÍBRIDOS ADAPTÁVEL A FERRAMENTAS DE PROCESSAMENTO DE LINGUAGEM NATURAL(PLN)

QUIXADÁ

2024

FRANCISCA LUZIA NOGUEIRA ARAÚJO

MODELO DE ARQUITETURA PARA USO DE BANCO DE DADOS HÍBRIDOS ADAPTÁVEL A FERRAMENTAS DE PROCESSAMENTO DE LINGUAGEM NATURAL(PLN)

Dissertação apresentada ao Curso de Mestrado Acadêmico em Computação do Programa de Pós-Graduação em Computação do Campus de Quixadá da Universidade Federal do Ceará, como requisito parcial à obtenção do título de mestre em computação. Área de Concentração: Ciência da Computação

Orientador: Prof. Dr. Gabriel A. L. Paillard

Coorientador: Prof. Dr. Leonardo Oliveira Moreira

QUIXADÁ

2024

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Sistema de Bibliotecas
Gerada automaticamente pelo módulo Catalog, mediante os dados fornecidos pelo(a) autor(a)

A689m Araújo, Francisca Luzia Nogueira.
Modelo de arquitetura para uso de banco de dados híbridos adaptável a ferramentas de Processamento de Linguagem Natural(PLN) / Francisca Luzia Nogueira Araújo. – 2024.
66 f. : il. color.

Dissertação (mestrado) – Universidade Federal do Ceará, Campus de Quixadá, Programa de Pós-Graduação em Computação, Quixadá, 2024.
Orientação: Prof. Dr. Gabriel Antoine Louis Paillard.
Coorientação: Prof. Dr. Leonardo Oliveira Moreira.

1. Banco de dados. 2. Arquitetura Híbrida. 3. Processamento de Linguagem Natural. I. Título.
CDD 005

FRANCISCA LUZIA NOGUEIRA ARAÚJO

MODELO DE ARQUITETURA PARA USO DE BANCO DE DADOS HÍBRIDOS ADAPTÁVEL A FERRAMENTAS DE PROCESSAMENTO DE LINGUAGEM NATURAL(PLN)

Dissertação apresentada ao Curso de Mestrado Acadêmico em Computação do Programa de Pós-Graduação em Computação do Campus de Quixadá da Universidade Federal do Ceará, como requisito parcial à obtenção do título de mestre em computação. Área de Concentração: Ciência da Computação

Aprovada em:

BANCA EXAMINADORA

Prof. Dr. Gabriel A. L. Paillard (Orientador)
Universidade Federal do Ceará (UFC)

Prof. Dr. Leonardo Oliveira Moreira (Coorientador)
Universidade Federal do Ceará (UFC)

Profa. Dra. Ticiana Linhares Coelho da Silva
Universidade Federal do Ceará (UFC)

Profa. Dra. Lívia Almada Cruz
Universidade Federal do Ceará (UFC)

Prof. Dr. Francisco Milton Mendes Neto
Universidade Federal Rural do Semi-Árido(UFERSA)

A Deus.

AGRADECIMENTOS

Agradeço à minha irmã mais velha, Francisca Jaqueline Nogueira Araújo, por todo o material didático, lanches, fardas e sapatos comprados; à minha irmã do meio, Francisca Jaquelane Nogueira Araújo, por cada tarefa ensinada, matrícula escolar realizada e por ser meu exemplo de dedicação e esforço na busca dos meus objetivos; à minha prima, Maria Suelen de Oliveira Nogueira, minha madrinha de graduação, por todo o ensaio de apresentação assistido e seu apoio emocional; ao meu namorado, Marcos Paulo Fernandes de Moura, por todo o seu companheirismo, sua paciência, apoio e ajuda principalmente nesses longos anos dessa jornada acadêmica; ao meu Orientador, Prof. Dr. Gabriel A. L. Paillard; ao meu coorientador, Prof. Dr. Leonardo Oliveira Moreira; às professoras, Dra. Ticiana Linhares Coelho da Silva e Dra. Lívia Almada Cruz; e ao Prof. Dr. João Marcelo Uchôa de Alencar, por toda a orientação, apoio e incentivo.

“Educação não transforma o mundo, educação muda pessoas. Pessoas transformam o mundo!.”

(Paulo Freire)

RESUMO

Atualmente, os bancos de dados se tornaram onipresentes. Quase todos os aplicativos de Tecnologia da Informação (TI) estão armazenando e recuperando informações de Banco de Dados (BD). O armazenamento de dados de tipos diferentes é um grande desafio, podendo ser necessário utilizar mais de um tipo de banco de dados, tornando mais complexa a obtenção de informações sobre esses dados. Além disso, a dinâmica das organizações modernas frequentemente lidam com a necessidade de conciliar requisitos opostos, fornecidos por bancos de dados de diferentes tipos, como, por exemplo, bancos de dados relacionais (ou SQL) e não-relacionais (ou NoSQL). Portanto, usuários não especialistas que necessitem interagir com dados heterogêneos carecem de um meio pelo qual possam acessar os bancos de dados de maneira transparente. Por outro lado, o Processamento de Linguagem Natural ou PLN permite a comunicação entre pessoas e máquinas através de técnicas que possibilitam a interpretação da linguagem natural empregada pelo homem por meio de um dispositivo computacional. Este trabalho apresenta um modelo de arquitetura de sistema adaptável a ferramentas de PLN capazes de traduzir consultas em linguagem natural para linguagem formal de bancos de dados, e após a tradução, permitir executar as consultas em bases de dados armazenados em bancos de dados híbridos, locais ou distribuídos. Visando permitir adequações a arquitetura aqui proposta, em decorrência da evolução do estado da arte atual, foi projetada para possibilitar adições de novos bancos de dados, novos algoritmos e/ou novas ferramentas de tradução de linguagem natural para linguagem formal de consulta de bancos de dados, além de permitir adaptações para reconhecer novos idiomas nas consultas de entrada. A estratégia utilizada foi a criação de módulos com funcionalidades bem definidas e separadas dos demais, onde para adicionar uma nova ferramenta de tradução à proposta, será necessário que apenas um módulo seja modificado, por exemplo. Para assegurar a adaptabilidade da proposta, o código-fonte foi disponibilizado e testes foram conduzidos em um cluster de computadores, com a possibilidade de implementação também em uma infraestrutura de serviços de computação em nuvem, além disso, os usuários podem realizar ajustes, para também suportar dados do tipo *big data*.

Palavras-chave: Banco de dados. Arquitetura Híbrida. Processamento de Linguagem Natural.

ABSTRACT

Currently, databases have become omnipresent. Almost all IT applications are storing and retrieving information from databases. Storing data of different types is a significant challenge, and it may be necessary to use more than one type of database, making it more complex to obtain information about this data. In addition, the dynamics of modern organizations often deal with the need to reconcile opposing requirements provided by databases of different types, such as relational databases (or SQL) and non-relational databases (or NoSQL). Therefore, non-expert users who need to interact with heterogeneous data lack a means by which they can access databases transparently. On the other hand, Natural Language Processing or NLP enables communication between people and machines through techniques that allow the interpretation of natural language used by humans through a computational device. This paper presents an architecture model of a system adaptable to NLP tools capable of translating queries in natural language to formal database query language and, after translation, allowing the execution of queries on databases stored in hybrid, local, or distributed databases. Aimed at enabling adjustments to the proposed architecture, due to the evolution of the current state of the art, it was designed to enable additions of new databases, new algorithms, and/or new natural language translation tools to formal query language of databases, as well as allowing adaptations to recognize new languages in input queries. The strategy used was the creation of modules with well-defined and separated functionalities from others, where to add a new translation tool to the proposal, only one module needs to be modified, for example. To ensure the adaptability of the proposal, the source code was made available and tests were conducted on a cluster of computers, with the possibility of implementation also in a cloud computing services infrastructure; moreover, users can make adjustments to also support big data.

Keywords: Databases. Hybrid architecture. Natural Language Processing.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE QUADROS

LISTA DE ABREVIATURAS E SIGLAS

TI	Tecnologia da Informação
BD	Banco de Dados
BDs	Bancos de Dados
SQL	*Structured Query Language*
NLDBIs	*Natural Language Database Interfaces*
PLN	Processamento de Linguagem Natural
NoSQL	*Not only Structured Query Language*
LN	Linguagem Natural
DQL	*Data Query Language*
CRUD	*Create, Read, Update, Delete*
OGD	*Open Government Data*
LF	Linguagem Formal

SUMÁRIO

1 INTRODUÇÃO

Segundo Deshpande e Devale (2012), os Bancos de Dados (BDs) se tornaram onipresentes. Quase todos os aplicativos de TI estão armazenando e recuperando informações de BD. Porém, a recuperação de informações de BD requer conhecimento de linguagens formais, como *Structured Query Language* (SQL). No entanto, a maioria dos usuários que interagem com os BDs não tem formação técnica e ficam intimidados com a ideia de usar linguagens como SQL. Couderc e Ferrero (2015a) destacam que alguns cargos que não requerem nenhum treinamento em administração de dados exigem um trabalho próximo a bancos de dados, como contabilidade ou secretariado, por exemplo. Por isso, visando permitir que uma pessoa que não possui competência na área de gerenciamento de banco de dados, consiga executar tarefas simples, como, por exemplo, realizar consultas a uma base de dados armazenada em um banco de dados, algumas técnicas foram desenvolvidas, como *Natural Language Database Interfaces* (NLDBIs) que traduzem consultas de linguagem natural para a linguagem formal de banco de dados.

1.1 Motivação

Segundo Bjeladinovic *et al.* (2020), o *modus operandi* das organizações modernas lidam frequentemente com a necessidade de conciliar requisitos distintos, fornecidos por bancos de dados de diferentes tipos, como, por exemplo, bancos de dados relacionais e não relacionais. Assim, é necessário permitir o acesso a dados com vários níveis de estruturação, independentemente da fonte dos dados, bem como simultaneamente garantir a integridade com disponibilidade máxima em um sistema, ajudando a justificar uso simultâneo de tipos de bancos de dados diferentes devido à capacidade de manter a eficiência enquanto são usados para seus respectivos fins.

Pensando nisso, alguns estudiosos propuseram a utilização de arquiteturas de bancos de dados híbridas, as quais devem ser percebidas como um único banco de dados lógico e consultado por uma única linguagem para facilitar o seu uso pelos usuários. Pelos motivos citados na seção anterior a esta e com base na pesquisa bibliográfica realizada, pode-se concluir que os usuários não especialistas precisam de um meio para interagir com BD relacionais em uma linguagem natural, e devido ao contexto atual, interagir também com BD não relacionais.

Uma justificativa para a elaboração da proposta deste trabalho é a participação no

projeto “Plataforma *Big Data* para Acelerar a Transformação Digital do Estado do Ceará”, no qual um dos objetivos consiste em implementar uma política que permita que qualquer pessoa do estado consiga, de forma transparente, acessar ao conteúdo desejado nos portais do governo. Por esta razão, foi desenvolvida uma solução para tornar acessível os Dados Governamentais Abertos (OGD, acrônimo em inglês) do estado do ceará para toda a população. Os OGD abrangem dados valiosos sobre nossa sociedade, e podem ser acessados por qualquer pessoa, englobando registros públicos que se referem ao contexto de transporte, infraestrutura, educação, saúde, meio ambiente e outros segmentos.

Os OGD tornou-se popular não apenas devido à prática de transparência governamental, mas também pela capacidade de gerar valores tanto sociais quanto econômicos por meio das percepções baseadas em dados (Zheng *et al.*, 2020). A prática de OGD, em diversos países, disponibiliza um grande volume de dados em formatos diferentes e sem correlação entre eles, dificultando a extração de informações úteis desses dados. Neste sentido, surge a necessidade de técnicas, métodos ou ferramentas que auxiliem o acesso aos OGD por grande parte da população.

Assim, esta dissertação de mestrado teve como motivação propor um modelo de arquitetura que possa ser adaptável a múltiplos tipos de bancos de dados, que armazenem dados homogêneos ou heterogêneos e aceite consultas de entrada em linguagem natural, proporcionando o acesso às informações de forma transparente a todos os que delas precisam, mesmo aqueles que não possuem conhecimento em linguagem formal de banco de dados.

1.2 Objetivos

A definição do objetivo de pesquisa deste trabalho baseou-se nos passos comentados por Wazlawick (2009): i) Definir um tema de pesquisa que determina uma área de conhecimento na qual se deseja explorar; ii) Realizar uma revisão bibliográfica ou revisão do estado da arte; iii) Formalizar o objetivo da pesquisa.

1.2.1 Definir um tema de pesquisa que determina uma área de conhecimento na qual se deseja explorar

A definição do tema de pesquisa deste trabalho envolveu vários processos. Primeiramente, a identificação de áreas de interesses que seriam oportunas para explorar mais profundamente, chegando aos seguintes tópicos: Inteligência Artificial, *Machine Learning*, *Big*

Data, Análise de desempenho e Banco de Dados. Depois, foi realizada uma pesquisa para adquirir mais conhecimentos sobre tais áreas e identificar questões não resolvidas. Considerando a relevância e a originalidade das questões em abertos, iniciou-se uma busca para encontrar um ângulo único ou uma perspectiva original para abordar tais temas. E após esse processo, foi formulada uma pergunta que serviu para nortear este estudo. Como desenvolver uma arquitetura para uso de banco de dados híbridos adaptável a ferramentas de tradução de consultas em linguagem natural, visando facilitar o acesso e a manipulação de dados de forma transparente para o usuário?

1.2.2 Realizar uma revisão bibliográfica ou revisão do estado da arte

Inicialmente, foi realizada uma pesquisa através do google acadêmico buscando o tópico "*Hybrid database architecture*" e nas bases de dados IEEE, ACM e *Science Direct* (bases de dados mais específicas da área de computação), o que retornou uma grande quantidade de trabalhos relacionados ao assunto, por isso, foi adicionado um filtro para ano de publicação superior a 2014 e inferior a 2023. Nessa primeira busca, alguns trabalhos foram selecionados com base no título e da leitura do resumo. Novas buscas foram realizadas utilizando como filtro o ano de publicação superior a 2014 e inferior a 2023 e as palavras-chave no idioma inglês *"Relational and non-relational databases"*, *"SQL and NoSQL databases"*, *"Natural Language Processing"*, "NLP", *"Natural language interface"*, *"NLP tools"*, *"Natural Language Processing tools"*, *"Natural language to SQL"*, *"Natural language to NoSQL"*, *"SQL to NoSQL"*. As palavras-chave foram aplicadas individualmente, e juntamente com os conectores AND e OR para montar as strings de busca.

A revisão bibliográfica realizada serviu para fazer o levantamento dos trabalhos relacionados ao tema desta pesquisa, além de identificar soluções que usam gestão de dados heterogêneos, processamento de consultas em linguagem natural e elaboração de arquiteturas de software. Assim, reunindo o conhecimento necessário para propor uma solução possível que resolva o problema de pesquisa, cumpram os objetivos e validem/refutem a hipótese deste trabalho.

1.2.3 Formalizar o objetivo da pesquisa

O objetivo geral desta pesquisa foi desenvolvido baseados na afirmação de Wazlawick (2009), que define que uma boa sintetização de um objetivo de pesquisa, geralmente, estará em

uma forma para constatar que a hipótese especificada é verdadeira. Este trabalho visa defender a hipótese de que a utilização da arquitetura proposta não gera atraso significativo nas consultas aos bancos de dados.

Neste contexto, este trabalho propõe um modelo de arquitetura que permite que usuários não especialistas acessem informações contidas em base de dados armazenadas em um ou mais tipo de bancos de dados diferentes. A solução proposta provê simplicidade com acesso transparente a diferentes bancos de dados manipulados por linguagens formais distintas. Permitindo que um usuário sem conhecimento em linguagens formais dos bancos de dados acesse as informações formulando consultas em linguagem natural. Com isso, o objetivo geral desta dissertação de mestrado é propor um modelo de arquitetura capaz de possibilitar a usuários não especialistas extrair a partir de linguagem natural informações de base de dados homogêneas ou heterogêneas, armazenadas em banco de dados híbridos, locais ou distribuídos.

Para alcançar o objetivo geral, os seguintes objetivos específicos são necessários:

- Prover um modelo de arquitetura de bancos de dados que possibilite a fácil permuta de bancos de dados e ferramentas de traduções;
- Permitir o uso desta interface entre usuários e as bases de dados sem incremento perceptível dos custos computacionais vinculados à solução implementada;
- Validar solução na perspectiva de desempenho em relação a volume de dados e quantidade de acessos.

1.3 Contribuições

O problema abordado nesta dissertação de mestrado é a criação de um modelo de arquitetura que viabilize a utilização de consultas em linguagem natural para acesso a dados armazenados em bancos de dados relacionais e não relacionais, visando facilitar e otimizar o processo de consulta e recuperação de informações para todos os que delas precisam, mesmo aqueles que não possuem conhecimento em linguagem formais de consulta a banco de dados.

Este problema já foi investigado na literatura por pesquisas que buscavam soluções para realizar traduções de uma linguagem natural para uma linguagem formal, ou utilizando uma linguagem formal para acessar bancos de dados de tipos diferentes que possuem também linguagens formais diferentes.

Citando exemplos de trabalhos que também investigaram o problema desta dissertação, temos os seguintes: os trabalhos Deshpande e Devale (2012), Malik e Rishi (2015),Singh e

Solanki (2016) que aceitam como entrada consultas em linguagem natural para acesso a dados armazenados em bancos de dados relacionais, realizando a tradução da linguagem natural para a linguagem SQL. Os trabalhos Mondal *et al.* (2019), Pradeep *et al.* (2019) que aceitam como entrada consultas em linguagem natural para acesso a dados armazenados em bancos de dados não relacionais, realizando a tradução de consultas em linguagem natural para linguagem formal de bancos de dados não relacionais. E por último, os trabalhos Li e Gu (2019), Vathy-Fogarassy e Hugyák (2017), Bjeladinovic *et al.* (2020) que aceitam como entrada consultas em linguagem formal SQL, para acessar bancos de dados relacionais e não-relacionais.

Porém, até o momento da escrita deste trabalho, não foram encontrados trabalhos que abordem a realização de consulta em linguagem natural a base de dados homogêneas ou heterogêneas. Sendo assim, esta dissertação apresenta as seguintes contribuições:

- Um modelo de arquitetura que gera transparência para o usuário através de um módulo de interface único para acesso a múltiplos bancos de dados por meio de consultas formuladas em linguagem natural;
- Uma arquitetura projetada para suportar diferentes ferramentas de Processamento de Linguagem Natural (PLN); e
- Uma arquitetura estruturada com módulos independentes que permitem fácil adaptação a novas ferramentas ou adaptações decorrentes da necessidade dos usuários.

1.4 Organização do Trabalho

O restante deste documento está organizado da seguinte forma. O Capítulo 2 apresenta os principais conceitos para a compreensão deste trabalho, tais como *cluster* de computadores, bancos de dados relacionais e não relacionais e PLN. O Capítulo 3 apresenta os trabalhos relacionados utilizados como base bibliográfica para este trabalho. O Capítulo 4 mostra a arquitetura proposta dividida em módulos, sua descrição e funcionalidades detalhadas. O Capítulo 5 apresenta a metodologia deste trabalho, os passos para realização e desenvolvimento desta proposta. No Capítulo 6 é apresentado a avaliação experimental e seus resultados, onde é mostrado os passos de execução do experimento até análise dos resultados obtidos. O Capítulo 7 apresenta a conclusão e as possibilidades de extensões desta dissertação de mestrado.

2 FUNDAMENTAÇÃO TEÓRICA

Esse capítulo apresenta as definições necessárias para o bom entendimento deste trabalho. A arquitetura proposta foi hospedada em um *cluster* de computadores. Dois tipos de banco de dados: um banco de dados relacional (MySQL) e um não relacional (MongoDB), foram utilizados para armazenar dados estruturados e semiestruturados e no módulo de tradução da arquitetura proposta, ferramentas de tradução de linguagem natural para linguagem formal com técnicas de PLN foram empregados.

2.1 Cluster de computadores

Segundo Stallings (2010) os *clusters* são uma alternativa para multiprocessamento simétrico, como uma abordagem para fornecer alto desempenho e disponibilidade para servidores de aplicações. Podendo ser definido como um grupo de computadores completos interconectados que trabalhando juntos, como um recurso computacional unificado que pode criar a ilusão de ser uma única máquina. O termo computador completo, ainda segundo o mesmo autor, significa um sistema que pode funcionar por si só, à parte do *cluster*. Na literatura, cada computador em um *cluster* é normalmente chamado de nó. Um *cluster* pode trazer vários benefícios a um projeto, tais como os destacados por (Stallings, 2010).

- **Escalabilidade:** é possível criar grandes *clusters* que ultrapassam em muito o poder de máquinas maiores que trabalham sozinhas. Um *cluster* pode ter dezenas, centenas ou até milhares de máquinas, cada uma sendo um multiprocessador. Além disso, um *cluster* pode ser configurado de tal forma que permite adicionar novas máquinas ao *cluster* com pequenas modificações. Assim, um usuário pode começar com um sistema modesto e expandi-lo conforme a necessidade, sem ter que fazer grandes atualizações para transformar ou substituir um sistema existente por um sistema maior;
- **Alta disponibilidade:** como cada nó no *cluster* é um computador independente, a falha de um nó não significa a perda do serviço. Em muitos produtos, a tolerância a falhas é tratada automaticamente por software;
- **Preço/desempenho superiores:** usando a ideia de blocos de construção, é possível montar um *cluster* com poder computacional igual ou maior do que uma única máquina de grande porte, com custo bem menor.

2.2 Banco de dados relacionais e não relacionais

Segundo (Bjeladinovic *et al.*, 2020), os bancos de dados relacionais ou SQL (referenciando a linguagem de consulta *Structured Query Language*) têm proporcionado aos usuários um alto nível de eficiência para armazenamento e gerenciamento de dados estruturados. Essa eficiência é especialmente notável ao fornecer consistência de dados e executar consultas complexas (com muitas operações de junção). Este tipo de banco de dados se baseia no modelo relacional, o qual permite uma definição formal da estrutura, operações e regras de integridade, utilizando a linguagem de consulta SQL de forma padronizada para operações em dados. A estrutura deste tipo de banco de dados é definida usando relacionamentos, domínios, chaves, tabelas e esquema de banco de dados.

Bancos de dados relacionais são baseados no modelo ACID acrônimo para *Atomicity, Consistency, Isolation, Durability*, em português: atomicidade, consistência, isolamento e durabilidade. A atomicidade garante a integridade das transações, a consistência fornece estabilidade dos dados em um banco de dados, o isolamento garante a independência de várias transações que podem ser executadas ao mesmo tempo, e a durabilidade garante que as transações armazenadas não mudem de estado, mesmo na presença de falha. O ACID fornece consistência e disponibilidade como propriedades fortes que tornaram os bancos de dados relacionais populares (Kunda; Phiri, 2017).

Já os bancos de dados não relacionais foram desenvolvidos devido ao crescimento exponencial da internet e das aplicações web. Os bancos de dados não relacionais fornecem acesso eficiente e rápido para dados do tipo *big data* (Solanke; Rajeswari, 2017). Os domínios do *big data* envolvem dados da web, com suporte a milhões de usuários interativos ou execução de análises em terabytes de dados, como *logs* da web e fluxos de cliques, entre outros. Os dados nesses domínios são semiestruturados, não estruturados, variáveis e massivos. Criar um modelo relacional adequado para suportar esses dados pode ser bem difícil (Lawrence, 2014).

Os bancos de dados não relacionais ou *Not only Structured Query Language* (NoSQL) são baseados no modelo BASE, acrônimo para *Basically Available, Soft State e Eventually Consistent*, em português: disponibilidade básica, estado flexível e consistente eventual. A natureza distribuída dos bancos de dados não relacionais traz a possibilidade dos dados estarem parcialmente disponíveis enquanto outras partes do banco de dados distribuído não estiverem operando ou não puderem ser alcançadas. O estado flexível permite que os dados variem ao longo do tempo, com ou sem entrada. A propriedade de consistente eventual garante que os

dados se tornarão consistentes no futuro e não imediatamente após uma operação. O BASE dá aos bancos de dados não relacionais a capacidade de escalar facilmente, oferecer melhor desempenho e maiores níveis de disponibilidade para seus usuários (Kunda; Phiri, 2017).

Outra característica que diferem os bancos de dados não relacionais dos relacionais, é a maior flexibilidade de armazenamento, por permitirem armazenar vários tipos de dados, estruturados, semiestruturados ou mesmo não estruturados, segundo (Jose; Abraham, 2017).

Os bancos de dados não relacionais são usados para muitos propósitos distintos, e por isso possuem diferentes categorias (Bathla *et al.*, 2018). Tais categorias são definidas abaixo:

- **Orientado a coluna:** Nessa categoria os dados são armazenados em um conjunto de colunas e possuem uma chave de identificação, sendo acessados em conjunto. Devido à forma de armazenamento, esse tipo de banco de dados remove a desvantagem dos bancos de dados relacionais tradicionais, nos quais os valores são armazenados como nulos para as linhas onde os valores não são conhecidos, ocupando mais espaço de armazenamento do que o necessário (Bathla *et al.*, 2018). Exemplos de bancos de dados dessa categoria: HBase, Amazon SimpleDB e Cassandra;
- **Baseado em grafos:** Os bancos de dados baseados em grafos facilitam o armazenamento de entidades e conexões ou relacionamentos entre eles (Jose; Abraham, 2017). Muitos autores utilizam da nomenclatura de nós e arestas para referir-se às entidades e conexões. Assim, a ocorrência de um objeto é anotada como um nó, as linhas que conectam os nós e mostram as relações, são conhecidas como arestas, as últimas mostram a direção na qual os nós são organizados com base nessas relações (Jose; Abraham, 2017). Este tipo de conexão permite a análise e interpretação dos dados de diferentes maneiras com base em relacionamentos, embora sejam armazenados em uma única instância. Exemplos de bancos de dados dessa categoria: Neo4J e OrientDB;
- **Chave e valor:** Nessa técnica de armazenamento, os dados são armazenados na forma de uma chave que possui um valor único, como mapa ou dicionário (Bathla *et al.*, 2018). Esse tipo de armazenamento possibilita acesso rápido aos dados e é muito eficiente para armazenamento distribuído. Exemplos: Redis, DynamoDB e Oracle NoSQL;
- **Orientado a documentos:** Nessa categoria os dados são armazenados em uma única instância, em formato de documentos XML, JSON, e/ou BSON(Jose; Abraham, 2017). Exemplos de bancos de dados dessa categoria: MongoDB, CouchDB e OrientDB.

Um ponto a destacar é que muitas empresas e organizações optaram por não substituir

o banco de dados relacional existente e sim migrar para a solução de banco de dados híbrido (relacional e não relacional). As soluções híbridas de banco de dados combinam o benefício de serem adequadas a muitos aplicativos já existentes e permitem a alta escalabilidade. Na solução híbrida, os aplicativos comunicam-se com o banco de dados relacional para lidar com solicitações de dados pequenos e em escala. As consultas que exigem grande quantidade de dados são tratadas usando banco de dados não relacional. O banco de dados não relacional funciona como *pool* de dados e fornece operações em lote, backups e análises (Solanke; Rajeswari, 2017).

2.3 Processamento de linguagem natural (PLN)

Na década de 1980, o conceito de linguagem natural se expandiu e houve uma percepção crescente da necessidade de encontrar soluções para as limitações da programação em linguagem natural e um impulso geral para aplicativos que funcionassem com a linguagem em um amplo contexto do mundo real (Hazboun *et al.*, 2021).

O campo do PLN também conhecido como linguística computacional, envolve a engenharia de modelos e processos computacionais para resolver problemas práticos na compreensão de linguagens humanas. Essas soluções são usadas para construir softwares úteis.

Por isso, o PLN é uma gama de técnicas computacionais motivadas para analisar e representar textos que ocorrem naturalmente em um ou mais níveis de análise linguística com a finalidade de alcançar processamento de linguagem semelhante ao humano para uma série de tarefas ou aplicações. Ao referenciar níveis de análise linguística, refere-se à análise fonética, morfológica, lexical, sintática, semântica, discursiva e pragmática da linguagem, cuja suposição é que os humanos normalmente utilizam todos esses níveis para produzir ou compreender a linguagem, segundo (Zhao *et al.*, 2020).

O processamento de linguagem natural está se tornando uma das áreas mais ativas na interação homem-computador e pode fazer o computador interpretar as linguagens usadas naturalmente pelos humanos. Embora a linguagem natural possa fazer o uso dos sistemas mais acessíveis, demonstrou ser uma tarefa desafiadora para os computadores dominarem (Deshpande; Devale, 2012).

Atualmente existe uma considerável quantidade de algoritmos e técnicas de PLN, embora, segundo Sanyal *et al.* (2021), o PLN depende principalmente de *Machine Learning* ou *Deep Learning* para implementar com sucesso um vínculo entre máquinas e humanos. Algumas das metodologias usadas nessa área são definidas por este mesmo autor, (Sanyal *et al.*, 2021) e

listadas abaixo.

- **Reconhecimento de Entidade Nomeada (NER):** Usado frequentemente em análise semântica, ou seja, o significado, oculto ou não, transmitido por qualquer texto. Nesse método, os algoritmos recebem entradas como textos, parágrafos ou frases e, em seguida, identificam rapidamente todos os nomes ou substantivos presentes neles;
- **Tokenização:** Significa dividir uma frase ou texto inteiro em um número diferente de *tokens*. *Tokens* são as menores unidades identificáveis, sejam palavras, pequenas frases, números, pontuações, caracteres, etc;
- ***Stemming* e Lematização:** *Stemming* é definido como o processo para reduzir as palavras para sua forma de palavra base ou raiz. Isso é feito basicamente cortando os sufixos, por qual pode retornar erros. A Lematização também trata basicamente em reduzir as palavras para sua forma base ou raiz, mais com melhor uso do vocabulário ou da análise morfológica das palavras. Isso é feito para remover terminações abruptas e fornecer uma forma de dicionário adequada de qualquer caractere ou palavra, conhecida como "Lemma". Ele basicamente remove palavras desnecessárias interpretando a parte do discurso (POS) ou o contexto da palavra. Por exemplo, aplicar *stemming* na palavra "carreiras" retornará "carr" e aplicar lematização na mesma palavra retornará "carreira";
- ***Sentence Segmentation:*** A segmentação de frases ajuda a dividir o texto em frases significativas e abrangentes. Envolve, a identificação de frases entre palavras em textos. Isso é feito pelo uso e ajuda de pontuações encontradas nos textos. Também chamado de detecção de limite de frase, desambiguação de limite de frase ou reconhecimento de limite de frase;
- **Remoção de *stopword:*** essa técnica consiste em um pré-processamento que retira palavras que não possuem um significado relevante para análise, como, por exemplo, artigos. A retirada dessas palavras reduz o trabalho de processamento;

3 TRABALHOS RELACIONADOS

Neste capítulo, são apresentados os trabalhos que possuem algum aspecto semelhante a esta pesquisa. Os trabalhos relacionados apresentados nesta seção foram organizados nas seguintes categorias: pesquisas que realizam tradução de consultas em linguagem natural para SQL; pesquisas que realizam a tradução de consultas em linguagem natural para linguagem de consultas a banco de dados não relacional; pesquisas que utilizam linguagem natural para acesso a dados armazenados em uma arquitetura híbrida.

3.1 Trabalhos relacionados a tradução de linguagem natural para linguagem SQL

No trabalho Deshpande e Devale (2012), os autores propõem uma interface de banco de dados que aceita consultas em linguagem natural no idioma inglês e as realiza com base em uma gramática livre de contexto probabilística (acrônimo do inglês, PCFG) em um banco de dados relacional, utilizando um dicionário de dados para auxiliar sua tradução. Para a tradução da consulta ser realizada, um dicionário de dados limitado deve ser criado, no qual todas as possibilidades relacionadas a um sistema particular devem ser armazenadas e o mesmo deve ser atualizado regularmente. Possíveis ambiguidades entre as palavras podem ocorrer, nesse caso, as mesmas devem ser tratadas durante a etapa de processamento da consulta em linguagem natural.

A arquitetura do sistema de interface de banco de dados em linguagem natural desenvolvida pelos autores realiza a tradução por meio de marcação de voz seguida pela marcação de palavras, utilizando *tags* para isso, então analisa a frase marcada por uma gramática e gera a(s) árvore(s) gramatical(s). Em seguida, o tradutor SQL coleta informações da árvore, o que pode ser realizado por duas técnicas: estrutura de dependência e sub categorização do verbo. Tais técnicas também são usadas na desambiguação, uma vez que uma PCFG é livre de contexto e ancestral. Uma estrutura de dependência é usada para capturar as relações inerentes que ocorrem nos textos do *corpus* que podem ser críticas em aplicativos do mundo real e é geralmente descrito por dependências digitadas. Uma dependência digitada representa relações gramaticais entre palavras individuais com rótulos de dependência, como sujeito ou objeto indireto. Na sub-categorização de verbos podemos encontrar os objetos desse verbo e com isso o alvo da consulta.

Os autores relatam que os resultados dos experimentos de testes em um ambiente controlado foram bem-sucedidos e satisfatórios e que se pode concluir que é viável traduzir uma consulta em linguagem natural para SQL com uma abordagem probabilística, embora seja

possível melhorar o desempenho do sistema através de métodos que melhorem a coerência e a fluência dos textos de saída.

O artigo Malik e Rishi (2015) apresenta uma interface que converte uma consulta de texto inglês em uma consulta de linguagem SQL. A conversão é feita em três estágios principais: construção de banco de dados descritivo, filtragem de consulta não estruturada e mapeamento. Na primeira etapa, é construído um banco de dados de construção de linguagem, que contém as seguintes informações: informações específicas do domínio; a construção da linguagem em diferentes categorias e as palavras mapeadas. Na filtragem da consulta é feita a extração das palavras-chave identificadas como Construções de Domínio ou Linguagem, nessa etapa o texto de entrada entra no processo de filtragem sendo submetido a dois processos principais: o primeiro é a remoção das palavras da lista de *stopword* e depois as palavras restantes são identificadas como as palavras-chave sob diferentes categorias do documento, então uma análise dessas palavras-chave é realizada sob as construções de linguagem e as construções de domínio. Para organizar as consultas, o autor definiu regras sob PLN. No mapeamento é feita a conversão da consulta não estruturada e filtrada em SQL utilizando as palavras-chave identificadas na etapa anterior, então os dados desejados são recuperados do conjunto do BD.

O sistema proposto no trabalho citado foi implementado em ambiente Java no *Enterprise Employee Database*, o banco de dados foi construído usando *SQLyog* (MYSQL) e integrado à interface usando JDBC e ODBC, já que todo o sistema é implementado em ambiente JAVA. No trabalho citado, os autores destacam um avanço em relação aos sistemas disponíveis com uma gama mais ampla de manipulação de consultas devido às construções de consultas com palavras-chave especiais, como “pelo menos” e “no máximo”. No final do artigo, os autores afirmam que resultados obtidos mostram a efetiva conversão de consultas e extração de registros do *Dataset* utilizado nos testes do trabalho e mostram uma tabela que apresenta as consultas em Linguagem Natural (LN) e em SQL.

No trabalho de Singh e Solanki (2016), os autores apresentam uma interface desenvolvida com técnica de correspondência semântica para traduzir a consulta de linguagem natural no idioma inglês para SQL, usando um conjunto de regras de produção e dicionário de dados, onde o dicionário de dados consiste em conjuntos semânticos de relações e atributos. Os autores nomearam o sistema como *Natural Language to SQL Convertor* (NLTSQLC). Para realizar a tradução, a consulta passa por uma série de etapas como conversão de minúsculas, *tokenização*, marcação de voz, extração de elemento de banco de dados, extração de elemento SQL e remoção

de ambiguidade, então é convertida e o resultado é retornado para o usuário.

Segundo os autores, a principal vantagem dessa estrutura é que pode ser expandida sempre que algum novo conhecimento for descoberto. Para avaliar o novo sistema, o mesmo é comparado a uma versão anterior, e um conjunto de 28 e 50 perguntas foram disparadas, e com base no tipo de consultas, uma matriz de confusão foi criada para avaliar e verificar a precisão dos sistemas.

Os trabalhos relacionados apresentados nessa subseção realizam tradução de consultas em linguagem natural para linguagem formal SQL para acesso a banco de dados relacional e se relacionam a este por esse objetivo, porém diferem, pois este trabalho busca acessar os dois tipos de bancos de dados, relacional e não relacional, através de um modelo de arquitetura híbrida.

3.2 Trabalhos relacionados a tradução de linguagem natural para linguagem formal de bancos não relacionais

No artigo Mondal *et al.* (2019) é proposto um modelo de consulta-resposta automatizado de *Natural Language Query to NoSQL Generation* que usa *Query-Response Model*, nomeado NLNSM. O sistema proposto pelos autores usa *POS-Tagging* e uma técnica baseada em combinação para lidar com sentenças de consulta assertivas, interrogativas, imperativas, compostas e complexas do usuário.

Para realizar a tradução, o sistema usa um algoritmo que gera uma consulta NoSQL equivalente para cada uma das consultas de linguagem natural realizada. Com essa consulta é possível recuperar dados do banco de dados MongoDB residente no NLNSM. Na tradução, o sistema remove as palavras repetidas e usa um banco de dados de sinônimos para auxiliar a tradução. Os autores ressaltam que escolheram o banco de dados MongoDB pelo fato dele usar o formato de dados JSON ou JSON binário, facilitando o uso de técnicas de análise, além de poder conter uma coleção de documentos diferentes onde o número de campos, conteúdo e tamanho do documento podem diferir de um documento para outro e suportar consultas dinâmicas.

O artigo citado propõe um sistema automático capaz de gerar respostas imediatas para vários tipos de consultas em linguagem natural do usuário, e tem como seu público alvo as organizações que oferecem e-serviços como: e-Educação, e-Hospitalidade, e-Turismo, e-Banking, e-College Management etc.

O artigo de Pradeep *et al.* (2019) propõe a conversão de uma consulta em linguagem

natural formulada em inglês para uma consulta em linguagem formal de banco de dados não relacional da subcategoria de armazenamento baseada em documentos. Os autores usaram o método de tradução automático baseado em codificador-decodificador para conversão, onde é criado um vetor de pensamento com base na consulta em inglês fornecida e então uma rede neural é usada para construir a consulta em linguagem formal. O sistema proposto pelos autores lida com dez tipos de consultas do MongoDB (*Count, Find, Insert, Get, Select, Logical, Create, Delete, Remove, Update* e *Sort*), e utiliza dez modelos separados de aprendizado profundo (um modelo para cada tipo de consulta). Segundo os autores, o sistema mostrou resultados satisfatórios, embora não consiga lidar muito bem com consultas complexas.

Os autores descrevem que o sistema é implementado em duas fases: a fase de treinamento, onde é feito a criação e pré-processamento do conjunto de dados e a construção dos modelos de aprendizado profundo e a fase de teste, onde é realizado a detecção do tipo de consulta (que encaminha para o modelo de aprendizagem adequado) e geração da consulta em linguagem formal de banco de dados não relacional equivalente. Para realização dos testes foi utilizado o banco de dados MongoDB. Como resultados, o sistema proposto apresenta 71,5% de acerto global, sendo as consultas de remoção obtendo maior precisão, 87,1% e consulta de inserção com menor precisão, 48,4%. A métrica utilizada pelos autores para avaliar a sentença gerada a partir de uma sentença de referência foi o *Bilingual Evaluation Understudy Score*, ou BLEU.

Os trabalhos relacionados apresentados nessa subseção realizam tradução de consultas em linguagem natural para linguagem formal de banco de dados não relacionais e se relacionam a este por esse objetivo, diferenciando, pois este trabalho busca acessar os dois tipos de bancos de dados, relacional e não relacional através de um modelo de arquitetura híbrida.

3.3 Trabalhos relacionados à arquitetura híbrida

No artigo de Li e Gu (2019), os autores propõem uma abordagem de integração de bancos de dados de tipos diferentes, incluindo MySQL, MongoDB e Redis, através de uma arquitetura que permita que os usuários consultem dados simultaneamente dos bancos de dados relacionais e não relacionais através de uma única consulta em linguagem SQL. O sistema proposto foi chamado MSI (em português, integração de múltiplas fontes), que se baseia nas camadas subjacentes do sistema operacional e dos sistemas de gerenciamento de banco de dados. O sistema inclui oito componentes: componente API *dispatcher*, componente SQL *parser*,

componente SQL otimizador, componente SQL *router*, componente SQL executor, componente de adaptador de DBMS (sistema de gerenciamento de banco de dados), componente de fusão de resultados e componente de gerenciamento de metadados, além de utilizar algoritmos-chave que podem transformar uma consulta aninhada complexa em uma forma não aninhada equivalente, que geralmente contém o predicado de junção.

O autor afirma que essa abordagem trata tipos de fontes de dados integrados, otimização de consultas aninhadas complexas e as simplifica em cláusulas condicionais, o que pode reduzir a complexidade do desenvolvimento e melhorar a eficiência dos sistemas de software com múltiplos bancos de dados em ambiente de computação em nuvem. No final do artigo citado, é realizada uma análise comparativa qualitativa e a quantitativa com sistemas similares discutidos nos trabalhos relacionados do artigo.

No artigo de Vathy-Fogarassy e Hugyák (2017), é proposto uma metodologia de integração para consultar dados de diferentes sistemas de banco de dados (relacionais e não relacionais) através de um método de generalização de consultas. O método utilizado pelos autores foi baseado em uma abordagem de metamodelo que abrange as heterogeneidades estruturais, semânticas e sintáticas dos sistemas-fonte.

A solução proposta pelo artigo citado coleta dados de diferentes sistemas de gerenciamento de banco de dados separados e os migra através de um único objeto JSON. O objeto JSON possui, independente do modelo de dados, descrições de todas as informações estruturais e semânticas do sistema de banco de dados de origem. A solução cria uma camada de banco de dados para fornecer todos os serviços de banco de dados sobre os sistemas de origem, porém não oferece suporte às junções e agregações entre fontes de dados. As consultas são realizadas individualmente em cada banco de dados e as respostas são reunidas em um único JSON, podendo ser analisadas em conjunto.

Os autores do trabalho citado destacam que a principal contribuição é um novo método de integração de dados e fluxo de trabalho, que implementa a integração de dados de diferentes fontes. Uma aplicação web foi desenvolvida utilizando o método descrito, e chamada de HybridDB. Para o desenvolvimento da aplicação foi utilizado do lado do servidor, a linguagem JavaScript, *framework* Node.js, modelo de E/S assíncrono sem bloqueio e orientado a eventos e padrão de arquitetura *model-view-controller* (MVC), do lado cliente foram utilizadas a linguagem JavaScript e o *framework* AngularJS. A aplicação oferece suporte aos BDs MySQL e MongoDB, porém os autores destacam que a mesma pode ser adaptada para suportar outros bancos de dados

relacionais e não relacionais. Uma análise de desempenho foi realizada para verificar o tempo de execução das consultas e seus resultados foram satisfatórios e apresentados por meios de gráficos.

No artigo de Bjeladinovic *et al.* (2020), é proposto uma arquitetura para fornecer integração e uso uniforme de bancos de dados relacionais e não relacionais como componentes de um banco de dados híbrido. A abordagem apresentada pelos autores apresenta design de arquitetura dedicado, sendo uma ampliação da arquitetura de três camadas com a adição de novos componentes especialmente desenvolvidos.

O artigo citado utilizou dois requisitos não funcionais para o desenvolvimento da arquitetura proposta: integração e usabilidade. A integração dos bancos de dados relacionais e não relacionais realizada para fornecer um sistema com um único banco de dados lógico, enquanto a usabilidade garante que um sistema complexo, com bancos de dados de diferentes tipos, mantenha um nível aceitável de simplicidade. Uma característica da simplicidade destacada pelos autores é o uso de uma interface simples que aceita instruções SQL para acessar ambos os tipos de bancos de dados.

Como resultados, os autores apresentaram casos de uso representativos e os efeitos alcançados com a integração e uso da arquitetura proposta, além de uma análise com testes que empregaram operações CRUD (*Create, Read, Update e Delete*) para mostrar o impacto no desempenho. Para essa análise os bancos de dados relacionais e não relacionais foram testados individualmente e posteriormente o banco de dados híbrido. O Oracle 12c DBMS foi selecionado para o componente relacional do banco de dados híbrido, enquanto o MongoDB 3.6 foi usado para o componente não relacional. os experimentos evidenciaram a eficiência e a flexibilidade dos bancos de dados híbridos em lidar com as operações de manipulação de dados, bem como a capacidade de integrar dados de diferentes fontes de forma uniforme e eficaz.

Os trabalhos relacionados apresentados nesta subseção formam uma espécie de pesquisa bibliográfica no uso de uma única linguagem para acesso a bancos de dados relacionais e não relacionais por meio de um modelo de arquitetura híbrida e se relacionam a este por este motivo, porém diferem, pois este trabalho tem em vista acessar bancos de dados relacionais e não relacionais por meio de um modelo de arquitetura híbrida que aceita como entrada consultas em linguagem natural, enquanto os trabalhos citados nessa subseção utilizam a linguagem SQL para esse objetivo.

3.4 Quadro de análise comparativa dos trabalhos relacionados

Uma análise comparativa é mostrada no Quadro 1. Alguns pontos a observar são que Deshpande e Devale (2012), Malik e Rishi (2015),Singh e Solanki (2016),Mondal *et al.* (2019), Pradeep *et al.* (2019) e este trabalho, permitem a obtenção de informações a partir da Linguagem Natural(LN) no idioma inglês, enquanto os demais necessitam que o usuário tenha conhecimento em linguagem de acesso a banco de dados relacional (SQL); Li e Gu (2019), Vathy-Fogarassy e Hugyák (2017), Bjeladinovic *et al.* (2020) e este trabalho apresentam soluções que suportam bancos de dados relacionais e não relacionais; Deshpande e Devale (2012), Malik e Rishi (2015),Singh e Solanki (2016),Mondal *et al.* (2019), Pradeep *et al.* (2019) não realizam uma análise sob bancos de dados implementados, utilizam como métrica técnicas que comparam a tradução obtida como saída a consultas em linguagem formal formuladas anteriormente. Dos trabalhos citados, apenas este trabalho, Deshpande e Devale (2012) e Singh e Solanki (2016) focam em consultas de agregações e apenas este trabalho aceita como entrada linguagem natural, para acessar bancos de dados relacionais e não relacionais e realiza análise da solução sobre bancos de dados implementados.

Quadro 1 – Quadro comparativa dos trabalhos relacionais

Trabalhos	**Entrada**	**BD**	**Métrica**	**Consultas Implementado**	**Implementado**
Deshpande e Devale (2012)	LN	Relacional	Acurácia	Seleção e Agregação simples	Não
Malik e Rishi (2015)	LN	Relacional	Acurácia	DQL (Exceto funções de agregação)	Não
Singh e Solanki (2016)	LN	Relacional	Matriz de confusão	DQL	Não
Mondal *et al.* (2019)	LN	Não relacional (MongoDB)	Acurácia	Não Informa	Não
Pradeep *et al.* (2019)	LN	Não relacional (MongoDB)	BLEU	Count, Find, Insert, Get, Select, Logical, Create, Delete, Remove, Sort Update .	Não
Li e Gu (2019)	SQL	Relacional e Não relacional (MySQL, MongoDB e Redis)	Tempo de Execução	SQL(CRUD) e NoSQL (find, insert, update e remove)	Distribuído
Vathy-Fogarassy e Hugyák (2017)	SQL	Relacional e Não relacional (MySQL e MongoDB)	Tempo de Execução	SQL(CRUD) e NoSQL (find, insert, update e remove)	Distribuído
Bjeladinovic *et al.* (2020)	SQL	Relacional e Não relacional (Oracle e MongoDB)	Tempo de Execução	SQL(CRUD) e NoSQL (find, insert, update e remove)	Distribuído
Este Trabalho	LN	Relacional e Não relacional (MySQL e MongoDB)	Tempo de Execução	Agregação simples	Distribuído

Fonte: Elaborado pela autora (2023).

4 MODELO DE ARQUITETURA PROPOSTO

Este trabalho apresenta um modelo de arquitetura adaptável a múltiplos tipos de bancos de dados, que aceita consultas de entrada em linguagem natural. Inicialmente a arquitetura foi projetada com os BDs MySQL e MongoDB, os mesmos foram escolhidos por estarem entre os 5 bancos de dados mais utilizados no mundo, segundo o ranqueamento realizado pelo site db-engines. Além de serem os bancos de dados utilizados nos trabalhos relacionados: Mondal *et al.* (2019), Pradeep *et al.* (2019), Li e Gu (2019), Vathy-Fogarassy e Hugyák (2017) e Bjeladinovic *et al.* (2020).

O modelo de arquitetura escolhido para a implementação foi o modelo cliente-servidor (Kurose; Ross, 2010), onde o servidor é responsável por atender requisições por demanda para o processamento de consultas em linguagem natural de múltiplos clientes (Tanenbaum; Steen, 2007). Já o cliente, é o responsável por realizar e estabelecer um canal de comunicação para as requisições serem enviadas ao servidor para fins de processamento (Tanenbaum; Steen, 2007).

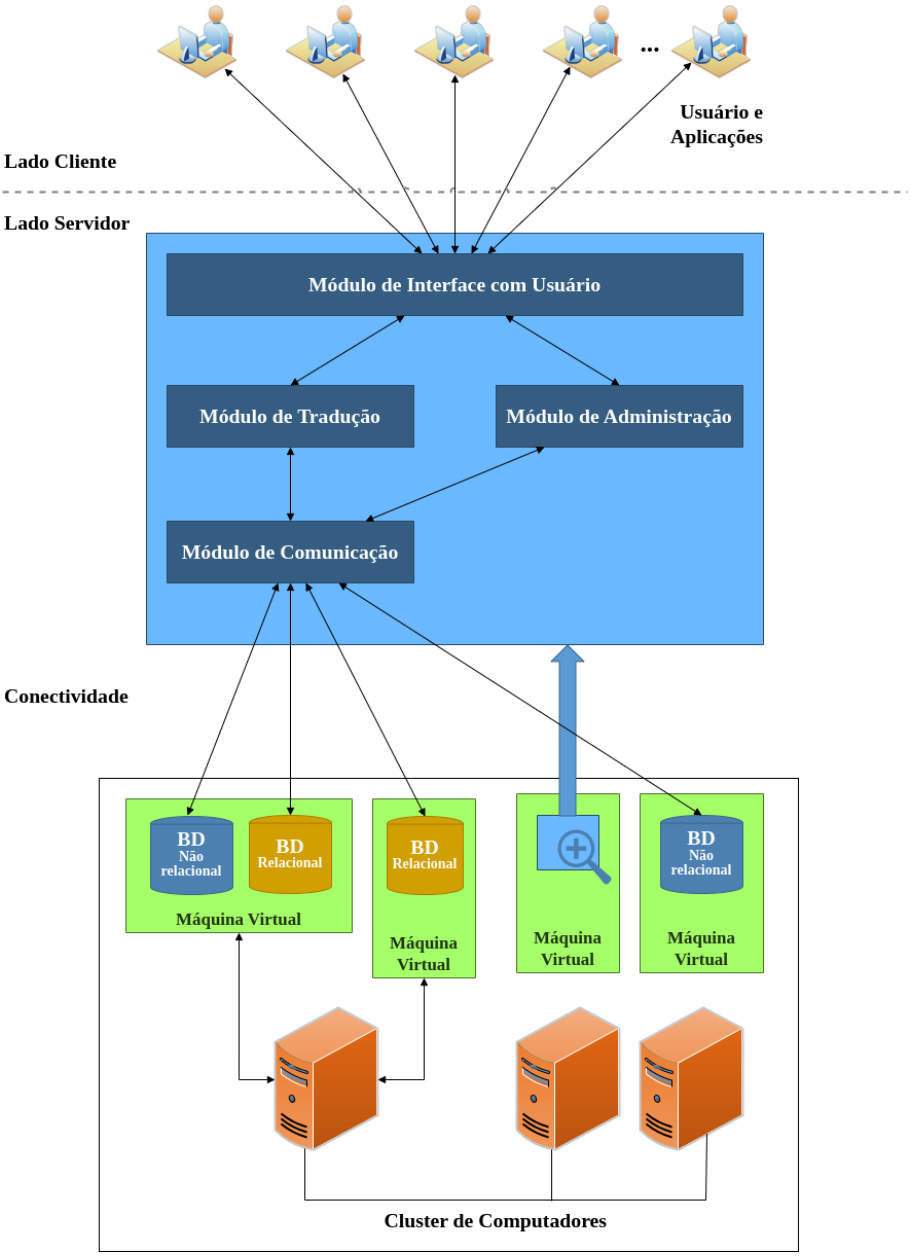
Como primeiro resultado, um modelo de arquitetura foi projetada para atender os objetivos estabelecidos neste trabalho. A Figura 1 exibe a arquitetura de sistema projetada. O lado servidor possui quatro módulos: i) **Módulo de Interface com Usuário**, ii) **Módulo de Tradução**, iii) **Módulo de Comunicação** e o iv) **Módulo de Administração**.

– **(i) Módulo de Interface com Usuário** é responsável pelos processos de autenticação do usuário, verificando quais são as ações de manipulação de dados permitidas e quais conjuntos de dados podem ser manipulados. Os usuários devem ser classificados em dois tipos: usuários convencionais, com acesso à leitura dos dados que possuem permissão e usuários administradores, que estendem o perfil de usuário convencional e possuem permissão para adicionar novos usuários e configurar permissões de acesso. Esse módulo também é responsável por receber as requisições em LN do usuário e apresentar a resposta. Para auxiliar na busca correta das informações, esse módulo deve contém um corretor ortográfico, pois os usuários podem cometer erros que variam entre erros ortográficos, uso de linguagem coloquial e abreviações.

  Neste módulo, os usuários poderão realizar consultas do tipo agregação sobre dados armazenados no mesmo banco de dados por meio de consultas em linguagem natural no idioma inglês.

  As entradas do sistema foram inicialmente fornecidas por meio de uma função que realiza

Figura 1 – Modelo de arquitetura proposta



Fonte: Elaborada pela autora(2023)

a leitura das consultas em linguagem natural em um arquivo de texto e as encaminha para a interface, onde as saídas são direcionadas para gravação em outro arquivo de texto;

– **(ii) Módulo de tradução** é responsável pela tradução de consulta em linguagem natural para as linguagens formais utilizadas pelos bancos de dados implantados no ambiente. O módulo de tradução foi implementado na linguagem de programação Python, servindo-se da ferramenta LN2SQL publicada em (Ferrero, 2022) e descrita no artigo (Couderc; Ferrero, 2015b) para tradução das consultas em LN para linguagem SQL e a ferramenta *SQL To MongoDB Query Converter*, publicada em (Russell, 2022) para a tradução das

consultas em linguagem SQL para linguagem MongoDB. A ferramenta LN2SQL recebe como entrada uma estrutura de um banco de dados relacional e uma consulta em linguagem natural e traduz a última em uma consulta em linguagem SQL válida, capaz de ser processada na estrutura do banco de dados relacional informado na entrada. Para a tradução, a ferramenta utiliza um dicionário de dados limitado. Apesar da ferramenta LN2SQL ser extensível para a tradução de consultas em linguagem natural em diversos idiomas, para este trabalho, foi utilizado o idioma inglês, pois o dicionário em inglês já estava disponível e os exemplos facilitam os testes. A ferramenta *SQL To MongoDB Query Converter* recebe como entrada uma consulta em linguagem SQL e transforma em uma consulta para o formato compreendido pelo MongoDB. Essa ferramenta utiliza técnica de análise sintática para interpretar consultas SQL, analisa a estrutura da consulta SQL para identificar os elementos-chave, como cláusulas SELECT, FROM, WHERE, etc., e depois mapear esses elementos para gerar a consulta MongoDB correspondente com base nessa análise sintática.

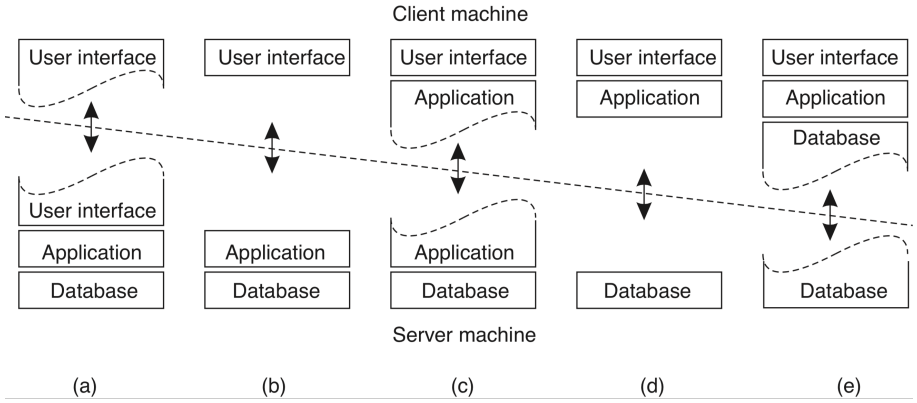
- **(iii) Módulo de comunicação** é responsável pela interação com os SGBDs implantados em ambientes virtualizados, como os clusters de computadores ou nuvens computacionais. O módulo de comunicação recebe a consulta traduzida pelo **módulo de tradução** e realiza o processo de execução da consulta. Após a execução da consulta, o módulo de comunicação recebe e encaminha os resultados do processamento da consulta para o **módulo de interface com usuário**.
- **(iv) Módulo de administração** é responsável pelo gerenciamento de todos os usuários e seus respectivos perfis. Através desse módulo, o administrador do sistema cadastra, atualiza, remove usuários e gerência as permissões de acesso aos dados.

4.1 Modelo físico

Conforme mencionado, o modelo proposto aqui se baseia na arquitetura cliente-servidor. De acordo com (Tanenbaum; Steen, 2007), uma aplicação cliente-servidor oferece várias opções de distribuição física entre as máquinas, como ilustrado na Figura 2. Na Figura 2(a), apenas a interface dependente do terminal está localizada na máquina cliente, enquanto o restante da aplicação e os bancos de dados estão na máquina servidor. Na Figura 2(b), toda a parte da aplicação de interação com o cliente ocorre no lado do cliente, enquanto na Figura 2(c), além da interface do cliente, parte da aplicação é alocada na máquina cliente, isso ocorre

especialmente quando há necessidade de operações sobre dados em caches locais ou memória. As organizações representadas pelas Figuras 2(d) e (e) são utilizadas para conexões de clientes com sistemas de arquivos distribuídos ou bancos de dados. Nestes casos, a maior parte da aplicação é executada na máquina cliente, porém todas as operações com arquivos ou acessos a bancos de dados são direcionadas ao servidor.

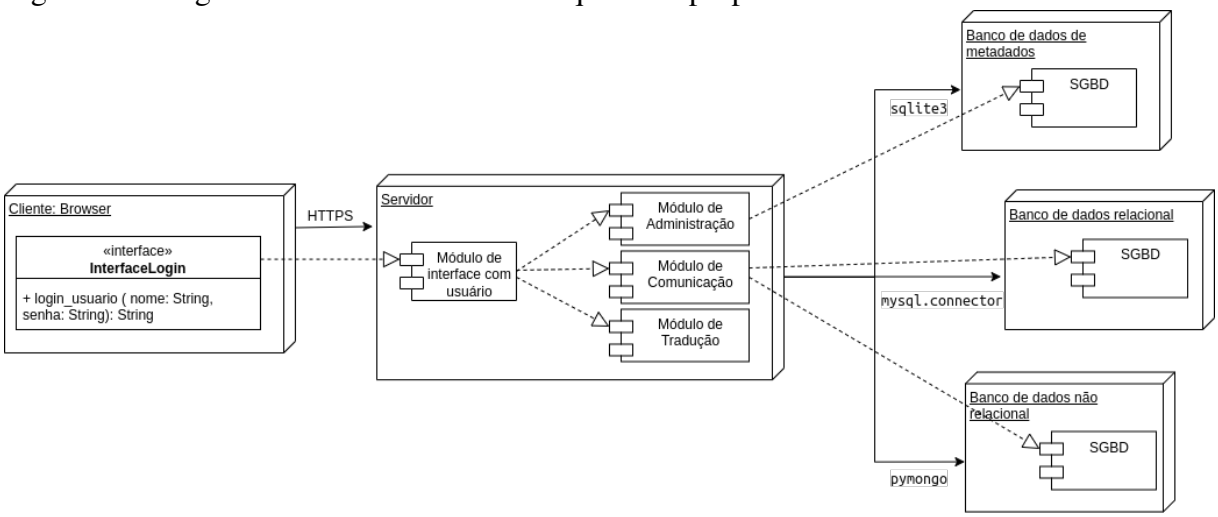
Figura 2 – Alternativas de organização cliente-servidor



Fonte: (Tanenbaum; Steen, 2007)

A Figura 3 apresenta o modelo de arquitetura física da proposta deste trabalho, o qual a organização da distribuição da aplicação baseasse na organização ilustrada na Figura 2(b). A Figura apresenta 3 a combinação do diagrama de implantação e do diagrama de componentes.

Figura 3 – Diagrama físico do modelo de arquitetura proposta



Fonte: Elaborada pela autora(2024)

– **Cliente: Browser (Cliente Web):** Representa o navegador do cliente que interage com o sistema por meio de uma interface de login. É o responsável por receber as credenciais dos usuários para validação do acesso ao sistema.

- **Servidor:** O servidor é onde a lógica da aplicação e o processamento principal do sistema ocorrem. Ele abriga os diferentes módulos do sistema e gerência a comunicação com os BDs. Parte do módulo interface com usuário, e os módulos de administração, comunicação e tradução pertencem ao lado servidor, além dos bancos de dados.
- **Módulo de Interface com Usuário:** Responsável pela interação direta com o usuário.
- **Módulo de Administração:** Gerencia os aspectos administrativos do sistema, como cadastro, listagem, atualização e exclusão de usuários e bancos de dados. Esse módulo é acessado apenas por usuários autorizados, como administradores do sistema.
- **Módulo de Comunicação:** Facilita a comunicação entre diferentes partes do sistema, incluindo a conexão com bancos de dados e gerência a troca de dados entre os componentes do sistema.
- **Módulo de Tradução:** Responsável pela tradução de consultas entre linguagem natural utilizada pelo sistema e pelos bancos de dados. Como, por exemplo, tradução de consultas de linguagem natural para SQL ou MongoDB.
- **Banco de Dados de Metadados:** Armazena metadados sobre os bancos de dados utilizados pelo sistema e os dados dos usuários.
- **Banco de Dados Relacional:** Representa um banco de dados relacional utilizado pelo sistema para armazenar dados estruturados.
- **Banco de Dados Não Relacional:** Representa um banco de dados não relacional utilizado pelo sistema para armazenar dados não estruturados ou semi-estruturados.

Este diagrama fornece uma visão geral da arquitetura do sistema proposto, incluindo os componentes principais e suas interações. Ele ajuda a entender como os diferentes elementos do sistema se comunicam e como são implantados em um ambiente físico.

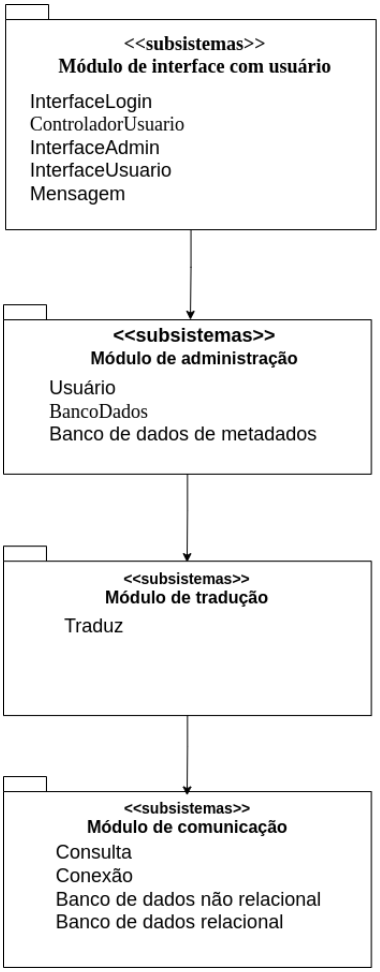
4.2 Modelo lógico

A Figura 4 representa o modelo proposto na perspectiva lógica através do diagrama de subsistemas da proposta. Cada subsistema é composto por um conjunto de classes relacionadas que trabalham juntas para fornecer funcionalidades específicas no sistema geral. Esses subsistemas são identificados e definidos para modularizar e organizar o sistema de acordo com sua arquitetura.

Podemos observar os seguintes subsistemas:

- **Módulo de Interface com Usuário:** Este subsistema abriga componentes relacionados à

Figura 4 – Diagrama lógico do modelo de arquitetura proposta



Fonte: Elaborada pela autora(2024)

interface do usuário. Inclui classes como InterfaceLogin, ControladorUsuario, InterfaceAdmin, InterfaceUsuario e Mensagem. Inicialmente, a comunicação com este módulo é realizada através interfaces de linha de comando, porém sendo possível adicionar interfaces gráficas.

- **Módulo de Administração:** Este subsistema é responsável por gerenciar aspectos administrativos do sistema. Contém classes relacionadas à administração de usuários e bancos de dados. Possui as classes Usuário, BancoDados e um Banco de dados de metadados, para realizar operações relacionadas à gestão de dados do sistema.
- **Módulo de Tradução:** Este subsistema trata de operações relacionadas à tradução de consultas. Possui a classe traduz que lidam com a tradução de linguagem natural para SQL e MongoDB.
- **Módulo de Comunicação:** Este subsistema contém as classes responsáveis pela comuni-

cação com bancos de dados e possui as classes como Consulta, Conexão e as representações de bancos de dados relacionais e não relacionais.

Para complementar a entendimento da parte lógica do modelo de arquitetura aqui proposto, também é apresentado o diagrama de classes do sistema.

4.2.1 Diagrama de classes

A Figura 5 mostra o diagrama de classes que representa a estrutura simplificada do sistema proposto. Cada módulo e classe desempenha um papel específico no fluxo de trabalho do sistema.

O **módulo de Interface com Usuário** (destacado em verde) é o responsável pela interação com o usuário. Esse módulo possui as seguintes classes:

- **Interfacelogin:** possui a função login_usuario encarregada de receber os dados de login do usuário (nome de usuário e senha) e enviá-los para o ControladorUsuario para validação. O ControladorUsuario, após a validação bem-sucedida, utiliza a função abrir_tela para chamar a interface apropriada (InterfaceAdmin ou InterfaceUsuario) com base no tipo de usuário. Tais interfaces informam as opções de ações que cada tipo de usuário pode realizar.
- **ControladorUsuario:** possui a função validar_login, que verifica se o usuário informado está cadastrado na base de dados usuário do banco de dados SQLite, e a função abrir_tela, que dependendo do tipo de usuário (administrador ou convencional), irá abrir a InterfaceAdmin ou a InterfaceUsuario.
- **InterfaceAdmin:** possui a função mostrar_opções que exibe as ações permitidas para um usuário do tipo administrador, que é manipulação da base de dados usuário e da base de dados bancos de dados, além da realização de consultas.
- **InterfaceUsuario:** possui a função mostrar_opções que exibe as ações permitidas para um usuário do tipo convencional, ou seja, realização de consultas.
- **Mensagem:** possui os atributos privados csql (abreviação para consulta em SQL) e cnosql (abreviação para consulta NoSQL), e suas funções de manipulação (set's e get's), e os atributos consultaLN (abreviação para consulta em linguagem natural) e resposta, inicializadas na função __init__, possuem as funções get's. A função __str__ gera um texto contendo as consultas em linguagem natural, em SQL e em linguagem MongoDB.

O **módulo de tradução** (destacado em vermelho) realiza a tradução das consultas

em linguagem natural para linguagens formais utilizadas pelos bancos de dados. Esse módulo possui a seguinte classe:

- **Traduz:** que contém uma função para cada ferramenta utilizada para tradução, por exemplo, a função ln2sql, recebe a consulta, trata, chamar a ferramenta LN2SQL, passa a consulta tratada, recebe a resposta da ferramenta, trata a resposta e a retorna para quem a invocou. A função sqltomongo realiza as mesmas ações, porém chama a ferramenta *SQL To MongoDB Query Converter*.

O **módulo de comunicação** (destacado em laranja) facilita a interação com os SGBDs, executando as consultas e gerenciando a comunicação entre o sistema e os bancos de dados. Esse módulo possui as seguintes classes:

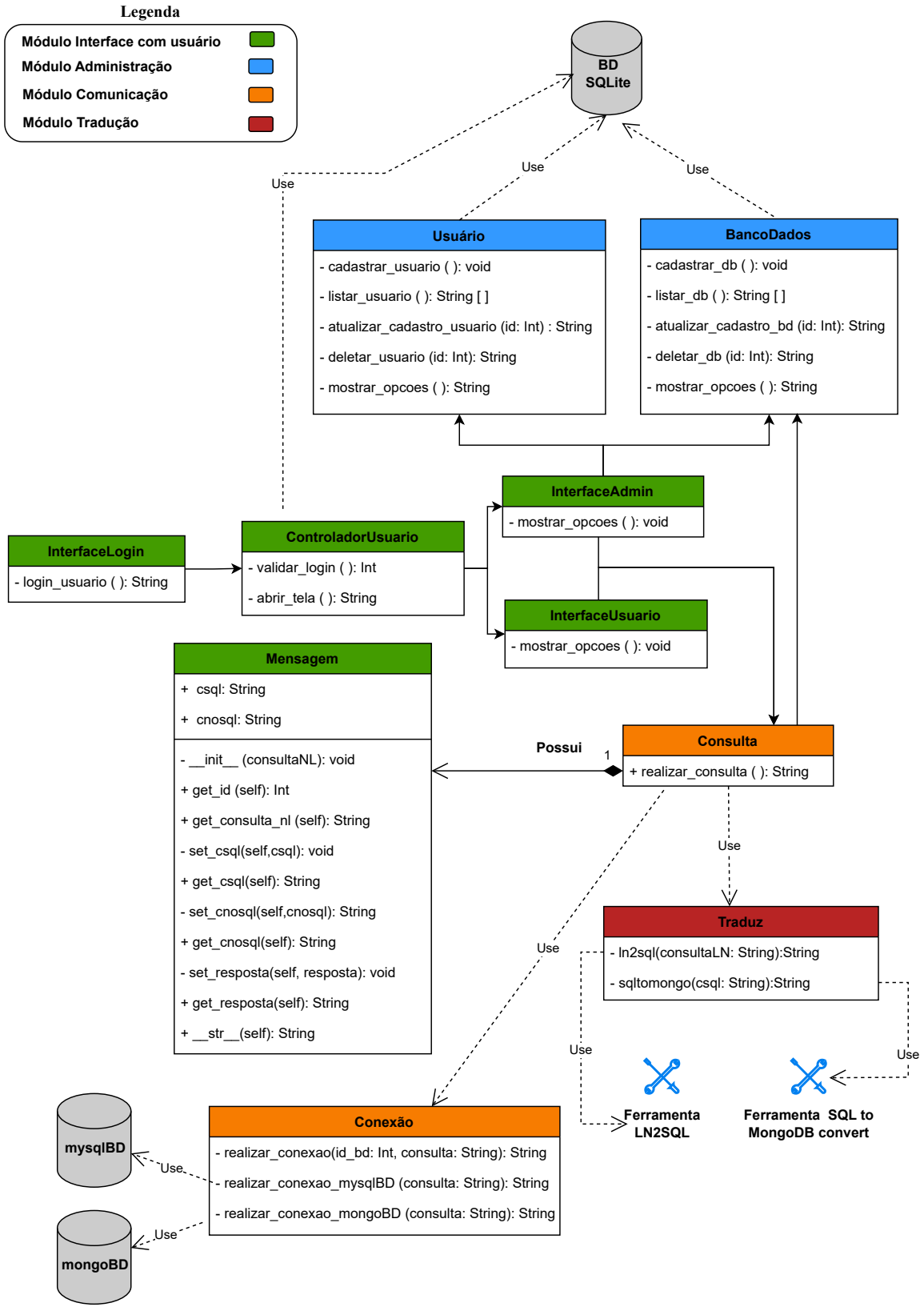
- **Consulta:** possui a função realizar_consulta que instancia mensagem, invoca traduz para realizar as traduções e as grava em mensagem, chama conexão para realizar a conexão com os bancos de dados, encaminha as consultas para os bancos, recebe e grava as respostas em mensagem.
- **Conexão:** Estabelece a conexão com os bancos de dados e executa as consultas. Para cada banco de dados, existe uma função específica para realizar a conexão, a função realiza_conexão_mysqlBD realiza a conexão com o BD MySQL e a função realiza_conexão_mongoBD realiza a conexão com o MongoBD.

O módulo de administração (destacado em azul), gerencia os usuários e os bancos de dados, permitindo operações de cadastro, listagem, atualização e exclusão.

- **BancoDados:** possui as funções que permitem ao o administrador cadastrar, listar, atualizar e deletar bancos de dados.
- **Usuário:** possui as funções que permitem ao o administrador cadastrar, listar, atualizar e deletar usuários.

O fluxo de execução é o seguinte, o usuário informa as suas credenciais através da interface de login. Inicialmente o sistema conta com dois usuários: um usuário administrador com nome de usuário "admin"e senha "12345"e um usuário convencional com nome de usuário "user1"e senha "12345". A interface de login envia ao controlador de usuário os dados de acesso recebidos para realizar validação de login com bases nas informações guardadas no banco de dados de metadados. Após validar o login e verificar o tipo de usuário, o controlador chama a interface admin para usuários administradores, e a interface usuários, para usuários convencionais.

Figura 5 – Diagrama de classes do modelo de arquitetura proposta



Fonte: Elaborada pela autora(2024)

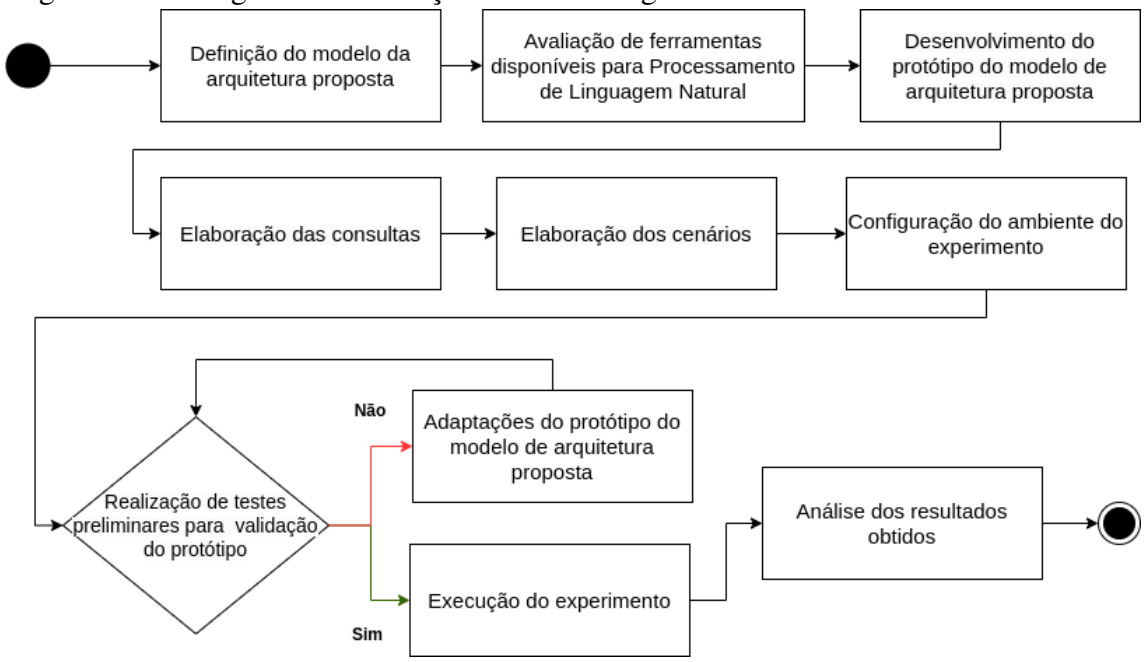
Através da interface admin o administrador pode cadastrar, listar, atualizar e deletar bancos de dados ou usuários. Os dados cadastrados dos bancos de dados e dos usuários são salvos em um banco de dados relacional SQLite. O administrador também pode realizar consultas. Ao optar por realizar uma consulta, a classe consulta é invocada, essa classe cria um objeto mensagem para guardar todas as informações, como a consulta em linguagem natural (informada pelo usuário), a tradução para uma consulta em SQL e/ou a tradução para uma consulta em MongoDB, além da resposta da consulta. A classe consulta também invoca a classe tradução, que realiza as traduções utilizando as ferramentas e retorna a consulta em SQL e MongoDB, para serem salvos em mensagem. Com as consultas traduzidas, a classe consulta invoca a classe conexão, que estabelece a comunicação com o banco de dados escolhido pelo usuário, envia a consulta na linguagem correspondente a linguagem do banco de dados e retorna a resposta para ser salva em mensagem e que será enviada para o usuário. O usuário convencional tem acesso a interface de usuário, e pode realizar consulta, porém não pode cadastrar, listar, atualizar e deletar bancos de dados ou usuários.

Para adicionar um novo banco de dados relacional, basta criar uma função na classe conexão que possua os dados que possibilitem estabelecer a conexão com esse novo banco. O administrador também deve cadastrar os metadados desse BD no BD SQLite para fins de controle. Para adicionar um novo banco de dados não relacional, além da criação da função para realizar a conexão, na classe conexão e o cadastro dos metadados desse BD no BD SQLite, uma nova ferramenta capaz de realizar a tradução das consultas em linguagem natural para a linguagem do novo BD deve ser adicionada a classe traduz. Para adicionar uma nova ferramenta, uma função capaz de invocar e receber a tradução realizada por essa ferramenta deve ser criada na classe traduz, e a deve ser adicionada a uma pasta com seu nome dentro do módulo tradução.

5 METODOLOGIA

Neste capítulo, são apresentados os passos realizados no desenvolvimento deste trabalho. Inicialmente alguns passos comuns de pesquisa foram executados a fim de estabelecer os objetivos que serviriam para guiar essa pesquisa, sendo eles: definição do tema de pesquisa; revisão bibliográfica; formalização dos objetivos da pesquisa; e um estudo de trabalhos relacionados. Os demais passos da pesquisa são apresentados na Figura 6 e comentados mais detalhadamente em seguida.

Figura 6 – Fluxograma de execução da metodologia



Fonte: Elaborada pela autora(2024)

5.1 Definição do modelo da arquitetura proposta

A arquitetura proposta foi elaborada para suportar mudanças decorrentes do avanço do estado da arte, inclusão de novos bancos de dados, algoritmos adicionais e/ou ferramentas de tradução de linguagem natural para consulta formal de bancos de dados, bem como para o reconhecimento de novos idiomas nas consultas de entrada. A abordagem adotada envolve a criação de módulos com funcionalidades distintas e bem definidas, simplificando a atualização e adaptação da arquitetura. Para assegurar a adaptabilidade da proposta, o código-fonte foi disponibilizado, e a infraestrutura dos experimentos foi implementada em um cluster de computadores, com a possibilidade de ser implementada também em serviços de infraestrutura de computação

em nuvem.

5.2 Avaliação de ferramentas disponíveis para Processamento de Linguagem Natural

Inicialmente, desejava-se encontrar uma única ferramenta ou algoritmo capaz de traduzir consultas em linguagem natural para linguagens formais de bancos de dados relacionais e não relacionais, porém a busca não obteve êxito, entretanto, o seguinte trabalho (Dar *et al.*, 2019) foi encontrado. O artigo (Dar *et al.*, 2019) faz uma revisão do estado da arte das estruturas de tradução de linguagem natural para linguagens de consulta estruturadas e não estruturadas. Neste artigo, foram revisadas 47 estruturas, publicadas entre 2008 e 2018. Dessas 47 estruturas, 35 foram discutidas no trabalho, porém nenhuma solução discutida no artigo lida com bancos de dados relacionais e não relacionais simultaneamente, o que motivou a busca por alternativas, como: soluções para tradução de linguagem natural para linguagem formal de banco de dados relacionais (linguagem SQL), linguagem natural para linguagem formal para banco de dados não relacional (mais especificamente para o banco de dados MongoDB), ou tradução da linguagem SQL para a linguagem formal do banco de dados não relacionais MongoDB, sendo encontradas as seguintes ferramentas LN2SQL, Seq2SQL e SQL To MongoDB Query Converter. Considerando a facilidade de uso, eficiência, e disponibilidade do código-fonte, as ferramentas escolhidas para serem utilizadas foram: LN2SQL e *SQL To MongoDB Query Converter*.

5.3 Desenvolvimento do protótipo do modelo de arquitetura proposta

O protótipo do modelo de arquitetura aqui proposto pode ser observado na Figura 1. O modelo de arquitetura escolhido para a implementação foi o modelo cliente-servidor, onde, segundo (Kurose; Ross, 2010), o servidor é responsável por atender requisições por demanda para o processamento de consultas em linguagem natural de múltiplos clientes e o cliente, é o responsável por realizar e estabelecer um canal de comunicação para as requisições serem enviadas ao servidor para fins de processamento. O modelo de arquitetura proposto apresenta quatro módulos do lado servidor: i) **Módulo de Interface de Usuário**, ii) **Módulo de Tradução**, iii) **Módulo de Comunicação** e o iv) **Módulo de Administração**. Os módulos foram implementados na linguagem de programação Python e cada um exerce uma função específica na arquitetura. O **Módulo de Interface de Usuário** e o responsável pela interação com o usuário, recebendo a consulta e mostrando o resultado da mesma, entretanto, nesse primeiro momento,

a interface gráfica não foi implementada. Para permitir a validação da arquitetura, as entradas do sistema foram inicialmente fornecidas por meio de uma função que realiza a leitura das consultas em linguagem natural em um arquivo de texto e as encaminha para a interface e as saídas são direcionadas para gravação em outro arquivo de texto. O **Módulo de Tradução** é o responsável por hospedar as ferramentas de tradução, e para o experimento deste trabalho, serviu-se da ferramenta **LN2SQL** publicada em (Ferrero, 2022) e descrita no artigo (Couderc; Ferrero, 2015b) para tradução das consultas em LN para linguagem SQL e a ferramenta **SQL To MongoDB Query Converter**, publicada em (Russell, 2022), para a tradução das consultas em linguagem SQL para linguagem MongoDB. O **Módulo de Comunicação** estabelece a conexão e envia as consultas aos bancos de dados e o **Módulo de Administração** faz o gerenciamento de permissões e usuários.

5.4 Elaboração das consultas

Para o experimento, primeiramente foram elaboradas 13 consultas em LN. As consultas em linguagem natural (LN) utilizadas foram elaboradas com base no conhecimento prévio do esquema do banco de dados e são exclusivamente do tipo agregação, envolvendo funções como COUNT, SUM, MAX, MIN e AVG. As funções de agregação são empregadas para sumarizar informações ou criar agrupamentos de subconjuntos de dados.

O esquema do banco de dados relacional foi planejado e construído com base em um conjunto de dados de *Open Government Data* (OGD) disponível em https://catalog.data.gov/dataset. Este conjunto de dados consiste em uma única tabela contendo informações sobre o histórico de frequência diária por escola. As colunas incluem identificação da escola, data, ano escolar, matrícula do aluno, total de presenças, total de faltas e total de liberações. Os mesmos dados foram adaptados e enviados para o banco de dados não relacional.

Posteriormente, as consultas elaboradas em LN foram transcritas para linguagem SQL e para a linguagem do banco de dados MongoDB. As tabelas contendo as consultas podem ser observadas no apêndice A (consultas em linguagem natural), apêndice B (consultas em linguagem SQL) e apêndice C (consultas em linguagem MongoDB).

5.5 Definição dos cenários

Visando confirmar a hipótese, foram desenvolvidos 18 cenários que podem ser observados na Tabela 2, dos quais 9 cenários tiveram como entrada as consultas em LN e foram executados através da proposta e 9 cenários com entrada de consultas em Linguagem Formal (LF) executados diretamente nos SGBDs. Os cenários são resultados da combinação das métricas, fatores e dos níveis estabelecidos a fim de comprovar que a utilização da proposta não adiciona atraso significativo nas consultas aos BDs. Como métrica deste trabalho, foi escolhido o tempo de execução das consultas e os fatores observados foram: números de acessos simultâneos, representados pelo aumento do tamanho das cargas (20, 100 e 200 consultas); e o volume de dados, testado com bases de dados de 10GB, 50GB e 100GB, considerando espaço de armazenamento ocupado no banco de dados MySQL que é povoado primeiro e depois tem os dados transferidos para o banco de dados MongoDB. Em relação à carga de consultas, considere que em cada carga, metade das consultas são direcionadas para o BD relacional e metade para o BD não-relacional.

Tabela 1 – Cenários

Arquitetura proposta			**SGBDs**		
Cenário	**Tamanho do dataset**	**Carga (consultas)**	**Cenário**	**Tamanho do dataset**	**Carga (consultas)**
C1	10GB	20	**C4**	10GB	20
C2	10GB	100	**C5**	10GB	100
C3	10GB	200	**C6**	10GB	200
C7	50GB	20	**C10**	50GB	20
C8	50GB	100	**C11**	50GB	100
C9	50GB	200	**C12**	50GB	200
C13	100GB	20	**C16**	100GB	20
C14	100GB	100	**C17**	100GB	100
C15	100GB	200	**C18**	100GB	200

Fonte: Elaborado pela autora (2023).

5.6 Configuração do ambiente do experimento

Um ambiente de testes foi configurado para a execução do experimento. Inicialmente a configuração do ambiente da proposta foi realizada no ambiente de computação em nuvem da AWS, porém, em decorrência aos custos operacionais para manter os experimentos executados neste provedor de serviços em nuvem, uma estrutura foi montada em um cluster proprietário. O cluster proprietário contém 3 Máquinas Virtuais (VM), ambas com sistemas operacionais ubuntu 22.04.1 LTS, CPU (4vcpus, 16 GB) e disco para armazenamento de 200 GB. Uma máquina

possuía o banco de dados relacional, em outra máquina estava o banco de dados não relacional e a terceira máquina hospedava o servidor, não foi necessário alocar máquinas clientes, pois a entrada era fornecida através de um arquivo de texto. A virtualização do cluster foi feita através da ferramenta Proxmox, que realiza virtualização com o hypervisor KVM no Linux.

5.7 Realização de testes preliminares para validação do protótipo

Após a implementação do protótipo e da configuração do ambiente de execução do experimento, utilizando-se do *dataset* de 10GB e das ferramentas de tradução (LN2SQL e *SQL To MongoDB Query Converter*) foram disparadas 3 repetições de cada tamanho de carga, ou seja, 3 repetições da carga de 20 consultas, 3 repetições da carga de 100 consultas e 3 repetições da carga de 200 consultas através da arquitetura proposta e observado se a conexão com os bancos estava estabelecida, se as respostas das consultas realizadas retornavam (sendo gravadas no arquivo de texto) e se as respostas das consultas estavam coerentes. Caso identificado algum erro, a etapa de adaptações do protótipo era executada;

5.8 Adaptações do protótipo do modelo de arquitetura proposta ou execução do experimento

A etapa de adaptações do protótipo do modelo de arquitetura proposta é executada quando na etapa realização de testes preliminares para validação do protótipo é detectado algum erro, como a falha de conexão com os bancos de dados, a resposta da consulta não retorna ou retorna em branco. O retorno de uma consulta como resposta em branco significava geralmente a falha na conexão ou erro de escrita na consulta. Tais ocorrências fazem necessário a verificação do código e a realização de novos testes preliminares. Caso nenhum erro for detectado, é realizada a execução do experimento.

5.9 Análise dos resultados obtidos

Nesta etapa, foram conduzidas duas análises distintas. A primeira teve como objetivo verificar a precisão das traduções realizadas pelas ferramentas e se os dados retornados eram os esperados. Já a segunda análise visou comparar os tempos de execução das consultas em linguagem natural (LN) executadas utilizando a solução proposta com as consultas executadas diretamente nos SGBDs na linguagem formal de banco de dados. Essa análise tinha o propósito

de confirmar a hipótese central deste trabalho: que a adoção da arquitetura proposta não resultaria em atrasos significativos nas consultas aos bancos de dados.

Durante este experimento, foram avaliados os impactos do aumento do número de acessos dos usuários e do crescimento da base de dados. A comparação entre os tempos de execução das cargas utilizando a solução proposta e aqueles das cargas executadas diretamente nos SGBDs, sem tradução, foi realizada através da análise dos tempos de resposta das consultas em LN utilizando a solução, em contraposição às consultas nos SGBDs na linguagem de consulta de dados.

Conforme mencionado, a avaliação desta solução foi conduzida por meio de uma análise de desempenho comparativa entre a proposta e os SGBDs "puros", utilizando 13 consultas formuladas em linguagem natural e traduzidas para SQL e Linguagem MongoDB (os detalhes das consultas estão disponíveis nos apêndices deste trabalho). As cargas possuem os tamanhos de 20 consultas, 100 consultas e 200 consultas, podendo haver repetição ou não das consultas na carga. Cada carga foi executada cinco vezes, e a média dos resultados de cada execução foi considerada como valor de referência para o cenário. Os experimentos foram configurados para disparar as consultas seguindo uma distribuição de Poisson com λ =(75).

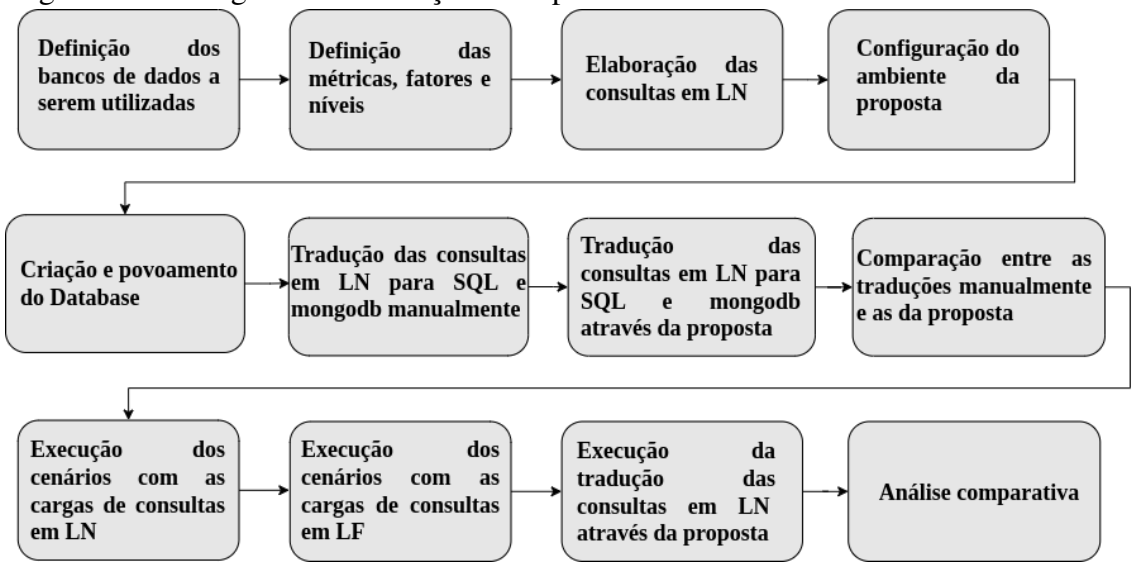
O valor de λ =(75) foi determinado após analisar o tempo de execução da carga de 20 consultas em um *dataset* de 10 GB, considerando um intervalo de tempo de 3600 segundos (1 hora). Este valor foi mantido constante para todos os cenários do experimento, a fim de observar o impacto do aumento do número de acessos dos usuários e do crescimento da base de dados, sendo adotado como configuração padrão.

6 AVALIAÇÃO EXPERIMENTAL E RESULTADOS

Neste capítulo, são apresentados os passos de execução do experimento deste trabalho. Ressaltando que este trabalho não tem como objetivo implementar uma ferramenta de tradução, e sim, propor uma arquitetura capaz de suportar tais ferramentas. Porém, com o intuito de analisar e validar a proposta, duas ferramentas foram utilizadas para compor o módulo de tradução, LN2SQL e SQL to MongoDB Query Converter. Para o experimento, 13 consultas foram formuladas, sendo do tipo agregação (*COUNT, SUM, MAX, MIN e AVG*) e dois tipos de análises foram realizadas.

Os passos de execução do experimento deste trabalho são mostrados na Figura 7 e discutidos mais detalhadamente em sequência:

Figura 7 – Fluxograma de execução do experimento



Fonte: Elaborada pela autora(2024)

A definição dos bancos de dados a serem utilizados para compor o ambiente de experimento foram: Mysql, como representante dos bancos relacional, e MongoDB, como representante dos bancos não-relacionais. Para a escolha, foram considerados, os BDs mais utilizados no mundo (segundo o site DB-Enginee o banco de dados MySql é o segundo banco de dados mais utilizado, sendo o segundo banco de dados relacional mais utilizado, e o MongoDB é o quinto banco de dados, sendo o primeiro banco de dados não-relacional na lista.), os BDs utilizados nos trabalhos relacionais, onde Malik e Rishi (2015), Li e Gu (2019), Vathy-Fogarassy e Hugyák (2017) utilizam o MySql e Mondal *et al.* (2019), Pradeep *et al.* (2019), Li e Gu (2019), Vathy-Fogarassy e Hugyák (2017) e Bjeladinovic *et al.* (2020) utilizam o MongoDB e a

familiaridade com os mesmos. As versões utilizadas foram: MySQL versão 8.0.30 e o MongoDB versão 4.2.18.

A definição das métricas, fatores e dos níveis foram estabelecidos a fim de comprovar que a utilização da proposta não adiciona atraso significativo nas consultas aos BDs. Como métrica deste trabalho, foi escolhido o tempo de execução das consultas e os fatores observados foram: números de acessos simultâneos, representados pelo aumento do tamanho das cargas (20, 100 e 200 consultas); e o volume de dados, testado com bases de dados de 10GB, 50GB e 100GB, considerando espaço de armazenamento ocupado no banco de dados MySQL que é povoado primeiro e depois tem os dados transferidos para o banco de dados MongoDB. Em relação à carga de consultas, considere que em cada carga, metade das consultas são direcionadas para o BD relacional e metade para o BD não-relacional.

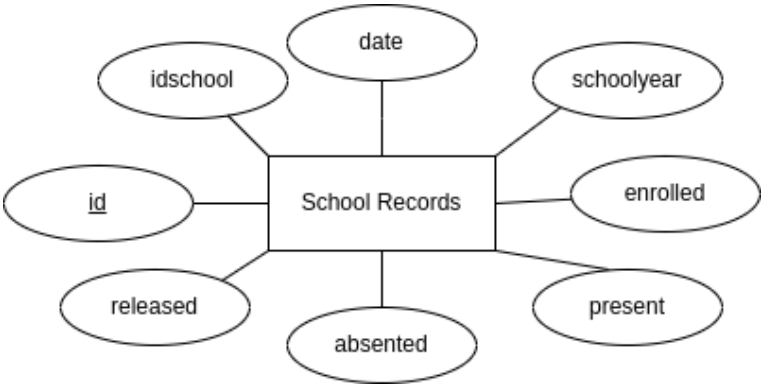
A elaboração das consultas em LN utilizadas foram formuladas com base no conhecimento prévio das bases de dados e são exclusivamente do tipo agregação (*COUNT, SUM, MAX, MIN e AVG*). As funções de agregação são usadas para resumir informações ou gerar agrupamentos de subconjuntos. Esse trabalho foca em consultas referentes a uma única tabela ou *single table*. A proposta apresentada não oferece suporte às operações CRUD(*Create, Read, Update e Delete*). O mesmo conjunto de dados foi armazenado nos dois bancos de dados, sendo realizadas adaptações para o banco de dado não-relacional.

Inicialmente a configuração do ambiente da proposta foi realizada no ambiente de computação em nuvem da AWS, porém, em decorrências aos custos operacionais para manter os experimentos executando neste provedor de serviços em nuvem, uma estrutura foi montada em um cluster proprietário, para execução dos experimentos. O cluster proprietário contém 3 Maquinas Virtuais(VM), ambas com sistemas operacionais ubuntu 22.04.1 LTS, CPU (4vcpus, 16 GB) e disco para armazenamento de 200 GB. A virtualização do cluster foi feita através da ferramenta Proxmox, que realiza virtualização com o hypervisor KVM no Linux.

A criação e povoamento do *Databases* foram realizadas em três etapas. Na primeira etapa, o esquema do banco de dados relacional foi planejado e construído utiliza como modelo um *dataset* de OGD disponível em https://catalog.data.gov/dataset e contendo uma única tabela (*single table*) com informações sobre o histórico de frequência diária por escola, através das colunas: identificação da escola, data, ano escolar, matrícula do aluno, total de presenças, total de faltas e total de liberações. O esquema do banco de dados histórico escolar (*School Records*) criado é apresentado na Figura 8, onde temos os atributos id (chave primaria), identificação da

escola (idschool), data (*date*), ano escolar (*schoolyear*), matrícula (*enrolled*), presenças (*present*), faltas (*absented*) e liberações (*released*). O atributo id foi adicionado com intuito de permitir a migração dados para o banco de dados MongoDB. Na migração dos dados, as tuplas são transformados em documentos individuais.

Figura 8 – Esquema do banco de dados Histórico Escolar



Fonte: Elaborada pela autora(2024)

Na segunda etapa, um benchmark foi desenvolvido na linguagem Java, para povoar o banco de dados relacional. Esse benchmark, recebe como entrada, o endereço do BD relacional que irá receber os dados, usuário e senha de acesso ao BD relacional e o tamanho da base de dados que deseja criar. Os dados são criados a partir de valores randômicos. Na terceira etapa, um código, desenvolvido em Python, se conecta ao BD relacional e ao BD não relacional para transferir os dados de uma base para outra. Um ponto a observar é que este trabalho usa a replicação total, com os mesmos nomes de colunas e de atributos para os dois BDs utilizados no experimento (MySQL e MongoDB), evitando ambiguidade no acesso às bases.

A tradução das consultas em LN para linguagem SQL e MongoDB foi manualmente realizada com o intuito de verificar se as ferramentas retornavam consultas válidas e se essas consultas, quando submetidas aos bancos, retornavam os resultados esperados. Posteriormente foi realizado a tradução das consultas em LN para linguagem SQL e MongoDB através da proposta e com o resultado das traduções foi realizado a comparação entre as traduções feitas manualmente e as traduções feitas através da proposta.

No total foram executados 18 cenários, dos quais 9 cenários tiveram como entrada as consultas em LN e 9 cenários como entrada de consultas em LF. Para realizar a análise comparativa, foram realizados a execução dos cenários com as cargas de consultas em LN e captura do tempo de execução de cada repetição para o cálculo do tempo médio de execução da carga através da proposta; a execução dos cenários com as cargas de consultas em linguagem

formal (SQL e MongoDB) e captura do tempo de execução de cada repetição para o cálculo do tempo médio de execução da carga diretamente direcionada aos SGBDs. Os tempos médios de execução das cargas foram usados como métrica para comparação e a execução da tradução das consultas em LN para linguagem formal (SQL e MongoDB) através das ferramentas utilizadas e captura do tempo de execução de cada repetição foi realizada para ter-se o conhecimento do tempo médio que cada carga demorou para ser traduzida.

A análise comparativa entre os tempos de execução das cargas através da proposta e o tempo de execução das cargas diretamente nos SGBD's, sem tradução, foi realizada através da comparação entre o tempo de respostas das consultas em LN realizadas através da solução e das consultas realizadas diretamente nos SGBDs em linguagem de consulta de dados, a fim de confirmar que a solução não gera atraso de tempo significativo. Como mencionado, a análise desta solução foi realizada através da análise de desempenho comparativa entre a proposta e aos SGBDs "puros", através de 13 consultas formuladas em linguagem natural e traduzidas para SQL e para Linguagem MongoDB (as consultas podem ser encontradas nos apêndices deste trabalho). As cargas possuem os tamanhos de 20 consultas, 100 consultas e 200 consultas, as consultas podendo se repetir ou não dentro da carga. Cada carga é repetida 5 vezes, e a média dos resultados das repetições de cada carga tem seus valores tomados como valor de referência para o cenário. Os experimentos foram configurados para disparar as consultas conforme distribuição de Poisson com λ=(75). O valor de λ=(75) foi definido ao analisar o tempo de execução da carga de 20 consultas, no *dataset* de 10GB, considerando o intervalo de tempo de 3600 segundos (1 Hora). E como o experimento deste trabalho visa observar o impacto do aumento do número de acesso dos usuários e do crescimento da base de dados, esta λ foi mantida para todos os cenários e tomada como configuração.

Dois tipos de análises foram realizadas. A primeira análise foi feita para verificar se as traduções realizadas pelas ferramentas geraram uma consulta traduzida corretamente, e se os dados retornados eram os esperados (Passo 8). A segunda análise foi realizada para comparar o desempenho dos tempos de execução das consultas em LN realizadas através da solução e das consultas realizadas diretamente nos SGBDs em linguagem formal de dados a fim de confirmar a hipótese desse trabalho: A utilização da arquitetura proposta não gera atraso significativo nas consultas aos bancos de dados (passo 12).

Nesse experimento foi observado o impacto do aumento do número de acesso dos usuários e do crescimento da base de dados. Outro detalhe é que mesmo a carga das consultas

sendo a mesma em todos os testes, existem fatores fora do controle, como, por exemplo, outros processos (de funcionamento interno inclusive) executando na máquina física que hospeda as MVs e que podem contribuir para uma variação de desempenho.

Na Tabela 2 são apresentados os cenários do experimento, observando que para cada tamanho da base de dados, as cargas de 20, 100 e 200 consultas são disparadas em linguagem natural (através da arquitetura proposta) e em linguagem formal (diretamente nos SGBDs).

Tabela 2 – Cenários

Arquitetura proposta			**SGBDs**		
Cenário	**Tamanho do dataset**	**Carga (consultas)**	**Cenário**	**Tamanho do dataset**	**Carga (consultas)**
C1	10GB	20	**C4**	10GB	20
C2	10GB	100	**C5**	10GB	100
C3	10GB	200	**C6**	10GB	200
C7	50GB	20	**C10**	50GB	20
C8	50GB	100	**C11**	50GB	100
C9	50GB	200	**C12**	50GB	200
C13	100GB	20	**C16**	100GB	20
C14	100GB	100	**C17**	100GB	100
C15	100GB	200	**C18**	100GB	200

Fonte: Elaborado pela autora (2023).

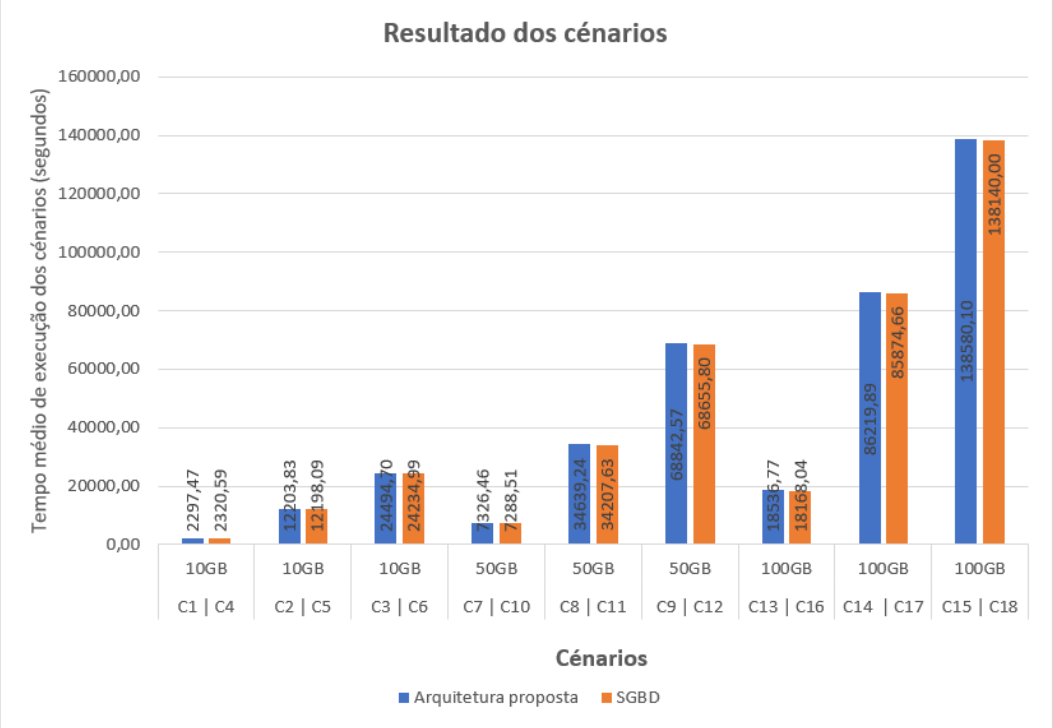
6.1 Resultados obtidos

A Figura 9 exibe os tempos médios de execução de todos os cenários. No gráfico podemos observar que a diferença dos tempos médio de execução dos cenários através da proposta e das cargas executadas diretamente nos SGBDs são quase imperceptíveis. Outro fator que justifica essa diferença é que os experimentos foram configurados para disparar as consultas das cargas conforme distribuição de Poisson com λ =(75). A distribuição de Poisson é uma distribuição de probabilidade discreta que calcula as ocorrências de determinado evento em determinado intervalos de tempo.

Comparando os resultados dos cenários C1 e C4, a execução das cargas através da proposta foram 23,12 segundos mais rápida do que a execução direta; C2 e C5, a execução das cargas através da proposta foram 5,73 segundos mais demorada do que a execução direta; C3 e C6, a execução das cargas através da proposta foram 259,71 segundos mais demorada do que a execução direta; C7 e C10, a execução das cargas através da proposta foram 26,28 segundos mais demorada do que a execução direta; C8 e C11, a execução das cargas através da proposta foram 431,61 segundos mais demorada do que a execução direta; e nos C9 e C12, a execução das cargas através da proposta foram 186,77 segundos mais demorada do que a execução direta;

C13 e C16, a execução das cargas através da proposta foram 368,73 segundos mais demorada do que a execução direta; C14 e C17, a execução das cargas através da proposta foram 345,22 segundos mais demorada do que a execução direta; C15 e C18, a execução das cargas através da proposta foram 440,12 segundos mais demorada do que a execução direta.

Figura 9 – Análise comparativa de tempo médio de execução por cenário



Fonte: Elaborada pela autora (2023).

A Tabela 3 mostra os resultados expressos em números para comparações mais detalhadas das médias.

Outra análise sobre os dados também foi executada, utilizando o Teste de Wilcoxon podemos comprovar que os tempos de execução dos cenários através da proposta não são tão diferentes dos tempos de execução dos cenários executados diretamente nos SGBD's. Para realização dessa análise, um código foi implementado em python usando a função wilcoxon() da biblioteca SciPy. A função usa as duas amostras como argumentos e retorna a estatística calculada e o valor-p. A suposição padrão (hipótese nula) é que as duas amostras têm a mesma distribuição, que pode comprovar a tese de que não há diferença significativamente entre os tempos de execução através da proposta e diretamente nos SGBD's. O código utilizado foi o disponibilizado em https://machinelearningmastery.com/nonparametric-statistical-significance-tests-in-python/ e mostrado no Código-fonte 1 – Código do Teste de Wilcoxon. O vetor data1 possui os tempos de execução de cada repetição dos cenários C1, C2, C3, C7, C8, C9, C13, C14 e C15

Tabela 3 – Tempo médio de execução por cenário e desvio padrão

Arquitetura proposta				
Cenário	**Tamanho do dataset**	**Carga (consultas)**	**Tempo médio de execução do cenário**	**Desvio Padrão**
C1	10GB	20	2297,47	14,96
C2	10GB	100	12203,83	106,11
C3	10GB	200	24494,7	63,90
C7	50GB	20	7314,79	42,80
C8	50GB	100	34639,24	246,90
C9	50GB	200	68842,57	1080,36
C13	100GB	20	18536,77	339,60
C14	100GB	100	86219,89	2713,33
C15	100GB	200	138580,11	752,60
SGBDs				
Cenário	**Tamanho do dataset**	**Carga (consultas)**	**Tempo médio de execução do cenário**	**Desvio Padrão**
C4	10GB	20	2320,59	37,52
C5	10GB	100	12198,09	48,15
C6	10GB	200	24234,99	112,98
C10	50GB	20	7288,51	34,35
C11	50GB	100	34207,63	144,79
C12	50GB	200	68655,8	173,64
C16	100GB	20	18168,04	120,41
C17	100GB	100	85874,66	8507,22
C18	100GB	200	138139,99	1084,70

Fonte: Elaborado pela autora (2023).

(Cenários executados através da proposta) e o vetor data2 possui os tempos de execução de cada repetição dos cenários C4, C5, C6, C10, C11, C12, C16, C17 e C18 (Cenários executados diretamente nos SGBD's). A saída da execução do código foi *"fail to reject H0"*, o que significa "Mesma distribuição (falha em rejeitar H0(hipótese nula))", ou seja, os dados possuem a mesma distribuição.

Código-fonte 1 – Código do Teste de Wilcoxon

```
# Wilcoxon signed-rank test
from numpy.random import seed
from numpy.random import randn
from scipy.stats import wilcoxon
# seed the random number generator
seed(1)
# generate two independent samples
data1=[2324.97, 2299.93, 2293.62, 2283.70, 2285.12, 12155.50, 12114.46,
    12126.68, 12219.21, 12403.28, 24490.28, 24480.95, 24475.81, 24415.33,
    24611.12, 7268.07, 7340.21, 7295.66, 7280.02, 7389.97, 34180.50,
    34802.64, 34619.64, 34890.73, 34702.69, 70377.86, 69920.79, 67811.83,
    68027.09, 68075.26, 17990.97, 18352.17, 18930.00, 18834.21, 18576.48,
    82729.49, 84537.63, 86405.44, 86553.82, 90873.06, 137114.58,
    139129.40, 139033.75, 138998.15, 138624.67]
data2=[2286.04, 2364.11, 2363.79, 2275.47, 2313.53, 12257.40, 12168.42,
    12126.10, 12242.88, 12195.65, 24273.61, 24097.07, 24195.68, 24177.49,
    24431.10, 7300.59, 7281.57, 7256.53, 7255.44, 7348.42, 34104.66,
    34210.06, 34022.71, 34252.16, 34448.54, 68393.06, 68769.88, 68899.25,
    68558.67, 68658.13, 18242.68, 18124.78, 17958.08, 18207.21, 18307.43,
    83243.00, 98669.79, 77977.96, 92687.85, 76794.70, 137315.01,
    138500.82, 138786.08, 139571.87, 136526.18]
# compare samples
stat, p = wilcoxon(data1, data2)
print('Statistics=%.3f, p=%.3f' % (stat, p))
# interpret
alpha = 0.05
if p > alpha:
    print('Same distribution (fail to reject H0)')
else:
    print('Different distribution (reject H0)')
```

Código-fonte 2 – Resultado do teste de Wilcoxon

```
Statistics=165.000, p=0.171
Same distribution (fail to reject H0)
```

7 CONCLUSÃO

Este trabalho apresentou um modelo de arquitetura adaptável a múltiplos tipos de bancos de dados, que armazenem dados homogêneos ou heterogêneos e aceitem consultas de entrada em linguagem natural, proporcionando o acesso às informações de forma transparente a todos os que delas precisam, mesmo aqueles que não possuem conhecimento em linguagem de formal de banco de dados. Para isso, utilizou-se de estudos que envolveram BD relacionais e não relacionais, arquiteturas hibridas para tipos diferentes de bancos de dados e PLN. A arquitetura proposta foi desenvolvida para permitir adaptações decorrentes da evolução do estado da arte atual, adições de novos bancos de dados, novos algoritmos e/ou ferramentas de tradução de linguagem natural para linguagem formal de consulta de bancos de dados e reconhecimento de novos idiomas nas consultas de entrada. A estratégia utilizada foi a criação de módulos com funcionalidades bem definidas e separadas dos demais, o que facilita a atualização e adaptação da arquitetura. Para garantir que a proposta possa ser adaptada, o código-fonte foi disponibilizado em https://github.com/LuziaNOG/Arq-para-uso-de-BD-hibrido-adaptativo-a-ferramentas-de-PLN, e a infraestrutura utilizada nos experimentos foi implementada em um cluster de computadores, mas com possibilidade de implementação em serviços de computação em nuvem. Como trabalhos futuros, almeja-se: i) adicionar uma interface que permita receber como entrada consultas em outros idiomas além do inglês; ii) adicionar ferramentas que sejam capazes de traduzir consultas em linguagem natural para outras linguagens formais de bancos de dados, por exemplo, bancos de dados não relacionais baseados em grafos, orientado a coluna e/ou baseados em chave e valor; iii) adicionar suporte às operações do tipo CRUD e iv) adaptar a interface de entrada para recebimento de consultas em linguagem natural no formato de voz, facilitando ainda mais a utilização por parte de usuários não especialistas e tornando o sistema mais acessível.

7.1 Publicações

O atual trabalho resultou em uma publicação na Revista Sistemas e Mídias Digitais (RSMD), com o artigo intitulado como “Um Estudo Exploratório entre Banco de Dados NoSQL”, em 2020.

7.2 Agradecimentos

Este estudo foi parcialmente financiado pelo projeto "Plataforma de Big Data para Acelerar a Transformação Digital do Ceará"da Fundação Cearense de Apoio ao Desenvolvimento Científico e Tecnológico (FUNCAP nº. 04772551/2020).

REFERÊNCIAS

BATHLA, G.; RANI, R.; AGGARWAL, H. Comparative study of nosql databases for big data storage. **International Journal of Engineering & Technology**, v. 7, n. 2.6, p. 83–87, 2018.

BJELADINOVIC, S.; MARJANOVIC, Z.; BABAROGIC, S. A proposal of architecture for integration and uniform use of hybrid sql/nosql database components. **Journal of Systems and Software**, Elsevier, v. 168, p. 110633, 2020.

COUDERC, B.; FERRERO, J. fr2sql: interrogation de bases de données en français. In: **22ème Traitement Automatique des Langues Naturelles**. [*S. l.*: *s. n.*], 2015.

COUDERC, B.; FERRERO, J. fr2sql: interrogation de bases de données en français. In: **22ème Traitement Automatique des Langues Naturelles**. Caen, France: [*S. n.*], 2015.

DAR, H. S.; LALI, M. I.; DIN, M. U.; MALIK, K. M.; BUKHARI, S. A. C. Frameworks for querying databases using natural language: a literature review. **arXiv preprint arXiv:1909.01822**, 2019.

DESHPANDE, A. K.; DEVALE, P. R. Natural language query processing using probabilistic context free grammar. **International Journal of Advances in Engineering & Technology**, Citeseer, v. 3, n. 2, p. 568–573, 2012. ISSN 2231-1963.

FERRERO, J. **ln2sql**: a nlp tool to query a database in natural language. 2022. Disponível em: https://github.com/FerreroJeremy/ln2sql. Acesso em: 08 jan. 2022.

HAZBOUN, F. H.; OWDA, M.; OWDA, A. Y. A natural language interface to relational databases using an online analytic processing hypercube. **AI**, v. 2, n. 4, p. 720–737, 2021. ISSN 2673-2688. Acesso em: 18 jun. 2023. Disponível em: https://www.mdpi.com/2673-2688/2/4/43.

JOSE, B.; ABRAHAM, S. In: IEEE. **2017 International Conference on Networks & Advances in Computational Technologies (NetACT)**. [*S. l.*], 2017. p. 266–271.

KUNDA, D.; PHIRI, H. A comparative study of nosql and relational database. **Zambia ICT Journal**, v. 1, n. 1, p. 1–4, 2017.

KUROSE, J. F.; ROSS, K. W. **Redes de computadores e a internet**: uma abordagem top-down. 5a.. ed. [*S. l.*]: Pearson Addison Wesley, 2010. ISBN 9788588639973.

LAWRENCE, R. Integration and virtualization of relational sql and nosql systems including mysql and mongodb. In: IEEE. **2014 International Conference on Computational Science and Computational Intelligence**. [*S. l.*], 2014. v. 1, p. 285–290.

LI, C.; GU, J. An integration approach of hybrid databases based on sql in cloud computing environment. **Software: Practice and Experience**, Wiley Online Library, v. 49, n. 3, p. 401–422, 2019.

MALIK, A.; RISHI, R. A domain and language construct based mapping to convert natural language query to sql. **International Journal of Computer Applications**, Citeseer, v. 116, n. 4, 2015.

MONDAL, S.; MUKHERJEE, P.; CHAKRABORTY, B.; BASHAR, R. Natural language query to nosql generation using query-response model. In: IEEE. **2019 International Conference on Machine Learning and Data Engineering (iCMLDE)**. [*S. l.*], 2019. p. 85–90.

PRADEEP, T.; RAFEEQUE, P.; MURALI, R. Natural language to NoSQL query conversion using deep learning. 2019.

RUSSELL, V. **SQL to MongoDB query converter**. 2022. Disponível em: https://github.com/vincentrussell/sql-to-mongo-db-query-converter. Acesso em: 10 mar. 2022.

SANYAL, H.; SHUKLA, S.; AGRAWAL, R. Natural language processing technique for generation of sql queries dynamically. In: **2021 6th International Conference for Convergence in Technology (I2CT)**. [*S. l.*: *s. n.*], 2021. p. 1–6.

SINGH, G.; SOLANKI, A. An algorithm to transform natural language into sql queries for relational databases. **Selforganizology**, v. 3, n. 3, p. 100–116, 2016.

SOLANKE, G. B.; RAJESWARI, K. Sql to nosql transformation system using data adapter and analytics. In: IEEE. **2017 IEEE International Conference on Technological Innovations in Communication, Control and Automation (TICCA)**. [*S. l.*], 2017. p. 59–63.

STALLINGS, W. **Arquitetura e organização de computadores**. 8. ed. [*S. l.*]: São Paulo: Prentice Hall do Brasil, 2010.

TANENBAUM, A. S.; STEEN, M. van. **Distributed systems**: principles and paradigms. 2. ed. [*S. l.*]: Pearson Prentice Hall, 2007. ISBN 9780132392273.

VATHY-FOGARASSY, Á.; HUGYÁK, T. Uniform data access platform for sql and nosql database systems. **Information Systems**, Elsevier, v. 69, p. 93–105, 2017.

WAZLAWICK, R. S. **Metodologia de pesquisa para ciência da computação**. Rio de Janeiro: Elsevier Editora, 2009. ISBN 978-85-352-3522-7.

ZHAO, L.; ALHOSHAN, W.; FERRARI, A.; LETSHOLO, K. J.; AJAGBE, M. A.; CHIOASCA, E.-V.; BATISTA-NAVARRO, R. T. Natural language processing (nlp) for requirements engineering: A systematic mapping study. **arXiv preprint arXiv:2004.01099**, 2020.

ZHENG, L.; KWOK, W.-M.; AQUARO, V.; QI, X.; LYU, W. Evaluating global open government data: methods and status. In: **Proceedings of the 13th International Conference on Theory and Practice of Electronic Governance**. New York, NY, USA: Association for Computing Machinery, 2020. (ICEGOV 2020), p. 381–391. ISBN 9781450376747. Acesso em: 21 jun. 2023. Disponível em: https://doi.org/10.1145/3428502.3428553.

APÊNDICE A – QUADRO DE CONSULTAS FORMULADAS EM LINGUAGEM NATURAL

O Quadro 2 apresenta as 13 (treze) consultas elaboradas em Linguagem Natural (LN) utilizadas no experimento deste trabalho e formuladas com base no conhecimento prévio das bases de dados. As consultas são exclusivamente do tipo agregação (COUNT, SUM, MAX, MIN e AVG). As funções de agregação são usadas para resumir informações ou gerar agrupamentos de subconjuntos. Esse trabalho foca em consultas referentes a uma única tabela ou chamadas de *single table*.

Quadro 2 – Consultas em Linguagem Natural

ID	**Consulta em Linguagem Natural (inglês)**
1	What is the average present of SchoolRecords?
2	What is the average absented of SchoolRecords?
3	What is the average released of SchoolRecords?
4	What is the minimum present of SchoolRecords?
5	What is the minimum absented of SchoolRecords?
6	What is the minimum released of SchoolRecords?
7	What is the maximum present of SchoolRecords?
8	What is the maximum absented of SchoolRecords?
9	What is the maximum released of SchoolRecords?
10	What is sum absented from SchoolRecords?
11	What is sum released from SchoolRecords?
12	What is sum present from SchoolRecords?
13	Count how many distinct enrolled exist of SchoolRecords?

Fonte: Elaborado pela autora(2023).

APÊNDICE B – QUADRO DE CONSULTAS FORMULADAS EM LINGUAGEM FORMAL SQL

O Quadro 3 apresenta 13 (treze) consultas em linguagem SQL. Tais consultas são o resultado da tradução manual das consultas em Linguagem Natural (LN) que foram utilizadas no experimento deste trabalho. A tradução manual das consultas foi realizada com o intuito de verificar se as ferramentas retornavam consultas validas e se essas consultas, quando submetidas aos bancos, retornavam os resultados esperados.

Quadro 3 – Consultas em Linguagem Formal SQL

ID	**Consulta em Linguagem SQL**
1	SELECT AVG(SchoolRecords.present) FROM SchoolRecords;
2	SELECT AVG(SchoolRecords.absented) FROM SchoolRecords;
3	SELECT AVG(SchoolRecords.released) FROM SchoolRecords;
4	SELECT MIN(SchoolRecords.present) FROM SchoolRecords;
5	SELECT MIN(SchoolRecords.absented) FROM SchoolRecords;
6	SELECT MIN(SchoolRecords.released) FROM SchoolRecords;
7	SELECT MAX(SchoolRecords.present) FROM SchoolRecords;
8	SELECT MAX(SchoolRecords.present) FROM SchoolRecords;
9	SELECT MAX(SchoolRecords.present) FROM SchoolRecords;
10	SELECT SUM(SchoolRecords.absented) FROM SchoolRecords;
11	SELECT SUM(SchoolRecords.released) FROM SchoolRecords;
12	SELECT SUM(SchoolRecords.present) FROM SchoolRecords;
13	SELECT COUNT(DISTINCT SchoolRecords.enrolled) FROM SchoolRecords;

Fonte: Elaborado pela autora(2023).

APÊNDICE C – QUADRO DE CONSULTAS FORMULADAS EM LINGUAGEM FORMAL PARA BANCO DE DADOS MONGODB

O Quadro 4 apresenta 13 (treze) consultas em linguagem MongoDB. Tais consultas são o resultado da tradução manual das consultas em LN que foram utilizadas no experimento deste trabalho. A tradução manual das consultas foi realizada com o intuito de verificar se as ferramentas retornavam consultas validas e se essas consultas, quando submetidas aos bancos, retornavam os resultados esperados.

Quadro 4 – Consultas em Linguagem MongoDB

ID	Consultas em Linguagem MongoDB
1	*db.SchoolRecords.aggregate*([{"$*group*" : {"*_id*" : {},"*avg_SchoolRecords_present*" : {"$*avg*" : "$*SchoolRecords.present*"}}},{"$*project*" : {"*avg_SchoolRecords_present*" : 1,"*_id*" : 0}}])
2	*db.SchoolRecords.aggregate*([{"$*group*" : {"*_id*" : {},"*avg_SchoolRecords_absented*" : {"$*avg*" : "$*SchoolRecords.absented*"}}},{"$*project*" : {"*avg_SchoolRecords_absented*" : 1,"*_id*" : 0}}])
3	*db.SchoolRecords.aggregate*([{"$*group*" : {"*_id*" : {},"*avg_SchoolRecords_released*" : {"$*avg*" : "$*SchoolRecords.released*"}}},{"$*project*" : {"*avg_SchoolRecords_released*" : 1,"*_id*" : 0}}])
4	*db.SchoolRecords.aggregate*([{"$*group*" : {"*_id*" : {},"*min_SchoolRecords_present*" : {"$*min*" : "$*SchoolRecords.present*"}}},{"$*project*" : {"*min_SchoolRecords_present*" : 1,"*_id*" : 0}}])
5	*db.SchoolRecords.aggregate*([{"$*group*" : {"*_id*" : {},"*min_SchoolRecords_absented*" : {"$*min*" : "$*SchoolRecords.absented*"}}},{"$*project*" : {"*min_SchoolRecords_absented*" : 1,"*_id*" : 0}}])
6	*db.SchoolRecords.aggregate*([{"$*group*" : {"*_id*" : {},"*min_SchoolRecords_released*" : {"$*min*" : "$*SchoolRecords.released*"}}},{"$*project*" : {"*min_SchoolRecords_released*" : 1,"*_id*" : 0}}])
7	*db.SchoolRecords.aggregate*([{"$*group*" : {"*_id*" : {},"*max_SchoolRecords_present*" : {"$*max*" : "$*SchoolRecords.present*"}}},{"$*project*" : {"*max_SchoolRecords_present*" : 1,"*_id*" : 0}}])
8	*db.SchoolRecords.aggregate*([{"$*group*" : {"*_id*" : {},"*max_SchoolRecords_absented*" : {"$*max*" : "$*SchoolRecords.absented*"}}},{"$*project*" : {"*max_SchoolRecords_absented*" : 1,"*_id*" : 0}}])
9	*db.SchoolRecords.aggregate*([{"$*group*" : {"*_id*" : {},"*max_SchoolRecords_released*" : {"$*max*" : "$*SchoolRecords.released*"}}},{"$*project*" : {"*max_SchoolRecords_released*" : 1,"*_id*" : 0}}])
10	*db.SchoolRecords.aggregate*([{"$*group*" : {"*_id*" : {},"*sum_SchoolRecords_absented*" : {"$*sum*" : "$*SchoolRecords.absented*"}}},{"$*project*" : {"*sum_SchoolRecords_absented*" : 1,"*_id*" : 0}}])
11	*db.SchoolRecords.aggregate*([{"$*group*" : {"*_id*" : {},"*sum_SchoolRecords_released*" : {"$*sum*" : "$*SchoolRecords.released*"}}},{"$*project*" : {"*sum_SchoolRecords_released*" : 1,"*_id*" : 0}}])
12	*db.SchoolRecords.aggregate*([{"$*group*" : {"*_id*" : {},"*sum_SchoolRecords_present*" : {"$*sum*" : "$*SchoolRecords.present*"}}},{"$*project*" : {"*sum_SchoolRecords_present*" : 1,"*_id*" : 0}}])
13	*db.SchoolRecords.aggregate*([{"$*group*" : {"*_id*" : {},"*count*" : {"$*sum*" : 1}}},{"$*project*" : {"*count*" : 1,"*_id*" : 0}}])

Fonte: Elaborado pela autora(2023).

APÊNDICE D – TABELA DE RESULTADOS DETALHADOS

A Tabela 4 mostra o tempo de execução de cada repetição executada nos cenários com *dataset* de 10GB, observando que para cada cenário foram realizadas 5 (cinco) repetições. Além disso, a Tabela 4 também apresenta o menor tempo de execução entre as repetições(tempo mínimo) e o maior tempo de execução entre as repetições (tempo máximo), e o desvio padrão entre os valores.

Tabela 4 – Tempos de execução dos cenários com *dataset* de 10GB

	Dataset 10GB							
Cenário	**R1**	**R2**	**R3**	**R4**	**R5**	**Tempo mínimo**	**Tempo máxima**	**Desvio Padrão**
C1	2324,97	2299,93	2293,62	2283,70	2285,12	2283,70	2324,97	14,96
C2	12155,5	12114,46	12126,68	12219,21	12403,28	12114,46	12403,28	106,11
C3	24490,28	24480,95	24475,81	24415,33	24611,12	24415,33	24611,12	63,90
C4	2286,04	2364,11	2363,79	2275,47	2313,53	2275,47	2364,11	37,52
C5	12257,40	12168,42	12126,10	12242,88	12195,65	12126,10	12257,4	48,15
C6	24273,61	24097,07	24195,68	24177,49	24431,10	24097,07	24431,10	112,98

Fonte: Elaborado pela autora (2023).

A Tabela 5 mostra o tempo de execução de cada repetição executada nos cenários com *dataset* de 50GB, observando que para cada cenário foram realizadas 5 (cinco) repetições. Além disso, a Tabela 5 também apresenta o menor tempo de execução entre as repetições(tempo mínimo) e o maior tempo de execução entre as repetições (tempo máximo), e o desvio padrão entre os valores.

Tabela 5 – Tempos de execução dos cenários com *dataset* de 50GB

	Dataset 50GB							
Cenário	**R1**	**R2**	**R3**	**R4**	**R5**	**Tempo mínimo**	**Tempo máxima**	**Desvio Padrão**
C7	7268.07	7340,21	7295,66	7280,02	7389,97	7280,02	7389,97	42,80
C8	34180,50	34802,64	34619,64	34890,73	34702,69	34180,50	34890,73	246,90
C9	70377,86	69920,79	67811,83	68027,09	68075,26	67811,83	70377,86	1080,36
C10	7300,59	7281,57	7256,53	7255,44	7348,42	7255,44	7348,42	34,35
C11	34104,66	34210,06	34022,71	34252,16	34448,54	34022,71	34448,54	144,79
C12	68393,06	68769,88	68899,25	68558,67	68658,13	68393,06	68899,25	173,64

Fonte: Elaborado pela autora (2023).

A Tabela 6 mostra o tempo de execução de cada repetição executada nos cenários com *dataset* de 100GB, observando que para cada cenário foram realizadas 5 (cinco) repetições. Além disso, a Tabela 6 também apresenta o menor tempo de execução entre as repetições(tempo mínimo) e o maior tempo de execução entre as repetições (tempo máximo), e o desvio padrão entre os valores.

Tabela 6 – Tempos de execução dos cenários com *dataset* de 100GB

	Dataset 100GB							
Cenário	**R1**	**R2**	**R3**	**R4**	**R5**	**Tempo mínimo**	**Tempo máxima**	**Desvio Padrão**
C13	17990,97	18352,17	18930,00	18834,21	18576,48	17990,97	18930,00	339,60
C14	82729,49	84537,63	86405,44	86553,82	90873,06	82729,49	90873,06	2713,33
C15	137114,58	139129,4	139033,75	138998,15	138624,67	137114,58	139129,4	752,60
C16	18242,68	18124,78	17958,08	18207,21	18307,43	17958,08	18307,43	120,41
C17	83243,00	98669,79	77977,96	92687,85	76794,70	76794,70	98669,79	8507,22
C18	137315,01	138500,82	138786,08	139571,87	136526,18	136526,18	139571,87	1084,70

Fonte: Elaborado pela autora (2023).