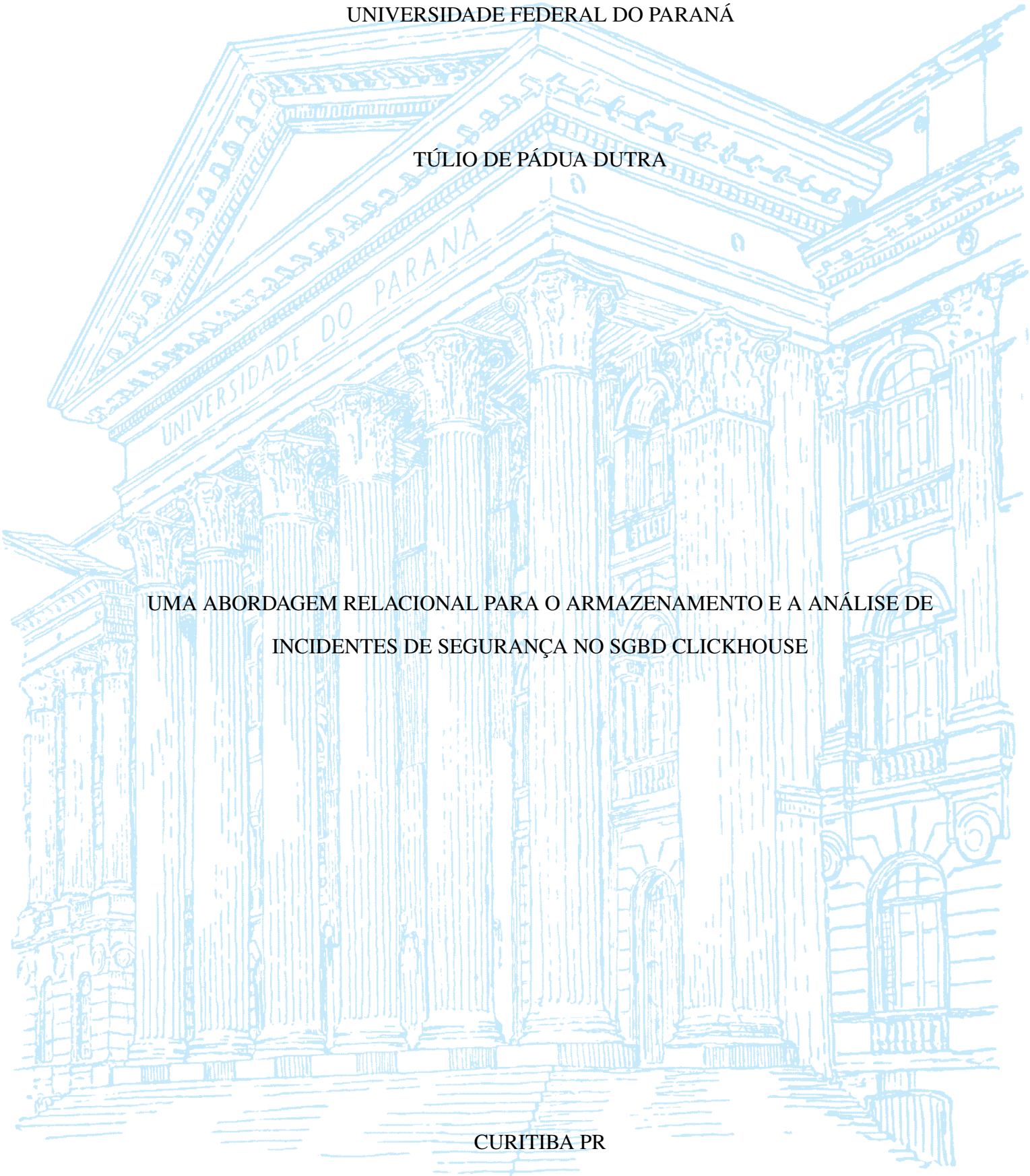
UNIVERSIDADE FEDERAL DO PARANÁ

TÚLIO DE PÁDUA DUTRA



UMA ABORDAGEM RELACIONAL PARA O ARMAZENAMENTO E A ANÁLISE DE INCIDENTES DE SEGURANÇA NO SGBD CLICKHOUSE

CURITIBA PR

2025

TÚLIO DE PÁDUA DUTRA

UMA ABORDAGEM RELACIONAL PARA O ARMAZENAMENTO E A ANÁLISE DE INCIDENTES DE SEGURANÇA NO SGBD CLICKHOUSE

Trabalho apresentado como requisito parcial à conclusão do Curso de Bacharelado em Ciência da Computação, Setor de Ciências Exatas, da Universidade Federal do Paraná.

Área de concentração: *Computação*.

Orientador: Simone Dominico.

CURITIBA PR

2025

Universidade Federal do Paraná
Setor de Ciências Exatas
Curso de Ciência da Computação

Ata de Apresentação de Trabalho de Conclusão de Curso 2

Título do Trabalho: Uma Abordagem Relacional para o armazenamento e Análise de Incidentes de Segurança no SGBD ClickHouse

Autor(es):

GRR20206155 Nome: **Túlio de Pádua Dutra**

Apresentação: Data: 09/12/2025 Hora: 10h00 Local: Sala de videoconferência

Orientador: Simone Dominico _____ Documento assinado digitalmente gov.br SIMONE DOMINICO Data: 09/12/2025 15:55:22-0300 Verifique em https://validar.iti.gov.br _____

Membro 1: André Ricardo Abed Grégio_____ Documento assinado digitalmente gov.br ANDRE RICARDO ABED GREGIO Data: 09/12/2025 11:34:30-0300 Verifique em https://validar.iti.gov.br _____

Membro 2: Eduardo Vudala Senoski_____ Documento assinado digitalmente gov.br EDUARDO VUDALA SENOSKI Data: 09/12/2025 15:27:24-0300 Verifique em https://validar.iti.gov.br _____

(nome) (assinatura)

AVALIAÇÃO – Produto escrito		ORIENTADOR	MEMBRO 1	MEMBRO 2	MÉDIA
Conteúdo	(00-40)				
Referência Bibliográfica	(00-10)				
Formato	(00-05)				
AVALIAÇÃO – Apresentação Oral					
Domínio do Assunto	(00-15)				
Desenvolvimento do Assunto	(00-05)				
Técnica de Apresentação	(00-03)				
Uso do Tempo	(00-02)				
AVALIAÇÃO – Desenvolvimento					
Nota do Orientador	(00-20)		**********	**********	
NOTA FINAL		**********	**********	**********	**80**

Os pesos indicados são sugestões.

Conforme decisão do colegiado do curso de Ciência da Computação, a entrega dos documentos comprobatório de trabalho de Conclusão de Curso 2 deve deve respeitar os seguintes procedimentos: o orientador deve abrir um processo no Sistema Eletrônico de Informações (SEI – UFPR); Selecionar o tipo: *Graduação: Trabalho Conclusão de Curso*; informar os interessados: nome do aluno e o nome do orientador; anexar esta ata escaneada e a versão final do PDF da monografia do aluno; Tramitar o processo para CCOMP (Coordenação de Ciência da Computação).

RESUMO

O avanço constante da tecnologia da informação e o aumento no volume de incidentes cibernéticos têm gerado uma grande quantidade de dados de segurança em formatos semiestruturados, como o JSON. Embora amplamente utilizado, esse formato apresenta desafios para armazenamento e análise em larga escala, especialmente em bancos relacionais tradicionais. Este trabalho propõe uma abordagem relacional para o armazenamento e a análise de relatórios de incidentes de segurança no Sistema Gerenciador de Banco de Dados (SGBD) ClickHouse, que é orientado a colunas e voltado para consultas analíticas de alto desempenho. A pesquisa investiga como documentos JSON podem ser transformados em estruturas relacionais simplificadas, normalizadas até a Segunda Forma Normal (2FN), de modo a preservar a semântica dos dados e permitir consultas eficientes. São discutidos conceitos de segurança da informação, integração de dados semiestruturados e fundamentos de bancos analíticos, além da análise de trabalhos relacionados. Os resultados obtidos demonstram que a modelagem proposta é viável e oferece vantagens para análises em larga escala, contribuindo para o aprimoramento da integração entre dados de cibersegurança e sistemas analíticos de alto desempenho.

Palavras-chave: Segurança da Informação. JSON. ClickHouse. Banco de Dados Relacional. Análise de Dados.

ABSTRACT

The continuous advancement of information technology and the growing number of cyber incidents have produced massive amounts of security data in semi-structured formats such as JSON. Although widely adopted, this format poses challenges for large-scale storage and analysis, especially within traditional relational databases. This study proposes a relational approach for storing and analyzing security incident reports using the ClickHouse Database Management System (DBMS), a column-oriented engine designed for high-performance analytical queries. The research investigates how JSON documents can be transformed into simplified relational structures normalized up to the Second Normal Form (2NF), preserving data semantics while enabling efficient queries. The study discusses key concepts in information security, semi-structured data integration, and analytical databases, as well as related works. The results demonstrate that the proposed modeling is feasible and provides performance advantages for large-scale analyses, contributing to the improvement of cybersecurity data integration into high-performance analytical systems.

Keywords: Information Security. JSON. ClickHouse. Relational Database. Data Analysis.

LISTA DE FIGURAS

4.1 Esquema Relacional em 2FN 20

5.1 Query 1: Quais organizações ou alvos (target) sofrem mais incidentes? 25
5.2 Query 2: Quais analistas estão trabalhando no maior número de incidentes? . . . 26
5.3 Query 3: Quais são os 10 hashes de arquivo (MD5) mais comuns encontrados em todos os incidentes? 27
5.4 Query 4: Quais são as URLs mais acessadas nas requisições de rede dos incidentes? 28
5.5 Query 5: Qual a evolução do número de incidentes reportados ao longo do tempo (por mês)? 29
5.6 Query 6: Total de hashes md5 únicos no PostgreSQL 31
5.7 Query 7: Total de hashes md5 únicos no ClickHouse 31
5.8 Query 8: Top 10 Organizações Mais Atacadas no ClickHouse. 31
5.9 Query 9: Top 10 Organizações Mais Atacadas no PostgreSQL 32
5.10 Query 10: Hash Mais Perigoso (Máxima Dispersão) no PostgreSQL 33
5.11 Query 11: Hash Mais Perigoso (Máxima Dispersão) no ClickHouse 33
5.12 Gráfico dos tempos de execução no PostgreSQL e ClickHouse 35

LISTA DE TABELAS

3.1 Mapeamento entre a nomenclatura do Dataset JSON e a modelagem com UCO . 16

5.1 Resultados detalhados: Queries 6 e 7. 34
5.2 Resultados detalhados: Queries 8 e 9. 34
5.3 Resultados detalhados: Queries 10 e 11 34
5.4 Tempo médio de execução das consultas no PostgreSQL e ClickHouse 34

LISTA DE ACRÔNIMOS

UFPR	Universidade Federal do Paraná
SGBD	Sistema Gerenciador de Banco de Dados
1FN	Primeira Forma Normal
2FN	Segunda Forma Normal
APT	Advanced Persistent Threat
BOM	Byte Order Mark
CSV	Comma-Separated Values
EDR	Endpoint Detection and Response
ETL	Extract, Transform and Load
FTP	File Transfer Protocol
HTTP	Hypertext Transfer Protocol
IoC	Indicator of Compromise
IP	Internet Protocol
JSON	JavaScript Object Notation
MD5	Message Digest Algorithm 5
NoSQL	Not Only Structured Query Language
OLAP	Online Analytical Processing
OLTP	Online Transaction Processing
RDF	Resource Description Framework
SQL	Structured Query Language
UCO	Unified Cybersecurity Ontology
URL	Uniform Resource Locator
UUID	Universally Unique Identifier

SUMÁRIO

1 INTRODUÇÃO 9
1.1 CONTEXTUALIZAÇÃO 9
1.2 OBJETIVOS GERAIS 10
1.3 OBJETIVOS ESPECÍFICOS 10
1.4 CONTRIBUIÇÕES 10
1.5 ORGANIZAÇÃO DO TRABALHO 11
2 FUNDAMENTAÇÃO TEÓRICA 12
2.1 SEGURANÇA DA INFORMAÇÃO E INDICADORES DE COMPROMETIMENTO 12
2.2 DADOS JSON E INFORMAÇÕES SEMIESTRUTURADAS 12
2.3 BANCOS DE DADOS ANALÍTICOS E COLUNARES 13
2.4 MODELO RELACIONAL E NORMALIZAÇÃO APLICADA AO CONTEXTO 13
2.5 CONCLUSÃO 14
3 TRABALHOS RELACIONADOS 15
3.1 ONTOLOGIAS EM SEGURANÇA DA INFORMAÇÃO 15
3.2 INTERPRETAÇÃO SEMÂNTICA DE LOGS 16
3.3 INTEGRAÇÃO DE JSON EM BANCOS RELACIONAIS 16
3.4 CONCLUSÃO 17
4 METODOLOGIA 18
4.1 FORMATO DO DATASET JSON DE ENTRADA 18
4.2 MODELAGEM RELACIONAL (2FN). 20
4.3 SCRIPT DE INSERÇÃO DOS DADOS NO CLICKHOUSE 21
4.4 CONCLUSÃO 24
5 VALIDAÇÃO. 25
5.1 CONSULTAS EXPLORATÓRIAS DE VALIDAÇÃO 25
5.2 AVALIAÇÃO DE DESEMPENHO: POSTGRESQL VS CLICKHOUSE 30
6 CONCLUSÃO 36
6.1 TRABALHOS FUTUROS 37
6.2 CONSIDERAÇÕES FINAIS 37
REFERÊNCIAS 38

1 INTRODUÇÃO

1.1 CONTEXTUALIZAÇÃO

O grande desafio enfrentado atualmente na área de segurança da informação é lidar com a enorme quantidade de dados gerados diariamente por sistemas, aplicações e redes. Cada requisição HTTP, cada arquivo analisado por um antivírus, cada alerta de firewall ou de sistema de detecção de intrusão gera registros que precisam ser armazenados e analisados (Nimbalkar et al., 2016). Esses registros, muitas vezes disponibilizados em formato *JavaScript Object Notation* (JSON), que são semiestruturados e heterogêneos, o que dificulta sua manipulação em bancos de dados tradicionais (Petković, 2017). Além disso, incidentes de segurança se tornam cada vez mais complexos, envolvendo múltiplos vetores de ataque e longos períodos de execução, o que exige uma visão integrada e analítica dos dados (Yao et al., 2014). Esse cenário reforça a necessidade de soluções que unam escalabilidade, desempenho e consistência para apoiar a tomada de decisão em ambientes de cibersegurança.

A literatura recente destaca que, embora existam avanços em ontologias e representações semânticas para eventos de segurança (Yao et al., 2014; Gyrard et al., 2021), ainda há uma lacuna prática na integração entre o armazenamento eficiente de logs e a capacidade analítica de sistemas relacionais. Trabalhos como os de (Chaudhuri et al., 2016) e (Liu et al., 2018) apontam para o crescimento exponencial do volume de eventos de rede, enquanto estudos como (Nimbalkar et al., 2016) evidenciam a falta de padronização nos formatos de logs e a dificuldade de interpretação automática. Assim, o presente trabalho busca contribuir com uma abordagem que combina modelagem relacional normalizada e tratamento semântico de dados semiestruturados, de forma reprodutível e voltada à análise de incidentes reais.

A motivação para este estudo surge da constatação de que métodos convencionais de armazenamento e consulta, baseados em bancos relacionais, não conseguem atender plenamente às demandas de flexibilidade e escalabilidade impostas pelos dados de segurança (Bharthan e Bharathan, 2014). Por outro lado, soluções *NoSQL* oferecem maior maleabilidade para lidar com dados em JSON, mas carecem de suporte transacional e de mecanismos avançados de consulta (Bahta e Atay, 2019). Nesse contexto, o uso de sistemas analíticos orientados a colunas, como o ClickHouse, apresenta-se como uma alternativa promissora para consultas agregadas e análises em larga escala. O ClickHouse foi escolhido como base da proposta justamente por sua vocação analítica e eficiência no processamento de grandes volumes de dados. O PostgreSQL, por sua vez, é utilizado exclusivamente neste trabalho como ferramenta de comparação, com o objetivo de demonstrar, de forma empírica, que o desempenho de consultas analíticas em um SGBD relacional tradicional é inferior ao obtido no ClickHouse.

Os dados utilizados neste trabalho são reais, provenientes de registros de incidentes de segurança da informação coletados entre os anos de 2010 e 2019. O conjunto abrange múltiplas fontes de log, incluindo requisições HTTP, artefatos de arquivos e eventos de rede, e totaliza dezenas de milhões de linhas em formato JSON. Esse volume permite reproduzir um cenário autêntico de análise em larga escala e testar, na prática, as estratégias de normalização e desempenho propostas.

1.2 OBJETIVOS GERAIS

O objetivo geral deste trabalho é desenvolver uma abordagem para o armazenamento e a análise de dados de incidentes de segurança da informação representados em formato JSON, integrando técnicas de modelagem relacional e conceitos de representação semântica. A proposta busca criar uma solução que permita converter, armazenar e consultar de forma eficiente dados semiestruturados em um ambiente relacional, de modo a viabilizar análises estruturadas e consistentes voltadas à segurança cibernética.

1.3 OBJETIVOS ESPECÍFICOS

De forma mais específica, este estudo pretende investigar métodos e arquiteturas existentes para o processamento e a análise de grandes volumes de dados de segurança, com foco em logs e eventos estruturados em formato JSON. Além disso, busca-se compreender e aplicar técnicas de integração entre dados semiestruturados e sistemas de gerenciamento de bancos de dados relacionais, tomando como referência trabalhos que propõem estratégias de mapeamento entre JSON e estruturas relacionais (Bahta e Atay, 2019; Bharthan e Bharathan, 2014; Petković, 2017).

Com base nesse fundamento teórico, o trabalho propõe o projeto de um modelo de dados relacional capaz de representar de forma eficiente e coerente as informações contidas nos registros de incidentes, preservando a integridade e a normalização dos dados. Também se pretende implementar um processo de extração, transformação e carga (ETL) que realize a conversão e o carregamento automático de dados provenientes de arquivos JSON para o modelo relacional, de forma reprodutível. A avaliação da abordagem inclui a comparação entre o ClickHouse, como sistema analítico proposto, e o PostgreSQL, empregado apenas para fins de comparação de desempenho. Assim, é possível demonstrar de forma quantitativa as vantagens do modelo analítico em consultas agregadas e em cenários de grande volume de dados.

Por fim, o trabalho visa avaliar a eficiência da solução proposta quanto à capacidade de representar semanticamente os dados de segurança e disponibilizar um conjunto de scripts automatizados para a criação e a carga do banco de dados, em sintonia com a filosofia de reprodutibilidade de benchmarks como o TPC-H.

1.4 CONTRIBUIÇÕES

A contribuição esperada deste trabalho é dupla. Em primeiro lugar, oferecer uma modelagem prática para dados de cibersegurança em JSON, demonstrando como eles podem ser convertidos para um formato relacional em Segunda Forma Normal (2FN), adequada ao ClickHouse. A 2FN é um estágio de normalização que requer que todos os atributos não-chave dependam totalmente da chave primária, evitando dependências parciais e redundâncias. A escolha pela 2FN, em detrimento de modelos analíticos baseados em *star schema*, justifica-se pela necessidade de manter a integridade e a flexibilidade do modelo frente à natureza dinâmica dos dados de segurança. Enquanto o *star schema* é eficiente para análises predefinidas com dimensões fixas, a 2FN permite lidar com estruturas de dados variáveis sem perda de consistência.

Em segundo lugar, o trabalho contribui ao disponibilizar publicamente scripts reprodutíveis para a criação e a carga do banco de dados, permitindo que as tabelas sejam geradas e populadas automaticamente a partir dos arquivos JSON. Além disso, apresenta um conjunto de consultas analíticas de referência voltadas ao domínio de segurança da informação, úteis para avaliação e comparação de desempenho em outros estudos. Finalmente, realiza-se uma avaliação

empírica comparando o ClickHouse e o PostgreSQL quanto à eficiência no armazenamento e nas consultas, demonstrando o ganho de desempenho obtido pela adoção de um SGBD analítico em detrimento de um relacional tradicional.

1.5 ORGANIZAÇÃO DO TRABALHO

A organização do documento é a seguinte. O Capítulo 2 apresenta a fundamentação teórica, abordando conceitos de segurança da informação, bancos de dados analíticos e integração de JSON em sistemas relacionais. O Capítulo 3 descreve os trabalhos relacionados, revisando propostas anteriores de uso de ontologias de cibersegurança, frameworks de interpretação de logs e métodos de tradução de JSON. O Capítulo 4 expõe a metodologia adotada para a modelagem e transformação dos dados, detalhando as escolhas técnicas feitas. O Capítulo 5 apresenta os resultados obtidos com a aplicação do modelo no ClickHouse e a comparação de desempenho com o PostgreSQL, discutindo consultas e análises possíveis. Por fim, o Capítulo 6 traz as conclusões do trabalho, as limitações encontradas e possíveis direções para pesquisas futuras.

2 FUNDAMENTAÇÃO TEÓRICA

2.1 SEGURANÇA DA INFORMAÇÃO E INDICADORES DE COMPROMETIMENTO

A segurança da informação evoluiu significativamente nas últimas décadas. Em um primeiro momento, o foco estava na confidencialidade dos dados, ou seja, em impedir acessos não autorizados. Com o tempo, os princípios fundamentais passaram a incluir também a integridade, a disponibilidade, a autenticidade e o não repúdio (Yao et al., 2014). Atualmente, o campo da segurança da informação envolve desde técnicas preventivas até mecanismos de detecção e resposta a incidentes, além de métodos de auditoria e análise forense.

Nesse contexto, os registros e logs de sistemas se tornaram uma das principais fontes de evidência para a detecção de anomalias e ameaças. Cada requisição HTTP, cada tentativa de acesso a um arquivo ou cada alerta emitido por um antivírus ou firewall representa um fragmento de informação potencialmente útil para identificar padrões de comportamento malicioso (Nimbalkar et al., 2016). Esses registros, no entanto, são altamente heterogêneos: variam conforme o sistema de origem, o formato de armazenamento e o nível de detalhamento. Muitos deles contêm dados aninhados, campos opcionais e estruturas que se alteram ao longo do tempo, o que dificulta sua análise automatizada.

Do ponto de vista técnico, esses arquivos de incidentes representam um desafio por reunirem as chamadas "três V's" dos dados modernos (volume, variedade e velocidade), além de uma quarta dimensão crítica: a necessidade de correlação. Em ambientes corporativos, uma investigação pode envolver milhões de registros distribuídos entre diferentes fontes (como servidores, firewalls, proxies e antivírus), exigindo a correlação entre eventos distintos para identificar uma cadeia de ataque. No caso específico deste trabalho, os dados originais não foram recebidos diretamente em formato JSON, mas em múltiplos arquivos separados e compactados (ZIP), contendo diferentes tipos de relatórios que precisaram ser extraídos, padronizados e convertidos para o formato semiestruturado.

Portanto, compreender a natureza desses dados e estruturar uma forma eficiente de armazená-los é essencial para que possam ser utilizados em análises forenses, monitoramento contínuo e sistemas de detecção de intrusão.

2.2 DADOS JSON E INFORMAÇÕES SEMIESTRUTURADAS

O formato *JavaScript Object Notation* (JSON) tornou-se um padrão amplamente adotado para o intercâmbio de informações na web e em sistemas distribuídos. Sua estrutura flexível permite representar dados hierárquicos e aninhados por meio de pares chave–valor, o que facilita a leitura humana e o processamento por máquinas. Essa versatilidade, contudo, é acompanhada de desafios quando se trata de armazenar e consultar grandes volumes de dados JSON em sistemas relacionais.

Segundo (Petković, 2017), bancos de dados relacionais clássicos, como o PostgreSQL e o MySQL, foram projetados para lidar com dados rigidamente estruturados, com esquemas pré-definidos e tipos fixos. Já o JSON, por natureza, é semiestruturado: diferentes documentos podem apresentar conjuntos de atributos distintos, e os mesmos campos podem conter valores de tipos variados. Isso dificulta o uso direto de técnicas tradicionais de normalização e indexação.

Diversas abordagens foram propostas para superar essa incompatibilidade. O método RelationalJSON (Bharthan e Bharathan, 2014) introduz a ideia de mapear a árvore de um

documento JSON para tabelas relacionais, reduzindo o número de tabelas necessárias e mantendo a integridade dos dados. Abordagens mais recentes, como a de (Bahta e Atay, 2019), propõem estratégias *schema-oblivious*, em que o mapeamento é dinâmico e não depende de um esquema fixo, permitindo adaptar-se automaticamente a diferentes estruturas de JSON. Há ainda propostas híbridas, como o ESQLite (Hermkens, 2016), que aplicam técnicas de *shredding* (fragmentação de JSON em pares chave–valor) para armazenar os dados de forma relacional e permitir consultas no estilo XPath.

Essas estratégias ilustram o esforço da comunidade em conciliar a flexibilidade dos dados semiestruturados com a consistência dos modelos relacionais, buscando um equilíbrio entre desempenho e preservação da semântica. No contexto deste trabalho, essa integração é fundamental, pois os relatórios de segurança convertidos em JSON precisam ser transformados em tabelas relacionais que mantenham o significado dos atributos originais, sem perder a capacidade de análise agregada.

2.3 BANCOS DE DADOS ANALÍTICOS E COLUNARES

Os sistemas de gerenciamento de bancos de dados podem ser classificados, de forma geral, em dois grandes grupos: os voltados para processamento de transações (OLTP — *Online Transaction Processing*) e os voltados para processamento analítico (OLAP — *Online Analytical Processing*). Os bancos OLTP priorizam a escrita rápida e a integridade transacional, sendo ideais para sistemas operacionais de uso contínuo, como aplicações bancárias ou de controle de estoque. Já os bancos OLAP são otimizados para leitura intensiva e análise de grandes volumes de dados, com foco em consultas agregadas e exploratórias.

Enquanto os bancos tradicionais armazenam dados linha a linha (row-oriented), os bancos analíticos orientados a colunas (column-oriented) organizam os valores por coluna. Essa estrutura permite que apenas os atributos relevantes para uma consulta sejam lidos, reduzindo significativamente o volume de dados acessados e acelerando operações de agregação, contagem e filtragem. Além disso, a compactação por coluna é muito mais eficiente, já que os valores de um mesmo tipo tendem a se repetir com frequência (Hermkens, 2016).

Nesse cenário, o ClickHouse destaca-se como um dos sistemas mais eficientes para análise de grandes volumes de dados em tempo quase real. Projetado para processar bilhões de linhas com latência mínima, ele combina a arquitetura colunar com paralelismo massivo e otimizações específicas para agregações. Essa abordagem o torna especialmente adequado para aplicações de segurança da informação, em que é necessário identificar padrões, correlações e tendências em logs volumosos. Exemplos práticos incluem a detecção de endereços IP recorrentes em múltiplos ataques ou a análise de comportamento de requisições HTTP em diferentes períodos (Chaudhuri et al., 2016).

Assim, enquanto bancos como o PostgreSQL servem de referência para cenários transacionais, o ClickHouse representa a categoria de bancos analíticos, justificando sua escolha como base para o modelo proposto neste trabalho.

2.4 MODELO RELACIONAL E NORMALIZAÇÃO APLICADA AO CONTEXTO

O modelo relacional, proposto originalmente por Codd na década de 1970, introduziu uma forma sistemática de representar dados por meio de tabelas, atributos e relacionamentos. Essa estrutura tornou-se o alicerce dos SGBDs modernos, possibilitando consultas complexas e garantindo integridade referencial por meio da linguagem SQL. Um dos conceitos fundamentais

desse modelo é a normalização, que busca eliminar redundâncias e dependências parciais entre atributos.

A Primeira Forma Normal (1FN) estabelece que todos os campos de uma tabela devem conter apenas valores atômicos, ou seja, indivisíveis. A Segunda Forma Normal (2FN), por sua vez, exige que todos os atributos não-chave dependam inteiramente da chave primária, eliminando dependências parciais em chaves compostas. Essa etapa é particularmente importante quando se busca converter estruturas hierárquicas, como documentos JSON, em tabelas relacionais. A aplicação da 2FN reduz a duplicação de dados e evita anomalias de inserção ou exclusão.

No contexto deste trabalho, a adoção da 2FN foi uma escolha deliberada. Modelos mais analíticos, como o *star schema*, são amplamente utilizados em *data warehouses*, mas partem de um conjunto fixo de dimensões e fatos, o que não se adapta bem à natureza variável e dinâmica dos incidentes de segurança. A 2FN, por outro lado, permite representar esses dados de forma consistente e normalizada, preservando a semântica das relações entre entidades, como incidentes, artefatos e requisições, e garantindo a integridade necessária para análises precisas. Essa estrutura também favorece o desempenho no ClickHouse, cuja arquitetura colunar compensa a sobrecarga normalmente associada à normalização.

Dessa forma, o modelo relacional aplicado neste trabalho atua como uma ponte entre a flexibilidade do JSON e a eficiência dos bancos analíticos, permitindo armazenar dados de segurança de forma organizada, normalizada e escalável.

2.5 CONCLUSÃO

Neste capítulo foram discutidos os principais fundamentos teóricos que embasam a proposta deste trabalho. Inicialmente, abordou-se a evolução da segurança da informação e o papel dos indicadores de comprometimento como fontes de dados essenciais para detecção e resposta a incidentes. Em seguida, analisaram-se as particularidades dos dados em formato JSON e os desafios associados à sua integração em modelos relacionais. Discutiu-se também o funcionamento e as vantagens dos bancos de dados analíticos orientados a colunas, destacando o ClickHouse como plataforma principal para análise em larga escala. Por fim, explorou-se o modelo relacional e o processo de normalização, justificando a adoção da 2FN como estratégia adequada para o contexto dos incidentes de segurança.

Com isso, a fundamentação teórica estabelece o embasamento necessário para o desenvolvimento metodológico apresentado no próximo capítulo, no qual serão descritas as etapas de modelagem, transformação e avaliação do desempenho do modelo proposto.

3 TRABALHOS RELACIONADOS

A discussão realizada na fundamentação teórica mostrou que a integração de dados de cibersegurança em formatos semiestruturados, como JSON, é um problema relevante e ainda em aberto. Neste capítulo, são apresentados os principais trabalhos relacionados, agrupados em três linhas de pesquisa que dialogam diretamente com este estudo: (I) ontologias aplicadas à segurança da informação, (II) interpretação semântica de logs e (III) integração de JSON em bancos de dados relacionais. Além de apresentar cada linha de pesquisa, busca-se aqui discutir em que medida essas abordagens se aproximam da proposta deste trabalho e quais lacunas permanecem.

3.1 ONTOLOGIAS EM SEGURANÇA DA INFORMAÇÃO

O uso de ontologias tem se mostrado uma alternativa eficaz para organizar e padronizar conceitos relacionados à cibersegurança. Yao et al. (Yao et al., 2014) propuseram um método de construção de bases de conhecimento para segurança da informação, fundamentado em ontologias que descrevem ativos, vulnerabilidades, ameaças e relações entre esses elementos. Essa abordagem permite que diferentes fontes de dados sejam integradas sob uma mesma estrutura conceitual, favorecendo análises mais consistentes e reduzindo ambiguidades.

Iniciativas como a Unified Cybersecurity Ontology (UCO) reforçam essa tendência, oferecendo um modelo unificado para conectar informações de diferentes ferramentas de monitoramento e prevenção de ataques (Nimbalkar et al., 2016). A UCO estabelece um vocabulário comum para representar incidentes, atores maliciosos, técnicas de ataque e indicadores de comprometimento, facilitando a interoperabilidade entre sistemas. Entretanto, sua implementação prática ainda é limitada, pois exige que os dados originais estejam previamente estruturados em triplas semânticas, algo que nem sempre é viável em ambientes reais de monitoramento.

Neste trabalho, a UCO está sendo utilizada principalmente para nomear as entidades e seus atributos, facilitando dessa forma a execução de queries, devido à padronização proporcionada por esta ontologia.

A tabela abaixo (Tabela 3.1) mostra como foram feitas as transformações dos nomes relativos ao dataset em JSON para a modelagem final utilizando a UCO:

Tabela 3.1: Mapeamento entre a nomenclatura do Dataset JSON e a modelagem com UCO

Nomenclatura do Dataset JSON	**Nomenclatura da modelagem com UCO**
data do relatório	CYBER_INCIDENTS.report_datetime
instituição	CYBER_INCIDENTS.organization
analista	CYBER_INCIDENTS.analyst
incidente	CYBER_INCIDENTS.incident_id
url	CYBER_INCIDENTS.initial_url
hashes (md5)	INCIDENT_HASHES.md5_hash
arquivos criados/acessados	FILE_ARTIFACTS
alvos	CYBER_INCIDENTS.target
get	NETWORK_REQUESTS.method
post	NETWORK_REQUESTS.method
email	CYBER_INCIDENTS.observed_email
banco de dados sql	SQL_DATABASES
ftp	FTP_SERVERS
observações	OBSERVATIONS

3.2 INTERPRETAÇÃO SEMÂNTICA DE LOGS

Outra linha de pesquisa relevante é a transformação de registros de sistemas em representações semânticas. Nimbalkar et al. (Nimbalkar et al., 2016) propuseram um framework para interpretar arquivos de log estruturados, mapeando campos como IPs, URLs e timestamps para triplas RDF compatíveis com ontologias como a UCO. Essa abordagem possibilita que os logs, tradicionalmente utilizados apenas para auditoria, sejam aproveitados em processos de raciocínio automático e detecção de padrões.

Esses trabalhos têm em comum o objetivo de atribuir significado aos dados de segurança, permitindo que eventos distintos sejam correlacionados a partir de suas propriedades semânticas. Em outras palavras, eles tratam o problema da “interpretação”, e não apenas do “armazenamento”. No entanto, apesar de poderosos, esses frameworks apresentam limitações quanto à escalabilidade e à integração com sistemas analíticos de grande volume. O processamento semântico de logs em tempo real ainda é uma tarefa custosa, e muitas dessas soluções não priorizam o desempenho de consulta sobre grandes conjuntos de dados.

Dessa forma, a presente pesquisa se diferencia por adotar uma abordagem mais pragmática: em vez de converter logs em triplas RDF, propõe-se convertê-los para um modelo relacional normalizado e analiticamente eficiente. Assim, busca-se manter parte da capacidade de correlação entre eventos, como a relação entre requisições HTTP e artefatos de arquivo, sem recorrer a camadas semânticas complexas. Em outras palavras, o trabalho não visa substituir os frameworks semânticos, mas complementá-los com uma estrutura relacional que possa servir como base para análises de maior escala.

3.3 INTEGRAÇÃO DE JSON EM BANCOS RELACIONAIS

A terceira linha de trabalhos relacionados diz respeito à integração de documentos JSON em esquemas relacionais. Petković (Petković, 2017) analisou como SGBDs tradicionais (Oracle, PostgreSQL, SQL Server) incorporaram suporte nativo a JSON, destacando limitações em relação ao padrão ANSI SQL/JSON e às operações de consulta sobre dados complexos.

Embora úteis, essas implementações geralmente oferecem apenas funcionalidades básicas, sem otimizações específicas para cenários de análise de dados em larga escala.

Bharathan e Bharathan (Bharthan e Bharathan, 2014) apresentaram o *RelationalJSON*, um método para traduzir documentos JSON em tabelas relacionais a partir de sua estrutura em árvore, reduzindo redundâncias e otimizando consultas. Já Bahta e Atay (Bahta e Atay, 2019) sugeriram abordagens *schema-oblivious*, nas quais o mapeamento entre JSON e tabelas é feito de maneira dinâmica, sem a necessidade de esquemas fixos, o que aumenta a flexibilidade para diferentes formatos de documento. Hermkens (Hermkens, 2016), por sua vez, desenvolveu o ESQLite, uma extensão do SQLite que aplica técnicas de *shredding* (quebra em pares chave-valor) e permite consultas no estilo XPath sobre dados JSON.

Essas propostas atacam o mesmo problema central deste trabalho, o armazenamento relacional de dados semiestruturados, mas com objetivos diferentes. Enquanto o *RelationalJSON* e o ESQLite focam na transformação técnica e no suporte a consultas estruturadas, a presente pesquisa concentra-se na etapa seguinte: avaliar o impacto dessas transformações em um cenário analítico real. Além de converter os dados JSON de incidentes de segurança para um modelo relacional em Segunda Forma Normal (2FN), este trabalho também compara empiricamente o desempenho de consultas entre o ClickHouse e o PostgreSQL, demonstrando a eficiência da abordagem colunar em relação à relacional tradicional. Assim, busca-se preencher uma lacuna entre as propostas teóricas de mapeamento e sua aplicação prática em ambientes de análise de segurança da informação.

3.4 CONCLUSÃO

Este capítulo apresentou os principais trabalhos relacionados em três eixos complementares: ontologias de segurança da informação (Satyapanich et al., 2019; Joshi et al., 2013), frameworks de interpretação semântica de logs e estratégias de integração de JSON em bancos relacionais. Verificou-se que, embora ontologias e modelos semânticos ofereçam uma estrutura conceitual rica, eles ainda são complexos para adoção prática em grandes volumes de dados. Da mesma forma, frameworks de interpretação de logs permitem correlação semântica, mas não priorizam desempenho ou escalabilidade. Por outro lado, as soluções de integração de JSON propõem mecanismos eficientes de mapeamento, porém, em geral, não avaliam de forma empírica o impacto dessas técnicas em bancos analíticos modernos.

Dessa forma, a contribuição deste trabalho se insere exatamente na interseção dessas três linhas. Propõe-se um modelo relacional normalizado para dados de incidentes de segurança, inspirado em métodos como o *RelationalJSON* e o ESQLite, mas voltado especificamente para análise em larga escala no ClickHouse. O PostgreSQL é utilizado apenas para fins comparativos, evidenciando o ganho de desempenho do modelo analítico colunar. O próximo capítulo detalha a metodologia adotada para essa transformação, descrevendo as etapas de modelagem, carga e avaliação de desempenho dos dados.

4 METODOLOGIA

Depois de analisar os trabalhos relacionados no capítulo anterior, ficou claro que, embora existam várias abordagens para lidar com dados de segurança em formatos como JSON, quase nenhuma delas se encaixa perfeitamente na necessidade deste projeto, exceto a utilização de ontologias. As ferramentas genéricas de integração de JSON não aproveitariam todo o potencial de um banco de dados colunar como o ClickHouse para análises rápidas.

Diante disso, foi preciso desenvolver uma metodologia própria. Neste capítulo, será detalhado o passo a passo da solução que foi criada, mostrando como os relatórios de incidentes em formato JSON foram transformados e carregados em um esquema relacional bem definido.

Para organizar a explicação, o capítulo foi dividido em três partes principais. Primeiro, será apresentado o formato exato do dataset JSON de entrada, para que se entenda a estrutura dos dados brutos. Em seguida, será detalhada a modelagem relacional proposta, justificando a criação das tabelas e seus relacionamentos. Por fim, será descrito o script em Python desenvolvido para automatizar todo o processo de extração, transformação e inserção dos dados no ClickHouse.

4.1 FORMATO DO DATASET JSON DE ENTRADA

O dataset que utilizaremos é composto de quase 100 mil arquivos no formato JSON que totalizam aproximadamente 2.5GB de dados de incidentes cibernéticos. São dados referentes ao período de 2010 a 2019, que foram anteriormente parseados de arquivos zip semiestruturados para JSON.

Este exemplo abaixo retrata como o arquivo JSON vem formatado para que o script de extração, transformação e carregamento (ETL) dos dados feito em Python seja executado posteriormente e consequentemente seja feita a inserção dos dados no ClickHouse:

```
{
  "data do relatório:": "2010-07-01_084421 17:17",
  "instituição:": "BB",
  "analista:": "cos",
  "incidente:": "2010-07-01_AA",
  "url:": "http://www.holycross.tw/Item/www/CompraFacil.com.br/
  Pedidos-2010/Pedidos-CompraFacil-3465183BR.exe",
  "hashes (md5):": [
    "2b26548dfa3d390a82a2322d392ce47a connect32.dll",
    "b12420b6f4777433cdba487e05b2b91f Pedidos-CompraFacil-3465183BR.exe"
  ],
  "arquivos criados/acessados:": [
    "C:\\Documents and Settings\\kurumin\\Desktop
    \\Pedidos-CompraFacil-3465183BR.exe",
    "C:\\Documents and Settings\\kurumin\\connect32.dll"
  ],
  "alvos:": "BB",
  "get": [
    "http://208.64.66.172:8099/NETWORK.pac",
    "http://208.64.66.172:8099//",
    "http://208.64.66.172:8099//docs/img/bbfav.html",
    "http://208.64.66.172:8099//inicia/"
  ],
  "post": [
```

```
    "http://208.64.66.172:8099//inicia/login.jsp_aapf-conectar.IDH.php",
    "http://208.64.66.172:8099//inicia/login.jsp.php?aapf.IDH=sim",
    "http://208.64.66.172:8099//inicia/principal.jsp.php??aapf.IDH=sim",
    "http://www.christell.nl//Item/2010/LONG/index.php"
  ],
  "email": "brandon.aaron@gmail.com",
  "banco de dados sql": [],
  "ftp": [],
  "observações": [
    "Proxy no IE:",
    "AutoConfigURL\tSZ\thttp://dns.rejecworks.com/",
    "Proxy no Firefox:",
    "network.proxy.autoconfig_url http://dns.rejecworks.com/",
    "POST downloads?client=navclient-auto-ffox&appver=3.6.3&pver=2.2&
    wrkey=AKEgNitOfLHo1j3E2-I5CIjiHzKRDzOIfvrhZinKGBbvMY5y
    wqco_N6eIwRBFh1rZegN8phnMAi5u-TCox4khmtEKjstVoarg==",
    "POST login.jsp_aapf-conectar.IDH.php",
    "POST login.jsp.php?aapf.IDH=sim",
    "POST login.jsp.php?aapf.IDH=sim",
    "POST login.jsp.php?aapf.IDH=sim",
    "POST login.jsp.php?aapf.IDH=sim",
    "POST login.jsp.php?aapf.IDH=sim",
    "POST login.jsp.php?aapf.IDH=sim",
    "POST login.jsp.php?aapf.IDH=sim",
    "POST login.jsp.php?aapf.IDH=sim",
    "POST login.jsp.php?aapf.IDH=sim",
    "POST login.jsp.php?aapf.IDH=sim",
    "POST login.jsp.php?aapf.IDH=sim",
    "POST login.jsp.php?aapf.IDH=sim",
    "POST login.jsp.php?aapf.IDH=sim",
    "POST login.jsp.php?aapf.IDH=sim",
    "POST login.jsp.php?aapf.IDH=sim",
    "POST login.jsp.php?aapf.IDH=sim",
    "POST login.jsp.php?aapf.IDH=sim",
    "POST login.jsp.php?aapf.IDH=sim",
    "POST login.jsp.php?aapf.IDH=sim",
    "POST login.jsp.php?aapf.IDH=sim",
    "POST login.jsp.php?aapf.IDH=sim",
    "POST login.jsp.php?aapf.IDH=sim",
    "POST login.jsp.php?aapf.IDH=sim",
    "POST login.jsp.php?aapf.IDH=sim",
    "POST login.jsp.php?aapf.IDH=sim",
    "POST principal.jsp.php??aapf.IDH=sim",
    "POST principal.jsp.php??aapf.IDH=sim",
    "POST principal.jsp.php??aapf.IDH=sim",
    "POST principal.jsp.php??aapf.IDH=sim",
    "POST principal.jsp.php??aapf.IDH=sim",
    "POST principal.jsp.php??aapf.IDH=sim",
    "POST principal.jsp.php??aapf.IDH=sim",
    "POST principal.jsp.php??aapf.IDH=sim",
    "POST principal.jsp.php??aapf.IDH=sim",
    "POST index.php"
  ]
}
```

4.2 MODELAGEM RELACIONAL (2FN)

A etapa de modelagem dos dados é um pilar fundamental neste trabalho, pois define a estrutura que irá suportar todas as análises futuras. Para traduzir a natureza hierárququica e semiestruturada do JSON de entrada em um formato otimizado para consultas analíticas e consistência, foi projetado um esquema relacional em Segunda Forma Normal (2FN). Esta abordagem garante que cada atributo dependa totalmente da chave primária e que as redundâncias sejam minimizadas, preservando a integridade das relações entre entidades como incidentes, artefatos e eventos. Além disso, a normalização em 2FN permite representar a complexidade dos relatórios de segurança de forma organizada e escalável, facilitando tanto a execução de consultas agregadas no ClickHouse/PostgreSQL quanto a manutenção e expansão futura do modelo.

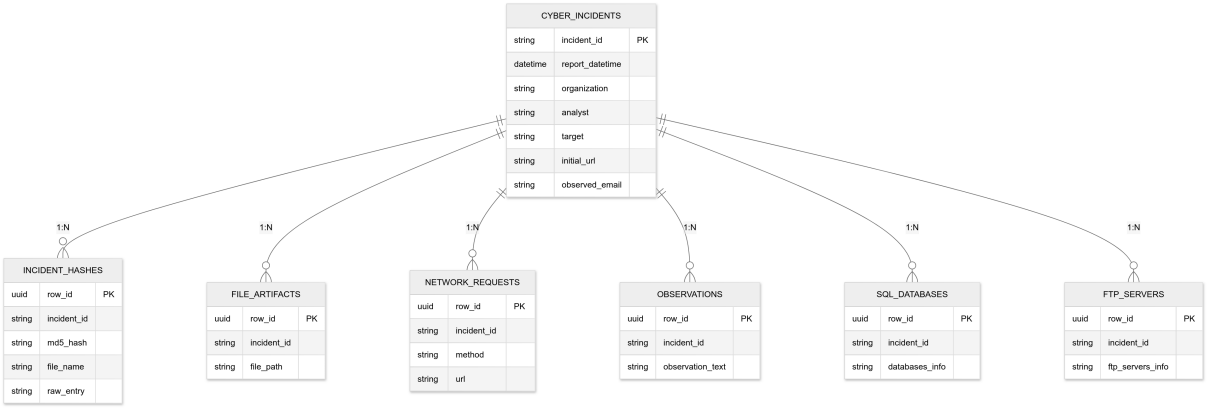


Figura 4.1: Esquema Relacional em 2FN

O modelo proposto é composto por uma tabela central chamada `CYBER_INCIDENTS`, que armazena os dados principais de cada incidente, como identificador único, data do relatório, instituição envolvida, analista responsável, alvo, URL inicial e e-mail observado. Essa tabela atua como a entidade principal do esquema e se relaciona com outras seis tabelas secundárias, cada uma representando um aspecto específico dos relatórios.

As relações entre a tabela principal e as tabelas associadas são do tipo **1:N** (um para muitos). Isso significa que um único incidente pode estar associado a vários registros em cada tabela dependente. Por exemplo, um incidente pode conter diversos hashes de arquivos maliciosos identificados durante a análise, que são armazenados na tabela `INCIDENT_HASHES`. Da mesma forma, a tabela `FILE_ARTIFACTS` registra múltiplos arquivos criados ou modificados no contexto do incidente, enquanto a `NETWORK_REQUESTS` armazena as requisições de rede observadas, incluindo o método HTTP e a URL correspondente. Já as tabelas `SQL_DATABASES` e `FTP_SERVERS` armazenam informações adicionais sobre bancos de dados e servidores FTP comprometidos ou relacionados ao ataque. Por fim, a tabela `OBSERVATIONS` consolida anotações qualitativas dos analistas, permitindo associar comentários e descrições a cada incidente específico.

Essa decomposição segue os princípios da 2FN, pois elimina dependências parciais e garante que cada tabela contenha atributos diretamente relacionados à sua chave primária. Por exemplo, o hash MD5 de um arquivo depende exclusivamente do identificador do registro em `INCIDENT_HASHES` e não de outros atributos do incidente principal. O mesmo raciocínio vale para as URLs e observações, que dependem apenas da ocorrência a que pertencem. Dessa forma, o modelo evita redundâncias, como a repetição de dados do incidente em cada registro, e melhora a integridade referencial, uma vez que todas as tabelas secundárias se conectam à tabela `CYBER_INCIDENTS` por meio da chave estrangeira `incident_id`.

Além de favorecer a consistência, essa estrutura relacional facilita a execução de consultas analíticas no ClickHouse, como o cálculo de quantos incidentes contêm um determinado hash, a contagem de URLs suspeitas por organização ou a análise de padrões de ataques em diferentes períodos. Assim, o modelo proposto não apenas cumpre os requisitos de normalização, mas também se mostra adequado para consultas de grande escala e exploração de dados no contexto da cibersegurança.

O modelo é composto pelos seguintes elementos:

- **Tabela de Incidentes Cibernéticos (`CYBER_INCIDENTS`):** No centro do esquema, esta tabela funciona como a entidade principal, armazenando os atributos que descrevem unicamente cada incidente de segurança. Campos como `report_datetime`, `organization` e `analyst` são atributos de valor único para cada evento. A coluna `incident_id` serve como a chave primária, sendo o identificador exclusivo que conecta o incidente a todos os seus artefatos e observações associadas.

- **Tabelas Dimensionais:** O principal desafio da modelagem era representar os atributos multivalorados presentes no JSON, como listas de hashes e arquivos. A solução foi criar tabelas dimensionais separadas para cada um desses atributos, resolvendo a relação "um-para-muitos"(1:N). Tabelas como `INCIDENT_HASHES`, `FILE_ARTIFACTS` e `NETWORK_REQUESTS` contêm cada uma um único artefato por linha (e.g., um único hash MD5) e estão diretamente ligadas ao incidente correspondente através da chave estrangeira `incident_id`.

- **Otimização para Consultas Analíticas:** A escolha por este modelo simplifica e acelera drasticamente as consultas. Em vez de analisar estruturas complexas de arrays dentro de um campo JSON, é possível realizar operações relacionais padrão, como `JOIN` e `GROUP BY`, de forma altamente eficiente. Por exemplo, para encontrar todos os incidentes relacionados a um hash específico, basta uma busca indexada na tabela `INCIDENT_HASHES`.

- **Integridade e Escalabilidade:** A estrutura normalizada garante a consistência e facilita a manutenção dos dados, pois cada peça de informação é armazenada em seu local lógico, evitando duplicação e anomalias de atualização. Este esquema relacional, portanto, não é apenas um repositório, mas a fundação estratégica que habilita a transformação de dados brutos em uma fonte de inteligência organizada e pronta para ser explorada.

4.3 SCRIPT DE INSERÇÃO DOS DADOS NO CLICKHOUSE

O script desenvolvido em Python tem como objetivo principal automatizar a ingestão de relatórios de incidentes de cibersegurança, originalmente em formato JSON, para um banco de dados colunar ClickHouse, seguindo o esquema relacional previamente definido. A solução foi projetada para ser resiliente e tolerante a falhas, garantindo a integridade dos dados e a continuidade do processamento mesmo diante de inconsistências nos arquivos de entrada.

O processo pode ser dividido em quatro etapas principais:

- **Leitura e Análise Robusta dos Arquivos:** O script inicia varrendo recursivamente um diretório de entrada em busca de arquivos com extensão `.json` ou `.jsonl`. Para cada arquivo, é empregada uma rotina de leitura robusta (`robust_read_json_file`) que tenta múltiplas estratégias de decodificação. Ela é capaz de lidar com diferentes codificações de texto (UTF-8, latin-1), remover o *Byte Order Mark* (BOM), corrigir

vírgulas penduradas no final de objetos JSON e interpretar corretamente tanto um único objeto JSON, uma lista de objetos, o formato JSON Lines (JSONL), quanto múltiplos objetos JSON concatenados no mesmo arquivo.

Algoritmo 1 Função `robust_read_json_file(path)`

```
tentar abrir o arquivo em UTF-8
if falhar then
    tentar novamente em latin-1
end if
remover Byte Order Mark, se presente
tentar carregar como objeto JSON único
if falhar then
    tentar carregar como lista JSON
end if
if falhar then
    tentar carregar linha a linha (formato JSONL)
end if
return lista de objetos JSON válidos
```

- **Limpeza e Normalização dos Dados:** Cada documento JSON lido passa por um processo de normalização. A função `normalize_document` padroniza as chaves do objeto, removendo acentos, convertendo para minúsculas e mapeando sinônimos (e.g., "instituicao" para "instituição:") para um formato canônico. Isso assegura consistência, independentemente de pequenas variações nos arquivos de origem. Funções auxiliares especializadas, como `parse_report_datetime`, são usadas para converter datas em formatos variados para o tipo `DateTime` do ClickHouse.

Algoritmo 2 Função `normalize_document(doc)`

```
for all chave, valor em doc do
    chave_normalizada ← remover acentos e converter para minúsculas
    chave_normalizada ← mapear sinônimos conhecidos
    substituir chave antiga pela chave_normalizada
end for
return documento normalizado
```

Algoritmo 3 Função `parse_report_datetime(valor)`

```
tentar converter valor para datetime em formato ISO
if falhar then
    tentar converter usando formatos regionais alternativos
end if
return objeto DateTime ou nulo em caso de erro
```

- **Mapeamento para o Esquema Relacional (ETL):** Esta é a etapa central da transformação. A função `map_incident_record` recebe um objeto JSON normalizado e o desestrutura em conformidade com o modelo relacional em 2FN.

Ela extrai os campos da tabela principal `CYBER_INCIDENTS`. Um ponto crucial é a gestão do `incident_id`: se presente no arquivo, ele é utilizado como chave primária; caso contrário, um UUID é gerado automaticamente para garantir a unicidade, a menos que o modo estrito (`--strict-missing-id`) seja ativado.

Os campos que contêm listas, como "hashes (md5):", "arquivos criados/acessados:", "get" e "post", são iterados. Para cada item da lista, uma nova linha é gerada para a tabela correspondente (`INCIDENT_HASHES`, `FILE_ARTIFACTS`, `NETWORK_REQUESTS`, etc.), sempre vinculada ao `incident_id` do registro pai. A função `split_md5_entry` é utilizada para separar o hash MD5 do nome do arquivo na tabela de hashes.

Algoritmo 4 Função `map_incident_record(doc)`

if `incident_id` não presente **then**
  gerar UUID
end if
extrair campos principais para `CYBER_INCIDENTS`
for all hash em doc["hashes (md5)"] **do**
  (`md5`, `filename`) ← `split_md5_entry(hash)`
  inserir em `INCIDENT_HASHES`
end for
for all arquivo em doc["arquivos criados/acessados"] **do**
  inserir em `FILE_ARTIFACTS`
end for
for all requisição em doc["get", "post"] **do**
  inserir em `NETWORK_REQUESTS`
end for
return tuplas mapeadas por tabela

Algoritmo 5 Função `split_md5_entry(entry)`

dividir `entry` por espaço ou dois-pontos
return (`md5`, `filename`)

- **Ingestão e Controle de Duplicidade:** A função `ingest_path` gerencia todo o fluxo de ingestão. Antes de inserir um novo incidente, ela verifica se o `incident_id` já foi processado na execução atual, utilizando um `set` em memória (`processed_incident_ids`). Caso o ID já exista, o registro duplicado é ignorado, e um aviso é emitido. Esta verificação previne a inserção de dados redundantes. Os dados, já mapeados para tuplas, são inseridos em batches nas respectivas tabelas do ClickHouse através da função `insert_batches`. O script é projetado para nunca interromper a execução por causa de um único documento malformado; em vez disso, erros de leitura, mapeamento ou inserção são registrados em um arquivo de log opcional (`--errors-log`) para análise posterior, garantindo que o máximo de dados válidos seja processado.

Algoritmo 6 Função `ingest_path(path)`

```
processed_incident_ids ← conjunto vazio
for all arquivo em diretório recursivo do
    for all doc em robust_read_json_file(arquivo) do
        record ← map_incident_record(doc)
        if record.incident_id em processed_incident_ids then
            emitir aviso e CONTINUE
        end if
        adicionar incident_id ao conjunto
        tentar inserir usando insert_batches(record)
        if erro then
            registrar erro no log
            CONTINUE
        end if
    end for
end for
```

Ao final da execução, um resumo é exibido, informando o total de arquivos lidos, incidentes processados, registros duplicados e pulados, e o número de erros encontrados, fornecendo uma visão clara do resultado da operação.

4.4 CONCLUSÃO

Neste capítulo, foi detalhada a metodologia completa para a estruturação e ingestão de relatórios de incidentes de cibersegurança. A abordagem metodológica partiu da definição de um formato de entrada JSON, representativo dos artefatos de um incidente, e estabeleceu como destino um esquema relacional alinhado com os princípios da Segunda Forma Normal (2FN). Conforme visualizado no diagrama, o modelo centraliza os dados factuais na tabela `CYBER_INCIDENTS`, enquanto os atributos multivalorados, como hashes, artefatos de arquivo e requisições de rede, são organizados em tabelas dimensionais satélites (`INCIDENT_HASHES`, `FILE_ARTIFACTS`, `NETWORK_REQUESTS`, etc.).

O componente central desta metodologia foi o desenvolvimento de um script de ingestão em Python, que executa um processo de ETL (Extração, Transformação e Carga) robusto e resiliente. Este script efetivamente desagrega as listas e arrays do JSON de origem, convertendo cada elemento em um registro individual nas tabelas dimensionais correspondentes. A transformação dos dados aninhados para o modelo relacional foi implementada de forma a preservar a integridade referencial, utilizando o `incident_id` como a chave estrangeira que conecta cada dimensão ao seu respectivo fato.

A adoção deste modelo não apenas resolve o desafio da representação de dados complexos, mas também otimiza a base de dados ClickHouse para consultas analíticas de alto desempenho, permitindo agregações e filtros eficientes sobre qualquer uma das dimensões. A implementação da verificação de duplicidade e o registro detalhado de erros, sem interromper o processo, garantem a alta disponibilidade e a confiabilidade da ingestão contínua de dados.

Com a arquitetura de dados estabelecida e o pipeline de ingestão validado, o caminho está preparado para a próxima etapa deste estudo. O capítulo seguinte apresentará os resultados e as análises realizadas sobre o dataset consolidado, demonstrando a eficácia da solução proposta para extrair inteligência e insights a partir dos dados de incidentes de segurança.

5 VALIDAÇÃO

A etapa de validação tem o objetivo de verificar se a modelagem relacional proposta realmente cumpre os objetivos definidos no trabalho e se responde às hipóteses levantadas na introdução. Nesta seção, busca-se demonstrar se a estrutura construída é capaz de resolver perguntas comuns da área de segurança da informação por meio de consultas que exploram os dados armazenados no formato relacional. Assim, este capítulo estabelece a ligação entre a fundamentação teórica da modelagem e sua aplicação prática sobre o conjunto de incidentes.

A validação também permite observar se a separação dos elementos do JSON em tabelas específicas, como artefatos, hashes, URLs e arquivos, contribui para a organização dos dados e para a execução de análises mais claras e eficientes. A intenção é verificar se essa abordagem melhora o processo de extração de informação, possibilita identificar padrões relevantes e auxilia na correlação entre diferentes eventos que aparecem nos incidentes.

Dessa forma, ao aplicar consultas reais no banco de dados e analisar seus resultados, torna-se possível avaliar se a modelagem relacional desenvolvida no trabalho realmente oferece maior estrutura, flexibilidade e precisão para o estudo dos incidentes. Com isso, esta seção contribui para confirmar ou refutar as hipóteses apresentadas anteriormente e para mostrar, na prática, o alcance da proposta do trabalho.

5.1 CONSULTAS EXPLORATÓRIAS DE VALIDAÇÃO

```
SELECT
    organization,
    count() AS total_incidentes
FROM CYBER_INCIDENTS
GROUP BY organization
ORDER BY total_incidentes DESC
LIMIT 10

Query id: bfb78142-5bdd-42b4-8311-9919b4c04758

    ┌─organization─┬─total_incidentes─┐
 1. │ NULL         │            50483 │
 2. │ BB           │              941 │
 3. │ Bradesco     │              405 │
 4. │ Itau-Unibanco│              188 │
 5. │ Banestes     │              179 │
 6. │ 021          │               78 │
 7. │ BNB          │               61 │
 8. │ BANRISUL     │               46 │
 9. │ CEF          │               24 │
10. │ Itaú Unibanco│               13 │
    └──────────────┴──────────────────┘

10 rows in set. Elapsed: 0.009 sec. Processed 52.44 thousand rows, 482.23 KB (6.05 million rows/s., 55.62 MB/s.)
Peak memory usage: 783.20 KiB.
```

Figura 5.1: Query 1: Quais organizações ou alvos (target) sofrem mais incidentes?

A primeira consulta (Figura 5.1) realiza uma operação de agregação sobre a tabela `CYBER_INCIDENTS` para contar quantos incidentes existem por organização. A consulta agrupa os registros pela coluna `organization` e ordena os resultados em ordem decrescente. Esse tipo de operação permite identificar quais instituições aparecem com maior frequência nos registros analisados. No contexto de segurança da informação, essa consulta auxilia na identificação de possíveis organizações mais expostas ou com maior número de notificações de incidentes, o que pode indicar padrões de ataque, problemas de reporte ou concentração de eventos em instituições específicas.

A resposta a esta pergunta é fundamental para a priorização de recursos de defesa. Ao identificar quais organizações ou alvos são mais visados, a equipe de segurança pode alocar mais atenção, monitoramento e recursos de mitigação para proteger esses ativos críticos.

```
SELECT
    analyst,
    count() AS incidentes_atribuidos
FROM CYBER_INCIDENTS
GROUP BY analyst
ORDER BY incidentes_atribuidos DESC

Query id: 40854099-9c0b-43ec-a4f0-a53f16a25235

Connecting to database secdb at localhost:9000 as user default.
Connected to ClickHouse server version 25.9.3.

    ┌─analyst──┬─incidentes_atribuidos─┐
 1. │ west     │                 18121 │
 2. │ barreira │                  7492 │
 3. │ Lirineu  │                  6817 │
 4. │ luislago │                  3573 │
 5. │ cos      │                  2829 │
 6. │ Wilsonr  │                  2469 │
 7. │ diogo    │                  2044 │
 8. │ Pandolfi │                  1867 │
 9. │ Katia    │                  1481 │
10. │ quintas  │                  1283 │
11. │ vander   │                   832 │
12. │ barone   │                   776 │
13. │ katia    │                   726 │
14. │ toledo   │                   585 │
15. │ Vander   │                   359 │
16. │ Altoe    │                   330 │
17. │ lirineu  │                   181 │
18. │ altoe    │                   127 │
19. │ guimaraes│                   118 │
20. │ mikail   │                    90 │
21. │ geraldo  │                    66 │
22. │ freitas  │                    57 │
23. │ zander   │                    39 │
24. │ venancio │                    30 │
25. │ fernando │                    27 │
26. │ Ferreira │                    23 │
27. │ dantas   │                    19 │
28. │ Vanda    │                    18 │
29. │ ferreira │                    15 │
30. │ goulart  │                    11 │
```

Figura 5.2: Query 2: Quais analistas estão trabalhando no maior número de incidentes?

A segunda consulta (Figura 5.2) também utiliza agregação, mas agrupa os incidentes pelo analista responsável. O objetivo é contar quantos incidentes cada analista teve atribuídos. Essa consulta ajuda a entender a distribuição da carga de trabalho entre os analistas envolvidos no tratamento dos incidentes. Em segurança da informação, essa métrica evidencia como o fluxo de análise está distribuído e se há concentração de triagens em poucos analistas, o que pode causar gargalos ou indicar especializações.

Esta consulta permite uma visão gerencial da distribuição de carga de trabalho (workload). Identificar analistas sobrecarregados pode indicar a necessidade de redistribuição de tarefas ou de contratação de mais pessoal, garantindo que os incidentes sejam tratados de forma eficiente e evitando o esgotamento da equipe (burnout).

```
SELECT
    md5_hash,
    count() AS frequencia
FROM INCIDENT_HASHES
GROUP BY md5_hash
ORDER BY frequencia DESC
LIMIT 10

Query id: e7430964-ad0a-4628-8e7c-765f13d1ad06

    ┌─md5_hash─────────────────────────┬─frequencia─┐
 1. │ NULL                             │     311148 │
 2. │ 8836bfdaed1f5f17792f6e95586bf65c │       1106 │
 3. │ 81051bcc2cf1bedf378224b0a93e2877 │        651 │
 4. │ 66e836ea5dc4651bd907f8af30bd8ed5 │        431 │
 5. │ a5ea0ad9260b1550a14cc58d2c39b03d │        427 │
 6. │ cb492b7df9b5c170d7c87527940eff3b │        418 │
 7. │ 489ead3a89cae244857f6a82ce8ef8c1 │        392 │
 8. │ b0747e5e0c4bec220f081bbf63f8e145 │        390 │
 9. │ 30f3680e007d924960fd65524de36601 │        372 │
10. │ 386dd1f6578d71879ae57297535ee7fc │        370 │
    └──────────────────────────────────┴────────────┘
```

Figura 5.3: Query 3: Quais são os 10 hashes de arquivo (MD5) mais comuns encontrados em todos os incidentes?

A terceira consulta (Figura 5.3) agrupa os registros da tabela `INCIDENT_HASHES` para contar quantas vezes cada hash de arquivo aparece no conjunto de incidentes. Como antes, ela ordena os resultados de forma decrescente para destacar os hashes mais frequentes. Esse tipo de agregação é fundamental para identificar amostras de malware que surgem repetidamente nos sistemas analisados. Em segurança da informação, observar a recorrência de determinados hashes auxilia na detecção de campanhas de ataque e na priorização de respostas para artefatos mais disseminados.

Hashes de arquivos são indicadores de comprometimento (IoCs) cruciais. Ao listar os hashes mais frequentes, pode-se identificar campanhas de malware em larga escala ou ameaças persistentes que utilizam os mesmos artefatos. Esta informação é vital para criar regras de detecção e bloqueio no antivírus ou EDR (Endpoint Detection and Response).

```
SELECT
    url,
    count() AS requisicoes
FROM NETWORK_REQUESTS
GROUP BY url
ORDER BY requisicoes DESC
LIMIT 20

Query id: c83a099d-fe0c-489a-a311-859ea865eaff

    ┌─url──────────────────────────────────────────────────────────────────────────────────────────┬─requisicoes─┐
 1. │ http://crl3.digicert.com/DigiCertHighAssuranceEVRootCA.crl                                   │         590 │
 2. │ http://sa.symcb.com/sa.crl                                                                   │         585 │
 3. │ http://apps.identrust.com/roots/dstrootcax3.p7c                                              │         559 │
 4. │ http://se.symcb.com/se.crl                                                                   │         539 │
 5. │ http://crl.geotrust.com/crls/secureca.crl                                                    │         539 │
 6. │ http://www.bradesco.com.br/html/classic/index.shtm                                           │         525 │
 7. │ http://mscrl.microsoft.com/pki/mscorp/crl/MSIT%20Machine%20Auth%20CA%202(1).crl              │         423 │
 8. │ http://wspf.bradesco.com.br/wsgeoip/                                                         │         405 │
 9. │ http://mscrl.microsoft.com/pki/mscorp/crl/Microsoft%20Secure%20Server%20Authority(8).crl     │         367 │
10. │ http://tools.google.com//service/update2                                                     │         349 │
11. │ http://mscrl.microsoft.com/pki/mscorp/crl/mswww(6).crl                                       │         345 │
12. │ http://crl.geotrust.com/crls/gtglobal.crl                                                    │         313 │
13. │ http://pki.google.com/GIAG2.crl                                                              │         287 │
14. │ http://modnovembro.thaieasydns.com/mod//processxxx3.zip                                      │         281 │
15. │ http://modnovembro.thaieasydns.com/mod//processxxx.zip                                       │         270 │
16. │ http://mscrl.microsoft.com/pki/mscorp/crl/mswww(5).crl                                       │         254 │
17. │ http://www.gstatic.com/GoogleInternetAuthority/GoogleInternetAuthority.crl                   │         248 │
18. │ http://crl3.digicert.com/ca3-g28.crl                                                         │         234 │
19. │ http://i1.ytimg.com/crossdomain.xml                                                          │         218 │
20. │ http://www.public-trust.com/cgi-bin/CRL/2018/cdp.crl                                         │         213 │
    └──────────────────────────────────────────────────────────────────────────────────────────────┴─────────────┘

20 rows in set. Elapsed: 0.042 sec. Processed 134.61 thousand rows, 10.41 MB (3.22 million rows/s., 248.97 MB/s.)
Peak memory usage: 29.26 MiB.
```

Figura 5.4: Query 4: Quais são as URLs mais acessadas nas requisições de rede dos incidentes?

A quarta consulta (Figura 5.4) calcula quais URLs são mais requisitadas dentro da tabela `NETWORK_REQUESTS`. A operação agrupa os registros por url e conta a quantidade total de requisições associadas a cada uma delas. Consultas desse tipo são úteis para destacar padrões de comunicação recorrentes que podem estar associados a comportamentos maliciosos, como contatos frequentes com servidores de comando e controle, domínios suspeitos ou serviços externos usados por malware. Com essa lista, é possível atualizar regras de firewall e proxy para bloquear o tráfego de rede para esses destinos maliciosos.

```
SELECT
    toStartOfMonth(report_datetime) AS mes,
    count() AS total_incidentes
FROM CYBER_INCIDENTS
GROUP BY mes
ORDER BY total_incidentes DESC

Query id: 6dbcdc4e-4964-4746-adaf-265e890db5e5

    ┌────────mes─┬─total_incidentes─┐
 1. │       NULL │            45423 │
 2. │ 2014-10-01 │              219 │
 3. │ 2014-05-01 │              155 │
 4. │ 2015-06-01 │              150 │
 5. │ 2015-10-01 │              140 │
 6. │ 2015-08-01 │              138 │
 7. │ 2014-03-01 │              126 │
 8. │ 2014-07-01 │              121 │
 9. │ 2014-11-01 │              120 │
10. │ 2012-05-01 │              117 │
11. │ 2014-09-01 │              112 │
12. │ 2014-04-01 │              111 │
13. │ 2014-08-01 │              109 │
14. │ 2017-01-01 │              109 │
15. │ 2015-04-01 │              108 │
16. │ 2015-07-01 │              108 │
17. │ 2010-08-01 │              108 │
18. │ 2015-03-01 │              106 │
19. │ 2012-03-01 │              105 │
20. │ 2015-11-01 │              104 │
21. │ 2014-02-01 │              102 │
22. │ 2015-12-01 │              101 │
23. │ 2015-09-01 │              100 │
24. │ 2016-01-01 │               99 │
25. │ 2013-09-01 │               99 │
26. │ 2011-06-01 │               97 │
27. │ 2016-02-01 │               97 │
28. │ 2017-03-01 │               97 │
29. │ 2011-08-01 │               96 │
30. │ 2012-06-01 │               95 │
31. │ 2015-05-01 │               95 │
32. │ 2013-06-01 │               93 │
```

Figura 5.5: Query 5: Qual a evolução do número de incidentes reportados ao longo do tempo (por mês)?

A quinta consulta (Figura 5.5) realizada identifica a quantidade total de incidentes por mês por meio da função `toStartOfMonth` aplicada ao campo de data. Esse agrupamento temporal é importante para entender a evolução dos incidentes ao longo dos anos e detectar períodos de maior atividade maliciosa. Em segurança da informação, visualizar sazonalidades ou picos abruptos pode ajudar na investigação de campanhas coordenadas e na alocação de recursos de defesa.

A análise de tendências temporais é essencial para o planejamento estratégico. Ela permite visualizar se o número de incidentes está aumentando ou diminuindo e avaliar o impacto

de novas ferramentas de segurança ou mudanças de política. Esses dados são cruciais para relatórios de gestão e para justificar investimentos na área de segurança.

5.2 AVALIAÇÃO DE DESEMPENHO: POSTGRESQL VS CLICKHOUSE

Nesta seção é apresentada uma comparação de desempenho entre PostgreSQL e ClickHouse a partir das consultas executadas sobre o conjunto de dados modelado neste trabalho. O objetivo é mostrar como cada sistema de gerenciamento de banco de dados se comporta diante de operações típicas de análise de incidentes de segurança, especialmente consultas com agregações, junções e contagem de valores distintos. Essa comparação permite avaliar se a abordagem relacional proposta consegue aproveitar de forma eficiente o processamento analítico oferecido pelo ClickHouse em relação ao PostgreSQL, que segue um modelo mais tradicional de banco transacional.

Os testes foram executados no mesmo ambiente de hardware e software, garantindo condições equivalentes para ambos os sistemas. Em ambos os bancos o conjunto de dados foi carregado integralmente, respeitando o mesmo esquema relacional definido nos capítulos anteriores.

Cada consulta foi executada 5 (cinco) vezes em cada SGBD, e o resultado aqui apresentado corresponde à média desses 5 valores, considerando a forma como cada sistema lida com cache, otimização interna e leitura de páginas em disco. A opção por realizar cinco execuções, em vez de apenas uma, deve-se ao fato de que múltiplas repetições reduzem o impacto de flutuações momentâneas do ambiente, variabilidade de cache e outros fatores externos. Assim, obter a média de várias execuções fornece um retrato mais estável, confiável e representativo do desempenho real de cada SGBD diante do volume de dados utilizado.

Por meio dessa avaliação, o capítulo busca demonstrar como as características arquiteturais de cada sistema influenciam o tempo de resposta das consultas e como isso se relaciona com as necessidades de análise em segurança da informação discutidas ao longo do trabalho.

Especificações de Hardware e Software:

- Equipamento: Dell Inspiron 15 3530
- Processador (CPU): Intel® Core™ i7-1355U (13ª Geração, Raptor Lake) @ 5.000GHz
- Memória Principal (RAM): 16 GB DDR4 @ 1333 MHz
- Sistema Operacional: Ubuntu 24.04.3 LTS (64-bit)
- Versão do PostgreSQL: 16.10
- Versão do ClickHouse: 25.9.3.48

```
secdb=# SELECT
    COUNT(DISTINCT md5_hash) AS total_unique_hashes
FROM
    INCIDENT_HASHES
WHERE
    md5_hash IS NOT NULL;
 total_unique_hashes
---------------------
              155056
(1 row)

Time: 299,465 ms
```

Figura 5.6: Query 6: Total de hashes md5 únicos no PostgreSQL

```
SELECT uniqExact(md5_hash) AS total_unique_hashes
FROM INCIDENT_HASHES
WHERE md5_hash IS NOT NULL

Query id: 699e8a67-b727-47dc-b943-9cbbc000a8db

    ┌─total_unique_hashes─┐
1.  │              155056 │
    └─────────────────────┘

1 row in set. Elapsed: 0.055 sec. Processed 575.23 thousand rows, 13.63 MB (10.53 million rows/s., 249.53 MB/s.)
Peak memory usage: 17.68 MiB.
```

Figura 5.7: Query 7: Total de hashes md5 únicos no ClickHouse

Como podemos perceber, de acordo com as consultas 6 e 7 acima, o tempo gasto para identificar o total de hashes md5 únicos no PostgreSQL foi de 299,465ms, enquanto no ClickHouse a mesma consulta foi feita em 55ms. Portanto pode-se concluir que nessa query específica o desempenho do ClickHouse foi aproximadamente 5,45x superior ao PostgreSQL, que é um banco de dados mais tradicional e não otimizado para esse tipo de consulta.

```
SELECT
    organization,
    uniq(ci.incident_id) AS incidents,
    uniq(ih.md5_hash) AS unique_malware,
    uniq(nr.url) AS unique_urls
FROM CYBER_INCIDENTS AS ci
LEFT JOIN INCIDENT_HASHES AS ih ON ci.incident_id = ih.incident_id
LEFT JOIN NETWORK_REQUESTS AS nr ON ci.incident_id = nr.incident_id
GROUP BY organization
ORDER BY incidents DESC
LIMIT 10

Query id: de46610d-7d2b-42b8-8fd6-f7f6d82d68b7

    ┌─organization─┬─incidents─┬─unique_malware─┬─unique_urls─┐
1.  │ NULL         │     50483 │         149813 │       66047 │
2.  │ BB           │       941 │           2713 │        2351 │
3.  │ Bradesco     │       405 │           1232 │         985 │
4.  │ Itau-Unibanco│       188 │            415 │         507 │
5.  │ Banestes     │       179 │            173 │         260 │
6.  │ 021          │        78 │            174 │         207 │
7.  │ BNB          │        61 │            376 │         290 │
8.  │ BANRISUL     │        46 │            247 │         450 │
9.  │ CEF          │        24 │             97 │         119 │
10. │ Itaú Unibanco│        13 │              7 │          20 │
    └──────────────┴───────────┴────────────────┴─────────────┘

10 rows in set. Elapsed: 0.207 sec. Processed 762.27 thousand rows, 40.54 MB (3.68 million rows/s., 195.78 MB/s.)
Peak memory usage: 242.27 MiB.
```

Figura 5.8: Query 8: Top 10 Organizações Mais Atacadas no ClickHouse

```
secdb=# SELECT
    organization,
    COUNT(DISTINCT ci.incident_id) AS incidents,
    COUNT(DISTINCT ih.md5_hash) AS unique_malware,
    COUNT(DISTINCT nr.url) AS unique_urls
FROM cyber_incidents ci
LEFT JOIN incident_hashes ih ON ci.incident_id = ih.incident_id
LEFT JOIN network_requests nr ON ci.incident_id = nr.incident_id
GROUP BY organization
ORDER BY incidents DESC
LIMIT 10;
 organization  | incidents | unique_malware | unique_urls
---------------+-----------+----------------+-------------
               |     50483 |         150287 |       66234
 BB            |       941 |           2713 |        2350
 Bradesco      |       405 |           1232 |         984
 Itau-Unibanco |       188 |            415 |         506
 Banestes      |       179 |            173 |         259
 021           |        78 |            174 |         206
 BNB           |        61 |            376 |         289
 BANRISUL      |        46 |            247 |         449
 CEF           |        24 |             97 |         118
 Itaú Unibanco |        13 |              7 |          19
(10 rows)

Time: 10302,902 ms (00:10,303)
```

Figura 5.9: Query 9: Top 10 Organizações Mais Atacadas no PostgreSQL

As queries 8 e 9 acima foram desenvolvidas para identificar quais organizações estão sendo mais visadas por ataques cibernéticos e, ao mesmo tempo, entender o nível de sofisticação desses ataques. Basicamente, ela agrupa todos os incidentes por organização e calcula três métricas importantes para cada uma delas. A primeira métrica é simplesmente a contagem de incidentes, ou seja, quantas vezes aquela organização foi atacada. Mas contar só o número de ataques não é suficiente para entender o cenário completo. Por isso, a query também calcula quantos malwares diferentes foram usados contra cada organização, através da contagem de hashes MD5 únicos. Além disso, ela também conta quantas URLs maliciosas diferentes apareceram nos ataques. Essas métricas juntas contam uma história interessante. Por exemplo, se uma organização tem muitos incidentes mas poucos hashes únicos, isso geralmente indica que ela está sofrendo ataques automatizados, provavelmente do mesmo grupo criminoso usando as mesmas ferramentas repetidamente. Por outro lado, se uma organização tem relativamente poucos incidentes mas cada um usa um malware completamente diferente, isso sugere que ela está sendo alvo de oportunidade de vários atacantes diferentes, sem coordenação entre eles. Do ponto de vista técnico, essa query é pesada para o banco de dados porque precisa fazer joins entre três tabelas diferentes e calcular contagens distintas em colunas com alta cardinalidade. O PostgreSQL precisa criar estruturas de hash em memória para rastrear todos os valores únicos, enquanto o ClickHouse consegue processar isso muito mais rápido devido ao seu modelo de armazenamento colunar.

```
secdb=# SELECT
    ih.md5_hash,
    COUNT(DISTINCT ih.incident_id) AS incidents,
    COUNT(DISTINCT ci.organization) AS orgs,
    COUNT(DISTINCT ci.analyst) AS analysts,
    EXTRACT(DAY FROM (MAX(ci.report_datetime) - MIN(ci.report_datetime))) AS campaign_days
FROM incident_hashes ih
JOIN cyber_incidents ci ON ih.incident_id = ci.incident_id
GROUP BY ih.md5_hash
ORDER BY orgs DESC, incidents DESC
LIMIT 1;
 md5_hash | incidents | orgs | analysts | campaign_days
----------+-----------+------+----------+--------------
          |     34598 |   12 |       34 |          3274
(1 row)

Time: 917,244 ms
```

Figura 5.10: Query 10: Hash Mais Perigoso (Máxima Dispersão) no PostgreSQL

```
SELECT
    ih.md5_hash,
    uniq(ih.incident_id) AS incidents,
    uniq(ci.organization) AS orgs,
    uniq(ci.analyst) AS analysts,
    dateDiff('day', min(ci.report_datetime), max(ci.report_datetime)) AS campaign_days
FROM INCIDENT_HASHES AS ih
INNER JOIN CYBER_INCIDENTS AS ci ON ih.incident_id = ci.incident_id
GROUP BY ih.md5_hash
ORDER BY
    orgs DESC,
    incidents DESC
LIMIT 1

Query id: 7e161d61-31da-49f7-ba68-055310ef25a1

    ┌─md5_hash─┬─incidents─┬─orgs─┬─analysts─┬─campaign_days─┐
1.  │ NULL     │     34598 │   12 │       34 │          3274 │
    └──────────┴───────────┴──────┴──────────┴───────────────┘

1 row in set. Elapsed: 0.149 sec. Processed 627.67 thousand rows, 28.33 MB (4.21 million rows/s., 190.10 MB/s.)
Peak memory usage: 137.53 MiB.
```

Figura 5.11: Query 11: Hash Mais Perigoso (Máxima Dispersão) no ClickHouse

As queries 10 e 11 acima têm um objetivo bem específico: encontrar qual malware teve a maior dispersão e impacto no dataset inteiro. Em vez de olhar para as organizações, aqui o foco está nos arquivos maliciosos, identificados pelos seus hashes MD5. A query processa todos os dados e retorna apenas uma linha, correspondente ao hash que foi mais bem-sucedido em sua campanha de ataque. Para determinar qual hash é o "mais perigoso", a query considera várias dimensões simultaneamente. Primeiro, ela conta em quantos incidentes diferentes esse hash apareceu, o que mostra a escala dos ataques. Depois, conta quantas organizações diferentes foram atingidas, revelando a dispersão geográfica ou setorial do malware. A query também verifica quantos analistas diferentes investigaram esse hash, o que indica o quão amplamente reconhecida é essa ameaça. Por fim, calcula a duração da campanha, medindo os dias entre a primeira e a última vez que o hash foi visto. Quando um hash aparece em muitas organizações diferentes, foi investigado por diversos analistas e manteve atividade por meses, isso geralmente indica um malware profissional, possivelmente de um grupo APT ou parte de uma operação de ransomware-as-a-service. Essa query é particularmente desafiadora para o banco de dados porque precisa fazer um join completo entre a tabela de hashes e a de incidentes, calcular quatro métricas de cardinalidade distintas simultaneamente, processar funções de agregação como MIN e MAX para calcular datas, e então ordenar todos os resultados antes de retornar apenas o top 1. É exatamente o tipo de operação analítica onde bancos de dados colunares como o ClickHouse mostram sua superioridade sobre sistemas transacionais como o PostgreSQL.

As tabelas abaixo apresentam o tempo de execução de cada uma das queries feitas acima nos respectivos SGBDs:

Tabela 5.1: Resultados detalhados: Queries 6 e 7

Queries 6 e 7	**PostgreSQL**	**ClickHouse**
Execução 1	0,299s	0,055s
Execução 2	0,308s	0,054s
Execução 3	0,295s	0,058s
Execução 4	0,311s	0,060s
Execução 5	0,304s	0,057s
Média	0,303s	0,057s
Desvio Padrão	0,0065038450s	0,0023874673s

Tabela 5.2: Resultados detalhados: Queries 8 e 9

Queries 8 e 9	**PostgreSQL**	**ClickHouse**
Execução 1	0,917s	0,149s
Execução 2	0,952s	0,156s
Execução 3	0,913s	0,148s
Execução 4	0,928s	0,154s
Execução 5	0,955s	0,158s
Média	0,933s	0,153s
Desvio Padrão	0,0195320250s	0,0043588989s

Tabela 5.3: Resultados detalhados: Queries 10 e 11

Queries 10 e 11	**PostgreSQL**	**ClickHouse**
Execução 1	10,303s	0,207s
Execução 2	9,988s	0,209s
Execução 3	10,215s	0,215s
Execução 4	10,359s	0,199s
Execução 5	9,996s	0,219s
Média	10,172s	0,210s
Desvio Padrão	0,1723447127s	0,0076941536s

Tabela 5.4: Tempo médio de execução das consultas no PostgreSQL e ClickHouse

Consultas	**PostgreSQL**	**ClickHouse**
Queries 6 e 7	0,303s	0,057s
Queries 8 e 9	0,933s	0,153s
Queries 10 e 11	10,172s	0,210s

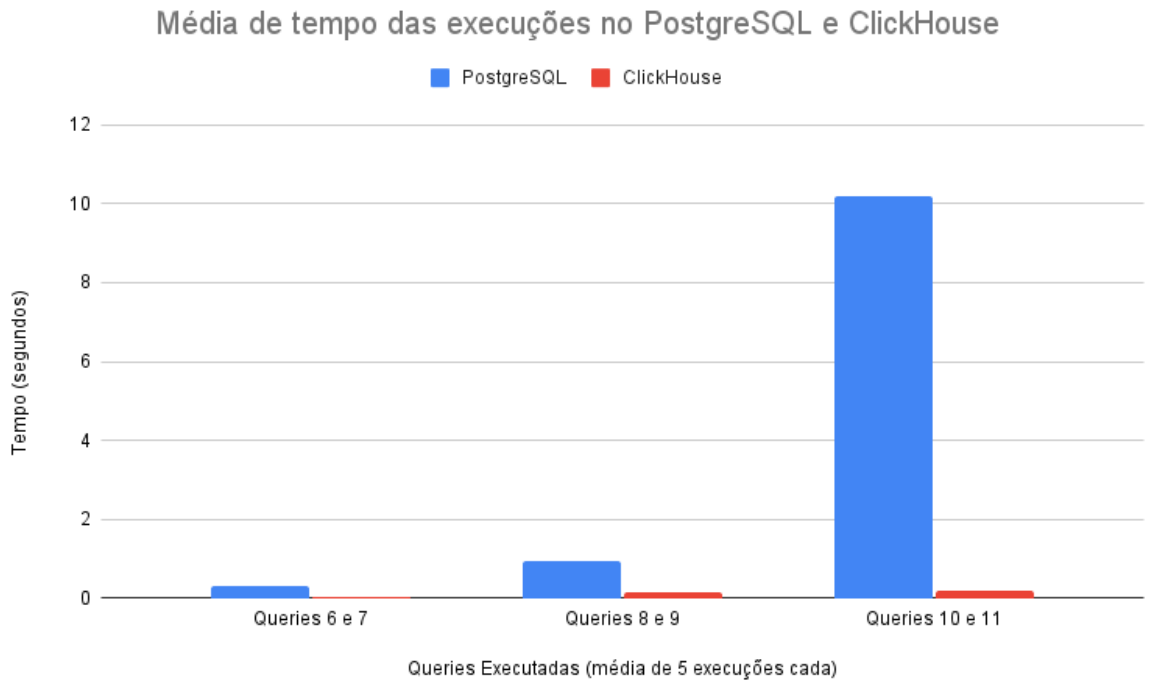


Figura 5.12: Gráfico dos tempos de execução no PostgreSQL e ClickHouse

Os dados apresentados na Figura 5.12 demonstram a viabilidade de executar consultas complexas em um banco tradicional OLTP, como o PostgreSQL. No entanto, observa-se que o ClickHouse apresenta um tempo de resposta inferior, resultado esperado de sua arquitetura colunar otimizada para cargas analíticas. Ressalta-se que o objetivo deste trabalho não é realizar uma comparação direta de desempenho entre SGBDs com propósitos distintos, mas sim evidenciar que a modelagem proposta é funcional no PostgreSQL, ainda que com menor eficiência temporal.

É importante também reconhecer as limitações dessas execuções. As consultas utilizadas não incluem cenários verdadeiramente pesados, como junções complexas envolvendo múltiplas tabelas de grande porte, operações com janelas analíticas ou agregações sobre bilhões de registros. Além disso, o PostgreSQL foi executado sem qualquer processo de otimização, como criação de índices específicos, ajustes de parâmetros de memória, configuração de paralelismo ou tunning voltado para cargas analíticas. Por esse motivo, os resultados apresentados refletem o comportamento padrão das duas ferramentas em um ambiente simples, e não o potencial máximo que o PostgreSQL pode atingir com uma configuração apropriada.

Ainda assim, os resultados têm valor no contexto deste trabalho. Eles mostram como o ClickHouse, por ser um SGBD orientado a colunas e otimizado para cargas analíticas, tende a apresentar tempos de resposta menores em consultas desse tipo. Por outro lado, também evidenciam que o PostgreSQL não é inadequado, mas sim voltado a um uso distinto, mais alinhado a cenários transacionais. Isso reforça que a escolha entre os dois depende da natureza da carga de trabalho e que, em ambientes reais, tanto a estrutura do dataset quanto o nível de configuração aplicado a cada SGBD influenciam significativamente os resultados.

Esse reconhecimento das limitações não diminui a relevância da comparação realizada. Pelo contrário, demonstra que a análise foi conduzida de forma consciente e que os resultados apresentados devem ser interpretados como um ponto de partida, e não como uma avaliação definitiva do desempenho absoluto de cada sistema.

6 CONCLUSÃO

Este trabalho de conclusão de curso apresentou uma investigação prática sobre como dados de segurança da informação podem ser armazenados, processados e analisados de forma eficiente utilizando uma abordagem relacional implementada no SGBD ClickHouse. A partir da construção de um pipeline completo, desde o tratamento inicial dos arquivos até a execução de consultas analíticas de alta performance, foi possível demonstrar que a combinação adequada de modelagem, padronização semântica e processamento otimizado produz resultados sólidos e aplicáveis no contexto real de análise de incidentes de segurança.

A primeira contribuição relevante do trabalho foi a definição de uma modelagem relacional em Segunda Forma Normal (2FN) voltada especificamente para cargas analíticas. Essa modelagem buscou equilibrar normalização e desempenho, garantindo que consultas envolvendo agregações, junções e contagens pudessem ser executadas de maneira eficiente no ClickHouse. Embora simples do ponto de vista conceitual, a modelagem mostrou-se robusta o suficiente para lidar com centenas de milhares de registros parseados a partir de arquivos JSON heterogêneos.

A segunda contribuição foi a adoção da UCO (Unified Cybersecurity Ontology) como mecanismo de padronização. A utilização de um vocabulário semântico consistente permitiu uniformizar termos encontrados nos incidentes e preparar o banco de dados para análises mais avançadas, diminuindo ambiguidades tanto na fase de ingestão dos dados quanto na formulação das consultas. Ainda que de forma inicial, o trabalho demonstrou como ontologias podem auxiliar na organização e no entendimento de grandes volumes de eventos de segurança.

O terceiro ponto significativo foi a implementação de um tratamento criterioso dos arquivos JSON antes de sua inserção no banco. Essa etapa eliminou arquivos incompletos, inconsistentes ou com informações insuficientes, garantindo que o dataset final fosse confiável e adequado para análises posteriores. O uso de Python para estruturar esse pipeline reforçou a independência da solução, possibilitando que outras instituições ou equipes adaptem ou ampliem facilmente o processo, seja para outros tipos de logs, seja para cenários fora do âmbito estritamente relacionado a incidentes de segurança.

Integrados, esses elementos resultaram em uma solução factível e relativamente simples de reproduzir, mas ao mesmo tempo poderosa o bastante para demonstrar o potencial do ClickHouse como ferramenta de análise em ambientes que lidam com grandes volumes de dados. A comparação com o PostgreSQL, apesar de suas limitações, evidenciou a diferença arquitetural entre um SGBD transacional tradicional e um motor analítico moderno, reforçando a adequação do ClickHouse para workloads de natureza investigativa, onde o foco está em leitura intensiva, agregações rápidas e tempo de resposta reduzido.

Ao mesmo tempo, o estudo reconhece suas limitações. Não foram realizadas otimizações no PostgreSQL, não houve criação de índices específicos, nem ajustes finos nos parâmetros de memória e paralelismo. Além disso, as consultas utilizadas, embora representativas, não exploram todo o espectro de análises que um ambiente de um Centro de Operações de Segurança real demanda. Esses fatores indicam que os resultados apresentados devem ser interpretados como uma demonstração de viabilidade, e não como um veredito definitivo sobre a superioridade de uma ferramenta sobre a outra.

6.1 TRABALHOS FUTUROS

A partir dos resultados obtidos, diversos caminhos promissores surgem como possibilidade de continuidade deste trabalho.

Um primeiro ponto é a exploração aprofundada da camada semântica. Embora a UCO tenha sido utilizada para padronizar nomenclaturas, ainda não foi explorada a integração plena com um triplestore, como Apache Jena ou GraphDB. Esses sistemas, por serem bancos NoSQL voltados para representação semântica, permitem consultas baseadas em significado, relacionamentos ontológicos e inferências, recursos que podem complementar de forma poderosa a camada relacional utilizada neste trabalho. A combinação entre um banco colunar para análises estatísticas rápidas e um triplestore para raciocínio semântico abre caminho para arquiteturas híbridas capazes de oferecer respostas mais ricas sobre incidentes complexos.

Outro caminho relevante consiste em expandir o volume de dados analisados. A ingestão atual, embora representativa, não alcança a escala de bilhões de registros comumente vista em empresas de grande porte. Replicar o pipeline em ambientes distribuídos ou em clusters ClickHouse permitiria avaliar sua escalabilidade e limites operacionais.

Também se destaca a possibilidade de incorporar técnicas de processamento em fluxo (stream processing). Como logs de segurança são frequentemente gerados em tempo real, explorar ferramentas como Kafka, Flink ou ClickHouse Keeper poderia aproximar ainda mais a solução de cenários de monitoramento contínuo.

Além disso, melhorias na etapa de pré-processamento podem ser investigadas, como detecção automática de esquemas, inferência de tipos, identificação de anomalias nos arquivos JSON e enriquecimento dos registros com fontes externas (por exemplo, bases de reputação de hashes ou domínios maliciosos).

Por fim, há espaço para otimizações dentro dos próprios SGBDs avaliados. Criar índices, particionar dados, ajustar paralelismo e explorar funcionalidades específicas de cada plataforma pode alterar significativamente a performance observada. Uma comparação mais completa, considerando diferentes graus de tunning, permitiria compreender com maior precisão os limites e vantagens de cada solução.

6.2 CONSIDERAÇÕES FINAIS

Em síntese, o trabalho atingiu seus objetivos ao demonstrar que uma abordagem relacional cuidadosamente planejada, associada a padronização semântica e a um pipeline robusto de tratamento de dados, pode sustentar análises eficientes em segurança da informação. Ao mesmo tempo, abriu portas para investigações futuras que podem expandir essa base para níveis muito mais avançados, seja integrando camadas semânticas, seja explorando novas arquiteturas ou ampliando a escala de dados. Dessa forma, a pesquisa contribui não apenas com uma solução prática, mas também com um ponto de partida para estudos mais amplos sobre análise de dados de segurança em larga escala.

Este projeto encontra-se disponível em: `https://gitlab.c3sl.ufpr.br/tpd20/pipeline-json-to-relational-db-for-security-data`.

O uso do software está autorizado sob os termos da licença MIT License, cujo texto completo encontra-se disponível no repositório ou em: `https://opensource.org/licenses/MIT`.

REFERÊNCIAS

Bahta, M. e Atay, M. (2019). Schema-oblivious json data management: A relational approach. Em *2019 IEEE International Conference on Big Data (Big Data)*, Los Angeles, CA, USA.

Bharthan, A. e Bharathan, D. (2014). Relationaljson: An enriched method to store and query json records. *International Journal of Computer Applications*, 98(7):1–7.

Chaudhuri, S., Dayal, U. e Narasayya, V. (2016). Big data analytics for security. *Proceedings of the IEEE*, 104(11).

Gyrard, D., Barnaghi, P. e Iannella, L. (2021). How fair are security core ontologies? a systematic mapping study. *Journal of Cybersecurity and Privacy*, 1(2).

Hermkens, E. G. W. (2016). Esqlite: A relational database solution for json data with applications in mobile computing. Dissertação de Mestrado, Eindhoven University of Technology, Eindhoven, Netherlands.

Joshi, A., Lal, R., Finin, T. e Joshi, A. (2013). Extracting cybersecurity related linked data from text. Em *2013 IEEE Seventh International Conference on Semantic Computing (ICSC)*. IEEE.

Liu, H., Zhu, Y., Wang, L. e Zhang, J. (2018). A high throughput distributed log stream processing system for network security analysis. *IEEE Access*, 6.

Nimbalkar, P., Mittal, S. e Joshi, A. (2016). Semantic interpretation of structured log files. Em *2016 IEEE International Conference on Big Data (Big Data)*, páginas 3709–3718, Washington, DC, USA.

Petković, M. (2017). Json integration in relational database systems. *Computer Science and Information Systems*, 14(2).

Satyapanich, T., Finin, T. e Ferraro, F. (2019). Extracting rich semantic information about cybersecurity events. Em *2019 IEEE International Conference on Big Data (Big Data)*, Los Angeles, CA, USA. IEEE.

Yao, Y., Zhang, Y., Chen, H. e Xu, W. (2014). A semantic knowledge base construction method for information security. *Journal of Computers*, 9(2).