Conceptual Modeling of Algorithm Parameterization and Results Considering Volatile Data Requirements: A Case Study in the Context of Primary Healthcare

Dimas Cassimiro Nascimento
dimas.cassimiro@ufape.edu.br
Universidade Federal do Agreste de Pernambuco
Garanhuns, Pernambuco, Brazil

Igor Medeiros Vanderlei
igor.vanderlei@ufape.edu.br
Universidade Federal do Agreste de Pernambuco
Garanhuns, Pernambuco, Brazil

Daliton da Silva
daliton.silva@ufape.edu.br
Universidade Federal do Agreste de Pernambuco
Garanhuns, Pernambuco, Brazil

Abstract

Context: Conceptual modeling in software projects is often affected by volatile data requirements, which influence design, logic, and performance. Flexible data modeling is essential to adapt to these changes, especially in healthcare applications where diverse, evolving data types are common. Ensuring adaptability in these settings supports operational efficiency and timely decision-making. **Problem**: Frequent updates in the requirements of Primary Healthcare (PHC) systems increase maintenance costs and cause operational issues, especially when systems need to meet new legislative or operational demands. These changes can disrupt database schemas and impact software development costs. **Proposed Solution**: This paper introduces six schema alternatives designed to model algorithm parameters and outputs flexibly, reducing the impact of changing requirements on system structure. These adaptable schemas aim to minimize the need for frequent reconfigurations. **IS Theory**: The research is based on the Theory of Information Systems Flexibility, which focuses on creating adaptable data structures to ensure long-term efficiency in dynamic settings. This theoretical foundation aligns well with PHC's evolving data needs, requiring resilient and adaptable systems. **Method**: A prescriptive approach was used, combining theoretical evaluation of schema options with a case study in a real-world PHC project. This mixed-method approach provided insights into schema flexibility and usability. **Summary of Results**: We concluded that the proposed database schema is highly effective for adapting to new data requirements, allowing algorithm data and results to be stored as JSON attributes. This flexibility in data inputs and outputs supports evolving PHC requirements. **Contributions and Impact on IS**: This study introduces a database schema for managing volatile data requirements in healthcare IS and reducing IS development costs.

CCS Concepts

• **Information systems**; • **Data management systems**; • **Information systems applications**;

Keywords

Modelagem Conceitual de Dados, Requisitos de Dados Volúveis, Atenção Primária à Saúde

1 Introdução

Na condução de projetos de pesquisa e/ou de desenvolvimento de software, é comum que os requisitos de dados não estejam consolidados em fases iniciais do projeto. Na prática, os requisitos de dados podem ser alterados e/ou evoluídos com base em uma série de fatores, tais como: melhor entendimento dos objetivos do projeto, mudanças na legislação vigente, alterações no orçamento disponível e/ou mudança de requisitos definidos por stakeholders. Os sistemas também podem evoluir ao longo do tempo por diferentes razões, seja pelo fato de expansão de funcionalidades, pela modificação de requisitos técnicos, assim como manutenções corretivas, evolutivas ou adaptativas [14].

Na prática, modificações no esquema conceitual podem impactar dramaticamente os dados e consultas [4], colocando em risco a integridade dos dados [15], gerar manutenções de software dispendiosas visando a atualização de consultas e produzir tempos de manutenção e inatividade inaceitáveis para o sistema [5, 10]. Por consequência, a evolução do esquema é considerada uma tarefa crucial em aplicações de banco de dados, uma vez que exige a co-evolução de dados e do código de implementação do sistema quando alterações de esquema são aplicadas [1]. Portanto, a evolução dos esquemas de banco de dados representa uma tarefa crítica, demorada e propensa a erros [5].

No contexto de requisitos volúveis, torna-se desafiadora a modelagem conceitual da parametrização dos algoritmos a serem desenvolvidos em um projeto de software, uma vez que atualizações recorrentes na modelagem conceitual podem acarretar em impactos significativos em diversos artefatos do software, tais como elementos de interface *front-end*, a implementação da lógica de negócio no *back-end*, assim como os esquemas lógicos e físicos do banco de dados. Como consequência, as mudanças na modelagem conceitual dos dados podem elevar consideravelmente os custos de desenvolvimento e/ou manutenção do software. Além disso, atualmente uma grande variedade de arcabouços e bibliotecas podem ser usadas para acessar bancos de dados na camada de aplicação, os quais permitem que os desenvolvedores usem instruções na linguagem de programação com a qual se sentem confortáveis para acessar o banco de dados, em vez de usar instruções em linguagens de banco de dados. Como consequência, as interações entre o código-fonte e o banco de dados se tornam mais dinâmicas, e portanto, mais complexas de serem entendidas [11]. Outro aspecto desafiador consiste no fato de que a implementação de procedimentos de banco de dados (*stored procedures*) também realizam referências a objetos de bancos de dados [6] e, como consequência, são mais complexos de serem atualizados.

No intuito de diminuir os custos no desenvolvimento de software decorrentes das mudanças na modelagem conceitual, uma das estratégias possíveis consiste em modelar os dados do projeto da

maneira mais flexível e genérica possível, de modo que o esquema de dados possa ser adaptado mais facilmente para representar novos requisitos de dados e/ou possíveis alterações nos requisitos de dados existentes. Neste contexto, este artigo foca na modelagem da parametrização e resultados provenientes da execução de algoritmos a serem explorados em um projeto de software e/ou de pesquisa. Nosso objetivo consiste em propor e avaliar um conjunto de alternativas de modelagens conceituais para representar parâmetros e resultados de algoritmos. As contribuições apresentadas são relevantes para auxiliar na modelagem e desenvolvimentos de sistemas que necessitem representar a parametrização e os resultados da execução de algoritmos visando minimizar os impactos de futuras alterações nos requisitos de dados.

Este trabalho também apresenta uma motivação prática, decorrente da modelagem conceitual de um banco de dados para um projeto de pesquisa e desenvolvimento real[1], cujo objetivo envolve a modelagem e armazenamento de algoritmos a serem empregados para otimizar os serviços de atenção primária à saúde disponibilizados no âmbito do Sistema Único de Saúde (SUS) do Brasil, considerando um contexto de em que o esquema produzido necessita atender a constantes evoluções de requisitos de dados.

Em suma, este trabalho apresenta três contribuições principais: i) modelagem de seis alternativas de esquemas conceituais para modelar parâmetros e resultados de algoritmos de modo flexível e genérico; ii) análise teórica da alternativas de esquema propostos, considerando aspectos como: eficiência, nível de flexibilidade e facilidade de consulta ao dados; e iii) avaliação experimental das alternativas de esquemas conceituais, considerando uma variedade de consultas aos dados e diferentes níveis de volume de dados.

O restante deste artigo está estruturado da seguinte maneira. Na Seção 2, são discutidos os trabalhos relacionados com esta pesquisa. Na Seção 3, são propostas seis alternativas de esquemas conceituais para representar a parametrização e os resultados de algoritmos de maneira genérica e flexível. Na Seção 4, é apresentada a avaliação das alternativas de esquemas conceituais propostos. Na Seção 5, é apresentada a integração do esquema selecionado na arquitetura do sistema desenvolvido. Na Seção 6, é explicada a estratégia adotada para representar o esquema conceitual no nível lógico de banco de dados. Na Seção 7, é apresentado um estudo de caso. Na Seção 8, é apresentada uma avaliação qualitativa dos esquemas conceitual e lógico adotados. Por fim, na Seção 9, são resumidas as principais conclusões do trabalho.

2 Trabalhos Relacionados

A significativa diversidade de partes interessadas e o ambiente usualmente colaborativo e de rápido progresso, que é típico das empresas atuais, dos projetos científicos e de sistemas web tendem a acelerar o ritmo da evolução de requisitos de dados e reduzem tolerância para tempo de inatividade da migração [3]. A seguir, são apresentados e discutidos diversos trabalhos do estado da arte que visam facilitar a tarefa de evolução de esquemas de banco de dados.

Em [11], os autores discutem as complexidades envolvidas na evolução de esquemas e migração de dados em bancos de dados NoSQL, destacando as diferenças em comparação com bancos de dados relacionais tradicionais. Além disso, o artigo explora as abordagens atuais para lidar com a evolução de esquemas e migração de dados em sistemas NoSQL, identificando oportunidades para melhorias e inovações futuras nesta área. Por sua vez, em [6], os autores investigam o problema de lidar com inconsistências que podem surgir em programas de software devido à evolução do esquema de um banco de dados. O artigo demonstra potenciais problemas decorrentes da evolução do esquema do banco de dados e como essa evolução pode afetar a consistência e o desempenho dos programas que interagem com o banco de dados. O artigo também propõe técnicas e abordagens para detectar e prevenir tais inconsistências.

Outros trabalhos investigam uma perspectiva mais prática dos problemas associados com a evolução de esquemas de bancos de dados. Os autores de [1] propõem abordagens para atualizar esquemas, migrar dados e manter a integridade do sistema durante o processo de evolução dos esquemas. O artigo discute como aplicar versionamento de esquemas, migrações automatizadas e testes de regressão para minimizar o impacto da evolução do banco de dados. Em [2], os autores investigam os desafios relacionados à evolução de esquemas em bancos de dados por meio de um estudo de caso específico. O artigo examina como as mudanças no esquema de banco de dados impactam sistemas existentes e como os desenvolvedores e administradores lidam com essas mudanças. Os autores exploram as estratégias existentes para realizar alterações no esquema, incluindo a avaliação de impacto, a migração de dados e a atualização de sistemas que dependem do banco de dados. Com base no estudo de caso, o artigo apresenta diretrizes e recomendações para melhorar a gestão da evolução do esquema do banco de dados em cenários do mundo real.

Com base no levantamento realizado na literatura relacionada, no melhor do nosso conhecimento, não foram encontrados trabalhos que focam explicitamente na modelagem conceitual da parametrização e resultados de algoritmos com foco em requisitos de dados volúveis. Neste contexto, este trabalho visa endereçar este problema ao propor e avaliar diferentes alternativas de esquemas conceituais utilizando uma metodologia baseada na agregação quantitativa de valores de métricas para avaliação de esquemas conceituais.

3 Definição do Problema

Nesta seção, o problema investigado é apresentado formalmente. Com base em um conjunto de requisitos de dados inicial R_0, é realizado um projeto de banco de dados conceitual, produzindo uma versão inicial do esquema conceitual EC_0. Com base em EC_0, são gerados um ou mais esquemas lógicos de banco de dados ($\{L_1, L_2, \cdots, L_n\}$), os quais são utilizados por um conjunto de sistemas $\{S_1, S_2, \cdots, S_m\}$. Após uma mudança nos requisitos de dados, produzindo uma nova versão dos requisitos R_1, os requisitos podem acarretar alterações em EC_0, possivelmente produzindo uma nova versão no esquema conceitual denominada EC_1. Exemplos de alterações no esquema conceitual são [1, 3]: adição, remoção, atualização ou renomeação de entidades, relacionamentos, restrição de cardinalidade, restrição de participação ou atributos. Neste caso, as alterações refletidas em EC_1 podem requerer a alteração no conjunto de esquemas lógicos $\{L_1, L_2, \cdots, L_n\}$ e/ou nos respectivos sistemas ($\{S_1, S_2, \cdots, S_m\}$) que utilizam estes esquemas lógicos.

[1]o referido projeto de pesquisa é financiado pelo Ministério da Saúde e apoiado pelo CNPq

Neste trabalho, consideramos um conjunto de requisitos inicial de dados R_0^{alg} que consiste no armazenamento da execução, configuração paramétrica e resultados produzidos por algoritmos empregados no sistema, os quais são executados por um ou mais usuários de um sistema. O problema consiste em modelar um esquema conceitual flexível e genérico EC^*, de modo que, caso ocorram mudanças no conjunto de requisitos inicial da configuração paramétrica e/ou resultados dos algoritmos, produzindo um novo conjunto de requisitos R_1^{alg}, os custos de atualizar EC^*, $\{L_1, L_2, \cdots, L_n\}$ e $\{S_1, S_2, \cdots, S_m\}$ sejam minimizados. As características de flexibilidade e nível de generalização do esquema almejado (EC^*) estão relacionadas com a métrica de manutenibilidade do esquema conceitual, i.e., a capacidade de alterar facilmente o esquema, característica esta que representa um fator chave na concepção e evolução do esquemas de dados [9].

Além de flexibilidade e nível de generalização, objetiva-se também a modelagem de um esquema conceitual EC^* que apresente facilidade de acesso aos dados de configuração e execução dos algoritmos, assim como um bom desempenho na execução das consultas sobre estes dados.

4 Modelagens Propostas

Nesta seção, são propostas seis alternativas de modelagem conceitual da parametrização e resultados provenientes de algoritmos utilizando o modelo ER. As alternativas diferem entre si em relação a: i) número de entidades e relacionamentos; ou; ii) forma de representação dos dados (na forma de atributos ou entidades). Os conceitos (concretos ou abstratos) a serem representados pela modelagem conceitual são: usuário do sistema, configuração de valores de parâmetros, tipos e valores de parâmetros, algoritmo, execução do algoritmo, tipos e valores dos resultados produzidos pelo algoritmo. Na Figura 1, são apresentadas três alternativas de modelagem, as quais são denominadas: A, B e C. Na Figura 2, são propostas outras três alternativas de modelagem: D, E e F.

A alternativa A modela as entidades Usuário, Parâmetro e Resultado, sendo Usuário e Parâmetro associados pelo relacionamento Configura e as entidades Usuário e Resultado associadas pelo relacionamento Produz. A alternativa B difere da alternativa B ao considerar um relacionamento adicional entre as entidades Parâmetro e Resultado, permitindo identificar quais valores de parâmetros produziram os respectivos resultados gerrados pelo algoritmo executado. Por sua vez, a alternativa de modelagem C representa a execução do algoritmo como uma entidade denominada Execução, a qual é associada com as entidades Resultado e Parâmetro por meio dos relacionamentos Produz e Configura, respectivamente.

As alternativas de modelagem D, E e F, apresentadas na Figura 2, diferem das alternativas anteriores pelo fato de explorarem atributos do tipo JSON para representar os parâmetros e/ou os resultados provenientes da execução dos algoritmos. Desse modo, a alternativa de modelagem D difere da alternativa C ao substituir a entidade Parâmetro pela representação dos valores dos parâmetros do algoritmo como um atributo JSON na entidade Execução. Na alternativa E, a cardinalidade do relacionamento Produz é alterada de (1..n) para (1..1), uma vez que todos os valores produzidos pelo resultado do algoritmo são armazenados na forma de um atributo JSON na entidade Resultado. Por fim, na alternativa de modelagem F, tanto os parâmetros quanto os resultados do algoritmo são representados na forma de atributos JSON da entidade Execução. É importante ressaltar que, ainda que a associação de tipos de dados aos atributos de um esquema conceitual não faça parte da notação padrão do modelo ER, esta notação foi adotada neste trabalho no intuito de facilitar a interpretação dos diagramas proposto.

5 Avaliação

Considerando o problema definido na Seção 3, a avaliação das alternativas de modelagem propostas na Seção 4 foi realizada levando em consideração o nível de manutenibilidade associado às alternativas. O nível de manutenibilidade pode ser mensurado como o nível de facilidade para adaptar e evoluir o esquema conceitual, e esta características está intrinsicamente relacionada com a capacidade de adaptar a modelagem conceitual para considerar novos requisitos de dados [9]. O nível de manutenibilidade de um esquema conceitual de dados depende no seu nível del compreensibilidade [7], isto porque, o esquema conceitual deve ser bem compreendido antes que quaisquer alterações desejadas possam ser identificadas, projetadas e implementadas [7].

Métricas de modelagem como a compreensibilidade e manutenibilidade são difíceis de medir objetivamente e usualmente precisam de ser avaliadas de uma forma subjetiva, por exemplo, utilizando a opinião de especialistas, expressas por meio de uma abordagem de uma avaliação quantitativa que atribua pontuações e ponderações a diferentes aspectos do esquema conceitual. O mapeamento de aspectos relacionados com a compreensibilidade e manutenibilidade por meio de uma abordagem de avaliação quantitativa permite a comparação mais objetiva de alternativas de design e, portanto, uma seleção entre várias alternativas de modelagem de dados conceituais [9].

5.1 Metodologia de Avaliação

Apesar de existirem métricas para avaliar a complexidade de modelagens conceituais de dados [8, 9, 12], estas métricas não são indicadas para avaliar a manutenibildiade de um diagrama ER. Por este motivo, será adotada uma avaliação quantitativa dos diagramas ER por meio de uma abordagem sistemática que mapeie a avaliação qualitativa dos modelos em pontuações e agregue as pontuatações produzidas por meio de um sistema de ponderações. As ponderações adotadas, por sua vez, consideram as métricas que influenciam de maneira mais expressiva a manutenibilidade do modelo. As dimensões consideradas na avaliação qualitativa são: Simplicidade - Si, Flexibilidade Fl, Completude - Cp, Interpretabilidade - Int e Facilidade de Implementação - FImp. É notório que estas dimensões do diagrama possuem influência na manutenibilidade do banco de dados, considerando potenciais alterações e evoluções nos requisitos de dados. Além disso, uma vez que estas dimensões influenciem umas nas outras, são ponderadas de modo mais expressivo as dimensões que possuem maior influência sobre a manutenibilidade do diagrama ER, assim como as dimensões que influenciam positivamente outras dimensões que influenciam a manutenibilidade. As influências positivas (denotadas por +) entre as dimensões consideradas são [12, 13]: Si+It, Si+Fl, Si+FImp e Int+Cp. Com base nestas interações, as dimensões Si eInt são ponderadas (valor de

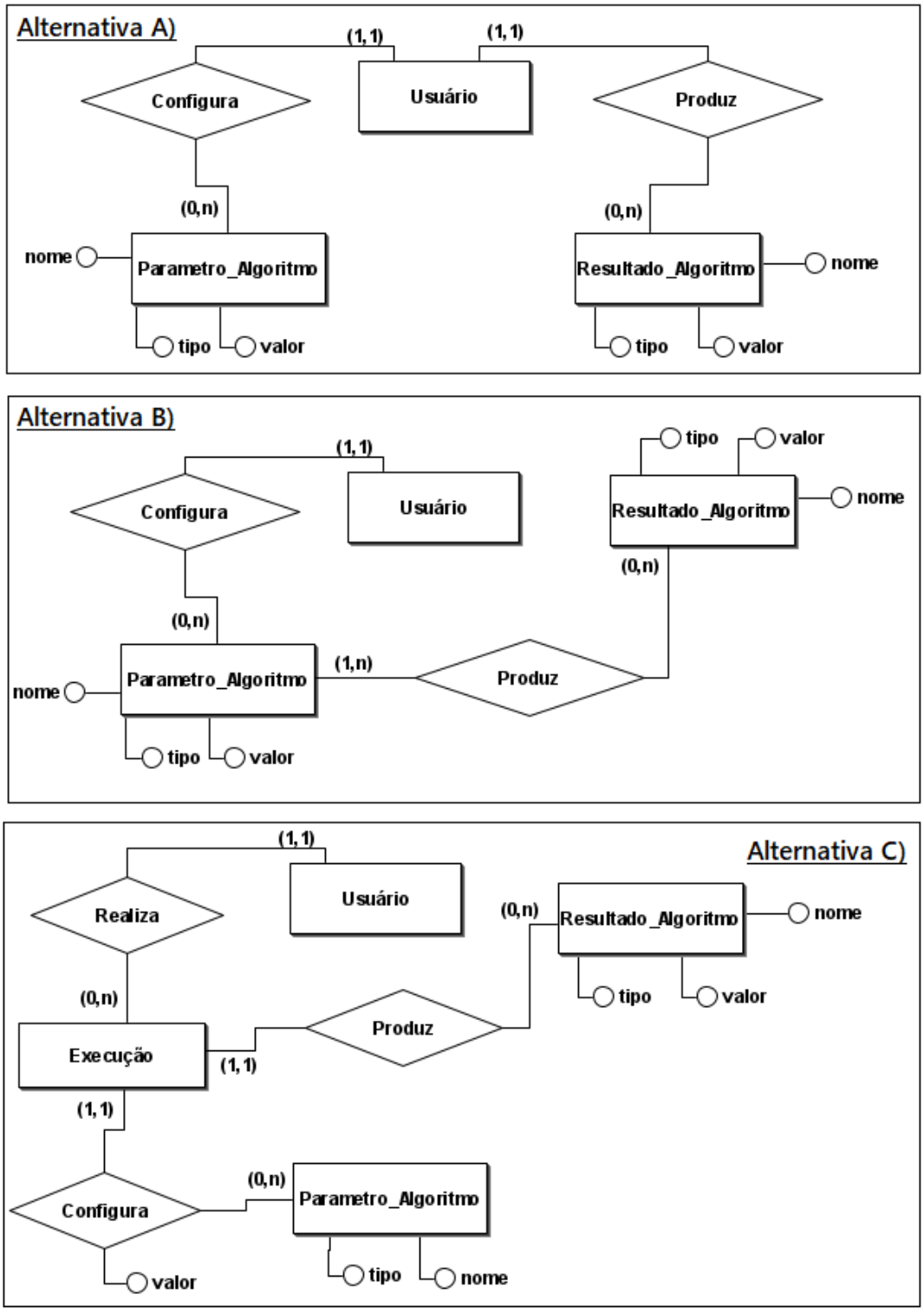

Figure 1: Três alternativas propostas (A, B e C) para modelagem conceitual de configuração paramétrica e resultados produzidos por algoritmos.

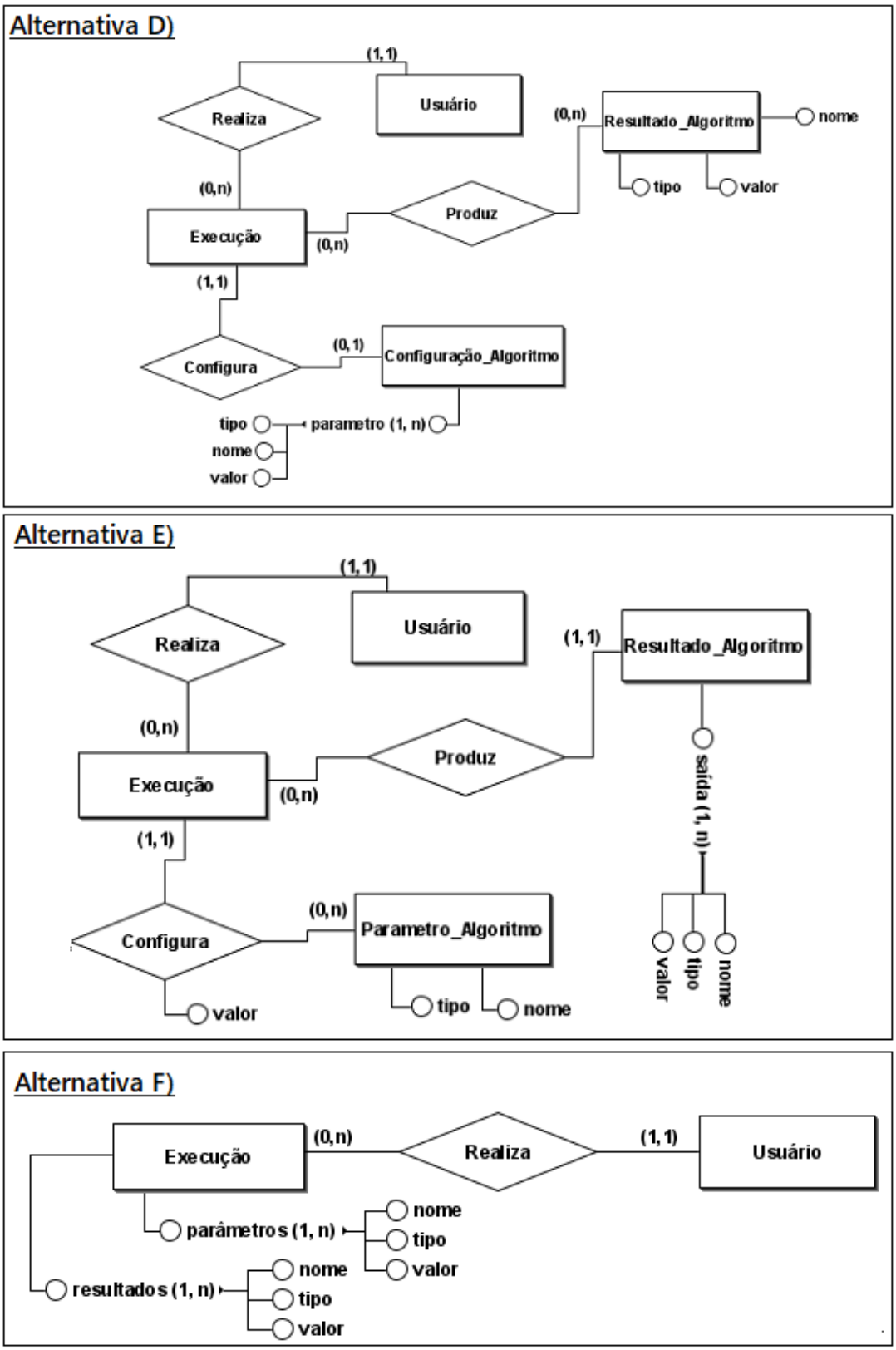

Figure 2: Três alternativas propostas (D, E e F) para modelagem conceitual de configuração paramétrica e resultados produzidos por algoritmos.

peso w) de forma mais expressiva na avaliação das alternativas de diagrama propostas: Si ($w = 3$), Fl ($w = 3$), Cp ($w = 3$), Int ($w = 1$) e FImp ($w = 1$). As dimensões Si e Fl foram ponderadas com maior peso porque estão associadas a mais influências positivas em outras métricas. Por sua vez, as dimensões Fl e Cp foram também ponderadas com maior peso porque possuem relação direta com o problema investigado (Seção 3), em outras palavras, o objetivo do diagrama deve priorizar a completude dos requisitos de dados e facilitar a evolução do esquema (flexibilidade).

5.2 Resultados

Para cada diagrama proposto nas Figuras 1 e 2, foi atribuída uma das seguintes pontuações (s) considerando cada dimensão avaliada: 1, 2 ou 3, tal que pontuações maiores representam maior qualidade da dimensão. No intuito de agregar as pontuações considerando os pesos atribuídos às dimensões, foi empregada uma média ponderada das pontuações, produzindo uma métrica agregada ($quality$) de qualidade do diagrama, calculada como $quality(E) = \left(\sum_{d \in D} s_d(E) * w_d\right) * \left(\sum_{d \in D} d\right)^{-1}$, onde E é um esquema conceitual, D é o conjunto de dimensões avaliadas, $s_d(E)$ é a pontuação do esquema E considerando a dimensão d e w_d é o peso atribuído à dimensão d. A pontuação atribuída aos diagramas propostos é apresentada na Tabela 1.

Diagrama	Si	Fl	Cp	Int	FImp
A	3	2	1	3	3
B	3	2	2	2	2
C	2	3	3	3	3
D	2	3	3	3	2
E	2	3	3	3	2
F	3	3	3	3	1

Table 1: Avaliação das dimensões de qualidade dos diagramas conceituais propostos.

As pontuações dos diagramas, mostradas na Tabela 1, foram obtidas com base na média das pontuações atribuídas pelos pesquisadores e desenvolvedores do projeto. As principais justificativas associadas às pontuações atribuídas são detalhadas a seguir.

O diagrama A dificulta a identificação de quantas execuções foram realizadas pelo usuário, assim como não permite identificar quais combinações de configurações paramétricas produziram quais conjuntos de resultados pelo algoritmo. O diagrama B resolve o último problema mencionado por meio do relacionamento Produz, mas dificulta a identificação de quantas execuções do algoritmo distintas foram disparadas pelo usuário. O diagrama C resolve estes problemas ao representar tanto a entidade Entidade quanto o relacionamento Produz, porém, apresenta maior complexidade em relação ao número de entidades e relacionamentos, quando comparado com as demais alternativas de diagramas. Aos diagramas D e E foram atribuídas pontuações idênticas, tendo sido penalizados nas dimensões Si e FIm devido a quantidade de entidades e relacionamentos e a necessidade de armazenar e manipular dados no formato JSON, respectivamente. Por fim, o diagrama F claramente apresenta maior simplicidade de representação, porém requer uma implementação mais complexa devido à necessidade de manipular os dados de parâmetros e resultados ambos em formato JSON.

Considerando os pesos atribuídos às dimensões neste trabalho e as pontuações associadas aos diagramas propostos apresentadas na Tabela 1, as seguintes pontuações de qualidade agregada foram calculadas: $quality(A) = 2,18$, $quality(B) = 2,27$, $quality(C) = 2,73$, $quality(D) = 2,64$, $quality(E) = 2,64$, $quality(F) = 2,82$. Portanto, com base na metodologia de avaliação adotada, os esquemas conceituais C e F apresentam-se como as melhores alternativas para a representação da configuração paramétrica e resultados de algoritmos, visando maximizar principalmente a flexibilização do esquema considerando requisitos de dados futuros.

6 Representação do Esquema no Nível Lógico do Banco de Dados

Para o projeto discutido, o esquema conceitual escolhido foi representado no nível lógico utilizando arquivos JSON para definir os parâmetros e resultados dos algoritmos. Essa decisão foi fundamentada em quatro principais motivos. O primeiro motivo está relacionado com o fato de que o formato JSON permite estruturar dados de maneira flexível, aceitando facilmente a inclusão de novos campos e a modificação de campos existentes sem exigir alterações significativas no esquema do banco de dados. Essa característica é particularmente valiosa quando os requisitos de dados são suscetíveis a mudanças, pois evita a necessidade de revisões frequentes no esquema conceitual, simplificando a adaptação a novos requisitos.

O segundo motivo decorre do fato de que o formato JSON é altamente adequado para representar entradas e saídas complexas dos algoritmos, incluindo listas, mapas e estruturas hierárquicas. Ao modelar parâmetros de algoritmos que podem variar em complexidade e formato, JSON proporciona uma forma prática de encapsular essa variabilidade, permitindo que diferentes algoritmos e versões utilizem estruturas de dados personalizadas sem prejudicar a consistência dos dados armazenados. O terceiro motivo está associado com a capacidade do formato JSON em ser amplamente compatível com diversas plataformas e linguagens de programação. Isso facilita a comunicação entre os componentes do sistema e a integração com APIs de serviços externos. Como JSON é um padrão comum de intercâmbio de dados, ele permite que diferentes módulos interajam com o sistema sem necessidade de conversões complexas, economizando tempo e reduzindo possíveis erros de implementação.

Por fim, é importante ressaltar que muitos Sistemas de Gerenciamento de Banco de Dados, como PostgreSQL, possuem suporte nativo para JSON, permitindo consultas diretas aos dados estruturados nesse formato. Esse suporte possibilita a execução de consultas detalhadas e a recuperação de dados específicos dentro do objeto JSON, facilitando a análise de resultados dos algoritmos e a extração de informações sem a necessidade de transformar os dados para outros formatos.

Esses fatores tornam o uso de JSON uma solução robusta e adaptável para a modelagem lógica dos dados, garantindo que o sistema seja preparado para mudanças futuras nos requisitos de dados e proporcionando uma integração ágil e eficiente entre os diferentes módulos e algoritmos do projeto.

7 Estudo de Caso

Após a definição do modelo escolhido, foi realizado um estudo de caso que envolveu sua implementação em um sistema em desenvolvimento real. Conforme mencionado brevemente na introdução, esse sistema tem como objetivo principal a captação e o armazenamento de dados relacionados ao Sistema Único de Saúde (SUS) do Brasil, que serão posteriormente utilizados como entrada para algoritmos de previsão e otimização.

Os dados a serem capturados pelo sistema são provenientes de diversas fontes, como o IBGE, o Ministério da Saúde, sites de prefeituras, entre outras, e apresentam uma variedade de formatos, estruturas e granularidades. A natureza heterogênea desses dados foi um dos principais motivadores para a elaboração do modelo proposto, uma vez que, nas fases iniciais do desenvolvimento, a equipe enfrentava um alto grau de incerteza em relação aos dados que seriam obtidos.

Para lidar com esse problema, foram adotadas estratégias como o uso de padrões abertos, APIs, mecanismos de transformação de dados e identificadores padronizados. No estágio atual do desenvolvimento, constatou-se que o esquema proposto também favorece a interoperabilidade com sistemas externos, uma vez que já foi possível integrar dados de fontes como IBGE, DATASUS e e-SUS sem grandes dificuldades. Esta seção apresenta detalhes deste estudo de caso.

7.1 Arquitetura do Sistema

O projeto divide-se em três áreas especializadas: o **Serviço de Dados** foca na coleta e processamento de dados para criar previsões baseadas em informações históricas e atuais; o **Serviço de Otimização** desenvolve algoritmos para modelagem avançada, visando decisões eficientes na gestão de recursos; e o **Serviço de Software** trabalha no desenvolvimento de um sistema web interativo, permitindo que os usuários interajam com os algoritmos criados. Essa arquitetura modular, representada na Figura 3, permite que cada módulo, desenvolvido por uma equipe diferente, seja construído com diferentes tecnologias e linguagens de programação. O banco de dados compartilhado **Modelo Interno** contribui para a integridade dos dados, proporcionando uma visão unificada para todos os componentes e otimizando a comunicação entre os serviços, ao reduzir a necessidade de transferência frequente de grandes volumes de informações.

7.2 Comunicação Entre Serviços com Parametrização Volúvel

A comunicação entre os serviços é realizada exclusivamente por meio do modelo de comunicação assíncrona, mediado pelo componente RabbitMQ. Essa escolha garante o atendimento às demandas de tempo de execução mais longo dos algoritmos de otimização e previsão, adicionando flexibilidade ao permitir que os serviços operem de maneira independente e não bloqueante. Os parâmetros e os resultados da execução dos algoritmos são armazenados no banco de dados relacional, na relação Execucao, representada na Figura 4, que implementa o esquema conceitual definido na Figura 2 (E). Assim, nas mensagens enviadas ao RabbitMQ, é necessário informar apenas o identificador do registro que contém os parâmetros. Os atributos Inicio e Fim armazenam o *timestamp* referente aos instantes de inicialização e término da execução do algoritmo, já os atributos Cod_Retorno e Msg_Retorno armazenam, respectivamente, o código e a mensagem de retorno da execução, através dos quais o usuário poderá ser informado sobre o status da finalização da execução do algoritmo.

Uma Execucao de algoritmo tem início quando um usuário faz uma solicitação, através do *front-end*, informando o algoritmo desejado e configurando os parâmetros necessários. A partir desse ponto, o *front-end*, codifica os parâmetro em um objeto JavaScript (json) e os envia através de uma requisição *HTTP* para o **Servico de Software**. Em seguida, o **Serviço de Software** armazena o registro correspondente à requisição na tabela Execucao, captura o identificador gerado e adiciona uma mensagem à fila do *RabbitMq*. O **Servico de Otimização** então: (i) captura essa mensagem; (ii) utiliza o identificador para ler o registro no banco de dados; (iii) inicializa o objeto que implementa o algoritmo na versão indicada; (iv) converte o parâmetro do formato JSON para um objeto da linguagem de programação utilizada; (v) executa o algoritmo; e (vi) escreve o resultado, o código e a mensagem de retorno no banco de dados.

7.3 Exemplo de Execuções

A seguir, serão apresentadas duas iterações com algoritmos de otimização. Na Figura 5 (A) observa-se a chamada ao algoritmo de otimizar localizações de unidades de saúde, cujo enunciado é o seguinte: *"Dado o município de código 2611606, encontre o conjunto ótimo de localidades para instalação de unidades de saúde, garantindo que todas as áreas prioritárias estarão cobertas. Para realizar esta otimização, utilize os dados gerados na previsão de demanda de código 1123"*. Por sua vez, na Figura 5 (B) observa-se a chamada ao algoritmo de realocação de unidades de saúde com o enunciado: *"Considerando o município de código 2611606 e utilizando os dados de demandas históricos, encontre as melhores localizações para realocar as unidades de saúde de código 116, 123, 145 e 152"*. É evidente que as listas de parâmetros das duas chamadas diferem significativamente em quantidade e tipo de dados. Dessa forma, cada algoritmo deve implementar uma classe, no padrão POJO, para armazenar os seus parâmetros. Ambos algoritmos retornam como saída uma lista de pontos geográficos.

Além dos exemplos apresentados acima, o esquema demonstrou adaptabilidade para outros tipos de processamento, como **importação de grandes volumes de dados heterogêneos** e **predição de indicadores estatísticos**, demonstrando capacidade de adaptação do sistema a diferentes domínios dentro da saúde pública, maximizando sua aplicabilidade.

7.4 Observações Realizadas no Estudo de Caso

A implementação realizada no estudo de caso desempenhou um papel fundamental na validação da escolha entre os modelos conceituais apresentados na Seção 4 e avaliados na Seção 5. Através desta implementação, foi possível reafirmar os valores subjetivos das métricas atribuídas ao modelo F. Em outras palavras, a prática permitiu observar que o modelo selecionado demonstra características favoráveis em termos de simplicidade, flexibilidade, completude, interpretabilidade e facilidade de implementação. Como consequência, foi possível avançar bastante no desenvolvimento do sistema

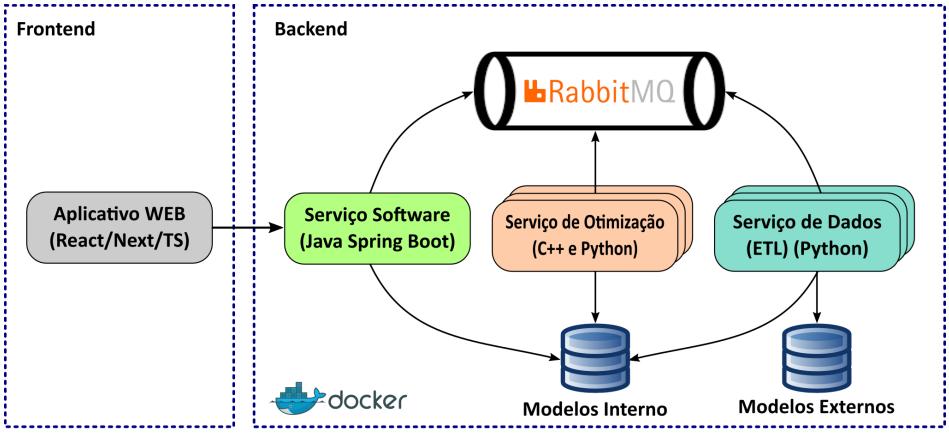

Figure 3: Arquitetura modular do sistema.

EXECUCAO

ID	ALGORITMO	VERSAO	PARAMETROS	RESULTADOS	INICIO	FIM	COD_RETORNO	MSG_RETORNO

Figure 4: Relação Execução.

```
{
  "algotimo": "otimizar_localizacao",
  "versao": "v1",
  "parametros": {
    "cod_municipio": 2611606,
    "tipo_otimizacao": "previsao",
    "id_previsao": 1123,
    "areas_prioritarias": [{
      "Type": "Polygon",
      "coordinates": []
    }]
  }
}
```

(A)

```
{
  "algotimo": "realocar_unidades",
  "versao": "v1",
  "parametros": {
    "cod_municipio": 2611606,
    "tipo_otimizacao": "historico",
    "unidades_realocacao": [116, 123, 145, 152]
  }
}
```

(B)

Figure 5: Execuções de Algoritmos de Otimização.

mesmo diante das incertezas referentes ao modelo de dados dos algoritmos.

Quando comparada aos modelos tradicionais, a abordagem adotada oferece maior flexibilidade, pois evita a rigidez estrutural e elimina a necessidade de constantes atualizações na camada de dados, regras de negócio e interfaces. Isso reduz significativamente os custos de manutenção e facilita a evolução do sistema.

Destaca-se ainda que as tecnologias e linguagens de programação empregadas na implementação do modelo desempenharam um papel significativo na obtenção da simplicidade de implementação. Neste contexto, algumas ferramentas se destacam: o framework Spring facilitou o processo de inicialização de objetos a partir de nomes de interface, permitindo inclusive a coexistência de duas versões de um mesmo algoritmo, diferenciadas apenas pelo qualificador de versão; a biblioteca Jackson simplificou o processo de conversão entre objetos Java e JavaScript; e o suporte ao tipo JSON no PostgreSQL possibilitou a realização de consultas complexas navegando na estrutura dos objetos.

8 Avaliação Qualitativa

Além das avaliações teórica apresentada na Seção 5, foi também conduzida uma avaliação qualitativa da alternativa de esquema selecionada para representar a parametrização e resultados de algoritmos no sistema desenvolvido. Neste sentido, desenvolvedores de software e pesquisadores da área de Gerenciamento de Dados — que não participaram da elaboração das modelagens conceituais

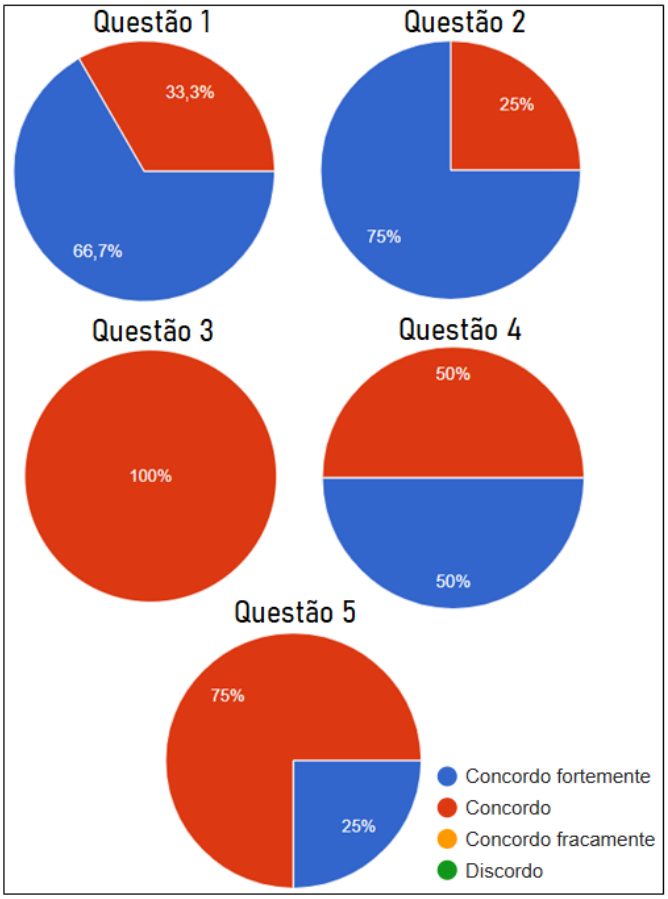

Figure 6: Avaliação qualitativa conduzida com desenvolvedores de software e pesquisadores.

deste trabalho — foram convidados a responder, de forma anônima, às seguintes perguntas:

(1) Você considera a modelagem conceitual proposta para representar parâmetros e resultados de algoritmos como sendo simples de ser entendida e interpretada?
(2) Você considera que a modelagem conceitual proposta para representar parâmetros e resultados de algoritmos facilita a elaboração de consultas sobre os dados?
(3) Você considera que o esquema conceitual proposto para modelar parâmetros e resultados de algoritmos facilita a manutenção do sistema quando ocorrem mudanças nos requisitos relacionados aos algoritmos?
(4) Você considera que a representação dos parâmetros e resultados dos algoritmos no BD utilizando o formato JSON (nível lógico) facilita a realização de consultas?
(5) Você considera que a representação dos parâmetros e resultados dos algoritmos no BD utilizando o formato JSON (nível lógico) facilita a representação dos algoritmos quando ocorrem mudanças nos requisitos relacionados aos algoritmos?

Os resultados obtidos a partir da aplicação do questionário com desenvolvedores de software e pesquisadores são apresentados na Figura 6.

Os resultados do questionário apontam que a maioria dos desenvolvedores e pesquisadores consideram os esquemas conceitual e lógico adotados como sendo simples de serem entendidos e interpretados. Uma porcentagem ainda maior dos participantes concordam fortemente que os esquemas adotados facilitam a implementação de consultas sobre o banco de dados. Em relação às questões 3, que aborda a capacidade de flexibilização e evolução do esquema adotado, todos os participantes concordam que o esquema selecionado facilita a representação do armazenamento de uma variedade de algoritmos, além de facilitar a adaptação do sistema ao ocorrerem mudanças nos requisitos. Por fim, em relação às questões 4 e 5, que tratam especificamente da utilização do formato JSON para representar os parâmetros e resultados dos algoritmos no nível lógico, a maioria dos participantes concorda que a estratégia adotada no nível lógico simplifica tanto a implementação de consultas quanto a evolução do sistema, corroborando com a argumentação apresentada na Seção 6 deste trabalho.

9 Conclusões e Trabalhos Futuros

A avaliação conduzida neste trabalho representa um passo importante no sentido de modelar a coexistência de diferentes versões

de algoritmos que possuem diferentes configurações paramétricas e/ou de resultados produzidos. A avaliação englobou a análise qualitativa de cinco dimensões associadas às alternativas de modelagem investigadas. Com base na metodologia de avaliação adotada, conclui-se que as alternativas C (que modela a execução do algoritmo como entidade) e F (que modela os parâmetros e resultados do algoritmo como atributos multivalorados) apresentam-se como mais promissoras a serem empregadas em soluções práticas. Cabe destacar que os esquemas de banco de dados propostos na Seção 4 não são exaustivos, podendo ser ajustados ou expandidos conforme as necessidades de diferentes contextos. Outros pesquisadores podem explorar novos esquemas e compará-los com os propostos no artigo, priorizando dimensões de qualidade específicas.

Com base na utilização de uma das modelagens propostas no desenvolvimento de um sistema real, foi demonstrada por meio de um estudo de caso a aplicabilidade de um dos esquemas conceituais propostos e discutidos os impactos desta escolha sobre a evolução e interoperabilidade do sistema. Como trabalhos futuros, pretende-se estender a evolução dos esquemas conceituais propostos considerando o impacto dos esquemas sobre aspectos específicos associados com diferentes modelos lógicos de bancos de dados.

Acknowledgments

Agradecemos o apoio e financiamento do Conselho Nacional de Desenvolvimento Científico e Tecnológico – CNPq e do Departamento de Ciência e Tecnologia da Secretaria de Ciência, Tecnologia, Inovação e Complexo da Saúde do Ministério da Saúde - Decit/SECTICS/MS no projeto de pesquisa em execução.

References

[1] Alberto Hernández Chillón, Jesús García Molina, José Ramón Hoyos, and María-José Ortín-Ibáñez. 2023. Propagating Schema Changes to Code: An Approach Based on a Unified Data Model.. In *EDBT/ICDT Workshops*.

[2] Anthony Cleve, Maxime Gobert, Loup Meurice, Jerome Maes, and Jens Weber. 2015. Understanding database schema evolution: A case study. *Science of Computer Programming* 97 (2015), 113–121.

[3] Carlo Curino, Hyun Jin Moon, Alin Deutsch, and Carlo Zaniolo. 2013. Automating the database schema evolution process. *The VLDB Journal* 22 (2013), 73–98.

[4] Carlo A Curino, Hyun Jin Moon, Letizia Tanca, Carlo Zaniolo, et al. 2008. SCHEMA, EVOLUTION IN WIKIPEDIA Toward a Web Information System Benchmark. (2008).

[5] Carlo A Curino, Hyun J Moon, and Carlo Zaniolo. 2008. Graceful database schema evolution: the prism workbench. *Proceedings of the VLDB Endowment* 1, 1 (2008), 761–772.

[6] Julien Delplanque, Anne Etien, Nicolas Anquetil, and Stéphane Ducasse. 2020. Recommendations for evolving relational databases. In *Advanced Information Systems Engineering: 32nd International Conference, CAiSE 2020, Grenoble, France, June 8–12, 2020, Proceedings 32*. Springer, 498–514.

[7] Andrew Gemino and Yair Wand. 2005. Complexity and clarity in conceptual modeling: Comparison of mandatory and optional properties. *Data & Knowledge Engineering* 55, 3 (2005), 301–326.

[8] Marcela Genero, Geert Poels, and Mario Piattini. 2003. Defining and validating metrics for assessing the maintainability of entity-relationship diagrams. *Faculteit Economie en Bedrijfskunde Hoveniersberg, Gent, Working Paper Series* 11, 03 (2003), 199.

[9] Marcela Genero, Geert Poels, and Mario Piattini. 2008. Defining and validating metrics for assessing the understandability of entity–relationship diagrams. *Data & Knowledge Engineering* 64, 3 (2008), 534–557.

[10] Shuhao Li. 2022. *MBench: a benchmark suite designed for database schema migration.* Master's thesis. University of Twente.

[11] Loup Meurice, Csaba Nagy, and Anthony Cleve. 2016. Detecting and preventing program inconsistencies under database schema evolution. In *2016 IEEE International Conference on Software Quality, Reliability and Security (QRS)*. IEEE, 262–273.

[12] Daniel L Moody. 2005. Theoretical and practical issues in evaluating the quality of conceptual models: current state and future directions. *Data & Knowledge Engineering* 55, 3 (2005), 243–276.

[13] Daniel L Moody and Graeme G Shanks. 1994. What makes a good data model? Evaluating the quality of entity relationship models. In *International Conference on Conceptual Modeling*. Springer, 94–111.

[14] Arlei José Calajans MORAES. 2009. *AutonomousDB: uma ferramenta para propagação autônoma de atualizações de esquemas de dados.* Master's thesis. Universidade Federal de Pernambuco.

[15] Pablo Suárez-Otero, Michael J Mior, Maria José Suárez-Cabal, and Javier Tuya. 2020. Maintaining nosql database quality during conceptual model evolution. In *2020 IEEE International Conference on Big Data (Big Data)*. IEEE, 2043–2048.

Received 18 November 2025; revised 11 February 2025; accepted 21 February 2025