Pós-Graduação em Ciência da Computação

INÁCIO DE LOIOLA SOUZA SILVA

Um Modelo Conceitual de Dados e uma Ferramenta CASE para Aplicações de Persistência Poliglota

Universidade Federal de Pernambuco
posgraduacao@cin.ufpe.br
www.cin.ufpe.br/~posgraduacao

Recife
2017

INÁCIO DE LOIOLA SOUZA SILVA

Um Modelo Conceitual de Dados e uma Ferramenta CASE para Aplicações de Persistência Poliglota

Dissertação de Mestrado apresentada ao Programa de Pós-graduação em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco, como parte dos requisitos necessários à obtenção do título de Mestre em Ciência da Computação.

Orientadora: Valéria Cesário Times

Recife
2017

Catalogação na fonte
Bibliotecária Monick Raquel Silvestre da S. Portes, CRB4-1217

S586m Silva, Inácio de Loiola Souza
Um modelo conceitual de dados e uma ferramenta CASE para aplicações de persistência poliglota / Inácio de Loiola Souza Silva. – 2017.
93 f.: il., fig., tab.

Orientadora: Valéria Cesário Times.
Dissertação (Mestrado) – Universidade Federal de Pernambuco. CIn, Ciência da Computação, Recife, 2017.
Inclui referências e apêndices.

1. Banco de dados. 2. Modelagem conceitual. I. Times, Valéria Cesário (orientadora). II. Título.

025.04 CDD (23. ed.) UFPE- MEI 2017-240

Inácio de Loiola Souza Silva

Um Modelo Conceitual de Dados e uma Ferramenta CASE para Aplicações de Persistência Poliglota

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Pernambuco, como requisito parcial para a obtenção do título de Mestre Profissional em 28 de agosto de 2017.

Aprovado em: 28/08/2017.

BANCA EXAMINADORA

Profº. Fernando da Fonseca de Souza

Centro de Informática / UFPE

Profª. Marizete Silva Santos

Universidade Federal Rural de Pernambuco

Profª. Valeria Cesário Times

Centro de Informática / UFPE

(Orientadora)

Dedico este trabalho primeiramente a Jesus Cristo, nosso salvador, que nos mantém livres de todos os males. Em seguida, dedico aos meus pais, Maúricio e Guia, por serem meus maiores incentivadores e por nunca terem medido esforços para me oferecer o melhor, sempre me ensinando a tratar todas as pessoas com educação e a nunca desistir dos meus sonhos.

AGRADECIMENTOS

A Deus, por me manter saudável e em paz espiritual para conseguir concluir esta dissertação.

À minha esposa, Tatiane, pelo companheirismo, pelo amor e por sempre estar ao meu lado, apoiando-me em meus projetos de vida.

À minha família — meus pais, minha irmã Verônica — por todo apoio e toda força que me dão.

Ao apoio do Instituto Federal de Ciência e Tecnologia do Rio Grande do Norte, especificamente ao Campus Apodi, no qual trabalhei por 6 anos e ao qual pretendo retornar, um dia, se Deus assim permitir.

À professora Valéria Times, pela compreensão, paciência, orientação e pelo apoio emocional, que Deus a proteja e zele por todos os seus familiares.

Ao amigo Edson Alves, doutorando no Centro de Informática, pelas dicas e orientação na implementação da ferramenta CASE.

Por fim, a todos os amigos que me incentivaram e que me ajudaram de alguma forma ao longo dessa jornada.

“Um dia aprendi que sonhos existem para tornar-se realidade. E, desde aquele dia, já não durmo para descansar. Simplesmente durmo para sonhar.”

(Walt Disney)

RESUMO

A persistência poliglota refere-se ao uso de diversos SGBD com modelos de dados diferentes em uma mesma aplicação. Uma das motivações para a utilização da persistência poliglota vem da crescente quantidade de dados de variados tipos (estruturados, semiestruturados e não estruturados) que são manipulados em aplicações como: redes sociais, comércio eletrônico, aplicativos móveis. Os SGBD NoSQL (Not Only SQL) representam um conjunto de sistemas de bancos de dados não relacionais e de alto desempenho, projetados para manipular vastos volumes de dados, além de possibilitarem o armazenamento de dados semiestruturados e não estruturados. Diversas pesquisas propõem modelos conceituais para auxiliar o projeto de bancos de dados NoSQL, entretanto, não foi encontrada qualquer proposta na literatura que aborde o projeto conceitual de dados de aplicações de persistência poliglota. Portanto, este trabalho especifica um modelo conceitual de dados, chamado de ERNoSQL, o qual estende o modelo Entidade-Relacionamento (ER) adicionando construtores específicos para possibilitar a modelagem de aplicações de BD com persistência poliglota. Para fornecer uma visão geral do modelo conceitual proposto, apresenta-se um metamodelo especificado em UML que fornece um entendimento sobre como os construtores do ERNoSQL se relacionam. O trabalho também especifica um conjunto de regras de mapeamento do modelo ERNoSQL para as estruturas lógicas dos modelos NoSQL (documentos, grafos, chave-valor e família de colunas). Para auxiliar as atividades de modelagem de esquemas poliglotas, este trabalho propõe, ainda, uma ferramenta CASE para a construção de esquemas conceituais de dados a partir dos construtores de modelagem de ERNoSQL. A ferramenta, intitulada NoSQLCASE, possui um ambiente gráfico para a construção do esquema conceitual e provê funcionalidades de exportação para *scripts* expressos em linguagens de SGBD NoSQL. Finalmente, um estudo de caso foi realizado para comparação entre esquemas conceituais de dados construídos por duas ferramentas CASE existentes e baseadas no modelo ER, e esquemas conceituais de dados projetados pela ferramenta NoSQLCASE proposta. Além disso, as funcionalidades de exportação de NoSQLCASE foram ilustradas pela implementação de *scripts* gerados pela ferramenta proposta no SGBD MongoDB.

Palavras-chave: NoSQL. Projeto de banco de dados. Modelagem conceitual.

ABSTRACT

The polyglot persistence refers to the use of several DBMS with different data models in the same application. One of the motivations for the use of polyglot persistence comes from the growing amount of data of any types (structured, semi-structured and unstructured) that are handled in applications such as: social networking, e-commerce, mobile applications. NoSQL DBMS represent a set of non-relational database systems of high performance designed to handle vast volumes of data, besides allowing the storage of semi-structured and unstructured data. Several researches propose conceptual models to support the design of NoSQL databases. However, no proposal that addresses the conceptual data design of polyglot persistence applications was found in the literature. Therefore, this work specifies a conceptual data model, called ERNoSQL, that extends the Entity-Relationship model by adding specific constructors to enable the modeling of DB applications with polyglot persistence. To provide an overview of the proposed conceptual model, a meta-model that provides an understanding of how ERNoSQL constructors are related was specified in UML. The work also specifies a set of mapping rules of the ERNoSQL model for the logical structures of the NoSQL models (documents, graphs, key-value and family of columns). To support the activities of modeling polyglot schemes, this paper also proposes a CASE tool for the construction of conceptual data schemas from the ERNoSQL modeling constructors. The tool, called NoSQLCASE, has a graphical environment for constructing the conceptual schema that provides export functionality for scripts expressed in NoSQL DBMS languages. Finally, a case study was conducted to compare conceptual data schemas constructed by two existing CASE tools that are based on the ER model and conceptual schemas of data designed by the proposed NoSQLCASE tool. In addition, the export functionality of NoSQLCASE was illustrated by the implementation of scripts generated by the proposed tool in the MongoDB DBMS.

Keywords: NoSQL. Database design. Conceptual modeling.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE ABREVIATURAS E SIGLAS

BD	Banco de Dados
EGL	Epsilon Generation Language
E-R	Entidade e Relacionamento
GUI	Graphical User Interface
JSON	JavaScript Object Notation
MER	Modelo Entidade-Relacionamento
NoSQL	Not only SQL
OMG	Object Management Group (do português, Grupo de Gerenciamento de Objetos)
SGBD	Sistema de Gerenciamento de Banco de Dados
SQL	Structured Query Language
UML	Unified Modeling Language
XMI	XML Metadata Interchange

SUMÁRIO

1 INTRODUÇÃO

Este capítulo evidencia os principais conceitos que envolvem a utilização de persistência poliglota e sua aplicação no projeto de bancos de dados. Ele também discute a motivação para o desenvolvimento da pesquisa e relata os objetivos gerais e específicos do trabalho, além de descrever a organização desta dissertação.

1.1 Contextualização

A modelagem conceitual é a primeira fase de um projeto de banco de dados, na qual se abstrai de qualquer tecnologia e que se interessa pelo levantamento de informações sobre o domínio da aplicação projetada. Nesta fase, consolidou-se o uso do Modelo E-R, por sua simplicidade e capacidade de representar conceitos de alto nível (ELMASRI; NAVATHE, 2010).

As aplicações de *software* estão aumentando a cada dia, tanto no nível de complexidade dos algoritmos, quanto na quantidade de dados a serem manipulados. Além disso, é exigido um nível de escalabilidade maior, tendo em vista requisitos básicos, como disponibilidade e confiabilidade dos dados. Nesse contexto, surgiram as tecnologias de BD NoSQL, que proporcionam um ambiente propício para manipular grandes volumes de dados, além de possivelmente promoverem o armazenamento de dados em *clusters* que traz benefícios, como, por exemplo, a tolerância a falhas na rede de comunicação que permite uma maior disponibilidade de acesso aos dados.

Os sistemas de BD NoSQL surgiram principalmente por iniciativas isoladas de grandes empresas da área de tecnologia, como Google[1] e Amazon[2] , que possuíam problemas na utilização de BD convencionais em suas aplicações e não existiam opções no mercado para suprir suas necessidades. No ano 2000, ambas publicaram artigos sobre suas ferramentas, BigTable[3] (Google) e Dynamo[4] (Amazon) (SADALAGE; FOWLER, 2013). Essas publicações incentivaram outras empresas e desenvolvedores de *software* a explorarem a criação de ferramentas semelhantes para solucionar dificuldades específicas e existentes nos sistemas de BD relacionais, dentre as quais, destaca-se a necessidade de esquemas bem definidos impossibilitando a manipulação de dados não estruturados e semiestruturados.

[1] https://www.google.com/intl/pt-BR/about/
[2] https://aws.amazon.com/pt/?nc2=h_lg
[3] https://cloud.google.com/bigtable/?hl=pt
[4] https://aws.amazon.com/pt/dynamodb/

1.2 Motivação

A utilização de BD NoSQL vem aumentando consideravelmente junto com a evolução das aplicações de *software*, principalmente nos sistemas construídos para Internet, os quais devem possuir flexibilidade para escalar de acordo com a demanda de uso. Isso cria um desafio maior para os projetistas de *software*, pois, além das escolhas de linguagens e *frameworks* de programação, eles devem analisar a persistência dos dados, selecionando SGBD que mais se adequem aos tipos de dados e aos requisitos não funcionais — escalabilidade, disponibilidade, de cada projeto.

Persistência poliglota refere-se ao uso de diversos SGBD com modelos de dados diferentes em uma mesma aplicação (SADALAGE; FOWLER, 2013). Uma aplicação de persistência poliglota é projetada para se beneficiar das características de cada SGDB de acordo com os requisitos levantados para o projeto de construção do *software*. Por exemplo, no projeto de uma aplicação web que necessita manipular tipos de dados estruturados, não estruturados e semiestruturados, pode-se utilizar SGBD relacionais para os dados estruturados e SGBD NoSQL para os outros tipos de dados, por exemplo, dados temporários que podem ser armazenados em BD NoSQL orientado a chave-valor.

Os benefícios do uso da persistência poliglota se tornam mais evidentes nos projetos de sistemas para Internet que são elaborados com foco em atender maiores fluxos de manipulação de dados, manter tipos de dados complexos e possibilitar um menor esforço na alteração do esquema do banco de dados.

Diversos trabalhos de pesquisa propõem modelos para a construção de esquemas conceituais de BD NoSQL, como o GOOSSDM (BANERJEE et al., 2015) e NoSQL Abstract Model (BUGIOTTI et al., 2014). Existe também a extensão do E-R para modelagem de tipos de dados Big Data (VILLARI et al., 2016). No entanto, pouca atenção tem sido dedicada à seguinte questão: *quais construtores de modelagem são necessários para criar um esquema conceitual que represente aplicações de persistência poliglota?* A solução consiste em criar construtores de modelagem conceitual para representar um único esquema de dados que represente conceitos de diferentes sistemas de BD NoSQL, além de definir a semântica e as notações diagramáticas destes construtores.

A dificuldade em criar um esquema conceitual de dados para um projeto de aplicação de persistência poliglota foi a principal motivação para o desenvolvimento desta pesquisa. Tendo em vista a importância da fase de modelagem conceitual de dados, é primordial a existência de um modelo que possibilite a construção de esquemas conceituais de persistência poliglota, auxilie os projetistas de banco de dados e facilite a comunicação entre a equipe técnica e usuários requisitantes do

projeto.

1.3 Objetivos

Esta seção discorre sobre os objetivos gerais (Seção 1.2.1) e específicos (Seção 1.2.2) do trabalho detalhado neste documento.

1.3.1 Objetivos Gerais

Esta dissertação propõe um modelo conceitual de dados, intitulado ERNoSQL, para auxiliar na modelagem de aplicações que utilizem persistência poliglota. O ER-NoSQL estende o modelo E-R adicionando construtores específicos para possibilitar a modelagem de esquemas conceituais poliglotas. Para fornecer uma visão geral do modelo conceitual proposto, este trabalho apresenta um metamodelo especificado em UML para representar, por meio de um conjunto de classes, os conceitos utilizados na elaboração do ERNoSQL. O metamodelo fornece um entendimento sobre como os construtores do ERNoSQL se relacionam entre si.

Além da descrição dos construtores do ERNoSQL com suas respectivas notações gráficas, o trabalho objetiva, ainda, a elaboração de um conjunto de regras de mapeamento do modelo ERNoSQL para as estruturas dos SGBD NoSQL baseados em documentos, grafos, chave-valor e família de colunas.

Com o objetivo de auxiliar as atividades de modelagem de esquemas poliglotas, este trabalho propõe, também, uma ferramenta CASE para a construção de esquemas a partir dos construtores de ERNoSQL. A ferramenta, intitulada NoSQLCASE, possui um ambiente gráfico para a construção do esquema conceitual e para provimento das funcionalidades de exportação para *scripts* expressos em linguagens específicas de sistemas de BD NoSQL.

1.3.2 Objetivos Específicos

Descreve-se abaixo as atividades que compõem os objetivos específicos para alcançar as contribuições deste trabalho:

I. Especificar os construtores de modelagem com suas respectivas propriedades e notações gráficas;

II. Especificar as regras de mapeamento para esquemas lógicos de BD NoSQL a partir do esquema conceitual ERNoSQL;

III. Especificar um metamodelo em UML que representa os construtores de ERNoSQL e que servirá como base para a implementação da ferramenta CASE;

IV. Desenvolver uma ferramenta CASE baseada na notação gráfica dos construtores de ERNoSQL que possibilita a criação de esquemas conceituais baseados no modelo ERNoSQL; e

V. Exemplificar a aplicação da ferramenta CASE por meio da construção de esquemas conceituais de dados, com base em um cenário real de uma organização acadêmica. Realizar a modelagem deste cenário em NoSQLCASE e em outras ferramentas para demonstrar os diferenciais da ferramenta proposta.

1.4 Estrutura da Dissertação

Os capítulos restantes desta dissertação encontram-se estruturados da seguinte forma:

Capítulo 2: Fundamentação Teórica

Esse capítulo tem como objetivo descrever a fundamentação teórica e os conceitos básicos utilizados para a elaboração deste trabalho e exibir um relato sobre estudos de modelagem conceitual e lógica de BD NoSQL.

Capítulo 3: Modelo ERNoSQL

Esse capítulo detalha o modelo conceitual proposto, definindo seus construtores, suas notações gráficas e um exemplo de uso, além de representar, por meio de UML[5] , o metamodelo que retrata conceitos e relacionamentos entre conceitos de ERNoSQL. Por fim, este capítulo define os grupos de regras de mapeamento criados a partir das equivalências identificadas entre o ERNoSQL e os modelos de dados lógicos de SGBD NoSQL existentes. Essas regras são utilizadas para a geração do esquema lógico de dados em linguagens específicas de definição e manipulação de dados para implementação de bases de dados NoSQL.

Capítulo 4: A Ferramenta NoSQLCASE

Esse capítulo mostra a arquitetura de software da ferramenta NoSQLCASE e apresenta sua GUI (Graphical User Interface), juntamente com suas principais funcionalidades e os recursos disponíveis para a criação de esquemas conceituais de dados de aplicações de persistência poliglota. Além disso, este capítulo discorre sobre o processo de criação de esquemas conceituais por meio da ferramenta NoSQLCASE, especificado em um diagrama de atividades da UML que define os passos para criar um esquema conceitual na ferramenta.

Capítulo 5: Um Estudo de Caso com NoSQLCASE

Esse capítulo descreve um estudo de caso comparando a ferramenta NoSQLCASE e outras ferramentas existentes, por meio da criação de um esquema conceitual

[5] http://www.uml.org/

para um domínio de aplicação e da demonstração de uso das funcionalidades de NoSQLCASE. O esquema conceitual construído para o estudo de caso modela um cenário real, cujos requisitos de dados fazem parte do cotidiano de uma instituição de ensino do Estado do Ceará.

Capítulo 6: Conclusão

Esse capítulo discute as principais contribuições desta dissertação e sugere trabalhos futuros que possivelmente contribuirão para o avanço do estudo realizado. Além disso, descreve-se as limitações do trabalho.

2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo contém a fundamentação teórica deste trabalho. Ele exibe conceitos básicos de sistemas NoSQL e discorre sobre seus principais modelos de dados lógicos (Seção 2.1). Finalmente, este capítulo exibe a descrição de trabalhos correlatos (Seção 2.2), incluindo um relato dos principais estudos realizados pela comunidade científica sobre projeto conceitual e lógico de BD NoSQL.

2.1 Conceitos Básicos

Esta seção contém os principais conceitos associados ao desenvolvimento deste trabalho. A Seção 2.1.1 descreve os conceitos básicos e as características de SGBD NoSQL, além de definir os modelos de dados lógicos utilizados nestes tipos de sistemas. A Seção 2.1.2 discorre sobre os principais modelos utilizados na construção de esquemas de dados conceituais, enquanto a Seção 2.1.3 descreve o conceito de persistência poliglota. Por fim, na seção 2.1.4, é descrito o *framework* de modelagem Eclipse[1] e seus componentes que são utilizados neste trabalho para o desenvolvimento da ferramenta NoSQLCASE proposta, a qual é exposta no Capítulo 4.

2.1.1 Sistemas de BD NoSQL

O movimento intitulado “NoSQL”, que significa “Não apenas SQL” (do inglês “Not Only SQL”), surgiu devido à insatisfação de muitas empresas de tecnologia da informação com os SGBD relacionais disponíveis no mercado. Grande parte desta insatisfação deveu-se às limitações dos SGBD relacionais para lidar com dados não estruturados e à baixa eficiência para processamento em *clusters*, além de exigir esquemas de dados rígidos que obrigam uma definição dos tipos e formatos dos dados para possibilitar o armazenamento. Em vista disso, essas empresas projetaram sistemas de bancos de dados que eliminassem as limitações existentes nos SGBD relacionais, focalizando na criação de soluções que permita a escalabilidade horizontal e a possibilidade de manipular grandes conjuntos de dados, incluindo dados semiestruturados e não estruturados.

SGBD NoSQL utilizam modelos de dados diferentes do modelo relacional, tais como: Baseado em Documentos, Baseado em Grafos, Chave-Valor e Famílias de Colunas. Cada um desses modelos utiliza estruturas de dados diferentes para armazenar e manipular os dados. A decisão sobre qual BD NoSQL utilizar deve ser analisada de acordo com o domínio envolvido e as necessidades do projeto. Metodologias e

[1] https://eclipse.org/epsilon/

ferramentas de apoio ao projeto lógico de BD NoSQL são temas pouco explorados na literatura de banco de dados (LIMA; MELLO, 2015).

Os SGBD NoSQL ainda não são conhecidos por uma parcela dos profissionais de tecnologia da informação, mas a popularidade destes SGBD vem crescendo no mundo. O site db-engines[2] mantém uma classificação da popularidade dos SGBD disponíveis e verifica-se o aumento na adoção dos tipos de SGBD NoSQL, de forma que, o MongoDB[3] (Documentos) encontra-se na quinta colocação da lista e os outros representantes do grupo NoSQL vêm mantendo de forma crescente o nível de popularidade, como por exemplo: o Cassandra[4] (Famílias de Colunas) na oitava colocação, o Redis[5] (Chave-valor) em décimo e o Neo4j[6] (Grafos) em vigésimo primeiro.

A literatura sobre NoSQL, em sua maioria, não propõe a substituição total dos SGBD relacionais, mas demonstra que, para determinados tipos de dados, operações (escrita, leitura) e regras de negócio de aplicação, um SGBD NoSQL terá desempenho mais eficaz (JEON; AN; LEE, 2015). Portanto, os sistemas NoSQL aumentam as possibilidades de escolha sobre como manter os dados de uma aplicação de BD. Em alguns casos, pode-se utilizar variados tipos de SGBD para uma mesma aplicação. Isso acontece devido ao volume e variedade de tipos de dados que podem ser gerenciados. Nessas situações, utiliza-se o termo *Persistência Poliglota,* que evidencia essa pluralidade de uso de modelos de dados lógicos em uma única aplicação.

O termo Big Data, comumente usado na academia e indústria, refere-se a um vasto volume de dados de um determinado domínio de aplicação. Um exemplo de Big Data são os dados gerados e manipulados a todo instante pelos usuários de ferramentas de mídias sociais. As características principais de Big Data são: o volume, que indica a quantidade de dados a ser manipulado; a variedade, que implica na possibilidade de representação de diversos tipos de dados (estruturados, semiestruturados e não-estruturados); a velocidade, que se refere à rapidez na qual os dados são gerados pelas aplicações atuais; e, por último, veracidade — característica indicativa de que nem todos os dados são precisos e relevantes. Todos esses aspectos criam desafios quanto à captura, ao armazenamento, à manipulação e à visualização desses dados (BANERJEE et al., 2015).

SGBD NoSQL permitem a persistência de dados no formato JSON. Por exemplo, o SGBD Cassandra possui funções para armazenar documentos JSON[7] em suas tabelas através da linguagem de consulta CQL (Cassandra Query Language). Outro

[2] https://db-engines.com/en/ranking
[3] https://www.mongodb.com/
[4] http://cassandra.apache.org/
[5] https://redis.io/
[6] https://neo4j.com/
[7] http://www.json.org/

exemplo é o SGBD Redis, que também disponibiliza a funcionalidade de armazenar diretamente um documento JSON através da sua interface de programação de aplicativos, a qual é utilizada para comunicação dos aplicativos com o SGBD.

O Quadro 1 mostra o exemplo dos códigos utilizados nos SGBDs NoSQL Cassandra (Modelo de dados baseado em família de colunas) e Redis (Modelo de dados baseado em chave-valor) para armazenar um documento no formato JSON.

Quadro 1 – Quadro com exemplos para armazenar documentos JSON nos SGBDs NoSQL Cassandra e Redis

SGBD NoSQL	Código de exemplo
Cassandra	INSERT INTO nomebd.nometabela JSON '{ "atributo" : "valor", "atributo" : valor, "atributo" : "valor" }';
Redis	JSON.SET object . '{"atributo": "valor", "atributo": "valor"}'

Fonte: Elaborado pelo Autor (2017)

2.1.1.1 SGBD NoSQL Baseado em Documento

Os sistemas de bancos de dados baseados em documentos compõem a categoria de SGBD NoSQL que utilizam como estrutura de armazenamento uma coleção de documentos que podem ser XMLhttp://www.xml.org/[8] , JSON[9], BSON[10], entre outros. Esses documentos são estruturados na forma de árvores hierárquicas e autodescritivas (SADALAGE; FOWLER, 2013). Cada documento contém propriedades especificadas no formato "nome : valor". O valor dessas propriedades pode ser um dado escalar, um mapa, uma coleção, entre outros. Um documento representa um registro armazenado, sendo equivalente a uma linha em um SGBDR, com a diferença de não possuir esquema fixo definido; ou seja, cada documento pode conter propriedades diferentes. Dessa forma, se uma propriedade não se aplica a determinado documento, ela sequer será identificada, evitando-se, assim, a criação de atributos com valores vazios.

Os documentos pertencem a estruturas nomeadas de *coleções*, as quais, geralmente, agrupam documentos referentes a um mesmo assunto. Por exemplo, pode-se criar uma coleção chamada *Escolas*, na qual se armazenem documentos com atri-

[8] http://www.xml.org/
[9] http://www.json.org/
[10] http://bsonspec.org/

butos deste domínio. Cada documento possui um atributo chave único, utilizado para identificá-lo na coleção correspondente (KAUR; RANI, 2013).

Esse tipo de banco de dados é otimizado para a manipulação de informações textuais (SRIVASTAVA; GOYAL; KUMAR, 2015), sendo um bom candidato para utilização em sistemas de gestão de conteúdo, plataformas de *blog*, aplicativos de comércio eletrônico, entre outros (SADALAGE; FOWLER, 2013). Domínios de aplicações que necessitam de flexibilidade de esquema e com requisitos que passem por atualizações constantes são adequados ao uso de SGBD baseados em documentos. Nos casos de transações complexas com necessidade de execução de operações atômicas ou de consultas complexas envolvendo várias coleções, contudo, não é indicado o uso desse tipo de SGBD NoSQL. Isto ocorre porque os SGBD de documentos não foram projetados para processar muitos relacionamentos entre os dados, pois umas das principais vantagens é utilizar documentos que contenham todas as informações relativas a uma determinada consulta a base de dados.

Existem duas principais abordagens para inserir os relacionamentos entre os dados de BD de documentos: a primeira é seguir a utilização do conceito de "agregados" (SADALAGE; FOWLER, 2013), no qual a intenção é montar documentos com os objetos relacionados e aninhados, facilitando-se, assim, o acesso e a distribuição dos dados para utilização em *clusters*; a segunda forma é utilizar as chaves identificadoras dos documentos como ligação, relacionando-os. Essa última abordagem deve ser analisada com cuidado pelo projetista, pois, quanto maior o número de ligações entre os documentos, maior o custo de processamento e menor a eficiência do SGBD.

Entre os tipos baseados em documentos, o SGBD que possui maior popularidade é o MongoDB (KAUR; RANI, 2013). Na documentação do MongoDB[11] existe uma seção voltada para a modelagem de banco de dados, mas não orienta o uso de um modelo conceitual — apenas ilustra as estratégias existentes para relacionar documentos de forma aninhada ou por meio de identificadores.

A persistência de dados baseada em documentos está se popularizando cada vez mais, fato comprovado pelas iniciativas de SGBDR renomados, como PostgreSQL[12] e MySQL[13] que adicionaram suporte ao armazenamento de documentos em suas ferramentas.

2.1.1.2 SGBD NoSQL Baseado em Grafos

SGBD NoSQL baseados em grafos são constituídos basicamente por duas estruturas de dados: nós e arestas. Os nós possuem propriedades que armazenam

[11] https://docs.mongodb.com
[12] https://www.postgresql.org/docs/9.5/static/datatype-json.html
[13] https://dev.mysql.com/doc/refman/5.7/en/document-store.html

dados de objetos, enquanto as arestas, por sua vez, representam os relacionamentos que conectam os nós, podendo ser unidirecionais ou bidirecionais, de acordo com seus significados. As arestas (relacionamentos) também podem possuir propriedades específicas. Por exemplo, em um relacionamento *Amigos* entre dois nós, pode-se adicionar como propriedade a data de início da amizade. Esses SGBD não necessitam de esquema definido, o que possibilita sua utilização em domínios que possuem mudanças frequentes dos requisitos de dados armazenados.

Comparando-se o modelo de dados lógico baseado em grafos com o modelo conceitual ER, os nós representam as entidades, as propriedades representam os atributos de entidade e os relacionamentos são representados pelas relações entre nós (KAUR; RANI, 2013).

No modelo relacional, para consultar e interligar dados de estruturas distintas utilizam-se junções — operações que requerem bastante processamento. Nos SGBD baseados em grafo, as consultas são executadas com o uso de algoritmos de busca em grafos, as quais realizam a travessia no grafo e selecionam as informações desejadas sem a necessidade de qualquer junção, pois os dados já estão naturalmente interligados no grafo. Dessa forma, o desempenho para processar consultas entre dados relacionados é superior ao processamento feito sobre outros modelos de persistência de dados (relacional, documentos, família de colunas) Singh e Kaur (2015), Mathew e Kumar (2015). Outro aspecto importante desses tipos de SGBD é o fato de permiterem transações ACID, o que garante dados íntegros, disponíveis e confiáveis.

O modelo de grafos é indicado para utilização em domínios de aplicações que foquem nas relações entre os dados, como redes sociais, roteamento, mecanismos de recomendação e serviços baseados em localização (SADALAGE; FOWLER, 2013).

2.1.1.3 SGBD NoSQL Chave-Valor

SGBD NoSQL baseados em chave-valor têm um modelo de dados lógico muito simples. Os dados são organizados como uma matriz associativa de entradas constituídas por chave e valor (KAUR; RANI, 2013). Cada chave é única e é utilizada nas consultas para recuperar o valor associado. As chaves podem ser geradas pelo SGBD ou definidas pelo usuário, e o valor armazenado pode ser um *blob*, texto, JSON, XML, entre outros.

Nesses SGBD, as consultas somente são realizadas por meio da chave que retorna todos os dados armazenados no registro. Não é possível realizar consultas diretamente ao conteúdo armazenado, tampouco utilizar junções entre valores ou qualquer análise sobre os dados (por exemplo, somar valores, calcular médias). Essas restrições tornam os BD chave-valor muito rápidos e eficientes nas operações de leitura e escrita.

Os relacionamentos entre as estruturas de dados não são definidos por esses tipos de SGBD, tornando-os apropriados somente para alguns tipos de domínios de aplicações. Uma iniciativa para tentar relacionar os dados é adicionar identificadores na composição da chave. Por exemplo, para uma entidade *Cliente* que se relaciona com uma entidade *Pedido*, pode-se ter a chave IdCliente-IdPedido. Essas alternativas são adaptações para a representação de relacionamentos nos SGBD chave-valor, já que, eles não foram projetados para possibilitar o uso de relacionamentos. Nesses casos, todo processamento que envolva algum relacionamento mantém-se sob a responsabilidade do aplicativo.

Existe uma vantagem desses SGBD: eles não definem limites para o tamanho das estruturas armazenadas. Assim, uma boa opção para relacionar dados é a utilização de dados agregados. Um *agregado* contém dados relacionados que formam uma unidade de acesso e de distribuição, tornando-se possível criar uma chave que represente um determinado agregado que contém outros dados relacionados.

Vários SGBD chave-valor estão disponíveis para uso, e os mais populares são o Redis, o Riak[14] , o Memchached[15] e o Voldemort[16] (KAUR; RANI, 2013). Alguns deles permitem a adição de novas funcionalidades como a possibilidade de consultas ao conteúdo armazenado, além da tradicional consulta por chaves, equiparando-se a SGBD baseados em documentos.

Existem vários casos de sucesso no uso destes SGBD para determinados domínios de aplicações. Eles são apropriados, por exemplo, para armazenar informações de sessão, listas de compras, projetos de software embarcados, chat e serviços de troca de mensagens (SADALAGE; FOWLER, 2013). A aplicação de microblog pesssoal, conhecida por Twitter[17] , utiliza um BD chave-valor para armazenar os dados das postagens (tweet). Estas mensagens são rapidamente identificadas através de uma única chave para cada postagem cadastrada (SRIVASTAVA; GOYAL; KUMAR, 2015).

2.1.1.4 SGBD NoSQL Basedo em Famílias de Colunas

SGBD NoSQL baseados em famílias de colunas são projetados para lidar com três problemas: grande número de colunas, natureza esparsa de dados e mudanças frequentes no esquema (KAUR; RANI, 2013). Dessa forma, estes sistemas permitem que se tenha flexibilidade no armazenamento de dados, pois cada linha tem a quantidade de colunas necessária para cada registro. Por exemplo, para armazenar dados de professores, pode-se persistir o atributo “instituicao_doutorado” somente para

[14] http://basho.com/products/riak-kv/
[15] https://memcached.org/
[16] http://www.project-voldemort.com/voldemort/
[17] https://twitter.com/

os professores que possuam doutorado, enquanto para os demais professores este atributo não seria criado.

SGBD baseados em famílias de colunas armazenam a coluna inteira de uma tabela em um conjunto e a mapeia para uma única chave. Toda entrada na coluna possui índice, o que possibilita a execução de pesquisas somente sobre partes específicas. Uma coluna também pode conter colunas aninhadas, tornando-se uma família de super-colunas, oferecendo acesso rápido a um agregado de dados relacionados (MATHEW; KUMAR, 2015).

A maior parte das bases de dados colunares são compatíveis com o MapReduce, que acelera o processamento de grandes quantidades de dados, fragmentando a carga de processamento de uma determinada análise de dados em servidores distribuídos. Os SGBD de famílias de colunas mais populares são o Hypertable[18] , o HBase[19] e o Cassandra. O Hypertable e o HBase são derivados do BigTable[20]; já o Cassandra tem características do BigTable e do Dynamo (KAUR; RANI, 2013).

Grandes empresas que mantêm sistemas com abrangência mundial utilizam SGBD colunares. O principal exemplo destas empresas é o Facebook[21] , que utiliza o Cassandra como solução de persistência em sua rede social. Outro exemplo é a Uber[22], empresa que conecta motoristas particulares a clientes por meio de um aplicativo para dispositivos móveis, e que utiliza o Cassandra como sistema de banco de dados para o gerenciamento de seus serviços.

2.1.1.5 SGBD NoSQL Multimodelo

Os SGBD NoSQL multimodelo foram criados com intuito de consolidar em uma única solução mais de um modelo de dados de SGBD NoSQL (documentos, grafos, chave-valor e famílias de colunas). SGBD multimodelo permitem que os dados fiquem consolidados num único produto, oferecendo uma linguagem de consulta única para todos os modelos de dados (OLIVEIRA; CURA, 2016).

Os SGBD NoSQL multimodelo mais conhecidos são o OrientDB[23] e o ArangoDB[24]. Esses dois dão suporte aos modelos de dados orientados a documentos, grafos e chave-valor. Para manipulação dos dados nesses SGBD, o ArangoDB fornece a linguagem de consulta AQL[25] que difere do SQL mas possui uma sintaxe simples e de

[18] http://www.hypertable.com/
[19] https://hbase.apache.org/
[20] https://cloud.google.com/bigtable/
[21] https://www.facebook.com/
[22] https://eng.uber.com/mysql-migration/
[23] http://orientdb.com/orientdb/
[24] https://www.arangodb.com/
[25] https://docs.arangodb.com/3.1/AQL/

fácil compreensão. Já o OrientDB utiliza a linguagem SQL[26] com algumas adaptações para possibilitar consultas em grafos, mas mantendo a sintaxe do SQL para facilitar a adaptação dos desenvolvedores.

2.1.2 Modelagem Conceitual

A modelagem conceitual é uma etapa fundamental do projeto de banco de dados. O artefato gerado pela modelagem conceitual é um esquema conceitual que descreve os requisitos de dados de um domínio de aplicação. Esse esquema é composto por tipos de informações, atributos, relacionamentos e restrições, e pode ser representado por meio de um diagrama gráfico, que facilita o entendimento sobre o domínio do sistema e a comunicação da equipe técnica com os usuários finais (ELMASRI; NAVATHE, 2010).

O modelo Entidade-Relacionamento (MER) é amplamente utilizado na academia e na indústria para realizar a etapa de modelagem conceitual de banco de dados. O MER possui construtores gráficos para a representação de esquemas conceituais por meio de um diagrama Entidade-Relacionamento. Os principais construtores desse modelo representam tipos de entidades, atributos e os relacionamentos entre as entidades. Existem, na literatura, diversos trabalhos publicados que utilizam os conceitos do MER e incluem construtores para modelagem de dados com restrições específicas, como, por exemplo, dados de aplicações de organizações de saúde (ARAÚJO, 2012).

A Unified Modeling Language (UML) é uma linguagem para especificação de projetos de *software* que inclui um conjunto de diagramas, divididos em diagramas *estruturais* e *comportamentais. Os diagramas comportamentais* especificam os aspectos dinâmicos, como requisitos funcionais e fluxos de atividades, enquanto os diagramas estruturais fazem referência à estrutura do sistema, ou seja, à arquitetura da aplicação, representando, assim, as características estáticas de um *software*. Dentre esses diagramas, há o diagrama de classes, utilizado para definir estruturas estáticas de dados e que permite também a definição dos métodos que irão manipular os dados. A representação abstrata do diagrama de classes sem a definição das operações é utilizada para a modelagem conceitual de dados, tornando-se uma opção para representar conceitualmente os dados de uma aplicação.

2.1.3 Persistência Poliglota

O termo *Persistência Poliglota* faz referência à utilização de diferentes SGBD em um único domínio de aplicação. O uso de mais de um SGBD é influenciado pelos tipos de dados (estruturados, semiestruturados e não estruturados) e os requisitos de negócio do projeto (SADALAGE; FOWLER, 2013). Essa necessidade é bem evidenciada em aplicações web que geralmente manipulam uma maior quantidade de dados e que

[26] http://orientdb.com/docs/last/SQL.html

necessitam de estruturas que permitam escalar (expandir os recursos de *hardware* e a distribuição de máquinas em localizações geográficas diferentes) de acordo com o aumento da demanda. Além disso, as necessidades de manipulação de dados também devem ser avaliadas. Por exemplo, uma aplicação de comércio eletrônico que necessita utilizar dados temporários referentes à sessão do usuário e ao carrinho de compras pode se beneficiar de um SGBD NoSQL do tipo chave-valor que armazena dados em memória primária (RAM), permitindo uma maior eficiência de leitura/escrita. Nas páginas de visualização de produtos que envolvem atributos e dados diferentes e específicos de cada produto, por outro lado, pode ser utilizado um banco de dados orientado a documentos, o qual permite o armazenamento de dados semiestruturados em documentos (XML, JSON). Neste caso, cada documento é equivalente a um registro armazenado no SGBD e cada registro mantém os atributos necessários de cada documento.

Com a popularização dos SGBD NoSQL, os projetistas de banco de dados começaram a usar modelos de dados diversos em uma única aplicação. Dessa forma, eles se beneficiam das características de cada SGBD NoSQL para melhor representação de tipos de dados e necessidades do projeto.

O uso da persistência poliglota acrescenta complexidade ao projeto, pois inclui mais ferramentas para configurar e gerenciar. Isso pode demandar a contratação de uma equipe maior ou de profissionais mais qualificados. Mesmo com o aumento da complexidade, a persistência poliglota se torna viável em aplicações com maior abragência, pois inclui beneficíos na operação, gerenciamento e evolução do banco do dados em ambiente de produção devido à capacidade dos SGBD NoSQL para manipulação de dados volumosos.

2.1.4 Framework de Modelagem Eclipse

Esta seção descreve as principais tecnologias e *framework* utilizados para o desenvolvimento da ferramenta CASE que compõe uma das contribuições deste trabalho. Todas as tecnologias descritas nas próximas seções estão incluídas no pacote Eclipse Epsilon[27] que agrupa linguagens e ferramentas para projetar software baseados em modelo (do inglês, Model Driven Design). Esta arquitetura utiliza um modelo como artefato principal para a elaboração de software.

2.1.4.1 Eclipse Modeling Framework (EMF)

O EMF[28] é um *framework* que permite a modelagem e a geração de códigos para a construção de ferramentas e de outras aplicações baseadas em um modelo de

[27] https://eclipse.org/epsilon/
[28] http://www.eclipse.org/modeling/emf/

dados. A partir de uma especificação do modelo, o framework EMF fornece ferramentas que produzem, em tempo de execução, classes Java[29] e adaptadores que permitem a edição visual de instâncias do modelo por meio de um editor gráfico.

O *framework* inclui um metamodelo intitulado ECore[30] , utilizado para descrever modelos, que serve de base para a geração do código que implementa a ferramenta de suporte aos modelos representados. O EMF provê persistência com padrão XMI, o qual possibilita a troca de informações baseada em arquivos XML.

Os principais componentes do Ecore são: 1) *EClass*: utilizada para representar uma metaclasse; 2) *EAttribute*: representa o atributo de uma EClass; e 3) *EReference*: usado para definir as associações entre os componentes do tipo *EClass*. Além desses componentes, existem outros que podem ser utilizados para detalhar as especificidades dos modelos. O *framework* possui um editor para criação do modelo por meio de diagrama UML, o que facilita a visualização e a construção do modelo em EMF Ecore (SOUZA, 2011).

O EMF permite três níveis de geração de código: 1) *Modelo*: fornece interfaces Java e classes de implementação para todas as classes no modelo; 2) *Adaptadores*: geram classes de implementação que se adaptam às classes de modelo para edição e exibição; e 3) *Editor*: produz um editor gráfico básico que pode ser personalizado.

O EMF possui a linguagem Emfatic[31] , que é uma representação textual do metamodelo ECore. Emfatic usa uma sintaxe compacta e legível, semelhante ao Java. Arquivos descritos em Emfatic possuem a extensão (.emf).

2.1.4.2 Epsilon Generation Language (EGL)

EGL[32] é uma linguagem para a transformação de modelo para texto (Model to text - M2T) e pode ser usada para converter modelos em vários tipos de artefatos textuais, como, por exemplo, código executável Java, relatórios, código HTML[33] e *scripts* de linguagens específicas geradas a partir de modelos. EGL possui um gerador de código baseado em modelos e fornece vários recursos para simplificar e permitir a automatização da representação textual das instâncias do modelo.

Os principais recursos do EGL são 1) um mecanismo de fusão que une as seções estáticas (escritas) às dinâmicas (geradas pelo EGL), criando uma saída em texto; 2) um sistema de template extensível que possibilita a geração de texto para diversas fontes, como *scripts* de linguagens de banco de dados, arquivos javascript[34] , html,

[29] http://www.oracle.com/technetwork/java/javase/downloads/index.html
[30] https://wiki.eclipse.org/Ecore
[31] http://www.eclipse.org/emfatic/
[32] http://www.eclipse.org/epsilon/doc/egl/
[33] https://www.w3.org/html/
[34] https://js.org/

entre outros; 3) algoritmos de formatação para a produção de textos; e 4) mecanismo de rastreabilidade que faz a ligação do texto gerado aos modelos relacionados (KOLOVOS et al., 2017).

2.1.4.3 Graphical Modeling Framework (GMF)

O GMF[35] é um framework que provê a implementação de editores gráficos com base em metamodelos definidos em EMF (Seção 2.1.4.1). O GMF disponibiliza um conjunto de componentes reutilizáveis para editores gráficos, como impressão, exportação de imagens, ações, barras de ferramentas, entre outros. Para facilitar a criação de editores GMF, é disponibilizado um painel (GMF Dashboard) para auxiliar nas etapas do processo de criação do editor gráfico.

A ferramenta EuGENia[36] é útil na criação de editores GMF, pois gera automaticamente o editor a partir de um metamodelo EMF. EuGENia fornece um conjunto de anotações de alto nível para configurar a notação gráfica dos construtores, reduzindo a complexidade e a curva de aprendizagem para a utilização do GMF. A construção dos objetos que representam a notação gráfica do modelo é realizada por meio de um conjunto de anotações disponibilizadas pelo *framework* GMF. As principais anotações disponíveis são: 1) *Gmf.diagram,* define o componente raiz do metamodelo que representará o esquema do diagrama como um todo e definirá a extensão dos arquivos do diagrama gerado no editor; 2) *Gmf.node,* indica que um componente *EClass* será representado no diagrama e disponibiliza atributos para formatar a notação gráfica do componente, como tamanho, formato, bordas e cores; 3) *Gmf.link*, usado em componentes *EReferences* para definir as ligações entre os componentes *EClass*, identificando origem (*source*), destino (*target*) e formato da notação gráfica do *link*.

2.2 Trabalhos Correlatos

Esta seção contém um relato dos principais estudos realizados pela comunidade científica sobre a modelagem e sobre projeto conceitual e lógico de BD NoSQL. A Seção 2.2.1 aborda trabalhos que propõem modelos conceituais para aplicações de BDs NoSQL, enquanto a Seção 2.2.2 inclui a descrição de trabalhos que realizam a modelagem de BD NoSQL, e a proposição ou utilização de ferramentas CASE.

2.2.1 Modelos de Dados Conceituais para Aplicações de BD NoSQL

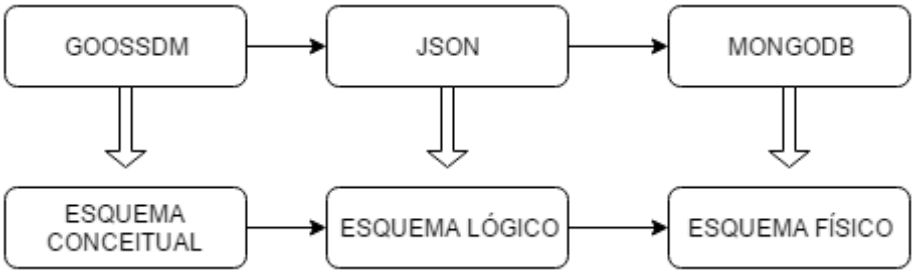
Esta seção apresenta modelos de dados conceituais propostos na literatura para auxiliar o projeto de BD NoSQL.

[35] https://www.eclipse.org/gmf-tooling/
[36] http://www.eclipse.org/epsilon/doc/eugenia/

2.2.1.1 Um framework para BD NoSQL

Banerjee et al. (2015) propõem um *framework* para modelagem conceitual de BD NoSQL. A Figura 1 mostra a visão geral do *framework*, na qual verifica-se que o esquema conceitual é convertido para um esquema lógico, representado em JSON, tendo em vista que o formato JSON pode ser diretamente armazenado em um banco NoSQL orientado a documentos. Os autores apresentam um estudo de caso, inserindo no MongoDB, um documento JSON gerado a partir do modelo conceitual proposto.

Figura 1 – Framework proposto



Fonte: Adaptado de Banerjee et al. (2015)

Para a fase de modelagem conceitual, o trabalho propõe um modelo de dados conceitual, intitulado GOOSSDM (Graph Object Oriented Semi Structured Data Model), no qual são criados objetos gráficos que representam os dados de forma conceitual. Os objetos criados são: grupo semântico elementar, o qual modela atributos que não determinam a entidade; grupo semântico determinante, o qual modela atributos identificadores das entidades; grupo semântico contextual, o qual representa as entidades do domínio; o objeto anotação, o qual é utilizado quando se pretende adicionar informações ao modelo; e, por fim, o conector de associação, o qual é utilizado para representar as conexões entre as entidades. Quanto aos tipos de relacionamentos possíveis, os principais são: contenção, o qual é o relacionamento utilizado para representar os atributos contidos nas entidades; associação, é o relacionamento entre os conectores de associação; e o relacionamento de ligação, o qual possibilita representar a herança entre os objetos.

O procedimento de conversão da modelagem conceitual definida em GOOSSDM em um esquema JSON é feito a partir de doze regras de conversão que foram definidas pelos autores. Cada regra mapeia determinados objetos da notação gráfica para as respectivas representações do esquema JSON.

A validação da notação gráfica proposta foi realizada por meio do mapeamento dos construtores do modelo GOOSSDM para o esquema JSON, especificando a correspondência entre cada elemento conceitual para uma respectiva saída em JSON.

Além disso, foi realizada a comparação da notação proposta com as características lógicas de sistemas NoSQL, verificando-se a presença de vários aspectos importantes, como estruturas hierárquicas e não hierárquicas, estruturas heterogêneas e restrições de participação.

Para validar as regras de conversão propostas, foi realizado um estudo de caso baseado em um sistema de gestão de projetos, partindo-se da modelagem conceitual em GOOSSDM, convertida no esquema JSON, o qual representa o nível lógico, e, posteriormente, implementada fisicamente no MongoDB.

Contudo, o *framework* proposto não permite a modelagem de sistemas de persistência poliglota. A tendência é a utilização de sistemas híbridos que usem tanto o modelo relacional quanto diversos modelos lógicos de BD NoSQL (VILLARI et al., 2016). Dessa forma, a criação de um modelo que estenda o modelo E-R, adicionando construtores específicos para a modelagem de BD NoSQL e de aplicações de persistência poliglota, constitui o foco da pesquisa descrita neste documento.

2.2.1.2 Projeto de BD para Sistemas NoSQL

Bugiotti et al. (2014) propõem uma metodologia para projetos de bancos de dados NoSQL. A metodologia baseia-se nas seguintes etapas principais:

- Modelagem conceitual de dados - identificar as entidades e os relacionamentos de uma aplicação;
- Concepção global - identificar as entidades de grupos de agregados;
- Particionamento de agregados - os agregados são divididos em elementos de dados menores;
- Projeto conceitual de banco de dados NoSQL - os agregados são mapeados para o modelo Noam (NoSQL Abstract Model), que tem por objetivo modelar o projeto de banco de dados NoSQL de forma intermediária, ou seja, independente da tecnologia de persistência NoSQL que será adotada;
- Implementação - mapear a representação de dados intermediária para o banco NoSQL específico.

O modelo NOAM se baseia em UML e foi definido da seguinte forma: um banco de dados Noam é um conjunto de coleções, no qual cada coleção tem um nome distinto. Uma coleção é um conjunto de blocos e cada bloco é identificado por um bloco chave, o qual é único no seu conjunto. Um bloco é um conjunto não vazio de entradas, no qual cada entrada é um par $<ek, ev>$, em que "*ek*" é o identificador

da entrada (único dentro do seu bloco) e "*ev*" é o seu valor (complexo ou escalar), chamado de valor de entrada.

A Figura 2.2 mostra o exemplo de um banco de dados modelado em NOAM no qual as linhas interiores mostram as entradas (chave - valor) e o quadro exterior denota o bloco. Cada bloco modela uma unidade de acesso e de distribuição de dados, também chamada de "*agregado*", o qual pode ser manipulado individualmente. Um agregado garante atomicidade e escalabilidade — dois elementos importantes na persistência de dados. O primeiro garante que, ao fazer uma manipulação em um determinado conjunto de dados, esta deve ser concluída totalmente ou em caso de exceções deve retornar para o estado inicial. Já a escalabilidade é a capacidade de dividir o conjunto de dados em servidores diferentes, sem abdicar da consistência dos dados.

Figura 2 – Exemplo resumido de banco de dados NOAM

Jogador

mary	login	"mary"
	nome	"Mary"
	sobrenome	"Wilson"
	jogos[0]	{jogo : **Jogo:2345**, oponente: **Jogador: rick**}
	jogos[1]	{jogo : **Jogo:2611**, oponente: **Jogador: ann**}

Jogo

2345	id	2345
	primeiroJogador	**Jogador: mary**
	segundoJogador	**Jogador: rick**
	rounds[0]	{movimentos: ... , comentarios: **...**}
	rounds[1]	{movimentos: ... , acoes: ..., magias: ...}

Fonte: Adaptado de Bugiotti et al. (2014)

A modelagem conceitual utilizada faz uso do diagrama de classes da UML (Unified Modeling Language) para representar os conceitos envolvidos no projeto, definindo as entidades e os relacionamentos. A utilização de UML é feita seguindo o DDD (Domain Driven Design) — uma metodologia de desenvolvimento de *software* orientada a objetos.

A próxima etapa é a identificação dos "*agregados*", que na sequência são mapeados para blocos NOAM. Essa etapa também se baseia na metodologia DDD, na qual cada "*agregado*" possui uma entidade como sua raiz e pode ter vários objetos de valor relacionados. A definição dos agregados está diretamente ligada às operações de acesso aos dados que deverão ser implementadas. Por exemplo, para o exemplo de BD exibido na Figura 2.2, suponha que o usuário (*Jogador*) deseja obter todos os seus dados pessoais e uma visão geral dos jogos em andamento. Quando um jogador completa uma rodada, o jogo deve ser atualizado. Assim, sugere-se que as classes de agregados sejam "Jogadores" e "Jogos".

NOAM possui maneiras de representar os agregados que são especificadas basicamente de duas formas: entrada por objeto agregado (EAO) e entrada por campos de nível superior (ETF). EAO representa cada agregado individualmente, utilizando uma única entrada, na qual a chave fica vazia e o valor são os dados completos do agre-

gado. Na ETF, a representação é feita por meio de múltiplas entradas, utilizando uma propriedade para cada campo do agregado, sendo a chave representada pelo nome do campo, e o valor da entrada correspondendo à representação dos dados persistidos.

A principal limitação para a utilização da representação EAO é o fato de representar a estrutura do agregado como um todo, além de não levar em consideração as operações de acesso aos dados — o que gera, em alguns casos, baixo desempenho do banco de dados, tendo em vista que haverá manipulação de todo o agregado para cada operação. Dessa forma, os autores sugerem o particionamento de agregados, que deve ser feito com a utilização da representação ETF, seguindo os conceitos abaixo:

- Se um agregado é pequeno e a maioria dos seus dados são acessados ou modificados juntos, ele deve ser representado por uma única entrada;
- Um agregado deve ser dividido em várias entradas, se ele for grande e possuir operações que frequentemente acessam ou modificam apenas partes específicas dele;
- Dois ou mais elementos de dados devem pertencer à mesma entrada, se eles são geralmente acessados ou modificados juntos;
- Dois ou mais elementos de dados devem pertencer a entradas distintas, se eles são geralmente acessados ou modificados separadamente.

A implementação do esquema gerado em NOAM foi realizada no Oracle NoSQL[37] , que é baseado em chave-valor (Seção 2.1.1.3).

Resultados experimentais foram coletados, os quais são baseados nas variáveis tempo de execução e tamanho do repositório. Os resultados demonstraram que operações envolvendo a recuperação de jogos no banco de dados são favorecidas pela representação EAO, independentemente do tamanho do banco de dados. Na operação de adição de rodadas aos jogos, a representação ETF produziu um melhor desempenho. Por fim, em operações com carga de trabalho mista, a representação ETF tem uma vantagem geral sobre EAO, diminuindo, no entanto, à medida que o tamanho do banco de dados aumenta.

Resultados dos testes indicaram que o projeto de bancos de dados NoSQL é uma atividade importante e que deve ser executada com planejamento prévio, pois afeta consideravelmente o desempenho, a consistência e a eficiência dos bancos de dados. A orientação a agregados é uma das principais estratégias para a modelagem de bancos NoSQL, principalmente quando requer um projeto de fragmentação para tornar as operações atômicas e aumentar a consistência dos dados.

[37] http://www.oracle.com/technetwork/database/database-technologies/nosqldb/overview/index.html

2.2.1.3 Modelo E-R Estendido para Big Data

Para Villari et al. (2016), os SGBD NoSQL substituirão as tecnologias de armazenamento relacionais nos próximos anos, mas os SGBD relacionais ainda devem ser utilizados até uma efetivação total dos SGBD NoSQL. Por isso, a solução é utilizar formas híbridas que envolvam tanto SGBD relacionais quanto sistemas NoSQL e os autores partem do princípio de que a modelagem conceitual já deve indicar os dados que serão manipulados por SGBD relacionais e os que serão guardados por sistemas NoSQL. Os autores definem dados mantidos em SGBD NoSQL como Big Data que indica um vasto volume de dados que incluem dados estruturados, não estruturados e semiestruturados.

Na etapa de modelagem conceitual de um banco de dados, foi escolhido o Digrama ER, por ser considerado um modelo melhor do que o Diagrama de Classes da UML, por definir nomes de relacionamentos e representações de participação. Identifica-se que o modelo ER possui, ainda, algumas limitações para representar Big Data. Por isso, os autores propõem a adição de elementos gráficos, gerando um modelo ER estendido capaz de representar aplicações híbridas. Essas aplicações híbridas utilizam tanto SGBD relacionais quanto BD NoSQL.

Os elementos gráficos propostos são: *Entidade Big Data*, que representa dados que devem ser armazenados em algum SGBD NoSQL; *Relacionamento Big Data*, que representa relacionamentos entre entidades Big Data; *Ligação Big Data*, que é utilizada para conectar as entidades e relacionamentos Big Data às entidades e aos relacionamentos tradicionais do modelo ER, o que demonstra a interação que deve existir entre SGBD de diferentes tipos. Por último, tem-se o elemento *Ligação Big Data Prioritária*, o qual possui a função de indicar a prioridade de um relacionamento. Esta prioridade é analisada de acordo com necessidade de processamento das possíveis operações envolvidas.

O Modelo ER estendido proposto foi usado para modelagem de um sistema de Informações em saúde. A primeira etapa realizada foi a identificação de entidades e relacionamentos que devem representar um vasto volume de dados. No estudo de caso realizado, a entidade *Perfil* e os relacionamentos *Agendamento* e *Retorno sobre infraestrutura* foram considerados para utilização de SGBD NoSQL, devido às características dos dados modelados. A segunda etapa foi a identificação das ligações entre as entidades e os relacionamentos Big Data e as entidades e os relacionamentos do modelo ER tradicional. Por último, adicionou-se uma ligação prioritária entre o relacionamento *Agendamento* e a entidade *Profissionais*, indicando que a maior parte das consultas do SGBD serão voltadas à visualização de agendamentos que envolvam determinados profissionais.

Percebe-se, portanto, que ao contrário de outros trabalhos, os quais propõem modelos específicos para BD NoSQL propostos por Bugiotti et al. (2014), Banerjee et al. (2015), Villari et al. (2016). Este propõe a adição de novos elementos gráficos ao Modelo ER. Entretanto, os construtores propostos não abordam todos os conceitos que envolvem a representação de dados para Modelos NoSQL, como, por exemplo, a utilização de dados agregados. Outro fator negativo é que não foram propostas regras de mapeamento do esquema conceitual para um esquema lógico. Isso deixa o trabalho relacionado somente ao nível conceitual de dados, não demonstrando uma evolução para os níveis lógico e físico.

2.2.2 Projetos de BD NoSQL

Esta seção contém relatos de projetos de modelagem de BD NoSQL, nos quais realiza-se a modelagem conceitual utilizando desde modelos existentes mais genéricos como UML, até ferramentas especifícas para um determinado SGBD NoSQL como a Neoclipse que dá suporte ao modelo orientado a Grafos.

2.2.2.1 Modelagem e consulta de dados em bancos NoSQL

Kaur e Rani (2013) descrevem a importância de SGBD NoSQL para as necessidades atuais das aplicações web, sobretudo devido à capacidade de fragmentação da base de dados e à possibilidade de armazenamento de dados não estruturados ou semiestruturados.

Os autores fazem uma descrição das abordagens de armazenamento de dados NoSQL existentes e realizam a comparação entre modelos de armazenamento NoSQL e o modelo ER, identificando as possíveis semelhanças. Por exemplo, para bancos de dados orientados a grafos foi visto que cada entidade do modelo ER corresponde a um *nó* do grafo, e que os atributos de entidade são equivalentes às propriedades dos *nós*, e os relacionamentos seriam as ligações entre os *nós*.

Na fase de modelagem conceitual representam-se os dados que fazem parte do domínio do sistema a ser implementado. Essa modelagem faz parte das primeiras etapas do projeto de banco de dados, ou seja, da etapa de levantamento dos requisitos de *software*, que geralmente é realizada junto às pessoas interessadas pelo projeto (usuários finais, patrocinadores). Os SGBD NoSQL não necessitam que o esquema de dados seja obrigatoriamente definido, mas a modelagem conceitual é importante por facilitar a comunicação entre as pessoas envolvidas no projeto. Devido às diversas opções de SGBD relacionais e NoSQL disponíveis no mercado, os projetistas de bancos de dados devem avaliar e definir quais ferramentas terão maior eficiência. Os autores acreditam que a modelagem conceitual de bancos NoSQL representa uma etapa crucial para o projeto destes bancos de dados, pois representa uma visão inicial

dos dados, facilitando a compreensão dos dados pelas pessoas envolvidas e auxiliando as decisões de projeto.

No estudo de caso apresentado pelos autores, o MongoDB e o Neo4J foram usados para implementar um domínio de um *site* de notícias. Esse domínio também foi implementado no SGBD relacional PostgreSQL[38] .

O esquema conceitual do domínio foi criado para o SGBD MongoDB por meio do diagrama de classes da UML, no qual, cada classe modelou uma coleção de documentos e os relacionamentos foram representados por meio de referências nos documentos. No caso de classes diretamente relacionadas, tais como "*Postagem*" e "*Comentários*", foi utilizada a notação de composição, o que demonstra que essas duas classes são armazenadas em um mesmo documento de uma coleção no MongoDB. Entende-se que essa abordagem de utilização da UML seja favorável ao caso demonstrado — o que não impede que outras abordagens de modelagem conceitual para projetos de BD NoSQL sejam investigadas.

O esquema conceitual modelado para o domínio do *site* de notícias foi representado utilizando o Neoclipse que é uma ferramenta de apoio à implementação de bancos de dados no SGBD Neo4J. Os autores apontam que a grande vantagem do uso dos bancos orientados a grafos é o desempenho na execução das consultas, pois estes sistemas não realizam junções, como acontece nos bancos relacionais tradicionais. Na realidade, suas consultas são realizadas por meio da travessia no grafo, percorrendo somente os nós que possuem relação com a consulta executada. Outra característica do Neo4J é a garantia das propriedades ACID em suas transações.

Após a criação dos bancos de dados, os autores realizaram uma carga de dados em cada instância dos SGBD MongoDB, Neo4J e PostgresSQL. Foram definidas consultas a serem realizadas nos bancos de dados como forma de demonstrar que cada SGBD provê uma linguagem de consulta diferente. No estudo de caso, as linguagens de consulta utilizadas foram respectivamente: Cypher[39] para Neo4J, MongoShell[40] para MongoDB, e SQL[41] para o PostgresSQL. Os autores chegaram à conclusão que não existe uma linguagem padrão de consulta para SGBD NoSQL, e que seria difícil a sua criação, devido às diferenças estruturais entre os tipos de SGBD NoSQL. A maioria dos desenvolvedores de *software* adotam a linguagem SQL, que é a mais comumente encontrada em sistemas de BD relacionais. Por isso, as linguagens criadas para SGBD NoSQL geralmente utilizam uma sintaxe baseada em SQL, como forma de facilitar a adaptação dos desenvolvedores.

[38] https://www.postgresql.org/
[39] https://neo4j.com/developer/cypher-query-language/
[40] https://docs.mongodb.com/getting-started/shell/client/
[41] http://pgdocptbr.sourceforge.net/pg80/plpgsql.html

2.2.2.2 Modelagem de BD NoSQL para monitoramento de veículos

Jeon, An e Lee (2015) detalham a modelagem e implementação de um BD NoSQL baseado em documentos para um sistema de monitoramento de reciclagem de veículos. Foi escolhido o MongoDB como SGBD NoSQL devido à expressividade, pois ele permite o uso de dados agregados, e também por causa de sua capacidade de processamento. Os autores consideram que o MongoDB possue os atributos adequados para a aplicação, pois permite o armazenamento de dados não-estruturados e não requer um esquema fixo — além de ser escalável, o que possibilita o uso distribuído em máquinas com menor capacidade de processamento e gerencia automaticamente a fragmentação através da funcionalidade de auto-fragmentação.

A arquitetura do sistema projetado para desenvolvimento do estudo de caso utiliza uma camada de infraestrutura composta pelo armazenamento distribuído dos dados em vários servidores nos locais de reciclagem, e um servidor de configuração central MongoDB. A camada de serviços compõe a arquitetura e fornece as informações armazenadas para os clientes.

A modelagem conceitual do BD define coleções de dados a serem armazenados e relacionamentos entre essas coleções. Para modelar relacionamentos entre coleções, foram utilizadas referências ($ref), o que possibilitou as conexões conceituais do domínio modelado. Especificamente na coleção que armazena as peças retiradas dos veículos desmontados foi utilizado um documento aninhado para persistir dados referentes a peso e volume. Essa utilização de documentos aninhados é defendida pelo conceito de *orientação agregada* que define um agregado de dados como um conjunto de objetos que formam uma unidade de acesso e de distribuição (SADALAGE; FOWLER, 2013).

Usando o SGBD Relacional MySQL e o MongoDB foram realizados testes padronizados com a execução de operações de inserção, consulta, atualização e consultas com junção. Resultados indicaram que nas operações de inserção e de atualização, à medida em que a quantidade de dados manipulada aumentava, utilizando como parâmetro de avaliação o tempo de execução, obteve-se melhor desempenho com o MongoDB. Nas operações de consultas, por outro lado, o MySQL produziu um menor tempo de resposta, quando comparado ao MongoDB.

Jeon, An e Lee (2015) mostra que a adoção de BDs NoSQL deve fundamentar-se na validação de um determinado tipo de persistência de acordo com o modelo de negócio e requisitos do projeto de BD. Contudo, a análise realizada não considerou o uso de servidores distribuídos. Por isso, pode-se considerá-la incompleta, já que aplicações web que envolvem a manipulação de vastos volumes de dados devem prever a escalabilidade.

2.3 Considerações Finais do Capítulo

Esse capítulo apresentou os principais conceitos relacionados a SGBD NoSQL, descrevendo modelos de persistência de dados destes SGBD. Foi verificado que a utilização de SGBD NoSQL tem sido crescente e, dessa forma, percebe-se que a implementação de novas ferramentas para auxiliar o projeto desses bancos de dados contribui para o avanço científico e tecnológico.

Além dos conceitos básicos de NoSQL e de persistência poliglota, foram descritas também as principais técnicas de modelagem conceitual encontradas na literatura. Também foi abordado o Framework de Modelagem Eclipse que agrupa tecnologias para desenvolver software baseado em modelos. Neste trabalho, o *framework* citado foi utilizado no desenvolvimento da ferramenta NoSQLCASE que será descrita no Capítulo 4.

Foram descritos trabalhos encontrados na literatura que utilizam SGBD NoSQL para gerenciar dados volumosos que geralmente não são estruturados, e os sistemas NoSQL fornecem o suporte para manipular estes tipos de dados. A persistência poliglota é utilizada para prover os benefícios de cada modelo de dados de acordo com os tipos de informações e necessidades de negócio das aplicações. Os dados de aplicações com persistência poliglota devem ser modelados a fim de facilitar o entendimento entre a equipe técnica e os usuários, além de auxiliar as etapas de codificação, por se tornar uma documentação única dos requisitos de dados do projeto, evitando documentos separados para cada tipo de BD. De acordo com a literatura estudada, não existe um modelo conceitual que permita a modelagem de sistemas com persistência poliglota. Por isso, esta pesquisa propõe o Modelo ERNoSQL que será descrito no próximo capítulo.

3 O MODELO DE DADOS CONCEITUAL ERNOSQL

Este capítulo apresenta o modelo proposto e descreve os construtores de modelagem, detalhando suas principais características, notações gráficas e propriedades básicas, além de exemplificar o uso dos construtores por meio de conceitos do domínio de uma aplicação de comércio eletrônico. Este capítulo inclui, ainda, a especificação de um metamodelo utilizando a notação da UML para exibir os relacionamentos entre os componentes do modelo ERNoSQL. Por fim, as regras de mapeamento do modelo ERNoSQL para a construção de esquemas lógicos NoSQL são detalhadas.

3.1 Introdução

O uso da persistência poliglota tem aumentado devido à variedade de tipos de dados, sejam eles estruturados, semiestruturados ou não estruturados que são mantidos nas aplicações web atuais, e às necessidades de manipulação distintas, como consultas complexas que envolvem muitos relacionamentos, estruturas com dados esparsos. Outros fatores relevantes para este crescimento é o aumento constante da quantidade de dados gerados pelas aplicações web, as quais requerem de ferramentas e tecnologias de banco de dados que permitam o armazenamento de dados não estruturados e a possibilidade de escalabilidade por meio da fragmentação dos dados em *clusters*, permitindo o aumento do volume de dados armazenado e evitando problemas de indisponibilidade de recursos de *hardware*, melhorando o desempenho e a operação dos SGBD.

Este trabalho de pesquisa propõe o modelo ERNoSQL para modelagem de aplicações com persistência poliglota. ERNoSQL se diferencia de outros modelos existentes, pois permite a representação dos dados, em único esquema, para aplicações de persistência poliglota que utilizam os sistemas de BD NoSQL (Seção 2.1.1), além disso, ERNoSQL também possibilita a representação de dados agregados.

Este capítulo está organizado como segue. As Seções 3.2 e 3.3 descrevem o modelo ERNoSQL e seus construtores de modelagem. A Seção 3.4 detalha um cenário de exemplo de uso de ERNoSQL, enquanto a Seção 3.5 descreve a construção de um esquema de dados com ERNoSQL. A Seção 3.6 apresenta um metamodelo especificado em UML que fornece um entendimento sobre como os construtores do ERNoSQL se relacionam entre si. A Seção 3.7 descreve as regras de mapeamento identificadas a partir dos construtores de ERNoSQL para os modelos de sistemas de BD NoSQL. Por fim, na Seção 3.8, apresenta-se uma comparação entre modelos de dados para BD NoSQL encontrados na literatura (Seção 2.2) e o modelo ERNoSQL

proposto nesta dissertação.

3.2 Modelo ERNoSQL

ERNoSQL é um modelo de dados conceitual com propósito de representar um domínio de aplicação que utilize persistência poliglota, ou seja, aplicações em que os projetistas tenham pretensão de utilizar diversos modelos de dados. Durante esta pesquisa foram estudados diversos modelos para representação conceitual de bancos NoSQL (Seção 2.2), mas nenhum deles possibilita representar esquemas conceituais poliglotas. O esquema conceitual poliglota é definido neste trabalho pela representação de diversos modelos de dados NoSQL em um único esquema conceitual, e pela possibilidade de representar dados estruturados, semi-estruturados e não estruturados.

3.3 Construtores de Modelagem do ERNoSQL

Os construtores propostos por ERNoSQL estendem o modelo E-R tradicional objetivando a modelagem de sistemas que utilizem persistência poliglota, a qual envolve variados tipos de dados estruturados, semiestruturados e não estruturados. A Figura 3.2 mostra cada construtor e a sua devida representação gráfica. A descrição de cada um dos construtores é especificada abaixo:

- *Entidade Não Estruturada* - Este construtor representa uma entidade de dados não estruturados, ou seja, não se tem uma definição de quais tipos de atributos serão armazenados por meio dela. Desta forma, este construtor não permite o uso de atributos, já que supostamente só existe o conhecimento a respeito da Entidade.

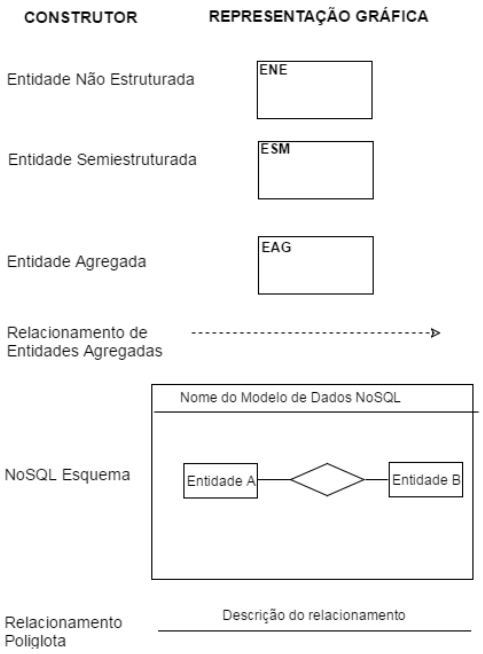
- *Entidade Semiestruturada* - Este construtor representa dados que possuem alguma estrutura definida, mas que não definem previamente limites ou tipos para os dados.

- *Entidade Agregada* - Representa uma entidade de dados modelada utilizando os conceitos de dados agregados, estruturas com vasta utilização em bancos de dados NoSQL.

- *Relacionamento de Entidades Agregadas* - Construtor utilizado para representar a conexão entre as entidades agregadas e que denota o sentido da hierarquia de agregação. Este relacionamento permite as cardinalidades Um-para-Um (1:1) e Um-para-Muitos (1:N), pois dados agregados referem-se sempre a uma entidade de dados principal que agrega outras entidades relacionadas.

- *NoSQL Esquema* - Encapsula um conjunto de instâncias de entidades e seus relacionamentos, identificando que estes representam um esquema conceitual criado para um modelo de persistência de dados NoSQL específico (documentos, grafos, chave-valor ou família de colunas).
- *Relacionamento Poliglota* - Este construtor representa o relacionamento entre instâncias do construtor *NoSQL Esquema* exibindo a relação semântica entre os esquemas que formam um esquema conceitual poliglota do tipo ERNoSQL. Este construtor possui a propriedade descrição, a qual denota uma definição do relacionamento entre as instâncias de *NoSQL Esquema*. A única cardinalidade possível deste relacionamento é Um-para-Um, pois uma instância de *NoSQL Esquema* está sempre relacionada com no máximo outra instância deste construtor.

Figura 3 – Construtores do Modelo ERNoSQL



Fonte: Elaborada pelo Autor (2017).

3.4 Cenário de Exemplo de uso de ERNoSQL

Para facilitar o entendimento sobre os construtores do ERNoSQL, relata-se, nesta seção, o cenário hipotético de criação de um fictício *site* de comércio eletrônico, intitulado *vendas2017.com* (domínio inexistente).

O processo de negócio que envolve o site *vendas2017.com* é descrito nesta seção e será usado para exemplificar a modelagem conceitual poliglota por meio dos conceitos do ERNoSQL. Este cenário de exemplo não visa detalhar todos os requisitos

de dados de um cenário real de uma empresa de comércio eletrônico, mas tem como objetivo a ilustração dos principais conceitos de ERNoSQL.

A etapa de levantamento e análise de requisitos de um projeto de banco de dados é a fase em que os projetistas utilizam técnicas (entrevistas, reuniões, questionários, etnografia) para compreensão do domínio da aplicação, descrevendo e documentando os requisitos de dados. O resultado desta etapa é um conjunto de requisitos que devem ser especificados da forma mais detalhada e completa possível. Após a fase de levantamento de requisitos, a próxima etapa é criar um esquema conceitual para o banco de dados (ELMASRI; NAVATHE, 2010).

No levantamento de requisitos sobre a operacionalização do *site* de vendas de produtos identificaram-se as seguintes informações:

- Catálogo de produtos - Lista com todos os produtos disponíveis para venda, a qual deverá exibir os detalhes de cada produto. Os produtos vendidos possuem detalhes específicos dependendo da categoria em que estejam inseridos ou do seu modelo. Por exemplo, no catálogo de televisores existem produtos da mesma categoria, intitulados Smart TVs que possuem características como: sistema operacional, quantidade de memória, processador. Essas características específicas serão diferentes em cada produto ou grupo de produtos.
- Usuários - Pessoas que acessam o *site* e se cadastram para efetuar compras.
- Carrinho de compras - Lista individual de produtos que um usuário do *site* está interessado em adquirir.
- Recomendações - Lista de produtos ofertada a cada usuário cadastrado no *site*. A lista de recomendações é baseada nas compras anteriores do usuário e em compras semelhantes de outros usuários.
- Gestão de pagamentos - Envolve as transações de pagamentos e finalizações de compras.

Durante a análise sobre o domínio relativo ao *site* de comércio eletrônico, e baseando-se nos requisitos levantados, verifica-se que as maiores cargas de acessos em *sites* de compras estão nas páginas de visualização dos produtos, pois nem todos os usuários que acessam o *site* irão efetuar a compra. A maior parte dos acessos é feita para buscar, especular e comparar preços de produtos. Então, o fluxo principal de acesso dos usuários começa na visualização do catálogo de produtos. Caso seja tomada a decisão de compra, o cliente insere o produto no carrinho. Quando o cliente insere um item no carrinho, o *site* recomenda a aquisição de outros itens. Após

adicionar todos os produtos que deseja comprar, o cliente solicita a finalização da compra, sendo redirecionado para a página de pagamento.

3.5 Construindo um Esquema Conceitual com ERNoSQL

Para a modelagem conceitual do *site vendas2017.com* descrito na seção anterior, entende-se que os projetistas possuem todas as informações sobre os requisitos da aplicação. Considera-se, neste cenário hipotético, que os projetistas envolvidos na criação do *site* têm conhecimento sobre modelos NoSQL, e, além disso, entendem os benefícios da utilização de persistência poliglota (Seção 2.1.3).

Para a construção de um esquema conceitual poliglota, define-se, inicialmente, quais as possíveis entidades que envolvem o domínio do projeto. Desta forma, foram elencadas como possíveis entidades: usuário, produto, pedido, item de pedido, carrinho de compras, pagamento, entrega e recomendação. Após esta etapa, os relacionamentos existentes entre as entidades são definidos. Os relacionamentos identificados são:

- Usuário seleciona produtos;
- Usuário finaliza pedido;
- Pedido possui pagamento;
- Pedido contém itens de pedido;
- Pedido possui entrega;
- Usuário recebe recomendação; e
- Usuário recomenda produto.

Após o levantamento inicial das possíveis entidades e dos relacionamentos envolvidos, o projetista deve responder às seguintes perguntas como forma de orientar a modelagem do esquema em ERNoSQL:

Pergunta 1 - Existem entidades que representam dados não estruturados ou semiestruturados?

Pergunta 2 - Existem dados que devem ser modelados em forma de agregados?

Com a resposta para os questionamentos acima mencionados, o projetista deve utilizar os construtores do modelo ERNoSQL na elaboração do seu esquema conceitual. Nesta fase, deve-se levar em consideração as características de dados não estruturados (dados que não possuem estrutura definida, geralmente representados

por textos, imagens, áudios, vídeos) e semiestruturados (dados que possuem estrutura irregular e mudança frequente de esquema, e considerados de natureza fortemente evolutiva). Dados não estruturados em ERNoSQL devem ser modelados por meio do construtor *Entidade Não Estruturada* que não permite atributos, quando não há informação sobre os tipos de dados a serem utilizados. Os dados semiestruturados são representados em ERNoSQL pela *Entidade Semiestruturada* e permitem a utilização de atributos, criando-se uma estrutura inicial que é flexível e pode ser alterada sem restrições.

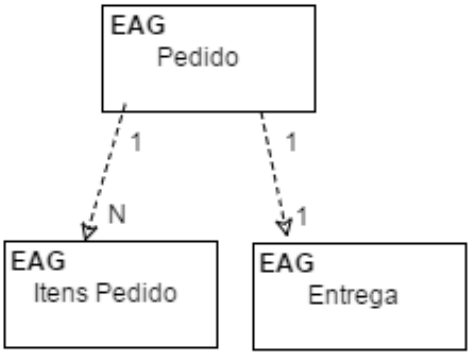
No cenário do site *vendas2017.com* a entidade *Produto* foi definida como uma entidade semiestruturada, pois cada produto pode possuir atributos específicos. A Figura 3.2 mostra um exemplo da entidade semiestruturada *Produto* com dois atributos.

Figura 4 – Exemplo de uso do construtor *Entidade Semiestruturada*

Fonte: Elaborada pelo Autor (2017)

Os dados agregados são bastante utilizados em BD NoSQL, pois um agregado, nos esquemas lógico e físico, é visto como uma estrutura única que possui dados relacionados. Foi decidido que a entidade *Pedido* deve ser modelada em forma de um agregado juntamente com as entidades itens pedido e entrega, pois elas possuem dados relacionados, os quais devem ser considerados como uma estrutura de dados única para o SGBD NoSQL. Um exemplo de uma estrutura agregada é de um documento (Seção2.1.1.1) que registra dados referentes a entidades relacionadas. Essa representação de agregados em ERNoSQL é realizada por meio do construtor *entidade agregada* e do *relacionamento de entidades agregadas* que indica quais entidades serão incorporadas na entidade de mais alto nível da hierarquia de agregados. A Figura 3.3 mostra as entidades *Itens pedido* e *Entrega* agregadas à entidade *Pedido* por meio do uso do construtor de relacionamento de entidades agregadas.

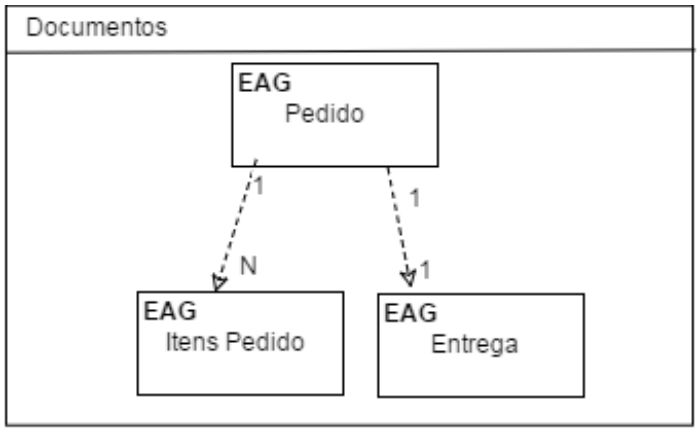
Figura 5 – Exemplo de uso do construtor Relacionamento de Entidades Agregadas



Fonte: Elaborada pelo Autor (2017)

Um dos princípios da persistência poliglota é se beneficiar dos diversos modelos de dados NoSQL existentes, utilizando os pontos fortes de cada modelo para determinadas partes do domínio de uma aplicação, como forma de atender melhor aos requisitos do projeto. Essa diversidade de bancos de dados em uma aplicação gera um ambiente poliglota. No projeto conceitual de aplicações, nas quais se pretende utilizar persistência poliglota, os projetistas se deparam com a dificuldade de modelar os dados, tendo em vista que envolve diversos conceitos que não são abordados pelo modelo E-R tradicional. Uma dessas dificuldades é o fato de visualizar quais dados estão sendo moldados para um determinado modelo de dados NoSQL. Com intuito de solucionar essa problemática, o modelo ERNoSQL propõe o construtor *NoSQL Esquema* que agrupa instâncias dos construtores, modeladas para um tipo de modelo NoSQL. A Figura 3.4 exibe o uso do construtor *NoSQL Esquema* aplicado ao cenário do *site vendas2017.com.* Esta figura mostra um esquema NoSQL definido para o modelo orientado a Documentos, na qual foram incluídas as entidades agregadas relacionadas pedido, Itens pedido e Entrega. Na parte superior da representação do construtor, é especificado o nome do modelo NoSQL que está sendo representado.

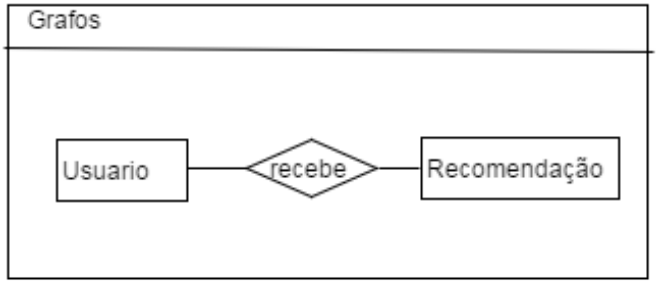
Figura 6 – Exemplo de uso do construtor NoSQL Esquema



Fonte: Elaborada pelo Autor (2017)

Como ERNoSQL é um modelo estendido do E-R tradicional, ele permite o uso dos construtores básicos do modelo E-R que devem ser utilizados para entidades que representem dados estruturados, ou seja, aquelas em que já se tem a definição de atributos representados. Dessa forma, as entidades *Usuário* e *Recomendação* do cenário de exemplo são definidas como tipos de entidades tradicionais do E-R. Mesmo com o uso de dados estruturados pode-se desejar utilizar um modelo de dados NoSQL, devido aos possíveis requisitos não funcionais do projeto, como por exemplo o desempenho nas consultas aos dados. As entidades *Usuário* e *Recomendação* estão vinculadas por meio do relacionamento "usuário recebe recomendação". Devido às características desse tipo de relacionamento que modela dados de recomendação, foi decidido utilizar um modelo de dados orientado a grafos, que é indicado para atender requisitos de consultas mais eficientes (KAUR; RANI, 2013). A Figura 3.5 mostra uma instância do construtor *NoSQL Esquema* referenciada para o modelo de banco de dados orientado a grafos, onde estão inseridas instâncias dos construtores Entidade e Relacionamento tradicionais do modelo E-R.

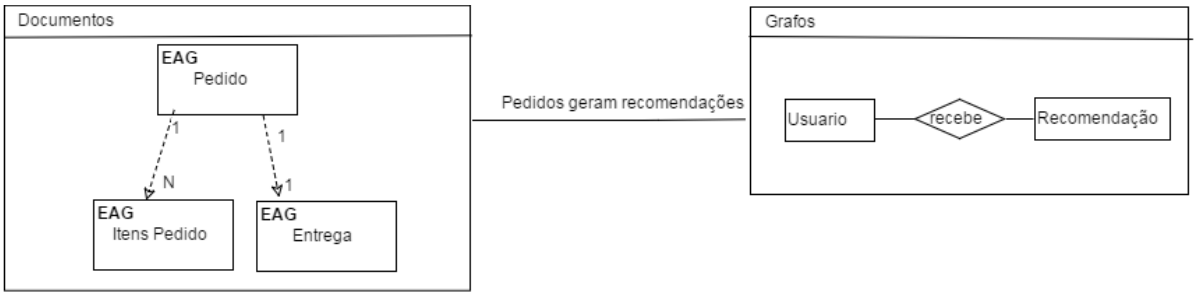
Figura 7 – Exemplo de uso do construtor NoSQL Esquema referenciado para o modelo de Grafos



Fonte: Elaborada pelo Autor (2017)

A relação entre as instâncias do construtor *NoSQL Esquema* é dada pelo uso do construtor "Relacionamento poliglota", o qual permite o uso da propriedade descrição como forma de descrever o significado do relacionamento entre os esquemas NoSQL. A Figura 3.6 mostra um exemplo de relacionamento entre esquemas NoSQL com a utilização do construtor NoSQL Esquema. Verifica-se que uma instância de NoSQL Esquema referenciada ao modelo de Documentos se relaciona com outra instância referenciada para o modelo de Grafos; a descrição desse relacionamento identifica a relação de significado conceitual entre os esquemas NoSQL. Nesse caso, é definida na propriedade descrição da instância do construtor Relacionamento poliglota em que pedidos geram recomendações.

Figura 8 – Exemplo de uso do construtor *Relacionamento entre esquemas NoSQL*



Fonte: Elaborada pelo Autor (2017)

3.6 O Metamodelo ERNoSQL

O metamodelo ERNoSQL (E-R NoSQL Metamodel) foi definido a partir do metamodelo ECore (Seção 2.1.4.2). Os construtores que compõem o metamodelo ERNoSQL são divididos em: (i) construtores base do MER (Modelo Entidade-Relacionamento) para os quais foi utilizado como referência o metamodelo ER (SOUZA, 2011); e (ii) construtores do ERNoSQL (Seção 3.3). O metamodelo fornece uma visão conceitual dos construtores do ERNoSQL.

O metamodelo ERNoSQL objetiva a representação de entidades e dos relacionamentos envolvidos na construção de um esquema conceitual de dados, e deve fornecer ao projetista da aplicação uma visão conceitual de como projetar, implementar e manter sistemas de informação com persistência poliglota. O metamodelo foi especificado em UML. No Apêndice A mostra-se o metamodelo completo e cada classe dele é detalhada a seguir.

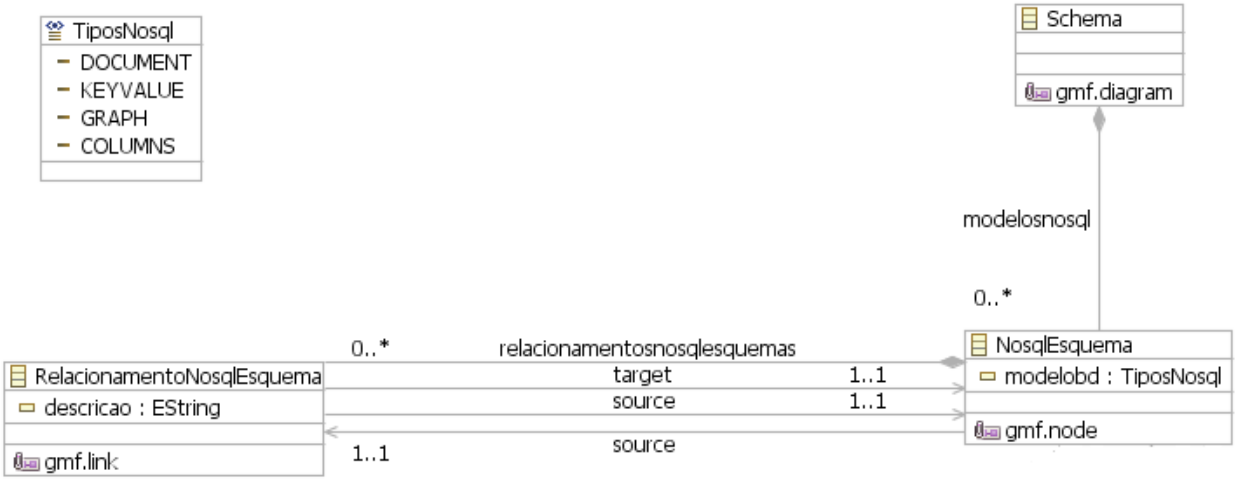
A classe *Esquema (Schema)* é o elemento raiz do metamodelo e representa a composição de objetos NoSQL Esquema. Cada instância de *NoSQL Esquema* representa um esquema conceitual de um modelo NoSQL específico. No metamodelo, uma instância de *NoSQL Esquema* pode relacionar-se com outra instância deste construtor, e o relacionamento entre eles é representado pela classe *RelacionamentoNoSQLEsquema* que possui tanto uma associação *Fonte (Source)* quanto *Alvo (Target)* para a classe *NoSQL Esquema*, indicando que o relacionamento ocorrerá entre instâncias de *NoSQL Esquema*.

A Figura 3.7 mostra um fragmento do metamodelo que contém as classes *Schema*, *NoSQLEsquema* e r*elacionamentoNoSQLEsquema* descritas no parágrafo anterior. Essas classes possibilitam a representação do esquema conceitual poliglota, tendo em vista a definição de um Esquema *(Schema)* principal que é composto por esquemas NoSQL (*NoSQLEsquema*) e seus relacionamentos (*RelacionamentoNoSQLEsquema*). Além disso, verifica-se no fragmento exposto que a classe *NoSqlEsquema* possui uma propriedade modelo de BD (modelobd) que registra o tipo de BD NoSQL

associado ao referido esquema NoSQL (*NoSQLEsquema*). Os valores válidos para a propriedade *modelobd* são definidos na classe *TiposNosql*, que trata-se de uma enumeração com os seguintes valores possíveis: DOCUMENT para o modelo NoSQL orientado a Documentos; KEYVALUE para NoSQL Chave-Valor; GRAPH que representa o modelo de Grafos; e COLUMNS para NoSQL orientado a Famílias de Colunas. A propriedade *descricao* da classe *RelacionamentoNoSQLEsquema* registra a descrição do significado do relacionamento.

A razão de cardinalidade é definida nos objetos que relacionam duas classes e especifica a quantidade de instâncias de relacionamentos que podem existir entre as mesmas, sendo representada no metamodelo da seguinte forma: um para um (1..1); nenhum até muitos (0..*), um para muitos (1..*) e muitos para muitos (*..*). Na Figura 3.7, mostra-se que a razão de cardinalidade máxima permitida no relacionamento de associação entre as classes *NoSQLEsquema* e *RelacionamentoNoSQLEsquema* é de um para um.

Figura 9 – Fragmento do Metamodelo ERNoSQL com classes que representam o esquema poliglota



Fonte: Elaborada pelo Autor (2017)

A Figura 3.8 apresenta outro fragmento do metamodelo, no qual exibe-se o restante das classes que o compõem e representam as entidades, relacionamentos e atributos. A classe *Elemento* generaliza as classes *Entidade*, *Relacionamento* e *Atributo* que representam os construtores base para a criação do esquema conceitual.

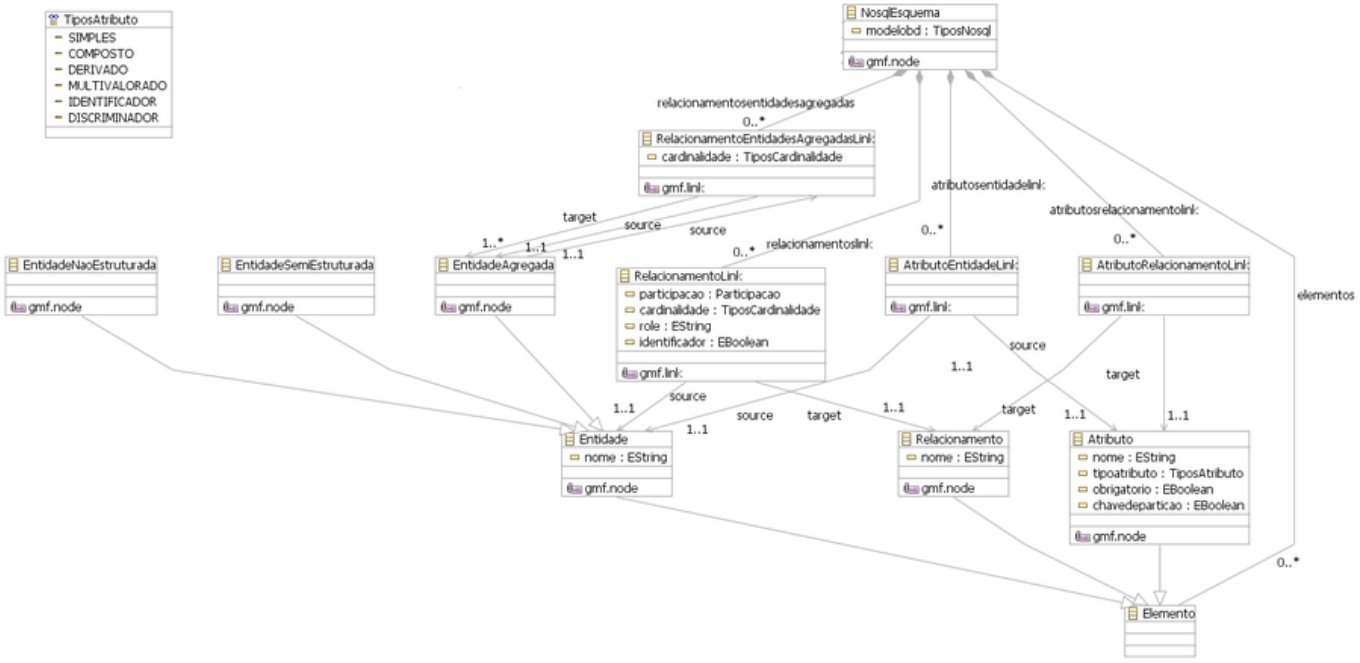
As entidades são definidas no metamodelo por meio do uso da generalização da classe pai *Entidade* que possui a propriedade *nome*. Essa classe *Entidade* é especializada em classes específicas para representar os construtores de entidades do ERNoSQL (Seção 3.3) que são respectivamente as classes *EntidadeNãoEstruturada*, *EntidadeSemiEstruturada* e *EntidadeAgregada*. As classes *EntidadeAgregada* e *Rela-*

cionamentoEntidadesAgregadasLink se relacionam por meio de associações *Fonte (Source)* e *Alvo (Target)*. A ligação entre as classes *Entidade* e *Relacionamento* é realizada por meio da classe *RelacionamentoLink*, a qual possui uma propriedade que define a *cardinalidade* do relacionamento.

A classe *Atributo* é responsável por representar os atributos e possui as propriedades *nome* e *tipoatributo*, onde esta última registra um valor válido da classe *TipoAtributo*. O relacionamento entre as classes *Atributo* e *Entidade* é definido pela classe *AtributoEntidadeLink*, da mesma forma que o relacionamento entre as classes *Atributo* e *Relacionamento* se realiza por meio da classe *AtributoRelacionamentoLink*.

Todas as classes que representam entidades, atributos e seus relacionamentos são ligadas à classe *NoSQLEsquema* por meio do relacionamento de agregação com cardinalidade zero para muitos (0..*), indicando que podem existir nenhuma ou muitas instâncias vinculadas a um objeto da classe *NoSQLEsquema*.

Figura 10 – Fragmento do metamodelo ERNoSQL com classes que representam Entidades, Relacionamentos e Atributos



Fonte: Elaborada pelo Autor (2017)

3.7 Regras de Mapeamentos para os Modelos NoSQL

Dentre as contribuições desta dissertação, tem-se a proposição de regras de mapeamento dos construtores do modelo ERNoSQL para modelos de persistência específicos de sistemas NoSQL (Seção 2.1.1). Essas regras foram definidas por meio de correspondências existentes entre os construtores conceituais do ERNoSQL e

os elementos dos modelos de persistência de sistemas NoSQL.

Durante o levantamento das regras definiu-se para cada modelo NoSQL um grupo específico de regras, expostas a seguir.

As regras de mapeamento de ERNoSQL para sistemas NoSQL orientado a documento são:

Regra 1 - Cada entidade tradicional do MER, Entidade Não Estruturada ou Entidade Semiestruturada deve ser mapeada para uma coleção de documentos.

Regra 2 - Um conjunto de Entidades Agregadas relacionadas devem ser mapeadas para uma mesma coleção utilizando a orientação a agregados, incorporando os dados das entidades agregadas em um único documento.

Regra 3 - Cada atributo deve ser mapeado para um campo no documento referente à entidade ao qual está relacionado.

Regra 4 - Atributos identificadores de entidade devem ser mapeados para campos "_id" nos documentos.

Regra 5 - Os relacionamentos tradicionais do MER devem ser mapeados para convenções DBREF[1].

As regras de mapeamento de ERNoSQL para sistemas NoSQL orientado a grafo são:

Regra 1 - Cada entidade tradicional do MER, Entidade Não Estruturada ou Entidade Semiestruturada deve ser mapeada para um rótulo, que identifica um grupo de nós do grafo.

Regra 2 - Os atributos da entidade são mapeados para as propriedades do nó.

Regra 3 - Atributo identificador é mapeado para a propriedade de identificação do nó.

Regra 4 - Os relacionamentos entre entidades devem ser mapeados para as ligações entre os nós.

Regra 5 - Os relacionamentos com cardinalidade N:N devem ser mapeados como uma ligação entre os nós, inserindo os atributos existentes no relacionamento como propriedades da ligação.

As regras de mapeamento de ERNoSQL para sistemas NoSQL orientado a chave-valor são:

Regra 1 – Cada Entidade Tradicional do MER, Entidade Não Estruturada ou

[1] https://docs.mongodb.com/v3.2/reference/database-references

Entidade Semiestruturada deve ser mapeada para a instância chave-valor.

Regra 2 - Um conjunto de Entidades Agregadas relacionadas devem ser mapeadas para uma instância chave-valor, sendo a chave representada pelo atributo identificador da Entidade Agregada principal, e o valor compõe o restante dos dados agregados.

Regra 3 - O atributo identificador deve ser mapeado para a chave, e os outros atributos para o valor.

As regras de mapeamento de ERNoSQL para sistemas NoSQL orientado a famílias de colunas são:

Regra 1 - Cada Entidade Tradicional do MER, Entidade Não Estruturada ou Entidade Semiestruturada deve ser mapeada para uma tabela.

Regra 2 - Um conjunto de Entidades Agregadas relacionadas deve ser mapeado para uma tabela, sendo a chave primária representada pelo atributo identificador da Entidade Agregada principal.

Regra 3 - Os atributos devem ser mapeados para propriedades da tabela.

Regra 4 - Atributos identificadores devem compor a chave primária da tabela.

3.8 Comparação entre Modelos de Dados para BD NoSQL

O Quadro 2 exibe uma análise comparativa entre os modelos conceituais propostos para a modelagem de BDs NoSQL e o modelo ERNoSQL proposto nesta dissertação. Nesta análise, verificam-se os diferenciais do modelo ERNoSQL, pois possibilita a modelagem de esquemas conceituais poliglotas. No Quadro 2, também mostra-se que a modelagem de dados agregados não é possibilitada por todos os modelos analisados, pois os outros modelos observados não possuem construtores que permitam a representação de dados agregados. Verifica-se que o ERNoSQL propõe uma modelagem com diferenciais em relação aos modelos analisados, pois permite a modelagem de esquemas conceituais poliglotas e a representação de dados agregados.

Quadro 2 – Análise comparativa entre trabalhos relacionados e ERNoSQL

Trabalho Relacionado	**Modelo conceitual**	**Modelagem poliglota**	**Modelagem de agregados**	**Ferramenta CASE**	**Modelos NoSQL**
GOOSDM (Seção 2.2.1.1)	GOOSDM	Não	Não	Não	Não define modelos
NOAM (Seção 2.2.1.2)	NOAM	Não	Sim	Não	Não define modelos
Modelagem de dados em NoSQL (Seção 2.2.2.1)	UML para NoSQL Documentos. Neoclipse para NoSQL Grafos.	Não	Sim (Associação de Composição)	Neoclipse	Documentos e grafos
Modelagem de BD NoSQL para monitoramento de veículos (Seção 2.2.2.2)	UML	Não	Não	Não	Documentos
E-R Estendido para Big Data (Seção 2.2.1.3)	E-R Estendido	Não	Não	Não	Não define modelos.
ERNoSQL (Proposto nesta dissertação)	ERNoSQL	Sim	Sim	Sim (NoSQLCASE)	Documentos, grafos, chave-valor e famílias de colunas

Fonte: Elaborada pelo Autor (2017)

3.9 Considerações Finais do Capítulo

Este capítulo apresentou o modelo de dados conceitual ERNoSQL. O ERNoSQL estende o modelo ER tradicional adicionando construtores que possibilitam a representação de dados não estruturados, semiestruturados e o conceito de orientação a agregados que é relevante na criação de bases de dados NoSQL. O ERNoSQL também possibilita a representação de um esquema conceitual poliglota (Seção 2.1.3) e se diferencia dos demais trabalhos citados nesta dissertação por propor um conjunto de novos construtores capazes de criar um esquema de dados conceitual poliglota.

O objetivo de ERNoSQL é fornecer ao projetista de banco de dados uma abor-

dagem de modelagem que inclua os conceitos de dados utilizados no âmbito dos SGBD NoSQL e que possibilite a representação de diversos modelos de dados NoSQL em um esquema conceitual único. Para ilustrar os conceitos do ERNoSQL, um metamodelo especificado em UML foi descrito neste capítulo mostrando a representação dos objetos e dos relacionamentos envolvidos na construção de um esquema de dados conceitual ERNoSQL.

Para gerar um esquema lógico de dados para SGBDs NoSQL, este capítulo descreveu ainda regras de mapeamento para converter um esquema conceitual ERNoSQL em esquemas específicos de SGBDs NoSQL. Assim, além de construir um esquema conceitual para entendimento do domínio do problema, o projetista de banco de dados pode implementar o esquema lógico de dados em um SGBD NoSQL. Essa geração de esquemas lógicos a partir de um esquema ERNoSQL é facilitada pelo uso da ferramenta NoSQLCASE cujas funcionalidades e detalhes de implementação são descritos no capítulo seguinte.

4 A FERRAMENTA NOSQLCASE

Este capítulo apresenta a ferramenta NoSQLCASE que foi desenvolvida para dar suporte aos construtores de modelagem do modelo conceitual ERNoSQL. Para isto, este capítulo exibe uma visão geral da ferramenta CASE proposta, discute a arquitetura de software usada no seu desenvolvimento, apresenta o ambiente gráfico, detalha suas funcionalidades que são ilustradas por meio de um diagrama de casos de uso da UML, e, por fim, descreve o processo de utilização da ferramenta por meio de um diagrama de atividades da UML.

4.1 Introdução

NoSQLCASE é uma ferramenta computacional com o propósito de permitir a modelagem conceitual de aplicações com persistência poliglota baseada no modelo ERNoSQL (Capítulo 3). O *download* da ferramenta e as instruções iniciais de instalação e uso estão disponíveis em: https://sites.google.com/a/cin.ufpe.br/nosqlcase/.

Essa ferramenta auxilia os projetistas de software na elaboração de esquemas conceituais poligotas por meio de um ambiente de desenho gráfico que possibilita o uso dos construtores de modelagem de ERNoSQL. Além do editor gráfico, NoSQLCASE fornece funcionalidades de armazenamento dos esquemas criados em arquivos, no formato XMI, que é um padrão da OMG, sendo utilizado por diversas ferramentas de modelagem, além de facilitar integrações com outros esquemas de dados. NoSQLCASE também possibilita o mapeamento para esquemas lógicos expressos em *scripts* de linguagens especifícas de SGBD NoSQL de acordo com as regras definidas nesta dissertação (Seção 3.7), implementadas por meio de algoritmos de conversão criados utilizando a linguagem EGL (Seção 2.1.4.2). Os algoritmos utilizados na conversão de ERNoSQL para esquemas lógicos de SGBD NoSQL estão disponíveis no apêndice C e em https://github.com/inacioloy/NoSQLCase**.**

NoSQLCASE foi desenvolvida a partir da criação de um metamodelo (Seção 3.6) representado textualmente na linguagem Emfatic (Seção 2.1.4.1), a qual faz parte do Eclipse Epsilon que agrupa um conjunto de linguagens e ferramentas para geração de código, transformação de modelo para modelo, validação de modelo, comparação, migração e refatoração. O ambiente gráfico da ferramenta foi construído utilizando o EuGENia GMF Tool (Seção 2.1.4.3), que gera um editor gráfico a partir de um metamodelo descrito em Emfatic. Além disso, NoSQLCASE fornece anotações de alto nível para customização do ambiente gráfico.

Este capítulo está organizado como segue. A Seção 4.2 detalha a arquitetura

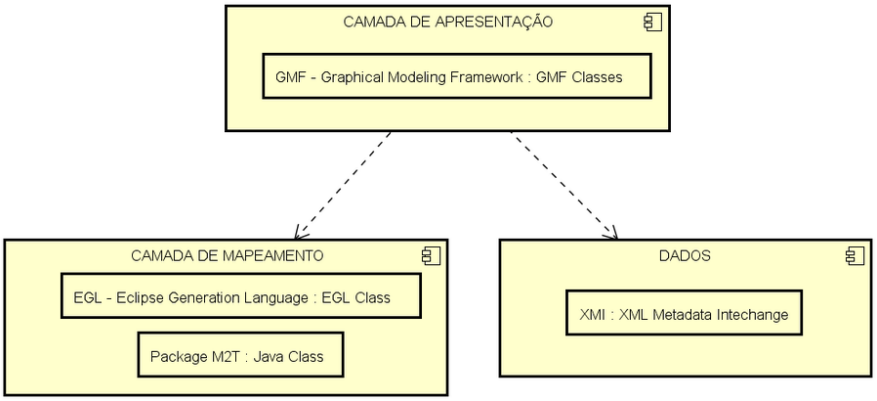
do software. A Seção 4.3 ilustra e descreve o ambiente gráfico disponibilizado pela ferramenta, enquanto a Seção 4.4 mostra o cenário de casos de uso, representado em UML, e descreve cada uma das funcionalidades disponíveis em NoSQLCASE. Por fim, na Seção 4.5, é descrito o processo de criação de esquemas conceituais de dados em NoSQLCASE, ilustrado em um diagrama de atividades da UML.

4.2 Arquitetura de Software

A arquitetura de software define os componentes do sistema e como eles se relacionam. Na ferramenta NoSQLCASE, a arquitetura está dividida em 3 camadas, sendo: (i) Camada de Dados, que se refere à forma como os dados são armazenados; (ii) Camada de Mapeamento, referindo-se às regras de negócio da aplicação; e (iii) Camada de Apresentação que está relacionada à interface gráfica do software. O desenvolvimento da ferramenta seguiu a abordagem de desenvolvimento orientada a modelos, a qual baseia-se na criação de modelos como classe principal de artefatos para o desenvolvimento do software (VARA; MARCOS, 2012).

A Figura 4.1, mostra um diagrama de componentes da UML que exibe uma visão da arquitetura com as linguagens e framework utilizados. Pode-se visualizar que na camada de apresentação, utilizou-se o framework GMF para geração do editor gráfico. Na camada de mapeamento, foram utilizadas classes expressas na linguagem EGL para realizar a conversão do esquema conceitual projetado em uma saída textual referente a linguagens específicas dos SGBD NoSQL. Neste componente também existe um pacote de classes java que implementam a tela de seleção do mapeamento e a realização da chamada às funcionalidades de conversão. Na camada de armazenamento de dados, verifica-se que os esquemas salvos pelo usuário ficam armazenados em arquivos XMI, que é o padrão utilizado pelo GMF.

Figura 11 – Componentes da Arquitetura de NoSQLCASE



Fonte: Elaborada pelo Autor (2017)

As tecnologias usadas no desenvolvimento da arquitetura de NoSQLCASE con-

sistem na linguagem de programação Java, sendo utilizado como IDE (Integrated Development Environment), o Eclipse Epsilon, o qual está disponível para download em http://www.eclipse.org/epsilon/download/. Tendo em vista a utilização de desenvolvimento orientado a modelos por meio do GMF, o artefato principal para o desenvolvimento da ferramenta foi o metamodelo definido em EMF, que serviu como classe principal para a geração do projeto da ferramenta CASE.

4.3 Ambiente Gráfico

A interação do usuário com a aplicação é realizada por meio da camada de apresentação onde encontram-se os componentes e as bibliotecas responsáveis pela interface gráfica da aplicação. No ambiente gráfico de NoSQLCASE o usuário visualizará: (1) Área de Edição; (2) Menu de Componentes; (3) Menu Superior; e (4) Área de Saída de Texto. A Figura 4.2 exibe o ambiente gráfico e define cada parte dele de acordo com a numeração dos itens descritos em sequência.

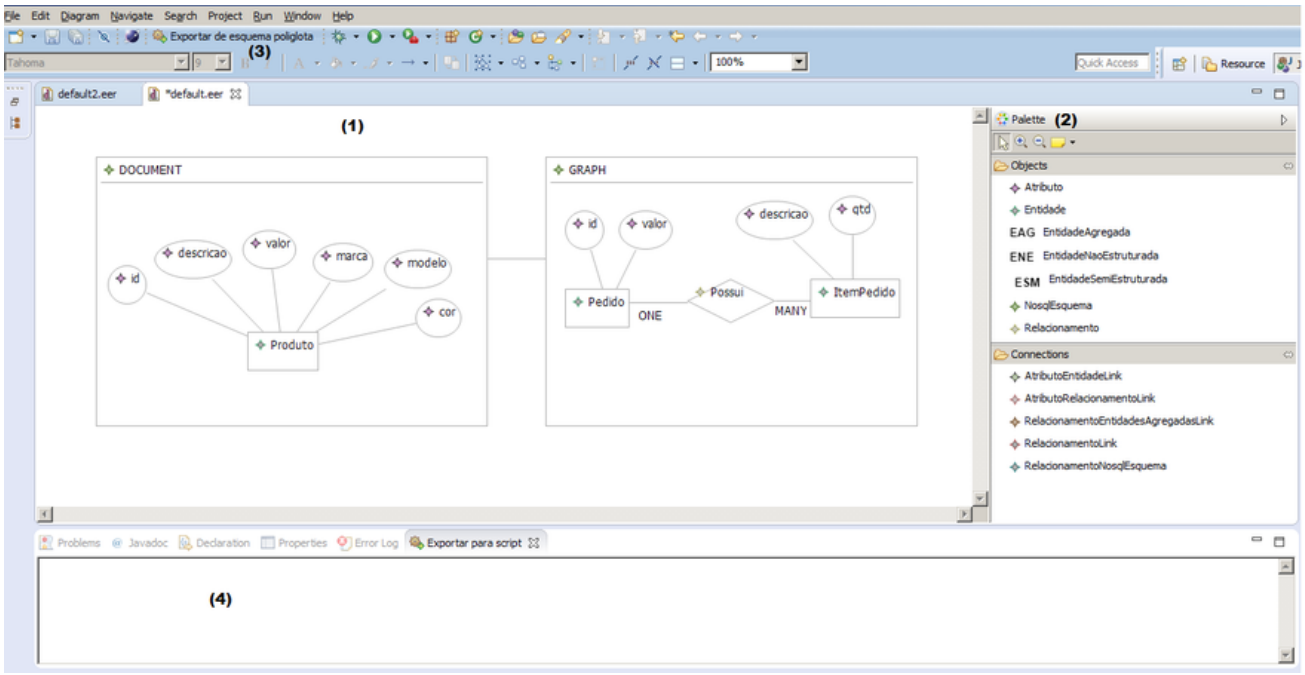
1) Área de Edição: Corresponde à área disponível para a criação, edição e visualização de esquemas conceituais de dados em consonância com o metamodelo ERNOSQL.

2) Menu de Componentes: Inclui os elementos propostos para a criação de esquemas conceituais, os quais podem ser adicionados pelo usuário por meio de um duplo clique ou arrastando o elemento desejado para a Área de Edição.

3) Menu Superior: Fornece funcionalidades de criar, abrir ou salvar arquivos pela ferramenta, além de disponibilizar os botões de acesso às funcionalidades de mapeamento do esquema conceitual de dados.

4) Área de Saída de Texto: Representa a área onde são exibidos os códigos gerados após a execução das funcionalidades de mapeamento.

Figura 12 – Ambiente Gráfico de Modelagem em NoSQLCASE

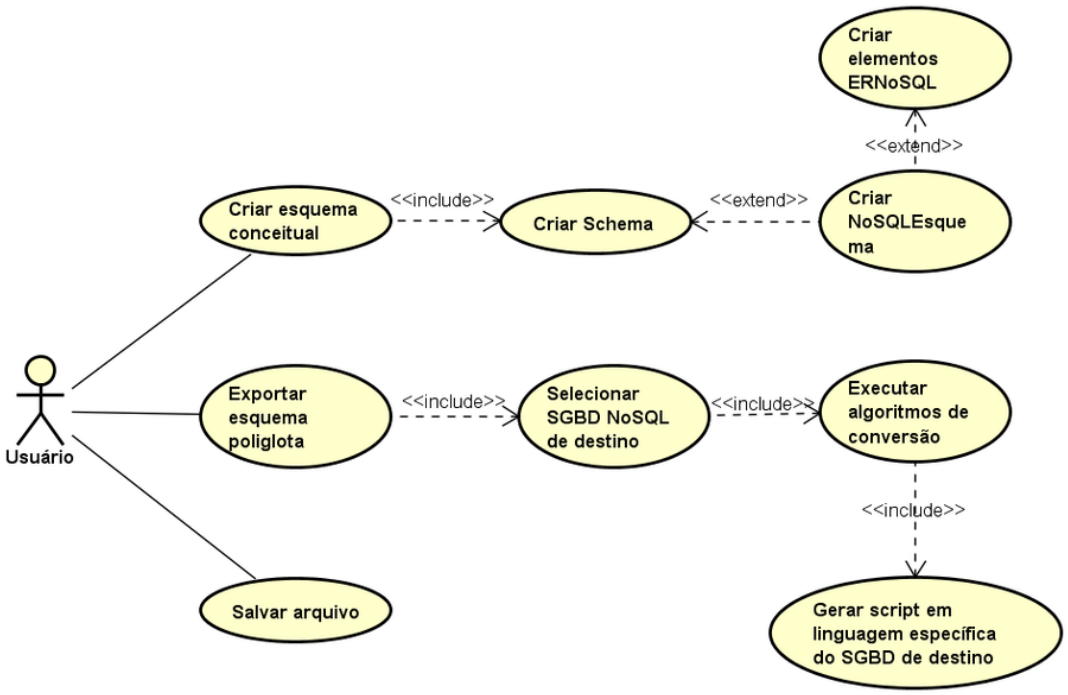


Fonte: Elaborada pelo Autor (2017)

4.4 Cenário de Casos de Uso

As funcionalidades da ferramenta NoSQLCASE foram modeladas por meio do Diagrama de Casos de Uso da UML. Propõe-se um cenário no qual o “ator”, representado como um usuário da ferramenta, interage com os “casos de uso” que correspondem às funcionalidades disponíveis. Os casos de uso da Figura 4.3 são detalhados a seguir:

Figura 13 – Cenário de Casos de Uso em NoSQLCASE



Fonte: Elaborada pelo Autor (2017)

- **Criar esquema conceitual -** Fornece as funcionalidades de criação de novos esquemas conceituais. Essa funcionalidade se inicia com a criação de uma instância do elemento “Schema”, e a partir deste o usuário deve inserir as instâncias dos construtores do modelo ERNoSQL, com os quais pode elaborar o seu esquema conceitual poliglota.
- **Exportar esquema poliglota -** Inclui as funcionalidades para conversão do esquema conceitual em um *script* de criação do BD NoSQL expresso por uma linguagem específica do SGBD NoSQL selecionado. A seleção do SGBD NoSQL para o qual será realizada a conversão é realizada pelo caso de uso “Selecionar SGBD NoSQL de destino”. Após a seleção do SGBD de destino, a ferramenta aplica as regras de mapeamento por meio de algoritmos de conversão e gera o *script*, exibido na área de saída de texto da ferramenta.
- **Salvar arquivo -** Agrupa funções de armazenamento de arquivos, no formato XMI, para os esquemas conceituais elaborados na ferramenta NoSQLCASE.

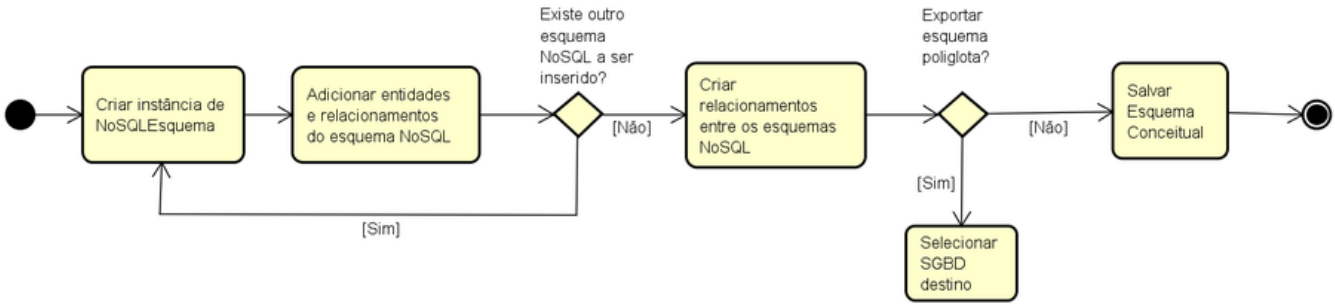
4.5 Construção de Esquemas com NoSQLCASE

O processo de construção de esquemas conceituais em NoSQLCASE é ilustrado na Figura 4.4. Esse processo se inicia com a criação de uma instância do construtor *NoSQL Esquema*, na qual devem ser inseridas as entidades e relacionamentos referentes àquele esquema NoSQL. Em seguida, se necessário, pode-se adicionar

outra instância de NoSQL Esquema referente a outro modelo NoSQL. Ao concluir a modelagem dos esquemas NoSQL que fazem parte do esquema conceitual poliglota, deve-se relacionar as instâncias de NoSQL Esquema definindo o significado existente entre esses relacionamentos.

O projetista que utiliza a ferramenta NoSQLCASE, após a elaboração do seu esquema conceitual poliglota, tanto pode salvar este esquema que representa uma visão conceitual dos dados para o domínio da aplicação projetada, como também, pode exportar o esquema para *scripts* de SGBD NoSQL. Estes *scripts* devem ser utilizados na criação das instâncias físicas do banco de dados modelado na ferramenta.

Figura 14 – Processo de criação de esquemas em NoSQLCASE



Fonte: Elaborada pelo Autor (2017)

4.6 Considerações Finais do Capítulo

Este capítulo apresentou a ferramenta NoSQLCASE desenvolvida para dar suporte ao modelo conceitual proposto nesta dissertação. O objetivo dessa ferramenta é fornecer ao projetista da aplicação um apoio computacional para auxiliá-lo nas atividades de modelagem de uma aplicação que use diversos tipos de dados (estruturados, semiestruturados e não estruturados), além do uso de diferentes modelos lógicos de sistemas NoSQL (Seção 2.1.1) definidos neste trabalho como aplicação de persistência poliglota.

Para o desenvolvimento dessa ferramenta, foram utilizados os conceitos da arquitetura em camadas e a metodologia orientada a modelos. O primeiro conceito permitiu a separação entre as camadas de apresentação, regras de negócios e a camada de dados, sendo essa abordagem vital para o isolamento da camada de dados. O segundo conceito foi utilizado na implementação por meio do desenvolvimento orientado a modelos que utiliza a criação de modelos como classe principal de artefatos para o desenvolvimento do *software* e oferece vantagens como, por exemplo, maior produtividade, facilidade de manutenção, entre outras.

Dentre as suas principais características, NoSQLCASE permite a criação de um

esquema conceitual de dados, no qual as instâncias dos construtores de modelagem do modelo ERNoSQL representam abstrações dos conceitos referentes à persistência poliglota de dados que incluem variados tipos de dados e modelos NoSQL. Isso permite que os requisitos de dados de um domínio de aplicação de persistência poliglota possam ser modelados e compreendidos pelo projetista da aplicação. Após a construção do esquema conceitual, a ferramenta CASE permite a geração do esquema lógico, representado por um *script* de criação do BD NoSQL gerado a partir das regras de mapeamento especificadas na Seção 3.7. Assim, o *script* gerado pode ser interpretado em um SGBD NoSQL. Para exemplificar o uso da ferramenta CASE em um domínio de problema real, o capítulo seguinte descreve a construção de um esquema conceitual de dados referente a uma aplicação para acompanhamento de indicadores acadêmicos de uma instituição federal de ensino técnico e superior do Nordeste do Brasil, e compara o esquema criado em NoSQLCASE com esquemas gerados por duas outras ferramentas de modelagem conceitual de dados.

5 UM ESTUDO DE CASO COM NOSQLCASE

Este capítulo implementa um estudo de caso por meio da construção de um esquema conceitual de dados por meio de ferramentas de modelagem existentes e da ferramenta NoSQLCASE proposta, além de demonstrar as funcionalidades especifícas de NoSQLCASE para modelagem de esquemas conceituais poliglotas. Para tanto, foi utilizado o domínio de uma aplicação para acompanhamento de indicadores acadêmicos, que está em processo de desenvolvimento por uma instituição de ensino técnico e superior do Nordeste brasileiro. Este capítulo está organizado da seguinte forma. A Seção 5.1 introduz o estudo de caso discutindo sua motivação e as etapas para sua elaboração. A Seção 5.2 discorre sobre o planejamento do estudo, enquanto a Seção 5.3 descreve o seu desenvolvimento. A Seção 5.4 discute as análises e os resultados obtidos com a realização do estudo de caso. Por fim, a Seção 5.5 apresenta as considerações finais deste capítulo.

5.1 Introdução

Um estudo de caso pode ser aplicado como uma estratégia de pesquisa comparativa, ao relacionar os resultados de usar um método ou tecnologia, com os resultados de usar outra abordagem. Uma das formas de realizar um estudo de caso é comparar projetos semelhantes (sister project). Nesse caso, a elaboração do estudo é realizada por meio da aplicação de projetos similares, sendo que um deles utiliza a tecnologia ou metódo novo proposto, enquanto os outros utilizam tecnologias ou métodos já conhecidos (WOHLIN et al., 2012).

As etapas que serão utilizadas na implementação do estudo de caso são: 1) planejamento (definição de objetivos e organização dos protocolos e dos procedimentos para o desenvolvimento do estudo); 2) Desenvolvimento (operação do estudo de caso a partir dos procedimentos definidos na etapa de planejamento); e 3) Análise e os resultados, que dizem respeito à execução do estudo, identificando os pontos principais de acordo com as questões definidas também no planejamento.

Como domínio de aplicação para o estudo abordado, utiliza-se os requisitos de uma aplicação para acompanhamento de indicadores acadêmicos que está sendo desenvolvida pela Diretoria de Gestão de Tecnologia da Informação do Instituto Federal de Ciência e Tecnologia do Ceará. Para este estudo de caso, será utilizado o nome *Indicadores Acadêmicos* como menção à aplicação utilizada.

As próximas seções estão organizadas como segue: a Seção 5.2 detalha o planejamento do estudo de caso realizado, definindo os seus objetivos, questões

norteadoras, descrição do domínio de aplicação utilizado, e, por fim, o protocolo com a lista de atividades que foram aplicadas e os instrumentos que foram utilizados na operacionalização do estudo. A Seção 5.3 explica o desenvolvimento do estudo de caso aplicando as atividades planejadas em cada uma das ferramentas CASE correlatas selecionadas e na ferramenta proposta nesta dissertação. A análise do estudo e discussão dos resultados são detalhadas na Seção 5.4. Por fim, a Seção 5.5 discorre sobre as considerações finais a respeito do estudo de caso implementado.

5.2 Planejamento do Estudo de Caso

Esta seção descreve o planejamento do estudo de caso, incluindo seus objetivos (Seção 5.2.1), questões do estudo (Seção 5.2.2), descrição do caso estudado (Seção 5.2.3) e, por fim, detalha-se o protocolo usado na operacionalização do estudo e os instrumentos utilizados (5.2.4).

5.2.1 Objetivos

O objetivo principal deste estudo de caso é realizar a comparação de ferramentas CASE existentes que permitem a modelagem conceitual de dados segundo o modelo ER, com a ferramenta proposta nesta dissertação. Esta comparação tem como objetivo investigar a viabilidade da ferramenta NoSQLCASE no auxílio de construções de esquemas conceituais de dados para aplicações de persistência poligota. Também objetiva-se demonstrar a utilização das funcionalidades de mapeamento, possibilitadas por NoSQLCASE, para scripts expressos em linguagens de SGBDs NoSQL.

Tem-se ainda como objetivo deste estudo, demonstrar a viabilidade da modelagem de dados agregados por meio dos construtores do modelo ERNoSQL, com o auxílio da ferramenta NoSQLCASE.

5.2.2 Questões do Estudo

As seguintes questões norteadoras foram definidas para o estudo de caso aqui descrito.

Questão 1: Quais ferramentas analisadas permitem a modelagem conceitual de dados para domínios de aplicações de persistência poliglota?

Questão 2: Quais ferramentas analisadas permitem a modelagem de dados baseada no conceito de dados agregados?

Questão 3: Quais ferramentas analisadas permitem o mapeamento do esquema conceitual criado para esquemas lógicos de modelos NoSQL?

5.2.3 Descrição do Domínio da Aplicação - Contexto de Estudo

O Sistema Nacional de Informações da Educação Profissional e Tecnológica (SISTEC) tem como finalidade promover mecanismos de registro e controle dos dados da educação profissional e tecnológica no país. SISTEC possibilita o acompanhamento de programas e de políticas públicas da educação profissional e tecnológica e disponibiliza para a sociedade, informações das ofertas de cursos técnicos de nível médio.

As unidades de ensino da Rede Federal de Educação Profissional, Científica e Tecnológica (RFEPCT) são responsáveis por cadastrar no SISTEC e manter atualizadas as informações sobre seus cursos, alunos e matrículas, conforme previsto na Resolução CNE n° 03 de 30/09/2009. SISTEC tem se consolidado em várias utilizações: estudos estatísticos, geração de indicadores de gestão, planejamento e monitoramento de políticas públicas e distribuição de recursos para as instituições na matriz orçamentária. O sistema disponibiliza planilhas com fórmulas estatísticas a serem utilizadas pelas unidades de ensino no cálculo de indicadores para o acompanhamento das metas e para a elaboração de relatórios de gestão.

Devido ao cálculo dos indicadores não estar automatizado em um sistema informatizado, gerando retrabalho e suscetibilidade a erros, foi solicitado à Diretoria de Gestão de Tecnologia da Informação do Instituto Federal de Educação, Ciência e Tecnologia do Ceará, a implementação de sistema informatizado que importe os dados do SISTEC e realize o cálculo dos indicadores. O objetivo desta solicitação é disponibilizar essas informações para acesso pelos departamentos de Estatítica e Pró-Reitoria de Ensino, auxiliando no controle e gerenciamento das ofertas de vagas e consequentemente, no cumprimento das metas institucionais planejadas. As informações detalhadas sobre o cálculo dos indicadores estão disponíveis em um manual disponibilizado pelo Ministério da Educação, o qual pode ser acessado em: http://sitesistec.mec.gov.br/images/arquivos/pdf/manual_de_indicadores_da_rfepct_2016.pdf.

Na etapa de levantamento dos requisitos da aplicação *Indicadores Acadêmicos*, analisou-se os fluxos de processo relativos ao domínio modelado, chegando-se à conclusão de que o sistema devia possuir as seguintes entidades:

- *Curso* - Possui atributos referentes aos cursos ofertados.
- *Ciclo de Matrícula* - Envolve a oferta de um curso com uma carga horária e um período de realização definidos, visando englobar um conjunto de matrículas de alunos para a obtenção de uma mesma certificação ou de um diploma.
- *Aluno* - Compreende as informações dos alunos que estão vinculados aos cursos e aos ciclos de matrículas.

- *Indicador* - Inclui dados referentes aos indicadores calculados para um determinado período e ciclo de matrícula. Esses indicadores devem ser armazenados para estatísticas e geração de relatórios gerenciais.

Os relacionamentos definidos no domínio analisado são:

- *Possui* - Relacionamento entre as entidades *Curso* e *Ciclo de Matrícula* com cardinalidade um para muitos (1 - N) indicando que um Curso "possui" muitos Ciclo de Matrícula, mas um Ciclo de Matrícula é referente a somente um Curso.
- *Pertence* - Relacionamento entre as entidades *Aluno* e *Ciclo de Matrícula* com cardinalidade muitos para muitos (N - M) indicando que um Aluno pertence a muitos Ciclos de Matrícula e um Ciclo de Matrícula tem vários Alunos.
- *Resulta* - Relacionamento entre as entidades *Ciclo de Matrícula* e *Indicador* com cardinalidade um para muitos (1 - N) indicando que um Ciclo de Matrícula pode resultar em muitos Indicadores, mas um Indicador é referente a somente um Ciclo de Matrícula.

5.2.4 Protocolo e Instrumentos

O protocolo de um estudo de caso descreve a lista de procedimentos a serem aplicados para realizar o estudo. Ele serve como um guia para orientar o pesquisador no desenvolvimento do estudo de caso (WOHLIN et al., 2012). O estudo de caso descrito nesta dissertação objetiva investigar a viabilidade da ferramenta NoSQLCASE na construção de esquemas conceituais de dados para aplicações de persistência poliglota. A seleção das ferramentas CASE utilizadas foi baseada em pesquisas por ferramentas gratuitas de modelagem conceitual de bancos de dados e fundamentadas no modelo ER. Para tanto, foram selecionadas as ferramentas EerCASE[1] e ERDPlus[2] para o desenvolvimento do estudo de caso por meio da criação de um esquema conceitual referente ao domínio da aplicação *Indicadores Acadêmicos*, descrito na seção anterior, nessas ferramentas e na ferramenta NoSQLCASE proposta. Foram definidos os seguintes procedimentos a serem realizados em cada ferramenta utilizada neste estudo de caso:

- Criação do esquema conceitual de dados de acordo com os requisitos do domínio da aplicação *Indicadores Acadêmicos*.
- Explorar as funcionalidades de mapeamento e de geração de esquemas lógicos disponíveis em cada ferramenta CASE.

[1] https://sites.google.com/a/cin.ufpe.br/eercase/
[2] https://erdplus.com/#/

- Identificar se a ferramenta possibilita a modelagem baseada no conceito de dados agregados.
- Identificar se a ferramenta permite a modelagem de aplicações de persistência poliglota.
- Identificar se a ferramenta apresenta erros em tempo de execução ou interrupções que impeçam ou prejudiquem o seu funcionamento e uso.

Quanto à instrumentação utilizada para a execução deste estudo de caso, foram utilizados os seguintes instrumentos e software:

- Computador portátil com os seguintes recursos: processador Intel(R) Core(TM) i5 CPU M 480 de 2.67GHz, 6 gibabytes de memória RAM, executando o sistema operacional Windows 7 na versão de 64 bits.
- Ambiente de execução Java na versão 8. Esse ambiente é necessário para a execução das ferramentas EerCASE e NoSQLCASE.
- Ferramenta EerCASE para o projeto conceitual de banco de dados, segundo o modelo E-R (SOUZA, 2011).
- Ferramenta ERDPlus para projeto de banco de dados segundo os modelos E-R e relacional.
- Ferramenta NoSQLCASE para projeto conceitual de banco de dados para aplicações de persistência poliglota.
- SGBD MongoDB na versão 3.4.5 gratuita, disponível em: https://www.mongodb.com/download-center#community. A escolha do MongoDB foi motivada pela sua popularidade entre os sistemas de BD NoSQL.
- Ferramenta web para validação de documentos JSON, disponível em: https://jsonlint.com/, acesso em: 10/07/2017.

5.3 Desenvolvimento do Estudo de Caso

O desenvolvimento do estudo de caso procedeu de acordo com os procedimentos planejados no protocolo, ou seja, efetuando a modelagem conceitual do domínio da aplicação *Indicadores Acadêmicos* em cada uma das ferramentas CASE. Além disso, foram exploradas as funcionalidades das ferramentas na conversão para esquemas lógicos e foi investigado se existe a possibilidade da modelagem de aplicações de persistência poliglota.

As próximas seções estão organizadas como segue. A Seção 5.3.1 descreve a execução do estudo por meio da ferramenta EerCASE e mostra o esquema conceitual criado nela. A Seção 5.3.2 detalha o estudo na ferramenta ERDPlus e discute as suas funcionalidades. A Seção 5.3.3 descreve o estudo feito com a ferramenta NoSQLCASE, e detalha, ainda, as funcionalidades de exportação para os esquemas lógicos considerados por NoSQLCASE.

5.3.1 Execução na Ferramenta EerCASE

A ferramenta EerCASE está disponível para *download* em: https://sites.google.com/a/cin.ufpe.br/eercase/. As principais funcionalidade de EerCASE são:

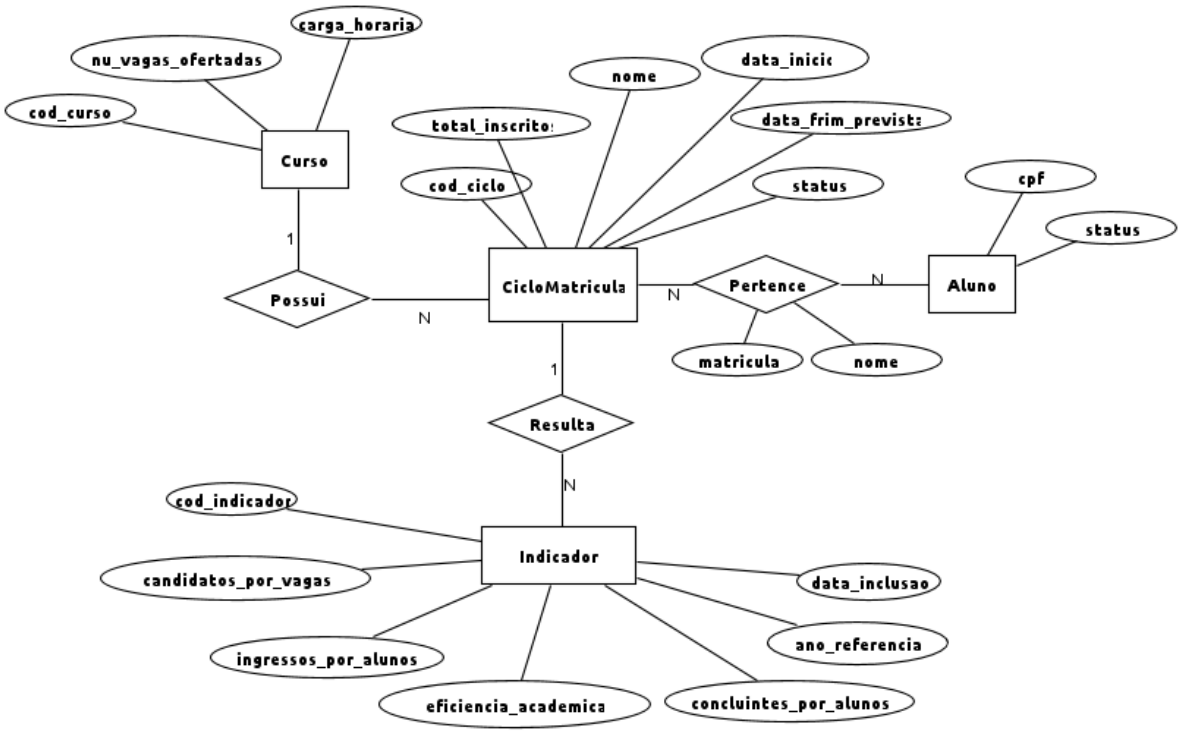
- Permite a criação e edição de esquemas conceituais E-R.
- Fornece suporte para geração de código e validação do esquema conceitual de dados de acordo com a notação clássica EER. Para o desenvolvimento deste estudo de caso, utilizou-se a versão 1.0.7.

A ferramenta EerCASE possibilitou a criação do esquema por meio dos construtores da notação clássica do modelo EER. A ferramenta possui funcionalidade de mapeamento do esquema conceitual EER para um *script* expresso pela linguagem de definição de dados (Data Definition Language - DDL), que é um dos vocabulários que compõem a linguagem SQL. Ao aplicar o teste para conversão do esquema conceitual criado, a ferramenta não conseguiu realizar devidamente a conversão e não exibiu o *script* gerado, apresentando somente a área de saída de texto em branco.

No tocante à representação da modelagem de dados agregados, identificou-se que não foi possível realizar esse tipo de modelagem na ferramenta EerCASE, pois ela objetiva a criação de esquemas conceituais segundo o modelo EER, e este não possui construtores que possibilitem a representação de dados agregados. Da mesma forma, a modelagem de esquemas conceituais de aplicações de persistência poliglota não pode ser realizada, pois a ferramenta não disponibiliza construtores de modelagem para essa finalidade. Por fim, no que condiz ao pleno funcionamento da ferramenta, verificou-se que ela não apresenta erros que prejudiquem a sua utilização.

A Figura 5.1 mostra o esquema conceitual criado com o auxílio da ferramenta EerCASE, na qual pode-se visualizar entidades, relacionamentos e atributos que condizem com o domínio de aplicação utilizado neste estudo de caso.

Figura 15 – Esquema conceitual modelado em EerCASE



Fonte: Elaborada pelo Autor (2017)

5.3.2 Execução na Ferramenta ERDPlus

A ferramenta ERDplus está disponível *online* por meio do endereço: https://erdplus.com. Essa ferramenta fornece as seguintes funcionalidades:

- Criação e edição de esquemas conceituais E-R.
- Criação e edição de esquemas relacionais de dados.
- Conversão de um esquema conceitual E-R em um esquema relacional.
- Exportação de esquemas relacionais para a linguagem SQL.

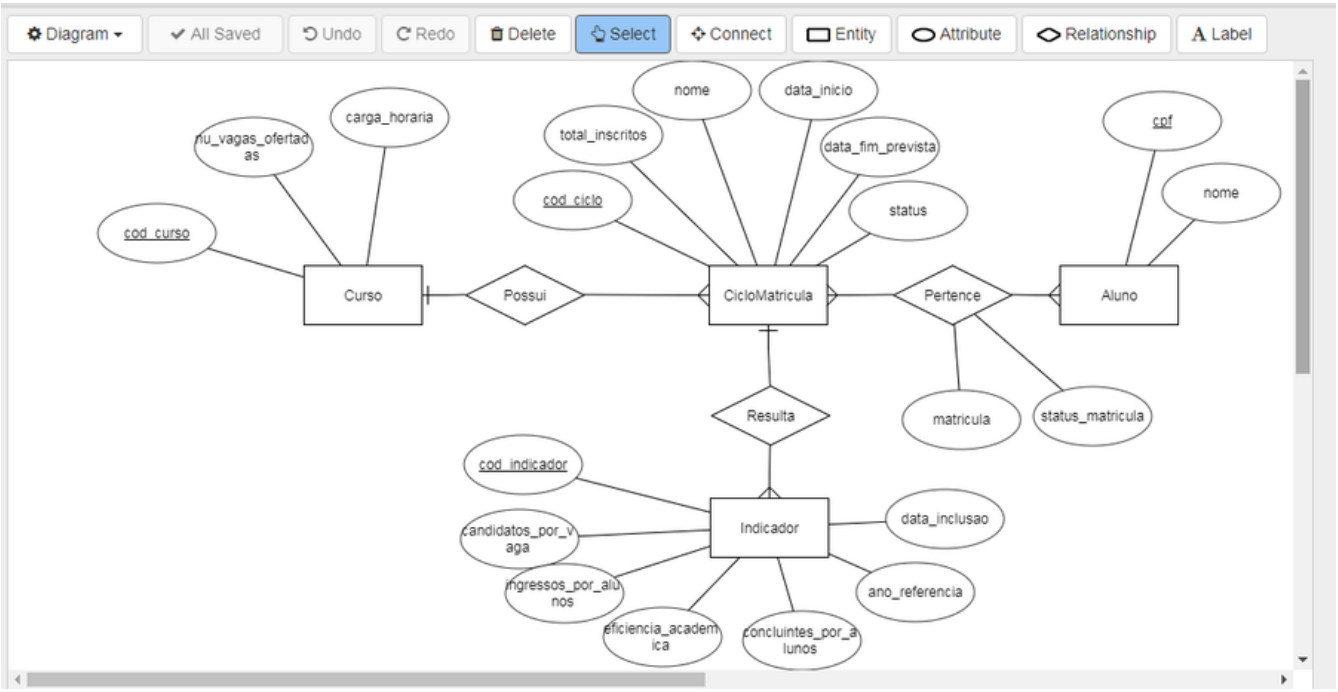
Neste estudo, utilizou-se a versão online de ERDPlus, acesso em: 21/02/2017. As etapas realizadas no *software* seguindo o planejamento do estudo de caso foram:

1) Criação do esquema conceitual E-R.
2) Conversão do esquema conceitual E-R para o esquema relacional.
3) Conversão do esquema relacional em código SQL.

4) Análise da possibilidade de modelagem baseada no conceito de dados agregados, além da verificação de alguma forma de modelagem de aplicações de persistência poliglota.

O uso da ferramenta ERDPlus foi iniciado com a construção do esquema conceitual E-R. Nesse sentido, a ferramenta atendeu ao propósito, possibilitando a criação do esquema de forma ágil e prática. A Figura 5.2 mostra a interface da ferramenta ERDPlus com a representação do esquema conceitual utilizado neste estudo de caso. Nela, pode-se visualizar entidades, relacionamentos e atributos modelados.

Figura 16 – Esquema conceitual modelado em ERDPlus



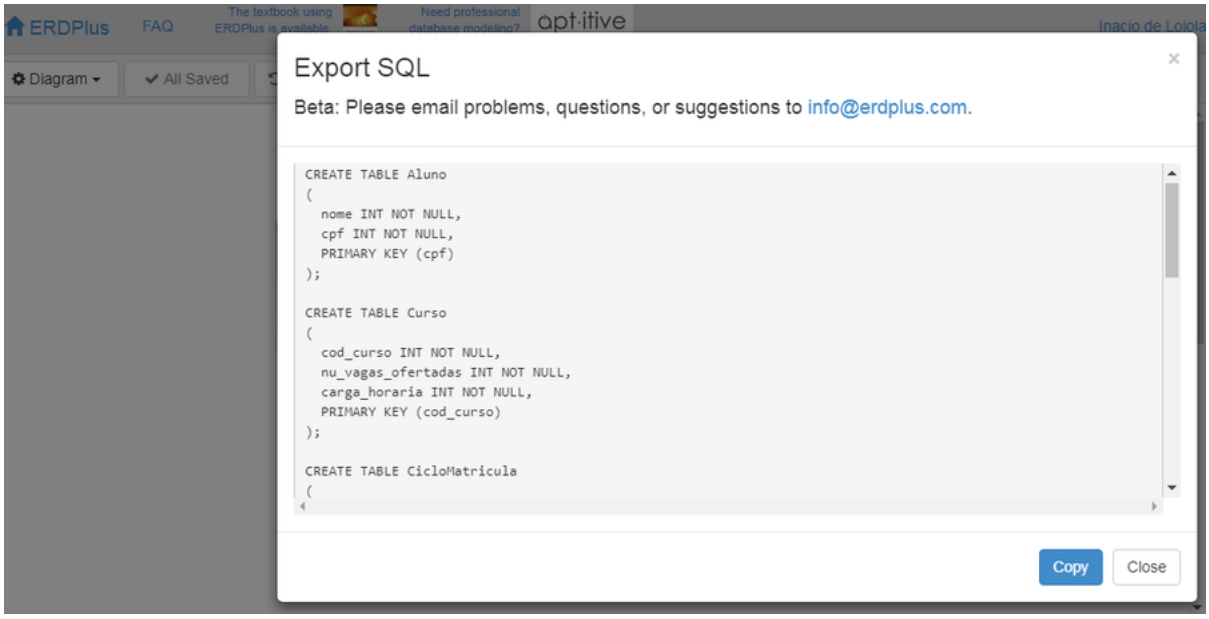
Fonte: Elaborada pelo Autor (2017)

No que se refere às funcionalidades de conversão do esquema conceitual para esquemas lógicos, realizou-se a utilização do mecanismo de conversão, disponível em ERDPlus, para geração de um esquema relacional. Na execução dessa funcionalidade, verificou-se que a ferramenta consegue realizar a conversão sem apresentar erros. A Figura 5.3 exibe o esquema relacional gerado pela ferramenta ERDPlus a partir do esquema conceitual (Figura 5.2). É importante observar que o esquema da Figura 5.2 é igual ao esquema conceitual gerado pela ferramenta EerCASE e mostrado na Figura 5.1.

Dando sequência aos procedimentos do protocolo (Seção 5.2.4), analisou-se a capacidade da ferramenta para conversão de esquemas lógicos. Em particular, para a

geração de um *script* expresso na linguagem de definição de dados DDL de SQL. A Figura 5.4 mostra a tela com o *script* DDL gerado pela ferramenta ERDPlus a partir do esquema relacional da Figura 5.3.

Figura 17 – Código DDL/SQL gerado pela ferramenta a partir do esquema relacional



Fonte: Elaborada pelo Autor (2017)

Finalizando o estudo com a ferramenta ERDPlus, investigou-se a sua capacidade de modelagem de dados agregados e a possibildade de representar aplicações de persistência poliglota. Verificou-se que a ferramenta não contempla essas formas de representação de dados. Por fim, durante a execução dos procedimentos deste estudo não foi encontrado qualquer erro na ferramenta ERDPlus que comprometa sua plena utilização.

5.3.3 Execução na Ferramenta NoSQLCASE

As seguintes etapas foram realizadas com a ferramenta NoSQLCASE seguindo o planejamento deste estudo:

1) Criação do esquema conceitual com base no conceito de persistência poliglota (Seção 2.1.3) e utilizando os construtores do modelo ERNoSQL (Capítulo 3).

2) Edição do esquema conceitual criado para representar dados agregados, usando os construtores de modelagem definidos no modelo ERNoSQL.

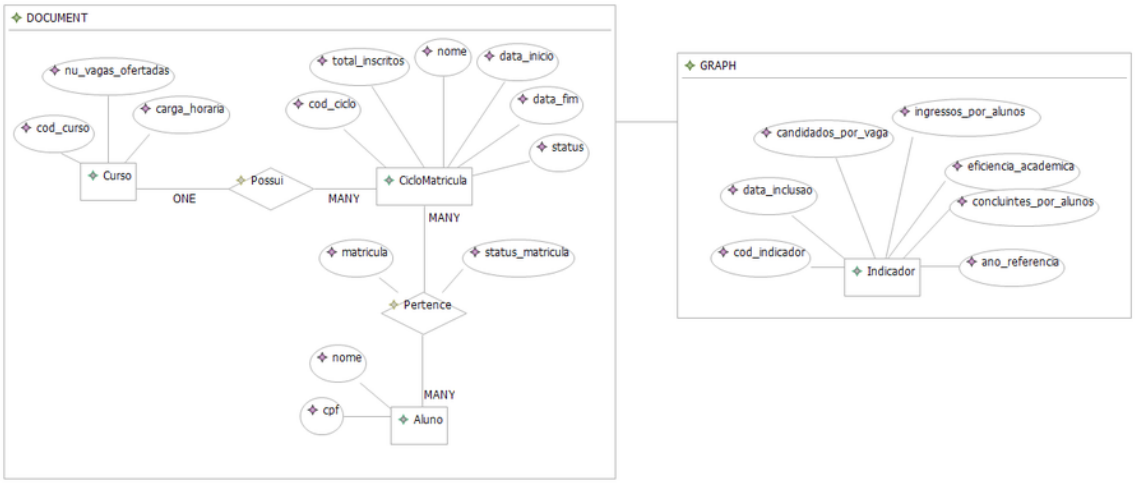
3) Conversão do esquema conceitual para o esquema lógico representado por um *script* expresso na linguagem de definição de dados do SGBD MongoDB. Além do uso da funcionalidade de conversão, também demonstra-se a execução

do *script* em uma instância do SGBD MongoDB, para mostrar a corretude do algoritmo de conversão implementado na ferramenta NoSQLCASE.

4) Conversão do esquema conceitual para um esquema lógico genérico, ou seja, indepente de um SGBD específico, representado no formato de um documento JSON. JSON é representado em formato de texto e é independente de linguagem, pois usa convenções que são familiares a maior parte das linguagens de programação, sendo um formato bastante utilizado na troca de dados entre aplicações.

A criação de esquemas conceituais poliglotas em NoSQLCASE está vinculada à adição de elementos NoSQLEsquema, como descrito na explicação do processo de utilização de NoSQLCASE (Seção 4.5). Neste estudo de caso, decidiu-se utilizar duas instâncias de *NoSQLEsquema*, sendo um deles relativo ao modelo de dados baseado em Documentos e o outro, relativo ao modelo baseado em Grafos. A Figura 5.5 exibe o diagrama que representa o esquema conceitual criado com o auxílio da ferramenta NoSQLCASE. Visualiza-se que a instância de *NoSQLEsquema* definida para o modelo de documentos representa os dados das entidades *Aluno*, *CicloMatrícula* e *Curso*, enquanto a instância de *NoSQLEsquema*, que referencia o modelo Grafos, representa os dados da entidade *Indicador*.

Figura 18 – Esquema Conceitual Poliglota criado em NoSQLCASE

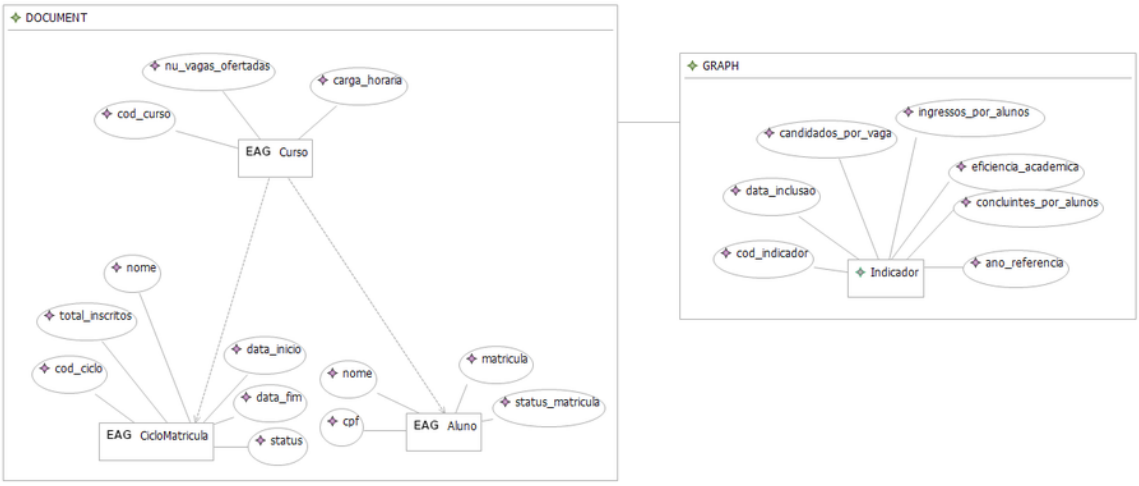


Fonte: Elaborada pelo Autor (2017)

A representação de dados agregados é possível em NoSQLCASE por meio do uso dos construtores *Entidade Agregada* e *Relacionamento de Entidades Agregadas* do modelo ERNoSQL (Capítulo 3). A Figura 5.6 exibe uma instância de *NoSQLEsquema* para o tipo de documentos contendo dados agregados. Visualiza-se que a entidade agregada *Curso* relaciona-se com as entidades *CicloMatricula* e *Aluno*. Esse

relacionamento entre entidades agregadas é visualizado pela notação da linha tracejada com uma seta indicando que uma entidade principal está agregando outra entidade, como descrito na Seção 3.5.

Figura 19 – Esquema Conceitual Poliglota com uso de Entidades Agregadas

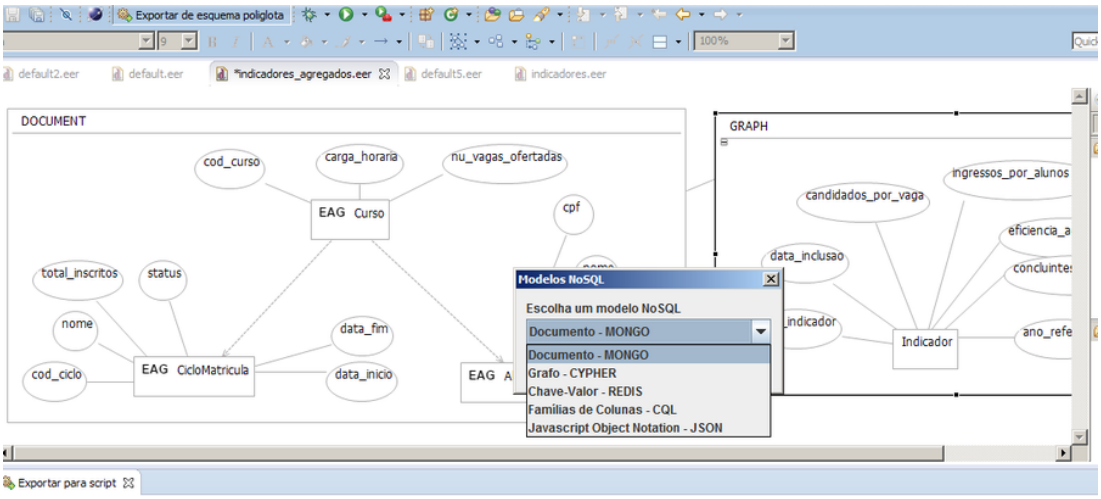


Fonte: Elaborada pelo Autor (2017)

5.3.3.1 Exportando Esquemas Conceituais com NoSQLCASE

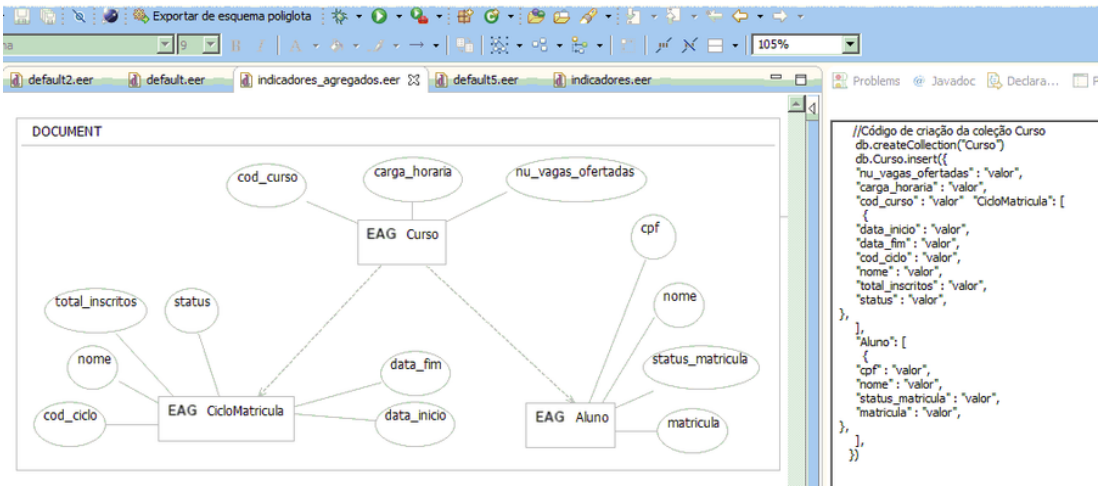
A utilização das funções de exportação dos esquemas conceituais está disponível no menu superior da ferramenta NoSQLCASE, por meio da opção *exportar esquema poliglota*. Após acessar a opção, é exibida uma caixa de seleção e o projetista seleciona o SGBD NoSQL para o qual deseja gerar o *script*, e clica no botão de confirmação. Após a confirmação, a ferramenta realiza a conversão e exibe o *script* na área de saída de texto. As Figuras 5.7 e 5.8, mostradas abaixo, exibem respectivamente a seleção do SGBD (Figura 5.7) para o qual se deseja realizar a conversão e, após a conversão, é exibido o *script* (Figura 5.8) na área de saída de texto da ferramenta.

Figura 20 – Ilustração da funcionalidade de exportação em NoSQLCASE



Fonte: Elaborada pelo Autor (2017)

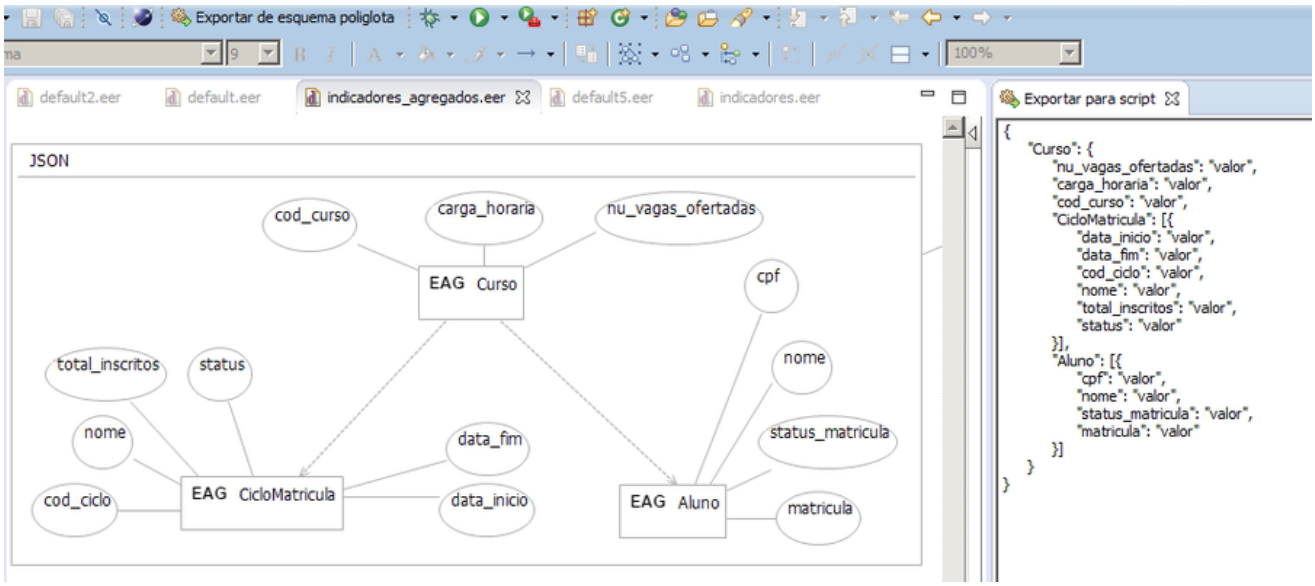
Figura 21 – Código gerado por NoSQLCASE para o SGBD MongoDB



Fonte: Elaborada pelo Autor (2017)

Para realizar a conversão do esquema conceitual para um formato JSON na ferramenta NoSQLCASE, o projetista deve utilizar a funcionalidade de exportação da ferramenta, selecionando a opção de conversão para *Javascript Object Notation - JSON*. A Figura 5.7 mostra a tela de NoSQLCASE, onde visualiza-se na área de saída de texto, o código do documento JSON gerado pela conversão do esquema conceitual criado para o estudo de caso.

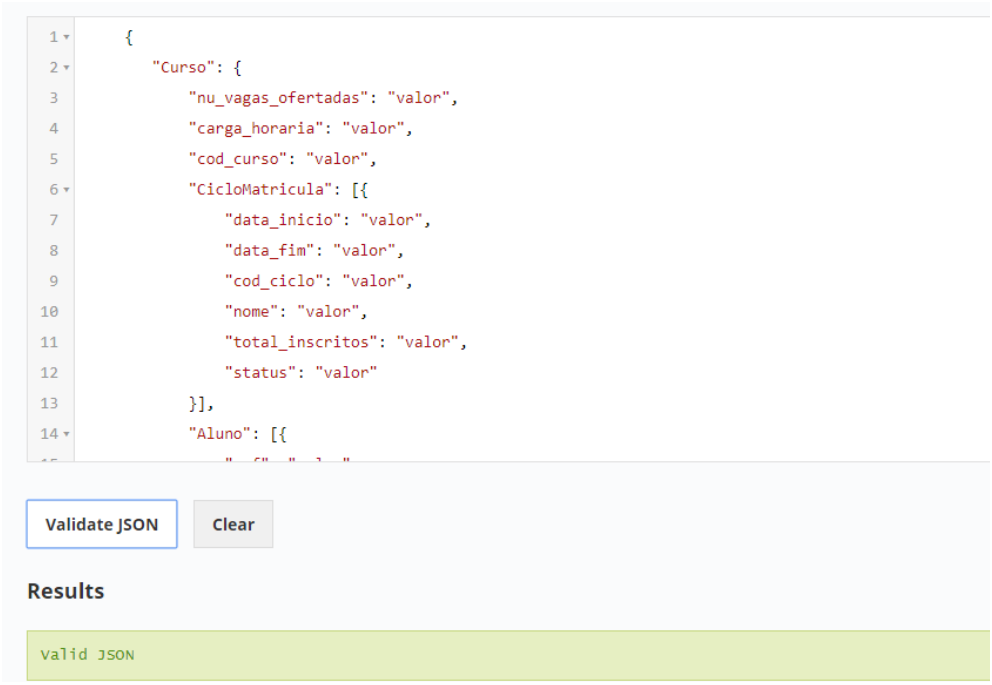
Figura 22 – Código gerado por NoSQLCASE no formato JSON



Fonte: Elaborada pelo Autor (2017)

Para verificar a validade da representação no formato JSON produzida por NoSQLCASE, realizou-se o acesso ao sítio https://jsonlint.com/, que fornece a funcionalidade de validar a sintaxe de documentos JSON. A Figura 5.10 exibe a imagem da página do sítio, na qual observa-se que o código JSON verificado obteve resultado positivo, de acordo com a mensagem indicativa gerada de que o documento é válido.

Figura 23 – Demonstração da validade do documento JSON gerado em NoSQLCASE

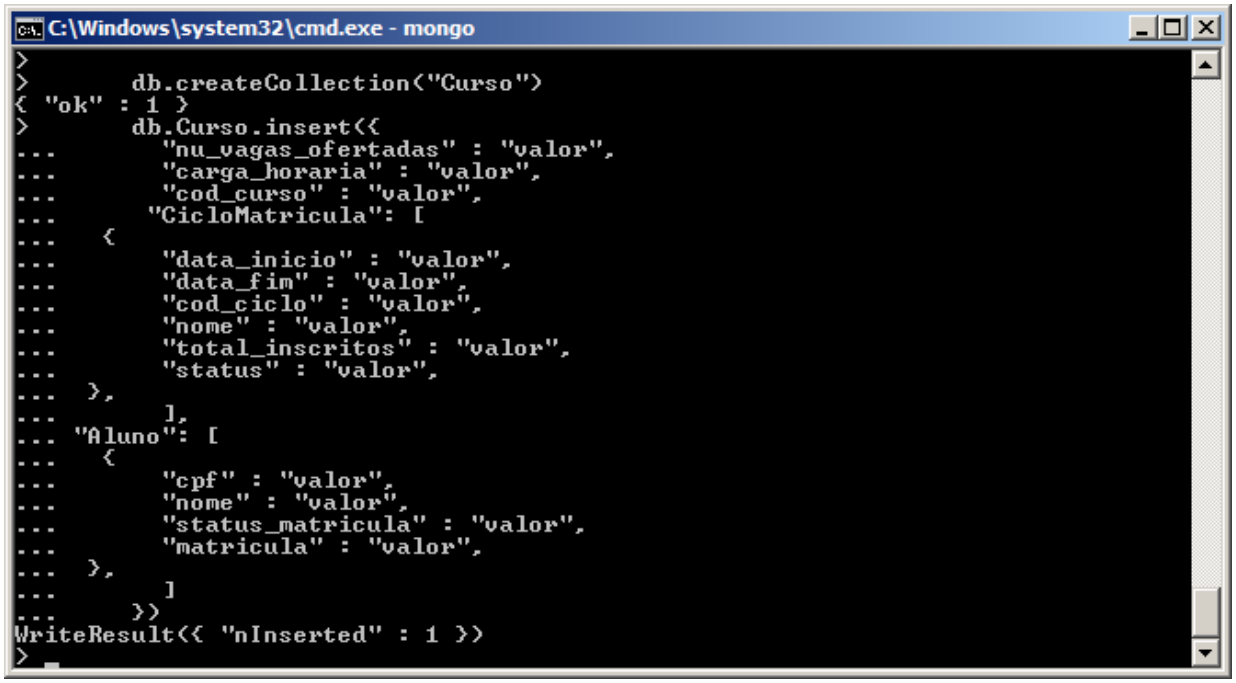


Fonte: https://jsonlint.com/

5.3.3.2 Implementação do Esquema Lógico gerado por NoSQLCASE

O código produzido pela ferramenta NoSQLCASE exibido na Figura 5.6 foi executado no SGBD MongoDB. A execução foi realizada em uma instância do SGBD MongoDB criada no computador especificado na Seção 5.2.4. O código foi copiado da área de saída de texto da ferramenta NoSQLCASE diretamente para o aplicativo de gerenciamento do SGBD MongoDB, sendo executado por meio do aplicativo de prompt de comando do sistema operacional Windows. A Figura 5.11 exibe a tela do aplicativo com os códigos executados. Como pode ser visto na terceira linha, obteve-se a resposta *"ok : 1"* para a execução do comando *db.createCollection("Curso")*, que cria uma nova coleção com o nome *Curso* no SGBD. As próximas linhas exibidas na Figura 5.11 são referentes à estrutura do documento que será armazenado no banco de dados. Na penúltima linha pode-se verificar a mensagem *"WriteResult({ "nInserted" : 1 })"*, a qual significa que um registro foi inserido com sucesso no banco de dados.

Figura 24 – Execução do script gerado por NoSQLCASE no SGBD MongoDB



Fonte: Elaborada pelo Autor (2017)

Por conseguinte, como forma de complementar o estudo de caso, realizou-se a execução do script alterando o valor dos atributos por dados fictícios. Foram adicionados dez registros no banco de dados e, em sequência, executou-se o comando *db.Curso.find()* que realiza a busca por todos os documentos armazenados na coleção *Curso* do banco de dados. A consulta foi executada com sucesso e sua resposta está disponível no Apêndice B desta dissertação.

5.4 Análise e Resultados

Os resultados gerados pelo estudo de caso descrito nesta dissertação foram analisados de acordo com as questões norteadoras (Seção 5.2.2) definidas na fase de planejamento. O Quadro 3 mostra cada questão e compara a sua viabilidade para cada caso aplicado. Verifica-se no quadro que em relação à possibilidade de modelagem de aplicações de persistência poliglota, somente a ferramenta NoSQLCASE fornece esse suporte. No tocante à segunda questão que analisa a perspectiva de modelagem de dados agregados, foi observado que as ferramentas EerCASE e EDRPlus não fornecem esse apoio, enquanto a ferramenta NoSQLCASE possibilita esse auxílio por meio do uso dos construtores propostos no modelo ERNoSQL (Capítulo 3). No que se refere ao intuito principal da terceira questão em verificar a viabilidade de conversão do esquema conceitual para um esquema lógico de SGBD NoSQL, conclui-se que a ferramenta NoSQLCASE permitiu a correta conversão do esquema conceitual criado para um esquema lógico do SGBD MongoDB. Ademais, avaliou-se também a utilidade de NoSQLCASE na conversão do esquema conceitual para um documento no formato JSON, que é considerado um esquema lógico genérico para BD NoSQL.

Quadro 3 – Quadro demonstrativo da análise das questões do estudo para cada caso aplicado

Questão	EerCASE	ERDPlus	NoSQLCASE
Questão 1: permite a modelagem conceitual de dados para domínios de aplicações de persistência poliglota?	Não	Não	Sim
Questão 2: permite a modelagem de dados baseada no conceito de dados agregados?	Não	Não	Sim
Questão 3: permite o mapeamento do esquema conceitual criado para esquemas lógicos de sistemas NoSQL?	Não	Não	Sim

Fonte: Elaborada pelo Autor (2017)

Os resultados encontrados no presente estudo sugerem que, em relação às necessidades de suporte à modelagem de aplicações de persitência poliglota e à representação do conceito de dados agregados para o projeto de bancos de dados NoSQL, verificou-se que a ferramenta NoSQLCASE foi a única que possibilitou esse suporte.

O modelo ERNoSQL e a ferramenta NoSQLCASE representam um avanço para área de pesquisa sobre projeto conceitual de bancos de dados NoSQL e aplicações de persistência poliglota, tendo em vista que as ferramentas de suporte à modelagem conceitual de bancos de dados existentes, representadas nesse estudo por EerCASE e ERDPlus, não fornecem construtores de modelagem que auxiliem o projetista na construção de esquemas conceituais de dados de acordo com as necessidades atuais, que envolvem aplicações que utilizam diversos modelos e tipos de dados, projetadas no intuito de fornecer a capacidade para manipular vastos volumes de dados, sejam eles estruturados, semiestruturados ou não estruturados.

5.5 Considerações Finais do Capítulo

Este capítulo apresentou um estudo de caso, que promoveu a construção de um esquema conceitual de dados nas ferramentas EerCASE, ERDPlus e NoSQLCASE. Verificou-se que a ferramenta NoSQLCASE se diferencia das demais pelo fato de possibilitar a modelagem de esquemas conceituais de dados para aplicações de persistência poliglota. Além disso, NoSQLCASE permite o uso dos construtores do modelo ErNoSQL que representam dados semiestruturados e não estruturados, e também possibilita a modelagem baseada no conceito de dados agregados que é bastante utilizada no projeto de BD NoSQL. Outro ponto que diferencia NoSQLCASE das demais ferramentas é o fato de possibilitar a exportação de um esquema conceitual para um *script* de criação de um banco de dados NoSQL. Como foi demonstrado, o *script* gerado em NoSQLCASE referente ao esquema conceitual modelado no estudo de caso foi executado com sucesso numa instância do SGBD MongoDB. Isso demonstra que a ferramenta NoSQLCASE facilita desde o projeto de modelagem de bancos de dados NoSQL até a própria implementação física do banco de dados.

6 CONCLUSÃO

Este capítulo tem como objetivo apresentar as considerações finais sobre os principais tópicos abordados nesta dissertação, incluindo as principais contribuições alcançadas, as indicações de trabalhos futuros e as limitações do trabalho.

6.1 Considerações Finais

Diversos trabalhos envolvendo a modelagem de bancos de dados NoSQL tanto propõem novos modelos conceituais quanto utilizam modelos existentes para auxiliar o projeto de BD NoSQL. Porém, pouca atenção tem sido dada à investigação de uma abordagem de modelagem de dados que forneça ao projetista, os construtores de modelagem capazes de criar um esquema de dados conceitual e poliglota.

Este trabalho propôs um modelo conceitual de dados, chamado ERNoSQL, o qual visa fornecer ao projetista de banco de dados, a possibilidade de criar um esquema conceitual de dados para aplicações de persistência poliglota, ou seja, que utilizem diversos modelos de dados estruturados, semiestruturados e não estruturados em uma mesma aplicação. A modelagem de um domínio de aplicação de persistência poliglota por meio do ERNoSQL permite uma visão unificada dos dados e tipos de sistemas NoSQL utilizados na aplicação, além de facilitar a comunicação entre a equipe técnica e os usuários finais. Para ilustrar os construtores do ERNoSQL e como eles se relacionam, este trabalho propôs ainda um metamodelo em notação UML. Esse metamodelo foi utilizado como artefato principal para a construção de uma ferramenta CASE, desenvolvida por meio do *framework* de modelagem Eclipse que implementa um editor gráfico a partir da especificação de um metamodelo. Por fim, uma ferramenta de modelagem, chamada NoSQLCASE, foi desenvolvida para dar suporte aos construtores de modelagem do ERNoSQL e para auxiliar o projetista nas atividades de modelagem de bancos de dados para aplicações de persistência poliglota.

O Instituto Federal de Educação, Ciência e Tecnologia do Ceará e as demais instituições da Rede Federal de Educação Profissional, Cientifíca e Tecnólogica podem utilizar a Ferramenta NoSQLCASE para auxiliar a modelagem conceitual de bancos de dados para aplicações de persistência poliglota desenvolvidas para atender as necessidades acadêmicas e administrativas destas instituições. Ademais, o autor deste trabalho adquiriu conhecimento sobre diversas tecnologias emergentes e se dispõe a realizar treinamentos ou capacitações dos servidores de Tecnologia da Informação no que condiz a utilização destas tecnologias.

O restante deste capítulo está organizado como segue. Na Seção 6.2 são abordadas as principais contribuições obtidas pelo desenvolvimento deste trabalho, e na Seção 6.3 são descritos os trabalhos futuros que podem ser realizados para dar continuidade ao projeto de pesquisa iniciado nesta dissertação.

6.2 Principais Contribuições

As principais contribuições deste trabalho são detalhadas a seguir de acordo com os itens elencados como objetivos específicos (Seção 1.2.2).

I. Especificação dos construtores de modelagem com suas respectivas propriedades e notação gráfica.

Neste trabalho, investigou-se a aplicação do conceito de persistência poliglota na construção de um modelo conceitual de dados. A pesquisa iniciou-se com o levantamento bibliográfico dos modelos existentes para projetos de BD NoSQL. Então, verificou-se que nenhum dos modelos estudados possibilita a modelagem de aplicações de persistência poliglota. A partir deste ponto, começou o desafio de especificar construtores de modelagem que pudessem representar a persistência poliglota. Após estudos, verificou-se que o ideal seria criar um modelo que estenda o modelo E-R, adicionando construtores específicos para modelagem de persistência poliglota. Utilizou-se um domínio hipotético de empresa de comércio eletrônico para analisar o uso dos construtores de modelagem e como eles se relacionam.

II. Regras de mapeamento para esquemas lógicos de BD NoSQL a partir do esquema conceitual ERNoSQL.

Realizou-se a comparação dos construtores do modelo ERNoSQL com os diferentes formatos de armazenamento equivalentes de tipos de SGBD NoSQL. Foram definidas regras de mapeamento para cada um dos tipos de sistemas NoSQL existentes. Essas regras são utilizadas na conversão do esquema conceitual ERNoSQL para esquemas lógicos expressos em *scripts* de linguagens de SGBD NoSQL.

III. Especificação do Metamodelo de ERNoSQL.

Um metamodelo especificado na notação UML foi criado para representar os conceitos do modelo conceitual ERNoSQL. Além disso, o metamodelo serve como artefato principal para a implementação da ferramenta CASE, pois define os construtores ERNoSQL e como eles se relacionam.

IV. Desenvolvimento de uma ferramenta CASE baseada na notação gráfica dos construtores do ERNoSQL

Para auxiliar as atividades de modelagem do projetista de banco de dados, e no intuito de fornecer uma ferramenta computacional que dê suporte aos construtores

de modelagem propostos no ERNoSQL, foi desenvolvida uma ferramenta de modelagem, chamada NoSQLCASE. Dentre as suas principais funcionalidades, NoSQLCASE permite a criação de esquemas conceituais de aplicações de persistência poliglota e possibilita a geração do esquema lógico expresso em *scripts* de criação dos BD de SGBD NoSQL. Além disso, ela permite a conversão para JSON, que é um formato de documento bastante utilizado para transferência de dados entre aplicações, sendo independente de uma tecnologia específica, ou seja, diversas linguagens de programação podem ler ou criar esses documentos.

V. Exemplo da aplicação da ferramenta CASE por meio da construção de esquemas conceituais de dados, considerando um cenário real de uma organização acadêmica. Foi feita a modelagem deste cenário em NoSQLCASE e em outras ferramentas para demonstrar os diferenciais da ferramenta proposta.

Foi realizado um estudo de caso que compara a ferramenta NoSQLCASe com outras ferramentas de modelagem. Esse estudo demonstrou que somente NoSQLCASE permite a modelagem de aplicações que utilizem diversos modelos de dados. Além disso, também foi demonstrado a execução dos *scripts* gerados por NoSQLCASE para o SGBD MongoDB, no qual foi executado o script e exibido o resultado que ilustra a corretude dos algoritmos de conversões implementados na ferramenta.

6.3 Trabalhos Futuros

A persistência poliglota tende a ser cada vez mais utilizada devido à variedade de tipos de dados e necessidades de manipulação (consultas, análises e processamento de dados volumosos) que são demandadas pelos sistemas de informações desenvolvidos para Internet, além do crescente uso de dados complexos (não estruturados e semiestruturados) oriundos de diversas fontes como mídias sociais, redes de sensores e dispositivos conectados. Esses dispositivos conectados, também nomeados como Internet das Coisas (do inglês, Internet of Things), estão presentes no dia a dia das pessoas, como por exemplo nos relógios inteligentes que armazenam dados coletados por sensores e sincronizam com uma aplicação web, de forma a monitorar a saúde do usuário e enviar notificações e alertas para um aplicativo móvel.

Diante do trabalho exposto nesta dissertação e da tendência crescente da persistência poliglota, tem-se algumas propostas de contribuições futuras para o avanço científico no projeto de aplicações de persistência poliglota que utilizem bancos de dados NoSQL:

- Conversão de um esquema por meio ERNoSQL em uma instância física de BD NoSQL. Essa conversão pode ser realizada através do refinamento das regras de mapeamento propostas neste trabalho (Seção 3.7), incluindo detalhes

específicos de um determinado SGBD NoSQL, e pode ainda incorporar as funções de criação da instância física na ferramenta NoSQLCASE, investigando formas de conectar esta ferramenta com instâncias de SGBD NoSQL.

- Criação de um assistente para elaboração de uma interface de programação de aplicativos (do inglês, API - Application Programming Interface). A API deve abstrair o acesso direto ao banco de dados e fornecer serviços que serão consumidos por sistemas de informação, como, por exemplo para inserir, manter e recuperar dados. Muitos SGBD NoSQL fornecem API web de serviços com operações básicas para inserir, remover e atualizar dados, mas no caso da proposta apresentada, o assistente para criação da API deve auxiliar na criação de operações que forneçam serviços específicos ao domínio de aplicação do esquema conceitual ERNoSQL. Um exemplo seria: dado um esquema conceitual ERNoSQL referente a uma aplicação de comércio eletrônico, após a implementação física em um SGBD NoSQL pode-se necessitar da criação de um serviço para consultar produtos que estão há muito tempo no estoque, como também a utilização de consultas para sugestões de compras, entre outros.
- Utilizar as regras de mapeamento definidas nesta dissertação como princípio para a implementação de algoritmos de conversão de um esquema de dados conceitual ERNoSQL para esquemas de dados lógicos de sistemas de BD NoSQL multi-modelos.
- Uso de técnicas de aprendizagem de máquina para escolha automática do tipo de SGBD NoSQL na geração do esquema lógico.

6.4 Limitações do Trabalho

Este trabalho também possui algumas limitações, as quais são descritas a seguir:

- A ferramenta NoSQLCASE não possui interface web, então o usuário sempre deverá instalar a ferramenta na máquina para utiliza-la.
- A usabilidade da ferramenta não foi avaliada nesta pesquisa, mas pode ser alvo de avaliação e melhorias.
- O usuário precisa conhecer o modelo ERNoSQL para utilizar a ferramenta, então é interessante adicionar uma opção de ajuda ou um guia incial na primeira utilização da ferramenta NoSQLCASe.

REFERÊNCIAS

ARAÚJO, A. M. C. de. *ArcheER*: Um Modelo Conceitual de Dados Arquetipados para Sistemas de Informação em Saúde. 2012. 112 p. Dissertação (Pós-graduação em ciência da computação) — Universidade Federal de Pernambuco, Recife.

BANERJEE, S. et al. Towards Logical Level Design of Big Data. In: *IEEE 13th International Conference on Industrial Informatics (INDIN)*. [S.l.: s.n.], 2015. p. 1665 – 1671.

BUGIOTTI, F. et al. Database Design for NoSQL Systems. In: *International Conference on Conceptual Modeling*. [S.l.: s.n.], 2014. p. 223 – 231.

ELMASRI, R.; NAVATHE, S. B. *Sistemas de Banco de Dados*. 6. ed. São Paulo: Pearson Education, 2010.

JEON, J.; AN, M.; LEE, H. NoSQL Database Modeling for End-of-Life Vehicle Monitoring System. *Journal of Software*, v. 10, n. 10, p. 1160 – 1169, Outubro 2015.

KAUR, K.; RANI, R. Modeling and Querying Data in NoSQL Databases. In: *2013 IEEE International Conference on Big Data*. [S.l.: s.n.], 2013.

KOLOVOS, D. et al. *The Epsilon Book*. Eclipse Public Licence, 2017. Disponível em: <http://www.eclipse.org/epsilon/doc/book/>. Acesso em: 03/05/2017.

LIMA, C. de; MELLO, R. dos S. A Workload-Driven Logical Design Approach for NoSQL Document Databases. In: *iiWAS '15*: Proceedings of the 17th international conference on information integration and web-based applications & services. [S.l.: s.n.], 2015.

MATHEW, A. B.; KUMAR, S. D. M. Analysis of Data Management and Query Handling in Social Networks using NoSQL Databases . In: IEEE, 2015. *2015 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*. [S.l.], 2015. p. 800 – 806.

OLIVEIRA, F. R.; CURA, L. del V. Performance Evaluation of NoSQL Multi-Model Data Stores in Polyglot Persistence Applications. In: ACM (Ed.). *Proceedings of the 20th International Database Engineering & Applications Symposium*. [S.l.: s.n.], 2016. p. 230 – 235.

SADALAGE, P. J.; FOWLER, M. *NoSQL Distilled - A brief guide to the emerging world of polyglot persistence*. 1. ed. [S.l.]: Pearson Education, 2013.

SINGH, M.; KAUR, K. SQL2Neo : Moving Health-care Data From Relational To Graph Databases. In: IEEE, 2015. *2015 IEEE International Advance Computing Conference (IACC)*. [S.l.], 2015. p. 721 – 725.

SOUZA, C. C. N. *Um Metamodelo e uma Ferramenta CASE para Projeto Conceitual de Banco de Dados segundo o Modelo ER*. 2011. 78 p. Dissertação (Mestrado em Ciência da Computação) — Universidade Federal de Pernambuco, Recife.

SRIVASTAVA, P. P.; GOYAL, S.; KUMAR, A. Analysis of Various NoSql Database . p. 539 – 544, 2015.

VARA, J. M.; MARCOS, E. A framework for model-driven development of information systems: Technical decisions and lessons learned. *Journal of Systems and Software*, v. 85, n. 10, p. 2368 – 2384, Outubro 2012. ISSN 01641212. Disponível em: <http://linkinghub.elsevier.com/retrieve/pii/S0164121212001367>.

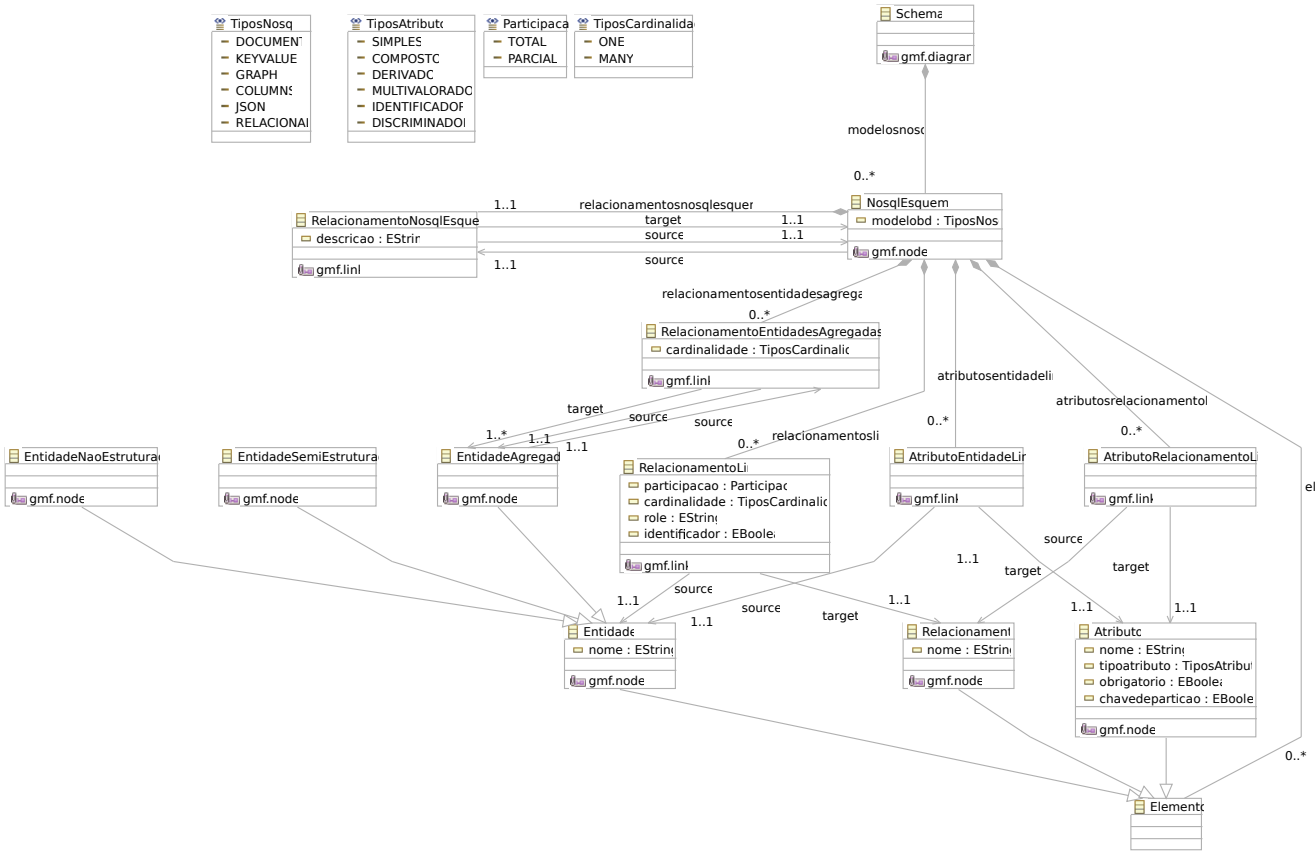
VILLARI, M. et al. Enriched E-R Model to Design Hybrid Database for Big Data Solutions. In: *2016 IEEE Symposium on Computers and Communication (ISCC)*. [S.l.: s.n.], 2016. p. 163 – 166.

WOHLIN, C. et al. *Experimentation in Software Engineering*. [S.l.]: Springer, 2012. ISBN 978-3-642-29043-5.

APÊNDICES

Apêndice A - Metamodelo ERNoSQL

Figura 25 - Metamodelo ERNoSQL



Fonte: Elaborada pelo Autor (2017)

Apêndice B - Demonstração da execução de consultas realizadas no SGBD MongoDB para o banco de dados criado no estudo de caso a partir do script gerado pela ferramenta NoSQLCASE

Figura 26 – Resultado da consulta ao banco de dados MongoDB implementado no estudo de caso

Fonte: Elaborada pelo Autor (2017)

Apêndice C - Algoritmos de conversão implementados em NoSQLCASE expressos na linguagem EGL

1 - ERNoSQL para JSON

```
[%for (schema in Schema) {%]
[%for (nosqlesquema in schema.modelosnosql) {%]
[% if (nosqlesquema.modelobd.value == 4){ %]
[%var elem = null;%]
[%var rel = null;%]
[%for (elemento in nosqlesquema.elementos) {%]
[%if ((elemento.type.name == “Entidade” or elemento.type.name
== “EntidadeNaoEstruturada” or elemento.type.name == “EntidadeSemiEstruturada” ))
{%]
[%elem = elemento;%]
{
“[%=elemento.nome%]”: {
[%for (atributo in nosqlesquema.atributosentidadelink) {%]
[%if (elemento == atributo.source) {%]
“[%=atributo.target.nome%]” : “valor”
[%if (atributo <> nosqlesquema.atributosentidadelink.last()){%],
[%}%]
[%}%]
[%}%]
}
[%} else if (elemento.type.name == “EntidadeAgregada”) {%]
[%if (rel == null) {%]
{
“[%=elemento.nome%]”: {
[%for (atributo in nosqlesquema.atributosentidadelink) {%]
[%if (elemento == atributo.source) {%]
“[%=atributo.target.nome%]” : “valor”[%if (atributo <>
nosqlesquema.atributosentidadelink.last()){%],
[%}%]
[%}%]
[%}%]
[%for (relacionamentoagregacao in nosqlesquema.
```

```
        relacionamentosentidadesagregadas) {%]
[%rel = relacionamentoagregacao.target[0].nome;%]
[%if (relacionamentoagregacao.cardinalidade.value == 1) {%]
“[%=relacionamentoagregacao.target[0].nome%]”: [
{
[%for (atributo in nosqlesquema.atributosentidadelink) {%]
[%if (relacionamentoagregacao.target[0] == atributo.source) {%]
“[%=atributo.target.nome%]” : “valor”[%if (atributo <>
        nosqlesquema.atributosentidadelink.last()){%],
[%}%]
[%}%]
[%}%] },
],
[%} else { %]    “[%=relacionamentoagregacao.target[0].nome%]”: {
[%for (atributo in nosqlesquema.atributosentidadelink) {%]
[%if (relacionamentoagregacao.target[0] == atributo.source) {%]
“[%=atributo.target.nome%]” : “valor”[%if (atributo <>
        nosqlesquema.atributosentidadelink.last()){%],
[%}%][%}%][%}%] },
[%}%][%}%]
[%}%]
[%}%]
[%}%]
[%}%]
}
[%}%]
[%}%]
```

2 - ERNoSQL para SGBD MongoDB

```
        [%for (schema in Schema) {%]
[%for (nosqlesquema in schema.modelosnosql) {%]
[% if (nosqlesquema.modelobd.value == 0){ %]
[%var elem = null;%]
[%var rel = null;%]
[%for (elemento in nosqlesquema.elementos) {%]
[%if ((elemento.type.name == “Entidade” or elemento.type.name ==
        “EntidadeNaoEstruturada” or elemento.type.name == “EntidadeSemiEstrutu-
rada”)) {%]
[%elem = elemento;%]
```

```
//Código de criação da coleção [%=elemento.nome%]
db.createCollection("[%=elemento.nome%]")
db.[%=elemento.nome%].insert({
[%for (atributo in nosqlesquema.atributosentidadelink) {%]
[%if (elemento == atributo.source) {%]
[%=atributo.target.nome%] : "valor"[%if (atributo <>
        nosqlesquema.atributosentidadelink.last()){%],
[%}%]
[%}%]
[%}%]
[%}else if ((elemento.type.name == "Relacionamento")) {%]
[%var a = 0;
var entidadeid = "";%]
[%for (relacionamentolink in nosqlesquema.relacionamentoslink) {%]
[%if (elemento == relacionamentolink.target){a = a+1;%]
[%for (atributo in nosqlesquema.atributosentidadelink) {%]
[%elem = elemento;%]
[%if (relacionamentolink.source == atributo.source)  {%]
[%if (a==2){%] [%=elemento.nome%]:
        [{[%for (atributo in nosqlesquema.atributosrelacionamentolink)
        {%][%if (elemento == atributo.source)
        {%][%=atributo.target.nome%]: 'valor',
        [%}%][%}%][%=atributo.source.nome%]_[%=atributo.target.nome%]:
        [%=atributo.source.nome%].[%=atributo.target.nome%]}],
[%}%][%}%][%}%][%}%][%}%]
[%} else if (elemento.type.name == "EntidadeAgregada") {%]
[%if (rel == null) {%]
//Código de criação da coleção [%=elemento.nome%]
db.createCollection("[%=elemento.nome%]")
db.[%=elemento.nome%].insert({
[%for (atributo in nosqlesquema.atributosentidadelink) {%]
[%if (elemento == atributo.source) {%]
"[%=atributo.target.nome%]" : "valor"
        [%if (atributo <> nosqlesquema.atributosentidadelink.last()){%],
[%}%]
[%}%]
[%}%]
```

```
[%for (relacionamentoagregacao in nosqlesquema.relacionamentosentidadesagregadas)
{%]
[%rel = relacionamentoagregacao.target[0].nome;%]
[%if (relacionamentoagregacao.cardinalidade.value == 1) {%]
“[%=relacionamentoagregacao.target[0].nome%]”: [
{
[%for (atributo in nosqlesquema.atributosentidadelink) {%]
[%if (relacionamentoagregacao.target[0] == atributo.source) {%]
“[%=atributo.target.nome%]” : “valor”
        [%if (atributo <> nosqlesquema.atributosentidadelink.last()){%],
[%}%]
[%}%]
[%}%] },
],
[%} else { %]    “[%=relacionamentoagregacao.target[0].nome%]”: {
[%for (atributo in nosqlesquema.atributosentidadelink) {%]
[%if (relacionamentoagregacao.target[0] == atributo.source) {%]
“[%=atributo.target.nome%]” : “valor”
        [%if (atributo <> nosqlesquema.atributosentidadelink.last()){%],
[%}%][%}%][%}%] },
[%}%][%}%]
})
[%}%]
[%}%]
[%}%]
[%}%]
[%}%]
[%}%]
```

3 - ERNoSQL para SGBD Cassandra

```
        [%for (schema in Schema) {%]
[%for (nosqlesquema in schema.modelosnosql) {%]
[%var rel = null;%]
[% if (nosqlesquema.modelobd.value == 3){ %]
[%for (elemento in nosqlesquema.elementos) {%]
[%if ((elemento.type.name == “Entidade” or elemento.type.name ==
        “EntidadeNaoEstruturada” or elemento.type.name == “EntidadeSemiestrutu-
```

```
rada”)){%]
//Código CQL de criação da tabela [%=elemento.nome%]
CREATE TABLE [%=elemento.nome%](
[%for (atributo in nosqlesquema.atributosentidadelink) {%]
[%if (elemento == atributo.source) {%]
ADD COLUMN  [%=atributo.target.nome%] datatype,
[%}%]
[%}%]
PRIMARY KEY ([%for (atributo in nosqlesquema.atributosentidadelink) {%]
[%if (elemento == atributo.source and atributo.target.tipoatributo.value == 4){%]
[%if (atributo.target.chavedeparticao = true){%]
([%=atributo.target.nome%]),[%}else {%][%=atributo.target.nome%],[%}%][%}%][%}%]));
[%} else if (elemento.type.name == “EntidadeAgregada”) {%]
[%if (rel == null) {%]
//Código de criação da tabela agregada [%=elemento.nome%]
CREATE TABLE [%=elemento.nome%]
(
[%for (atributo in nosqlesquema.atributosentidadelink) {%]
[%if (elemento == atributo.source) {%]
ADD COLUMN  [%=atributo.target.nome%] datatype,
[%}%]
[%}%]
[%for (relacionamentoagregacao in nosqlesquema.relacionamentosentidadesagregadas)
{%]
[%rel = relacionamentoagregacao.target[0].nome;%]
[%for (atributo in nosqlesquema.atributosentidadelink) {%]
ADD COLUMN [%=relacionamentoagregacao.target[0].nome%].
        [%=atributo.target.nome%] datatype
        [%if (atributo <> nosqlesquema.atributosentidadelink.last()){%],
[%}%]
[%}%]
[%}%]
);
[%}%] [%}%] [%}%] [%}%] [%}%] [%}%]
```

4 - ERNoSQL para SGBD Neo4J

```
        //Código Cypher de criação dos nós (Entidades)
[%for (schema in Schema) {%]
[%for (nosqlesquema in schema.modelosnosql) {%]
```

```
[% if (nosqlesquema.modelobd.value == 2){ %]
[%for (elemento in nosqlesquema.elementos) {%]
[%if ((elemento.type.name == “Entidade” or elemento.type.name ==
        “EntidadeNaoEstruturada” or elemento.type.name == “EntidadeSemiEstruturada”
or
        elemento.type.name == “EntidadeAgregada”)){%]
[%for (atributo in nosqlesquema.atributosentidadelink) {%]
[%if (elemento == atributo.source) {%]
[%if (atributo.target.tipoatributo.value == 4) {%]
        CREATE ([%=atributo.target.nome%]:[%=elemento.nome%] {[%}%]
        [%if (atributo.target.tipoatributo.value == 3){%]
        [%=atributo.target.nome%]: [’valor’, ’valor’], [%}
        else {%][%=atributo.target.nome%]: ’valor’, [%}%] [%}%] [%}%]})
[%}%] [%}%]
[* Fim das entidades*]
//Código Cypher de criação dos relacionamentos
[%for (elemento in nosqlesquema.elementos) {%]
[%if ((elemento.type.name == “Entidade” or elemento.type.name ==
        “EntidadeNaoEstruturada” or elemento.type.name ==
        “EntidadeSemiEstruturada” or elemento.type.name == “EntidadeAgregada”)) {%]
[%var a = 0;
var entidadeid = “”;%]
[%for (relacionamentolink in nosqlesquema.relacionamentoslink) {%]
[%if (elemento == relacionamentolink.target){a = a+1;%]
[%for (atributo in nosqlesquema.atributosentidadelink) {%]
[%if (relacionamentolink.source == atributo.source)  {%]
[%if (atributo.target.tipoatributo.value == 4) {%]
[%if (a==1){%]
CREATE ([%=atributo.source.nome%].[%=atributo.target.nome%])
        -[:[%=elemento.nome%]{[%}if (a==2){%][%for
        (atributo in nosqlesquema.atributosrelacionamentolink)
        {%][%if (elemento == atributo.source) {%][%=atributo.target.nome%]
        : ’valor’,[%}%][%}%]}]->([%=atributo.source.nome%]
        .[%=atributo.target.nome%])
[%}%][%}%][%}%][%}%][%}%][%}%][%}%][%}%][%}%]
```

[%}%] [%}%] [%}%]